University of Essex

Department of Mathematical Sciences

MA981 DISSERTATION

Feedback Analytics Suite

Ishita Agarwal

Registration Number: 2201110

Supervisor: **Osama Mahmoud**

August 19, 2023
Colchester

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

[1]In today's complex business environment, customer feedback stands out as a pivotal tool for understanding and meeting the ever-evolving needs of consumers. This feedback, a reflection of customer perceptions and expectations, offers invaluable insights that businesses can utilize to better align their products and services. Beyond just a collection of comments, feedback serves as a direct communication bridge between customers and companies. Through this bridge, concerns, suggestions, and commendations are shared, providing businesses with a clear roadmap for improvement. As the marketplace becomes increasingly saturated, companies that harness and act upon customer feedback are better positioned to distinguish themselves from competitors. This is because integrating customer insights into business strategies not only ensures products and services remain relevant, but it also cements a company's reputation as customer-centric. Furthermore, a continuous commitment to gathering and analyzing feedback enables organizations to identify and act on trends, preemptively addressing potential challenges and ensuring a consistently high-quality customer experience. In essence, customer feedback is not just a reactive tool but a strategic asset, one that can drive business growth, elevate customer satisfaction, and unlock unparalleled opportunities in the modern market landscape.

1.1 Background

Natural Language Processing (NLP) has emerged as an indispensable tool in deciphering vast amounts of unstructured customer feedback in the digital age. [2]Its roots can be

traced back to the 1950s when Alan Turing introduced the 'Turing Test' as a measure of a machine's potential to mimic human intelligence, which was followed by the "Georgetown experiment" in 1954, illustrating early machine translation capabilities. The 1970s witnessed a shift towards NLP systems that emphasized formal grammars, aiming to understand the intricate structures of language. As computational power and data availability surged in the latter part of the 20th century, the NLP field transitioned from rule-based approaches to more dynamic, data-driven methods, with machine learning becoming integral. The turn of the millennium brought forth sophisticated NLP tasks and the integration of advanced deep learning architectures, such as Recurrent Neural Networks (RNNs) and Transformers. Despite these advancements, contemporary NLP still faces challenges in comprehending the nuanced and contextual aspects of human language, underscoring the ongoing quest to enhance machine understanding of human communication

This realm of feedback, encapsulating text comments, social media interactions, and online reviews, remains dense and nebulous without the application of sophisticated techniques to understand its nuances. The indispensability of NLP in this context is accentuated by its ability to process and interpret textual data, converting raw text into actionable insights. One of the foremost applications of NLP in customer feedback analysis is sentiment analysis. This process allows businesses to gauge the underlying sentiment in feedback whether it exudes positivity, negativity, or neutrality by critically evaluating the language nuances and tonal indicators. Moreover, NLP facilitates the extraction of salient topics from feedback, enabling businesses to categorize and subsequently delve deeper into themes or areas that customers emphasize, thus aiding strategic prioritization. The strengths of NLP are manifold in the business context. These insights not only influence decision-making but also act as catalysts for strategic endeavors that bolster customer satisfaction and loyalty. Furthermore, the scalability of NLP offers an unparalleled advantage; businesses can meticulously analyze substantial data volumes, transcending the constraints of manual analysis and ensuring a more holistic understanding of customer sentiment. The role of NLP in enhancing the personalization of customer experiences cannot be overstated. By sifting through historical feedback and interactions, organizations can tailor their offerings, ensuring alignment with individual customer predilections. Such personalization invariably elevates cus-

tomer satisfaction. An additional, often underemphasized, facet of NLP is its capacity for real-time feedback surveillance. Automated alerts, driven by specific keywords or sentiments, equip businesses with the agility to respond to emergent customer concerns swiftly, reinforcing their commitment to superior customer service. In summation, NLP stands as an indispensable ally for businesses in the modern era, transforming the often daunting task of customer feedback analysis into a strategic advantage.

CHAPTER 2

Literature Review

In today's competitive market, a key determinant of business success lies in grasping customer behavior and prioritizing customer satisfaction. These aspects hold utmost importance for any company striving to excel in their industry.

2.1 Related Works

[3] the pivotal role customer feedback plays, emphasizing that insights into customer sentiment and its underlying causes can be transformed into actionable interventions, potentially steering businesses toward earning unwavering customer loyalty or facing customer churn. Moreover, this paper delves into the synergistic integration of Deep Learning and Natural Language Processing (NLP) as a powerful mechanism for making contextual and informed decisions. Analyzing customer perceptions necessitates intricate mining of data, a task aptly addressed by employing NLP techniques. However, the application of NLP in this realm is not a straightforward endeavor; it demands an in-depth understanding of business perspectives, necessitating the application of techniques ranging from rule-based methodologies to Named Entity Recognition.There is also need for more profound exploration of Natural Language Processing (NLP) technologies,also promoting an investigative approach towards uncovering relevant deep learning modalities, advocating the deployment of advanced models like Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) with the aim of increasing accuracy in sentiment and perception analysis.

[4] delves deep into the potential of machine learning models, with a special emphasis on the Natural Language Processing (NLP) techniques, for the analysis and categorization of datasets populated with diverse reviews. The paper sheds light on the intricate challenges encountered, the complexities of negation, sarcasm, and word ambiguity. In the technical terrain of the research, two classifiers, namely SVM , emerged as the primary tools of approach. Furthermore, the intricacies of deciphering sentiments that are couched in tones of negation or sarcasm were meticulously examined. One of the key highlights of this study is its introduction of a context-based sentiment analysis model, ingeniously employing the power of Bi-LSTM, fastText, and LSVM.The empirical results from the model training exhibited the clear superiority of fastText over its contemporaries, LSVM and SA-BLSTM, particularly when confronted with extensive datasets on servers with minimal configurations. Boasting an impressive accuracy rate of 90.71%, fastText's prowess was evident. Moreover, in terms of methodology, the n-gram technique, especially the trigram (n=3), showcased remarkable efficacy for both fastText and LSVM with respect to the chosen dataset. The paper also underscores the elevated accuracy achieved in sentiment scores when domain-specific datasets are trained and subsequently tested within that specific domain. In summation, the authors draw attention to the fact that fastText emerged as a robust contender in model training, registering rapid response times and superior accuracy in comparison to LSVM and SA-BLSTM.

[5] focuses on nuances of sentiment polarity, encompassing categories such as positive, negative, or neutral emotional responses, whether the emotional inclination is oriented towards the entirety of a document, localized at the sentence level, or specifically targeted at distinct entities within the text.[5] underscores the importance to undertake comprehensive experiments both sentence-level and review-level categorizations of sentiment.

[6]published a paper to talk about importance of sentimental analysis and difficulties associated with it . The paper discusses the challenges faced in sentiment analysis, such as dealing with informal writing styles and sarcasm. The discussion to the categorization systems of sentiment, comparing them based on their respective strengths and weaknesses. It highlights the frequent use of supervised machine learning techniques due to their simplicity and high accuracy, with Naive Bayes (NB) and Support Vector

Machine (SVM) algorithms often serving as the benchmark for new methods. The paper ends by recommending that future research continuously widen the comparison area with new findings, reiterating the fact that sentiment analysis is still a mostly unexplored field of study with many obstacles to overcome.

A pivotal work in sentiment analysis using the backdrop of social networking sites is presented in [7] The paper elaborates on several preprocessing necessities, the removal of URLs from the data, the purging of special characters and the omission of question words. Such preprocessing steps, as highlighted by the study, are foundational to ensuring the robustness and accuracy of sentiment analysis within the social networking sites.

[8] paper explores sentiment analysis on Twitter, acknowledging its potential as a rich source of diverse opinions. The focus here is on data preprocessing, a critical step in sentiment analysis, as the right preprocessing methods can enhance correct classification. The results validate that feature selection and representation can positively impact classification performance. . The performance of four well-known classifiers was evaluated. Findings reveal that with the right feature selection and representation, sentiment analysis accuracy's can be improved. [8] focuses on further exploration of preprocessing options to identify optimal settings, future research should consider embedded methods, which perform feature selection and model tuning simultaneously.

[9] primarily focuses on how crucial is to do sentimental analysis to prevent the dissemination of negative reviews. It allows businesses to detect and address any negative sentiments in reviews before they are published or circulated widely. This proactive approach are crucial in safeguarding a brand's positive image. and ensures that customers' concerns are addressed promptly, which ultimately contributes to improved customer satisfaction. The initial step in SA is preprocessing the data. Central to the sentiment analysis framework is the step of preprocessing the data. The study undertook a comparative analysis of various preprocessing techniques, aiming to identify an optimal combination. However, the authors caution that a singular preprocessing approach may not be universally effective. Thus, they advocate for the application of similar processes across diverse datasets to gain a holistic understanding of the efficacy of different preprocessing filters

2.2 Objective

Drawing from the collective insights of the surveyed papers,there emerges a consistent recommendation for the rigorous exploration of preprocessing steps in conjunction with deep learning models to bridge identified research gaps.Enhanced accuracy in sentiment analysis is envisaged to offer a more nuanced discernment of emotions such as sadness, happiness, and anger.The literature further underscores the potential of advanced NLP techniques, specifically Long Short-Term Memory networks (LSTM). These networks are adept at processing sequences, making them particularly suitable for sentiment analysis where understanding the sequence of words is critical.

CHAPTER 3

Data collection and feature selection

3.1 Summary of the data

The research includes data set from Kaggle and is tailored to probe sentiments derived from Spotify app reviews. Comprising reviews collated from the Google Play Store between January 1, 2022, and July 9, 2022, the data set is both extensive and detailed, containing a total of 61,594 entries. It is structured with columns representing the time of submission (Time_submitted), the content of the review (Review), the user rating (Rating), the total number of thumbs up the review garnered (Total_thumbsup), and any associated replies (Reply). [10] As of March 2022, Spotify's footprint in the music streaming domain is considerable, with a user base exceeding 422 million monthly active users, and among these, 182 million are paying subscribers. Such significant numbers emphasize the platform's widespread appeal and also suggest a vast and varied community of users. A sizeable segment of this user community actively shares their experiences and assigns ratings to the application, providing a rich avenue to assess user sentiments and overall satisfaction with the service.

3.2 Methods of Feature Selection

3.2.1 Stop words Removal

Stop words are frequently occurring words in a language that don't contribute much to the meaning in natural language processing (NLP) tasks. As such, they are often eliminated during pre processing. English stop words include terms like "the," "and," "is," "in," "a," "an," and so on. By omitting these, we can decrease the dataset size, facilitating quicker preprocessing and potentially enhancing accuracy. This removal can also enhance the relevance of results in information retrieval tasks. The Natural Language Toolkit (NLTK) is a Python library tailored for NLP operations. It simplifies the process of stop word removal by offering a predefined list, eliminating the need for manual definition. With more than 150 predefined stop words available in multiple languages, the NLTK makes it convenient to address this aspect of pre processing. To enhance the model capability,critical aspect of this process involves the elimination of uninformative words, which can hinder the accuracy and efficiency of subsequent analyses. To gain a deeper understanding of the datasetś lexical composition, the Counter function was employed to determine word frequencies. Recognizing the uniqueness of our dataset, custom stopwords words that offer limited analytical value were curated and integrated. This is in addition to the established list of common stopwords. This encompassed the removal of the uninformative words and the application of lemmatization techniques. Lemmatization simplifies words to their base form, ensuring uniformity and reducing variability caused by different word forms. Postpreprocessing, cleaned data is more suitable for further analysis, such as sentiment analysis or topic modeling.

3.2.2 Noise removal

In text analytics, converting all text to lowercase is a common and strategic practice. The main reason for this is consistency. For a computer, "TEXT" and "text" are seen as two different words, which can distort word count results. By making everything lowercase, the data becomes simpler and more consistent. This is especially important when working with informal text, like social media, where people may use varying capitalizations. Another advantage is that it makes combining and comparing texts

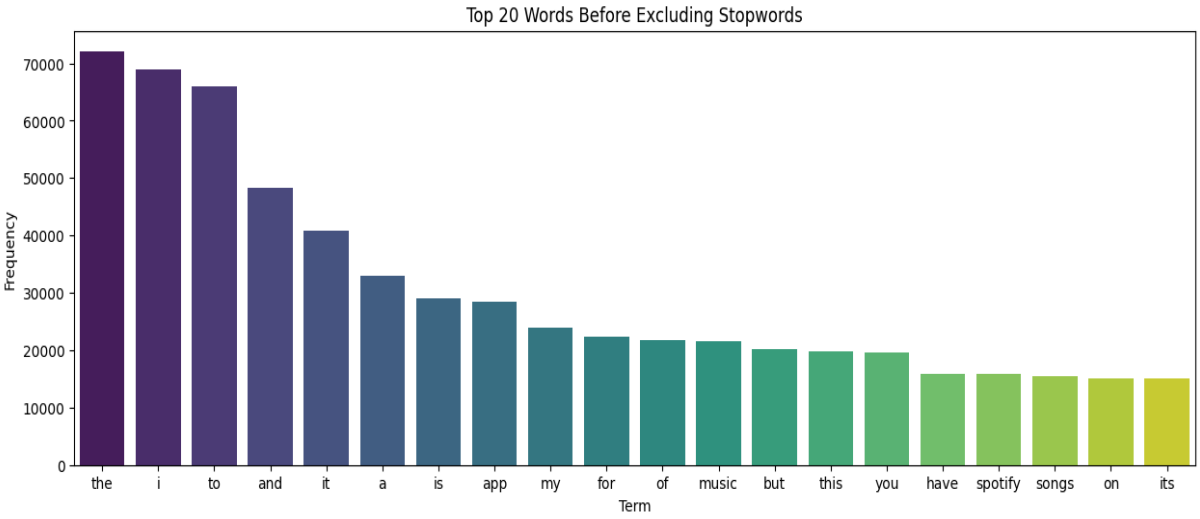

Figure 3.1: Word Frequencies before Stopwords Elimination

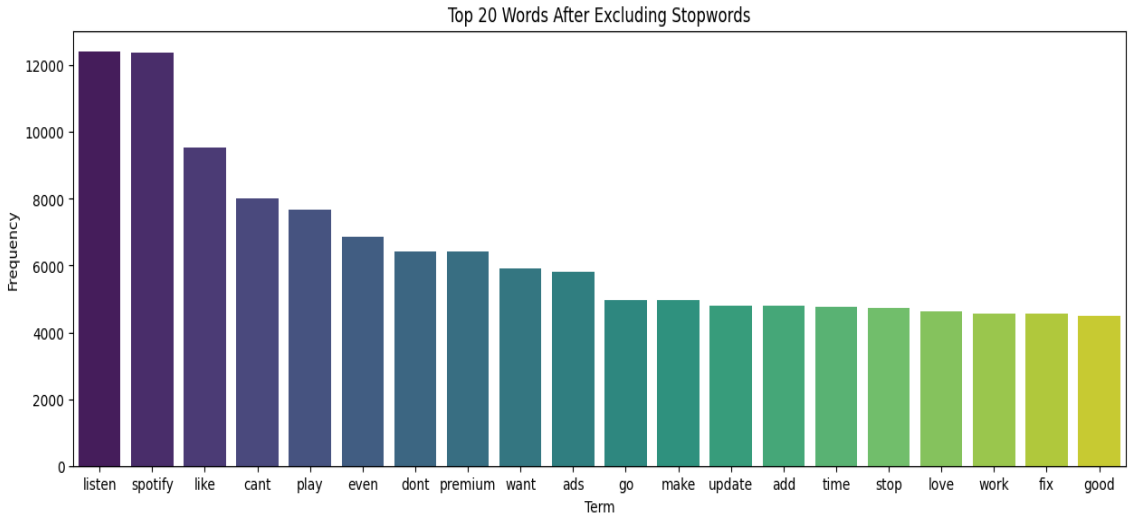

Figure 3.2: Word Frequencies after Stopwords Elimination

from different sources easier, as you won't miss matches just because of different capitalizations. Even though the focus is on understanding the main ideas of the text, sometimes the way a word is capitalized can be meaningful, so there are cases where the original format should be kept. Overall, using lowercase is a way to keep data clean and analysis accurate.

The dataset undergoes a series of noise reduction processes aimed at refining the 'Review' column. Initially, any URLs present, whether they begin with 'http' or 'www', are located and subsequently erased, ensuring that such web addresses, which typically do not contribute to the semantic content, are removed. The dataset is then further

cleansed by removing mentions that are often found in social media texts; these are represented by the '' symbol followed by characters, including potential hashtags that commence with the '#' symbol. Additionally, mentions that specifically begin with 'user' are eliminated, which could be a placeholder or a frequent mention pattern. To bolster the cleanliness of the data, special characters are removed, with a few exceptions like hashtags, '' etc. The word 'user', which can sometimes be a default placeholder or remnant from earlier cleaning steps, is also expunged. As a checkpoint for these preprocessing endeavors, the first few entries of the 'Review' column are printed, giving a glimpse into the effectiveness of the noise removal process

3.2.3 The Benefits of Lemmatization over Stemming

Stemming is the process of lowering inflection towards their root forms; it takes place in such a way that a collection of related words are depicted under the same stem, even when the root does not have the right meaning.In 3.3displays the appearance of words post-stemming.

```
[nltk_data]   Package stopwords is already up-to-date!
Original: great, Stemmed: great
Original: service, Stemmed: servic
Original: audio, Stemmed: audio
Original: high, Stemmed: high
Original: quality, Stemmed: qualiti
Original: easy, Stemmed: easi
Original: use, Stemmed: use
Original: also, Stemmed: also
Original: quick, Stemmed: quick
Original: friendly, Stemmed: friendli
Original: support, Stemmed: support
Original: please, Stemmed: pleas
Original: ignore, Stemmed: ignor
Original: previous, Stemmed: previou
Original: negative, Stemmed: neg
Original: rating, Stemmed: rate
Original: super, Stemmed: super
Original: great, Stemmed: great
Original: give, Stemmed: give
Original: five, Stemmed: five
<ipython-input-32-ee8a8cd52d88>:39: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

Figure 3.3: Visualization of Word Roots Obtained Through Stemming

Lemmatization is similar to stemming, but it is more time-consuming because it aims to derive a meaningful word or representation. To achieve the correct lemma, specifying the part of speech might be necessary. The word's meaning is crucial, especially for sentiment analysis, which is why we opt for lemmatization.3.4 displays the appearance of words after lemmatization [11].

```
[nltk_data]   Package wordnet is already up-to-date!
Original: great, Lemmatized: great
Original: service, Lemmatized: service
Original: audio, Lemmatized: audio
Original: high, Lemmatized: high
Original: quality, Lemmatized: quality
Original: easy, Lemmatized: easy
Original: use, Lemmatized: use
Original: also, Lemmatized: also
Original: quick, Lemmatized: quick
Original: friendly, Lemmatized: friendly
Original: support, Lemmatized: support
Original: please, Lemmatized: please
Original: ignore, Lemmatized: ignore
Original: previous, Lemmatized: previous
Original: negative, Lemmatized: negative
Original: rating, Lemmatized: rat
Original: super, Lemmatized: super
Original: great, Lemmatized: great
Original: give, Lemmatized: give
Original: five, Lemmatized: five
<ipython-input-33-cb13835348a9>:35: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

Figure 3.4: Visualization of Word Roots Obtained Through Lemmatization

3.2.4 Lemmatization Techniques

Wordnet Lemmatizer

[12]WordNet is a comprehensive lexical database for the English language that provides structured semantic relationships between words. Freely available to the public, it is among the earliest and most popular tools with lemmatization capabilities.This unique approach captures nuances in meanings, allowing for a richer understanding of word interrelations.

Wordnet Lemmatizer (with POS tag)

The WordNet lemmatization methods certain outcomes were notably suboptimal. Specifically, words such as "sitting" and "flying" were not altered during the lemmatization process.A deeper analysis revealed that these discrepancies arose due to the contextual interpretation of these words. Within the framework of the provided sentences, these words were treated as nouns rather than verbs, leading to the observed anomalies in the lemmatization results.To enhance the accuracy of the lemmatization process,the use of Part of Speech (POS) tagging. By incorporating POS tags, words are lemmatized based on their contextual role in the sentence, rather than relying solely on

their inherent lexical category.

SpaCy Lemmatizer (with POS tag)

A distinct advantage of the SpaCy lemmatizer is its ability to detect pronouns effectively. This feature played a pivotal role in achieving a more nuanced and accurate lemmatization outcome.Furthermore, the lemmatizer showcased its proficiency in dealing with comparative adjectives. An illustrative instance of this was the conversion of the word "best" to its base form, "good".The adoption of the SpaCy lemmatizer not only rectified the inconsistencies observed in previous methods but also paved the way for a more comprehensive and semantic-based lemmatization.

TextBlob Lemmatizer

By employing TextBlob objects, users can seamlessly parse and lemmatize entire sentences or individual words. This capability offers a significant advantage in refining the quality of textual data.[13]

3.2.5 Vectorization methods

[14] Vectorization is the process of transforming words or phrases into numerical values. Fundamental to Natural Language Processing (NLP), this method involves associating words or phrases from a given vocabulary with distinct vectors of real numbers. Through these numerical encodings including word prediction, similarity computation between words, and others, become feasible.
1.Tokenization
2.Vocabulary creation
3.Vector creation

Term Frequency - Inverse Document Frequency (TF-IDF)

TF-IDF is a smart way that computers use to understand which words are the most important in a piece of text. There are two steps to figure this out:
1. "Term Frequency" is like counting how many times each word shows up in an essay. If a word comes up a lot, it might be important.
2. "Inverse Document Frequency" is like looking at all the essays and checking how

often the word shows up in all of them. If the word is in almost every essay, it's not so special. But if it only shows up in a few, it's a big deal. [15] Multiplication of two numbers together for each word. In figure 3.5. The higher the number, the more special that word is to the essay.

Term Frequency (TF)

The term frequency $TF(t, d)$ of term t in document d is defined as:

$$TF(t, d) = \frac{\text{Number of times term } t \text{ appears in document } d}{\text{Total number of terms in document } d} \tag{3.1}$$

Inverse Document Frequency (IDF)

The inverse document frequency $IDF(t, D)$ of term t in a document set D is defined as:

$$IDF(t, D) = \log \frac{\text{Total number of documents in } D}{\text{Number of documents containing term } t} \tag{3.2}$$

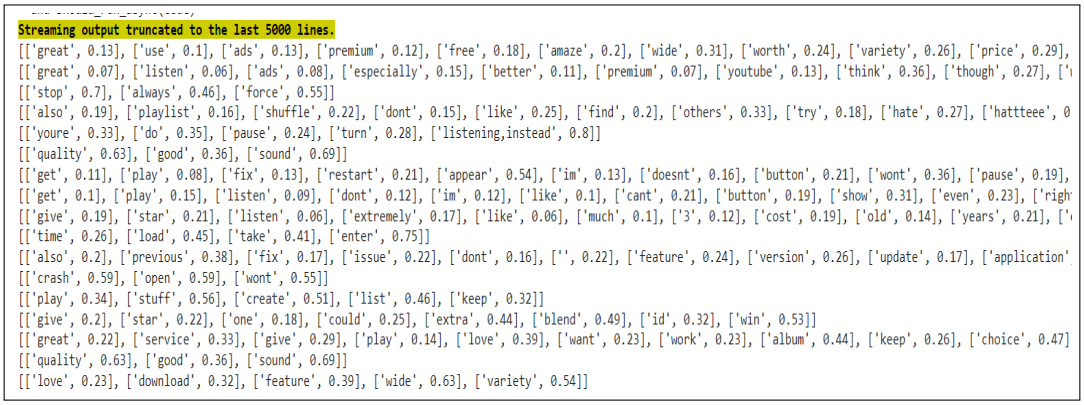

Figure 3.5: Visualization of TF-IDF scores

The word is a term in the document. The score is the TF-IDF score for that term in the document. For example:

- The term "great" has a TF-IDF score of 0.13.
- The term "use" has a TF-IDF score of 0.1.
- The term "ads" has a TF-IDF score of 0.13.

High TF-IDF scores are indicative of words that are important in a specific document, while being comparatively uncommon in the rest of the document corpus. For example, in the last document, the term "stop" has a high TF-IDF score of 0.7, indicating it's quite significant in that document, and doesn't appear as frequently across other documents.

Bag of Words(BoW)

Bag of Words (BoW) is a technique of text modelling counts the occurrence of words within a document disregarding the grammar details or any details about the structure or order of words.Bag of words is useful for converting text in documents to a fixed length vector. The key difference between bag of words and TF-IDF is that the former does not incorporate any sort of inverse document frequency (IDF) and is only a frequency count (TF).[16]

N-grams

To elucidate the concept of vectorization, let's delve into a practical example.
Consider two sentences:
1. "I love my dog. He's the best."
2. "I don't love the heat."
From these,extract the following distinct words: - love, dog, best, don't, heat.
When these sentences are vectorized based on the presence of the aforementioned words, we obtain:
1. "I love my dog. He's the best." translates to [1,1,1,0,0].
2. "I don't love the heat." translates to [1,0,0,1,1].
It is noteworthy that even though the first sentence has a positive sentiment and the second one bears a negative tone, the vector representations don't inherently reflect these sentiments. To tackle this representation challenge, the concept of n-grams is introduced.
An n-gram is essentially a contiguous sequence of 'n' words. For instance:
- A 2-gram (bigram) encompasses sequences like "don't love" or "the best".
- A 3-gram (trigram) might cover sequences such as "I love my" or "don't love the".
Considering bigrams for the sentence, "I don't love the heat", we obtain: - "I don't"
- "don't love"

- "love the"
- "the heat"

Integrating bigrams, as demonstrated, imparts more context to our tokens. This enriched representation allows our model to differentiate better between the sentiments of Sentence 1 and Sentence 2. Utilizing bigrams or trigrams thus provides a more meaningful tokenization that can capture nuanced sentiments and relations between words more effectively.In the figure 3.6 we can see how in our dataset bigram words are imparted

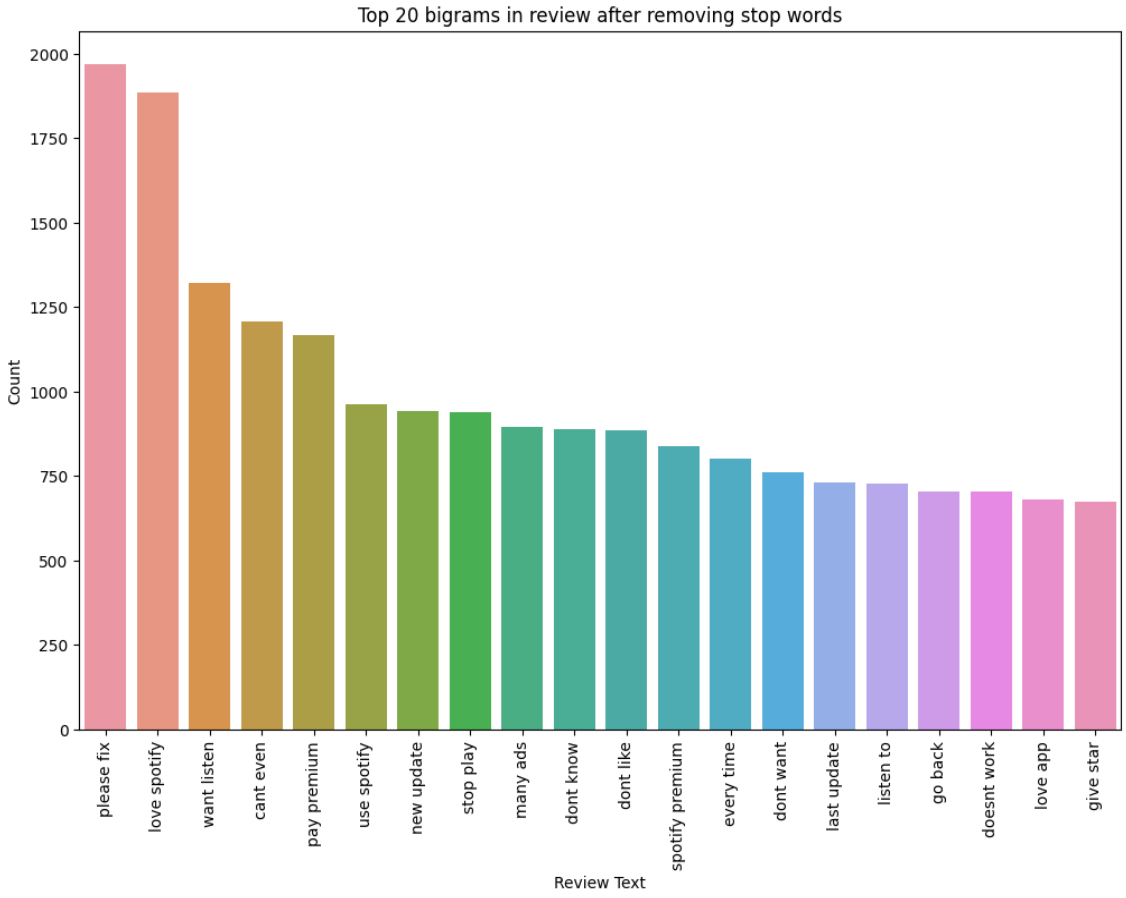

Figure 3.6: Visualization of Bigram Word Sequences

Word2Vec

[17] Word2Vec, utilizing a two-layered neural network, processes a given corpus to produce distinct vector representations for each word. This often necessitates further steps to amalgamate these individual vectors into a coherent representation for broader textual units like sentences or documents. Conversely, TF-IDF directly computes vectors

based on word frequencies within a document relative to their occurrence across multiple documents. A salient distinction between the two lies in their treatment of context. Word2Vec inherently captures the context in which words are positioned, emphasizing semantic relationships. In contrast, TF-IDF primarily hinges on word frequency, sidelining the contextual and semantic nuances of word usage. This comparative analysis underscores the need to judiciously select a vectorization approach that aligns with the specific objectives of an NLP endeavor.

3.2.6 Comparative Analysis: Label Encoder vs. One-Hot Encoder in Data Preprocessing

Label Encoding: Label encoding is a method of transforming categorical values into numeric format. It is used for ordinal data, where the categories have a natural order or ranking. Example: Category: ["Red", "Orange"] Label Encoding: [0, 1, 2]
One-Hot Encoding: One-hot encoding represents categorical variables as binary vectors. for each category, a new binary column is created, and only one of the columns has a value of 1 while the other columns have a value of 0. Example: Category: ["Red", "blue", "orange"]
One-Hot Encoding:

Red	Blue	Orange	Result
1	0	0	1
2	0	1	0
3	1	0	0

Table 3.1: Label Encoder

In figure 3.7 we observe the distribution of ratings prior to encoding, and then employ the Label Encoder to transform them into three distinct categories, as illustrated in 3.8

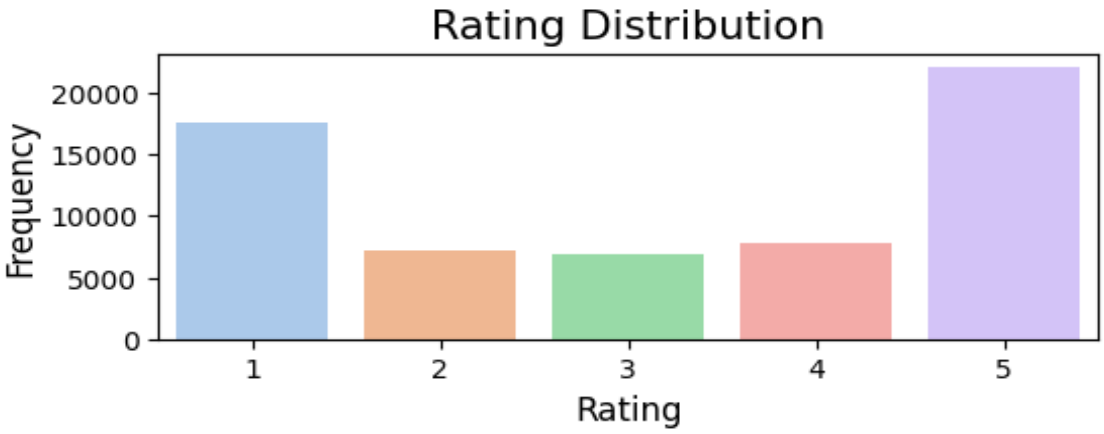

Figure 3.7: Rating distribution

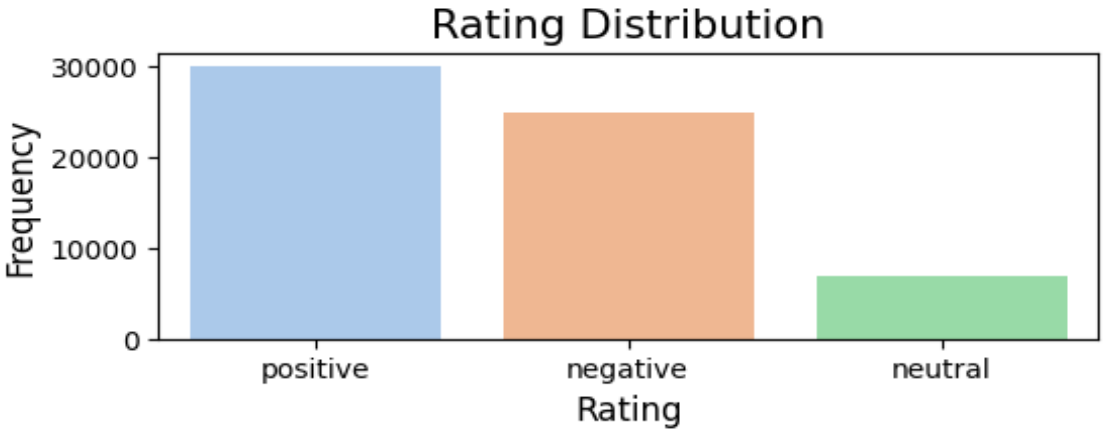

Figure 3.8: Distribution of ratings post-encoding.

3.3 Data Visualization

3.3.1 Topic Modelling

[18]Topic modeling is a type of statistical modeling to identify topics within a document. This is a text mining technique to gain insights to identify common themes to provide insights and understand the content.

The two primary topic modelling techniques that analyse big text files to categorise subjects, offer insightful information, and promote better decision-making are latent semantic analysis and latent Dirichlet analysis [19].

Latent Dirichlet Allocation(LDA)

The Variational Expectation Maximization (VEM) technique was utilized to derive the most probable estimate from the entire text corpus. Conventionally, one could address this issue by choosing the most frequent words from the bag of words. Yet, this method omits the semantic essence of the sentence. This model operates on the premise that any given document can be characterized by each topic and the probabilistic distribution of words within that topic.

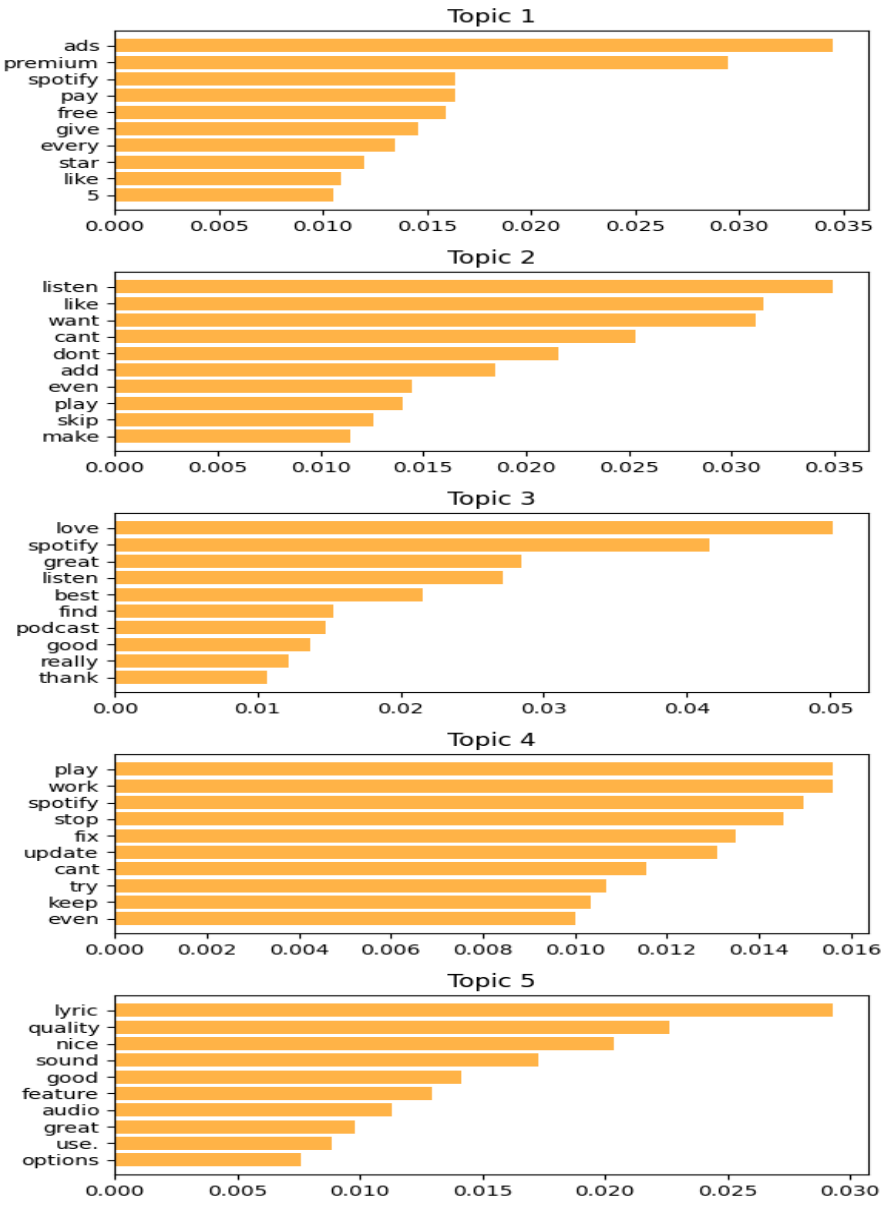

Figure 3.9: Topic modelling

Latent semantic analysis

Latent Semantic Analysis (LSA) is a technique in computational linguistics that identifies and captures the primary ideas within a text. The foundational belief of LSA is that words with comparable meanings often co-occur within similar contexts.

One application of LSA is in crafting tailored user experiences by constructing user profiles. Drawing from data sources like demographic details, historical transactions, and past user behaviors, organizations can categorize users into distinct profiles based on shared attributes.

By analyzing the context and occurrence frequency of words within documents, LSA discerns the semantic connections between them. In practice, users exhibiting similar behaviors and interaction trends are clustered into specific topics or categories. Utilizing unsupervised machine learning techniques, LSA automatically identifies unique topics from given datasets.

The underlying principle of LSA is that documents with analogous word frequency patterns are likely to share themes, allowing for the analysis of multiple documents concurrently. Being an integral part of natural language processing, LSA plays a pivotal role in understanding and decoding human language. Its applications are diverse, spanning from document classification to image categorization.

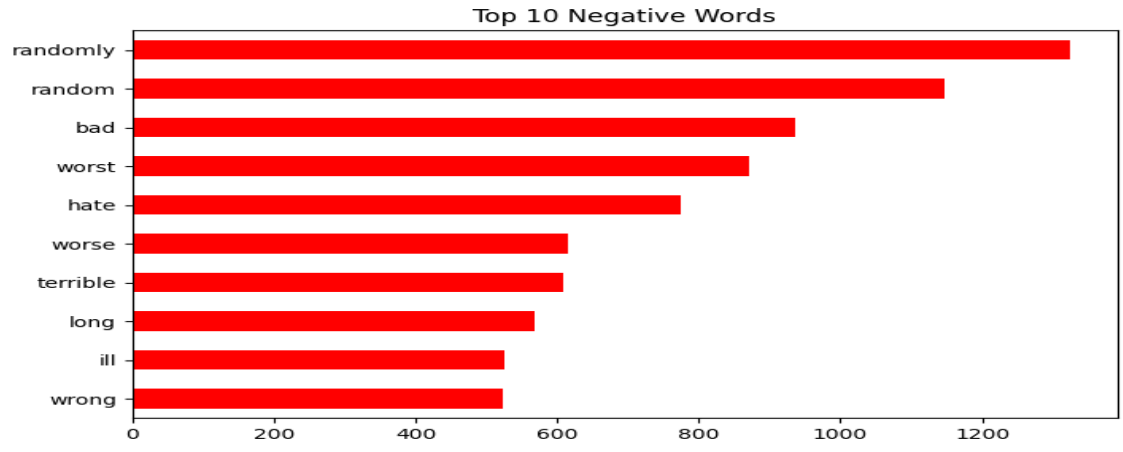

Figure 3.10: Top 10 negative terms

In Figure 3.11, reviews labeled as 'positive' are characterized by words exhibiting a positive sentiment, i.e., those with a sentiment polarity exceeding 0. The analysis excludes proper nouns, identified by the tag "NNP", to hone in on words with inherently

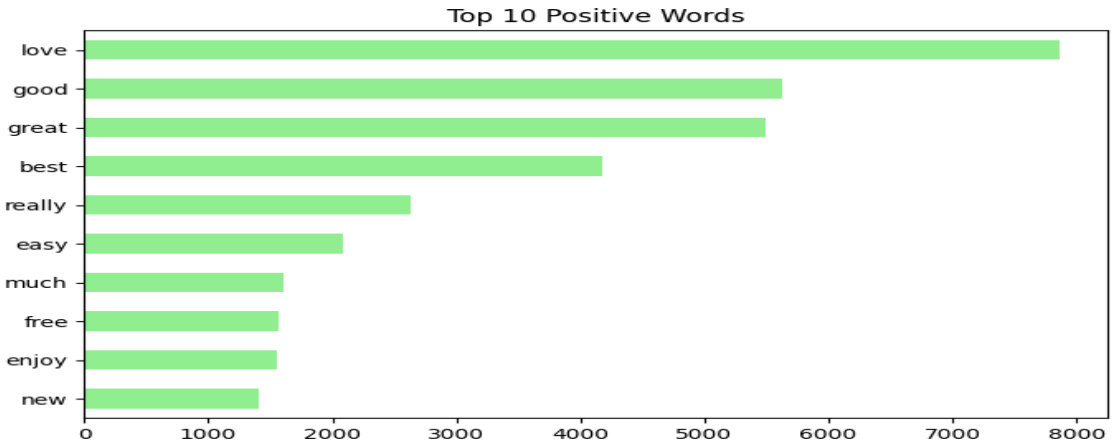

Figure 3.11: Top 10 positive terms.

positive connotations. Conversely, Figure 3.10 targets the opposite sentiment attributes.

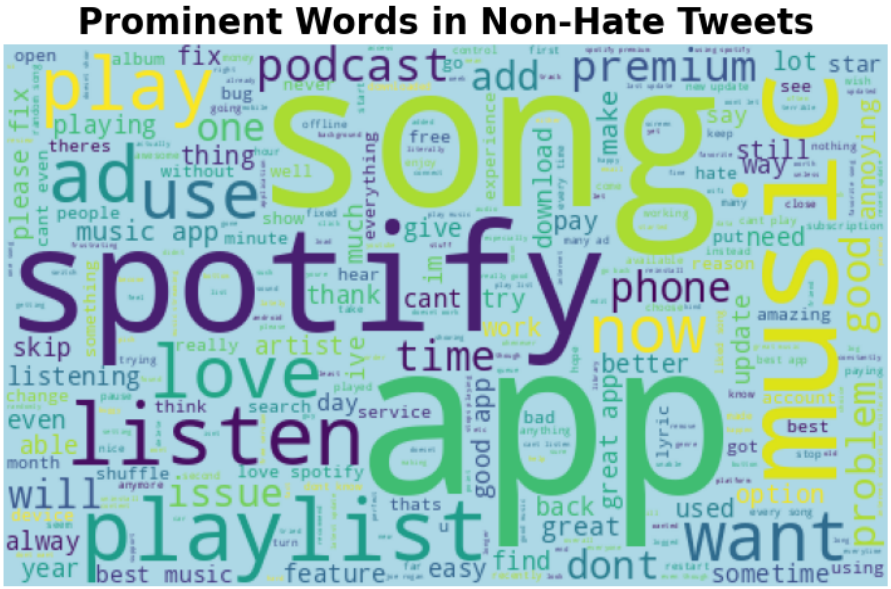

Figure 3.12: Visual Representation of Word Frequencies via WordCloud

The wordcloud depicted 3.12 displays the frequent words with larger fonts and conversely, less frequent terms are depicted in smaller fonts, signifying their lesser occurrence or emphasis within the dataset.

CHAPTER 4

Methodology

4.1 Support Vector Machine(SVM)

Support Vector Machine is machine learning algorithm to solve classification problems,SVMs perform binary classification and need to be extended for multi-class problems. This is relevant in sentiment analysis tasks with more than two sentiment classes. The core principle of the SVM classifier is to identify a hyperplane that best divides data points from distinct classes. Essentially, we aim to achieve the widest possible margin between points from two classes. This hyperplane is chosen to ensure the maximum distance to the nearest points of each class, known as support vectors. These specific points influence the placement and angle of the hyperplane. Decision boundaries with broad margins typically generalize better, reducing the likelihood of errors on new data, while those with narrow margins can lead to over fitting. SVM takes a lot of time for text classification problems because it deals with high-dimensional data, which makes it non-linear SVM hence the computation of kernel matrix complex.Support Vector Machines uses kernel trick which involves using kernel function to transform input data to high dimensional space and make it linearly separable Common kernels include linear, polynomial, RBF (Radial Basis Function), and sigmoid.

The parameters involved in SVM are:

- Gamma: Kernel coefficient for 'rbf', 'poly' and 'sigmoid'. It defines how far the influence of a single training example reaches. Low values mean 'far' and high

values mean 'close.

Support Vector Machines (SVM) utilize different kernels to transform the input data into another space where the classes might be more easily separable by a hyperplane. Two commonly used kernels are the Linear kernel and the Radial Basis Function (RBF) kernel. Here are the primary differences between them. **Choice of Kernel**

- **Linear Kernel**: Creates a linear decision boundary in higher dimensions.
- **RBF Kernel**: The decision boundary can be of any shape,where data is non-linearly separable.

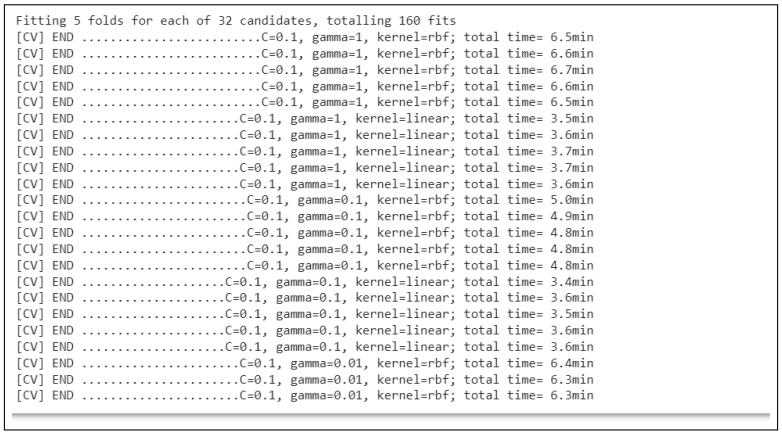

Figure 4.1: Kernel Comparison: RBF vs. Linear in High-Dimensional Data.

In figure 4.1 comparing Kernel Choices: RBF vs. Linear - Efficiency in high-dimensional data processing.

Regularization parameter or Cost parameter

The parameter C represents the regularization or cost parameter in the model. As depicted in Figure 4.2, C=10 prioritizes achieving accurate classification for all examples, potentially at the risk of overfitting. Conversely,C=0.1 provides a margin for errors with a possible tendency to underfit.

Comparative Performance of SVM: Count Vectorization vs. TF-IDF Approaches: In our research, Support Vector Machine (SVM) demonstrated superior performance when paired with the Count Vector representation method. The Count Vector approach converts textual data into vectors by simply enumerating the occurrences of words within a document. On the other hand, the TF-IDF (Term Frequency-Inverse Document Frequency) technique not only considers the frequency of words within a document but

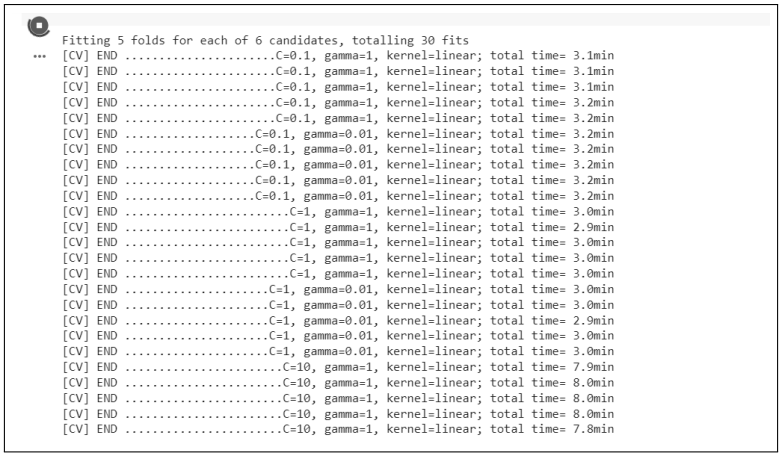

Figure 4.2: Effects of C-Value on Decision Boundaries Across Epochs

also evaluates the distinctiveness of each word across the entire document collection. Consequently, TF-IDF tends to down-weight words that are pervasive and less informative. Despite the intuitive appeal of the TF-IDF's ability to differentiate and prioritize distinctive terms, in our specific context, the SVM classifier yielded better results when utilizing the simpler Count Vector representation. This highlights the importance of methodological selection tailored to the specific nuances and requirements of the dataset under investigation.

Model performance after hyper parameter tuning, showcasing the optimal fit achieved through systematic parameter optimization.

4.2 Recurrent Neural Network(RNN)

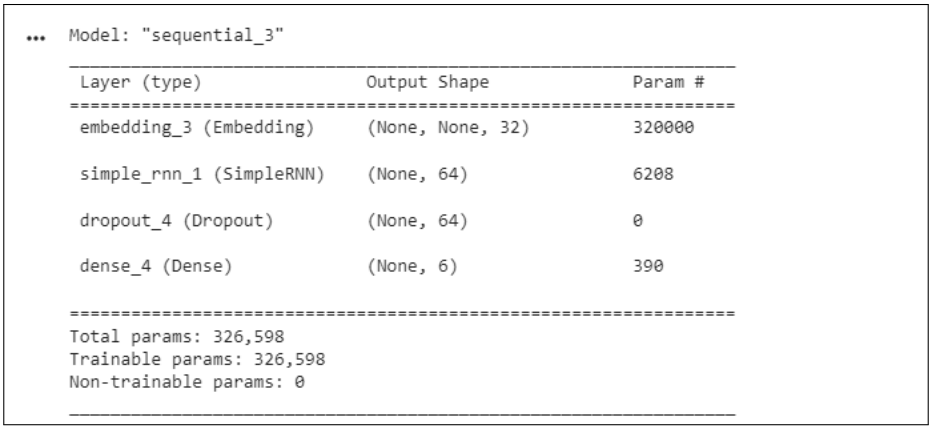

Figure 4.3: Recurrent Neural Networks Design.

Recurrent Neural Networks (RNNs) are architected to process sequential data,

encapsulating memories from previous stages of data input. They find profound applications in domains such as time series forecasting and natural language processing. However, RNNs face challenges. One is the "Vanishing Gradient Problem," where they might forget earlier data if there's a significant gap between when data is received and when it's needed. Another is the "Exploding Gradient Problem," where overly large gradients can destabilize the model.

Optimized Hyperparameter Settings

Vocab size: This captures the most common words in the dataset, while the less frequent ones are labeled as out-of-vocabulary (OOV) tokens, often denoted as 'unk'.

Embedding dimension: This parameter determines the dimensionality of the dense vectors. It's worth noting that an increase in dimensionality can have computational implications and might also require a more substantial dataset for training efficacy.

Max Length: In text,the lengths of sentences or paragraphs can differ, but neural networks need inputs that are uniformly sized. The 'max length' hyperparameter standardizes the length of these textual segments. For shorter texts, padding (typically zeros) is introduced to align with this set length, while longer ones are truncated.

Model Architecture: Simple RNN Design The Simple RNN architecture is specifically oriented towards managing sequential data, such as text. Our implemented model boasts 64 memory-like units termed neurons. L2 regularization is incorporated to mitigate overfitting, ensuring the model does not become overly tailored to the training data. The concluding layer of the model features 6 output neurons and employs a softmax activation function to derive its predictions.

Learning rate: Learning rates help manage how quickly and smoothly a model learns. By multiplying the learning rate by 0.9, it is reduced by 10%. So, if the learning rate for the first epoch is 0.001, the learning rate for the second epoch will be 0.0009, for the third epoch it will be 0.00081, and so on. Early Training: At the onset, the model's weights are usually set at random, meaning the model isn't near its best performance. By having a higher learning rate initially, the model can adjust its weights more drastically, helping it learn faster. Late Training:As the model trains, it approaches its best possible performance. If the learning rate remains high, the model might miss this optimal point. By gradually reducing the learning rate, the model can fine-tune its weights, ensuring it lands on the best solution.

4.3 Long-Short Term Memory(LSTM)

```
Total distinct words: 21471
y Train:  [0 1 0 0 0 2 2 2 2 1]
y Test:  [0 0 1 2 0 2 0 2 2 0]
Model: "sequential_10"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_10 (Embedding)    (None, None, 100)         2147200

 bidirectional_8 (Bidirectio  (None, 32)               14976
 nal)

 dropout_18 (Dropout)        (None, 32)                0

 dense_18 (Dense)            (None, 16)                528

 dropout_19 (Dropout)        (None, 16)                0

 dense_19 (Dense)            (None, 3)                 51

=================================================================
Total params: 2,162,755
Trainable params: 2,162,755
Non-trainable params: 0
_________________________________________________________________
```

Figure 4.4: Long Short-Term Memory Design.

Long-Short Term Memory (LSTM) is an improved version of the Recurrent Neural Network because it doesn't experience the vanishing gradient problem. In back propagation, the gradient can become so diminished that it approaches zero, rendering such a neuron ineffective for further processing. LSTMs retain essential information while recognizing and disregarding irrelevant data.

LSTM Architecture

[20]It possesses a sophisticated structure featuring mechanisms known as gates (including input, forget, and output gates). Additionally, it has a component known as the cell state that aids in retaining information across extended sequences. The input gate governs the flow of new information into the cell, the forget gate regulates the flow of information out of the cell, and the output gate manages the data flow into the LSTM's output. By controlling the flow of information in this way, LSTMs can forget information that aren't important while remembering other information for longer.

Optimized Hyperparameter Settings

Embedding Layer: The embedding layer functions as a word map, where akin words are positioned closely. This arrangement allows the computer to discern the connections between words, facilitating more effective text-based learning.It maps each word in the vocabulary to a 100-dimensional vector.

LSTM Layer(16): LSTM possesses 16 memory cells for information storage. A larger capacity implies a stronger capability to recall and interpret intricate patterns within a sequence. The bidirectional aspect suggests it can extract insights from both preceding and subsequent data points in the sequence.

Activation functions

Rectified Linear Unit(ReLU): It dictates the neuron's output in a neural network. It's about recognizing the primary patterns in the data, filtering out irrelevant components, and focusing on the valuable information.

Hyperbolic Tangent(Tanh): It ensures that the subsequent layer in a neural network can learn from the output of the present layer. The transformation and moderation of values in the LSTM maintain them within a range of -1 to 1. This normalization aids in more effective learning and back propagation within the network.

Sigmoid: It compresses any number into a range from 0 to 1. This is particularly beneficial for denoting probabilities. It's employed to determine which values in the cell state should be retained or discarded. Given its output range from 0 (discard) to 1 (retain), it's ideally suited for this binary decision-making mechanism.

Softmax: It involves making a conclusive decision based on the output of the preceding layer, ascertaining the most probable category or class to which the input aligns.

Dropout(0.5): This layer sporadically deactivates or "drops" 50% of the neurons from the preceding layer during each training iteration. This action ensures the network doesn't become overly dependent on any particular neuron, aiding in the prevention of "overfitting."

Loss function: The categorical crossentropy loss function are used when labels are in a one-hot encoded format whereas sparse categorical crossentropy as an alternative.

Adam Optimizer: Adam integrates the advantages of two other widely-used optimization methods: Momentum and RMSprop. Momentum ensures gradients move consistently, preventing unnecessary fluctuations, whereas RMSprop adjusts the learning rates for each parameter. Adam dynamically changes the learning rate throughout training, which reduces the need for manual tuning compared to certain other methods. Additionally, Adam is both computationally efficient and demands minimal memory.

Output Shape: This indicates a bidirectional LSTM layer with 16 units in each direction (total 32).

Param: 14,976 parameters,learnable parameters of the LSTM cells.

4.4 Gated Recurrent Unit (GRU)

```
Model: "sequential_11"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_11 (Embedding)    (None, None, 8)           171776

 gru_1 (GRU)                 (None, 8)                 432

 dropout_20 (Dropout)        (None, 8)                 0

 dense_20 (Dense)            (None, 3)                 27

=================================================================
Total params: 172,235
Trainable params: 172,235
Non-trainable params: 0
_________________________________________________________________
```

Figure 4.5: Gated Recurrent Network Model Design.

[21] The Gated Recurrent Unit (GRU) is a variant of the Recurrent Neural Network (RNN) and is known to be faster than the Long Short-Term Memory (LSTM) model. The GRU addresses the vanishing gradient issue using two gates: the update gate and the reset gate. These gates determine the information that gets passed to the output and can be optimized to enhance predictions. GRUs can capture long-term relationships or memories because of the internal calculations within the GRU cell that yield the hidden state. Unlike LSTMs, which utilize two distinct states the cell state for long term memory and the hidden state for short-term memory GRUs deploy just one hidden state across time steps. This single state can encapsulate both long-term and short-term dependencies simultaneously, because of the gate mechanisms and the interplay between the hidden state and the input data.

Update Gate (z): The update gate decides on how much information to forget or retain.

Reset Gate (r): The reset gate decides on how much past information to forget when considering the current input.

New Memory Content (h): It's a combination of the current input and a modified version of the previous memory, influenced by the reset gate.

Calculation of the Update and Reset Gates: The GRU computes the values for the update and reset gates based on the present input and prior memory. These gates employ the sigmoid activation function, producing outputs that lie between 0 and 1.
Compute the New Memory Content (h) : The candidate memory blends the current input with previous memory, adjusted by the reset gate. If the reset gate yields a value nearing 0, it suggests "overlook the prior memory and prioritize the present input." Conversely, if the value approaches 1, it indicates "also factor in the historical memory.".
Adjusting the Final Memory (h): At this stage, the update gate comes into play. If its output is near 1, the old memory is predominantly retained. On the other hand, if the value is closer to 0, it leans more towards the newly formed memory (h). This refined memory is then forwarded to the subsequent time step and the ensuing layer.
Optimized Hyperparameter Settings
Embedding Layer: This represents the total unique words or tokens in the text dataset, effectively defining the vocabulary where each word gets associated with an 8-dimensional vector that the model refines throughout its training phase.It changed words into lists of 8 numbers. The model has 171,776 different ways to do this for different words.
GRU Layer: This signifies the count of units or neurons present in the GRU layer, which further indicates the dimension of the output space. At its core, the GRU's hidden state is represented by an 8-dimensional vector.It helped the model remember information it uses 432 pieces of data that it uses for this.
Dropout Layer: This determines the proportion of input units that are omitted during the training phase. Using a dropout is a method of regularization, ensuring the model doesn't overfit to the training data.
Dense Layer: This pertains to the number of neurons embedded in the Dense layer. For classification problems, this might correspond to the number of target categories.This layer helps the model decide the final answer. It has 3 parts, which probably means there are 3 different answers or classes the model is choosing between.
activation='softmax': Activation functions in neural networks function as decision-making bodies, guiding the network's learning process. The softmax activation is a popular choice for classification tasks, as it morphs the model's output into a probability distribution spanning the designated classes.

CHAPTER 5

Results

5.0.1 Efficiency of the Support Vector Machine model

Support Vector Machine model with TF-IDF: The model applies cross-validation (CV) with 5 folds (k=5), dividing dataset into 5 equal parts, and in each iteration, 4 parts are used for training and 1 part is used for validation. This is done 5 times such that each of the 5 parts is used for validation once.We calculate the model performanc using below parameters

- **Precision**: Ratio of correctly predicted positive observations to the total predicted positives.

$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}} \tag{5.1}$$

  Out of all the tweets predicted as negative by the model, 73% were truly negative.Half of the tweets predicted as neutral were correctly classified, while the other half were misclassifications.But the model was most precise in identifying positive tweets, with 84% of tweets predicted as positive being true positives.

- **Recall (or Sensitivity)**: Ratio of correctly predicted positive observations to all the observations in the actual positive class.

$$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}} \tag{5.2}$$

  Out of all the actual negative tweets in the dataset, the model correctly identified 90% of them. This indicates strong sensitivity towards negative sentiments.Neutral

Tweets (0%)The model failed to identify any of the actual neutral tweets. All neutral tweets were presumably classified as either positive or negative.The model correctly identified 87% of the positive tweets from all actual positive tweets.

- **F1-score**: Weighted average of Precision and Recall. It tries to find the balance between precision and recall.

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5.3}$$

The balanced harmonic mean between precision and recall for negative tweets is 80%. This indicates a relatively good balance between precision and recall for this class.The F1 score is zero, emphasizing the model's poor performance in correctly predicting neutral tweets. F1 score is high for positive tweets, showing that the model maintained a good balance between precision and recall for this sentiment class.

To conclude ,The model performs well in detecting negative and positive tweets but struggles significantly with neutral tweets.The high recall for negative tweets indicates the model's efficiency in capturing most of the negative sentiments present in the dataset.However, the model's inability to correctly predict any neutral tweets (0% recall and F1 score) highlights a clear area of improvement.The performance for positive tweets, in terms of all metrics, is commendable and suggests the model's proficiency in recognizing positive sentiments.

Class	Precision	Recall	F1-Score	Support
0	0.73	0.90	0.80	4985
1	0.50	0.00	0.00	1351
2	0.84	0.87	0.85	5936
accuracy			0.78	12272
macro avg	0.69	0.59	0.55	12272
weighted avg	0.76	0.78	0.74	12272

Table 5.1: Detailed Classification Performance of the SVM Model with TD-IDF

Support Vector Machine model with Count Vectorizer: The overall accuracy of the model on the validation set is approximately 78%. Notably, while classes 0 (Negative) and 2(positive) exhibit satisfactory performance, Class 1 is discernibly under performing with low precision, recall, and F1-score. However, the model's under performance in Class 1 stands as a key area for future optimization. This disparity stresses the importance of not solely relying on overall accuracy and underscores the need for meticulous examination of class-wise metrics, especially in imbalanced datasets or scenarios with unequal class distributions. Future work could focus on strategies to improve the performance of Class 1, possibly through oversampling, feature engineering, or leveraging different machine learning algorithm.

Class	Precision	Recall	F1-Score	Support
0	0.74	0.88	0.81	4985
1	0.27	0.02	0.03	1351
2	0.83	0.88	0.86	5936
accuracy			0.78	12272
macro avg	0.69	0.59	0.55	12272
weighted avg	0.76	0.78	0.74	12272

Table 5.2: Detailed Classification Performance of the SVM Model with CountVectorizer

5.0.2 Efficiency of the Recurrent Neural Network model

```
Epoch 1/10
1534/1534 - 44s - loss: 0.9849 - accuracy: 0.5384 - val_loss: 0.7495 - val_accuracy: 0.7068 - 44s/epoch - 29ms/step
Epoch 2/10
1534/1534 - 23s - loss: 0.7285 - accuracy: 0.7301 - val_loss: 0.6730 - val_accuracy: 0.7484 - 23s/epoch - 15ms/step
Epoch 3/10
1534/1534 - 25s - loss: 0.6635 - accuracy: 0.7573 - val_loss: 0.6728 - val_accuracy: 0.7532 - 25s/epoch - 16ms/step
Epoch 4/10
1534/1534 - 22s - loss: 0.6427 - accuracy: 0.7666 - val_loss: 0.6979 - val_accuracy: 0.7557 - 22s/epoch - 14ms/step
Epoch 5/10
1534/1534 - 25s - loss: 0.6293 - accuracy: 0.7717 - val_loss: 0.6358 - val_accuracy: 0.7646 - 25s/epoch - 16ms/step
Epoch 6/10
1534/1534 - 25s - loss: 0.6242 - accuracy: 0.7718 - val_loss: 0.6412 - val_accuracy: 0.7713 - 25s/epoch - 17ms/step
Epoch 7/10
1534/1534 - 22s - loss: 0.6140 - accuracy: 0.7752 - val_loss: 0.6372 - val_accuracy: 0.7757 - 22s/epoch - 14ms/step
Epoch 8/10
1534/1534 - 24s - loss: 0.6088 - accuracy: 0.7769 - val_loss: 0.6156 - val_accuracy: 0.7755 - 24s/epoch - 15ms/step
Epoch 9/10
1534/1534 - 22s - loss: 0.6037 - accuracy: 0.7798 - val_loss: 0.6201 - val_accuracy: 0.7745 - 22s/epoch - 14ms/step
Epoch 10/10
1534/1534 - 23s - loss: 0.5985 - accuracy: 0.7817 - val_loss: 0.6191 - val_accuracy: 0.7798 - 23s/epoch - 15ms/step
```

Figure 5.1: Epoch-wise Performance Evolution of the RNN Model

- Batches:‘1534/1534‘ indicates that the entire dataset has been divided into 1534 parts or batches, and the model updates its weights after seeing each batch.
- Loss:This is a measure of the difference between the predicted outputs and the actual outputs. The smaller the loss, the better the model is performing. For the first epoch, the loss on the training data is 0.989 and keeps reducing with each epoch.
- Accuracy:Evaluates model’s predictions in the first epoch, the model’s accuracy on the training data is 50.91% and increases with every epoch.
- Validation Loss:The presented loss corresponds to the validation data, a distinct subset of the initial dataset that was not exposed to the model during its training phase. This validation data serves to assess the model’s capability to handle unfamiliar data. Notably, the validation loss registered for the initial epoch stands at 0.9795.
- Validation Accuracy : This is the accuracy of the model on the validation data.It increases with every epoch.
- Improvement: Over the course of the 10 epochs,it is observed that both the loss and the validation loss decrease, which means the model is learning and improving its predictions. Similarly, the accuracy and validation accuracy values increase, indicating better performance.
- Significant Jump: There’s a significant improvement from the 1st epoch to the 2nd epoch in terms of both training and validation metrics. The loss drops from 0.982 to 0.7285, and then further down to 0.598, while the accuracy jumps from 53.8% to 78.17%.

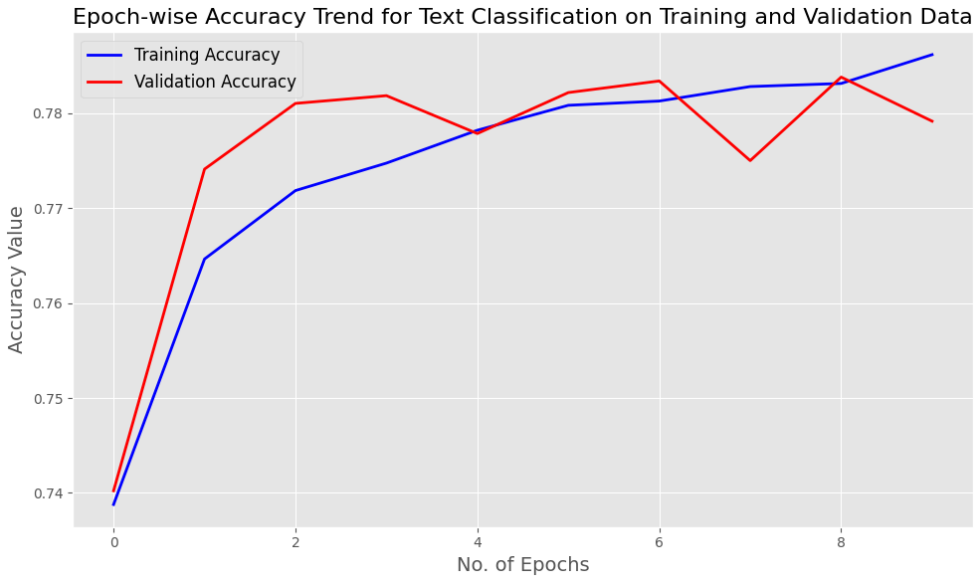

Figure 5.2: Training and Validation Accuracy trend across epochs

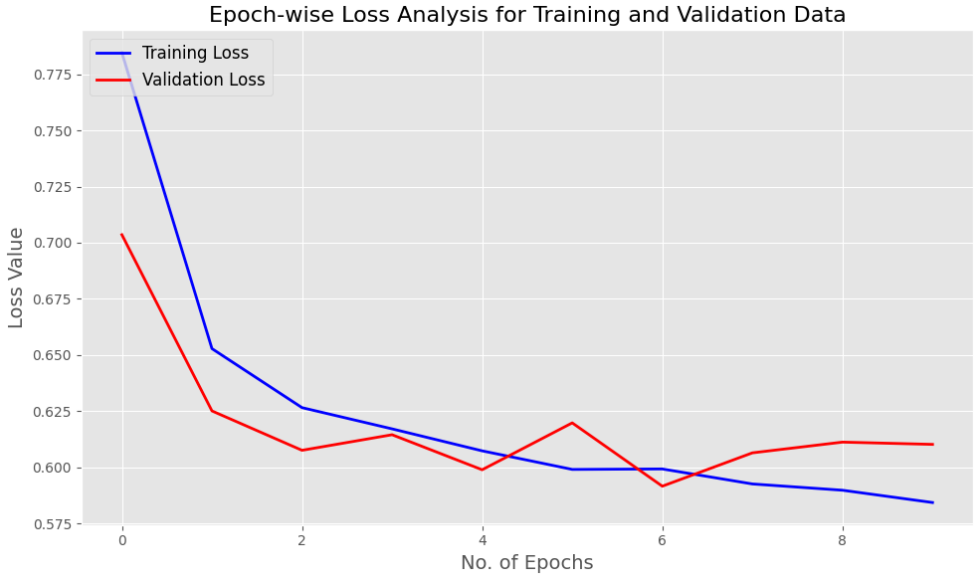

Figure 5.3: Training and Validation Loss trend across epochs

5.0.3 Efficiency of the Long-Short Term Memory model

- Number of Batches: 1534/1534 indicates that there are 1534 batches of data for each epoch.
- Duration:The time taken for each epoch is represented in seconds. For instance, the first epoch took 183 seconds.
- Loss and Accuracy:Training loss and accuracy for each epoch. For example, in the

```
_________________________________________________________________
Epoch 1/10
1534/1534 - 183s - loss: 0.8988 - accuracy: 0.5984 - val_loss: 0.6457 - val_accuracy: 0.7732 - 183s/epoch - 119ms/step
Epoch 2/10
1534/1534 - 173s - loss: 0.6784 - accuracy: 0.7633 - val_loss: 0.5853 - val_accuracy: 0.7847 - 173s/epoch - 113ms/step
Epoch 3/10
1534/1534 - 171s - loss: 0.6242 - accuracy: 0.7865 - val_loss: 0.5753 - val_accuracy: 0.7868 - 171s/epoch - 111ms/step
Epoch 4/10
1534/1534 - 170s - loss: 0.5984 - accuracy: 0.7959 - val_loss: 0.5688 - val_accuracy: 0.7895 - 170s/epoch - 111ms/step
Epoch 5/10
1534/1534 - 176s - loss: 0.5703 - accuracy: 0.8040 - val_loss: 0.5702 - val_accuracy: 0.7884 - 176s/epoch - 115ms/step
Epoch 6/10
1534/1534 - 170s - loss: 0.5545 - accuracy: 0.8090 - val_loss: 0.5723 - val_accuracy: 0.7878 - 170s/epoch - 111ms/step
Epoch 7/10
1534/1534 - 177s - loss: 0.5365 - accuracy: 0.8128 - val_loss: 0.5769 - val_accuracy: 0.7860 - 177s/epoch - 115ms/step
```

Figure 5.4: Epoch-wise Performance Evolution of the LSTM Model

first epoch, the loss was 0.8988 and the accuracy was 59.84%.

- Validation Loss and Accuracy:The model's performance on the validation dataset. In the first epoch, the validation loss was 0.6457 and the validation accuracy was 77.32%.The accuracy improves with every epoch signifying model is working well on the validation set.

- Improvement:The training and validation losses are decreasing initially, and the accuracy's are increasing, indicating that the model is learning.Starting around the 5th epoch, the validation loss starts increasing slightly while the training loss continues to decrease, which might be an early indication of overfitting. The validation accuracy also seems to plateau around this point.

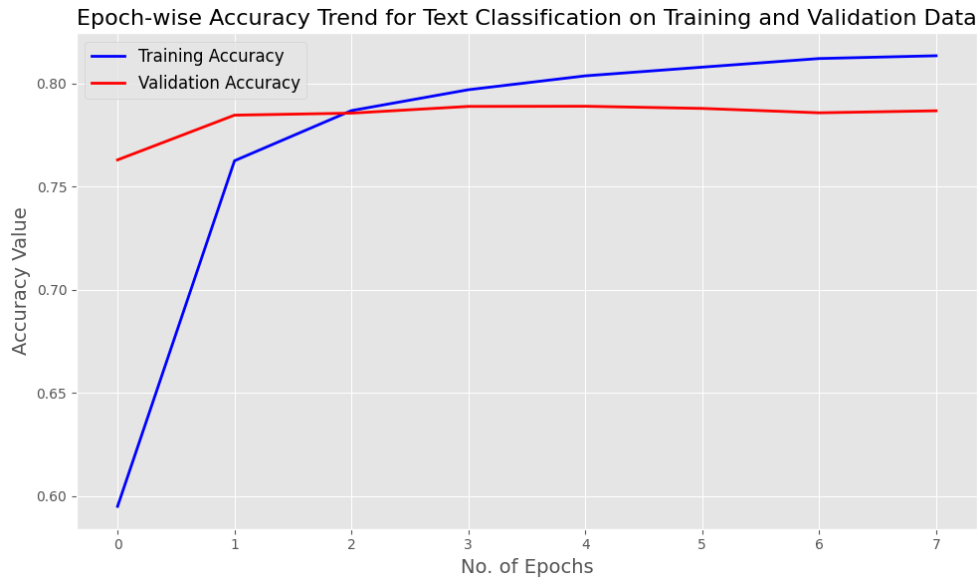

Figure 5.5: Training and Validation Accuracy trend across epochs

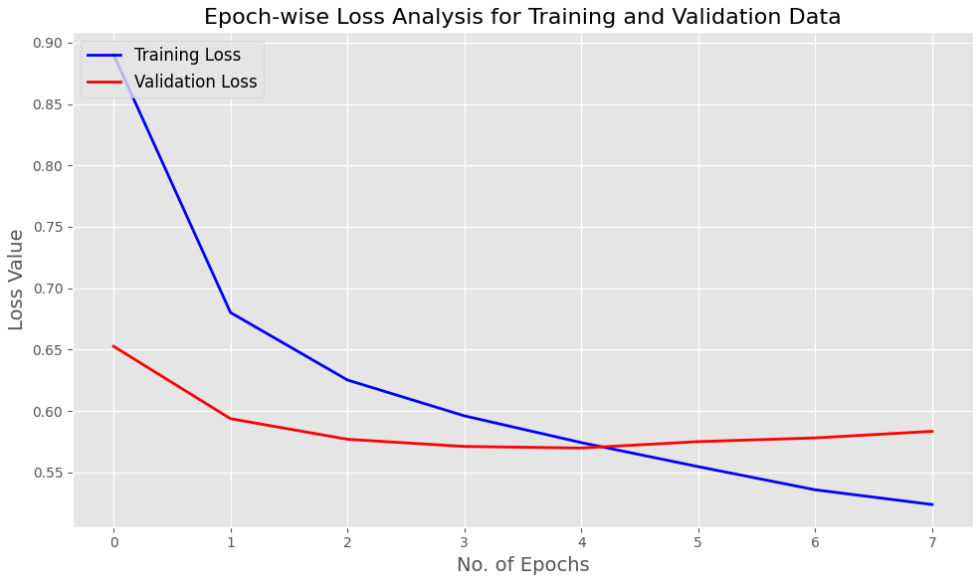

Figure 5.6: Training and Validation Loss trend across epochs

5.0.4 Efficiency of the Gated Recurrent Unit model

```
Epoch 1/25
1534/1534 [==============================] - 38s 23ms/step - loss: 1.0108 - accuracy: 0.4665 - val_loss: 0.9542 - val_accuracy: 0.5299
Epoch 2/25
1534/1534 [==============================] - 34s 22ms/step - loss: 0.9635 - accuracy: 0.5118 - val_loss: 0.9292 - val_accuracy: 0.5474
Epoch 3/25
1534/1534 [==============================] - 38s 25ms/step - loss: 0.8886 - accuracy: 0.5993 - val_loss: 0.6825 - val_accuracy: 0.7450
Epoch 4/25
1534/1534 [==============================] - 35s 23ms/step - loss: 0.6871 - accuracy: 0.7584 - val_loss: 0.6126 - val_accuracy: 0.7779
Epoch 5/25
1534/1534 [==============================] - 35s 23ms/step - loss: 0.6513 - accuracy: 0.7730 - val_loss: 0.5983 - val_accuracy: 0.7826
Epoch 6/25
1534/1534 [==============================] - 37s 24ms/step - loss: 0.6275 - accuracy: 0.7802 - val_loss: 0.5895 - val_accuracy: 0.7845
Epoch 7/25
1534/1534 [==============================] - 34s 22ms/step - loss: 0.6100 - accuracy: 0.7867 - val_loss: 0.5839 - val_accuracy: 0.7876
Epoch 8/25
1534/1534 [==============================] - 35s 23ms/step - loss: 0.5957 - accuracy: 0.7915 - val_loss: 0.5797 - val_accuracy: 0.7889
Epoch 9/25
1534/1534 [==============================] - 34s 22ms/step - loss: 0.5852 - accuracy: 0.7958 - val_loss: 0.5782 - val_accuracy: 0.7877
Epoch 10/25
1534/1534 [==============================] - 34s 22ms/step - loss: 0.5732 - accuracy: 0.8001 - val_loss: 0.5782 - val_accuracy: 0.7882
Epoch 11/25
1534/1534 [==============================] - 35s 23ms/step - loss: 0.5675 - accuracy: 0.8040 - val_loss: 0.5807 - val_accuracy: 0.7887
Epoch 12/25
1534/1534 [==============================] - 33s 22ms/step - loss: 0.5568 - accuracy: 0.8060 - val_loss: 0.5863 - val_accuracy: 0.7817
```

Figure 5.7: Epoch-wise Performance Evolution of the GRU Model

- The model goes through the data in groups or "batches". There are 1534 of these groups.
- The training and validation losses seem to be decreasing over the epochs, which suggests that the model is learning.
- Accuracy (both training and validation) increases with epochs, which means the model is getting better at predictions.

- Around the 10th epoch, the validation loss seems to start increasing slightly (from 0.5782 in the 10th epoch to 0.5807 in the 11th and 0.5863 in the 12th), which might suggest the beginning of overfitting.

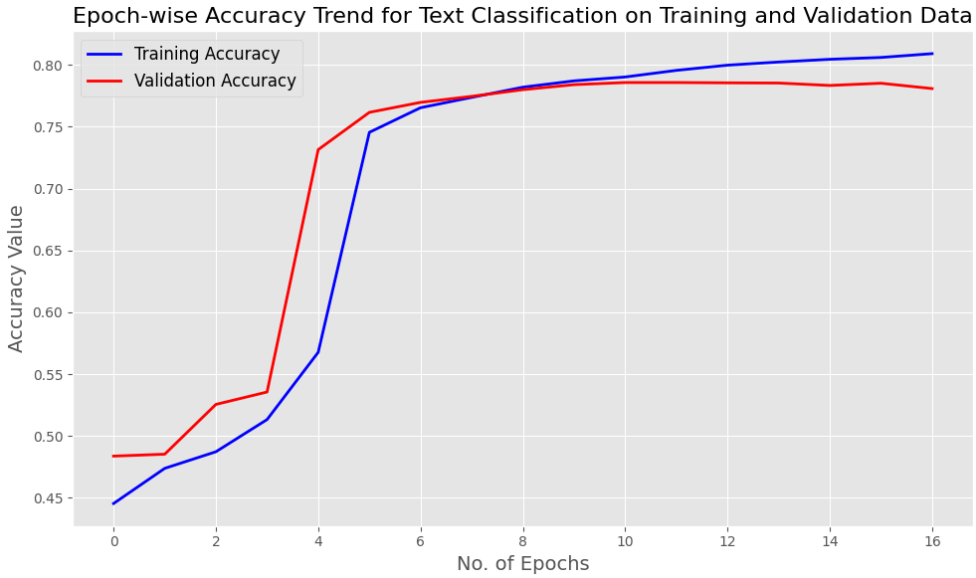

Figure 5.8: Training and Validation Accuracy trend across epochs

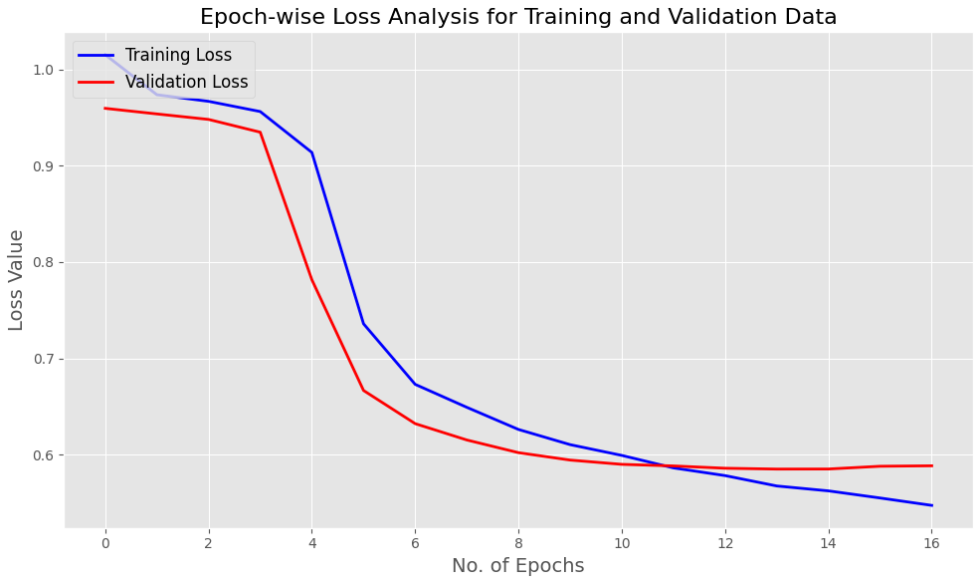

Figure 5.9: Training and Validation Loss trend across epochs

CHAPTER 6

Conclusions

To conclude, this dissertation delves into the use of deep learning techniques for the precise segmentation of sentiments within user reviews on Spotify. It endeavors to pinpoint and scrutinize feedback through both machine learning and deep learning classification methods. Chapter 2 encompasses a thorough literature review, shedding light on prior research on this subject by various scholars. Drawing from diverse perspectives and studies by these authors, the dissertation undertakes the evaluation of multiple deep learning models on customer feedbacks.

In the practical phase, we employed machine learning models using SVM combined with both count vectorizer and TF-IDF. For deep learning, we leveraged RNN, LSTM, and GRU architectures. The initial steps involved a detailed exploration of the dataset with various EDA techniques. Care was taken to ensure reviews were correctly sentiment-analyzed, mitigating the risk of overlooking negative feedback. Subsequently, models were trained exclusively on the training dataset. Post-training, these models underwent a prediction phase with the testing dataset.

The results showcase the superior performance of the LSTM model, in comparison to other models. Evaluation encompassed various metrics such as the confusion matrix, accuracy, and F1 score, offering a comprehensive insight into each model's efficacy. Opportunities for further refinement remain, touching upon concerns such as limited review data , exploring the potential of transformer models in contrast to conventional approaches for enhanced model accuracy. In essence, this research emphasizes the role of deep learning in classifying feedback analysis.

Bibliography

[1] Nlp-overview. https://cs.stanford.edu/people/eroberts/courses/soco/projects/2004-05/nlp/overview_history.html#:~:text=NLP%20%2D%20overview&text=The%20field%20of%20natural%20language,this%20sort%20of%20translation%20automatically.

[2] History and present of naturallanguageprocessing. https://www.deep-talk.ai/post/history-and-present-of-natural-language-processing.

[3] Natalie DeClerck Sridhar Ramaswamy. Customer perception analysis using deep learning and nlp. 2018.

[4] Ashok Kumar Durairaj Anandan Chinnalagu. Context-based sentiment analysis on customer reviews using machine learning linear models. 2021.

[5] Justin Zhan Xing Fang. Sentiment analysis using product review data. 2015.

[6] M. Wankhade, A.C.S. Rao, and C Kulkarni. A survey on sentiment analysis methods, applications, and challenges. 2022.

[7] A.Govardhan I.Hemalatha, G. P Saradhi Varma. Preprocessing the informal text for efficient sentiment analysis. 2022.

[8] Maria Virvou Akrivi Krouska, Christos Troussas. The effect of preprocessing techniques on twitter sentiment analysis. 2016.

[9] Giulio Angiani, Laura Ferrari, Tomaso Fontanini, Paolo Fornacciari, Eleonora Iotti, Federico Magliani, and Stefano Manicardi. A comparison between preprocessing techniques for sentiment analysis in twitter. 2016.

[10] Spotify app reviews 2022: A dataset on kaggle. https://www.kaggle.com/datasets/mfaaris/spotify-app-reviews-2022.

[11] What is stemming and lemmatization in nlp. https://www.analyticssteps.com/blogs/what-stemming-and-lemmatization-nlp.

[12] Python lemmatization approaches with examples. https://www.geeksforgeeks.org/python-lemmatization-approaches-with-examples/.

[13] Lemmatization examples in python. https://www.machinelearningplus.com/nlp/lemmatization-examples-python/.

[14] Vectorization techniques in nlp: A comprehensive guide. https://neptune.ai/blog/vectorization-techniques-in-nlp-guide/.

[15] Understanding tf-idf (term frequency-inverse document frequency). https://www.capitalone.com/tech/machine-learning/understanding-tf-idf/.

[16] Gentle introduction to the bag-of-words model. https://machinelearningmastery.com/gentle-introduction-bag-words-model/.

[17] Word2vec. https://www.analyticsvidhya.com/blog/2021/06/part-6-step-by-step-guide-to-master-nlp-word2vec/.

[18] An introduction to topic modeling. https://levity.ai/blog/what-is-topic-modeling#:~:text=Topic%20modeling%20is%20a%20type,predefined%20tags%20or%20training%20data.

[19] Various techniques for topic modelling.

[20] Using lstm for text classification: A tutorial. https://www.analyticsvidhya.com/blog/2021/06/lstm-for-text-classification/.

[21] A deep dive into gru with pytorch. https://blog.floydhub.com/gru-with-pytorch/.

APPENDIX

Appendix