Aspect Based Sentiment Analysis (ABSA) and Recommendation System on TripAdvisor Restaurant Reviews - RestoReview

FIT3162 Final Report
Group MCS8

Tan Won Lun[1]
Technical Lead

Wong Chee Hao[1]
Technical Lead

Yeoh Li Jun[1]
Project Manager

Ngeoh Khai Vin[1]
Quality Assurance

Lim Wern Han
Supervisor

June 6, 2024

10870 words

[1]Listing order is random and all members have equal contribution

Abstract

The proliferation of user-generated content (UGC) has presented unique opportunities to leverage these vast data sources for innovative solutions to existing challenges. In the context of food reviews, the overwhelming volume often leads to a reliance on numerical ratings, which can be unreliable and insufficient for comprehensive decision-making. Both consumers and business owners face difficulties: users must sift through numerous reviews to find new dining options, while business owners struggle to extract meaningful insights from extensive feedback.

This report details the implementation of an Aspect-Based Sentiment Analysis (ABSA) model designed to analyze UGC from TripAdvisor, providing a more nuanced understanding of restaurant performance. By evaluating reviews across various aspects such as food quality, ambience, and service, we generate time series data that illustrate how restaurants perform over time in these categories. Additionally, our solution includes an algorithm that recommends similar restaurants based on aspect performance and cuisine type. This implemented solution aims to enhance decision-making for both consumers seeking dining options and business owners aiming to improve their services.

Contents

Abstract 2

1 Introduction 6

2 Project Background 7

2.1 Background . . . 7

2.1.1 Introduction . . . 7

2.1.2 Rationale for The Project . . . 7

2.2 Literature Review . . . 8

2.2.1 Aspect-Based Sentiment Analysis . . . 8

2.2.2 Non-neural Network Approaches to Aspect-Based Sentiment Analysis . . . 8

2.2.3 Neural Network Approaches to Aspect-Based Sentiment Analysis . . . 9

2.2.4 Existing Frameworks . . . 10

2.2.5 Evaluation Metrics for Classifier . . . 11

2.2.6 Recommendation System . . . 11

2.2.7 Summary . . . 13

3 Methodology 14

3.1 Design . . . 14

3.1.1 Data Collection . . . 14

3.1.2 Data Pre-Processing . . . 15

3.1.2.1 Data Sampling . . . 15

3.1.2.2 Data Annotation . . . 16

3.1.3 Model Framework Evaluation . . . 16

3.1.4 Hyperparameter Tuning . . . 17

3.1.5 Evaluation on Performance . . . 17

3.1.6 Recommendation System Design . . . 18
3.1.7 Data Visualisation . . . 19
3.2 Deviations . . . 19
3.2.1 Data Collection . . . 19
3.2.2 Recommendation System Approach . . . 19
3.2.3 Inclusion of Restaurant Budget Information . . . 20
3.2.4 User Interface (UI) Design . . . 20
3.2.5 Conclusion . . . 20
3.3 How was the design implemented . . . 21

4 Software Deliverables 22
4.1 Summary of software deliverables . . . 22
4.1.1 Description of Usage . . . 22
4.2 Software Qualities . . . 23
4.2.1 Robustness . . . 23
4.2.2 Security . . . 23
4.2.3 Usability . . . 23
4.2.4 Scalability . . . 24
4.2.5 Documentation and Maintainability . . . 24
4.3 Sample Source Code . . . 24

5 Outcomes 25
5.1 What has been implemented . . . 25
5.2 Results Achieved/Product Delivered . . . 25
5.2.1 Model Performance . . . 25
5.2.2 Web Application GUI Overall User Satisfaction Rate . . . 26
5.3 How are requirements met . . . 26
5.4 Discussion of All Results . . . 28
5.4.1 Validation of Correctness of ABSA Model . . . 28
5.4.2 Validation of Correctness of ABSA Model II . . . 29
5.4.3 Validation of usefulness Web Application Prototype . . . 30
5.5 Limitations of Project Outcomes . . . 30
5.5.1 Limitations of RestoReview (Web application) . . . 30

5.5.2 Limitations of ABSA Model . . . 31
5.5.3 Limitations of TripAdvisor Dataset . . . 31
5.6 Justification of Decisions Made . . . 32
5.6.1 Obtaining our dataset . . . 32
5.6.2 Usage of T5 Model . . . 33
5.6.3 Model Training with 16 Epochs . . . 33
5.7 Future Work . . . 33

6 Critical discussion on the Software Project 35
6.1 Programming Language . . . 35
6.2 Dataset . . . 35
6.3 User Interface . . . 35
6.4 Recommendation System . . . 36
6.5 Model Evaluation . . . 37
6.6 Risk Management . . . 37
6.7 Summary . . . 38

7 Conclusion 39

A Data Flow Diagram 40

B Heuristic Evaluation 41

C User Interface 43

D Sample Source Code 46

E Software Modules Used 50

F Others 52

Bibliography 54

Chapter 1

Introduction

The advancement of technology continues to reshape industries, improve everyday life, and provide an outlet for users to voice their opinions on certain things. Particularly, User-generated content (UGC) significantly influences consumer decisions and enhances business interactions. It provides a diverse collection of insights and opinions contributed by individuals on various online platforms [33]. By examining the collective knowledge shared on social media and review websites, the performance of different dining establishments can be evaluated. In this context, our project, Aspect-Based Sentiment Analysis (ABSA) and Recommendation System on TripAdvisor Restaurant Reviews aims to address the challenges within the restaurant industry. Restaurant reviews can be lengthy and do not directly highlight the overall sentiment of the restaurant unless the user goes through each individual review.

The primary motivation behind this project is to leverage modern technologies to solve real-world problems that significantly impact consumers searching for good, reliable restaurants. The project seeks to improve the user's decision-making process through the implementation of advanced software techniques. We attempt to solve this problem by developing a robust and user-friendly web application that was brought to fruition by our model.

Our model leverages aspect-based sentiment analysis to precisely categorize different elements within individual reviews and evaluate their sentiment [29]. This approach enables users to comprehensively evaluate restaurants on multiple dimensions, such as pricing, location, taste, and more, instead of relying on the traditional simplistic numerical rating out of five. Furthermore, restaurant owners could also benefit from this solution by enabling them to recognize the sentiment of their restaurant down to each individual aspect without having to manually go through all the reviews themselves.

Our project's main aims are as follows:

1. **PA1:** To implement the relevant approach that performs aspect-based sentiment analysis.
2. **PA2:** To design and develop a software solution that meets the specific needs of users.
3. **PA3:** To ensure the software is scalable, secure, and maintainable, adhering to industry best practices.
4. **PA4:** To critically evaluate the software's performance and usability, providing a foundation for future enhancements and developments.

This report is structured to provide a comprehensive overview of our project, from its inception to the final outcomes. The following sections will detail the project's background (Chapter 2), including a thorough literature review, and the methodologies (Chapter 3) employed in the design and development phases. We will present the outcomes (Chapter 5), discussing the results achieved (Section 5.4) and how the software meets the specified requirements (Section 5.3). Furthermore, a critical discussion (Chapter 6) will analyze the project execution, reflecting on the deviations from the initial plan and lessons learned.

Chapter 2

Project Background

2.1 Background

2.1.1 Introduction

In the restaurant industry, customer feedback is crucial for assessing quality and service. Traditional numerical ratings on platforms like Yelp and TripAdvisor are commonly used by consumers to make dining decisions. However, these ratings lack the detailed insights necessary to evaluate specific aspects of a restaurant, such as food quality, service, ambiance, and cleanliness[5]. This project aims to overcome the limitations of traditional rating systems by implementing aspect-based sentiment analysis (ABSA). ABSA categorizes and analyzes sentiments related to specific aspects of the dining experience, providing a more comprehensive understanding of customer feedback. This approach not only aids consumers in making informed decisions but also helps restaurant owners gain actionable insights to improve their services.

2.1.2 Rationale for The Project

The impetus for our team undertaking this project stems from the significant challenges users face in analyzing restaurants based on mere numerical ratings. Our initiative aims to redefine how restaurant quality is evaluated and to provide both users and restaurant owners with more meaningful data.

Challenges of Numerical Ratings in Restaurant Analysis Traditional numerical rating systems fall short in providing a comprehensive picture of a restaurant's quality and service. These ratings do not offer insights into specific aspects such as food quality, service, ambiance, and cleanliness, making it difficult for users to make informed decisions. Research shows that numerical ratings can be misleading due to their lack of context and granularity[5].

The Burden of Review Overload Reading through all the reviews to evaluate a restaurant based on individual preferences is an arduous and time-consuming task. Each review might focus on different aspects, making it difficult to collate and compare specific attributes of interest. This disjointed information impedes users from effectively leveraging reviews to match their personal preferences and requirements[6].

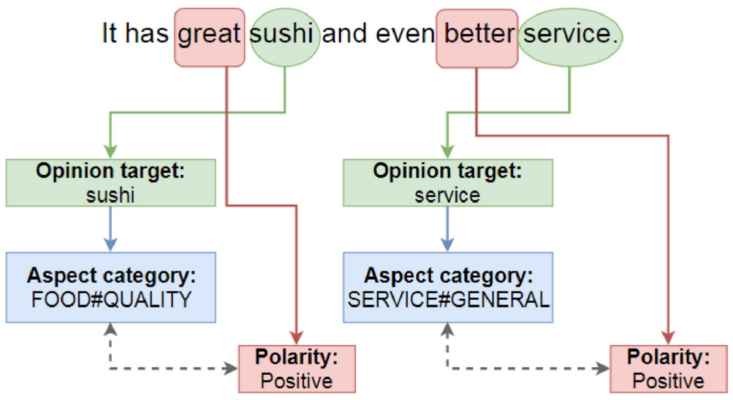
Implementing Sentiment Analysis Our project addresses these gaps by leveraging sentiment analysis techniques to analyze restaurant reviews. By focusing on aspect-based sentiment analysis, we aim to provide users with a nuanced understanding of restaurant quality based on their specific preferences.

This approach not only simplifies the decision-making process for users but also equips restaurant owners with detailed, aspect-specific feedback, enabling them to make informed improvements.

Goals and Expectations This initiative is driven by the need to enhance the dining experience for users and to support restaurant owners in their quest for excellence. By promoting a deeper understanding of customer feedback and preferences, our project aspires to foster a more informed, efficient, and satisfying restaurant selection process for users while empowering restaurant owners with the insights needed to thrive in a competitive market.

2.2 Literature Review

2.2.1 Aspect-Based Sentiment Analysis



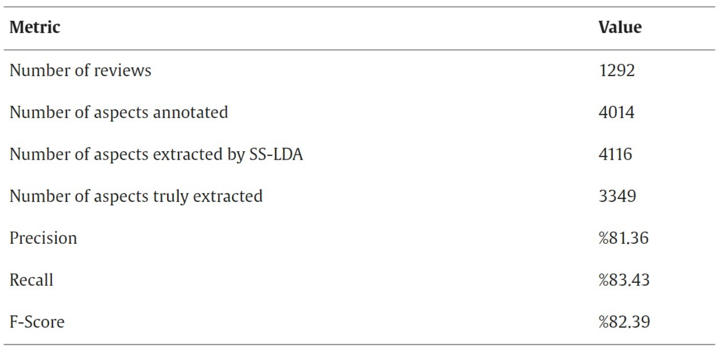
It has great sushi and even better service.
Opinion target: sushi
Aspect category: FOOD#QUALITY
Polarity: Positive
Opinion target: service
Aspect category: SERVICE#GENERAL
Polarity: Positive

Figure 2.1: The 4 phases of Aspect Based Sentiment Analysis (ABSA): Aspect Term Extraction (ATE), Opinion Extraction, Sentiment Classification,Aspect Categorization [17]

Aspect-Based Sentiment Analysis (ABSA) delivers a nuanced method for sentiment analysis by homing in on specific attributes within a context. ABSA involves four critical phases: aspect term extraction (ATE), where it identifies key attributes like ”sushi”; opinion extraction, pinpointing descriptive terms such as ”great”; sentiment classification (SC), which assesses the sentiment tone; and aspect categorization, which aligns the sentiment with broad categories such as food quality. By focusing on these phases, ABSA can accurately determine sentiments related to distinct elements or features, like those expressed in the statement, ”It has great sushi and even better service,” where it recognizes positive sentiments towards both ”sushi” and ”service” [50].

2.2.2 Non-neural Network Approaches to Aspect-Based Sentiment Analysis

Latent Dirichlet Allocation (LDA) Latent Dirichlet Allocation (LDA) is effective for detecting hidden topics in large text collections by analyzing aspects and sentiments without labeled data, though it sometimes struggles with topic clarity and requires careful setting of topic counts [8], [23], [35].



Metric	Value
Number of reviews	1292
Number of aspects annotated	4014
Number of aspects extracted by SS-LDA	4116
Number of aspects truly extracted	3349
Precision	%81.36
Recall	%83.43
F-Score	%82.39

Figure 2.2: Success level of SS-LDA in extracting product aspects

Sentence Segment LDA (SS-LDA) improves accuracy with short texts by addressing data sparsity and relies on sentiment dictionaries for better aspect extraction [35]. While experimental findings suggest that SS-LDA is reasonably effective in extracting product aspects as shown in 2.2, it's important to note that this approach relies on a sentiment dictionary.

Naive Bayes Naive Bayes excels in high-dimensional data processing, offering fast and computationally efficient classifications, but its assumption of independent features limits its effectiveness in complex text data [32], [31], [48].

Support Vector Machines (SVM) Support Vector Machines (SVM) manage high-dimensional spaces well and resist overfitting, although they require significant computational resources and careful kernel selection [24], [15], [48].

Conditional Random Field (CRF) Conditional Random Fields (CRF) show strong domain adaptability by integrating diverse features, though they face challenges with unrecognized aspect terms [13].

2.2.3 Neural Network Approaches to Aspect-Based Sentiment Analysis

Deep Neural Networks (DNN) Deep Neural Networks (DNNs) are highly effective in ABSA due to their ability to learn hierarchical features from raw data, capturing intricate language semantics necessary for understanding complex context in texts [28], [40]. However, they require large datasets and extensive computational resources, which can be limiting in resource-constrained environments [43].

Convolutional Neural Networks (CNN) Convolutional Neural Networks (CNNs) in ABSA utilize a cascaded model with aspect mappers and a sentiment classifier, reducing the need for extensive feature engineering and improving processing time compared to traditional methods like SVM [19].

Hierarchical Bidirectional LSTM Hierarchical Bidirectional LSTM models address interdependencies within texts, offering superior performance by utilizing sentence-level context for better sentiment and aspect recognition in ABSA tasks [38].

Selective Adversarial Learning LSTM This approach uses unsupervised domain adaptation to enhance ABSA by leveraging knowledge from other domains, which helps in scenarios with scarce data [30].

Transformers Transformers revolutionize ABSA with their self-attention mechanism, effectively managing word significance and intricate relationships in texts. Despite their high performance, they require substantial computational efforts and are prone to overfitting [45], [49].

BERT and Adversarial Training The use of BERT and adversarial training methods in ABSA helps in fine-tuning contextual word representations and enhancing the accuracy of aspect extraction and sentiment classification tasks [25]. However, the study pointed out that the potential impact of adversarial training in other ABSA tasks, specifically Aspect Category Detection and Aspect Category Polarity, requires further in-depth investigation.

Instruction-Based Learning Paradigms Instruction-based learning paradigms, such as InstructABSA, mark a significant advancement in Aspect-Based Sentiment Analysis (ABSA). InstructABSA enhances efficiency and effectiveness by using explicit instructions to guide the model's learning process.By leveraging pre-trained models like T5 and fine-tuning them with task-specific instructions, it improves performance in various subtasks, including aspect term extraction and sentiment classification.Research by Scaria et al. (2023) [39] demonstrates the effectiveness of instruction-based learning in improving ABSA performance through the use of explicit instructions tailored to specific tasks [12]. A study by Varia et al. (2023) [44] demonstrated significant improvements in both efficiency and accuracy when comparing instruction-based models to traditional methods, highlighting the potential of this approach in practical applications. .

Transfer Learning in ABSA Transfer learning involves using pre-trained models on large datasets and fine-tuning them for specific tasks. In ABSA, models like T5 and BERT have been effectively utilized to transfer learned knowledge to sentiment analysis tasks, improving accuracy and reducing the need for large amounts of task-specific data. Relevant studies include research by Raffel et al. (2023) [37], exploring the use of the T5 model and showcasing its versatility and effectiveness in various NLP tasks, including ABSA, and studies on BERT and its variants highlighting their success in enhancing sentiment classification and aspect extraction through transfer learning.

Comparative Analysis of Pre-trained Models Pre-trained models offer varying strengths in ABSA tasks. BERT is known for its strong performance in sentence-level understanding and bidirectional context representation, while T5 utilizes a text-to-text framework, offering greater flexibility in handling diverse NLP tasks. Other models like GPT, XLNet, and RoBERTa each contribute unique features that enhance their performance in ABSA. Comparative studies by Diogo Cortiz (2021)[16] indicate that while BERT excels in specific aspects, T5's text-to-text approach provides broader applicability and superior performance in multi-task settings. Additionally, models like RoBERTa and XLNet show improvements in certain areas of sentiment analysis, further enriching the landscape of pre-trained models used in ABSA.

2.2.4 Existing Frameworks

In recent years, there has been a surge in open-source models for aspect-based sentiment classification (ASC) and aspect term extraction and sentiment classification (ATESC). However, many of these models lack predictive functionalities and have become outdated. Notable predecessors like ABSA-PyTorch [41] and Aspect-based Sentiment Analysis [2] support only specific tasks or offer a limited set of models. PyABSA[46]stands out as a comprehensive, actively maintained framework with instant inference capabilities for multiple ABSA tasks and support for various languages, making it a more versatile alternative.

2.2.5 Evaluation Metrics for Classifier

In the study by Juri Opitz[34], the evaluation of classifiers using micro and macro metrics is scrutinized, particularly under varying class distributions and biases. The paper distinguishes between micro metrics, which directly correspond to class occurrences in a sample, and macro metrics, which provide a broader evaluation by averaging performance across all classes, thereby remaining unaffected by class frequency variations.

$$\text{Precision}_{\text{macro}} = \frac{\text{Precision}_{\text{Class A}} + \text{Precision}_{\text{Class B}} + \ldots + \text{Precision}_{\text{Class N}}}{N} \tag{2.1}$$

$$\text{Recall}_{\text{macro}} = \frac{\text{Recall}_{\text{Class A}} + \text{Recall}_{\text{Class B}} + \ldots + \text{Recall}_{\text{Class N}}}{N} \tag{2.2}$$

Macro Evaluation Macro metrics, such as Macro F1, Macro Precision, and Macro Recall, are emphasized for their ability to provide a balanced view of classifier performance across different classes, making them particularly useful in settings with imbalanced data. So, the macro-average gives equal weight to each class, regardless of the number of instances.

$$\text{Precision}_{\text{micro}} = \frac{TP_A + TP_B + \ldots + TP_N}{TP_A + FP_A + TP_B + FP_B + \ldots + TP_N + FP_N} \tag{2.3}$$

$$\text{Recall}_{\text{micro}} = \frac{TP_A + TP_B + \ldots + TP_N}{TP_A + FN_A + TP_B + FN_B + \ldots + TP_N + FN_N} \tag{2.4}$$

Micro Evaluation Micro-averaging, on the other hand, aggregates the counts of true positives, false positives, and false negatives across all classes and then calculates the performance metric based on the total counts. So, the micro-average gives equal weight to each instance, regardless of the class label and the number of cases in the class.

2.2.6 Recommendation System

Information overload can overwhelm individuals when they search for the ideal product online. To address this issue, a decision support system is essential . In the e-commerce sector, recommender systems are crucial for aiding customers in choosing products[9]. Leveraging various algorithms and data processing techniques, these systems analyze user behavior to provide suggestions tailored to individual preferences[5]. Their ability to personalize user experiences in the digital realm has made them indispensable tools for online platforms. For instance, Spotify's Discover Weekly uses collaborative filtering, by comparing the listening habits of millions of users to recommend new songs and artists to them[10]. In general, recommendation systems are divided into 2 categories: i) Collaborative Filtering, and ii) Content Filtering.

Collaborative Filtering Collaborative filtering, a key technique in recommendation systems, comes in two main types: item-based and user-based filtering, offering personalized suggestions based on the preferences of other users with similar tastes. In item-based filtering, the system suggests items similar to those a user has shown interest in by comparing item similarities. In user-based filtering, the system recommends items that like-minded users have enjoyed by identifying users with similar preferences. Filipsson explored both approaches using an adjusted cosine similarity function to measure similarities between users and items. This method considers both shared and differing data, resulting in more precise recommendations. The study found that user-based filtering generally outperforms item-based filtering in practical scenarios, especially with larger datasets. However, it also highlights the challenge of needing substantial user information and increased computational demands for new applications lacking sufficient user data[11].

Cosine Similarity Khatter examined various similarity functions including the dice coefficient, correlation-based measures, Euclidean distance, Pearson correlation coefficient, and Cosine similarity [26]. The study found that Cosine similarity was the optimal choice for their system due to its shorter execution time compared to adjusted-based and correlation-based similarities.

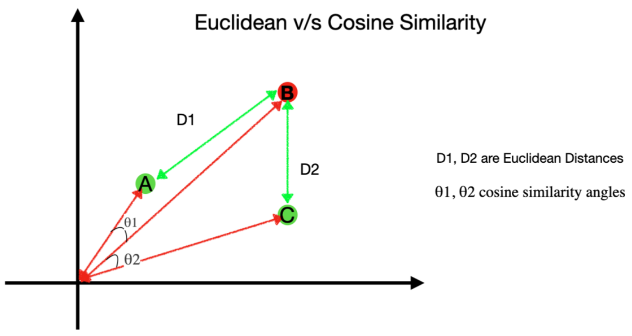

Euclidean v/s Cosine Similarity
D1
D2
A
B
C
θ1
θ2
D1, D2 are Euclidean Distances
θ1, θ2 cosine similarity angles

Figure 2.3: Cosine similarity compared to Euclidean distance

Furthermore, it was noted that Cosine similarity is capable of showing a smaller angle between two similar objects, regardless of their spatial distance as measured by Euclidean standards, as illustrated in Figure 2.3. This feature renders it particularly effective for use in recommendation systems.

Content-based Filtering Colucci's paper discussed and tested content-based filtering which relies on item metadata, such as a restaurant's name, cuisine type, location, and review rating, to assess similarity [14]. To illustrate this concept, consider the following example: if restaurant A and restaurant B are both Thai restaurants, they are likely to share some similarities. Furthermore, if Restaurant C is also a Thai restaurant and is located near Restaurant A, then Restaurant C is more similar to Restaurant A than Restaurant B. The advantage of a pure content filtering approach is that it operates without requiring prior user input. This makes it particularly valuable in a new system with limited or no user-generated data, where traditional collaborative filtering methods may not be feasible.

Colucci's study was able to prove the possibility of evaluating different item-item similarity algorithms or models with a user study [14]. Although content-based filtering performed lower compared to collaborative filtering, it proved advantageous for situations without the option of user data.

Vectorizing Attributes Vectorizing attributes in Python enhances the efficiency of code by eliminating the need for loops. This method is crucial for converting data from lists into matrices, facilitating faster processing in recommendation systems. Particularly important is the transformation of CSV data into matrix form, which is essential for machine learning applications[36].

Jaccard Similarity The Jaccard Similarity, also known as the Jaccard Index, is a statistic used to measure the similarity between sample sets. It is defined as the size of the intersection divided by the size of the union of the sample sets.

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{2.5}$$

Jaccard Similarity can then be defined according to the following formula.

$$d(A, B) = 1 - \text{sim}(A, B) = 1 - \frac{|A \cap B|}{|A \cup B|} \tag{2.6}$$

Getting closer to 1 value from Jaccard Similarity indicates similarity between two sets [7].

In the study by Sukestiyarno et al.[42], it is demonstrated that Jaccard Similarity, when applied alongside Cosine Similarity, effectively personalizes e-learning content recommendations. Notably, Jaccard Similarity is highlighted for its capability to discern user-item relationships purely based on shared attributes, making it ideal for environments with limited user interaction data. The study suggests that refining attribute inputs, like incorporating diverse content features, can enhance the recommendation system's accuracy, thus reducing mean absolute error rates in predictive models. This approach can significantly improve recommendation systems by ensuring more precise and user-specific outputs, which is crucial for platforms like e-learning where personalization directly impacts user engagement and satisfaction.

2.2.7 Summary

Aspect-Based Sentiment Analysis Advancements in aspect-based sentiment analysis (ABSA) have led to the development of various models, with PyABSA standing out for its comprehensive functionality and support for multiple languages. It surpasses older models like ABSA-PyTorch by offering superior performance and ease of use.

Evaluation Metrics for Classifiers Classifier performance is evaluated using macro and micro metrics. Macro metrics, such as Macro F1, Macro Precision, and Macro Recall, provide a balanced view across different classes, essential for imbalanced data. Micro metrics aggregate true positives, false positives, and false negatives across all classes, giving equal weight to each instance.

Recommendation Systems Recommendation systems are essential for aiding online product selection, using techniques like collaborative and content-based filtering. Collaborative filtering, particularly user-based, is effective with sufficient user data, while content-based filtering relies on item metadata and is useful in data-sparse environments. Combining Jaccard Similarity with Cosine Similarity enhances recommendation accuracy and personalization.

Chapter 3

Methodology

This section of the report provides an overview of the project's methodology, including the deviations from the initial proposed design in the last semester and details the implementation of the design for each stage. A.1 shows the Level 0 data flow diagram that describes how we process our data to obtain our final model.

3.1 Design

The methodology employed in our project is encapsulated in the pipeline diagram presented below. This diagram serves as a visual road map, illustrating the processes we utilized. It details the journey from initial user input through to the final output, showcasing how our team designed and implemented the recommendation system.

3.1.1 Data Collection

During the data collection stage, we employed Selenium web scraping techniques [27], utilizing a web crawler to extract our desired dataset.

Table 3.1: Dataset Details

Category	**Details**
Restaurant Details	Restaurant Name
	Total Number of Reviews
	Average Rating
	Cuisine Type
	Corresponding TripAdvisor URL
Review Data	Author Name
	Review Title
	Review Text
	Given Rating
	Review Date
	Restaurant Name

Our data collection efforts focused on two distinct locations: **Kuala Lumpur** and **Rome**. Each location dataset comprised approximately 500 restaurants and 40,000 reviews. To ensure uniformity and mitigate potential disparities in review counts among restaurants, we capped the maximum number of reviews collected for each restaurant at 105. This approach aimed to maintain consistency and reliability across our dataset while providing a comprehensive representation of restaurant ratings and user feedback.

3.1.2 Data Pre-Processing

Elimination of Non-Text Elements Emojis and other non-text elements were removed from the dataset to streamline the text analysis process and maintain consistency.

Removal of Non-English Text Any non-English text present in the dataset was identified and removed. This ensured that the analysis focused solely on English-language content, avoiding any potential complications arising from language differences.

Conversion of Dates to Datetime Format Dates present in the dataset were converted into datetime format (F.2). This conversion facilitated easier manipulation and analysis of temporal data.

Removal of Null Rows Rows containing null values were identified and subsequently removed from the dataset. This step was essential to prevent any potential bias or inaccuracies in the analysis caused by missing data.

Handling Duplicates Duplicate entries within the dataset were identified and either removed or merged as necessary. This helped maintain data integrity and accuracy.

Algorithm 1 Data Pre-processing

```
0: procedure DataCleaning(data)
0:     1: Remove non-text elements (e.g., emojis)
0:     2: Remove non-English text entries
0:     3: Convert dates to datetime format
0:     4: Remove rows with null values
0:     5: Identify and handle duplicates
0:     6: return Cleaned data
0: end procedure=0
```

By systematically applying these data-cleaning procedures, the dataset was prepared for further analysis and interpretation, ensuring the reliability and validity of any insights derived from it.

3.1.2.1 Data Sampling

Before fine-tuning the Aspect-Based Sentiment Analysis (ABSA) model using our proprietary dataset, we initiated the process with a crucial step: data sampling and data annotation. This phase involved selecting a representative subset of our dataset for annotation and model training.

To ensure the representativeness and diversity of our dataset for subsequent model fine-tuning, we implemented a stratified sampling approach based on the ratings of the reviews. This method allows us to maintain the proportional distribution of ratings while also ensuring diversity across different restaurants. The detailed sampling procedure is depicted in Algorithm 2.

Following the sampling procedure, we shuffled the sampled data to mix ratings, enhancing the dataset's randomness and reducing any potential biases.

The sampled dataset comprised 2500 reviews and over 7000 aspects in total, with 80 percent allocated for the training set and the remaining 20 percent for testing. Specifically, each location contributed 1250 reviews, ensuring balance across the dataset proportionally.

Algorithm 2 Stratified Sampling Based on Ratings

```
0: procedure StratifiedSampling(data, n_samples)
0:     1: $rating\_counts \leftarrow$ value counts of data['Rating'] normalized by $n\_samples$
0:     2: $rating\_counts \leftarrow$ round and convert to int($rating\_counts$)
0:     if sum($rating\_counts$) $\neq n\_samples$ then
0:         3: $max\_rating \leftarrow$ index of max($rating\_counts$)
0:         4: $rating\_counts[max\_rating] \leftarrow rating\_counts[max\_rating] + (n\_samples - \text{sum}(rating\_counts))$
0:     end if
0:     5: $sampled\_data \leftarrow$ empty DataFrame
0:     for $rating, count$ in $rating\_counts.items()$ do
0:         6: $sampled\_reviews \leftarrow$ sample $n = count$ from data where data['Rating'] == rating with random_state=42
0:         7: concatenate $sampled\_data$ with $sampled\_reviews$
0:     end for
0:     8: $sampled\_data \leftarrow$ shuffle $sampled\_data$ with random_state=42 and reset index
0:     9: return $sampled\_data$
0: end procedure=0
```

3.1.2.2 Data Annotation

Since we are dealing with real data, we engaged our peers to assist with data annotation. Annotators are required to specify aspect terms, opinion terms, sentiment polarity and aspect category for each aspect in a review. More specifically, annotators are told to take note of NULL aspect terms, such as the sentence ”Yummy!” where the aspect term is NULL but the opinion term, sentiment polarity and aspect category are ”Yummy”, positive and FOOD#QUALITY respectively.

3.1.3 Model Framework Evaluation

Our team has tested and compared three aspect-based sentiment analysis (ABSA) models: PyABSA, BERT ABSA, and ABSA PyTorch as shown in Table F.3. Each model was evaluated using its own dataset. Ultimately, **PyABSA** stood out due to its ”quadruple extraction” capability.

Why ABSA PyTorch and BERT ABSA were not chosen

- **ABSA PyTorch:** While it generates sentiment polarity, it requires pre-defined aspect categories, limiting its ability to automatically identify them in text.
- **BERT ABSA:** It identifies aspect terms with sentiment but doesn't provide the broader aspect category.

Why PyABSA was chosen PyABSA goes beyond just aspect terms and sentiment analysis. Its ”quadruple extraction” delivers the following outputs, which are useful for extracting all the aspect categories of a restaurant review:

- **Aspect Term:** The specific aspect mentioned in the review.
- **Sentiment Polarity:** Positive and negative.
- **Opinion Term:** The words expressing the sentiment.
- **Aspect Category:** The broader category the aspect belongs to, such as SUPPORT#GENERAL, SERVICE#GENERAL, and more.

3.1.4 Hyperparameter Tuning

Hyperparameter tuning plays a vital role in the training process. Adjustments are made to key parameters, such as the total number of training epochs, the interval between logging updates, and the accumulation steps for output tensors. To better monitor progress, we use several callbacks, including early stopping, logging, and checkpoints. These fine-tuning iterations are crucial for optimizing the model's performance and ensuring it generalizes well.

During this stage, our team continuously fine-tuned the model using the annotated dataset. The primary objective of aspect extraction was to identify the most important keywords in user reviews for the restaurant. Sentiment analysis was conducted to determine whether the reviews were positive, negative, or neutral. Additionally, opinion extraction was performed. For each review, we prepared a set of predefined aspect categories to classify which aspect category the reviews fell into.

However, our team discovered that the original model within the PyABSA framework was flawed. Consequently, we made the decision to conduct an evaluation of the model's performance. This evaluation encompassed four sub-tasks: aspect term extraction, polarity determination, aspect category classification, and opinion term identification. To ensure a thorough assessment of our model, we underwent multiple iterations across all sub-tasks. We adhered to the most rigorous evaluation criteria, which will be briefly explained in the following section, to guarantee accurate evaluation. For instance, any inaccuracies in aspect term extraction could lead to failures in correctly handling the other sub-tasks.

Parameters	**Value**
output_dir	model_out_path
evaluation_strategy	epoch
save_strategy	epoch
learning_rate	5e5
per_device_train_batch_size	4
per_device_eval_batch_size	16
num_train_epochs	16
weight_decay	0.01
warmup_ratio	0.1
load_best_model_at_end	True
push_to_hub	False
eval_accumulation_steps	1
predict_with_generate	True
logging_steps	1000000000
use_mps_device	False
fp16	False

Table 3.2: Table of training arguments for Hyperparameter Tuning

3.1.5 Evaluation on Performance

The evaluation metrics are categorized into micro and macro metrics. Macro metric involves computing evaluation metrics for each sub-task and then calculating the arithmetic mean across all sub-tasks. This method assigns equal importance to every sub-task, regardless of the number of instances it contains. Conversely, micro metrics consolidate the counts of true positives, false positives, and false negatives across all sub-task. It then computes the evaluation metric based on these total counts.

Precision: The ratio of correctly predicted positive instances to the total predicted positive instances.

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive} \tag{3.1}$$

Recall (Sensitivity): The ratio of correctly predicted positive instances to all relevant instances in the actual class.

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \tag{3.2}$$

F1-Score: The harmonic mean of the precision and recall of a model.

$$F1 - Score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{3.3}$$

As shown in F.5 and F.6 in the Appendix, there is the code for the micro and macro metric evaluation respectively on aspect term.

3.1.6 Recommendation System Design

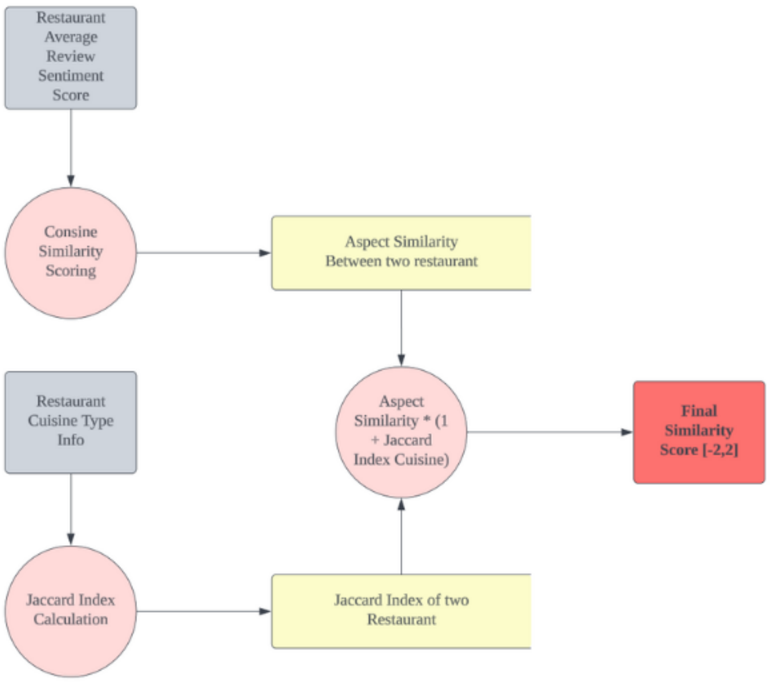

Restaurant Average Review Sentiment Score
Consine Similarity Scoring
Aspect Similarity Between two restaurant
Restaurant Cuisine Type Info
Aspect Similarity * (1 + Jaccard Index Cuisine)
Final Similarity Score [-2,2]
Jaccard Index Calculation
Jaccard Index of two Restaurant

Figure 3.1: Level 1 Data flow diagram of recommendation system design methodology

Cosine similarity is a measure used to determine how similar two vectors are, regardless of their magnitude. It is particularly useful in various fields for comparing datasets. Thus, we applied it to find the aspect similarity between two restaurants with the input of restaurant average review and sentiment score.

The Jaccard Index Calculation is statistically used for measuring the similarity and diversity of sample sets. It is defined as the size of the intersection divided by the size of the union of the sample sets. The Jaccard Index is particularly useful for comparing two different sets of data and is often used in recommendation systems. Therefore, our team calculated the Jaccard Index Cuisine for two different restaurants from its restaurant cuisine type info.

Lastly, we created a formula for the final similarity score within the range from 2 to -2:

$$finalsimilarityscore = aspectsimilarity * (1 + JaccardIndexCuisine) \tag{3.4}$$

3.1.7 Data Visualisation

During this stage, which is to output our findings, our team has crafted an innovative web application called RestoReview. This dynamic platform elegantly presents key metrics and detailed insights for each restaurant, fostering seamless exploration and interaction for users. For instance, the cleaned data, comprising review texts and restaurant details, is visualized in a consistently arranged format. This facilitates users to conveniently access all desired information.

We also visualized the information gained from the model. Specifically, we grouped the sentiment of each aspect category so that users could easily see which aspects are performing better or worse as shown in C.4. This approach provides a clear overview of the strengths and weaknesses of each restaurant across various parameters, aiding users in making more informed decisions.

Our designed time-series graphs of visualising the performance of each aspect category over time as shown in C.5 offer enhanced clarity by visualizing the performance of each aspect category over time. This feature was added in later iterations to provide more context to users, enabling them to understand the evolving landscape of each restaurant. The time labels on the graphs are set quarterly to synchronize with the data, ensuring a more accurate representation without presenting outdated information especially when it comes to restaurants that have changed. This addition allows users to make more informed dining decisions based on comprehensive and up-to-date insights.

3.2 Deviations

During the implementation of our project, we adhered to the overall methodology outlined in FIT3161 last semester but encountered several deviations due to various constraints. These deviations, although unplanned, were necessary to overcome obstacles and improve the final outcome of our web application.

3.2.1 Data Collection

Initially, our plan for data collection involved scraping data from TripAdvisor using its APIs. However, we quickly realized that this approach presented significant challenges.

Challenges with TripAdvisor APIs The cost associated with accessing the APIs was prohibitively high, and the lack of adequate documentation made it difficult to use effectively. Faced with these obstacles, we consulted our supervisor and decided to pivot from our original plan.

Switch to Selenium Web Scraping Instead of using the TripAdvisor APIs, we opted for Selenium web scraping, implementing a web crawler to retrieve our desired dataset. This approach proved to be more feasible and aligned better with our resources and technical capabilities.

3.2.2 Recommendation System Approach

Another major deviation in our project involved the approach to providing similar restaurant recommendations to users.

Initial Clustering Model Plan Initially, we planned to utilize a clustering model to group similar restaurants into clusters, thereby enabling our web app to recommend similar restaurants to users.

Transition to Cosine Similarity As we progressed to the implementation stage and conducted further research, we discovered that using cosine similarity scores was a more effective method. Calculating similarity scores is simpler and more efficient than training clustering models. Additionally, this approach allowed us to customize the similarity score based on different factors relevant to the selected restaurant, such as cuisine type. This realization led us to abandon the clustering model in favour of cosine similarity, which ultimately enhanced the functionality and performance of our recommendation system.

3.2.3 Inclusion of Restaurant Budget Information

The recommender system also saw another deviation related to the inclusion of restaurant budget information.

Initial Plan for Budget Information Initially, we intended to factor in the budget of restaurants in our recommendation algorithm.

Data Availability Issues However, we encountered significant gaps in the availability of this data across various restaurant pages. To ensure accurate and reliable data visualization, we decided not to include budget information in our recommendation algorithm. This decision allowed us to maintain the integrity of our data and provide users with a more consistent and dependable recommendation experience.

3.2.4 User Interface (UI) Design

The final major deviation pertained to the user interface (UI) of our web application.

Initial UI Design Our team had an initial UI design plan that met all the requirements. However, as we moved into the implementation stage, it became apparent that adjustments were necessary to enhance the user experience.

Key Adjustments One key adjustment was the decision to use different colours for key metrics, with the lightest colour representing the lowest value. This change improved visual clarity and made the metrics more intuitive for users. Additionally, we added a search bar to facilitate easier navigation, allowing users to find their desired restaurants quickly. We also enabled users to change locations before searching for restaurants, which helped reduce the display of unnecessary information and made the web application more user-friendly.

3.2.5 Conclusion

In conclusion, each deviation from our initial plan was aimed at overcoming specific challenges and improving the overall functionality and user experience of our web application. By adapting our approach to data collection, recommendation algorithms, and user interface design, we were able to create a more robust and user-friendly application that effectively presents key metrics and analyzed outputs. These adjustments not only addressed the constraints we faced but also contributed to a more successful and satisfactory project outcome.

3.3 How was the design implemented

The software specifications are outlined in E.1, the libraries used are listed in E.2, and the project management tools utilised are documented in Table E.3.

Chapter 4

Software Deliverables

4.1 Summary of software deliverables

The software deliverables for our project include a comprehensive suite of components designed to meet the specified requirements and ensure a high-quality user experience. The key deliverables are as follows:

1. **Model**
   The core of our software includes a sophisticated model called PyABSA. This model is responsible for generating all the necessary information that will be displayed to the user and is integral to the software's functionality.

2. **Backend**
   The backend of our software is robustly designed with MySQL Database and Flask API. It handles storage and communication between the database and the user interface.

3. **Executable Program**
   The primary deliverable is a fully functional web application. This web application is designed to address the specific needs identified in our project scope and provide a seamless user experience.

4. **Sample Code**
   The complete source code for the software (documented in appendix), written in Python (backend) and Flutter (frontend) are included. The code is well-documented, adhering to industry standards for readability and maintainability.

5. **Documentation**
   Comprehensive documentation accompanies the software, including user manuals, installation guides, and developer documentation. This ensures that users and future developers can effectively utilize and maintain the software.

6. **Sample Screenshots**
   A set of sample screenshots is provided to illustrate the user interface and demonstrate key functionalities of the software. These visuals offer a quick overview of what users can expect from the application.

Additionally, the following is our team's Github repository showcasing the entire source code of our project: https://github.com/regantan/FYP-FIT3162

4.1.1 Description of Usage

To start using our proposed solution, users will have to ensure the web application and our backend has been initialised. Step-by-step instructions have been provided in our User Guide to assist the user with this.

Once the user successfully initialises the backend and runs the web application, they will be greeted with our home page. The home page of the web application (as shown in Figure C.1) shows a list of restaurants in alphabetical order with some standard information for the user to view. The home page contains functionality such as the ability to search for particular restaurants and filter the restaurants based on their location. By default, the restaurants shown when the user first opens the web application will all be based in Kuala Lumpur. In terms of the filtering option, the user will have the option to filter restaurants in Kuala Lumpur and Rome.. If a user decides to search for a particular restaurant that does not exist, a message stating ”No restaurants available” will be displayed to the user (Figure C.2).

When a user clicks on a particular restaurant, the information about the restaurant is displayed in more detail, as illustrated in Figure C.4. It can be seen that we provide a list of the restaurant's aspects in a colour-coded form with a specific percentage to show how positive or negative the sentiment is towards each aspect. On the left of the screen is information about the restaurant and a time series graph detailing the restaurant's aspects over time. It provides an outlet for the user to understand better the progression of the aspects that they may care about instead of just relying on user reviews, which may be biased towards a certain sentiment, as older reviews may affect the overall average sentiment of the restaurant. The centre portion of the page contains all the restaurant reviews and the sentiment of aspects from each review are also extracted to better show the user how our model functions and to give the user a quick summary of the review itself. Lastly, the right side contains a button which, when clicked, will display a list of restaurants that are similar to the one the user is currently viewing (Figure C.6).

4.2 Software Qualities

4.2.1 Robustness

Robustness in our software is achieved through extensive testing and error-handling mechanisms. The system is designed to handle unexpected inputs and conditions gracefully without crashing. Part of it includes limiting the possible ways users can input (for example: a dropdown menu is provided for the user to filter the restaurants based on location instead of letting them manually type the location) to limit any potential chances of having an unexpected event occur as shown in C.3. However, there are limitations; certain edge cases might not be fully covered, and future iterations will focus on identifying and addressing these gaps.

4.2.2 Security

The software incorporates several security measures, including secure communication protocols, not collecting any data regarding the usage of our web application and limiting the transferring of sensitive data between the backend and the frontend. Despite these efforts, there are still potential security issues, particularly related to data integrity, which need continuous monitoring and improvement.

4.2.3 Usability

User-centric design principles have been employed to ensure the software is intuitive and easy to navigate. Usability testing with a diverse user group has been conducted to gather feedback and make necessary adjustments. In addition, we have also performed heuristic evaluation using the design principles set by Jakob Nielsen [4] to critically evaluate our user interface. We have also shown how we apply design principles by providing a breakdown of each heuristic that we complied with and the evidence that supports it as shown in Tables B.1, B.2 and B.3 located in the Appendix. Nonetheless, there may still be some usability issues that surface as the user base grows and the software is exposed to a wider audience.

4.2.4 Scalability

The software architecture is designed to support scalability, allowing it to handle increased loads and more complex tasks over time. Techniques such as breaking down retrieving data from the database through multiple APIs instead of retrieving everything from one single API and distributed processing are implemented. Furthermore, the paging system (as shown in the top right of the home page at C.1) is being used to query for a certain number of restaurants at a time to reduce the wait times in querying data from the front end and ensure that the user will have a smooth experience regardless of the amount of data that is in our system. Presently, we do not have any plans to deploy our web application and the web application will only run within the confines of the user's local environment. Therefore, the only limitations in terms of stability here would be the user's specifications of their local machine. However, there are potential bottlenecks in the system such as super long reviews that could affect performance under extreme conditions, which will need to be addressed in future updates.

4.2.5 Documentation and Maintainability

Comprehensive documentation is provided, covering user guides, installation procedures, and developer notes. The codebase is well-organized and follows industry standards to ensure maintainability. Despite these efforts, maintaining detailed and up-to-date documentation can be challenging and areas may require further clarity and detail to support future developers effectively.

4.3 Sample Source Code

The sample source code included is as follows:

1. **Training of the Model (shown in Sample Code D.1)**
   The following command allows the user to start training the model for ABSA.
2. **Utilising the Model (shown in Sample Code D.2)**
   The following command can be used to carry out inferencing using the trained model.
3. **API format (shown in Sample Code D.3)**
   The following piece of code shows an example of how the API would be structured. The API format here stores all the required information to be displayed when the user sees a list of restaurants on the home page of our web application.
4. **Executing an API call (shown in Sample Code D.4)**
   The following piece of code illustrates how an API call is handled when being requested by the web application. In this case, this function shows the execution of an API call when the web application needs a list of restaurants to be displayed on the home page based on the location chosen and the page of restaurants to display.
5. **Making an API call (shown in Sample Code D.5)**
   The following piece of code is used by our web application to query for the list of restaurants based on the location chosen and the page of restaurants to display to the user. Once the data from the API call is received, the data will be handled and processed into their classes of information accordingly.

The sample codes provided are meant to showcase how the user can train and use our model and how the data generated can be linked back to the web application. The function of retrieving a list of restaurants shows how an API call is being made in the frontend and how the backend function handles and processes the requests and returns the results in a form to be expected by the frontend.

Chapter 5

Outcomes

5.1 What has been implemented

Firstly, our team has implemented a data crawler to obtain a dataset of TripAdvisor restaurant reviews. We have scraped a total of over 50,000 reviews and 500 restaurants from two different locations which are Rome and Kuala Lumpur. We then carried out data pre-processing, which involved removing non-text elements and translating all the text into English.

Furthermore, one of the most important components we implemented was the development and deployment of Aspect-Based Sentiment Analysis (ABSA) model. Specifically, we tested three different ABSA frameworks including PyABSA, BERT ABSA, and ABSA Pytorch. After evaluation, we chose PyABSA as the best-performing model. This model which was fine-tuned by our team helped in facilitating the efficient analysis of different restaurants based on user reviews on TripAdvisor. Using the chosen model, we performed aspect extraction combined with sentiment analysis on the clusters, enabling us to identify restaurants with similar key aspects.

Finally, we transformed all the user-generated content into meaningful insights, including the assignment of sentiment scores, aspect terms, opinion terms, and categories for each restaurant. To deliver our findings, we developed a user-friendly web application using Flutter, supported by the Flash API for data provision. The web app features visual graphs to help users explore the analyzed data more effectively. Additionally, it included a filtering functionality and a search bar, allowing users to look up different restaurants and reviews.

5.2 Results Achieved/Product Delivered

5.2.1 Model Performance

In evaluating the performance of our PyABSA model, our team employed both macro and micro metrics. Macro metrics provide an average performance measure across all labels on the aspect, while micro metrics offer a more detailed view, analysing performance for each instance, regardless of the class. In addition, our team added the comparison of precision, recall and F1-score for both metrics on the aspects. The table below evaluates the performance of the PyABSA model:

Metrics	Precision		Recall		F1-score	
	Micro	Macro	Micro	Macro	Micro	Macro
Aspect	0.83	0.84	0.61	0.67	0.70	0.73
Polarity	0.98	0.97	0.60	0.66	0.74	0.76
Opinion	0.66	0.67	0.40	0.48	0.50	0.55
Category	0.80	0.79	0.49	0.55	0.61	0.63

Table 5.1: Table of performance of PyABSA model

Based on the outcomes in 5.1, the macro metrics consistently outperform the micro metrics in both recall and F1-score. This indicates that the model has a relatively balanced performance across all sub-tasks, including the less frequent ones. A higher macro F1-score suggests that the model maintains good performance across all sub-tasks rather than excelling in just some specific sub-tasks. Similarly, a higher macro recall reveals that the model effectively recalls instances from the less frequent sub-tasks, demonstrating strong performance in those areas as well.

Although the micro precision is better than the macro precision for both aspect category and polarity, the difference between them is minimal. This small discrepancy indicates that our model's ability to correctly predict positive instances is fairly consistent across all classes. Therefore, there is no significant concern regarding the model's performance in accurately predicting positive instances for each class in both aspects.

However, the performance of Opinion and Category sub-tasks are low for both micro and macro metrics, indicating that the model does not perform well in extracting the opinion terms and categorizing aspects. This suggests that while the model performs well in some areas, there are notable challenges in others that need to be addressed.

5.2.2 Web Application GUI Overall User Satisfaction Rate

As shown in F.1 in the Appendix, User Acceptance Testing (UAT) is conducted to evaluate the user-friendliness of our initial GUI developed, seeking feedback from participants and our supervisor. The obtained rating for each question, averaging at least 3.7 out of 5, surpasses our team's objective of 3/5, indicating a positive user response. Subsequent improvements are made based on the feedback provided to enhance the overall user experience.

5.3 How are requirements met

In the previous semester, our team defined and documented a set of requirements which act as guidelines for our project development in our project proposal to ensure the proper execution of the project. Throughout this semester, based on our supervisor's suggestions, we have made several modifications and introduced new requirements. The table below details how these functional requirements have been met.

ID	Functional requirements	How are requirements met
1	Process and analyse datasets obtained from TripAdvisor	Conduct descriptive analysis on the dataset to understand the structure of the dataset and data preprocessing is performed which includes the removal of missing or null data, removal of non-text, and conversion of date text into specified format before model training.
2	Develop, train, and fine-tune the ABSA model	Our ABSA model is developed by modifying the existing code functions sourced from the PyABSA framework. This model is trained on our annotated dataset, and then some of its parameters such as the 'num_train_epochs' are fine-tuned to improve model performance.
3	Develop an interactive user interface with functionalities.	The web application was successfully developed, allowing users to search for the restaurant.
4	Implement filter options for users to select locations.	The filtering functionality was implemented through a dropdown list.
5	The analysed data should be visualized in terms of user-friendliness.	A function is used to generate a time-series graph for review polarity based on aspect category.
6	Ensure the performance of the model on key metrics.	A function is used to calculate and evaluate the performance of the model through macro and micro precision, recall and F1 score on aspect term, polarity, aspect category and opinion term.
7	The web app should be able to provide similar restaurant recommendations.	A function is used to return a similar restaurant based on the cuisine type and similarity of aspect category and sentiment score.
8	There should be a user guide to help users understand how to access the web application properly.	A comprehensive user guide was created including the end user guide and technical guide.
9	Testing of the developed web application	We have developed comprehensive test cases to evaluate the web app, covering frontend-backend integration, API testing and various functions including testing the metric calculation functions. All these test cases have been successfully passed.

Table 5.2: Table of functional requirements

The table below details how these non-functional requirements have been met.

ID	Non-functional requirements	How are requirements met
1	Trained ABSA model achieves a minimum precision of 0.80.	Our final trained PyABSA model successfully achieved a micro and macro precision of 0.83 and 0.84 respectively on aspect term while a micro precision and macro precision of 0.98 and 0.97 respectively on polarity.
2	Adequate documentation of the model's code	Our GitHub codebase includes comprehensive function documentation and in-line comments to enhance code readability and understanding.
3	A satisfactory application in terms of user-friendliness.	Comprehensive usability testing with our coursemates returned with mostly positive feedback that the web app is easy to use.

Table 5.3: Table of non-functional requirements

5.4 Discussion of All Results

In this section, we will discuss and analyse the other results achieved from our ABSA and Recommendation System.

5.4.1 Validation of Correctness of ABSA Model

Models	Macro F1-Score
ABSA Model	0.63
ChatGPT 3.5	0.55
IMN-BERT	0.60
SPAN-BERT	0.62

Table 5.4: Table of F1 performance between models obtained from [47]

From Section 5.2.1, we saw that our model performed well in some sub-tasks but performed badly in others such as aspect categorisation and opinion extraction.

To further validate the correctness of our Aspect-Based Sentiment Analysis (ABSA) model, we conducted a series of tests and evaluations using established performance metrics from the aspect categorization subtask referenced in related academic papers such as yang & zhao [47]. These metrics provided a benchmark to compare the effectiveness and accuracy of our model.

In addition to traditional performance metrics, we employed ChatGPT to perform ABSA analysis on a set of test reviews. This provided a comparative analysis to gauge the performance of our model against a sophisticated language model. We calculated performance metrics such as the F1 score for both our model and the ChatGPT outputs. The F1 score is a harmonic mean of precision and recall, and it provides a single metric that balances both false positives and false negatives.

The results revealed that our model achieved an F1 score of 0.63, which, while the highest among the models compared, indicates room for improvement. This F1 score, though the best overall, highlights a need to enhance the model's accuracy and reliability.

The relatively low F1 score can primarily be attributed to the complexity and interdependence of the subtasks within the ABSA pipeline. Our model performs multiple steps: aspect extraction, opinion term extraction, sentiment classification, and finally, aspect categorization. Errors in the initial stages, such as aspect or opinion term extraction, propagate through the pipeline, leading to incorrect aspect categorization. This chaining effect underscores the challenge of maintaining high accuracy across all subtasks to achieve a robust overall performance.

Future work should focus on refining each subtask and improving the integration between these stages to minimize error propagation and enhance the overall performance of the ABSA model. Enhancements may include advanced techniques for aspect and opinion term extraction, as well as more sophisticated sentiment classification algorithms. By addressing these areas, we aim to achieve higher accuracy and reliability in the model's output.

5.4.2 Validation of Correctness of ABSA Model II

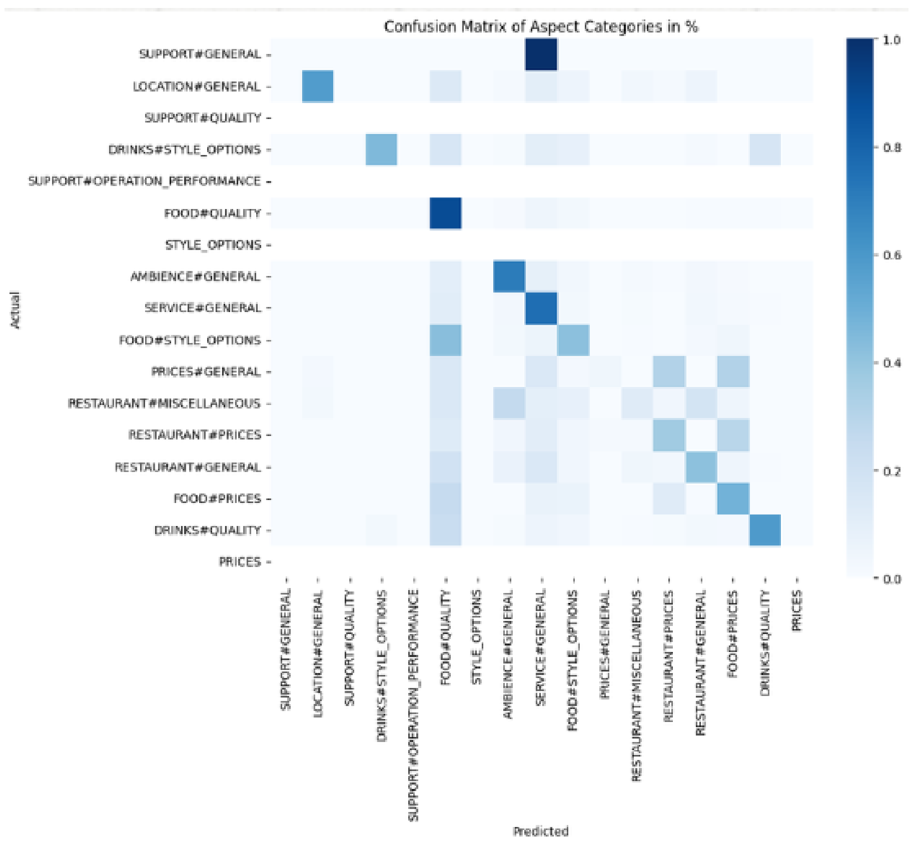


Figure 5.1: Confusion Matrix of the categorisation of aspects by the model

We also created a confusion matrix to understand why our aspect categorisation subtask was performing so badly when compared to other tasks as shown in 5.1

The confusion matrix provides insightful analysis into the performance of our Aspect-Based Sentiment Analysis (ABSA) model in categorizing restaurant review aspects. It highlights both strengths and weaknesses in the model's ability to differentiate between various aspect categories.

One notable observation is the frequent confusion between SUPPORT#GENERAL being mistaken as SERVICE#GENERAL. This confusion likely arises from the similarity in meaning between these terms, making it difficult for the model to discern the nuances in the text that differentiate them. This indicates a need for more context-aware training or additional feature engineering to help the model make finer distinctions.

The matrix also shows that most data are categorized correctly, with FOOD#QUALITY being an exception. FOOD#QUALITY is the most frequently discussed aspect in restaurant reviews, leading the model to incorrectly categorize other aspects under this label more often. This overrepresentation skews the model's predictions, suggesting that a more balanced dataset or data augmentation techniques might improve overall performance.

Another significant challenge highlighted by the confusion matrix is the lack of data in rare aspect categories. Categories such as SUPPORT#OPERATION_PERFORMANCE and SUPPORT#QUALITY are less common, making it harder for the model to learn and correctly categorize these aspects. This scarcity impacts the model's ability to generalize well across all categories, emphasizing the need for a more diverse and balanced dataset.

The confusion matrix further reveals that certain categories, like AMBIENCE#GENERAL and DRINKS#QUALITY, are predicted with reasonable accuracy, indicating that the model can effectively learn and distinguish these aspects when sufficient data is available. However, there is still room for improvement in categories that show high rates of misclassification.

In summation, the confusion matrix underscores the model's strengths in handling well-represented categories and highlights areas for improvement, particularly in addressing data imbalances and enhancing the model's ability to capture nuanced differences between similar aspect categories. Future work should focus on refining the training process, possibly through more sophisticated techniques or additional data collection, to address these challenges and improve the overall accuracy and robustness of the Aspect Categorisation sub-task.

5.4.3 Validation of usefulness Web Application Prototype

From section 5.2.2, to validate whether our final prototype was useful and met the project aims (PA4), we conducted usability testing and user acceptance testing (UAT). These tests ensured that our product was practical and achieved its intended use.

Initially, user feedback indicated that finding a specific restaurant to view aspect performance and discover similar restaurants was confusing. In response, we implemented a search feature to allow users to locate specific restaurants more easily. This change aimed to enhance user experience by streamlining navigation within the application.

After implementing this major feature change, we conducted another round of UAT. The results were relatively positive, with users expressing satisfaction with the improved functionality as shown in F.1. Minor feedback was received regarding the readability of the time series line graph, prompting us to add tooltips. These tooltips provide additional information when users hover over the graph, making it easier to interpret the data.

Additionally, users suggested incorporating more photos of the restaurants. However, due to time constraints and the lack of access to the TripAdvisor API, we decided not to include this feature in the final prototype. Despite this limitation, the feedback has been noted, and adding more visual content is a potential area for future improvement.

Overall, the iterative process of user feedback and subsequent enhancements has demonstrated the prototype's usefulness and its alignment with project goals. Future iterations could further refine the application by addressing additional user suggestions and incorporating more comprehensive data sources.

5.5 Limitations of Project Outcomes

In this section, we will discuss and analyse the limitations and problems encountered in our web application (RestoReview), ABSA model and TripAdvisor dataset.

5.5.1 Limitations of RestoReview (Web application)

The limitations of our application have already been discussed in the User Guide but will also be shown here in greater detail.

1. **Search Bar**
   While the search bar serves as a convenient tool for users to find specific restaurants, it's important to note that it currently only searches through the names of restaurants. As a result, users may encounter limitations in locating establishments based on cuisine types. This restriction means that users relying solely on the search bar may not efficiently explore restaurants that align with their preferred cuisine. As such, users are encouraged to utilize alternative methods, such as browsing through Google to find a restaurant that matches their desired cuisine type first, to enhance their search experience.

2. **Time Series Data Visualisation**
   Although the visualization provides valuable insights into the evolving reputation of restaurants, it's essential to acknowledge that it only displays a very broad view of the aspects. With numerous aspect categories available, this selective presentation may not fully represent the breadth of user opinions on various facets of the dining experience in detail. Users should be aware that the time series data may not capture insights into more specific time frames, potentially leading to an incomplete understanding of a restaurant's performance.

3. **Similar Restaurant Section**
   This feature does not take into account other contextual factors such as budget options or specific dietary preferences. As a result, users may miss out on discovering alternative dining options that align with their financial constraints or dietary restrictions. To address this limitation, users are encouraged to consider additional criteria beyond aspect performance and cuisine type when exploring similar restaurants, ensuring a more comprehensive and tailored selection process.

5.5.2 Limitations of ABSA Model

1. **Challenges in Annotating Data**
   Annotating a dataset of approximately 7,000 reviews presented significant challenges that delayed the timeline for training our ABSA model. Self-annotation is a time-consuming and labour-intensive process, requiring meticulous attention to detail to ensure accuracy and consistency. Although the annotations were carried out by peers, the process required constant oversight and verification to maintain high-quality annotations. This additional layer of quality control was necessary to ensure that the data was correctly annotated according to the specified criteria, which further extended the time needed to prepare the dataset. These challenges not only delayed the training phase but also impacted the overall project timeline, reducing the time available for other crucial tasks such as model fine-tuning, performance evaluation, and implementation of additional features in the web application. The extensive effort required for data annotation highlights the importance of having a well-structured and efficient process in place, as well as the potential benefits of using automated tools or pre-annotated datasets in future projects.

2. **Absence of Baseline Metrics**
   One of the significant challenges we faced was measuring the performance of our ABSA model before and after fine-tuning and training, due to the absence of baseline metrics provided by the PyABSA author and the model from Hugging Face. This lack of predefined metrics made it difficult to assess the initial performance of our model and quantify improvements achieved through fine-tuning. Without these baselines, we had to rely on our annotated dataset and custom performance evaluations, adding complexity to the validation process and requiring additional effort to establish our own benchmarks. This also hindered our ability to compare our results with other studies or implementations, limiting the contextualization of our findings within the broader research landscape. The absence of standardized metrics thus presented a significant obstacle in effectively measuring and demonstrating the improvements of our model, highlighting the need for comprehensive benchmarking in future model development and evaluation.

3. **Limitations in Fine Tuning**
   Fine-tuning our ABSA model to achieve the desired performance was hindered by several factors, including limited knowledge of deep learning techniques, resource constraints, and time limitations. Delving into the intricacies of fine-tuning models demands a significant investment of time and computational resources. Our efforts were further constrained by the computing capacity of our Tech Lead's personal computer. This disruption impacted our timeline and limited our capacity to optimally explore model parameters and architecture adjustments necessary for enhancing the model's performance.

5.5.3 Limitations of TripAdvisor Dataset

The section has already been discussed in the User Guide but will also be shown here in greater detail.

1. **Size of Dataset**
   Although our dataset comprises over 50,000 reviews spanning more than 1,000 restaurants scraped from TripAdvisor, it's essential to recognize that this dataset may not encompass the entirety of opinions and experiences for each restaurant. As such, users should approach the insights provided by our model with due consideration and supplement them with additional sources or personal experiences for a comprehensive assessment.

2. **Nature of user-generated content**
   Due to the use of user-generated text as our data, instances of incorrect English grammar or non-textual elements, such as emojis, may be present in the reviews. Although we've taken measures to filter out such anomalies during data processing, it's important to acknowledge that our model may not fully capture the syntactical nuances and emotional cues conveyed by emojis, potentially impacting the sentiment analysis.

3. **Limitations in Review Titles**
   Due to the limited text available in review titles, our Aspect-Based Sentiment Analysis (ABSA) model does not analyze the titles of each review. Consequently, the sentiment and aspect analysis provided by RestoReview may not fully reflect the content of review titles, potentially leading to a partial representation of the overall sentiment towards a restaurant. Users should be mindful of these limitations while interpreting the results provided by RestoReview to ensure a balanced understanding of the restaurant's reputation and performance.

4. **Imbalanced Dataset Distribution**
   One of the challenges we encountered was the imbalanced distribution of our dataset, primarily comprising 4-5 star reviews, with fewer low-ranking reviews (1-2 stars). Due to the nature of user-generated content, high-ranking reviews are more prevalent, while low-ranking reviews are less frequently written. This imbalance poses significant challenges for our ABSA model and the effectiveness of our application. With an overrepresentation of positive reviews, the model may become biased towards predicting positive sentiments, reducing its accuracy in identifying negative feedback. Addressing this imbalance is crucial for enhancing the model's performance and ensuring a more comprehensive analysis of all reviews. Techniques such as data augmentation, resampling methods, or obtaining additional low-ranking reviews could be explored to create a more balanced dataset, ultimately improving the robustness and accuracy of the model and increasing the overall utility of our application.

5.6 Justification of Decisions Made

5.6.1 Obtaining our dataset

In determining the method for obtaining the dataset for our project, several factors influenced our decision-making process. Firstly, we opted not to utilize the TripAdvisor API due to cost considerations. While the service offers the first 5000 API calls free every month, subsequent usage incurs charges, making it a pay-as-you-grow service [1]. Additionally, the unfamiliarity with using the API and the extensive documentation required posed significant challenges. The complexity of navigating through the documentation and the learning curve associated with API integration influenced our decision to explore alternative methods.

Consequently, we chose to manually scrape the data using a data crawler instead. Despite none of the team members having prior experience with web scraping using tools like Selenium, the abundance of online documentation and tutorials made it a viable option. Although this approach necessitates scraping the websites for updated data, unlike the API where updated data can be obtained easily, we prioritized the feasibility and cost-effectiveness of manual scraping given the constraints of our project.

In summary, while the API offered convenience and potentially updated data, cost considerations and the learning curve associated with its utilization led us to opt for manual data scraping using a data crawler. This decision allowed us to proceed with obtaining the necessary data for our project efficiently, despite the need for additional efforts in data retrieval.

5.6.2 Usage of T5 Model

In our project, we employed a tk-instruct model, leveraging a pre-trained T5 model to train and fine-tune our ABSA model using the PyABSA framework. This decision was deliberate and based on several factors.

Firstly, the pre-trained T5 model used in our framework was trained on positive, negative, and crucially, neutral sentiments [3]. This contrasts with other pre-trained models that only support positive and negative sentiments. Including neutral sentiment training data is vital for our ABSA task, as it allows the model to better understand and analyze nuanced sentiment expressions, enhancing its overall accuracy and performance.

Additionally, our model was based on InstructABSA, an instruction learning paradigm for Aspect-Based Sentiment Analysis (ABSA) subtasks. Exploring sample efficiency revealed that just 50% of training data is required to achieve competitive results with other instruction tuning approaches [39]. This approach enhances the efficiency of our model training process and allows for faster experimentation with different datasets and configurations.

Moreover, the T5 model was chosen over the proposed BERT model due to its flexibility in learning the model structure. Unlike BERT, which primarily focuses on understanding sentence-level representations, T5 is capable of performing various tasks using a text-to-text approach [20]. This versatility allows us to leverage T5 for aspect-based sentiment analysis while also potentially extending its use to other natural language processing tasks in the future [37].

In summary, our decision to utilize the T5 model framework was driven by its inclusion of neutral sentiment training data and its flexibility in learning complex model structures. These factors contributed to the effectiveness and adaptability of our ABSA model, aligning with our project goals of achieving accurate sentiment analysis across diverse review datasets.

5.6.3 Model Training with 16 Epochs

In determining the optimal number of epochs for training our ABSA model, we conducted experiments using different epoch configurations (2, 4, 8, and 16 epochs) and the results are shown in F.4. The aspect categorization subtask was chosen as the benchmark for comparison, considering the interdependence of subtasks and the potential impact on overall model performance. As discussed earlier, errors in earlier subtasks can propagate and affect aspect categorization, making it a critical metric for evaluation.

After analyzing the results, we found that training the model with 16 epochs yielded the highest F1 score for the aspect categorization subtask. This configuration demonstrated the most promising performance compared to the other epoch options tested. However, we did not explore training beyond 16 epochs due to time constraints. Fine-tuning the model with 16 epochs on the dataset of 7,000 reviews already required over 27 hours of computational time, limiting our ability to further extend the experimentation period.

In summary, our decision to train the model with 16 epochs was based on its superior performance in the aspect categorisation subtask, which we identified as a critical metric for evaluating model effectiveness. Despite the potential benefits of exploring additional epochs, time constraints necessitated prioritization, and we opted to focus on achieving the best results within the available timeframe.

5.7 Future Work

While our project has made significant strides in aspect-based sentiment analysis and restaurant recommendation, there remain several areas for improvement and future exploration.

Firstly, it's crucial to address the relatively low F1 score of our model. While we achieved promising results with the T5 model framework, exploring other transformer models such as BERT could potentially lead to improved performance. Fine-tuning the model architecture and experimenting with different hyperparameters may also yield better results in sentiment analysis tasks.

In terms of restaurant recommendation, our current algorithm focuses on aspects such as cuisine type and aspect performance. However, there is room for enhancement by incorporating additional factors such as price range, proximity to the user's location, and user preferences. Implementing a collaborative filtering [22] approach, which requires the collection of user data, could further personalize recommendations and improve user satisfaction.

Additionally, expanding the geographic coverage of our application is essential for catering to a broader user base. Currently, our project supports only two locations: Rome, Italy, and Kuala Lumpur, Malaysia. Future iterations could include support for additional cities and regions, allowing users from diverse locations to benefit from our platform.

Furthermore, enhancing the user interface and experience can contribute to the overall usability and adoption of our application. Incorporating features such as interactive maps, user reviews, and filtering options could improve user engagement and satisfaction.

Lastly, exploring novel techniques for data augmentation, such as generating synthetic reviews or leveraging unsupervised learning methods, could help address the imbalanced dataset distribution and improve the model's robustness.

In summary, there are numerous opportunities for improvement and expansion in our project, ranging from refining the model architecture and recommendation algorithm to enhancing the geographic coverage and user experience. Continued research and development in these areas will contribute to the evolution and success of our aspect-based sentiment analysis and restaurant recommendation platform.

Chapter 6

Critical discussion on the Software Project

In this section, we will discuss any matter relating to the project as a whole. Design deviations were mentioned in the Deviations section (3.2) but we will also delve deeper into other deviations as well as add more details to the ones mentioned earlier.

6.1 Programming Language

Our selection of Python as the primary language for developing our probabilistic and deep learning models, particularly leveraging PyTorch, aligned with our initial project proposal. Python's versatility and extensive libraries facilitated the implementation of complex machine-learning algorithms and data analysis tasks. Similarly, our use of the Dart language for the Flutter framework matched our project proposal's intent to build a cross-platform application, integrating Flutter for the front end and Flask for the backend API. However, due to time constraints, we couldn't fully implement the mobile aspect of our application as originally planned. Despite this deviation, our use of Python, Dart, Flutter, and Flask enabled us to develop a functional application with machine-learning capabilities for aspect-based sentiment analysis and restaurant recommendation.

6.2 Dataset

Our project proposal initially outlined the intention to collect TripAdvisor reviews of local restaurants to align with the domain and use case in our country. However, after consulting with our supervisor, we made the decision to expand the dataset to include reviews from two locations. This expansion allowed us to analyze the behaviour and outcomes of our model across different geographical regions. One notable observation from our dataset analysis was the tendency for restaurants in the Rome dataset to frequently recommend each other. This phenomenon could be attributed to the prevalence of Italian cuisine in Rome, resulting in a high degree of similarity among restaurants. Despite the deviation from the original plan, the decision to expand the dataset proved beneficial, providing valuable insights into the performance and applicability of our model across diverse locations.

6.3 User Interface

In terms of the user interface, our implementation closely followed the proposed design outlined in the initial project proposal. We meticulously adhered to the design specifications, ensuring consistency and coherence throughout the development process. While the core design remained intact, we made minor adjustments and refinements to enhance usability and functionality based on user feedback and evolving project requirements.

One significant enhancement was the incorporation of a search feature, as discussed earlier. Recognizing the importance of usability and accessibility, we introduced this feature to streamline the user experience and facilitate quicker navigation within the application. This addition not only improved the overall usability of the platform but also addressed user needs for efficient information retrieval.

Furthermore, we identified a crucial usability gap concerning the accessibility of restaurant performance data. Despite displaying overall aspect performance, we found it challenging to obtain a quick and comprehensive analysis of a restaurant's performance. To address this limitation, we introduced a time-series data line graph in a later development phase. This graph provided users with a visual representation of a restaurant's performance trends over time, empowering them to delve deeper into the data without the need to sift through individual reviews.

By integrating these iterative improvements, we optimized the application's usability and functionality to better meet user needs and expectations. While the core design remained faithful to the initial project proposal, these enhancements were pivotal in enhancing the overall user experience and ensuring the application's effectiveness in delivering valuable insights to users.

The success of adapting our user interface was significantly bolstered by the effectiveness of our Agile project management approach. Embracing Agile principles facilitated continuous iteration and improvement, allowing us to swiftly incorporate user feedback and evolving requirements into the interface design. Through close collaboration, transparent communication, and a focus on flexibility, Agile methodologies enabled us to identify and address usability issues effectively, ensuring that the user interface remained intuitive and user-friendly throughout the development process.

6.4 Recommendation System

Initially, our project proposal suggested the utilization of a clustering model for recommending similar restaurants to users. However, as the project progressed, we realized that the clustering model was not essential for our recommendation system. Instead, we opted for a more streamlined approach based on restaurant cuisine type and aspect performance.

Our decision to forego the clustering model was influenced by the project's evolving outcomes and the recognition that our recommendation system could operate effectively without it. Focusing on cuisine type and aspect performance allowed us to provide targeted and relevant recommendations to users, aligning more closely with their preferences and needs.

To calculate the similarities between restaurants, we employed cosine similarity [18] and the Jacquard Index [21]. This approach proved to be more direct and efficient compared to the clustering model. Unlike clustering, which can be slower and may incorrectly group similar restaurants, cosine similarity and the Jacquard Index offered a more accurate and precise method for determining restaurant similarities. Clustering models, depending on the algorithm used and the nature of the data, may inadvertently group restaurants based on factors other than their true similarity, leading to incorrect recommendations.

Overall, our recommendation system successfully achieved its intended purpose of providing users with personalized and relevant restaurant recommendations. By adapting to the project's evolving requirements and leveraging efficient similarity metrics, we were able to deliver a robust and effective recommendation system that enhances the overall user experience of our application.

6.5 Model Evaluation

In our project proposal, one of the non-functional requirements outlined in the requirement traceability matrix was to achieve a high accuracy of classification by the model, specifically targeting an accuracy of 0.70. However, as the project progressed, we encountered several complexities that influenced our approach to model evaluation and performance metrics.

Ultimately, we decided to use the F1 score as our main performance metric rather than accuracy. The rationale behind this choice is that the F1 score provides a better balance between precision and recall, especially in tasks with imbalanced datasets and multiple subtasks. Accuracy alone could be misleading in our context, as it doesn't account for the nuances of false positives and false negatives, which are critical in sentiment analysis and aspect categorization.

During the project proposal stage, we hadn't fully realized that our ABSA task would require the chaining of the subtasks—aspect extraction, opinion term extraction, sentiment classification, and aspect categorization. This chaining significantly increases the difficulty of achieving a high-performance model, as errors can propagate through the pipeline, compounding the challenge of accurate classification.

Despite these challenges, our model, while not achieving the initially targeted 0.70 F1 score, still performed admirably. Our research demonstrated the best overall performance compared to other models referenced in the literature F.3. This comparative success underscores the robustness of our approach, even if the absolute performance metric fell short of the initial projection.

In summary, while the actual outcomes diverged from the initial project proposal in terms of achieving the desired accuracy, the decision to prioritize the F1 score provided a more meaningful evaluation of our model's performance. The complexities of the ABSA task and the chaining of subtasks were significant factors that influenced the final results, yet our model's relative success against other benchmarks highlights the effectiveness of our approach.

6.6 Risk Management

Effective risk management was a cornerstone of our project, encompassing the identification, analysis, and mitigation of potential risks. Regular updates to the risk register (F.7), including the probability/impact matrix, were maintained throughout the project. Notable risks encountered and managed include:

1. **Project Delay**
   Trigger: A lack of communication during the summer break and some team members underestimating their internship workload led to project delays. This also impacted the timeline for training the model.
   Handling: The team responded by agreeing to increase work hours and updating the Gantt chart to set a new deadline. Specifically, we added an extra 7 man-days to the schedule to accommodate the delay. This reallocation of resources ensured that project deliverables could be achieved within the revised schedule.

2. **Unable to Obtain Desired Dataset**
   Trigger: Due to the cost of TripAdvisor APIs and insufficient documentation, we faced challenges in acquiring data from the website.
   Handling: Following consultations with our supervisor and Dr. Lim's guidance, we implemented an alternative plan using Selenium for web scraping to retrieve the desired dataset. This required an additional 5 man-days for research and implementation of the web crawler, but it effectively resolved the data acquisition issue without incurring extra costs from API usage.

3. **Misunderstanding in Project Scope**
   Trigger: A recent meeting with our supervisor revealed a misunderstanding regarding the project scope. We had initially planned to evaluate one model before implementation, whereas the requirement was to compare and fine-tune multiple model frameworks before choosing one.

Handling: We addressed this by reallocating 3 man-days to cover the evaluation of multiple models. Additionally, to prevent similar issues, we agreed to hold more frequent meetings with supervisors, approximately twice a week, to monitor progress and ensure alignment with project requirements.

4. **Failure to Train Model on Time**
Trigger: Similar to the project delay risk, underestimating the internship workload and a lack of experience in model training posed a challenge.
Handling: Under our supervisor's guidance, we decided to use pre-trained models and apply our fine-tuning methods. This decision saved us approximately 10 man-days, which would have been required to build and train a model from scratch, thus keeping the project on track.

6.7 Summary

The whole team agreed that the project was a success. Most aspects of the software development followed the initial project proposal closely, demonstrating our ability to plan and execute effectively. Where deviations occurred, they were declared honestly and justified with valid reasons such as time constraints, technical knowledge gaps, and resource limitations.

These deviations included decisions like opting for a more straightforward recommendation algorithm over a clustering model and choosing to use the F1 score as our primary performance metric instead of accuracy. Despite these changes, the project met its core objectives, delivering a functional and valuable application for analyzing restaurant reviews and providing recommendations. Overall, the project's outcomes align well with our initial goals, showcasing our adaptability and problem-solving skills throughout the development process.

Chapter 7

Conclusion

This report showcased our journey in developing an Aspect-Based Sentiment Analysis (ABSA) and recommendation system for TripAdvisor reviews of restaurants, encapsulated in the RestoReview Project. Each chapter delved into the various components and decisions integral to our project, providing a comprehensive overview of our work.

The project background and literature review demonstrated our understanding and contextual knowledge in the realm of ABSA and recommendation systems. We explored how companies implement recommendation systems and highlighted the latest technologies, such as transformers, to achieve optimal performance.

In our methodology, we compared various ABSA frameworks and ultimately decided on the PyABSA framework due to its extensive documentation and active maintenance. We detailed our data collection approaches, model training processes, and the strategies we used to measure and evaluate our model's performance.

The evaluation and success of our project were measured using metrics such as precision, recall, and F1 score across micro and macro performances. Our model's superior performance compared to existing models in the research literature affirmed the success of our project while also highlighting areas for improvement.

We also discussed the critical decisions made throughout the project, noting how our thinking and approaches evolved from the initial proposal. These reflections underscored our adaptability and problem-solving skills, essential for navigating the challenges encountered during the project.

In conclusion, this report illustrates the comprehensive journey we undertook to develop our novel solution, providing an in-depth analysis of restaurant reviews in the Malaysian context. The potential to incorporate more sophisticated recommendation systems and improve our existing models opens up numerous opportunities for future work. Ultimately, the RestoReview Project not only achieved its primary objectives but also laid a solid foundation for future enhancements and applications in the field of sentiment analysis and recommendation systems.

Future work includes improving the model by exploring other transformer models such as BERT or utilising data augmentation, improving the recommendation algorithm by adding other factors such as the price range of the restaurant and collaborative filtering as well as improving the software functionalities such as expanding the geographic coverage of restaurants or incorporating iterative features in the interface.

Appendix A

Data Flow Diagram

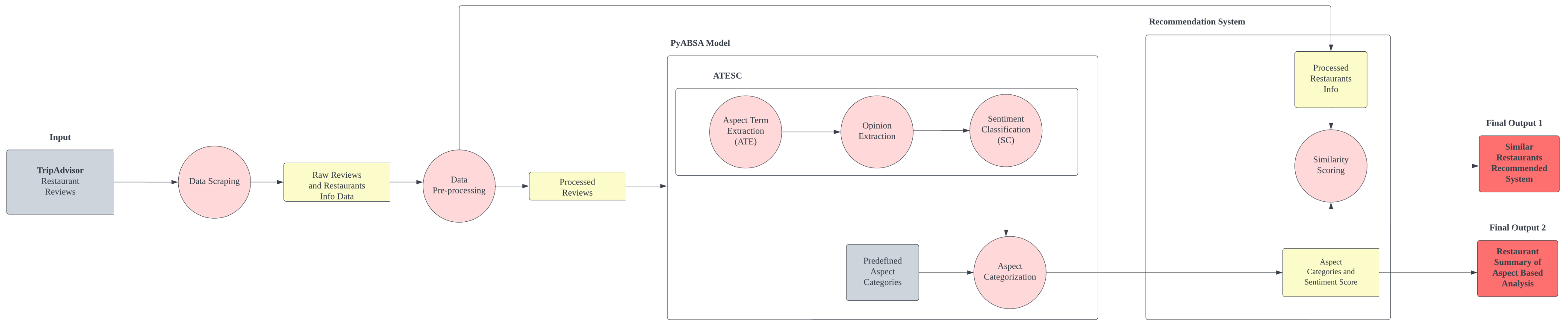

Input
TripAdvisor Restaurant Reviews
Data Scraping
Raw Reviews and Restaurants Info Data
Data Pre-processing
Processed Reviews
PyABSA Model
ATESC
Aspect Term Extraction (ATE)
Opinion Extraction
Sentiment Classification (SC)
Predefined Aspect Categories
Aspect Categorization
Recommendation System
Processed Restaurants Info
Similarity Scoring
Aspect Categories and Sentiment Score
Final Output 1
Similar Restaurants Recommended System
Final Output 2
Restaurant Summary of Aspect Based Analysis

Figure A.1: Level 0 Data flow diagram

Appendix B

Heuristic Evaluation

ID	Heuristic/Rule	Evidence
H1	Visibility of system status	When the system is processing a request or fetching data, the web application provides users with immediate feedback through visual indicators in the form of progress bars. For example, when a user searches for a particular restaurant, a progress bar appears, indicating that the system is fetching data. Upon completion, a list of restaurants is shown. If no restaurants match the search terms, a message appears stating no matches found (Figure C.2). This ensures users know their actions are being processed and whether the request was successful.
H2	Match between system and the real world	The user interface (shown in C) uses language easily understood by day-to-day users instead of technical terms. Terms like "Search Restaurant", "Ratings and Reviews", and "Recommendations" are used. When users search for a restaurant, the homepage displays information such as restaurant names, overall rating, number of reviews, and cuisine type, similar to other review websites. This familiar presentation helps users quickly understand and use the information, ensuring a seamless connection between the system and the real world.
H3	User control and freedom	The application offers a back button to the previous screen and the ability to tap the "RestoReview" texts at the top of the screen (C.4) to go back directly to the home page giving users a sense of control over their actions. It allows them to experiment and correct mistakes without fear of irreversible consequences, enhancing their confidence and willingness to explore the application.
H4	Consistency and standards	Consistency in design, terminology, and functionality help users predict how the system will behave and how to interact with it. In RestoReview, this means using consistent layouts, colors, and fonts across different pages of the application like having the same format for displaying the list of restaurants in the home page and the list of similar restaurants in the restaurant details page. It also involves using standard terminology and icons that users are familiar with, such as using yellow stars to represent the overall rating of the restaurant, which can be found in many other review websites as well. Consistency and adherence to standards can reduce cognitive load and improve user usability.

Table B.1: Heuristic Evaluation of our user interface (Part 1)

ID	Heuristic/Rule	Evidence
H5	Error prevention	We believe that it is always better to prevent errors from occurring in the first place than to provide error messages after the fact. In RestoReview, we have limited the possibilities of the ways users can input information into our system. For example, we provide the user with a drop-down menu (as shown in C.3 to filter the restaurants based on location instead of having them type the location themselves. This helps to prevent unnecessary errors that could caused as a result of user input. In addition, as a safeguard, we also implemented validation checks through our backend to ensure that users enter valid information. These steps help the application to prevent errors and enhance the user experience.
H6	Recognition rather than recall	Our goal is to minimize the user's memory load by making objects, actions, and options visible and easily accessible can improve usability. In RestoReview, this can be achieved by providing a list of restaurants in alphabetical order. When users are presented with a list of restaurants in alphabetical order, they can easily recognize and locate specific restaurants without having to recall their names from memory. This reduces the cognitive effort required to find a particular restaurant and makes the process more intuitive and user-friendly. In addition, providing a search bar further enhances this heuristic by allowing users to quickly find a specific restaurant by typing its name, further reducing the need for recall. Overall, these features help users navigate the application more effectively and contribute to a positive user experience.
H7	Flexibility and efficiency of use	Apart from making our web application itself user-friendly, the flexibility and efficiency of use could also be heavily influenced by the setup and installation experience. Due to our web application not being deployed, users will have to follow the instructions given to set up and install all necessary libraries and prerequisites in order to use our web application, Unfortunately, this process may be time-consuming and frustrating to the end user.
H8	Aesthetic and minimalist design	Our web application primarily focuses pm providing a simple, minimalist and eye-catching user interface. This involves laying out the list of restaurants in an organised grid form with each cell having the same width and height. Furthermore, material design principles are incorporated to help theme the user interface in a way that is clean, organised and fulfills our branding. Aesthetic and minimalist design can make the application more visually appealing, eliminate any unwanted distractions and make it easier to use, leading to a more enjoyable user experience.

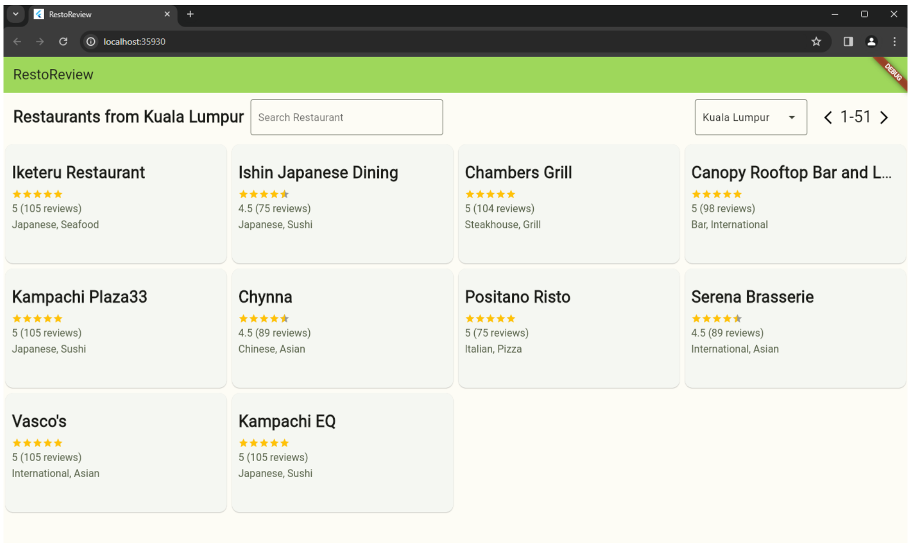
Table B.2: Heuristic Evaluation of our user interface (Part 2)

ID	Heuristic/Rule	Evidence
H9	Help users recognize, diagnose and recover from errors	Our application has been robustly designed to help ensure the user is able to solve any technical issues. For example, if the user did not properly setup or run the backend system before running the web application, error messages will be shown to the user that mentioned API calls cannot be made to the backend. These steps will provide the user with a clear message on the reasons behind the issue and the necessary steps needed to solve it.
H10	Help and documentation	In our user interface, we have provided labels and hints to provide guidance to the user on which part of the application they are in and how to access the relevant information. Furthermore, we have provided a detailed user guide detailing all the steps required on how to use the user interface. This helps relieve any concerns that users may have and can enhance user confidence and ensure that users can use the application effectively, even if they encounter difficulties.

Table B.3: Heuristic Evaluation of our user interface (Part 3)

Appendix C

User Interface



RestoReview

Restaurants from Kuala Lumpur | Search Restaurant | Kuala Lumpur | 1-51

Iketeru Restaurant
5 (105 reviews)
Japanese, Seafood

Ishin Japanese Dining
4.5 (75 reviews)
Japanese, Sushi

Chambers Grill
5 (104 reviews)
Steakhouse, Grill

Canopy Rooftop Bar and L...
5 (98 reviews)
Bar, International

Kampachi Plaza33
5 (105 reviews)
Japanese, Sushi

Chynna
4.5 (89 reviews)
Chinese, Asian

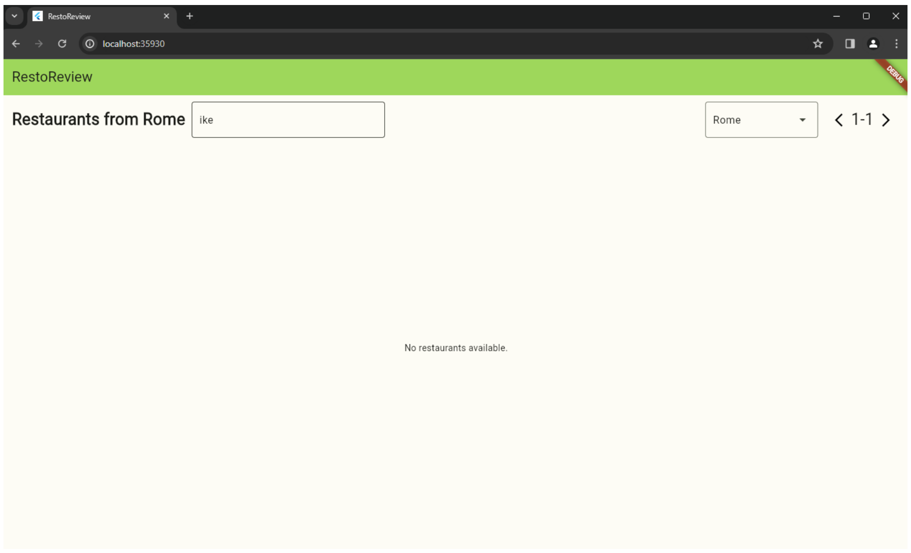
Positano Risto
5 (75 reviews)
Italian, Pizza

Serena Brasserie
4.5 (89 reviews)
International, Asian

Vasco's
5 (105 reviews)
International, Asian

Kampachi EQ
5 (105 reviews)
Japanese, Sushi

Figure C.1: Screenshot of Home page of RestoReview



RestoReview

Restaurants from Rome | ike | Rome | 1-1

No restaurants available.

Figure C.2: Screenshot of using the search functionality that results in no restaurants match

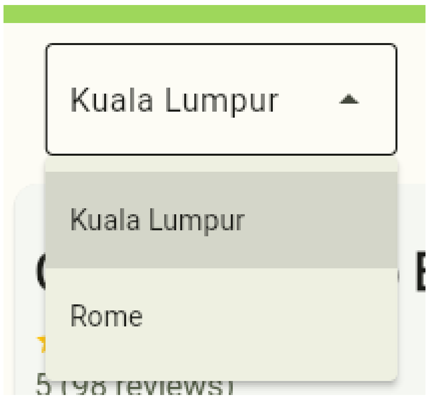
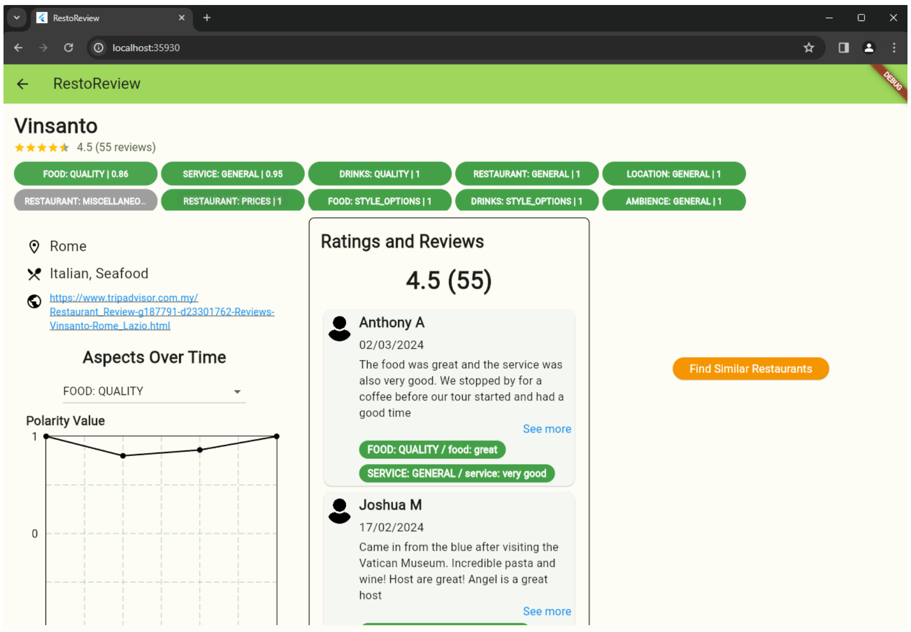


Figure C.3: Screenshot of Dropdown Menu of locations



Vinsanto

4.5 (55 reviews)

Rome

Italian, Seafood

https://www.tripadvisor.com.my/Restaurant_Review-g187791-d23301762-Reviews-Vinsanto-Rome_Lazio.html

Ratings and Reviews

4.5 (55)

Anthony A
02/03/2024
The food was great and the service was also very good. We stopped by for a coffee before our tour started and had a good time

Joshua M
17/02/2024
Came in from the blue after visiting the Vatican Museum. Incredible pasta and wine! Host are great! Angel is a great host

Figure C.4: Screenshot of individual restaurant page

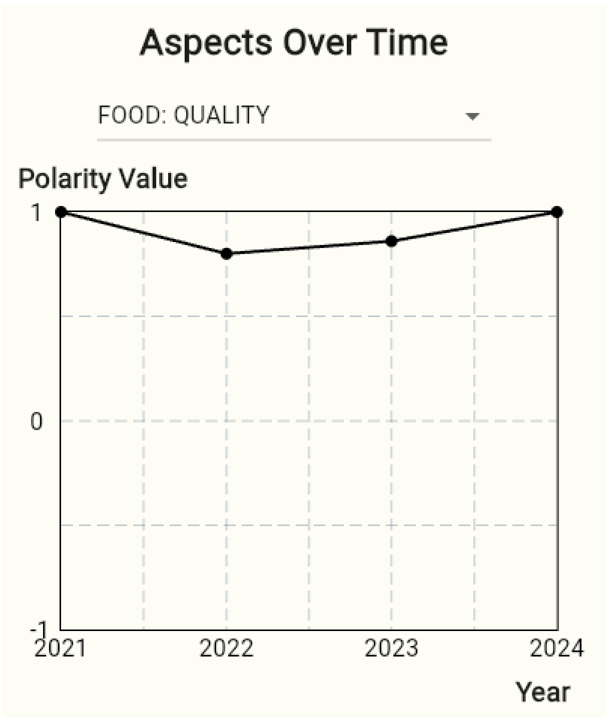
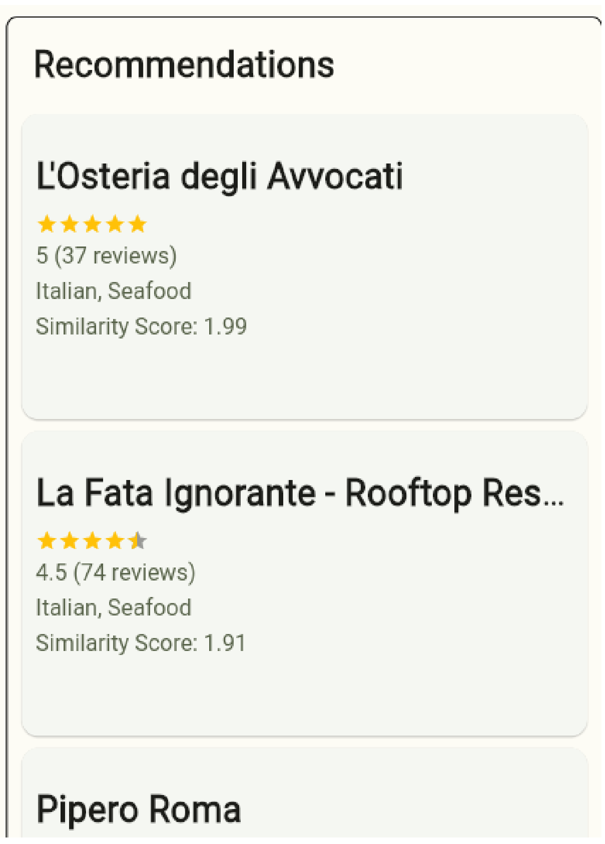


Figure C.5: Time Series graph of aspects section in restaurant page



Recommendations

L'Osteria degli Avvocati

5 (37 reviews)

Italian, Seafood

Similarity Score: 1.99

La Fata Ignorante - Rooftop Res...

4.5 (74 reviews)

Italian, Seafood

Similarity Score: 1.91

Pipero Roma

Figure C.6: Similar Restaurants page after clicking the button

Appendix D

Sample Source Code

```python
import os
import warnings

import findfile
from pyabsa import ABSAInstruction as absa_instruction

warnings.filterwarnings("ignore")
import pandas as pd


task_name = "multitask"
experiment_name = "instruction"
model_checkpoint = 'checkpoints\multitask\kevinscariaate_tk-
                    instruct-base-def-pos-neg-neut-combined-instruction'

print("Experiment Name: ", experiment_name)
model_out_path = "checkpoints"
model_out_path = os.path.join(
    model_out_path, task_name, f"{model_checkpoint.replace('/',
        ↪ '')}-{experiment_name}"
)
print("Model output path: ", model_out_path)

# Load the data
id_train_file_path = './data_annotation/full'
id_test_file_path = './data_annotation/full'


id_tr_df = absa_instruction.data_utils.read_json(id_train_file_path, "train")
id_te_df = absa_instruction.data_utils.read_json(id_test_file_path, "test")

id_tr_df = pd.DataFrame(id_tr_df)
id_te_df = pd.DataFrame(id_te_df)

loader = absa_instruction.data_utils.InstructDatasetLoader(id_tr_df, id_te_df)

if loader.train_df_id is not None:
    loader.train_df_id =
        ↪ loader.prepare_instruction_dataloader(loader.train_df_id)
if loader.test_df_id is not None:
    loader.test_df_id =
        ↪ loader.prepare_instruction_dataloader(loader.test_df_id)
if loader.train_df_ood is not None:
    loader.train_df_ood =
        ↪ loader.prepare_instruction_dataloader(loader.train_df_ood)
```

```python
if loader.test_df_ood is not None:
    loader.test_df_ood =
        ↪ loader.prepare_instruction_dataloader(loader.test_df_ood)

# Create T5 utils object
t5_exp = absa_instruction.model.T5Generator(model_checkpoint)

# Tokenize Dataset
id_ds, id_tokenized_ds, ood_ds, ood_tokenzed_ds = loader.create_datasets(
    t5_exp.tokenize_function_inputs
)

# Training arguments
training_args = {
    "output_dir": model_out_path,
    "evaluation_strategy": "epoch",
    "save_strategy": "epoch",
    "learning_rate": 5e-5,
    "per_device_train_batch_size": 4,
    "per_device_eval_batch_size": 16,
    "num_train_epochs": 16,
    "weight_decay": 0.01,
    "warmup_ratio": 0.1,
    "load_best_model_at_end": True,
    "push_to_hub": False,
    "eval_accumulation_steps": 1,
    "predict_with_generate": True,
    "logging_steps": 1000000000,
    "use_mps_device": False,
    'fp16': False,
}

model_trainer = t5_exp.train(id_tokenized_ds, **training_args)
```

Sample Code D.1: The instructions and parameters used to train our model

```python
import csv
import json
from pyabsa import ABSAInstruction

def predict_reviews(csv_file, model):
    predictions = []
    with open(csv_file, 'r', newline='', encoding='utf-8') as file:
        reader = csv.DictReader(file)
        for row in reader:
            review_text = row['Review']
            result = model.predict(review_text)
            row['Quadruples'] = result['Quadruples']
            predictions.append(row)
    return predictions

if __name__ == "__main__":
    model = ABSAInstruction.ABSAGenerator("checkpoints/multitask/
            kevinscariaate_tk-instruct-base-def-pos-neg-neut-
            combined-instruction-instruction")

    csv_file = "dataset\cleaned_reviews\cleaned_reviews_ROME.csv"
    jsonl_file = "reviews_ROME.jsonl"

    predictions = predict_reviews(csv_file, model)

    with open(jsonl_file, 'w', encoding='utf-8') as file:
        for prediction in predictions:
            json.dump(prediction, file)
            file.write('\n')

    print("Predictions saved to:", jsonl_file)
```

Sample Code D.2: The code used to carry out inferencing using the trained model

```
##### API models #####
recommended_restaurant_model = api.model('recommended_restaurant', {
    'id': fields.Integer(description='The unique identifier of a restaurant'),
    'restaurant_name': fields.String(required=True, description='Name of the restaurant'),
    'cuisine': fields.String(required=True, description='Type of cuisine offered'),
    'star_rating': fields.Float(description='Rating of the restaurant'),
    'no_reviews': fields.Float(description='Total number of reviews'),
    'trip_advisor_url': fields.String(description='TripAdvisor url')
})
```

Sample Code D.3: The api output format that will contains a list of recommended restaurants from a particular location

```
@api.route('/api/recommended_restaurants/<string:location>
/<int:page>')
@api.doc(params={'location': 'The location for which to find restaurants'})
class recommended_restaurants(Resource):
    # @api.marshal_list_with(recommended_restaurant_model)
    def get(self, location, page):
        per_page = 10
        offset = (page - 1) * per_page

        cursor = mysql.connection.cursor()

        cursor.execute("SELECT id, restaurant_name, cuisine, star_rating, url FROM restaurant_info WHERE location = %s ORDER BY restaurant_name LIMIT %s OFFSET %s", (location, per_page, offset))
        restaurants = cursor.fetchall()

        # Initialize a list to hold the final restaurant data
        restaurants_data = []

        # Process each restaurant to fetch the number of reviews
        for restaurant in restaurants:
            restaurant_id, restaurant_name, cuisine, star_rating, url = restaurant

            # Execute query to count the number of reviews for the current restaurant
            cursor.execute("SELECT COUNT(*) FROM reviews WHERE restaurant = %s", (restaurant_name,))
            total_reviews = cursor.fetchone()[0]

            # Append the restaurant data with the number of reviews
            restaurants_data.append({
                'id': restaurant_id,
                'restaurant_name': restaurant_name,
                'cuisine': [c.strip() for c in cuisine.split(',')] if cuisine else [],
                'star_rating': float(star_rating),
                'no_reviews': total_reviews,
                'trip_advisor_url': url
            })

        cursor.close()
        return jsonify(restaurants_data)
```

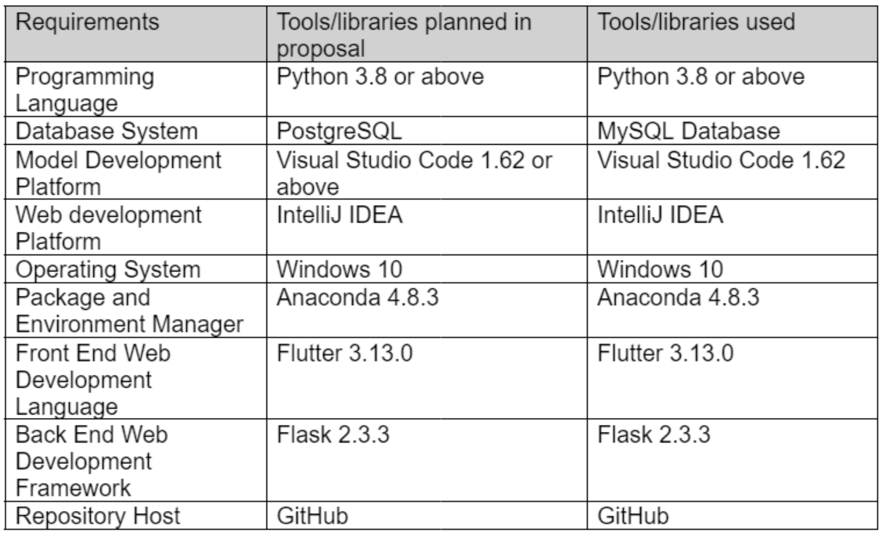
Sample Code D.4: The python code used to retrieve the list of recommended restaurants based on the location given and the page it requests

```
/**
 * Method to fetch a list of restaurants from the API
 */
Future<List<Restaurant>> fetchRestaurants(String city, int page, String
    ↪ searchTerm) async {
  http.Response response;
  if (searchTerm.isEmpty) {
    response = await http.get(Uri.parse('http://127.0.0.1:8079/api/
    recommended_restaurants/${city}/${page}'));
  }
  else {
    response = await http.get(Uri.parse('http://127.0.0.1:8079/api/
    search/${searchTerm}/${city}/${page}'));
  }

  if (response.statusCode == 200) {
    List<dynamic> data = json.decode(response.body);

    return data.map((json) => Restaurant.fromJson(json)).toList();
  } else {
    throw Exception('Failed to load restaurants');
  }
}
```

Sample Code D.5: The frontend calling the Flask API to retrieve a list of restaurants from the database

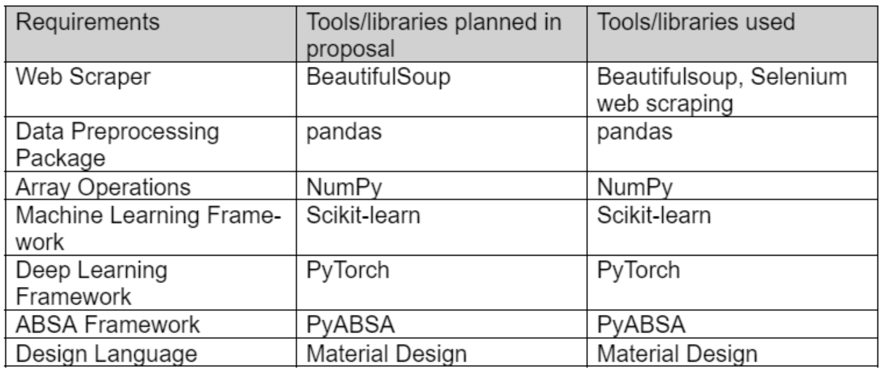
Appendix E

Software Modules Used



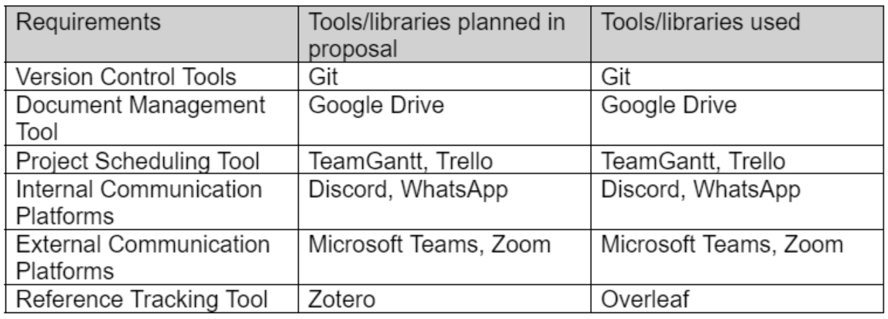
Requirements	Tools/libraries planned in proposal	Tools/libraries used
Programming Language	Python 3.8 or above	Python 3.8 or above
Database System	PostgreSQL	MySQL Database
Model Development Platform	Visual Studio Code 1.62 or above	Visual Studio Code 1.62
Web development Platform	IntelliJ IDEA	IntelliJ IDEA
Operating System	Windows 10	Windows 10
Package and Environment Manager	Anaconda 4.8.3	Anaconda 4.8.3
Front End Web Development Language	Flutter 3.13.0	Flutter 3.13.0
Back End Web Development Framework	Flask 2.3.3	Flask 2.3.3
Repository Host	GitHub	GitHub

Figure E.1: Software specifications used



Requirements	Tools/libraries planned in proposal	Tools/libraries used
Web Scraper	BeautifulSoup	Beautifulsoup, Selenium web scraping
Data Preprocessing Package	pandas	pandas
Array Operations	NumPy	NumPy
Machine Learning Frame-work	Scikit-learn	Scikit-learn
Deep Learning Framework	PyTorch	PyTorch
ABSA Framework	PyABSA	PyABSA
Design Language	Material Design	Material Design

Figure E.2: External libraries used

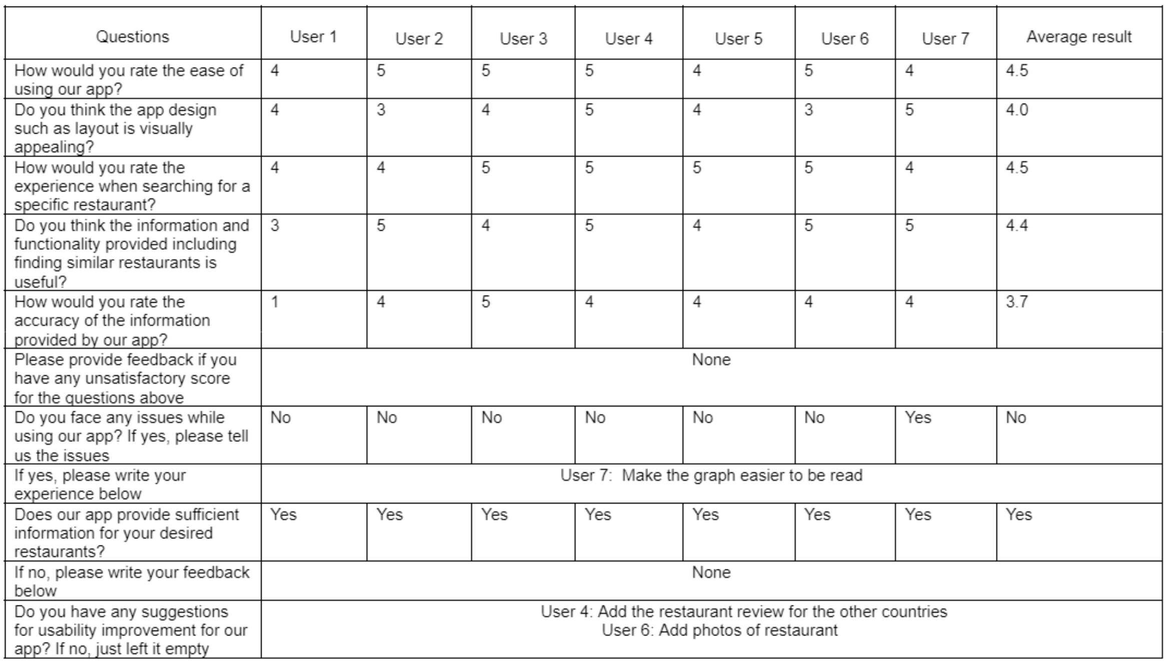


Requirements	Tools/libraries planned in proposal	Tools/libraries used
Version Control Tools	Git	Git
Document Management Tool	Google Drive	Google Drive
Project Scheduling Tool	TeamGantt, Trello	TeamGantt, Trello
Internal Communication Platforms	Discord, WhatsApp	Discord, WhatsApp
External Communication Platforms	Microsoft Teams, Zoom	Microsoft Teams, Zoom
Reference Tracking Tool	Zotero	Overleaf

Figure E.3: Project management tools used

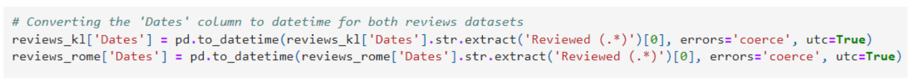
Appendix F

Others



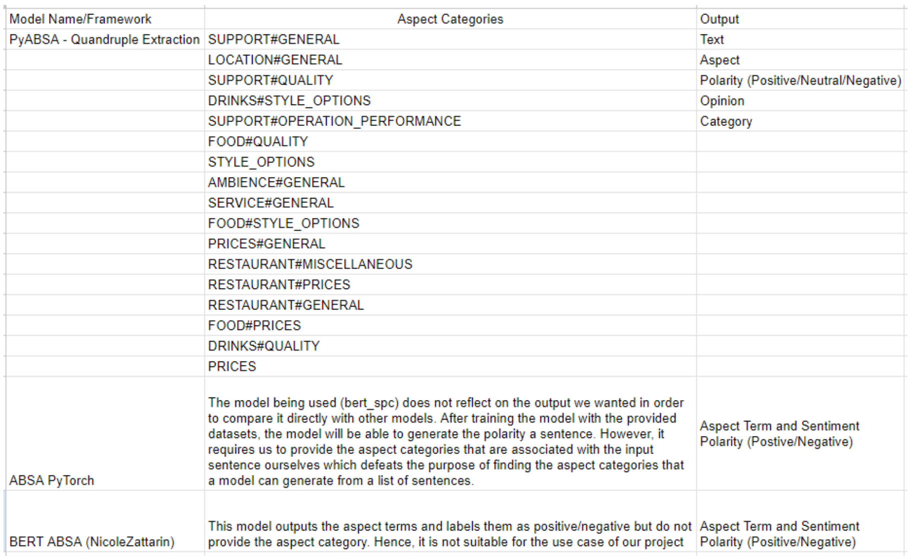
Questions	User 1	User 2	User 3	User 4	User 5	User 6	User 7	Average result
How would you rate the ease of using our app?	4	5	5	5	4	5	4	4.5
Do you think the app design such as layout is visually appealing?	4	3	4	5	4	3	5	4.0
How would you rate the experience when searching for a specific restaurant?	4	4	5	5	5	5	4	4.5
Do you think the information and functionality provided including finding similar restaurants is useful?	3	5	4	5	4	5	5	4.4
How would you rate the accuracy of the information provided by our app?	1	4	5	4	4	4	4	3.7
Please provide feedback if you have any unsatisfactory score for the questions above	None							
Do you face any issues while using our app? If yes, please tell us the issues	No	No	No	No	No	No	Yes	No
If yes, please write your experience below	User 7: Make the graph easier to be read							
Does our app provide sufficient information for your desired restaurants?	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
If no, please write your feedback below	None							
Do you have any suggestions for usability improvement for our app? If no, just left it empty	User 4: Add the restaurant review for the other countries User 6: Add photos of restaurant							

Figure F.1: Usability testing summary



```
# Converting the 'Dates' column to datetime for both reviews datasets
reviews_kl['Dates'] = pd.to_datetime(reviews_kl['Dates'].str.extract('Reviewed (.*)')[0], errors='coerce', utc=True)
reviews_rome['Dates'] = pd.to_datetime(reviews_rome['Dates'].str.extract('Reviewed (.*)')[0], errors='coerce', utc=True)
```

Figure F.2: Screenshot of code converting date into datetime



Model Name/Framework	Aspect Categories	Output
PyABSA - Quandruple Extraction	SUPPORT#GENERAL	Text
	LOCATION#GENERAL	Aspect
	SUPPORT#QUALITY	Polarity (Positive/Neutral/Negative)
	DRINKS#STYLE_OPTIONS	Opinion
	SUPPORT#OPERATION_PERFORMANCE	Category
	FOOD#QUALITY	
	STYLE_OPTIONS	
	AMBIENCE#GENERAL	
	SERVICE#GENERAL	
	FOOD#STYLE_OPTIONS	
	PRICES#GENERAL	
	RESTAURANT#MISCELLANEOUS	
	RESTAURANT#PRICES	
	RESTAURANT#GENERAL	
	FOOD#PRICES	
	DRINKS#QUALITY	
	PRICES	
ABSA PyTorch	The model being used (bert_spc) does not reflect on the output we wanted in order to compare it directly with other models. After training the model with the provided datasets, the model will be able to generate the polarity a sentence. However, it requires us to provide the aspect categories that are associated with the input sentence ourselves which defeats the purpose of finding the aspect categories that a model can generate from a list of sentences.	Aspect Term and Sentiment Polarity (Postive/Negative)
BERT ABSA (NicoleZattarin)	This model outputs the aspect terms and labels them as positive/negative but do not provide the aspect category. Hence, it is not suitable for the use case of our project	Aspect Term and Sentiment Polarity (Positive/Negative)

Figure F.3: Table comparing the aspects categories of model frameworks as well as their outputs

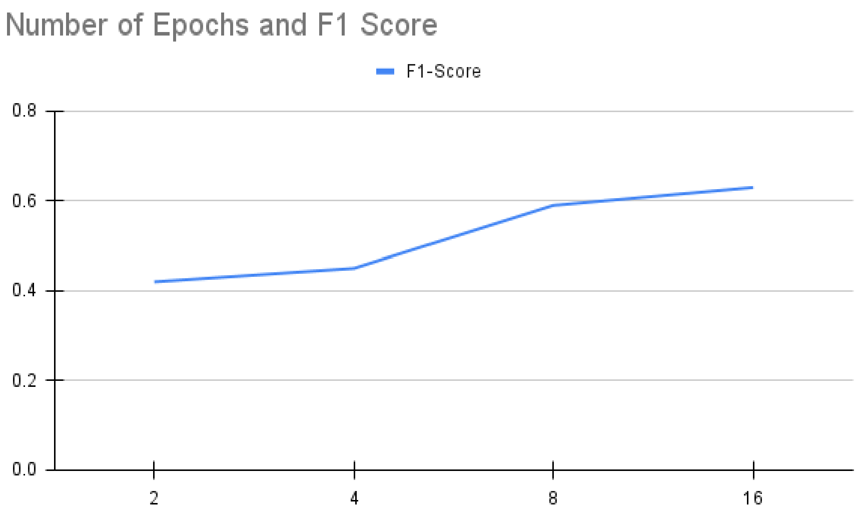
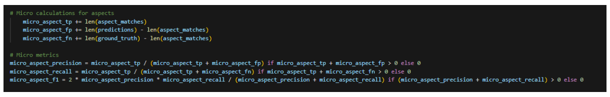
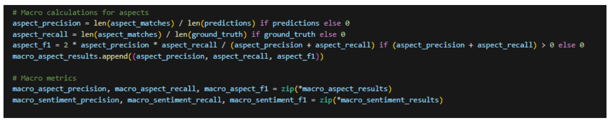


Figure F.4: Line graph comparing number of epochs and F1 score of trained model



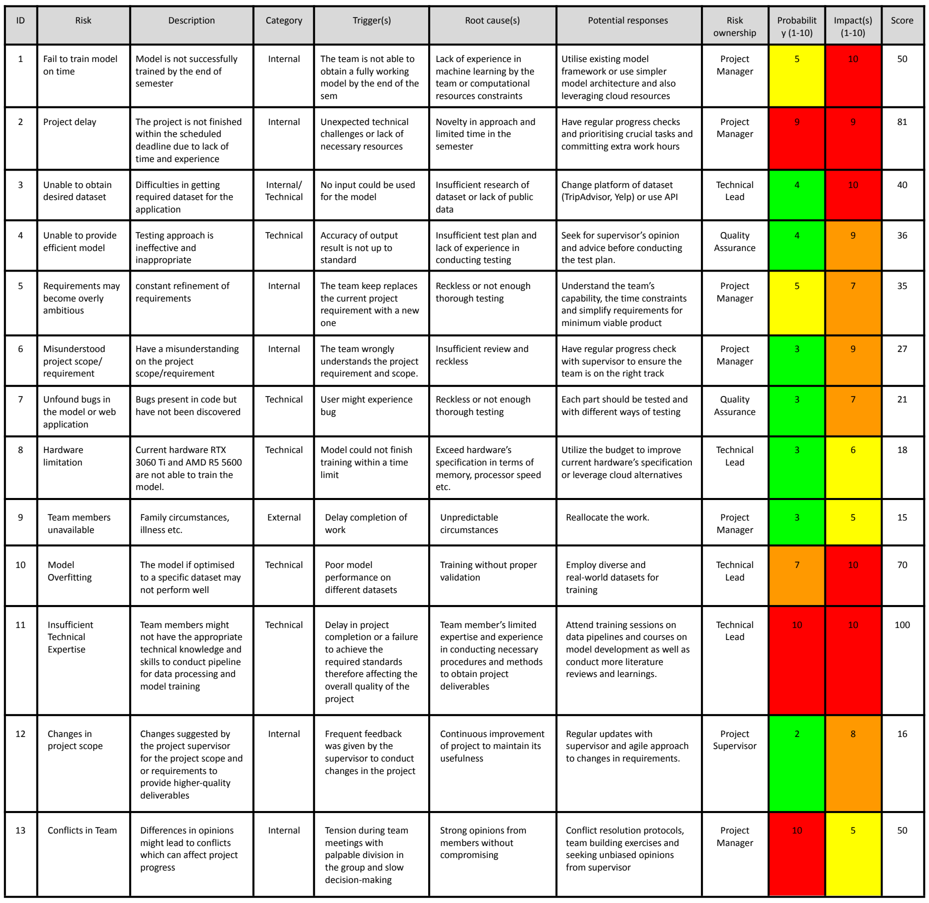
```
# Micro calculations for aspects
    micro_aspect_tp += len(aspect_matches)
    micro_aspect_fp += len(predictions) - len(aspect_matches)
    micro_aspect_fn += len(ground_truth) - len(aspect_matches)

# Micro metrics
micro_aspect_precision = micro_aspect_tp / (micro_aspect_tp + micro_aspect_fp) if micro_aspect_tp + micro_aspect_fp > 0 else 0
micro_aspect_recall = micro_aspect_tp / (micro_aspect_tp + micro_aspect_fn) if micro_aspect_tp + micro_aspect_fn > 0 else 0
micro_aspect_f1 = 2 * micro_aspect_precision * micro_aspect_recall / (micro_aspect_precision + micro_aspect_recall) if (micro_aspect_precision + micro_aspect_recall) > 0 else 0
```

Figure F.5: Screenshot of code for micro evaluation on aspect term



```
# Macro calculations for aspects
aspect_precision = len(aspect_matches) / len(predictions) if predictions else 0
aspect_recall = len(aspect_matches) / len(ground_truth) if ground_truth else 0
aspect_f1 = 2 * aspect_precision * aspect_recall / (aspect_precision + aspect_recall) if (aspect_precision + aspect_recall) > 0 else 0
macro_aspect_results.append((aspect_precision, aspect_recall, aspect_f1))

# Macro metrics
macro_aspect_precision, macro_aspect_recall, macro_aspect_f1 = zip(*macro_aspect_results)
macro_sentiment_precision, macro_sentiment_recall, macro_sentiment_f1 = zip(*macro_sentiment_results)
```

Figure F.6: Screenshot of code for macro evaluation on aspect term



ID	Risk	Description	Category	Trigger(s)	Root cause(s)	Potential responses	Risk ownership	Probabilit y (1-10)	Impact(s) (1-10)	Score
1	Fail to train model on time	Model is not successfully trained by the end of semester	Internal	The team is not able to obtain a fully working model by the end of the sem	Lack of experience in machine learning by the team or computational resources constraints	Utilise existing model framework or use simpler model architecture and also leveraging cloud resources	Project Manager	5	10	50
2	Project delay	The project is not finished within the scheduled deadline due to lack of time and experience	Internal	Unexpected technical challenges or lack of necessary resources	Novelty in approach and limited time in the semester	Have regular progress checks and prioritising crucial tasks and committing extra work hours	Project Manager	9	9	81
3	Unable to obtain desired dataset	Difficulties in getting required dataset for the application	Internal/ Technical	No input could be used for the model	Insufficient research of dataset or lack of public data	Change platform of dataset (TripAdvisor, Yelp) or use API	Technical Lead	4	10	40
4	Unable to provide efficient model	Testing approach is ineffective and inappropriate	Technical	Accuracy of output result is not up to standard	Insufficient test plan and lack of experience in conducting testing	Seek for supervisor's opinion and advice before conducting the test plan.	Quality Assurance	4	9	36
5	Requirements may become overly ambitious	constant refinement of requirements	Internal	The team keep replaces the current project requirement with a new one	Reckless or not enough thorough testing	Understand the team's capability, the time constraints and simplify requirements for minimum viable product	Project Manager	5	7	35
6	Misunderstood project scope/ requirement	Have a misunderstanding on the project scope/requirement	Internal	The team wrongly understands the project requirement and scope.	Insufficient review and reckless	Have regular progress check with supervisor to ensure the team is on the right track	Project Manager	3	9	27
7	Unfound bugs in the model or web application	Bugs present in code but have not been discovered	Technical	User might experience bug	Reckless or not enough thorough testing	Each part should be tested and with different ways of testing	Quality Assurance	3	7	21
8	Hardware limitation	Current hardware RTX 3060 Ti and AMD R5 5600 are not able to train the model.	Technical	Model could not finish training within a time limit	Exceed hardware's specification in terms of memory, processor speed etc.	Utilize the budget to improve current hardware's specification or leverage cloud alternatives	Technical Lead	3	6	18
9	Team members unavailable	Family circumstances, illness etc.	External	Delay completion of work	Unpredictable circumstances	Reallocate the work.	Project Manager	3	5	15
10	Model Overfitting	The model if optimised to a specific dataset may not perform well	Technical	Poor model performance on different datasets	Training without proper validation	Employ diverse and real-world datasets for training	Technical Lead	7	10	70
11	Insufficient Technical Expertise	Team members might not have the appropriate technical knowledge and skills to conduct pipeline for data processing and model training	Technical	Delay in project completion or a failure to achieve the required standards therefore affecting the overall quality of the project	Team member's limited expertise and experience in conducting necessary procedures and methods to obtain project deliverables	Attend training sessions on data pipelines and courses on model development as well as conduct more literature reviews and learnings.	Technical Lead	10	10	100
12	Changes in project scope	Changes suggested by the project supervisor for the project scope and or requirements to provide higher-quality deliverables	Internal	Frequent feedback was given by the supervisor to conduct changes in the project	Continuous improvement of project to maintain its usefulness	Regular updates with supervisor and agile approach to changes in requirements.	Project Supervisor	2	8	16
13	Conflicts in Team	Differences in opinions might lead to conflicts which can affect project progress	Internal	Tension during team meetings with palpable division in the group and slow decision-making	Strong opinions from members without compromising	Conflict resolution protocols, team building exercises and seeking unbiased opinions from supervisor	Project Manager	10	5	50

Figure F.7: Screenshot of updated Risk Register

Bibliography

[1] TripAdvisor Content API - Overview.

[2] Aspect-based-sentimentanalysis, 2020.

[3] kevinscaria/ate$_tk - instruct - base - def - pos - neg - neut - combinedHuggingFace, 8 2022.$

[4] Felipe Guimaraes E Equipe Aela. Nielsen039;s Heuristics: 10 Usability Principles To Improve UI Design - Aela School, 9 2022.

[5] Muhammad Ali, Ding Hooi Ting, Muhammad Ur Rahman, Shoukat Ali, Falik Shear, and Muhammad Mazhar. Effect of online reviews and crowd cues on restaurant choice of customer: Moderating role of gender and perceived crowding. *Frontiers in Psychology*, 12:780863, 12 2021.

[6] A. Almansour, R. Alotaibi, and H. Alharbi. Text-rating review discrepancy (trrd): an integrative review and implications for research. *Future Business Journal*, 8(1):3, 2022.

[7] A. Bhatia and B. Kaluza. *Machine Learning in Java: Second Edition Helpful techniques to design, build, and deploy powerful machine learning applications in Java*. Packt Publishing Ltd, 2018.

[8] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[9] Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. Recommender systems survey. *Knowledge-based systems*, 46:109–132, 2013.

[10] Geoffray Bonnin and Dietmar Jannach. Automated machine learning for recommendation. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 192–207. Springer, 2015.

[11] PETER BOSTRÖM and MELKER FILIPSSON. Comparison of User Based and Item Based Collaborative Filtering Recommendation Services. *KTH Royal Institute of Technology*, 2017.

[12] Laura Cabello and Uchenna Akujuobi. It is simple sometimes: A study on improving aspect-based sentiment analysis performance, 2024.

[13] Maryna Chernyshevich. IHS R&D Belarus: Cross-domain extraction of product features using CRF. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 309–313, Dublin, Ireland, August 2014. Association for Computational Linguistics.

[14] Lucas Colucci, Prachi Doshi, Kun-Lin Lee, Jiajie Liang, Yin Lin, Ishan Vashishtha, Jia Zhang, and Alvin Jude. Evaluating Item-Item Similarity Algorithms for Movies. *ACM Digital Library*, 5 2016.

[15] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[16] Diogo Cortiz. Exploring transformers in emotion recognition: a comparison of bert, distillbert, roberta, xlnet and electra, 2021.

[17] Hai Ha Do, PWC Prasad, Angelika Maag, and Abeer Alsadoon. Deep learning for aspect-based sentiment analysis: A comparative review. *Expert Systems with Applications*, 118:272–299, 2019.

[18] GeeksforGeeks. Cosine similarity, 7 2023.

[19] Xiaodong Gu, Yu Gu, and Haibing Wu. Cascaded Convolutional Neural Networks for Aspect-Based Opinion Summary. *Neural Processing Letters*, 46(2):581–594, 3 2017.

[20] Ali Issa. Transformer, GPT-3,GPT-J, T5 and BERT. - Ali Issa - Medium. 6 2023.

[21] Mayurdhvajsinh Jadeja. Jaccard Similarity made simple: a beginner's guide to data comparison. 3 2024.

[22] Harsh Jain. Netflix's hidden gems: Building a recommender system (season finale), 2023.

[23] Yohan Jo and John E Hopcroft. Aspect and entity extraction for opinion mining. In *Data Mining and Knowledge Discovery for Big Data*, pages 1–40. Springer, 2011.

[24] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer, 1998.

[25] Akbar Karimi, Leonardo Rossi, and Andrea Prati. Adversarial training for aspect-based sentiment analysis with bert. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 8797–8803, 2021.

[26] Harsh Khatter, Nishtha Goel, Naina Gupta, and Muskan Gulati. Movie recommendation system using cosine similarity with sentiment analysis. In *2021 Third International Conference on Inventive Research in Computing Applications (ICIRCA)*, pages 597–603, 2021.

[27] Ilya Krukowski. Web Scraping Tutorial Using Selenium Python (+ examples), 5 2024.

[28] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[29] Yu B. X. Li G. Li, H. and H. Gao. Restaurant survival prediction using customer-generated content: An aspect-based sentiment analysis of online reviews. 2023.

[30] Zheng Li, Xin Li, Ying Wei, Lidong Bing, Yu Zhang, and Qiang Yang. Transferable end-to-end aspect-based sentiment analysis with selective adversarial learning, 2019.

[31] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval.* Cambridge University Press, 2012.

[32] Andrew McCallum and Kamal Nigam. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48, 1998.

[33] T. K. Naab and A. Sehl. Studies of user-generated content: A systematic review. 18(10):1256–1273, 2017.

[34] Juri Opitz. From bias and prevalence to macro f1, kappa, and mcc: A structured overview of metrics for multi-class evaluation. 2022.

[35] Baris Ozyurt and M. Ali Akcayol. A new topic modeling based approach for aspect extraction in aspect based sentiment analysis: Ss-lda. *Expert Systems with Applications*, 168:114231, 2021.

[36] Josh Patterson and Adam Gibson. *Deep Learning: A Practitioner's Approach.* O'Reilly Media, Inc., 2017.

[37] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2023.

[38] Sebastian Ruder, Parsa Ghaffari, and John G. Breslin. A hierarchical model of reviews for aspect-based sentiment analysis, 2016.

[39] Kevin Scaria, Himanshu Gupta, Siddharth Goyal, Saurabh Arjun Sawant, Swaroop Mishra, and Chitta Baral. Instructabsa: Instruction learning for aspect based sentiment analysis, 2023.

[40] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

[41] Youwei Song, Jiahai Wang, Tao Jiang, Zhiyue Liu, and Yanghui Rao. Targeted sentiment classification with attentional encoder network. In Igor V. Tetko, Věra Kůrková, Pavel Karpov, and Fabian Theis, editors, *Artificial Neural Networks and Machine Learning – ICANN 2019: Text and Time Series*, pages 93–103, Cham, 2019. Springer International Publishing.

[42] Yohanes Sukestiyarno, Hasballah Sapolo, and Hizir Sofyan. Application of recommendation system on e-learning platform using content-based filtering with jaccard similarity and cosine similarity algorithms, 06 2023.

[43] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. *Proceedings of the IEEE international conference on computer vision*, pages 843–852, 2017.

[44] Siddharth Varia, Shuai Wang, Kishaloy Halder, Robert Vacareanu, Miguel Ballesteros, Yassine Benajiba, Neha Anna John, Rishita Anubhai, Smaranda Muresan, and Dan Roth. Instruction tuning for few-shot aspect-based sentiment analysis, 2023.

[45] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[46] Heng Yang, Chen Zhang, and Ke Li. Pyabsa: A modularized framework for reproducible aspect-based sentiment analysis, 2023.

[47] Yifei Yang and Hai Zhao. Aspect-based sentiment analysis as machine reading comprehension. In Nicoletta Calzolari, Chu-Ren Huang, Hansaem Kim, James Pustejovsky, Leo Wanner, Key-Sun Choi, Pum-Mo Ryu, Hsin-Hsi Chen, Lucia Donatelli, Heng Ji, Sadao Kurohashi, Patrizia Paggio, Nianwen Xue, Seokhwan Kim, Younggyun Hahm, Zhong He, Tony Kyungil Lee, Enrico Santus, Francis Bond, and Seung-Hoon Na, editors, *Proceedings of the 29th International Conference on Computational Linguistics*, pages 2461–2471, Gyeongju, Republic of Korea, October 2022. International Committee on Computational Linguistics.

[48] Yangyang Yu. Aspect-based Sentiment Analysis on Hotel Reviews. *Stanford University.*

[49] Dinghan Zhang, Xiaodong Han, Shuohang Wang, and Jun Jiang. Adversarial training for aspect-based sentiment analysis with bert. *Proceedings of The Web Conference 2020*, pages 2519–2525, 2020.

[50] Lei Zhang, Shuai Wang, and Bing Liu. Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1253, 2018.