TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND OF STUDY

In today's data-driven world, businesses are faced with a tremendous amount of textual data that needs to be analyzed and gain insights from to make informed decisions. According to a study by McKinsey & Company (2018), businesses that make decisions based on data-driven insights are 23 times more likely to get customers, six times more likely to keep customers, and 19 times more likely to make money than businesses that do not. Text classification has become an essential tool for analysing data, it is the task of assigning predefined categories or labels to a given text based on its content and context. This task has been widely studied in natural language processing (NLP) and has been used in many real-world applications such as spam filtering, document classification, and sentiment analysis. Nowadays, text classification has become an essential tool for businesses looking to make sense of large amounts of textual data.

The early development of text classification dates back to the 1960s when machine learning algorithms were first used to categorize documents. In the 2000s, the explosion of digital text data and the rise of the internet led to a significant increase in the interest and popularity of text classification. One of the example of text classification is a study by Turney (2002), where he proposed a method for using machine learning algorithms to classify movie reviews as either positive or negative. A study by Mikolov, Chen, Corrado, and Dean, (2015) proposed a novel method for automatically categorizing news articles into predefined topics using word embedding and clustering techniques.

An integral part of text classification is feature extraction, which aims to represent textual data in a meaningful way. One widely used technique is term frequency-inverse document frequency (TF-IDF), introduced by Karen Sparck Jones in the year (1972). Today, TF-IDF is one of the most widely feature extraction techniques used for text classification. It evaluates how important a word is to a document in a collection or corpus. This technique calculates a score for each word in a document, which is then used to determine the relevance of that word to the document's content. TF-IDF has been used successfully in various applications such as text classification, text clustering, and information retrieval (Kanaris, Stamatatos, & Fakotakis, 2020).

Word2Vec is an alternative feature extraction method used in text classification. It represents words as dense vectors in a high-dimensional space, considering the words contextual usage to capture semantic relationships between words (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013). This technique allows for the representation of each word in a text, wherein words with similar meanings have closer vectors in the vector space. By leveraging Word2Vec, text classification models can utilize these rich representations to better understand the underlying semantics of the text.

In contrast to traditional approaches like TF-IDF that primarily consider word frequency and significance, Word2Vec offers a more nuanced and contextually aware representation of textual data. Its ability to capture semantic information has made it a popular choice for feature extraction in various natural language processing applications, including text classification (Mikolov et al., 2013).

Once the features are extracted, classification algorithms are applied to learn patterns and make predictions. Classifiers employ a learning process to create a model that establishes the relationship between these features and the corresponding class labels. During the

training phase, the classifier endeavors to recognize patterns and associations within the features that signify particular classes or categories. This involves fine-tuning internal parameters or weights based on the training data, with the ultimate objective of minimizing errors or maximizing the model's predictive accuracy (Bishop, 2006).

There are several well-known classifiers commonly employed in text classification tasks. Logistic regression, for instance, models the connection between the extracted features and the probabilities associated with each class by employing a logistic function. Through estimating the parameters of this function, logistic regression can effectively classify new texts by calculating the likelihood of each class based on the acquired model (Raj, 2020).

Decision trees, on the other hand, provide another avenue for classification. They systematically split the feature space by utilizing specific criteria, such as entropy or information gain, to construct a hierarchical structure of decision rules. These rules enable the partitioning of data into subsets, and at each internal node, a decision is made based on the feature values to determine the appropriate path to follow. Ultimately, the leaves of the decision tree represent the predicted class labels for unseen texts (javaTpoint, 2021).

Support vector machines (SVMs) offer yet another approach to classification. Their primary objective is to identify an optimal hyperplane within the feature space that effectively separates different classes. SVMs achieve this by maximizing the margin between the hyperplane and the nearest instances of each class, facilitating a clear distinction between the classes. By leveraging kernel functions to map the textual data into a higher-dimensional space, SVMs can adeptly handle complex relationships and capture intricate decision boundaries (Sunil, 2019).

The choice of classifier depends on the specific task and dataset characteristics. Each

algorithm has its strengths and limitations, and researchers often experiment with different classifiers to find the most suitable one for a given text classification problem.

One specific area where text classification is crucial is in analyzing user reviews in the e-commerce industry, especially on platforms like the Google Play Store. Understanding these reviews is vital for developers and businesses to improve their applications and services. Apptentive (2019) emphasizes the importance of higher ratings, which have a positive impact on downloads and revenue. Therefore, it becomes imperative to analyze and categorize user reviews of Shopee, the leading e-commerce platform in Malaysia (Similarweb, 2023), which are posted on the Google Play Store. The significance of this analysis is particularly noteworthy in Malaysia, where Shopee holds the distinction of being the top shopping app on both the Google Play Store and the Apple App Store, as reported by iPrice Group (2021) and App Annie (2021).

Hu et al. (2019) conducted a study in which the researcher categorized customer reviews into three main aspects: e-commerce service, customer service, and application performance. Their research highlighted the importance of analyzing customer feedback in these specific areas. Furthermore, another relevant paper by Rajendran (2021) focused on the courier and delivery service industry, specifically addressing the challenges faced by courier companies in terms of customer ratings and the significance of employee commitment for customer loyalty.

Building upon the insights from these two papers, this research aimed to conduct text classification based on three classes of customer reviews: application performance, customer service and delivery service. Analyzing customer reviews in relation to these three classes provides insights into the functionality and usability of e-commerce application, while evaluating reviews concerning delivery and customer service. By

analyzing and categorizing user reviews on the Google Play Store, businesses and developers can gain valuable insights into the user experience of the Shopee application. This enables them to identify common issues and areas for improvement, taking actionable steps to enhance the application further.

The benefits of text classification in the big data world are vast. By categorizing large amounts of textual data, businesses can identify patterns, trends, and insights that would otherwise be impossible to detect. This can help businesses make informed decisions and enhance their customer experience, according to a study by Duan, Gu, and Whinston (2008).

However, it is important to note that there are also potential negative consequences of text classification. A study by Kim and Yoon (2020) argues that text classification algorithms may perpetuate existing biases and stereotypes present in the data, leading to discrimination against certain groups. Additionally, the authors argue that text classification may oversimplify complex issues, leading to inaccurate or incomplete conclusions. By categorizing customer feedback into specific categories like delivery service, customer service, and application performance, businesses can identify areas for improvement and enhance their offerings. TF-IDF and Word2Vec are two techniques that can help businesses analyze large amounts of textual data and make informed decisions. By using these techniques, businesses can gain insights into customer sentiments and preferences, improve their offerings, and stay ahead of the competition.

1.2 PROBLEM STATEMENT

The problem of text classification in the context of e-commerce is a complex one that has important implications for businesses, consumers, and society as a whole. In the digital

age, the growth of e-commerce has led to an increasing volume of customer reviews and social media mentions, providing valuable insights into public sentiment towards products and services.

One of the best ways ensure the successs of e-commerce business is to better serve customers and hearing them out when they have concerns or suggestions and improve their services accordingly. Due to this, Shopee must analyze thousands of complaints every day.

Despite the dire need for an automated text classification, the key problem in natural language processing (NLP) is choosing the best feature extraction method that can be used to allow machines to automatically classify textual material into preset classes or categories. Feature extraction methods include the term frequency-inverse document frequency (TF-IDF) and Word2Vec approach.

While both methods have been widely utilised in text categorization tasks, there has been a lack of studies comparing their performance of both method. As a result, the efficiency of Word2Vec and TF-IDF in diverse text classification applications such as sentiment analysis, topic labelling, and document classification must be evaluated and compared. Previous study and innovations in text classification has also yet to address sample study of Shopee user in Malaysia specifically, using the method of TF-IDF and Word2Vec.

1.3 RESEARCH OBJECTIVES

The following are the research objectives of this study:

1. To evaluate the performance of TF-IDF as a feature extraction technique in text classification of Shopee user reviews on Google Play Store.
2. To evaluate the performance of Word2Vec as a feature extraction technique in text classification of Shopee user reviews on Google Play Store.
3. To determine the best performance between TF-IDF and Word2Vec feature extraction techniques in text classification.

1.4 RESEARCH QUESTIONS

The following are the research questions of the study

1. What is the performance of TF-IDF as a feature extraction technique in text classification?
2. What is the performance of Word2Vec as a feature extraction technique in text classification?
3. Which feature extraction technique, TF-IDF or Word2Vec, yields better performance in text classification tasks?

1.5 SIGNIFICANCE OF STUDY

The findings of this research may not only assist e-commerce business by helping them boost their customer service, but also enhance customer's experience to a greater scale. The benchmark for customer's experience may be raised across the board for all types of businesses and services. When applied to other types of services, such as government-run public and education services, the same model and methods of text classification may

provide even greater improvements in the public's satisfaction level. Public satisfaction in the government services is important to maintain the public's loyalty to the government and ensure the success of government policies.

This research findings would assist other researchers in determining the best strategy for future specific text classification task, taking into account criteria like accuracy, efficiency, and scalability. Furthermore, by highlighting the strengths and drawbacks of each method, this study can help to advance the field of natural language processing by leading to the development of more effective and accurate text classification algorithms.

1.6 SCOPE AND LIMITATION OF STUDY

The scope of this research is 2309 textual reviews ranging from the year 2021 to 2023 of Shopee application made by the users on Google Play Store review from Malaysia. Some of limitations of this research are the lack of computational resources, there is limited access to some software features to process and analyze data due to lack of funding. The second limitation of the study is analyzing large volume of data manually is a time-consuming and resource-intensive task, and there is a need for automated methods to quickly and accurately gauge public sentiment. The third limitation is the unavailability of training data, the researcher has to manually label 2309 of user reviews, this data may be affected by researcher's bias and interpretation during manual labelling. Lastly, due to under a tight time constraint, an in depth study that covers all aspect of the two text classification method is infeasible.

CHAPTER 2

LITERATURE REVIEW

In this section, pieces of literature that are related to text classification is discussed. Literature review is intended to demonstrate the familiarity with and comprehension of the body of academic research available on a particular subject when it is put into perspective.

2.1 TRAINING SET SIZES

A training set is a subset of a larger data collection that is used to fit or train a model for predicting or classifying values that are known in the training set but unknown in other future data. The training set is used in conjunction with the validation and/or test sets to evaluate various models. In the context of text or document classification, training set is a set of tagged/categorized text (Gupta, 2018). The purpose of a machine learning training set is to provide labeled examples or data for a machine learning algorithm or model. The training set serves as the foundation for the learning process, allowing the model to understand underlying patterns, relationships, and structures in the data (Rouse, 2023). In this study, the model analyzes the input features in the training set of manually annotated reviews under three classes which is delivery service, customer service, and application performance to learn the relationship between them.

As the training sample size (n) increases, the variance tends to decrease (Beleites, 2013). In ensuring minimal variances and biases in text classification, a careful assessment of the required amount of training data becomes imperative for text classification tasks. The number of training data needed varies according to the use case and desired model performance. In a blog post by (Wolff, 2020) it is generally advised to have at least

500-1,000 samples per tag for sentiment analysis and 100-300 samples per tag for topic classification. The author emphasised that the more training data there is, the more accurate a certain model becomes. (Wolff, 2020). In another blog post by (Dorfman, 2022), the author elaborated on the "10 times rule", which is a general rule that states that a data set should be at least ten times larger than the number of parameters or degrees of freedom in a model. The rule offers a preliminary estimate to guarantee that enough data is available for model training. To capture the underlying patterns and generate trustworthy classification findings, a suitably representative and balanced dataset across all classes is recommended.

2.2 FEATURE EXTRACTION

The process of turning raw textual data into a collection of numerical characteristics that may be utilised as input for machine learning algorithms is known as feature extraction in natural language processing (NLP). It entails choosing and presenting pertinent data from text in a structured style that computational models can comprehend and use (Wei, 2019). Feature extraction tackles the challenge of finding the most useful and informative features. Creating feature vectors is the typical and convenient approach for representing data in classification and regression tasks (Dongyang Wang, 2020).

In the topic of text classification, two commonly used feature extraction techniques are Word2Vec and TF-IDF. Word2Vec leverages word embeddings to capture semantic relationships and contextual information, while TF-IDF assigns importance scores to words based on their frequency in a document collection.

TF-IDF as a feature extraction

TF-IDF, which is an acronym for Term Frequency and Inverse Document Frequency, is a technique used to identify the essential words or the uncommon words in the text data. TF-IDF as a feature extraction technique, converts strings to numbers for machine learning models and classifiers can ingest the data in numerical formats (Kulshrestha, 2020). To demonstrate how widely use TF-IDF as feature extraction technique, according to a 2015 survey, TF-IDF is used by 83% of text-based recommender systems in digital libraries to extract textual properties (Beel, Gipp, Langer, & Breitinger, 2016).

The simplicity and ease of usage of TF-IDF are its major benefits. It is inexpensive to compute and easy to understand. TF-IDF also efficiently handles insignificant and recurring words (e.g., stopwords) in natural languahe by giving the terms low score due to their frequent occurrence in documents. Making space to significant terms to have higher TF-IDF scores and more discriminative strength when separating documents according to Darla M (2021).

Nevertheless, TF-IDF, being an unsupervised feature extraction technique, has been criticized for its limitations in indicating the relevance of words within specific classes and its focus limited solely to individual documents. The Inverse Document Frequency (IDF) has a discriminatory power that disregards a word that appears more frequently in documents and the document corpus as a whole. IDF assigns a weight of 0 to a word if it appears in every document, IDF does not consider words that are significant to be particularly important if it is recurring, since it does not capture the semantic value of words. According to Ahuja (2020), an ideal approach for IDF (Inverse Document Frequency) would consider words that occur more frequently in documents, allowing it to differentiate itself from other words in the cluster.

Word2Vec as a feature extraction

Word2Vec is a technique for representing words as vectors of numbers. It allocates a unique vector to each word, capturing characteristics including meaning, context, and relationships with other words in the text. These vectors are learnt through a neural network-like process. Word2Vec identifies similarities and associations between terms by analyzing massive quantities of text data (Vatsal, 2022).

Word2Vec's strength rests in its ability to group together comparable terms. It can make accurate estimates of a word's meaning based on its occurrence in text when given a large dataset. For instance, "King" and "Queen" would have comparable vector representations. By executing algebraic operations on word vectors, such as subtracting "man" and adding "woman" to "king," we can approximate the vector representation of "queen." (Logunova, 2023). This numerical representation of words enables a number of applications, including measuring word similarity and discovering word associations.

Word2Vec does have some limitations that must be considered. The inability to effectively manage out-of-vocabulary (OOV) words is a notable deficiency. Due to the fact that Word2Vec is pre-trained on a fixed vocabulary, it may struggle to generate meaningful representations for terms not present in the training data. This limitation becomes more pronounced when dealing with domain-specific or uncommon words that the training corpus may not have adequately captured (Levy, Goldberg, & Dagan, 2015).

2.3 PERFORMANCE OF DIFFERENT FEATURES EXTRACTION TECNIQUES AND CLASSIFIERS

One of the most notable research in comparing feature extraction techniques is an article by Cahyani and Patasik, (2021).This article compares the efficacy of TF-IDF and Word2Vec models for emotion text classification. The purpose of this study is to classify emotional text data from commuter queue and transjakarta tweets using Support Vector Machine (SVM) and Multinomial Nave Bayes (MNB) techniques. The process of classification involves identifying the presence of emotion and classifying it into five categories: happy, angry, sad, frightened, and surprised. This paper evaluates three scenarios: SVM with TF-IDF, SVM with Word2Vec, and MNB with TF-IDF, with SVM and TF-IDF yielding the highest classification accuracy in both phases.

The article findings conclude that SVM and TF-IDF feature extraction technique outperform other methods in terms of precision, recall, and F1-measure. Notably, the SVM with TF-IDF method excels at recognising all five categories of emotions, including the difficult emotion of surprise, whereas other methods fail. Based on precision, recall, and F1-measure values, the TF-IDF model's succeeded in identifying every category of emotion. The superiority of the TF-IDF model over Word2Vec can be attributed to the unbalanced distribution of data among the emotion classes. Word2Vec relies on large training datasets to effectively extract semantic and syntactic information, which may explain its limitations with respect to lesser datasets. In contrast, the TF-IDF model is accurate despite having limited data. This study contributes to the advancement of the evaluation of classification accuracy and performance in emotion text classification. The finding highlight the importance of using appropriate feature extraction techniques in emotion text classification.

The paper "Term Weighting for Feature Extraction on Twitter: A Comparison Between BM25 and TF-IDF" by Kadhim, (2019), delves into extracting meaningful features from Twitter data. Specifically, it explores the effectiveness of two term weighting schemes of BM25 and TF-IDF. The findings of this study reveal that BM25 tends to outshine TF-IDF when it comes to extracting features from Twitter data. BM25, with its nuanced consideration of term frequency, document length, and document frequency, proves to be a more effective measure in capturing the relevance and significance of terms within tweets. In contrast, TF-IDF, while valuable in its own right, relies solely on term frequency and inverse document frequency, potentially overlooking crucial nuances and intricacies that characterize Twitter's unique language and context. In essence, the paper concludes that BM25 emerges as the preferred term weighting scheme for feature extraction on Twitter. Its ability to discern the relevance and importance of terms more accurately makes it a better fit for the dynamic nature of Twitter data.

In a journal article by (Pradhan, Taneja, Dixit, & Suhag, 2017), the researchers tested the performance of different classifier algorithm with varying degrees of efficiency and speed. The purpose of this study is to evaluate the classification efficacy of Support Vector Machine (SVM), Naive Bayes, Decision Tree, k-nearest neighbor (kNN) and Rocchio classifiers to news article. In order to effectively train the models and avoid over-fitting, the data set is separated into training and testing data. Next, the textual data is normalised and converted into a numerical representation using TF-IDF feature extraction method and Chi-Square test to determine the most informative features before testing the performative measure of the classifiers.

The results suggest that linear SVM obtains the highest accuracy in classifying news articles, followed closely by Naive Bayes. The Decision Tree classifier requires the longest training time, whereas SVM and the other classifiers have significantly reduced

training and testing times. Among the classifiers, the kNN classifier is the most efficient in terms of testing time, as it requires the fewest computations during the testing phase. In spite of this, the researchers recommend using linear SVM for greater accuracy, whereas Naive Bayes provides a viable alternative with less time complexity but with the expense of poor accuracy. The findings of this study demonstrate the superiority of linear SVM for accurately classifying news articles. The researcher suggested the performance of these classifiers to be evaluate on a broader range of dataset in order to validate the findings and evaluate their applicability in various contexts.

Finally, a literature review by Wang, Fu, Sui, and Ding, in (2015) aims to evaluate the performance of four popular classification algorithms: Decision Tree, Naive Bayes, Maximum Entropy, and K-nearest neighbor. The experiments conducted in this study focused on classifying movie reviews using the NLTK (Natural Language Toolkit) in Phyton. The accuracy and stability of each algorithm were compared to determine their effectiveness. Based on the results, the Naive Bayes classifier demonstrated the highest accuracy among the four algorithms, consistently achieving over 80% accuracy. It also exhibited good stability, making it a reliable choice for movie review classification. The K-nearest neighbor classifier performed relatively well with an average accuracy of around 73%, but it was not as stable as Naive Bayes. On the other hand, the Decision Tree classifier showed the highest stability but had an average accuracy of approximately 65%. The Maximum Entropy classifier, unfortunately, yielded the lowest accuracy and stability in this experiment. The conclusion of the review notes that The Naive Bayes algorithm emerges as the top performer in terms of accuracy, while the K-nearest neighbor algorithm demonstrates faster classification. The Decision Tree algorithm, although stable, lags behind in terms of accuracy.

2.4 APPLICATION OF TEXT CLASSIFICATION IN REAL LIFE

This section reviews some of the research that makes use of text classification so that the applications and impacts of text classification in actual situations may be observed.

Research paper by Badjatiya, Gupta, Gupta, and Varma (2017) focuses on identifying hate speech on Twitter, the objective was to evaluate the performance of applications of deep neural network architecture in hate speech detection to classify if a tweet was racist, sexist, or neutral. The identification of such hostile speech is critical for conducting public sentiment analysis on one group's attitude toward another group for preventing wrongdoing that is related with such negative speech. The techniques used embedding acquired from deep neural network models in conjunction with gradient boosted decision trees. The research suggested that use of deep neural network architectures for detecting hate speech is an essential step in contentious event extraction, the construction of AI chatterbot and content suggestion.

In another study by Del Vigna, Cimino, Dell'Orletta, Petrocchi, and Tesconi (2017), aims to restrict and prevent the frightening spread of hate campaigns on certain individual or establishment. Using Facebook as a standard, the researcher evaluates the linguistic content of comments on a collection of public Italian pages. The researches suggested a range of hate categories to differentiate between different types of hatred, up to five independent human annotators. Using morpho-syntactic features, sentiment polarity, and word embedding lexicons, the researchers designed and implement two classifiers for the Italian language based on different learning algorithms which is based on Support Vector Machines (SVM) and Recurrent Neural Network named. The findings demonstrate the efficacy of the two classification algorithms evaluated on the first manually annotated social media corpus of Italian hate speech.

In another research by Yu, Zhou, Zhang, and Cao (2017), an efficient SVM model was created for distinguishing between positive and negative sentiment in Yelp reviews, with an accuracy of 88.906% on the test set. Aside from gathering keywords from various cuisines, the model can also be used to automatically generate ratings from tips (brief reviews without ratings) on Yelp by assigning weights to tips based on the emotion score of phrases, resulting in more acceptable total ratings for restaurants.

Next, the implementation of sentiment analysis in Feine, Morana, and Gnewuch (2019) is used to measure service encounter satisfaction with customer service chatbots. An online service interaction with a chatbot demonstrated that sentiment scores may be used as proxies for Code Submission Evaluation System (CSES). This allows online service providers to obtain user information objectively and automatically during and after an online service interaction. This information may be utilized not only to initiate service recovery operations, but also to detect service quality problems and conduct real-time user analysis. Consequently, the findings lead to the development of user-adaptive service chat bots.

2.5 SUMMARY

In summary, text classification has emerged as a powerful tool that finds applications across diverse domains. Its effectiveness has been evident in numerous areas such as information retrieval, where it aids in organizing and retrieving relevant documents quickly. Sentiment analysis leverages text classification to gauge public opinion and sentiment towards products, services, or events. News classification benefits from this technique to automatically categorize news articles, making it easier for users to access the information they seek.

Moreover, text classification plays a crucial role in customer feedback analysis, where it enables businesses to efficiently analyze vast amounts of customer reviews and comments, gaining valuable insights into their products and services. This understanding empowers businesses to make data-driven decisions, improve customer satisfaction, and refine their offerings.

The versatility and widespread applicability of text classification make it a vital asset in modern data-driven environments. As technology advances, the potential applications of text classification continue to expand, making it an indispensable tool for extracting valuable knowledge from the ever-increasing volumes of textual data available in the digital era.

CHAPTER 3

METHODOLOGY

3.1 INTRODUCTION

This section describes in detail how the textual reviews of Shopee application user was collected and utilized for analysis to answer the research questions. This chapter demonstrate the techniques that has been used to analyze the data. The workflow begins with data gathering, data preparation and data analysis. First, the data gathering process starts with Python scraping of Shopee application reviews from the Google Play Store. The obtained data includes textual reviews, the number of 'thumbs up' votes, and other relevant metadata. Then, a manual labeling procedure is initiated in which 3000 reviews are assigned to three distinct categories: "delivery service," "customer service," and "application performance."

The reviews are then subjected to normalization procedures to facilitate reliable and consistent analysis. This includes spelling correction, conversion to lowercase, and removal of white space. In addition, standard text pre-processing procedures such as stop word removal, stemming, and lemmatization are implemented. After the data have been preprocessed, analysis begins. To accurately represent textual information, the TF-IDF and Word2Vec feature extraction methods are utilized. The extracted and preprocessed features are then used to train logistic regression, linear SVM, and decision tree classifier algorithms. The objective is to appropriately classify the textual reviews of Shopee application user and provide useful information regarding the comparative performance of the two feature extraction methods.

The Figure 3.1 shows the workflow of the research consisting of the first step which is data gathering, data preparation and data analysis.

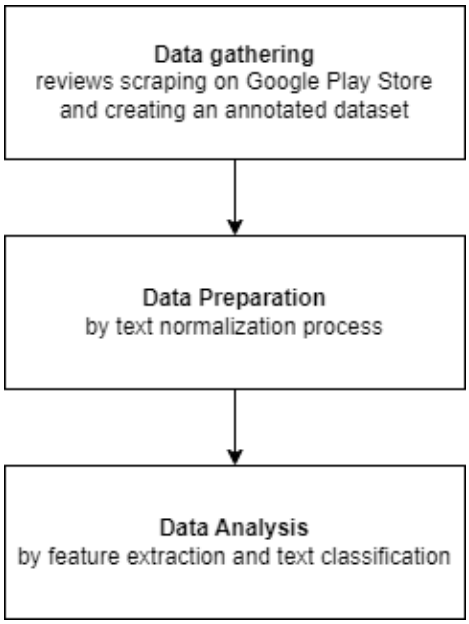

Figure 3.1: Research flowchart

3.2 DATA SCRAPING

3.2.1 Extraction of Shopee Reviews on Google Play Store

User reviews of Shopee mobile application from the year 2021 to 2023 is scraped from Google Play Store using the google_play_scraper library in Python. The scraping criteria is define by a target app package name of Shoopee (com.shopee.my), the language of English (en), the country of Malaysia (my), the sorting order of newest (Sort.NEWEST), and the desired number of reviews. The scraped reviews are then stored in a CSV file in write mode and uses the csv.writer to write the review data into the file. The CSV file is structured with headers: 'Text', 'Date' and 'Thumbs Up Count'. Each review is iterated through, and the relevant information (user name, review text, date, score, and thumbs-up count) is extracted and written as a row in the CSV file.

3.2.2 Creating an annotated data set

The first step in training a machine to carry the task of text classification is creating an annotated dataset. The process involves manually labelling 2309 text reviews into the class of customer service, delivery service and application performance. There are three steps of labelling textual reviews which is:

1. **Define label categories:** Determining the labels that is assigned to textual reviews. In this research, the defined labels are delivery service, customer service and application performance.

2. **Create a Labeling System:** Establish guidelines or criteria for assigning labels to the text samples. Ensuring consistency and clarity in the labeling process to maintain the quality and reliability of the labeled data set.The guidelines used are:

    - **Customer Service:** This label applies when the review specifically mentions the quality of customer service provided by the company or its representatives. Examples: “Excellent customer support,” “Unhelpful customer service,” “Quick response from support team.”

    - **Delivery Service:** This label applies when the review discusses the delivery process, shipping speed, or overall satisfaction with the delivery service. Examples: “Fast delivery,” “Delayed shipment,” “Damaged package upon arrival.”

    - **Application Performance:** This label applies when the review focuses on the performance or functionality of the company’s application, website, or digital platform. Examples: “App crashes frequently,” “Smooth user experience,” “Slow loading times.”

3. **Label the Text Samples:** Go through each text sample and assign the appropriate label based on its content and context.

Table 3.2 shows example of Shopee user application reviews and its respective class label, the table consist of three reviews that is assigned to different label which is application performance, delivery service and customer service. The reviews are labelled based on the labelling system created.

Table 3.2: Example of Shopee user reviews and its respective class label

Reviews	Label
Often times the app becomes unresponsive and laggy.	Application performance
Shopee express does not care to deliver the item on time.	Delivery service
I rarely buy anything here since the customer service is not very helpful.	Customer service

3.3 DATA PREPARATION

In this section, the steps needed to prepare the scraped data for text categorization process into the class of delivery service, application performance and customer service is discussed. The steps of text classification is done through the use of Phyton language in Pycharm environment, and the steps written in the environment is discussed thoroughly in the next subsection.

3.3.1 Data Split

The original dataset that has been annotated is divided into input features (X) and corresponding labels (y). The dataset is then split into two parts - a training set and a test set. The purpose of this split is to train the model on the training data and evaluate its performance on the test data. 20% of the data is used for testing. The test set obtained

from the previous step is further divided into two parts which are validation set and a final test set. 50% of the remaining test data is used for validation, and the remaining 50% is be used for the final test. The validation set is used for tuning the model's hyperparameters and selecting the best configuration while the final test set is kept separate and is used to evaluate the model's performance after all the hyperparameter tuning. Finally, separate datasets are created for the training, validation, and test sets, which include both the input features and their corresponding labels.

By performing these splits, the model is trained on one set of data, tuned on another set, and evaluated on a completely independent set to get an unbiased assessment of its performance.

3.3.2 Text Normalisation

Text normalization in text classification refers to the process of transforming raw text data into a standardized format to improve the accuracy and effectiveness of classification models. It is needed because text data often contains inconsistencies, variations, and noise, which can negatively impact the performance of classification algorithms. Text normalization involves techniques such as converting text to lowercase, removing punctuation, stemming or lemmatization, removing stop words, and handling numerical and special characters. By normalizing the text, we reduce the dimensionality of the data, eliminate irrelevant details, and create a more consistent representation, which helps in improving the efficiency and accuracy of text classification models. In this subsection all of the steps in the text normalisation process is discussed.

Conversion to lowercase

The conversion of reviews to lowercase is the first step in text normalization, it helps achieve consistency and standardization in the text data. By transforming all uppercase characters to their lowercase form, the distinction between uppercase and lowercase characters is eliminated. This ensures that the same word is not treated differently based on its capitalization.“Shopee” and “shopee” would be considered as the same word after converting to lowercase. Additionally, converting text to lowercase reduces the overall vocabulary size by treating words with different capitalizations as identical.

Input	“I enjoy using Shopee as MY NUMBER 1 online shopping platform!”
Ouput	“i enjoy using shopee as my number 1 online shopping platform!”

Removal of whitespaces and punctuations

The removal of whitespace and punctuation aimed at eliminating extraneous gaps and punctuation marks within a text. By removing additional spaces, such as multiple consecutive spaces, leading spaces at the beginning, and trailing spaces at the end of the text, this ensure a more uniform and standardized representation. Similarly, the removal of punctuation, including dots, quotation marks, and exclamation marks, contributes to a cleaner and more structured text. This process minimizes potential errors in language processing tasks, such as tokenization or part-of-speech tagging.

Input	“ Shopee is a good application.”
Ouput	“Shopee is a good application”

Removal of Unicode Character

Unicode characters are characters that are outside the standard ASCII range and include symbols, emojis, foreign language characters, and other special characters. Removing Unicode characters helps to simplify the text and ensure compatibility with text

classification algorithms that may not handle these characters well. By eliminating Unicode characters, the focus is then shifted on the core textual content, reducing noise and potential inconsistencies in the data. This process is particularly important since the text classification process involves working with text data from Google Play Store reviews where the presence of Unicode characters can be more common. By applying this normalization step, the text becomes more streamlined and suitable for further analysis.

Input	"Shopee Xpress lost my parcel ☹"
Ouput	"Shopee Xpress lost my parcel"

Substituition of Acronyms

Text normalization requires the replacement of acronyms with their corresponding expanded forms or meanings. Acronyms are frequently used in text communication and can make it difficult to comprehend the text, particularly if they are unfamiliar or ambiguous. To facilitate the substitution of acronyms, a predefined list of acronyms and their expansions is used. The "english_acronyms.json" file containing a list of English abbreviations and their expanded forms is used. This file is a guide for mapping acronyms to their expanded forms. When an acronym is encountered during the text normalization procedure, it is searched up in an acronym dictionary. If a match is discovered, the acronym is replaced with its expanded form. This modification improves comprehension, making it more accessible for analysis and interpretation. Table 3.3 shows example of acronyms and its full word form.

Table 3.3: Examples of acronym and its full form

	Acronyms	Full form
1	imo	in my opinion
2	tbh	to be honest
3	idc	i don't care

Stop Words Removal

Stop words are the words that are often filtered out prior to processing a natural language. These are the most frequent words in every language (articles, prepositions, pronouns, conjunctions, etc.) that contribute little to the text's meaning. Examples of English stop words include "the," "a," "an," "so," and "what".

Input	"My Shopee experience before was not great because of the excessive use of pop-ups ads and slow loading time."
Output	"Shopping experience online not great excessive use pop-up ads slow loading times."

Spelling correction

Spell checking plays a crucial role in text normalisation to ensure the accuracy and dependability of the processed text data. Spelling errors, can be caused by typos, accidental blunders, or variations in writing styles. Spell checking improves the text's legibility and comprehension. In text classification, misspelling of reviews can lead to confusion and misinterpretation. By utilizing the spellchecker library in Phyton, spelling errors in a text can easily be detected and corrected.

Input	"I enjy my Shopee xperience."
Ouput	"I enjoy my Shopee experience"

Lemmatization

Lemmatization are used in this text classification process to reduce word variations and simplify text data. Lemmatization seek to transform words into their root or base forms, this is a linguistically precise process that employs dictionaries and morphological analysis to reduce words to their lemma, or root form. Lemmatization takes into account the word's context and part of speech, ensuring valid base forms. For instance, "shopping" would be lemmatized to "shop," which is a valid English lemma.

Input	“I had a terrible shopping experience online before. The products were not as described, and the customer service was unhelpful
Ouput	“Terrible shopping experience online. Product not as describe, customer service unhelpful.“

Discarding non-alphabetic words

Removing non-alphabetic words makes focusing on reviws's relevant details easier. Typically, words containing numbers, symbols, or other non-alphabetic characters do not provide meaningful insights and can introduce noise into an analysis. Eliminating non-alphabetic words also assists in attaining text data consistency and standardisation. It minimizes variations and inconsistencies that could result from the inclusion of non-alphabetic elements. Moreover, the algorithms for text classification are designed to operate specifically with alphabetic words.

Input	“I frequently shop at Shopee, like 1000 times.”
Ouput	“I frequently shop at Shopee, like times.”

Retainment of relevant part of speech (POS)

Retaining the grammatical category of words in a text involves the preservation of part of speech (POS). Regular expressions (regex) can be used to identify and extract specified portions of speech. Regex patterns can be designed to match and capture specific patterns associated with various elements of speech, including nouns, verbs, adjectives, and adverbs. By applying these patterns, words that belong to a specific element of speech can be retained, while discarding the remainder. For this text normalization process, the implementation of the *keep_pos* function under the Natural Language Toolkit (NLTK) library is utilized. Its functionalities of part-of-speech tagging which assigns grammatical categories to words in a text is perfect for the aim of retaining the relevant part of reviews. Text normalization benefits from retaining parts of speech by maintaining the text's

syntactic structure and preserving the relationships between words and their grammatical functions.

As shown in Table 3.4, the POS keep tags include various part of speech categories such as nouns, pronouns, adverbs, and verbs that are kept in the process of text normalisation.

Table 3.4: Descriptions and Tags for POS Keep Tags

Description	**Tag**
Noun, singular or mass	NN
Noun, plural	NNS
Proper noun, singular	NNP
Proper noun, plural	NNPS
Foreign word	FW
Personal pronoun	PRP
Possessive pronoun	PRPS
Adverb	RB
Comparative adverb	RBR
Superlative adverb	RBS
Verb, base form	VB
Verb, past tense	VBD
Verb, gerund or present participle	VBG
Verb, past participle	VBN
Verb, non-3rd person singular present	VBP
Verb, 3rd person singular present	VBZ
Wh-determiner	WDT
Wh-pronoun	WP
Possessive wh-pronoun	WPS
Wh-adverb	WRB

Table 3.5 shows an example of how POS tagging work. The word 'I' has the POS tag of Personal pronoun (PRP), the word 'love' has the POS tag of Verb (VBP), while 'online' and 'shopping' has POS tags of Noun (NN).

Table 3.5: POS Tags for the Sentence “I love online shopping”

Word	POS Tags
I	PRP
love	VBP
online	NN
shopping	NN

3.4 DATA ANALYSIS

3.4.1 Descriptive Analysis

The purpose of descriptive analysis, is to summarise or characterise a collection of data by the use of statistical methods. The ability of descriptive analysis to provide understandable insights from data that would otherwise be uninterpreted is one of the primary reasons for its extensive use (Bush, 2020). The comparison of class frequencies, the distribution number of characters and the average word length of each class reviews are analysed.

3.4.2 Feature extraction

There are two methods of feature extraction that is used in this research, which is TF-IDF and Word2Vec. Feature extraction is a crucial step in text classification that entails transforming unstructured text data into a numerical representation. It seeks to identify the key characteristics of the text that can be used to differentiate between distinct classes or categories. These extracted features serve as input for the classifier, allowing it to discover patterns and make predictions based on the extracted features. Text classification relies on feature extraction because it enables the model to comprehend and interpret textual data effectively.

TF-IDF method

TF-IDF is a feature extraction method used in this text classification tasks of three classes which is application performance, delivery service, and customer service. It aims to capture the importance of words within each class by considering both the term frequency (TF) and the inverse document frequency (IDF).

TF measures how frequently a word appears in a specific reviews. It is calculated by dividing the frequency of a term (word) in a review by the sum of frequencies of all terms in that class. The equation for TF is given by:

$$TF(t,d) = \frac{f_{t,d}}{\sum_{w \in d} f_{w,d}} \tag{3.1}$$

where $f_{t,d}$ represents the frequency of term t in review d, and the denominator is the sum of frequencies of all terms in class d.

IDF, on the other hand, measures the rarity of a term across the entire class. It is calculated as the logarithm of the total number of documents in the class divided by the number of reviews that contain the term. The IDF equation is:

$$IDF(t,D) = \log \frac{N}{DF_t} \tag{3.2}$$

where N represents the total number of reviews in the class, and DF_t is the number of reviews in the class that contain the term t.

By combining TF and IDF, we obtain the TF-IDF weight, which reflects the importance of a word in a specific class. The TF-IDF score is calculated by multiplying the TF and IDF values:

$$TF-IDF(t,d,D) = TF(t,d) \times IDF(t,D) \tag{3.3}$$

TF-IDF assigns higher weights to words that are frequent within a specific reviews (high TF) and rare across the entire class (high IDF). These tend to be the most informative words for classification.

Using TF-IDF as a feature extraction method for three classes like application performance, delivery service, and customer service allows us to capture the specific importance of words within each class. It enables the classification model to focus on the most relevant words for each class, improving the accuracy of the classification task.

Word2Vec method

Word2Vec is a word embedding technique that learns vector representations of words based on their contextual relationships in a given corpus. The key idea behind Word2Vec is that words appearing in similar contexts tend to have similar meanings. This property allows Word2Vec to capture semantic relationships between words, such as analogies. In this research Word2Vec is employed to extract meaningful features for text classification. By using an already trained Word2Vec model, word embedding that encodes the semantic properties of words related to each category can be obtained. These embeddings can then be used as input features for the text classification.

Figure 3.3 shows the representation of semantic relationship of words in vector space.

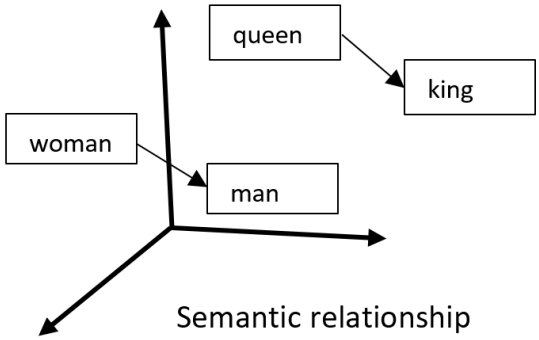

Figure 3.3: Demonstration of semantic relationship in Word2Vec

(Source : Zvornicanin(2023))

Figure 3.3 demonstrates the vector representations of words like “woman” and “queen” are similar or close to each other in the vector space. Similarly, the vectors for “man” and “king” are also close. This closeness indicates that there is a semantic relationship between these words. In simpler terms, the Word2Vec model understands that “queen” is related to “woman” in the same way “king” is related to “man.”

The Word2Vec model can be represented mathematically as follows:

$$Word2Vec(word) = \frac{1}{C} \sum_{context\,word} word\_vector(context\,word) \tag{3.4}$$

Where,

$Word2Vec(word)$ = Vector representation from Word2Vec for a given word

$contextword$ = Neighboring words in the context of the target word

$word\_vector(contextword)$ = Vector representation of the context word

C = Number of context words

The trained Word2Vec embeddings utilised in the code are derived from a Google pre-trained model. The code specifically imports the Word2Vec model from the file "GoogleNews-vectors-negative300.bin." This pre-trained model is based on an extensive corpus of Google News articles and contains vector representations for a variety of terms. Each word is represented as a 300-dimensional dense vector that captures semantic and syntactic information.

To obtain a single vector representation for each review, the average of each word (token) of the Word2Vec embedding is calculated. For each token in a review, the function checks if its corresponding vector is available in the Word2Vec model. If the vector is present, it is retrieved. Otherwise, a random vector or a zero vector is used as a fallback. Once the Word2Vec vectors for all the tokens in a review have been obtained, they are averaged to compute a single vector representation for that review. This averaging process combines the individual token vectors into one representative vector, capturing the overall semantics of the review. The equation for average Word2Vec embedding for each review is as follow:

$$\text{average\_embedding} = \frac{1}{n}\sum_{i=1}^{n}\text{Word2Vec}(\text{token}_i) \tag{3.5}$$

Where,

n = the total number of tokens in the reviews

$Word2Vec(token_i)$ = represents the Word2Vec vector for the ith token in the reviews

$average\_embedding$ = represents the average Word2Vec embedding for a review

The next step involves applying the average Word2Vec embeddings to each review in the dataset. A function is used for this purpose. This function takes two inputs: the Word2Vec model (Word2Vec) and the preprocessed words (tokens) of a review. The function then

generates a list of Word2Vec embeddings, where each embedding corresponds to a specific review in the dataset. These embeddings serve as the feature representation for the reviews.

By generating Word2Vec embeddings for each review, the code converts the reviews into dense numerical vectors that encode the semantic information of the words. These embeddings capture the contextual meaning of the reviews based on their relationships with other words in the pre-trained Word2Vec model. These resulting embeddings are now ready to be used as input features for subsequent text classification tasks. In this case, the goal is to classify the reviews into the three classes: delivery services, application performance, and customer service.

3.4.3 Classifier

A classifier is a machine learning algorithm or model that is used to predict the class of a review based on its features extracted using TF-IDF and Word2Vec method. The classifier is trained on a labeled dataset, where the input reviews are associated with their respective class labels. During the training phase, the classifier learns patterns and relationships in the data, and then making predictions on new, unseen samples

In this research logistic regression, decision tree, and support vector machines (SVM) classifiers are being explored to classify text reviews into three categories of delivery service, application performance, and customer service.

Linear Support Vector Machine

The classification technique Linear Support Vector Machine (Linear SVM) searches for the best hyperplane with the widest margin for separating the three classes of reviews. The

technique presumes that the data points are mapped to feature vectors in a large space. A decision boundary that divides the data points into various classifications is called hyperplane. Finding the hyperplane that maximises the margin that separates between the hyperplane and the nearest data points for each class is the goal. This margin guarantees improved generalisation and enhances the classifier's capacity to correctly classify novel, unseen data. Figure 3.4 shows the hyperplane separating the data points of three different classes.

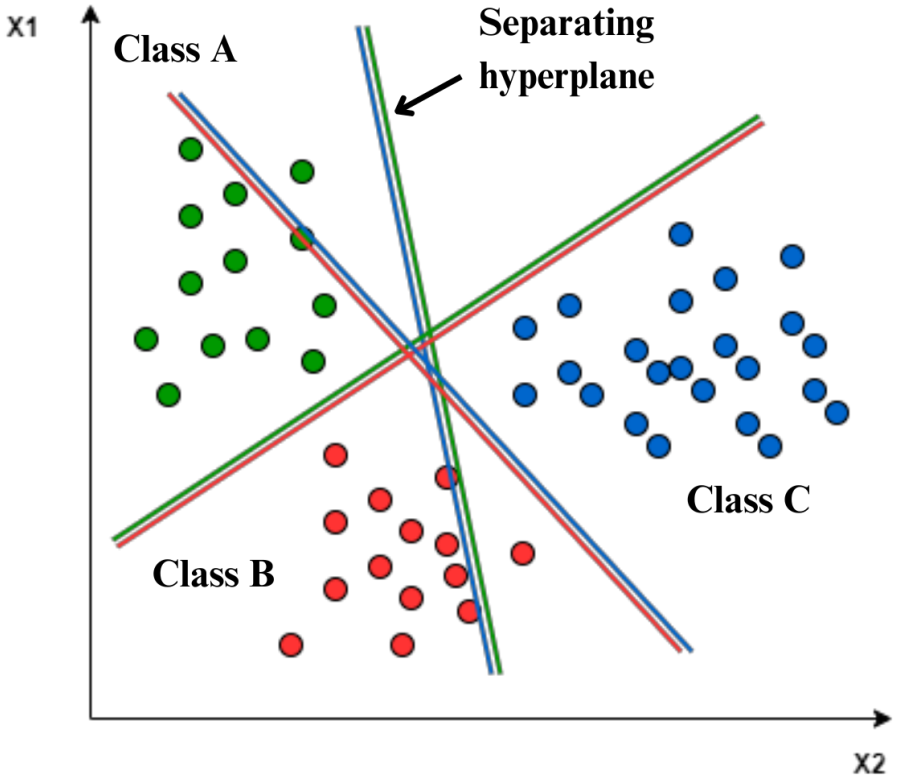

Figure 3.4: Hyperplanes separating three classes

Logistic Regression

Logistic regression uses the extracted features from TF-IDF and Word2Vec to estimate the probabilities of the documents belonging to each class. The logistic regression model applies the logistic function to the weighted sum of the input features, mapping the output to a probability value between 0 and 1. By setting a decision threshold, the model can

assign the documents to the most probable class based on the calculated probabilities.

The learned weights in logistic regression represent the importance or contribution of each feature (TF-IDF values or Word2Vec representations) towards the classification task. Features with higher weights have a stronger influence on the final classification decision. For each class, a separate logistic regression model is trained to distinguish that class from the other classes. The equation for the one-vs-rest logistic regression model can be written as:

$$P(y = c|\mathbf{x}) = \frac{1}{1 + \sum_{k=1}^{K-1} e^{-(\mathbf{w}_c - \mathbf{w}_k)^T \mathbf{x}}} \tag{3.6}$$

Where,

$P(y = c \mid x)$ = represents the probability of the document belonging to class c given the input features x.

w_c = is the weight vector for class in the logistic regression model.

w_k = is the weight vector for the comparison class k in the logistic regression model.

K = is the total number of classes.

In this formulation, the probability of a document belonging to class c is computed as the inverse of the sum of exponentiated differences between the weight vectors for class c and all other comparison classes. To classify a document, the predicted class is the one with the highest probability among all the classes.

Decision Tree

A decision tree is a machine learning algorithm that makes decisions based on the input features. It creates a hierarchical structure of decisions, where each decision splits the data into different regions. Each region represents a specific class or a combination of classes. At each internal node of the decision tree, a decision rule is applied to determine which feature or combination of features to consider for further splitting.

The leaf nodes of the decision tree represent the final classification label. When classifying a review, the decision tree traverses the tree from the root to a leaf node based on the feature values of the reviews. The class assigned to the leaf node reached by the review represents the predicted class for that review. In the case of three classes, the decision tree learns decision rules that differentiate between the three classes, considering the available features from TF-IDF and Word2Vec. The tree structure allows for classification into the three classes based on the feature values of a given document. Figure 3.5 visualises a simple decision tree with three classes as its terminal node.

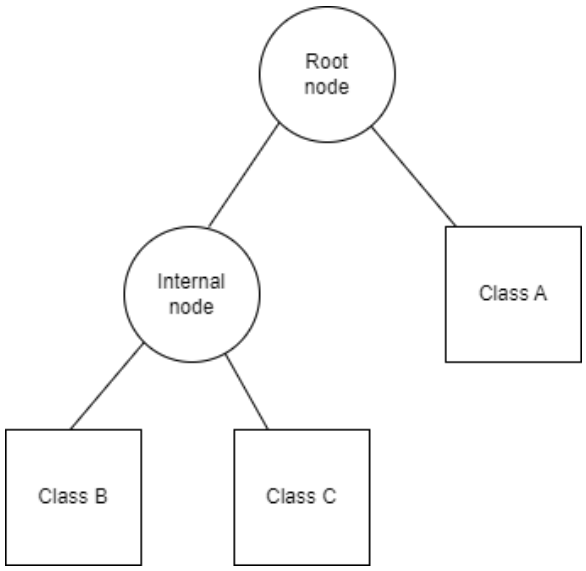

Figure 3.5: Example of Decision Tree Classifier

The decision rule used in this text classification is based on metric of Gini impurity, which

measure the ability of a feature to separate the classes effectively. Gini impurity measures the probability of incorrectly classifying a randomly chosen element in the dataset if it were randomly labeled according to the class distribution at the node. The 'DecisionTreeClassifier' in scikit-learn library is used with TF-IDF and Word2Vec features to build a decision tree model for text classification using the Gini impurity criterion, The algorithm seeks to minimize the Gini impurity when determining the splits in the decision tree.

The formula for calculating Gini impurity is :

$$\text{Gini}(p) = 1 - \sum_{i=1}^{C} p_i^2 \tag{3.7}$$

Where,

$Gini_p$ = represents the Gini impurity for a node with class probabilities $p = p_1, p_2, ..., p_C$

C = is the total number of classes

p_i = represents the probability of observing class i at the node

3.4.4 Training and Validation Accuracy

The training and validation accuracy of the classifers are calculated by comparing the predicted labels with the actual labels and returns the fraction of correctly classified samples. The accuracy score focuses on the overall correctness of the predictions by comparing the predicted labels with the true labels. The training accuracy represents the accuracy of the model on the training data, while the validation accuracy represents the accuracy of the model on the validation data.

$$\text{Accuracy} = \frac{\text{Number of correctly classified reviews}}{\text{Total number of reviews}}$$

3.5 SUMMARY

The research focuses on evaluating the performance of TF-IDF and Word2Vec as feature extraction methods for text classification of user reviews on the Shopee application available in the Google Play Store. The study aims to classify the user reviews into three distinct categories: delivery service, application performance, and customer service. Initially, a collection of user reviews is obtained and manually annotated to serve as the training data for the classifiers. The annotated dataset is then split into two sets: a train-test split and a validation-test split. This division enables independent evaluation and validation of the models. The text data in the datasets is preprocessed and normalized using TF-IDF and Word2Vec techniques to extract relevant features. These techniques assign weights to words based on their significance within individual documents and across the entire corpus. The extracted features are then utilized as input for three classifiers: linear Support Vector Machine (SVM), logistic regression, and decision tree. These classifiers are trained on the training data to predict the class labels of the user reviews. Finally, the accuracy of each classifier is calculated by comparing the predicted labels with the ground truth labels in the test sets. By comparing the accuracies of the different classifiers, insights can be gained regarding the effectiveness of TF-IDF and Word2Vec for text classification in the context of Shopee application user reviews.

Table 3.6 shows the summary of the research objectives and the respective method to answer the objective.

Table 3.6: Summary of data analysis

Research Objectives	Method of Analysis
To evaluate the performance of TF-IDF as a feature extraction technique in text classification on Shopee user application reviews.	TF-IDF
To evaluate the performance of Word2Vec as a feature extraction technique in text classification on Shopee user application reviews.	Word2Vec
To determine which feature extraction technique, TF-IDF or Word2Vec, yields better performance in text classification tasks	Linear SVM Logistic Regression Decision Tree

CHAPTER 4

RESULTS AND DISCUSSION

4.1 INTRODUCTION

The results and findings for each objective is addressed in this chapter based on the method used, which is Word2Vec and TF-IDF as feature extraction on text classification. In order to address the study's research purpose, the findings of this study is thoroughly examined in accordance with the following research questions:

1. What is the performance of TF-IDF as a feature extraction technique in text classification?

2. What is the performance of Word2Vec as a feature extraction technique in text classification?

3. Which feature extraction technique, TF-IDF or Word2Vec, yields better performance in text classification tasks?

4.2 DESCRIPTIVE ANALYSIS

This section discusses descriptive analysis of the data that has been gathered in the Data Gathering process. Descriptive analysis is a statistical technique used to summarise and describe a dataset's characteristics. During the process of data collection, descriptive analysis is essential for comprehending the characteristics of the data. It assists in understanding the distribution of variables, identifying outliers or absent values, and evaluating the overall integrity of the data.

4.2.1 Comparison of class frequencies

Comparison of class frequency in Figure 4.1 is visualized using a pie chart. The majority of the reviews (51.5%) are related to application performance. Delivery service and customer service account for 24.5%(554 reviews) and 24.0% (501 reviews) of the reviews, respectively. In total, there are 2309 reviews in the dataset.

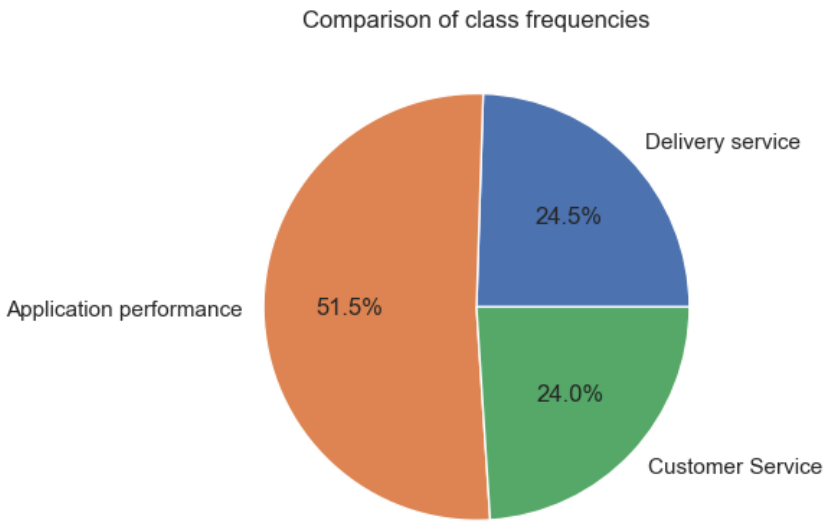

Figure 4.1: Comparison of class frequencies

4.2.2 Distribution number of characters of for each classes

Figure 4.2 shows the distribution number of characters of each reviews in the application performance class, the reviews with the number of characters between the range of 100 - 200 has the highest count compared to others. The lowest count of number of characters can be seen in the range 400 to 500 words. The graph of application performance distribution show right skewed distribution.

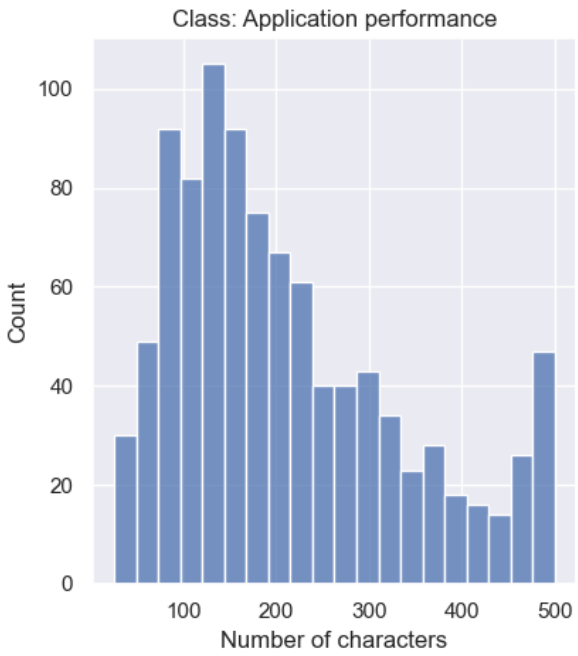

Figure 4.2: Distribution number of characters : Application Performance

Figure 4.3 and 4.4 show the distribution number of characters in review of delivery service and customer service classes respectively, the distribution number of characters for delivery service and customer service class both has the highest count of review withing 500 number of characters. Figure 4.3 has the lowest count of the distribution of number of character in the range 100 and below and 300-400 numbers of character. While for Figure 4.4 has the lowest count of number of characters of 400 words per reviews.

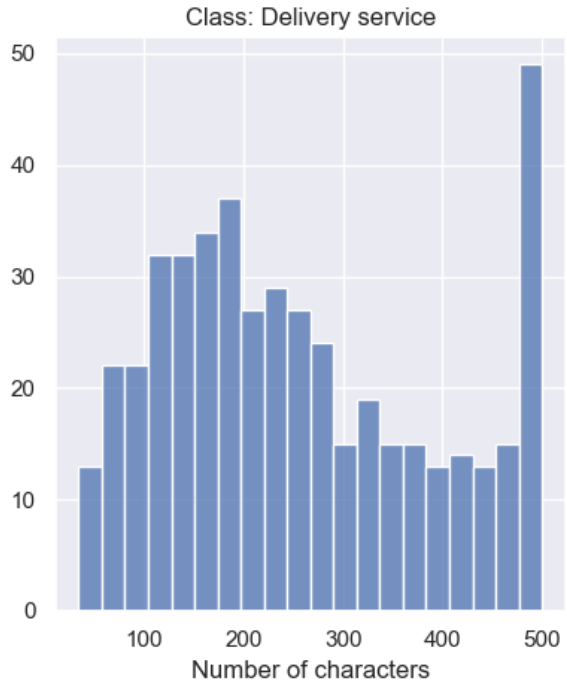

Figure 4.3: Distribution number of characters : Delivery Service

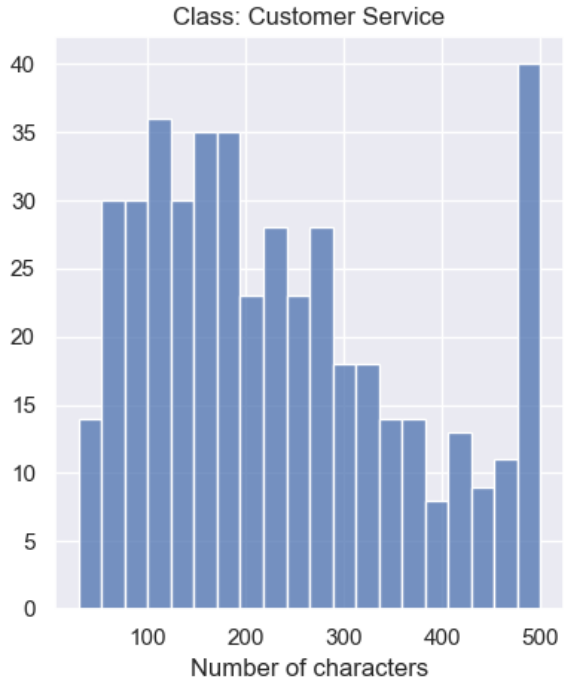

Figure 4.4: Distribution number of characters : Customer Service

4.2.3 Comparison size of training set, testing set and validation set

Figure 4.5 shows the sizes different between training, testing set and validation set, with the training set holds the majority of the data (80%) with 1847 of the reviews, and the remaining 20% set is divided equally between testing and validation set with 231 reviews respectively.

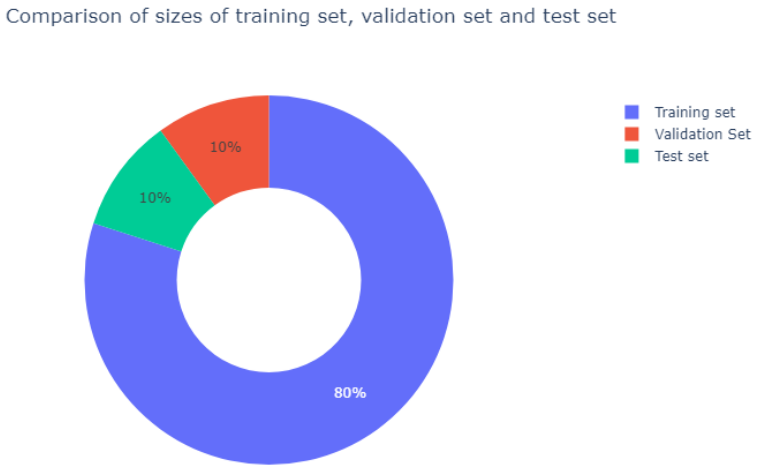

Figure 4.5: Comparison size of training set, testing set and validation set

4.3 RESULT OF TEXT NORMALISATION PROCESS

The text normalization process is an important step in preparing textual data for feature extraction techniques of TF-IDF and Word2Vec. Text normalization involves transforming text data to a standardized and consistent format, which helps in reducing noise and variations in the text, and improving the quality of extracted features.

4.3.1 TF-IDF normalized description

By employing text normalisation techniques and preparing the user reviews of Shopee application for TF-IDF feature extraction, the resulting data in Figure 4.6 is more consistent, less noisy, and optimised for capturing meaningful patterns and word

relationships. This procedure improves the ability of machine learning models to extract insightful information from the reviews dataset.

	normalized description	label
476	always crash log account minute install instal...	application
704	hope suckthey cut everything hope well canada ...	customer service
84	payment do status still pende contact customer...	customer service
879	lag even phone rainbow login hope hope help ac...	application
70	far use application shopping seller system how...	application
..	...	...
440	overtime app say something back homepage back ...	application
165	app nowadays lag muchwhen use totally crash in...	application
7	use hope almost year think hope try edit produ...	application
219	hope kai say version contact seller buy anythi...	application
326	buggy baggy gui useless function buggy gui use...	application

Figure 4.6: TF-IDF normalization result on Shopee user reviews from Google Play Store

The resulting partial text normalisation process on Shopee user reviews in the Figure 4.7 prepares the tokens for Word2Vec training phase to learn the word embeddings or vector representations.

	tokens	label
476	[very, bad, always, crashing, and, log, out, m...	application
704	[shopee, is, suck, they, cut, everything, not,...	customer service
84	[payment, done, but, status, still, pending, v...	customer service
879	[too, lag, even, phone, have, big, ram, now, i...	application
70	[so, far, i, am, using, shopee, application, f...	application
..	...	...
440	[everytime, i, open, the, apps, it, said, ther...	application
165	[why, this, app, nowdays, been, lagging, so, m...	application
7	[i, have, used, shopee, almost, about, a, year...	application
219	[the, shopee, ramadan, kasi, sayang, version, ...	application
326	[too, buggy, too, laggy, too, heavy, unrespons...	application

Figure 4.7: Word2Vec partial normalization result on Shopee user reviews from Google Play Store

4.4 THE PERFORMANCE OF TF-IDF AS A FEATURE EXTRACTION TECHNIQUE IN TEXT CLASSIFICATION ON SHOPEE USER APPLICATION REVIEWS.

In order to assess the performance of TF-IDF as a feature extraction technique, the performance metrics of three classifiers on the task of text classification using TF-IDF feature extraction is presented as depicted in Table 4.1 . The classifiers evaluated include Linear Support Vector Machine (SVM), Logistic Regression, and Decision Tree. The training accuracy and validation accuracy are reported for each classifier. The results provide insights into the effectiveness of these classifiers in accurately categorizing text documents based on TF-IDF features.

Table 4.1: Performance Comparison of Text Classification using TF-IDF Feature Extraction Techniques

No	Classifier	Training Accuracy	Validation Accuracy
1	Logistic Regression	0.885219	0.783550
2	Decision Tree	0.958311	0.696970
3	Linear SVM	0.908500	0.774892

Table 4.1, shows that Logistic regression classifier achieves a satisfactory degree of performance, with a training accuracy of 0.885219 and a validation accuracy of 0.783550. This demonstrates that the classifier performs well on both the training and validation sets, classifying nearly all instances accurately. Decision Tree, on the other hand, obtains a training accuracy of 0.958311 but a validation accuracy of 0.696970. The Linear SVM classifer exhibits strong performance, matching the training accuracy of the Linear SVM at 0.908500 and achieving a validation accuracy of 0.774. Overall, these classifiers exhibit varying levels of performance, with the Logistic Regression Classifier exhibiting the best generalisation capabilities to unseen data and demonstrating the highest level of performance.

Figure 4.8 shows the result of text classification using TF-IDF technique as feature extraction and Logistic Regression as classifier in a csv file, the textual reviews from Google Play Store, original manually annotated labels and predicted labels is shown in this figure. It is shown that Logistic Regression classifier managed to label review number 146 until 149 correctly, but fail to do so for review number 150 where it classifies a 'customer service' review to an 'application' review.

1	Reviews	Original Label	Predicted Label
146	Better online shopping..still hÃ ve a room for improvement. Most important get ready for heavy traffic due some time lag here and there.	application	application
147	I totally don't understand why its keep saying " oops something wrong, back to homepage" even after i updated the app. Pls fix this i wanna track my order	application	application
148	Can find everything there but quite difficult to talk to customer service when	customer service	customer service
149	I shop on shopee frequently and the experience so far has been good though sometimes the app becomes laggy/very delayed so I have to restart the app everytime it happens. I'd still use it but it can definately improve more.	application	application
150	Useless and fed up. I used to give you 5* but now I change it to 1*. As a regular shopee customer for so many years now, I am really disappointed. Seller gave me wrong item and refused to refund the balance to me. So I rated badly on his product and screenshot the item I received and invoice as proof. Shopee just deleted my review. Plus he still didn't refund my money. Where is justice for me as	customer service	application

Figure 4.8: Predicted labels using Feature Extraction TF-IDF and Logistic Regression Classifier

4.5 THE PERFORMANCE OF WORD2VEC AS A FEATURE EXTRACTION TECHNIQUE IN TEXT CLASSIFICATION ON SHOPEE USER APPLICATION REVIEWS.

For the second research question, the performance metrics of three classifiers on the task of text classification using Word2Vec feature extraction is presented.

Table 4.2: Performance Comparison of Text Classification using Word2Vec Feature Extraction Techniques

	Classifier	Training Accuracy	Validation Accuracy
1	Logistic Regression	0.796968	0.792208
2	Decision Tree	0.958311	0.645022
3	Linear SVM	0.806714	0.787879

In the Table 4.2 it shows that the Logistic Regression classifier performs well, reaching validation accuracy of 0.796968 and a training accuracy of 0.792208. This shows that the Logistic Regression classifier was successful in identifying the links and patterns present in the text data, and making satisfactory predictions. The Decision Tree and Linear SVM classifiers exhibit inferior validation accuracy. While Decision Tree gets comparable results with a training accuracy of 0.958311 and a validation accuracy of 0.645022, Linear SVM classifier achieves a training accuracy of 0.806714 and a validation accuracy of 0.787879. These lower accuracies imply that Decision Tree and Linear SVM may adequately represent the complexity of the text input, which result in fewer accurate predictions, but fell inferior to Logistic Regression classifier.

Figure 4.9 shows the result of text classification using Word2Vec technique as feature extraction and Logistic Regression as classifier in a csv file. It is shown that Logistic Regression classifier using Word2Vec as feature extraction technique managed to label review number 145 until 150 correctly.

1	Reviews	Original Label	Predicted Label
145	Removing 2 stars for adding meaningless notification feature that notify seller in 'seller updates' column. Edit: removing another star for limiting the uses of voucher to ONE ONLY. Not to mention the useless Shopee express delivery service. Will surely spread the word around about how far this company have fallen.	delivery service	delivery service
146	Better online shopping..still hÃ ve a room for improvement. Most important get ready for heavy traffic due some time lag here and there.	application	application
147	I totally don't understand why its keep saying " oops something wrong, back to homepage" even after i updated the app. Pls fix this i wanna track my order	application	application
148	Can find everything there but quite difficult to talk to customer service when products r not to my satisfaction	customer service	customer service
149	I shop on shopee frequently and the experience so far has been good though sometimes the app becomes laggy/very delayed so I have to restart the app everytime it happens. I'd still use it but it can definately improve more.	application	application
150	Useless and fed up. I used to give you 5* but now I change it to 1*. As a regular shopee customer for so many years now, I am really disappointed. Seller gave me wrong item and refused to refund the balance to me. So I rated badly on his product and screenshot the item I received and invoice as proof. Shopee just deleted my review. Plus he still didn't refund my money. Where is justice for me as a buyer? Seller no need to refund and my review gets deteled by shopee admin?? So what abt my money??	customer service	customer service

Figure 4.9: Predicted labels using Feature Extraction Word2Vec and Logistic Regression Classifier

4.6 EVALUATION OF FEATURE EXTRACTION PERFORMANCE

Validation accuracy is an important metric to consider when comparing the performance of method of feature extraction because it provides an estimate of how well the model generalizes to unseen data. The validation accuracy is computed using a separate dataset that was not used during the training phase.

By evaluating the performance of classifiers on the validation dataset, the ability to correctly classify new, unseen instances can be asses. A higher validation accuracy indicates that the classifier is better at generalizing patterns and making accurate predictions on new data. Considering the validation accuracy allow the best performing classifier to be selected on unseen data and make more reliable predictions in real-world scenarios.

Table 4.3: Performance Comparison of Text Classification using Word2Vec Feature Extraction Techniques

	Feature extraction methods	
Classifier	TF-IDF	Word2Vec
Linear SVM	0.774892	0.787879
Logistic Regression	0.783550	0.792208
Decision Tree	0.696970	0.645022

Table 4.3 shows the performance of feature extraction TF-IDF and Word2Vec respective validation accuracies is compared. Among the classifiers used to test the performance of feature extraction method Word2Vec, the Logistic Regression stood out with a validation accuracy of 0.792208 showing its good overall performance in correctly predicting the class labels for unseen data. For the Linear SVM classifier, the accuracy with Word2Vec (0.787879) is higher than with TF-IDF (0.774892). Similarly, for the Logistic Regression classifier, the accuracy with Word2Vec (0.792208) is higher than with TF-IDF (0.783550). Lastly, for the Decision Tree classifier, the accuracy with Word2Vec (0.645022) is also higher than with TF-IDF (0.696970). This concludes that feature extraction method Word2Vec achieved higher validation accuracies compared to TF-IDF, showing that it is a better feature extraction method in classifying the textual reviews of Shopee user reviews.

This comparison has concluded that the best feature extraction and classifier combination is Word2Vec and Logistic Regression.

4.6.1 Predicted Class Probabilities using Logistic Regression Classifier and Word2Vec Embeddings

This subsection discussess the best classifier approach which is multiclass logistic regression.

In this study, the multiclass logistic regression, serves as the tool to predict the categories of “Application,” “Delivery Service,” and “Customer Service” for each individual review. This technique offers a method that simplifies the process by using a single model to predict the probability distribution across all three classes. Instead of creating separate models for each category, the multiclass logistic regression or softmax regression allows training of one comprehensive model.

During the training phase, this model learns intricate patterns and associations between the words present in the reviews and the three distinct classes. It adapts its internal parameters, essentially its weights, to ensure the highest accuracy in predicting the appropriate class label for each review based on the input words it receives.

Softmax regression is able to predict probabilities. After the training is complete, the model can estimate the probability that a given review belongs to each of the three classes. The predicted probabilities are normalized, meaning they sum up to 1. This normalization ensures that the probabilities is used as a measure of the model’s confidence in assigning a specific review to each class. Subsequently, the class with the highest probability is chosen as the final predicted class for that particular review.

Table 4.4 shows the reviews that is then used to predict the label using the multiclass logistic regression approach.

Table 4.4: Reviews

Number	Review
1	So far so good. Reliable apps for shopping. I gave no bad experiences so far.
2	The most slowest & lagging apps ever use!
3	The apps suddenly restart and open back with pop-up message “Opss, something went wrong...we are looking into it..”
4	I had a very bad experience with shopee particularly with refund for faulty products. 3 times I received a faulty solar lamps from China and all 3 times I requested for refund shopee would asked me to to wait for1 whole week for them to review my request.

The presented output in Table 4.5 showcases the outcomes of predicting class probabilities for a specific set of validation data instances, utilizing the pre-trained logistic regression model with Word2Vec embeddings. In this matrix representation, each row corresponds to an individual validation data point, while each column signifies a distinct class. Within this matrix, the numbers signify the predicted probabilities associated with each class for the respective data point.

Review	**Application Performance**	**Customer Service**	**Delivery Service**
1	0.513	0.391	0.095
2	0.999	0.000	0.000
3	0.985	0.012	0.003
4	0.077	0.534	0.389

Table 4.5: Predicted Class Probabilities

For instance, analyzing the contents of Review Number 1, we observe that the predicted

probabilities for the classes are approximately as follows: [0.513, 0.391, 0.095]. This indicates that the model has assigned a probability of 0.513 for the data point to belong to the “Application Performance” class, 0.391 for the “Customer Service” class, and 0.095 for the “Delivery Service” class. Analyzing content Review Number 4, we can see that the model assigned the highest probability for the “Customer Service” class indicating Review Number 4 is predicted to be in that class.

4.7 SUMMARY OF FINDINGS

These findings does not align with the conclusions drawn in the article "Performance comparison of TF-IDF and Word2Vec models for emotion text classification" by Cahyani and Patasik (2021) which reported higher validation accuracies using the TF-IDF approach in emotion text classification. Instead, in this study, the findings demonstrate the superior performance of Word2vec feature extraction method compared to TF-IDF in the classification of text.

The are two possible reasons that may cause the difference in previous literature reviews and this research findings, first, the reason may be due to the instability of feature extraction techniques due to small training data size of only 2309 reviews and high dimensionality, this may lead to inconsistent accuracy in text classification as unstable feature extraction methods may yield different sets of selected features each time it is applied to the same data set (Kou et al., 2020). This inconsistency may lead to varying or poor results in terms of classification accuracy resulting the difference in previous in findings.

Secondly, in contrast to the study by Cahyani and Patasik in (2021), this study incorporates the use of Google pre-trained word embeddings with a vector length of 300 features in Word2Vec feature extraction method. This enables this research Word2Vec-based approach to capture more nuanced and context-specific information from the small training dataset, leading to improved accuracies compared to the findings reported in Cahyani and Patasik's paper. By leveraging the power of pre-trained word embeddings, this study benefits from a richer representation of the textual data, enhancing the discriminative power of the Word2Vec features.

CHAPTER 5

CONCLUSION AND RECOMMENDATION

5.1 CONCLUSION

For the first objective of evaluating the performance of TF-IDF as a feature extraction technique in text classification of Shopee user reviews on Google Play Store, the findings concluded that, using TF-IDF feature extraction, the Logistic Regression Classifier outperformed the other model and demonstrated the best generalization capabilities and proved to be the most effective in classifying text data accurately.

For the second objective of evaluating the performance of Word2Vec as a feature extraction technique findings shows that the Logistic Regression classifier demonstrated superior performance in text classification again, achieving a high validation accuracy of 0.796968 and a training accuracy of 0.792208. This indicates its ability to effectively identify patterns and links in the text data, resulting in satisfactory predictions.

For the final objective of determine the best performance between TF-IDF and Word2Vec feature extraction techniques in text classification, the result shows that Word2Vec has performed slightly better in text classification comparing to TF-IDF.

Comparing the validation accuracies between TF-IDF and Word2Vec feature extraction methods shows the superiority of Word2Vec method as the validation accuracy for each classifier exceeds of TF-IDF. This concludes that Word2Vec is a better feature extraction method than TF-IDF.

5.2 RECOMMENDATION

In light of the limitations of this study, it is recommended that future study comparing the performance of TF-IDF and Word2Vec is to be conducted on larger annotated data set. This is to reduce the instability of the features and to enhance the performance of feature extraction techniques. In order to achieve this, efforts should be made to increase the training data set in order to solve the issue of restricted training data availability. This can entail compiling more user reviews from other reliable sources or using labelled data sets that already exist. Next recommendation is for future study to explore the incorporation of complementary feature extraction techniques, such as combining TF-IDF with Word2Vec, to enhance the accuracy of text classification and gain deeper insight of new and efficient feature extraction techniques. Lastly, it is advised for further researcher to look into options for obtaining extra computational resources in order to get around the problem of restricted computational resources. This can entail looking for partnerships with academic institutes or businesses that have access to cutting-edge computer facilities. A further benefit of investigating cloud-based computing options is that they can offer scalable and affordable resources for data processing and analysis.

REFERENCES

Ahuja, P. (2020, Jan). *How good (or bad) is traditional tf-idf text mining technique?* Retrieved from https://medium.com/analytics-vidhya/how-good-or-bad-is-traditional-tf-idf-text-mining-technique-304aec920009

Annie, A. (2021). *The state of mobile 2021*. Retrieved from https://www.appannie.com/en/go/state-of-mobile-2021/

Apptentive. (2019). Mobile app benchmarks: The average ratings, reviews, and retention rates. *Apptentive Blog*. https://www.apptentive.com/blog/2019/03/11/mobile-app-benchmarks-the-average-ratings-reviews-and-retention-rates/. (Accessed: 2023-03-29)

Badjatiya, P., Gupta, S., Gupta, M., & Varma, V. (2017, 06). Deep learning for hate speech detection in tweets.
doi: 10.1145/3041021.3054223

Beel, J., Gipp, B., Langer, S., & Breitinger, C. (2016). Research-paper recommender systems : a literature survey. *International Journal on Digital Libraries*, *17*(4), 305–338. doi: 10.1007/s00799-015-0156-0

Beleites, C. (2013, June). *How large a training set is needed?* Cross Validated. Retrieved from https://stats.stackexchange.com/q/51527 (URL:https://stats.stackexchange.com/q/51527 (version: 2013-03-06))

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Bughin, J., Chui, M., & Manyika, J. (2018). The business value of data-driven decision making. *McKinsey Quarterly*, *2*.

Bush, T. (2020, Jun). *Descriptive analysis: How-to, types, examples.* Pestle Analysis. Retrieved from https://pestleanalysis.com/descriptive-analysis/

Cahyani, D. E., & Patasik, I. (2021, Oct). Performance comparison of tf-idf and

word2vec models for emotion text classification. *Bulletin of Electrical Engineering and Informatics*, *10*(5), 2780–2788. doi: https://doi.org/10.11591/eei.v10i5.3157

Del Vigna, F., Cimino, A., Dell'Orletta, F., Petrocchi, M., & Tesconi, M. (2017, 01). Hate me, hate me not: Hate speech detection on facebook..

Dongyang Wang, H. Y., Junli Su. (2020). Feature extraction and analysis of natural language processing for deep learning english language. *IEEE Access*, *8*, 46335-46345. doi: 10.1109/ACCESS.2020.2974101

Dorfman, E. (2022, Mar). *How much data is required for machine learning? - postindustria.* Postindustria, Inc. Retrieved from https://postindustria.com/how-much-data-is-required-for-machine-learning/#:~:text=The%20most%20common%20way%20to

Duan, W., Gu, B., & Whinston, A. B. (2008). Do online reviews matter? an empirical investigation of panel data. *Decision Support Systems*, *45*(4), 1007–1016.

Feine, J., Morana, S., & Gnewuch, U. (2019, 02). Measuring service encounter satisfaction with customer service chatbots using sentiment analysis..

Gupta, S. (2018, Jan). *Automated text classification using machine learning.* Retrieved from https://towardsdatascience.com/automated-text-classification-using-machine-learning-3df4f4f9570b#:~:text=The%20algorithms%20are%20given%20a

Hu, P., Li, Q., & Ye, Y. (2019). Customer review analysis using natural language processing techniques: a case study of e-commerce platforms. *Sustainability*, *11*(8), 2234.

iPrice Group. (2021). *iprice insights: State of ecommerce in southeast asia 2021.* Retrieved from https://iprice.my/insights/mapofecommerce/en/

javaTpoint. (2021). *Machine learning decision tree classification algorithm - javatpoint.* Retrieved from https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm

Kadhim, A. I. (2019). Term weighting for feature extraction on twitter: A comparison

between bm25 and tf-idf. In *2019 international conference on advanced science and engineering (icoase)* (pp. 124–128).

Kanaris, I., Stamatatos, E., & Fakotakis, N. (2020). Tf-idf vs word2vec vs glove: An overview. *arXiv preprint arXiv:2010.02545*.

Kim, M.-G., & Yoon, Y.-J. (2020). Negative consequences of text classification: A critical review and practical remedies. *Journal of the Association for Information Science and Technology*, *71*(8), 936–947.

Kou, G., Yang, P., Peng, Y., Xiao, F., Chen, Y., & Alsaadi, F. E. (2020). Evaluation of feature selection methods for text classification with small datasets using multiple criteria decision-making methods. *Applied Soft Computing*, *86*, 105836. Retrieved from https://www.sciencedirect.com/science/article/pii/S1568494619306179 doi: https://doi.org/10.1016/j.asoc.2019.105836

Kulshrestha, U. (2020, Jun). *Nlp — feature selection using tf-idf.* Retrieved from https://medium.com/analytics-vidhya/nlp-feature-selection-using-tf-idf-db2f9eb484fb#:~:text=TF%2DIDF%20acronym%20for%20Term

Levy, O., Goldberg, Y., & Dagan, I. (2015). Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, *3*, 211–225.

Logunova, I. (2023, May). *Word2vec: Explanation and examples.* Retrieved from https://serokell.io/blog/word2vec

M, D. (2021, Feb). *How tf-idf works.* Retrieved from https://towardsdatascience.com/how-tf-idf-works-3dbf35e568f0

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2015). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013, Oct). Distributed representations of words and phrases and their compositionality. *arXiv (Cornell University)*. doi: https://doi.org/10.48550/arxiv.1310.4546

Pradhan, L., Taneja, N. A., Dixit, C., & Suhag, M. (2017, Mar). Comparison of text classifiers on news articles. *International Research Journal of Engineering and Technology*, *4*(3). doi: e-issn:2395-0056

Raj, A. (2020, Nov). *Perfect recipe for classification using logistic regression.* Towards Data Science. Retrieved from https://towardsdatascience.com/the-perfect-recipe-for-classification-using-logistic-regression-f8648e267592#:~:text=Logistic%20regression%20is%20a%20classification

Rajendran, S. (2021). Improving the performance of global courier & delivery services industry by analyzing the voice of customers and employees using text analytics. *International Journal of Logistics Research and Applications*, *24*(5), 473-493. Retrieved from https://doi.org/10.1080/13675567.2020.1769042 doi: 10.1080/13675567.2020.1769042

Rouse, M. (2023, Jun). *Training data.* Retrieved from https://www.techopedia.com/definition/33181/training-data#:~:text=The%20training%20set%20is%20the

Similarweb. (2023, Jul). *Top websites ranking most visited ecommerce shopping websites in malaysia.* Similarweb LTD. Retrieved from https://www.similarweb.com/top-websites/malaysia/e-commerce-and-shopping/#:~:text=shopee.com.my%20ranked%20number,eCommerce%20%26%20Shopping%20websites%20in%20Malaysia.

Sparck Jones, K. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, *28*(1), 11–21.

Sunil, R. (2019, Mar). *Understanding support vector machine algorithm from examples (along with code).* Retrieved from https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/

Turney, P. D. (2002). Thumbs up or thumbs down? semantic orientation applied to

unsupervised classification of reviews. *Association for Computational Linguistics*.

Vatsal. (2022, Feb). *Word2vec explained.* Retrieved from https://towardsdatascience.com/word2vec-explained-49c52b4ccb71

Wang, Y., Fu, W., Sui, A., & Ding, Y. (2015). Comparison of four text classifiers on movie reviews. In *2015 3rd international conference on applied computing and information technology/2nd international conference on computational science and intelligence* (pp. 495–498).

Wei, H. (2019, Mar). *Nlp pipeline 101 with basic code example — feature extraction.* Voice Tech Podcast. Retrieved from https://medium.com/voice-tech-podcast/nlp-pipeline-101-with-basic-code-example-feature-extraction-ea9894ed8daf#:~:text=Feature%20extraction%20step%20means%20to

Wolff, R. (2020, Nov). *What is training data in machine learning?* Retrieved from https://monkeylearn.com/blog/training-data/

Yu, B., Zhou, J., Zhang, Y., & Cao, Y. (2017, 09). Identifying restaurant features via sentiment analysis on yelp reviews.

Zvornicanin, E. (2023, June). *Word2vec word embedding operations: Add, concatenate or average word vectors?* Baeldung. Retrieved from https://www.baeldung.com/cs/word2vec-word-embeddings (Last updated on June 8, 2023)

APPENDIX A

```
import csv
from google_play_scraper import Sort, reviews

result, continuation_token = reviews(
    'com.shopee.my',
    lang='en',
    country='my',
    sort=Sort.NEWEST,
    count=10000,
)
with open('C:/Users/Safawi/Desktop/shopee_reviews10k.csv',
   mode='w', newline='', encoding='utf-8') as file:
    writer = csv.writer(file)
    writer.writerow(['User', 'Text', 'Date', 'Score', 'Thumbs
       Up Count'])
    for review in result:
        writer.writerow([review['userName'], review['content'
           ], review['at'], review['score'], review['
           thumbsUpCount']])

print('Scraping complete!')
```

APPENDIX B

```
# File system manangement
import time, psutil, os

# Data manipulation
import numpy as np
import pandas as pd

# Plotting and visualization
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_theme()

# Recording the starting time, complemented with a stopping
    time check in the end to compute process runtime
start = time.time()
process = psutil.Process(os.getpid())

# Load data from Excel file
data = pd.read_csv("C:/Users/Safawi/Desktop/FYP CSV/SHOPEE
    REVIEWS TRAINING FINAL.csv",
     names = ['text' , 'label']
)
data = data[['text', 'label']]

print(pd.Series({"Memory usage": "{:.2f} MB".format(data.
    memory_usage().sum()/(1024*1024)),
```

```
24                    "Dataset shape": "{}".format(data.shape)}).
                          to_string())
25 print(data)
26
27 # Example description
28 data['label'].iloc[0]
29
30 # Missing values and duplicate observations
31 print(pd.Series({"Number of observations with missing values":
       len(data) - len(data.dropna()),
32                    "Number of duplicate observations": data.
                          duplicated().sum()}).to_string())
33
34 data.dropna(inplace = True) # Dropping observations with
      missing values
35 data.drop_duplicates(inplace = True) # Dropping duplicate
      observations
36 data.reset_index(drop = True, inplace = True) # Resetting
      index
37
38 # Manual encoding of labels
39 label_dict = {'delivery service': 0, 'application': 1, '
      customer service': 2}
40 data.replace({'label': label_dict}, inplace = True)
41
42 print(pd.Series({"Memory usage": "{:.2f} MB".format(data.
      memory_usage().sum()/(1024*1024)),
43                    "Dataset shape": "{}".format(data.shape)}).
                          to_string())
```

```
print(data.tail())

##Exploratory data analysis
#Class Frequencies
# Splitting the dataset by label
data_d = data[data['label'] == 0] # delivery
data_a = data[data['label'] == 1] # application
data_c = data[data['label'] == 2] # customer service

# Visualization of class frequencies using Matplotlib
class_counts = [len(data_d), len(data_a), len(data_c)]
class_labels = ['Delivery service', 'Application performance',
    'Customer Service']

plt.pie(class_counts, labels=class_labels, autopct='%1.1f%%')
plt.title("Comparison of class frequencies")
plt.show()

# Distribution of number of characters in description
data_a_char = data_a['text'].str.len()
data_d_char = data_d['text'].str.len()
data_c_char = data_c['text'].str.len()

# Distribution of number of characters in description
fig, ax = plt.subplots(1, 3, figsize=(15, 5), sharey=False)
sns.histplot(x=data_a_char, bins=20, ax=ax[0]).set_title('
   Class: Application performance')
sns.histplot(x=data_d_char, bins=20, ax=ax[1]).set_title('
   Class: Delivery service')
```

```
70 sns.histplot(x=data_c_char, bins=20, ax=ax[2]).set_title('
      Class: Customer Service')
71 fig.suptitle("Distribution of number of characters in
      description")
72
73 for i in range(3):
74     ax[i].set_xlabel("Number of characters")
75     ax[i].set_ylabel("Count")
76 plt.show()
77
78
79 # Distribution of average word-length in text
80 data_a_avg = data_a['text'].str.split().apply(lambda x : [len(
      i) for i in x]).map(lambda x: np.mean(x))
81 data_d_avg = data_d['text'].str.split().apply(lambda x : [len(
      i) for i in x]).map(lambda x: np.mean(x))
82 data_c_avg = data_c['text'].str.split().apply(lambda x : [len(
      i) for i in x]).map(lambda x: np.mean(x))
83
84 fig, ax = plt.subplots(1, 3, figsize=(12, 4), sharey=False)
85 sns.histplot(x=data_a_avg, bins=20, ax=ax[0]).set_title('Class
      : Application performance')
86 sns.histplot(x=data_d_avg, bins=20, ax=ax[1]).set_title('Class
      : Delivery service')
87 sns.histplot(x=data_c_avg, bins=20, ax=ax[2]).set_title('Class
      : Customer Service')
88
89 fig.suptitle("Distribution of average word-length in
      description")
```

```
90 for i in range(3):
91     ax[i].set_xlabel("Average word-length")
92     ax[i].axvline(x=np.mean(data_a_avg), linestyle='--', color
          ='red')
93     ax[i].axvline(x=np.mean(data_d_avg), linestyle='--', color
          ='red')
94     ax[i].axvline(x=np.mean(data_c_avg), linestyle='--', color
          ='red')
95 plt.show()
```

APPENDIX C

```
1
2  # File system manangement
3  import time
4
5  # Data manipulation
6  import numpy as np
7  import pandas as pd
8
9  # Plotting and visualization
10 import seaborn as sns
11 sns.set_theme()
12
13
14 import plotly.graph_objects as go
15
16
17 # NLP
18 import string, re, nltk
19 from string import punctuation
20 from nltk.tokenize import word_tokenize, RegexpTokenizer
21 from nltk.corpus import stopwords
22
23 from spellchecker import SpellChecker
24 from nltk.stem.porter import PorterStemmer
25 import spacy
26 from nltk.stem import WordNetLemmatizer
```

```
27
28 # TF-IDF
29
30 from sklearn.feature_extraction.text import TfidfVectorizer
31
32 # Scipy
33 import scipy
34 from scipy import sparse
35 from scipy.sparse import csr_matrix
36
37 # Train-test split and cross validation
38 from sklearn.model_selection import train_test_split,
       ParameterGrid
39
40 # Classifiers
41 from sklearn.linear_model import LogisticRegression
42
43 from sklearn.tree import DecisionTreeClassifier
44 from sklearn import svm
45
46
47 # Model evaluation
48 from sklearn import metrics
49 from sklearn.metrics import accuracy_score
50
51 # Others
52 import json
53 import gensim
54 from gensim.models import Word2Vec
```

```
55
56
57 # Recording the starting time
58 start = time.time()
59
60 # Load data from Excel file
61 data = pd.read_csv("C:/Users/Safawi/Desktop/FYP CSV/SHOPEE
       REVIEWS TRAINING FINAL.csv",
62     names = ['text' , 'label']
63 )
64 data = data[['text', 'label']]
65
66 #Train and validation Test Split
67 # Feature-target split
68 X, y = data.drop('label', axis = 1), data['label']
69
70 # Train-test split (from complete data)
71 X_train, X_test, y_train, y_test = train_test_split(X, y,
       test_size = 0.2, random_state = 40)
72 data_train = pd.concat([X_train, y_train], axis = 1)
73
74 # Validation-test split (from test data)
75 X_val, X_test, y_val, y_test = train_test_split(X_test, y_test
       , test_size = 0.5, random_state = 40)
76 data_val, data_test = pd.concat([X_val, y_val], axis = 1), pd.
       concat([X_test, y_test], axis = 1)
77
78 # Comparison of sizes of training set, validation set and test
       set
```

```
values = np.array([len(data_train), len(data_val), len(
    data_test)])
labels = ['Training set', 'Validation Set', 'Test set']
fig = go.Figure(data = [go.Pie(values = values, labels =
    labels, hole = 0.5, textinfo = 'percent', title = " ")])
text_title = "Comparison of sizes of training set, validation
    set and test set"
fig.update_layout(height = 500, width = 800, showlegend = True
    , title = dict(text = text_title, x = 0.5, y = 0.95))
fig.show()


#Text Normalisation

# RegexpTokenizer
regexp = RegexpTokenizer("[\w']+")

#Converting to lowercase
def convert_to_lowercase(text):
    return text.lower()

# Removing whitespaces
def remove_whitespace(text):
    return text.strip()

# Removing punctuations
def remove_punctuation(text):
    punct_str = string.punctuation
```

```
103     punct_str = punct_str.replace("'", "") # discarding
            apostrophe from the string to keep the contractions
            intact
104     return text.translate(str.maketrans("", "", punct_str))
105
106 # Removing HTML tags
107 def remove_html(text):
108     html = re.compile(r'<.*?>')
109     return html.sub(r'', text)
110
111 # Removing emojis
112 def remove_emoji(text):
113     emoji_pattern = re.compile("["
114                                u"\U0001F600-\U0001F64F"  #
                                       emoticons
115                                u"\U0001F300-\U0001F5FF"  # symbols
                                       & pictographs
116                                u"\U0001F680-\U0001F6FF"  #
                                       transport & map symbols
117                                u"\U0001F1E0-\U0001F1FF"  # flags (
                                       iOS)
118                                u"\U00002702-\U000027B0"
119                                u"\U000024C2-\U0001F251"
120                                "]+", flags = re.UNICODE)
121     return emoji_pattern.sub(r'', text)
122
123 # Removing other unicode characters
124 def remove_http(text):
```

```
125     http = "https?://\S+|www\.\S+" # matching strings
            beginning with http (but not just "http")
126     pattern = r"({})".format(http) # creating pattern
127     return re.sub(pattern, "", text)
128
129 acronyms_url = 'https://raw.githubusercontent.com/sugatagh/E-
        commerce-Text-Classification/main/JSON/english_acronyms.
        json'
130 acronyms_dict = pd.read_json(acronyms_url, typ = 'series')
131
132 # Dataframe of acronyms
133 pd.DataFrame(acronyms_dict.items(), columns = ['acronym', '
        original']).head()
134
135 # List of acronyms
136 acronyms_list = list(acronyms_dict.keys())
137
138 def convert_acronyms(text):
139     words = []
140     for word in regexp.tokenize(text):
141         if word in acronyms_list:
142             words = words + acronyms_dict[word].split()
143         else:
144             words = words + word.split()
145
146     text_converted = " ".join(words)
147     return text_converted
148
149 # Dictionary of contractions
```

```
contractions_url = 'https://raw.githubusercontent.com/sugatagh/E-commerce-Text-Classification/main/JSON/english_contractions.json'
contractions_dict = pd.read_json(contractions_url, typ = 'series')

# Dataframe of contractions
pd.DataFrame(contractions_dict.items(), columns = ['contraction', 'original']).head()

# List of contractions
contractions_list = list(contractions_dict.keys())

# Function to convert contractions in a text
def convert_contractions(text):
    words = []
    for word in regexp.tokenize(text):
        if word in contractions_list:
            words = words + contractions_dict[word].split()
        else:
            words = words + word.split()

    text_converted = " ".join(words)
    return text_converted

# Stopwords
stops = stopwords.words("english") # stopwords
addstops = ["among", "onto", "shall", "thrice", "thus", "twice", "unto", "us", "would"] # additional stopwords
```

```
allstops = stops + addstops

# Function to remove stopwords from a list of texts
def remove_stopwords(text):
    return " ".join([word for word in regexp.tokenize(text) if
        word not in allstops])

# pyspellchecker
spell = SpellChecker()

def pyspellchecker(text):
    word_list = regexp.tokenize(text)
    word_list_corrected = []
    for word in word_list:
        if word in spell.unknown(word_list):
            word_corrected = spell.correction(word)
            if word_corrected == None:
                word_list_corrected.append(word)
            else:
                word_list_corrected.append(word_corrected)
        else:
            word_list_corrected.append(word)
    text_corrected = " ".join(word_list_corrected)
    return text_corrected

# Stemming
stemmer = PorterStemmer()
def text_stemmer(text):
```

```
201     text_stem = " ".join([stemmer.stem(word) for word in
            regexp.tokenize(text)])
202     return text_stem
203
204 # Lemmatization
205 spacy_lemmatizer = spacy.load("en_core_web_sm", disable = ['
        parser', 'ner'])
206 lemmatizer = WordNetLemmatizer()
207
208 def text_lemmatizer(text):
209     text_spacy = " ".join([token.lemma_ for token in
            spacy_lemmatizer(text)])
210     text_wordnet = " ".join([lemmatizer.lemmatize(word) for
            word in word_tokenize(text)])
211     return text_spacy + " " + text_wordnet
212
213 def discard_non_alpha(text):
214     word_list_non_alpha = [word for word in regexp.tokenize(
            text) if word.isalpha()]
215     text_non_alpha = " ".join(word_list_non_alpha)
216     return text_non_alpha
217
218 def keep_pos(text):
219     tokens = regexp.tokenize(text)
220     tokens_tagged = nltk.pos_tag(tokens)
221     keep_tags = ['NN', 'NNS', 'NNP', 'NNPS', 'FW', 'PRP', '
            PRPS', 'RB', 'RBR', 'RBS', 'VB', 'VBD', 'VBG', 'VBN', '
            VBP', 'VBZ', 'WDT', 'WP', 'WPS', 'WRB']
```

```
222     keep_words = [x[0] for x in tokens_tagged if x[1] in
            keep_tags]
223     return " ".join(keep_words)
224
225 # Additional stopwords
226
227 alphabets = ["a", "b", "c", "d", "e", "f", "g", "h", "i", "j",
        "k", "l", "m", "n", "o", "p", "q", "r", "s", "t", "u", "v"
        , "w", "x", "y", "z"]
228 prepositions = ["about", "above", "across", "after", "against"
        , "among", "around", "at", "before", "behind", "below", "
        beside", "between", "by", "down", "during", "for", "from",
        "in", "inside", "into", "near", "of", "off", "on", "out", "
        over", "through", "to", "toward", "under", "up", "with"]
229 prepositions_less_common = ["aboard", "along", "amid", "as", "
        beneath", "beyond", "but", "concerning", "considering", "
        despite", "except", "following", "like", "minus", "onto", "
        outside", "per", "plus", "regarding", "round", "since", "
        than", "till", "underneath", "unlike", "until", "upon", "
        versus", "via", "within", "without"]
230 coordinating_conjunctions = ["and", "but", "for", "nor", "or",
        "so", "and", "yet"]
231 correlative_conjunctions = ["both", "and", "either", "or", "
        neither", "nor", "not", "only", "but", "whether", "or"]
232 subordinating_conjunctions = ["after", "although", "as", "as
        if", "as long as", "as much as", "as soon as", "as though",
        "because", "before", "by the time", "even if", "even
        though", "if", "in order that", "in case", "in the event
        that", "lest", "now that", "once", "only", "only if", "
```

```
    provided that", "since", "so", "supposing", "that", "than",
     "though", "till", "unless", "until", "when", "whenever", "
    where", "whereas", "wherever", "whether or not", "while"]
233 others = ["  ", "  ", "  ", "  ", "  m  ", "    ", "    ", "
        ", "  re  ", "  ve  ", "    ", "  s  ", "  we  "]
234 additional_stops = alphabets + prepositions +
    prepositions_less_common + coordinating_conjunctions +
    correlative_conjunctions + subordinating_conjunctions +
    others
235
236 def remove_additional_stopwords(text):
237     return " ".join([word for word in regexp.tokenize(text) if
          word not in additional_stops])
238
239 def text_normalizer(text):
240     text = convert_to_lowercase(text)
241     text = remove_whitespace(text)
242     text = re.sub('\n' , '', text) # converting text to one
        line
243     text = re.sub('\[.*?\]', '', text) # removing square
        brackets
244     text = remove_http(text)
245     text = remove_punctuation(text)
246     text = remove_html(text)
247     text = remove_emoji(text)
248     text = convert_acronyms(text)
249     text = convert_contractions(text)
250     text = remove_stopwords(text)
251     text = pyspellchecker(text)
```

```
    text = text_lemmatizer(text) # text = text_stemmer(text)
    text = discard_non_alpha(text)
    text = keep_pos(text)
    text = remove_additional_stopwords(text)
    return text

# Implementing text normalization
data_train_norm, data_val_norm, data_test_norm = pd.DataFrame
    (), pd.DataFrame(), pd.DataFrame()

data_train_norm['normalized description'] = data_train['text'
    ].apply(text_normalizer)
data_val_norm['normalized description'] = data_val['text'].
    apply(text_normalizer)
data_test_norm['normalized description'] = data_test['text'].
    apply(text_normalizer)

data_train_norm['label'] = data_train['label']
data_val_norm['label'] = data_val['label']
data_test_norm['label'] = data_test['label']

print(data_train_norm)

#TF-IDF MODEL

# Features and labels

X_train_norm, y_train = data_train_norm['normalized
    description'].tolist(), data_train_norm['label'].tolist()
```

```
X_val_norm, y_val = data_val_norm['normalized description'].
    tolist(), data_val_norm['label'].tolist()
X_test_norm, y_test = data_test_norm['normalized description'
    ].tolist(), data_test_norm['label'].tolist()

# TF-IDF vectorization
TfidfVec = TfidfVectorizer(ngram_range = (1, 1))
X_train_tfidf = TfidfVec.fit_transform(X_train_norm)
X_val_tfidf = TfidfVec.transform(X_val_norm)
X_test_tfidf = TfidfVec.transform(X_test_norm)

# Classifiers
names = [
    "Logistic Regression",
    "Decision Tree",
    "Linear SVM"
]

models = [
    LogisticRegression(max_iter = 1000),
    DecisionTreeClassifier(),
    svm.SVC(kernel = 'linear')
]

# Function to return summary of baseline models
def score(X_train, y_train, X_val, y_val, names=names, models=
    models):
    score_df, score_train, score_val = pd.DataFrame(), [], []
    x = time.time()
```

```
302     for model in models:
303         model.fit(X_train, y_train)
304         y_train_pred, y_val_pred = model.predict(X_train),
                model.predict(X_val)
305         score_train.append(accuracy_score(y_train,
                y_train_pred))
306         score_val.append(accuracy_score(y_val, y_val_pred))
307 
308     score_df["Classifier"], score_df["Training accuracy"],
            score_df[
309          "Validation accuracy"] = names, score_train, score_val
310     score_df.sort_values(by='Validation accuracy', ascending=
            False, inplace=True)
311     return score_df
312 
313 # Summary of baseline models
314 print(score(X_train_tfidf, y_train, X_val_tfidf, y_val, names
        = names, models = models))
315 
316 #WORD2VEC
317 
318 def convert_to_lowercase(text): return text.lower()
319 
320 contractions_url = 'https://raw.githubusercontent.com/sugatagh
        /E-commerce-Text-Classification/main/JSON/
        english_contractions.json'
321 contractions_dict = pd.read_json(contractions_url, typ='series
        ')
322 contractions_list = list(contractions_dict.keys())
```

```
323
324 def convert_contractions(text):
325     words = []
326     for word in regexp.tokenize(text):
327         if word in contractions_list:
328             words = words + contractions_dict[word].split()
329         else:
330             words = words + word.split()
331     return " ".join(words)
332
333 # Text normalization for Word2Vec
334 for df in [data_train, data_val, data_test]:
335     df['tokens'] = (df["text"].apply(convert_to_lowercase)
336                                       .apply(
                                           convert_contractions)
337                                       .apply(regexp.tokenize))
338 print(data_train[['tokens', 'label']])
339
340 # Loading the pre-trained Word2Vec model
341 word2vec_path = "C:/Users/Safawi/Desktop/FYP CSV/GoogleNews-
    vectors-negative300.bin"
342 word2vec = gensim.models.KeyedVectors.load_word2vec_format(
    word2vec_path, binary=True)
343
344 # Some useful functions for Word2Vec
345 def get_average_word2vec(tokens_list, vector, generate_missing
    =False, k=300):
346     if len(tokens_list) < 1:
347         return np.zeros(k)
```

```
348     if generate_missing:
349         vectorized = [vector[word] if word in vector else np.
               random.rand(k) for word in tokens_list]
350     else:
351         vectorized = [vector[word] if word in vector else np.
               zeros(k) for word in tokens_list]
352     length = len(vectorized)
353     summed = np.sum(vectorized, axis=0)
354     averaged = np.divide(summed, length)
355     return averaged
356 
357 def get_word2vec_embeddings(vectors, tokens, generate_missing=
    False):
358     embeddings = tokens.apply(lambda x: get_average_word2vec(x
           , vectors, generate_missing=generate_missing))
359     return list(embeddings)
360 
361 # Word2Vec embedding
362 X_train_embed = get_word2vec_embeddings(word2vec, data_train['
    tokens'])
363 X_val_embed = get_word2vec_embeddings(word2vec, data_val['
    tokens'])
364 X_test_embed = get_word2vec_embeddings(word2vec, data_test['
    tokens'])
365 
366 # Converting to Compressed Sparse Row matrix
367 X_train_w2v = scipy.sparse.csr_matrix(X_train_embed)
368 X_val_w2v = scipy.sparse.csr_matrix(X_val_embed)
369 X_test_w2v = scipy.sparse.csr_matrix(X_test_embed)
```

```
370
371 # Summary of baseline models
372 print(score(X_train_w2v, y_train, X_val_w2v, y_val, names =
      names, models = models))
373
374
375 best_model_w2v.fit(X_train_w2v, y_train)
376 y_val_pred_probabilities = best_model_w2v.predict_proba(
      X_val_w2v)
377
378
379 print(y_val_pred_probabilities)
380
381 # Fit the best model on the training data
382 best_model_w2v.fit(X_train_w2v, y_train)
383
384 # Predict labels for validation data
385 y_val_pred_w2v = best_model_w2v.predict(X_val_w2v)
386
387 # Create a DataFrame to store the original, predicted labels,
      and reviews
388 results_df_w2v = pd.DataFrame({'Original_Label': y_val, '
      Predicted_Label': y_val_pred_w2v, 'Reviews': data_val['text
      ']})
389
390 # Save the DataFrame to a CSV file
391 results_df_w2v.to_csv('predicted_labels_reviews_w2v.csv',
      index=False)
392
```

```
393
394 unique_classes_y_train = np.unique(y_train)
395 unique_classes_y_test = np.unique(y_test)
396
397 print("Unique classes in y_train:", unique_classes_y_train)
398 print("Unique classes in y_test:", unique_classes_y_test)
399 print("Shape of y_train:", len(y_train))
400 print("Shape of y_test:",len(y_test))
401
402 # Complemented with a stopping time check
403 end = time.time()
404 # Compute process runtime
405 runtime = end - start
406
407 # Print the runtime
408 print("Process runtime:", runtime, "seconds")
```