

Analysing Amazon Product Reviews Using the RoBERTa Model and an Enhanced Pre-trained LSTM Model

By
Pranjal Sinha (2220277)

Supervisor: Gordon Dickers

**Project submitted as part of the requirements for the award of
MSC Computer Networks and Cyber Security**

February 2024

Table of Contents

Abstract.....	7
1. Introduction.....	8
1.1. Background.....	8
1.2. Research Gap.....	10
1.3. Study Aim.....	11
1.4. Objectives.....	11
1.5. Significance.....	11
1.6. Dissertation Structure.....	12
1.7. Hypothesis.....	12
2. Literature Review.....	12
2.1. Artificial Neural Networks (ANNs).....	12
2.1.1. Activation Functions.....	13
2.1.2. Logistic Sigmoid.....	14
2.2. Natural Language Processing.....	14
2.2.1. NLP Techniques.....	15
2.2.1.1. Bag-of-Words.....	15
2.2.1.2. Word Embedding.....	16
2.2.1.3. Continuous Bag of Words (CBOW).....	16
2.3. Theoretical Framework.....	17
2.4. Related Studies.....	20
2.4.1. ULMFIT.....	21
2.4.2. RoBERTa.....	22
2.4.3. XLNet.....	22
2.4.4. DistilBERT.....	22
2.4.5. Bidirectional LSTM (BiLSTM).....	23
3. Methodology.....	24
3.1. Introduction to Methodology.....	24
3.2. Data Collection.....	24
3.2.1. Data Sources and Selection Criteria.....	24
3.2.2. Ethical Considerations.....	25
3.3. Data Preprocessing.....	25
3.3.1. Text Preprocessing.....	25
3.3.2. Annotations and Labels.....	26
3.4. Model Selection and Implementation.....	27
3.4.1. Transformer Models Overview.....	27
3.4.2. Model Fine-Tuning.....	28

3.5. Training Process.....	28
3.6. Sentiment Analysis Process.....	30
3.6.1. Bi-LSTM.....	30
3.6.2. RoBERTa.....	32
3.7. Evaluation and Validation.....	33
3.7.1. Evaluation Metrics.....	33
3.7.2. Validation Techniques.....	35
3.8. Challenges and Limitations.....	36
3.8.1. Methodological Challenges.....	36
3.8.2. Limitations.....	36
3.9. Conclusion.....	36
3.9.1. Summary.....	36
3.9.2. Implications.....	36
4. Implementation.....	37
4.1. Bidirectional LSTM Model Implementation.....	39
4.2. RoBERTa Model Implementation.....	46
5. Results.....	47
5.1. Evaluation Results.....	51
5.2. Analysis and Self Reflection.....	53
6. Discussion.....	54
6.1. Interpretation of Results.....	54
6.2. Limitations.....	55
6.3. Implications.....	56
6.4. Recommendations for Future Research.....	56
7. Conclusion.....	57
References.....	58
Appendices.....	64

List of Figures

Figure 1. Neural Network Model illustration [11].....	13
Figure 2. Bag of words technique illustration.....	15
Figure 3. CBOW Presentation.....	17
Figure 4. Traditional Model Task Learning [37].....	21

Figure 5. Sequential Transfer Learning Process.....	22
Figure 6. Bidirectional LSTM.....	23
Figure 7. Preprocessing Function Used in the Models.....	26
Figure 8. Bi-LSTM Model Architecture for Sentiment Analysis.....	31
Figure 9. RoBERTa Architecture.....	33
Figure 10. Label Classification.....	40
Figure 11. Implementation of Lemmatization.....	42
Figure 12. Text Length Tokenization.....	43
Figure 13. Rating Given to Amazon Products.....	48
Figure 14. Distribution of Different Ratings.....	49
Figure 15. Product Variation.....	49
Figure 16. User Feedback Classification.....	50
Figure 17. Most Words Used for Review.....	51
Figure 18. Evaluation of the Roberta Model.....	52
Figure 19. Evaluation of the Bi-LSTM Implementation.....	52

List of Tables

Table 1. Execution of Bi-LSTM Model.....	44
Table 2. Amazon Product Review Data.....	47
Table 3. Product Variation.....	50
Table 4. BI-LSTM Classification Accuracy.....	53
Table 5. Roberta Classification Accuracy.....	53

Abstract

The objective of this study was to propose a novel LSTM model for effective sentiment analysis. The conventional LSTM model was improved by using a high batch size of 64, along with the introduction of four additional layers, including a bi-directional layer, embedding length with tokenizer, and two dense layers with 128 units each, utilizing sigmoid function activation. Thereafter, its results were compared with the conventional Roberta model for sentiment analysis using the Amazon product review dataset. The aim of the preliminary experiment was to define the batch-size parameters and the number of epochs required for fine-tuning, using the LSTM model as a base and then replicating it for the other models. Previous studies have used lower batch sizes of less than 32 and fewer than four epochs. When fine-tuning the LSTM model with the Amazon corpus, an overfitting behavior was observed from the third epoch onwards, as the validation accuracy was higher at 0.94, precision at 0.86, recall at 0.75, and F1-score at 0.80. This is in contrast to Roberta, which was slightly lower than Bi-LSTM at 0.93 with a precision of 0.77, recall of 0.80, and F1-score of 0.97. To overcome this effect, the number of epochs was reduced to two. After reducing the number of epochs, the Bi-LSTM model, upon completing its fine-tuning, showed a validation accuracy of 0.825, which is an acceptable margin for validating the training. The parameters for fine-tuning were then set at two epochs and batch size at 64 for the Bi-LSTM models. It can be concluded that, by fine-tuning the LSTM model with more layers, it becomes more effective than the Roberta model in sentiment analysis.

1. Introduction

1.1. Background

The rapid growth of digital information has resulted in a vast amount of data, much of which is unstructured text from various sources like social media and customer reviews. This unstructured information has the potential to improve customer experience and business performance, but it also presents challenges that require new analysis techniques and tools [1].

AI (Artificial Intelligence) is a way of seeking, in a computationally simulated way, the automation of intelligent behavior [2]. Therefore, it allows the search for the resolution of new problems, through devices and methods, with the automation of rational behavior. This brings us to a level where we can assume that AI is used in today's technology landscape to solve problems and provide solutions that require a lot of human intellect in a fast, automatic, and reliable way. Two very important components of AI are Intelligent Agents and Prior Knowledge.

Sentiment analysis, also known as opinion mining, is defined as the computational study of people's opinions, feelings, emotions, and attitudes [3]. Extending this definition even further [14], we can say that sentiment analysis is the set of resources, methods, techniques, and tools that allow detecting, extracting, and eventually producing subjective information such as emotional tone, attitude, or preferences expressed through human language, especially through its expression in written form. It is nowadays one of the fastest-growing research areas within artificial intelligence, thanks to the availability of large amounts of subjective information on the Web (interestingly, 99% of the scientific articles in this area have been published in the last 15 years [3]). It is directly related to PLN, and together they address and try to solve the issues that contribute to the applicability of sentiment analysis, such as the handling of negation, irony, or multilingual support. Research in sentiment analysis has been carried out mainly at three levels of granularity, which are briefly described below:

1.1.1. Document

Analysis is carried out on texts of a certain length (several sentences and paragraphs) to determine the opinion as a whole. Each document is assumed to

speak about a single entity.

1.1.2. Sentence

The analysis is carried out at the level of independent sentences (short texts can also be considered) referring to a single entity. This level is closely related to the classification of subjectivity (factual information versus subjective opinion), so in some cases, they are often combined.

1.1.3. Aspect

Analysis at a higher level of detail (fine-grained). It consists of identifying the specific aspects of the entities and determining the sentiment expressed about each of them. It is identified by the acronym ABSA (Aspect-based Sentiment Analysis). In addition, there is the possibility of carrying it out at a higher level of granularity (coarse-grained), identifying the categories of aspects (conceptually more general) referred to in the text together with their polarity [10].

Understanding sentiment analysis as a textual classification task that can be carried out from many different approaches (see Annex B), the aim of such an analysis is to obtain a quantitative evaluation of the sentiment based on the polarity of the opinion expressed, where it can take categorical values such as positive, negative, or neutral, or equivalent numerical values (+1, -1, 0). It is still a subjective evaluation of an individual's opinion or attitude towards a reality in their environment, but through this type of scale, it can be analyzed and managed in a simpler way, obtaining useful and easily interpretable results.

This dissertation seeks to investigate sentiment analysis by comparing various dictionary approaches. The dataset chosen includes Amazon product reviews and ratings that reflect their sentiment. The goal is to assess the effectiveness of different models in predicting ratings. The main aim of this research is to contribute to the existing understanding of sentiment analysis and uncover insights that could lead to improved methodologies.

Online reviews and ratings are extremely important in the world of e-commerce as they shape consumer opinions and influence purchasing decisions. As a result, businesses are realizing the value of analyzing customer reviews and are searching for ways to utilize this information. This study focuses on analyzing reviews on Amazon, which has a large and diverse user base and uses both reviews and ratings. The method used involves assigning a predetermined score to each word in a review based on its sentiment, and then aggregating these scores to determine the overall sentiment of the review as positive, neutral, or negative.

1.2. Research Gap

For a computer, processing strings or data containing characters from a known alphabet is not a complex task, as it has the capacity to compute this type of information. However, a technique is needed so that, in the event this type of data exhibits high-level reasoning, i.e., it is qualitative, the AI can understand the contextually applied meaning. According to Mineiro, gestures, words, body language, and other individual characteristics are absorbed and transformed into knowledge over time, a gradual process. A child's understanding stems from repeating and reproducing what is presented in the environment through imitations. The process with a computer diverges when it becomes possible to import an entire dictionary into a robot, enabling it to know the words of the language and create linguistic connections based on a set of synonyms and antonyms [4]. However, there are subjective situations that demand more attention in natural language processing. Language processing follows a sequence of actions, which begins with reading raw text files, selecting data of interest for observation, and mining information through pre-selection. Patterns obtained from the previous stage are then used to derive specific knowledge, which in this work involves the identification of sentiments and opinions.

Human beings communicate using what we know as natural language, a language that can contain extremely objective phrases, completely subjective phrases, or a mixture of both. When a basic sentence structure containing a subject, a verb, a preposition, a noun, and an adjective is defined for a robot, it is understood by a Natural Language Processor (NLP) in a relatively simple way, because its linguistic structure does not pose any major challenges. If the words have objective meanings, free from figures of speech, the NLP can understand the transmitted message. However, most natural language sentences have more complex and peculiar structures. Even in English, a similarly structured sentence could contain synesthesia or irony, complicating comprehension for artificial intelligence.

To address this gap, Feldman (2014) presents sentiment analysis as 'the task of finding authors' opinions about specific entities.' Given a document with content that can be analyzed, it is responsible for ascertaining and classifying the sentiments contained therein, overcoming the challenges posed by natural language processing in search of information that adds value to the object of study [5].

It is worth noting the importance of studies such as the one conducted by Huang et al. (2019), which apply a model to extract and filter the sentiment of online reviews [6]. However, it has not been applied to online businesses, prompting readers to use these techniques for review analysis in e-commerce. Similarly, Dey et al. (2016) conducted a study where they analyze multiple movie reviews using different techniques, such as Naive Bayes, K-Nearest Neighbor, and RandomForest [7]. This study examines reviews of

multiple movies to find the best method for classification. The challenge of this project is to extract reviews directly from a website for a single product to determine the product's polarity.

1.3. Study Aim

The aim of this study, grounded in sentiment analysis theory, is to develop an enhanced sentiment analysis algorithm using a pre-trained LSTM model and compare its results with the conventional RoBERTa model. This serves the purpose of identifying an algorithm for effective sentiment classification.

1.4. Objectives

The main objective of this work is to develop a novel sentiment analysis algorithm using Artificial Intelligence technologies, followed by its validation and integration in a Big Data environment. To achieve this, the following specific objectives have been established:

- 1.4.1. To utilize an effective sentiment analysis algorithm for performing sentiment analysis on Amazon product reviews.
- 1.4.2. To perform sentiment analysis using the conventional algorithms of the RoBERTa model.
- 1.4.3. To propose an enhanced sentiment analysis algorithm using a pre-trained Long Short-Term Memory (LSTM) model for more accurate sentiment analysis.
- 1.4.4. To compare the sentiment analysis results between the conventional RoBERTa model and the proposed LSTM model based on speed, precision, and accuracy.
- 1.4.5. To reveal findings about the emotional tone present in the Amazon Fine Food Reviews.

1.5. Significance

The results of this research strongly support the use of dictionary approaches in sentiment analysis. Notably, most of the methods selected have been proven effective and reliable, achieving high levels of accuracy in predicting sentiment. A significant advantage of these approaches is their quick deployment capability, which allows for rapid insights into vast amounts of data. This contrasts with machine-learning techniques that often require complex adjustments and substantial computational power. Despite challenges, such as domain-dependence and the lack of contextual nuances, dictionary approaches remain highly valuable due to their accuracy and speed. This affords them a privileged position in real-time analysis, facilitating informed decision-making processes.

The proposed algorithm efficiently analyzes a large number of product reviews, employing a standard metric to evaluate all comments equally. This method helps avoid

human errors, such as overlooking certain comments or misinterpreting them. Broniarczyk and Griffin (2014) highlight the challenge of deciphering user preferences, especially when automating this task. They have developed a model that considers the credibility of a user by factoring in their opinion, thereby assigning it weight in the final review ranking [8].

1.6. Dissertation Structure

This report outlines the work conducted on the study, design, and implementation of a sentiment analysis system leveraging deep learning within a big data environment. The report is organized into five chapters. Following this introductory chapter, which outlines the context and objectives of the work, Chapter 2 examines the theoretical and practical framework underpinning the development of the proposed solution. Chapter 3 details previous work, the programming libraries utilized, and the design of the generic architecture, along with the specifics of each implementation. Chapter 4 then describes the conditions of the experimental environment and the results obtained. It concludes with a real-use case demonstrating the integration of the developed model. Finally, Chapter 5 presents the main conclusions derived from the work and suggests potential avenues for future research to further evolve the system. The report includes a series of appendices that provide further details on language aspects, sentiment analysis, and other relevant topics.

1.7. Hypothesis

- 1.7.1. Sentiment analysis is effective when utilizing the RoBERTa model;
- 1.7.2. The proposed pre-trained LSTM model demonstrates greater accuracy in sentiment analysis.
- 1.7.3. The LSTM model outperforms the conventional RoBERTa model in terms of speed, precision, and accuracy.

2. Literature Review

This chapter delves into the development of Artificial Neural Networks (ANNs), Natural Language Processing (NLP), and Sentiment Analysis (SA), focusing on the theoretical underpinnings that led to the creation of models such as BERT, RoBERTa, and DistilBERT.

2.1. Artificial Neural Networks (ANNs)

Artificial Neural Networks (ANNs) are motivated by the realization that the human brain processes information differently from traditional computer architecture [9]. The human brain operates with components called neurons, which, when connected, form links known as synapses. These neurons are responsible for various tasks, such as vision,

which creates a representation of the environment and provides all the information needed to interact with the space in question [10]. Another important factor is that the brain constantly performs these functions with ease, while computer systems struggle to solve problems of this nature.

In general, a neural network is designed to mimic the way the human brain performs certain tasks or functions in areas of interest [9]. Neural networks are trained using a learning algorithm, which aims to adjust the network's synaptic weights in an organized manner for a specific purpose. Modifying the synaptic weights is the traditional method of learning, but it is also possible for a network to change its topology by altering how its neurons are utilized. The input values for the neuron are represented by the vector x , which contains n values. When they reach the neuron, the input values are multiplied by their respective synaptic weights, which are the elements of the vector w , then added to the bias b to generate the value v , where v is commonly referred to as the activation potential. This process occurs according to the expression shown in Equation 1 [11].

$$v_k = \sum_{j=1}^n x_j w_{kj} + b \quad [1]$$

The value v then passes through a mathematical activation function ϕ , which is responsible for limiting the value to a certain range, producing the neuron's output y . Equation 2 shows the xy value that represents the neuron's output signal [11].

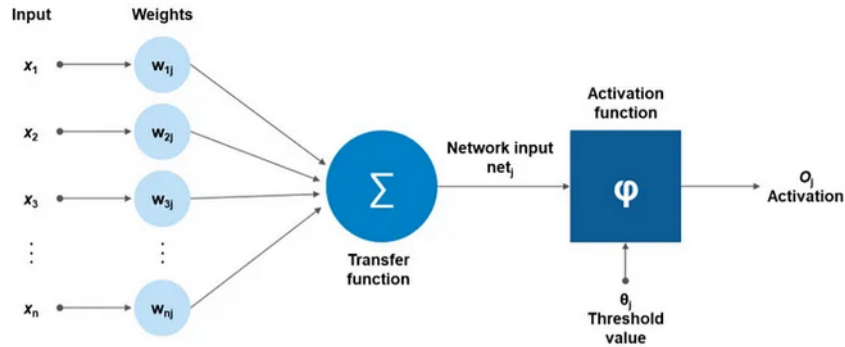


Figure 1. Neural Network Model illustration [11]

2.1.1. Activation Functions

The processing and propagation of each neuron's output signal are carried out by the activation function [12]. The choice of activation functions for a neural network is very important, as they define how the signal will be processed.

Some commonly used activation functions include the Threshold (Step), Logistic Sigmoid, Hyperbolic Tangent, and ReLU (Rectified Linear Unit).

Threshold Activation Function: This activation function is calculated as described in Equation 2.

$$y_k = \begin{cases} 1 & \text{if } v_k \geq 0 \\ 0 & \text{if } v_k < 0 \end{cases} \quad [2]$$

2.1.2. Logistic Sigmoid

The Logistic Sigmoid function is continuous, i.e., differentiable at all points. Its graph has an "S" shape. It is one of the most common forms and offers an interesting balance between linear and non-linear behavior (HAYKIN, 2007). It takes values between zero and one [13]. Its main issue is saturation when the function approaches values of zero and one, as the derivative values get very close to zero, which can compromise the network's learning in algorithms such as Back-Propagation.

2.2. Natural Language Processing

Natural language is a fundamental tool used by humans for everyday communication. Each human language is constantly evolving and adapting to address its unique challenges and ambiguities [14]. In the field of natural language processing (NLP), various techniques have been developed to manipulate and analyze human languages. As the development of language is an ongoing process, linguists and technologists are constantly striving to introduce new techniques to enhance and complement this process. This study focuses on the analysis of transfer learning techniques and their impact on solving two benchmark problems in natural language processing: sentiment analysis (SA) and named entity recognition (NER). SA aims to understand the opinions and sentiments of users or audiences towards a specific target by analyzing vast amounts of textual information from diverse sources. On the other hand, NER involves the complex task of segmenting different entities and establishing relationships among them. While significant efforts have been made to identify various entities, there is still a need for domain-specific entity segmentation. One of the major challenges in machine learning is the dependence on large amounts of high-quality data. Unfortunately, such data is often scarce and expensive to access. However, transfer learning can mitigate this issue by reducing the reliance on high-quality data [14].

Natural Language Processing (NLP) consists of developing computer models to carry out tasks that depend on information expressed in natural language. The aim of NLP is to

make computers perform useful tasks involving human language, such as enabling human-machine communication, improving human-human communication, or carrying out a useful process involving text or speech [15].

The main goal of a Named Entity Recognition (NER) system is to extract specific entities from unprocessed data and assign them to their appropriate categories. This information is highly valuable in numerous NLP tasks, including Information Extraction Systems, Question-Answer Systems, Machine Translation Systems, Automatic Summarizing Systems, and Semantic Annotation. However, the process of segmenting and establishing relationships among these entities can be quite challenging and time-consuming. Despite the considerable effort dedicated to identifying different entities, there is still a need for domain-specific entity segmentation techniques [15-16].

2.2.1. NLP Techniques

NLP techniques essentially involve neural networks, which are machine learning algorithms designed to work with numbers. Some techniques have been developed to include textual data:

2.2.1.1. Bag-of-Words

Bag-of-Words is a model in which words are transformed into tokens, and the frequency of each of these tokens is checked within a certain context [17]. With the number of occurrences of each token, a vector is created where each position represents a word, as illustrated in Figure 2.

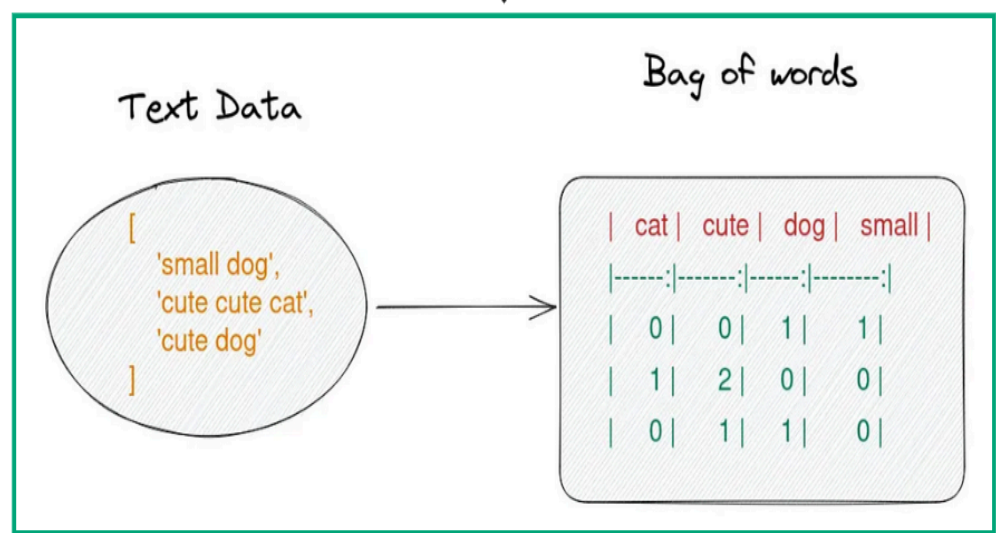


Figure 2. Bag of words technique illustration

In the example in Figure 2, the tokens identified are "is", "one", "dog",

"and", "that", "extremely", "cute". The frequency of each token is then checked. The dataset of this data constitutes the elements of the vector $v = [2, 1, 1, 1, 2, 1, 1]$. Once this vector has been obtained, the Bag-of-Words model can assist ANNs in tasks such as text sentiment analysis and text subject classification [17].

2.2.1.2. Word Embedding

Word Embedding is defined as a set of techniques that map the syntax and semantics of a language into a space known as embedding space [18-19]. Ideally, in this space, words like "cat" and "dog" should be mapped into very close vectors due to their similarity, contrasting with "cat" and "door", where "cat" should be closer to "dog" than to "door", considering that cats and dogs are domesticable animals, and a door is an inanimate object. Word Embedding is processed using dimensionality reduction techniques on the co-occurrence matrix generated from the text input. Processing models include Common Bag of Words (CBOW) and Skip-Gram.

2.2.1.3. Continuous Bag of Words (CBOW)

In this model, the entries are One-Hot-Encoded vectors. For a context with ten words, each position of the vector corresponds to a single word, and the One-Hot-Encoded vector of each word is a vector with the value 1 in the position that represents this word, and the value 0 in other positions [20]. For example, for the words "cat", "dog", and "bird", their One-Hot-Encoded vectors would be respectively: $v_1 = (1, 0, 0)$, $v_2 = (0, 1, 0)$, $v_3 = (0, 0, 1)$. This model predicts the target word from the context, where the inputs are One-Hot-Encoded vectors of size $C \times V$, with V being the vocabulary size and C the context size [21]. The hidden layer contains N neurons, and the output is a One-Hot-Encoded vector of size V , as shown in Figure 3.

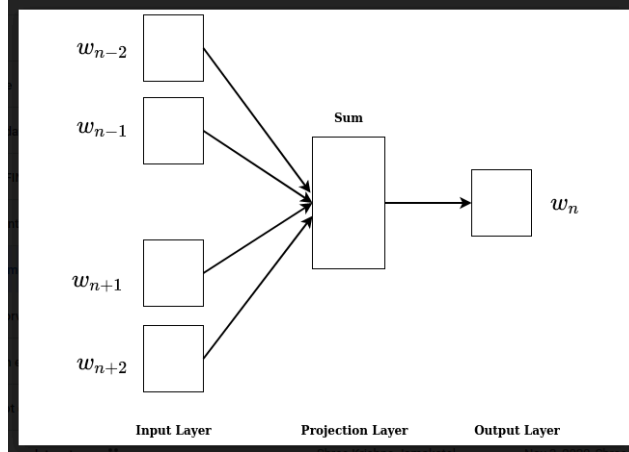


Figure 3. CBOW Presentation

For the phrase 'the sky is blue,' if we want to predict the word 'sky,' the context consists of 'the,' 'is,' and 'blue.' The model, shown in Figure 3, uses these context words to calculate the value of $WV \times N$, which is the weight matrix mapping the input vectors to the hidden layer of dimension N . Subsequently, $W' N \times V$ is the weight matrix that maps the hidden layer outputs to the output layer y_j .

2.3. Theoretical Framework

An integrated lexicon and rule-based approach was utilized to extract both explicit and implicit aspects, as well as to classify sentiment for these aspects [23]. This approach has confirmed the effectiveness of integrating sentiment and aspect lexicons with various rule settings to address different challenges in sentiment analysis. To enhance future work, proper classification and categorization should be implemented to ensure comprehensibility for governments. The technique described in [23] introduces Bidirectional Long Short-Term Memory networks (Att-BiLSTM) and carries out cross-dataset training and evaluation to determine the generalizability of this approach. In clinical NLP, one challenge is the potential for incorrect assertions, leading to inaccurate patient diagnoses and subsequent issues. To mitigate this, the proposed model for assertion detection is introduced.

In [24], a system using transfer learning with ANNs for NER, specifically for patient note de-identification, is proposed. This system transfers ANN parameters trained on a large labeled dataset to another dataset with limited human annotations. The authors propose a domain-adaptive transfer learning method and conduct transfer tests among different corpora to test the methodology's effectiveness. They suggest strategies to acquire transfer knowledge with less linguistic information for transfer learning methods.

The authors in [25] discuss two main aspects of NER research, focusing on target languages and technical approaches with statistical analysis. However, deep learning models require substantial amounts of corpus data, limiting their effectiveness for low-resource languages and domains with limited corpora. Knowledge Bases can improve NER system performance, while Transfer Learning can address the shortage of labeled data. A system for detecting valence tasks using transfer learning with BLSTM technology was proposed. To prevent overfitting, the layers of the pre-trained model were frozen. The system primarily focuses on single transfers but can also undertake multiple transfers to increase the data volume in the process. Transfers will be performed from two classes (positive and negative) to three classes (adding neutral), then to five classes, and finally to seven classes. Another study showed that a transfer learning model trained on an English corpus and transferred to a Chinese corpus achieved comparable performance with fewer annotations, demonstrating its effectiveness and robustness. However, the model is limited by the availability of annotations, hindering a comprehensive understanding of SA techniques and related fields. Additionally, there is a scarcity of resources and research for languages other than English, as well as a lack of benchmark datasets in this field [25-26].

A proposal was made to use unlabeled bilingual corpora to extract useful features by transferring information from a resource-rich language to a resource-poor language. These features, combined with a small amount of training data, will be used to create a supervised NER model [27]. However, the supervised method's efficiency may significantly decrease when the domain of texts changes, resulting in poor results. For example, training the Stanford NER tagger on the IUST training set and testing it on a bilingual test set for clickbait detection showed poor performance.

Researchers benefit from a summary of recent research findings on sentiment analysis with a focus on transfer learning algorithms and applications. The authors also examine the prospects of sentiment analysis, highlighting areas such as the underexplored use of cross-domain transfer learning in aspect extraction and the challenges of addressing negative transfer in text analysis through transfer learning training data.

The authors [27] introduced two methods that utilize unlabeled data to enhance sequence learning with recurrent networks. The first method involves predicting the next element in a sequence, a common technique in natural language processing known as a language model. The second method uses a sequence autoencoder, which encodes the input sequence into a vector and then reconstructs the sequence. However, this approach is unstable as increasing the number of hidden units or backpropagation steps can cause training to quickly deteriorate, with the objective function becoming uncontrollable even with careful gradient clipping. This instability is due to the sensitivity of LSTMs to hyperparameters, particularly in processing lengthy documents [29]. The

authors demonstrated that incorporating contextualized word representations significantly enhances existing models and improves performance in various NLP tasks such as question answering, textual entailment, and sentiment analysis. They emphasize the importance of revealing the inner workings of pre-trained networks, enabling downstream models to utilize different forms of semi-supervised signals. The application of ELMo to a broad range of NLP tasks has shown substantial improvements in model performance [30].

Liu and others have implemented an effective transfer learning method that can be applied to any NLP task, introducing techniques crucial for fine-tuning language models. This method significantly outperforms the current state-of-the-art across six text classification tasks, reducing error by 18-24% on most datasets [31]. They have also proposed several innovative fine-tuning techniques that, when employed in tandem, prevent catastrophic forgetting and enable robust learning across a broad spectrum of tasks. They suggest that the use of a fixed-length vector limits the performance improvement of this basic encoder-decoder architecture. Their proposed extension allows the model to automatically search for relevant parts of a source sentence that contribute to predicting a target word, without explicit segmentation. This novel approach aims to achieve translation performance comparable to the current state-of-the-art phrase-based systems in English-to-French translation. However, the challenge of handling unknown or rare words persists, necessitating improvements for broader application and performance matching of existing state-of-the-art machine translation systems in all contexts [31].

Additionally, they conducted a survey of event extraction technology, which involved describing tasks and related concepts, analyzing and comparing descriptions from various fields, and summarizing the three main methods of event extraction. However, this approach has drawbacks, one being that research on related technologies such as entity and relationship identification, as well as syntactic analysis, is not yet mature, leading to cascading errors. Moreover, the scalability and portability of the event extraction systems in practice are not ideal. For example, research on Chinese event extraction is primarily limited to specific areas such as biomedicine, microblogging, news, and emergencies. There is also a noticeable lack of large-scale, mature corpora and labeled data, indicating the need for further corpus development in this context [32].

The authors in [32] have integrated data mining and information extraction methods to harness the strengths of both. Their objective is to devise a new, high-quality information extraction methodology that also enhances the system's performance and reduces the engineering lifecycle for information extraction. By adopting data mining methods, they have achieved more precise feature selection, leading to a feature space that concentrates on the most pertinent information for mining new knowledge [33].

This chapter explores topic modeling using various tools and techniques, including the Gensim and Mallet Python libraries, to evaluate their applicability to our dataset. These techniques have considerable potential to impact the humanities, but their effectiveness is constrained by the availability, size, and domain of the training data. A primary issue identified is the failure to recognize most of the years mentioned in texts. There is a significant opportunity for NER algorithms to include more cultural contexts during training. A major challenge in machine learning is the dependence on large volumes of high-quality data, which is often inaccessible or prohibitively expensive. However, transfer learning can mitigate this challenge by utilizing pre-trained models that have been developed on tasks with extensive labeled training data, which can then be adapted to similar tasks with less data. These models are typically quicker than traditional machine learning models and contribute to state-of-the-art performance in various tasks. This chapter demonstrates the utility of transfer learning models in formulating predictive modeling tasks such as sentiment analysis (SA) and named entity recognition (NER).

2.4. Related Studies

In recent years, transfer learning has found applications in object detection and some NLP tasks. The primary advantage of transfer learning is the reduction in the need to design classifiers from scratch. It also enables the reuse or customization of models designed for other tasks. This saves considerable time that would otherwise be spent on development and training. In contrast, traditional learning requires creating a new model for each new task, as depicted in Figure 1. Transfer learning is mainly divided into transductive and inductive learning, with further subdivisions such as domain adaptation, cross-lingual learning, multi-task learning (MTL), and sequential transfer learning (STL) to address different variations. This section analyzes the impact of sequential transfer learning on solving benchmark problems in sentiment analysis and NER. Sequential transfer learning, as the name suggests, transfers insights in a sequence to complete tasks and has led to significant improvements. The usual practice involves pretraining representations on a large, unlabeled text corpus and then adapting these to a supervised target task with labeled data.

The task is achieved through sequential transfer learning in two phases: pretraining and adaptation. Pretraining is costlier as it requires extensive training and is executed only once on the source model [34]. It can be approached in three ways: distant supervision (using heuristics and domain expertise), traditional supervision (with manually annotated samples), and no supervision (using large unannotated samples). The adaptation phase employs feature extraction or fine-tuning methods. Feature extraction uses representations from a pre-trained model in another model, while fine-tuning

involves training the pre-trained model on the target task. The phases of feature extraction and fine-tuning are depicted in equations 1 and 2 [35]:

$$\eta^t(l) = 0, \quad \forall l \in [1, L_s] \quad (eq.1)$$

$$\eta^t(l) > 0, \quad \exists l \in [1, L_s] \quad \exists t \quad (eq.2)$$

where $\exists l$ means 'there exists an l'.

Sequential transfer learning is particularly useful when source and target task data are not simultaneously accessible, and the source task has more data than the target task. It is also advantageous for adapting to multiple target tasks [36].

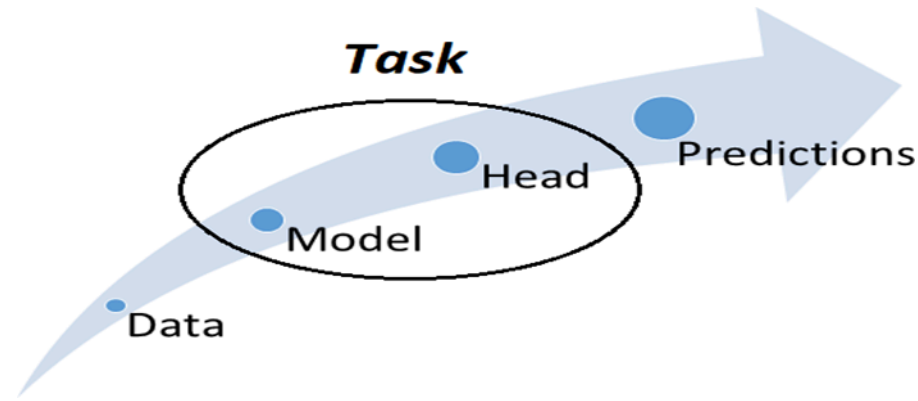


Figure 4. Traditional Model Task Learning [37]

This section reviews different STL models used in designing an SA application. Benchmark models like ULMFIT, RoBERTa, XLNet, and DistilBERT have been effectively employed to enhance task performance.

2.4.1. ULMFIT

Jeremy Howard and Sebastian Ruder designed a language model known as ULMFIT, which extends beyond embeddings and contextualized embeddings [38]. ULMFIT can be fine-tuned for a variety of NLP tasks and is capable of performing well with smaller corpora, unlike another language model mentioned in the study which requires more data for optimal performance. ULMFIT has proven to be more effective than another transfer learning model named Elmo, establishing its value in the fine-tuning process. Those interested in the ULMFIT model are encouraged to consult the original source for more information.

2.4.2. RoBERTa

Developed by Facebook, RoBERTa uses over ten times more textual data and computational power than the BERT model. To improve training, RoBERTa discards the Next Sentence Prediction (NSP) task from BERT's pre-training and implements dynamic masking, where the masked tokens vary with each epoch. RoBERTa's pre-training process utilizes 160GB of text, which includes the 16GB of Books Corpus and English Wikipedia used in BERT [39].

2.4.3. XLNet

XLNet is an advanced version of the BERT model, boasting increased computational power and enhanced accuracy. It employs permutation language modeling as opposed to BERT's architecture, randomly predicting tokens to better learn bidirectional relationships and word dependencies [40]. XLNet is considered an upgrade from BERT—an autoencoder language model that reconstructs sentences using masks—but it addresses BERT's limitations such as disparities during fine-tuning and the assumption of independence of masked tokens. XLNet's training requires approximately 130GB of data [40].

2.4.4. DistilBERT

DistilBERT is a streamlined and efficient Transformer model, derived by distilling the BERT base [41]. It runs faster and has fewer parameters than bert-base-uncased while preserving most of BERT's functionality. DistilBERT maintains the core architecture of BERT but introduces modifications, such as removing token-type embeddings and the pooling layer, as well as halving the number of layers [42].

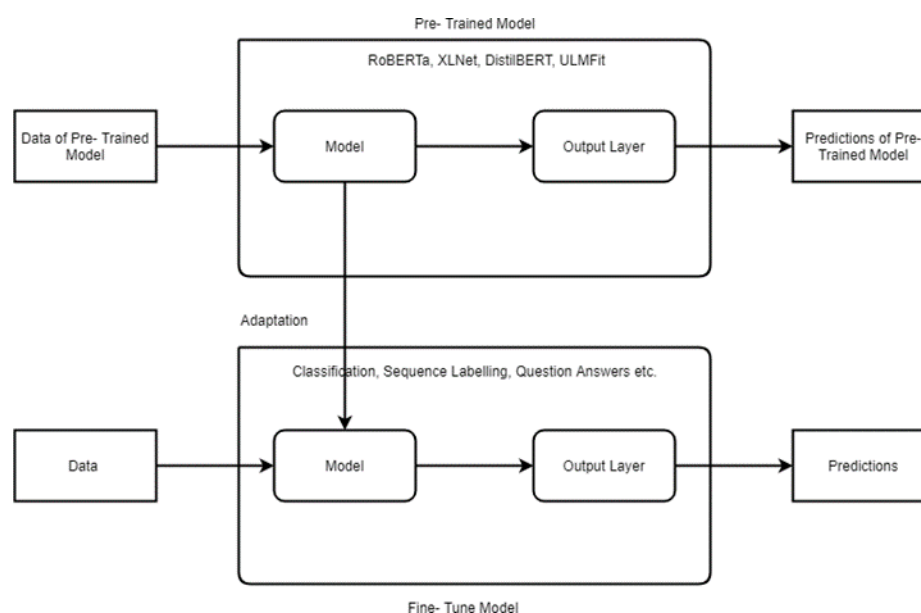


Figure 5. Sequential Transfer Learning Process

DistilBERT is designed for efficiency in linear algebra frameworks, focusing on reducing the number of layers and is trained with large batches using gradient accumulation and dynamic masking, without the next sentence prediction objective [42].

2.4.5. Bidirectional LSTM (BiLSTM)

Bidirectional LSTM (BiLSTM) neural networks operate by processing information not only from the previous time step ($t-1$) but also from the future time step ($t+1$) [49]. This is why they are termed 'bidirectional'; as illustrated in Figure 6, they allow the flow of inputs in both directions, distinguishing them from standard LSTMs where the input x_t moves in a single direction, either backward or forward. In BiLSTM networks, the input moves both ways, preserving information from both the past and the future.

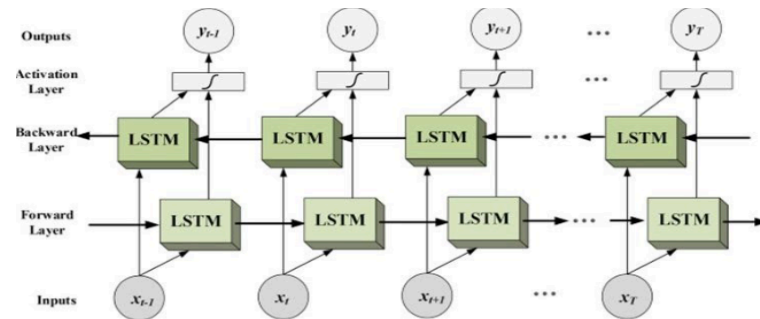


Figure 6. Bidirectional LSTM

As depicted in Figure 6, the information flow from upstream and downstream layers is evident. BiLSTM networks are particularly suited for tasks that require knowledge of the entire sequence, which makes them an optimal choice for text classification applications.

3. Methodology

3.1. Introduction to Methodology

The aim of this paper is to develop a bidirectional LSTM algorithm for the classification of Amazon product reviews and to compare its performance with that of the conventional RoBERTa model. The standard LSTM algorithm was fine-tuned using a two-layer architecture, and its results were benchmarked against the A-RoBERTa model. The Python programming language was chosen for its high-level, user-friendly nature and the extensive range of data science libraries it supports [43]. These attributes have contributed to Python becoming one of the most popular programming languages for research and development in artificial intelligence models.

3.2. Data Collection

3.2.1. Data Sources and Selection Criteria

One of the key objectives is to assess the capabilities of Bi-directional LSTM in performing sentiment analysis on the Amazon Product Review Dataset. For this purpose, the study requires a dataset composed of customer reviews along with their corresponding sentiment labels. The dataset in question contains high-quality food product reviews from Amazon, all stored in the English language. It spans a timeframe of over 10 years, incorporating approximately 500,000 reviews up to October 2012. These reviews offer insights into the products, users, and ratings, and include plain text descriptions. The dataset also encompasses reviews from various categories on Amazon and is accessible at Kaggle (<https://www.kaggle.com/snap/amazon-fine-food-reviews/data>).

For the development of this research, the "Amazon Reviews for Sentiment Analysis" corpus was employed. This dataset consists of 4 million Amazon reviews across different product categories. It was curated based on the scores given to each review: scores of 1 and 2 indicate negative sentiment, while scores of 4 and 5 indicate positive sentiment. Reviews with a score of 3 are considered neutral and were thus excluded from this study, as they do not contribute critical insights. The dataset is categorized into two classes: Class 1 for negative reviews and Class 2 for positive reviews. Each class provides 1,800,000 training samples and 200,000 test samples. For the purposes of this study, a subset of 2000 positive reviews and 2000 negative reviews was selected, amounting to a total of 4000 instances or 2% of the total reviews available in the dataset.

As illustrated in Figure 2, the implemented methodology comprises several steps, which were meticulously followed to ensure the correct execution of the research. Initially, it was necessary to convert the corpus into .txt format files. This conversion was done to create a separate .txt file for each Amazon user review, enabling the segregation of the dataset into negative and positive classes.

3.2.2. Ethical Considerations

The initial phase of this research involved securing permissions from the Amazon Developer platform to access and download the Amazon Product Review dataset. These permissions were formally requested for academic research purposes by the project's Director.

3.3. Data Preprocessing

3.3.1. Text Preprocessing

The purpose of data preprocessing is to achieve homogenization of words, as web-based messaging often exhibits greater variability and spelling mistakes. For preprocessing, the methods proposed by Jesús HerreroLlanos in his TFG were adopted, given the similarity of the problem. These methods include:

- 3.3.1.1. Normalization of upper and lower case: all words are converted to lowercase.
- 3.3.1.2. Standardization of jargon: removal of mentions, hashtags, retweets, and links; elimination of punctuation marks.

After conducting tests both with and without data preprocessing, it was found that the results were consistent, given that the models used include their own preprocessing in case the user did not perform it. If the possibility of preprocessing was not considered by the user, the results were found to match.

This preprocessing incorporates most of the aforementioned steps, with modifications such as replacing hashtags with "hashtag," mentions with "@user," and links with "url," or substituting special characters or emojis with text. Figure 5 illustrates the function definition and documentation included for the models of the pysentiment group. Since the results obtained were identical and the preprocessing included in the models covered almost all the listed techniques, it was decided to input the texts unmodified to utilize the models' preprocessing capabilities.

```

        text, lang="es", user_token=None, url_token=None, preprocess_hashtags=True, hashtag_token=None, char_replace=
        demoji=True, shorten=3, normalize_laughter=True, emoji_wrapper="emoji", preprocess_handles=True):
    """
    Basic preprocessing

    Arguments:
    -----

    text: str
        Text to preprocess

    lang: str (default 'es')
        Language used in the preprocessing. This is used for the demoji functionality and laughter preprocessing

    user_token: str (default "[USER]")
        Token used to replace user handles

    url_token: str (default "[URL]")
        Token used to replace urls

    preprocess_hashtags: boolean (default True)
        If true, applies preprocessing to hashtag, trying to split camel cases

    hashtag_token: str (default None)
        If preprocess_hashtags is True, adds hashtag_token before the preprocessed content of the hashtag

    shorten: int (default: 3)
        If not none, all occurrences of shorten or more characters are cut to this number

```

Figure 7. Preprocessing Function Used in the Models

The datasets were refined to include only six primary emotions: anger, fear, joy, love, sadness, and surprise. The text samples from all three datasets were processed using the pre-trained 'bert-base-uncased' model from the Hugging Face Transformers library [57]. The labels from all three datasets were converted into argmax encodings, with each label represented by an integer starting from 0.

3.3.2. Annotations and Labels

Tokenizing: To train a model on a new dataset, the data must first be prepared to be compatible with the model. Essentially, this involves executing the functions of the encoder layer. The Hugging Face library provides a tokenizer for each supported model to streamline the mapping functions. The dataset is divided into training and test sets and transformed into a format compatible with the library's dataset [44].

When applying the data to a tokenization function, as described in the documentation, the outcome depends on the tokenizer used. The following elements are returned:

- 3.3.2.1. **Text:** unchanged from our data.
- 3.3.2.2. **Label:** unchanged from our data.
- 3.3.2.3. **Input IDs:** assigns a unique number to each distinct word in the dataset, effectively digitizing the text.
- 3.3.2.4. **Token Type IDs:** used for tasks like question answering or language modeling. It involves adding special symbols such as masks or separators to tailor the input for the specific task at hand. In our case, it is not utilized, resulting in a list where all values are zero [45].
- 3.3.2.5. **Attention Masks:** informs the model which tokens should be focused on, according to the attention mechanism of the RoBERTa models.

3.4. Model Selection and Implementation

3.4.1. Transformer Models Overview

RoBERTa was set up following the guidelines provided in the Hugging Face RoBERTa transformers 3.0.2 documentation [58]. The model was trained with a batch size of 16 across 3 epochs. This batch size was maintained during both training and evaluation phases, regardless of whether a CPU or GPU was used. Evaluation followed the "steps" strategy, occurring every 1000 steps in accordance with the logging steps parameter [58].

The Simple Transformer library presents readily available implementations for various applications, including Text Classification, Named Entity Recognition, and Question Answering. This library incorporates pretrained Transformer models that can be fine-tuned to address specific tasks. It features a final classification layer atop the embeddings, enabling its use as a classifier without further adjustments. During training, two features (headline and article body) alongside labels were inputted. The RoBERTa model underwent training with maximum sequence lengths of 512 and 256, employing different learning rates [59]. All transformer models completed training over 5 epochs.

To convert the labeled dataset into vector representations (embeddings), the appropriate tokenizer for each model was invoked using the `Autotokenizer.tokenize()` instruction. Subsequently, the tokenizer was applied to each code file in the dataset, producing a sequence of tokens for each file. These sequences of tokens were then processed by the neural network during training, embodying a seq2seq (sequence-to-sequence) representation derived from code blocks within the training dataset.

3.4.2. Model Fine-Tuning

To fine-tune the bidirectional LSTM model proposed in this study, an architecture comprising four layers was selected based on literature recommendations for optimal performance in sentiment analysis tasks. These layers include:

- 3.4.2.1. **Embedding Layer:** Utilizes word embeddings to convert tweets into vectors of real numbers.
- 3.4.2.2. **Bidirectional LSTM Layer:** Efficiently processes data dependencies, leveraging the strengths of LSTM in both forward and reverse directions.
- 3.4.2.3. **Standard Layer:** Enhances the model's complexity, facilitating finer learning and faster convergence through its relay activation function.
- 3.4.2.4. **Output Layer:** Responsible for making predictions, it employs three neurons corresponding to the three possible tweet polarities, using a softmax activation function ideal for multiple classification problems [46-47].

Additionally, the novel Bi-LSTM model was fine-tuned with a high batch size of 64. Four additional layers were incorporated into the model: a bidirectional layer, an embedding length matched with the tokenizer, and two dense layers with 128 units each, employing a sigmoid activation function.

3.5. Training Process

The dataset should be divided into training, validation, and test sets. This division entails creating three distinct datasets: the first is the training set used to train the neural network; the second is the validation set, used to assess and optimize the network's hyperparameters; and the third is the test set, used for final validation of the network's predictions [50]. The dataset was divided using random sampling into proportions of 0.8, 0.1, and 0.1 for the training, validation, and test sets, respectively. The splitting of the datasets was conducted using the scikit-learn package's `train_test_split` function (Pedregosa et al., 2011).

Before inputting the data into the models, a specific preprocessing routine is followed. Initially, the data is divided into a test set and a training set, ensuring separate datasets for model evaluation. To augment the dataset, noise is introduced in a manner affecting one out of every four attributes, resulting in four unique dataset combinations, each linked to a specific noise source. The `create_table` function is tasked with generating these combinations. This preparatory process guarantees proper data

segmentation, variance computation, and noise augmentation to refine the dataset before model application.

Subsequently, the paths of each CSV file are extracted, totaling 455 files. These files are then sequentially processed. A critical step in this process is the computation of variance, a statistical measure indicating the extent to which data points deviate from the mean. A higher variance suggests a greater dispersion of numbers. This metric is crucial for understanding data distribution. Noise is added by randomly selecting samples from a Gaussian distribution, with the noise level set at 0.1, dictating the intensity of the introduced disturbances.

$$\text{Sample variance} = s^2 = \frac{\sum (x - \bar{x})^2}{n - 1}$$

The variance is a statistical measure that quantifies how much individual values in a dataset deviate from the mean. It offers insights into the spread or distribution of data points across the dataset.

To accomplish normalization, the study utilizes the `normalize_data` function. This function employs a simple sliding window approach, where each window comprises 10 consecutive timestamps. After selecting a window, we normalize its values. The window is then transposed to reorganize the data for further processing. Subsequently, the data is divided into two sets: 85% for training and 15% for testing purposes. This division ensures that we have separate datasets for model training and evaluation.

For the RoBERTa model utilized in this research, a four-layer architecture was adopted, deemed most effective for sentiment analysis issues according to existing literature. The layers comprise: a) an Embedding Layer [51], transforming tweets into vectors of real numbers; b) a RoBERTa Layer, responsible for efficiently processing data dependencies; c) a Standard Layer, enhancing the model's complexity for more refined learning and faster convergence, enabled by its relay activation function; and d) an Output Layer, tasked with prediction, equipped with three neurons to represent the possible tweet polarities, employing a softmax activation function suitable for multiple classification tasks [51].

```
def normalise_data(data_train, sequence_length): # data_train is numpy array of shape(102,5)
    record_min = []
    record_max = []
    data_windows = []
    for i in range(len(data_train) - sequence_length):
        data_windows.append(data_train[i:i + sequence_length]) # sequence data at every 10 day t
    data_windows = np.array(data_windows).astype(float)
```

For the RoBERTa model employed in this research, an architecture comprising four layers was utilized, as identified in the literature as optimal for sentiment analysis tasks. These layers include:

- 3.5.1. **Embedding Layer [51]:** Utilizes word embeddings to convert tweets into vectors of real numbers.
- 3.5.2. **RoBERTa Layer:** Processes data dependencies efficiently, leveraging the model's capabilities for understanding the context.
- 3.5.3. **Standard Layer:** Adds complexity to the model, enhancing the precision of learning and accelerating convergence through its relay activation function.
- 3.5.4. **Output Layer:** Responsible for predictions, this layer incorporates three neurons corresponding to the three possible tweet polarities. A softmax activation function is employed, ideal for addressing multiple classification problems [51].

3.6. Sentiment Analysis Process

3.6.1. Bi-LSTM

Two models were developed, utilizing recurrent and convolutional neural networks, both taking the variable 'text' as input, which was tokenized and padded. The output variable, 'selected_text,' was one-hot-encoded for processing by the models.

Sequential Model with Bidirectional Recurrent Neural Network:

- 3.6.1.1. **Embedding Layer:** Features a matrix of dimensions 12001 x 100, containing the word-to-word co-occurrence matrix derived either from a pre-trained corpus (via transfer learning) or from training on the dataset used in this study.
- 3.6.1.2. **Bidirectional LSTM Layer:** Includes 50 units with a hyperbolic tangent activation function.
- 3.6.1.3. **Dense Layer:** Contains 128 neurons with a 'relu' activation function.

- 3.6.1.4. Dropout Layer: Aims to mitigate overfitting by regularizing the weight updates at a rate of 0.2.
- 3.6.1.5. **Dense Layer (Output):** The network's output is composed of three neurons, one for each sentiment, utilizing a softmax activation function to ensure the outputs sum to 1. The 'Adam' optimizer and 'categorical-crossentropy' loss function are used for predicting more than two categories, with Recall, Precision, and F1 Score as the established metrics. Figure 6 displays the schematic of this network.

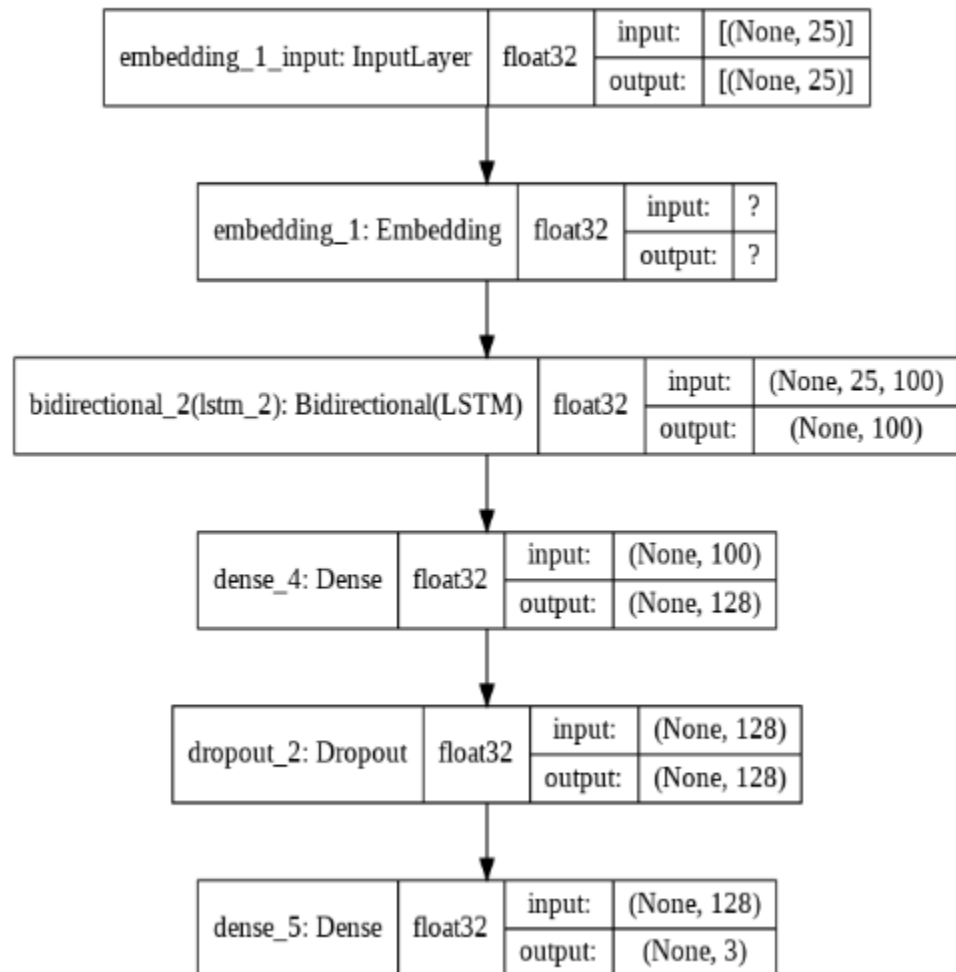


Figure 8. Bi-LSTM Model Architecture for Sentiment Analysis

The model also utilizes sequence representation, a technique that captures the natural execution order of code and the sequence in which expressions are processed. This method models the code intuitively, leveraging recurrent neural networks, bidirectional LSTM (Bi-LSTM), and transformer-based architectures. In this representation, each code token or identifier is encoded as a fixed-length vector. This representation is

fed into deep learning models that handle sequential data, facilitating tasks such as code generation, code completion, and code summarization [70].

Keyword Extraction with Associated Sentiment Loading:

Baseline Model Strategy 1: This model is designed with dual inputs: one for the tweet set, tokenized and padded to a uniform length, and the other for sentiment, categorized on a scale from zero to two. The tweet data is processed through an embedding layer, utilizing transfer learning on a co-occurrence matrix generated via the GloVe technique. The sentiment data is directed to a separate embedding layer trained within the model. Following this, the data proceeds to a Flatten layer, then to a RepeatVector layer, which broadcasts the sequence 25 times. The outputs from the RepeatVector and the initial embedding layer are concatenated, funneling through two bidirectional LSTM layers. The sequence then advances to a dense layer with 128 units and concludes in another dense layer equipped with a single neuron and sigmoid activation, producing a singular output.

3.6.2. RoBERTa

The model utilizes three inputs in RoBERTa's proprietary format [48], which are fed into the TFRobertaModel layer. This layer directs the inputs to two identical neural network segments, each producing different outputs—one for the start index and the other for the end index of the selected text. The architecture includes a Dropout layer with a rate of 0.1, a one-dimensional convolutional layer with a kernel size of one, a Flatten layer, and then a deactivation layer with a softmax function:

- 3.6.2.1. **Input Layer 1:** Accepts tokenized tweet IDs, padded to the maximum tweet length.
- 3.6.2.2. **Input Layer 2:** Receives the attention mask, a crucial feature for utilizing the RoBERTa layer in the model.
- 3.6.2.3. **Input Layer 3:** Establishes tokens representing types of IDs, similar to Layer 2, necessary for the RoBERTa layer's implementation.
- 3.6.2.4. **RoBERTa Layer:** Inputs 1, 2, and 3 are processed here.
- 3.6.2.5. **Dropout Layer:** Set at a rate of 0.1.
- 3.6.2.6. **1-Dimensional Convolutional Layer:** Configured with a kernel size of one.
- 3.6.2.7. **Flatten Layer:** Adjusts the convolutional layer's output size.

3.6.2.8. Activation Layer 1 & 2: Each triggers an output using the softmax function.

The model is compiled with the "Categorical_crossentropy" loss function, ADAM optimizer, and incorporates the Jaccard metric directly into its training. Figure 7 illustrates the model's architecture.

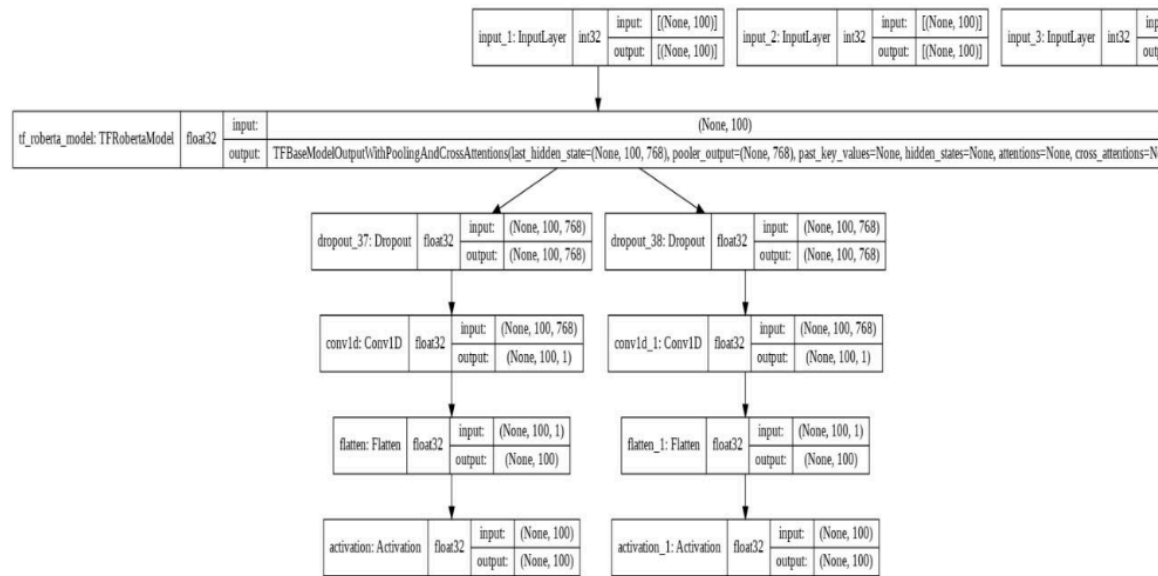


Figure 9. RoBERTa Architecture

The study employs a truncated knowledge transfer technique, as depicted in Figure 7. A new layer is added to the network's end for classifying files into 11 predefined categories. These categories, assigned decimal numbers from 0 to 10, are provided to the model during training. For RoBERTa implementation, the code is structured within a class named Model that inherits from PyTorch's neural network base class, Module. The forward method is overridden with a softmax activation function, utilizing the CrossEntropyLoss loss function. Preparing the neural network involves loading pre-trained models and their corresponding tokenizer, then creating an instance of the Model class by specifying the pre-trained model, tokenizer, and a configuration object.

3.7. Evaluation and Validation

In evaluating the performance of neural networks, several metrics play a crucial role, including the confusion matrix, accuracy, sensitivity, specificity, F1-score, and the Kappa coefficient. To comprehensively assess the model's quality, seven distinct evaluation techniques have been selected. Among these, one notable method is Cohen's Kappa

Coefficient. This section will detail the calculation of these metrics, particularly focusing on scenarios where there are $(k=3)$ categories.

3.7.1. Evaluation Metrics

The confusion matrix is an essential tool for model evaluation, enabling the identification of both errors and successes encountered by the model during the learning process. It categorizes predictions into four segments: true positives, false positives, false negatives, and true negatives [52]. These elements are defined as follows:

- 3.7.1.1. **True Positives (TP):** Correct positive predictions in comparison to the actual data.
- 3.7.1.2. **True Negatives (TN):** Correct negative predictions, where the model accurately identifies the negative class.
- 3.7.1.3. **False Positives (FP):** Also known as Type I error, these are incorrect positive predictions for instances that are actually negative.
- 3.7.1.4. **False Negatives (FN):** Also known as Type II error, these are incorrect negative predictions for instances that are actually positive [52].
- 3.7.1.5. **Accuracy:** This metric provides the total percentage of correctly predicted instances by the model and is crucial for evaluating its overall effectiveness. For a k-class problem, accuracy is derived by summing the elements on the main diagonal of the confusion matrix and dividing by the total sum of the matrix [53].

$$\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{FP} + \text{FN} + \text{TN})$$

- 3.7.1.6. **Precision:** Defined as the ratio of true positives (TP) to all predicted positive instances (TP + FP), precision measures the accuracy of positive predictions. It is mathematically expressed as follows [54]:

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$$

- 3.7.1.7. **Sensitivity (Recall or True Positive Rate):** This is the proportion of actual positive cases correctly identified by the model, calculated using the formula [55]:

$$\text{Sensitivity} = \text{TP} / (\text{TP} + \text{FN})$$

- 3.7.1.8. **Specificity (True Negative Rate):** Specificity measures the proportion of actual negative cases correctly identified by the model. It is crucial for understanding how well the model detects the negative class [55]:

$$\text{Specificity} = \text{TN} / (\text{TN} + \text{FP})$$

- 3.7.1.9. **F1-score:** To combine precision and sensitivity into a single metric, the harmonic mean of both is utilized. The F1-score is particularly valuable because it penalizes low values [55]:

$$\text{F1-score} = 2 \times (\text{Precision} \times \text{Sensitivity}) / (\text{Precision} + \text{Sensitivity})$$

- 3.7.1.10. **Kappa Coefficient:** This statistical measure accounts for the chance agreement, providing a more nuanced assessment of the model's accuracy than mere percentage calculations. It is defined as [56]:

$$\text{Kappa Coefficient} = (\text{P}_{\text{obs}} + \text{P}_{\text{exp}}) / (1 - \text{P}_{\text{exp}})$$

Where P_{exp} represents the empirical probability of agreement on the class assigned to any sample, and P_{exp} is the hypothetical probability of agreement by chance.

3.7.2. Validation Techniques

Training and validation loss and accuracy graphs play a crucial role in understanding the model's learning performance across epochs. They aid in diagnosing learning problems that could result in underfitting, overfitting, or other errors. Additionally, these graphs are instrumental in illustrating the weights of the trained model during the inference stage. To prepare the validation data for analysis, the study involved tokenizing, padding, and converting the data into tensors. To streamline this process, these operations were encapsulated into a function named `encode_pad`, significantly reducing code repetition when applying these operations to the training data.

To visualize the training and validation loss trends, the study imported pickle files containing the loss dictionaries recorded during the training of the Transformer model. Subsequently, the training and validation loss values were extracted from these dictionaries and plotted on a single graph for comparative analysis. The training graph, depicted with a blue line, reveals the learning progress of the two sentiment analysis algorithms. Conversely, the validation graph, marked with a red line, demonstrates the algorithms' ability to generalize effectively to new data.

3.8. Challenges and Limitations

3.8.1. Methodological Challenges

The datasets, which include fake and hate content, are imbalanced. This imbalance means that one class label significantly outnumbers the other, potentially causing algorithms to become biased towards the majority class and leading to decreased accuracy. To enhance model performance and mitigate this bias, future research could employ the SMOTE (Synthetic Minority Over-sampling Technique) for oversampling, thereby balancing the datasets.

3.8.2. Limitations

The primary limitations include an unequal distribution of samples across labels, insufficient data preprocessing, and the models' inability to achieve high accuracy rates. Furthermore, the dataset exhibited biases and lacked proper weighting, particularly in the removal of newline characters, potentially impacting the accuracy and generalizability of the emotion detection models employed.

3.9. Conclusion

3.9.1. Summary

This study showcased the implementation of sentiment analysis and learning models for text and emotion detection using the Amazon Product Review dataset. A straightforward approach utilizing the Hugging Face transformer, based on the RoBERTa transformers 3.0.2 documentation, was proposed. The training processes for the models were delineated, underscoring the distinctions between them. Additionally, graphical representations were employed to bolster the effectiveness of sentiment analysis.

3.9.2. Implications

The findings suggest that future methodologies in sentiment analysis should overcome the current limitations by incorporating weights and biases, eliminating newline characters, and exploring strategies to enhance the accuracy

and generalizability of text and emotion detection algorithms.

4. Implementation

The configuration of the GPU environment was accomplished using Google Colaboratory, underscoring the necessity of selecting suitable computational resources for the effective implementation of more demanding models such as Bi-LSTM and RoBERTa. These models necessitate a robust environment equipped with graphics processing units (GPUs) or a cluster of machines for optimal performance [62]. The deployment of 4 GPUs was pivotal in accelerating the training process, with the NVIDIA RTX 3090 GPUs playing a significant role in enhancing model training efficiency.

The initial step in the training process for both models involved importing essential libraries and mounting Google Drive. Libraries such as **Pandas**, **NumPy**, and **Seaborn**, alongside the **Natural Language Toolkit (nltk)**, were incorporated to facilitate model training and learning. **NumPy**, **Pandas**, **Seaborn**, and **Sklearn** are **Python** libraries designed for numerical computing, data analysis, data visualization, and machine learning, respectively. These tools empower developers to efficiently build applications leveraging data science capabilities in a timely and effective manner [61].

```
# Google Colab specific library for mounting Google Drive
from google.colab import drive
drive.mount('/content/gdrive')

# Standard libraries for basic operations and file handling
import re
import string
import warnings
import bz2

# Data manipulation and analysis libraries
import pandas as pd
import numpy as np

# Visualization libraries
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud
```

```

# Setting visualization styles
plt.style.use("fivethirtyeight")
sns.set_style("darkgrid")

# Machine learning libraries for modeling and evaluation
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report, roc_auc_score
from sklearn.ensemble import RandomForestClassifier,
AdaBoostClassifier, GradientBoostingClassifier,
DecisionTreeClassifier

# Deep learning libraries
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Bidirectional,
Dropout, Embedding
from tensorflow.keras.callbacks import EarlyStopping

# Text processing libraries
import nltk
from nltk.corpus import stopwords
from nltk.stem.wordnet import WordNetLemmatizer
from nltk import word_tokenize
import emoji

# Downloading necessary NLTK datasets
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('omw-1.4')
nltk

```

After converting the data into text files, the processing phase commenced, leveraging the Python programming language. Python facilitated numerous tasks throughout this study, including file conversion, data manipulation, transformation into ARFF format, and generating files with information gain insights.

During the data processing stage, several libraries were instrumental. The Natural Language Toolkit (NLTK) significantly eased tasks such as tokenization, and the removal of special

characters and stopwords. Additionally, the Spacy library was employed for its superior lemmatization capabilities, particularly its efficient handling of irregular verbs, setting it apart from other libraries.

Following these steps, a bag of words model was created and stored in a text file, encapsulating the processed data for further analysis.

4.1. Bidirectional LSTM Model Implementation

Initially, the testing data, after preprocessing, comprised 400,000 rows and 3 columns. It's common practice to aim for a dataset size that has roughly ten times more data samples than parameters. However, this guideline is not strict, as certain deep learning models, including some configurations of Bidirectional LSTM, can perform effectively even with a 1:1 ratio of data samples to parameters, especially in cases with fewer data columns.

```
izuo1 = df.loc[df[0] == 1][:100000].reset_index(drop=True) #
Select and reset index for group 1
izuo2 = df.loc[df[0] == 2][:100000].reset_index(drop=True) #
Select and reset index for group 2

# Concatenating the two subsets into a single DataFrame
result = pd.concat([izuo1, izuo2], ignore_index=True)
```

	0	1
0	1	Buyer beware
1	1	The Worst!
2	1	Oh please
3	1	Awful beyond belief!
4	1	Don't try to fool us with fake reviews.
...	..	...
199995	2	It Works!
199996	2	Love this book!
199997	2	Good basics book
199998	2	Must read for new parents
199999	2	great book!

The next phase involved identifying sentiments within the data through EDA. This analysis focused on the major data columns, the results of which are summarized in the table above. Preliminary analysis, depicted in Figure 8, classified sentiment labels in the dataset into two primary clusters before constructing any models.

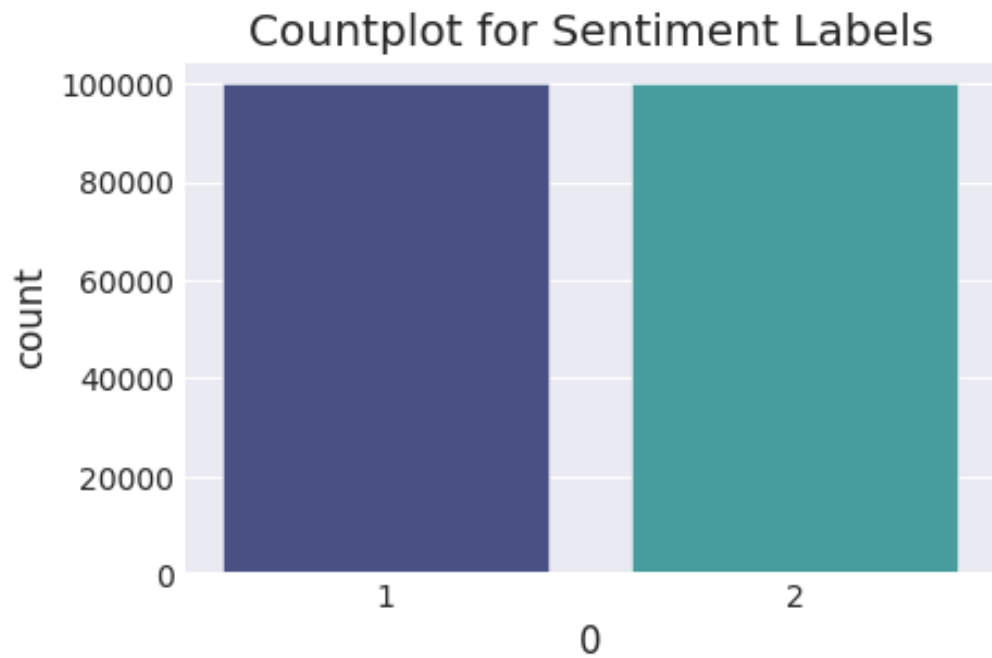


Figure 10. Label Classification

The sentiments were categorized as either negative or positive, with each cluster encompassing over 90,000 counts. Subsequently, the study employed word lemmatization to standardize text tensors, utilizing the "WordNetLemmatizer" function in Python. This process leveraged WordNet's morphy function to return words to their base form, unless the word could not be found in WordNet, in which case it was left unchanged [65].

```
# Precompiling the regular expression for performance
#emoji_pattern = re.compile(emoji.get_emoji_regexp())

# Initializing the Lemmatizer
lemmatizer = WordNetLemmatizer()

# Loading stopwords once to improve efficiency
stop_words = set(stopwords.words('english'))

def preprocess_text(text):
    # Expanding contractions
    text = re.sub(r"won't", "will not", text)
    text = re.sub(r"can't", "can not", text)
```

```

# Removing non-alphanumeric characters
text = re.sub('[^a-zA-Z0-9]', ' ', text)

# text = emoji_pattern.sub(r'', text)

# Lemmatizing words and removing stopwords
text = [lemmatizer.lemmatize(word) for word in
text.split() if word.lower() not in stop_words]

# Joining words back into a single string
text = ' '.join(text)

return text

# Applying the preprocessing function to specific DataFrame
columns
result["clean_title"] = result[1].apply(preprocess_text)
result["clean_text"] = result[2].apply(preprocess_text)

```

The focus then shifted to the embedding layer, pivotal for converting textual data into a numerical format and acting as the foundational layer in Keras deep learning models for the Bi-LSTM model.

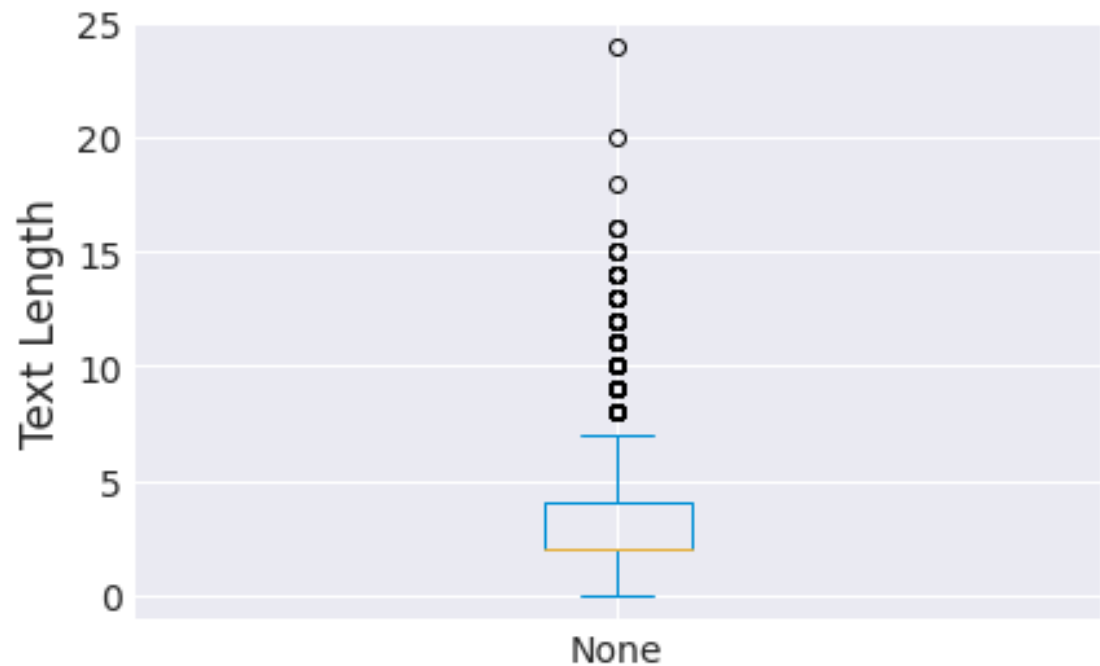


Figure 11. Implementation of Lemmatization

As illustrated in Figure 9, the lemmatization function effectively processed texts of varying lengths. Tokenization was applied to break down most texts into smaller units or tokens, facilitating the segmentation of texts into meaningful parts for easier processing by the models. This step was crucial for enhancing model accuracy and precision. It was noted that the semantic intensity of words could vary with text length; for instance, "good" might convey a stronger positive sentiment in shorter sentences compared to longer ones.

```
# Visualizing the distribution of text lengths in your dataset
using a histogram

plt.figure(figsize=(12, 8))
sns.histplot(text_length, color="skyblue", kde=True)
plt.xlabel("Text Length")
plt.ylabel("Frequency")
plt.title("Distribution of Text Lengths")
plt.show()
```

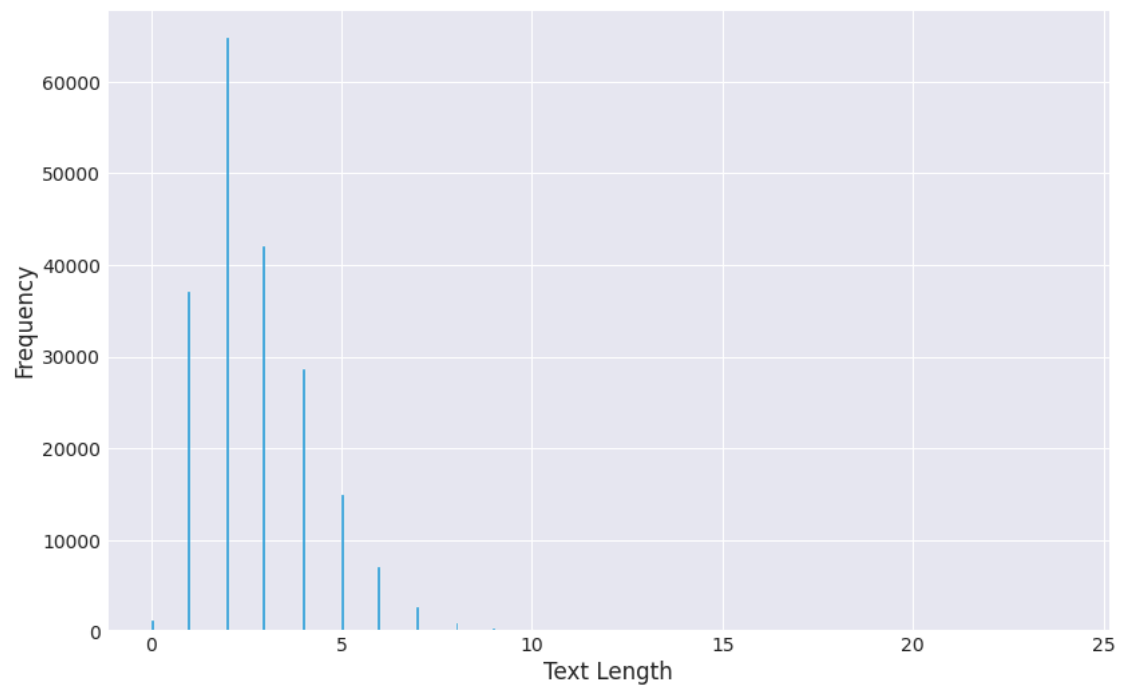


Figure 12. Text Length Tokenization

Following data preprocessing and tokenization, the construction of the Bidirectional LSTM model began. This model architecture involves duplicating the network's first recurrent layer to create two parallel layers, feeding the original input sequence to one layer and a reversed copy to the other. This approach overcomes limitations inherent in standard RNNs by utilizing a bidirectional recurrent neural network (BRNN), which leverages all available input information from both past and future contexts within a single timeframe.

```
BATCH_SIZE = 64

model = Sequential()
model.add(Embedding(len(tokenizer.index_word)+1,64))
model.add(Bidirectional(LSTM(100,
dropout=0,recurrent_dropout=0)))
model.add(Dense(128, activation="relu"))
model.add(Dense(1,activation="sigmoid"))

model.compile("adam","binary_crossentropy",metrics=["accuracy"])
model.summary()
```

```
plot_model(model, to_file='model_plot.png', show_shapes=True,
show_layer_names=True)
```

The Bi-LSTM model features a first hidden layer with 20 memory units, and its output layer, fully connected, produces one value at each timestep. A sigmoid activation function is applied to the output to predict the binary outcome. To enable forecasting a value per timestep using the entire sequence as input, a TimeDistributed wrapper layer is employed around the output layer. This approach requires the Bi-LSTM hidden layer to return a sequence of values (one for each timestep) rather than a single value summarizing the entire input sequence. Given the binary classification nature of the problem, binary log loss (binary_crossentropy in Keras) is used as the loss function. The model's weights are optimized using the ADAM method, known for its efficiency.

Table 1. Execution of Bi-LSTM Model

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, None, 64)	7614080
bidirectional (Bidirectional) 1)	(None, 200)	132000
dense (Dense)	(None, 128)	25728
dense_1 (Dense)	(None, 1)	129
Total params: 7,771,937		
Trainable params: 7,771,937		
Non-trainable params: 0		

The Bi-LSTM model's batch size was increased to 64, incorporating four types of layers: an embedding layer, bidirectional layers, and two dense layers, each with up to 128 units. The model was fine-tuned with 7,771,937 trainable parameters to achieve high precision. The code snippets below were utilized to calculate the Bi-LSTM model's validation accuracy, and training and validation losses.

```
model.evaluate(X_test_seq_padded, y_test)
```

```
# Extracting accuracy and loss from the training history for
both training and validation sets
acc = history.history["accuracy"]
val_acc = history.history["val_accuracy"]
loss = history.history["loss"]
val_loss = history.history["val_loss"]
```

```
# Plotting Training and Validation Accuracy
plt.figure(figsize=(9,6))
plt.plot(acc, label="Training Accuracy")
plt.plot(val_acc, label="Validation Accuracy")
plt.title("Training and Validation Accuracy")
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.legend()
plt.show()
```

```
# Plotting Training and Validation Loss
plt.figure(figsize=(9,6))
plt.plot(loss, label="Training Loss")
plt.plot(val_loss, label="Validation Loss")
plt.title("Training and Validation Loss")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.legend()
plt.show()
```

For model evaluation, training and validation losses ranging between 0.5 and 1.0 were selected. A dropout rate between 0.1 and 0.3 was implemented, with the understanding that a moderate rate would suffice. This setup ensures the training and validation data are shuffled and divided randomly. Moreover, if patterns identified in the training data are absent in the validation data, it might indicate overfitting. An initial data split of 80% for training, 10% for validation, and 10% for testing was deemed suitable. However, the optimal distribution among validation, test, and training sets can vary based on specific use cases, the model's architecture, and data size [64].

4.2. RoBERTa Model Implementation

The importation of the transformer facilitates the use of the pretrained RoBERTa model, incorporating the preprocessing techniques applied during its initial training. This setup provides a straightforward method for classifying texts into positive or negative categories using a pipeline. The `AutoTokenizer` is the tokenizer class automatically selected based on the `model_type` property found in the config object. If the `model_type` property is absent, the tokenizer class is determined by pattern matching on the `pretrained_model_name_or_path` string [66]. Furthermore, the `AutoModelForSequenceClassification` includes a classification head atop the pretrained model outputs, simplifying the integration of training with the base model [67].

```
from transformers import AutoTokenizer
from transformers import AutoModelForSequenceClassification
from scipy.special import softmax
```

A tokenizer was applied to the RoBERTa model to segment lengthy words into meaningful units, as demonstrated in the following code snippet:

```
# Code that will tokenize the example text and have model
predict an outcome on it and output the results in the form of
a dictionary
encoded_text = tokenizer(example, return_tensors='pt')
output = model(**encoded_text)
scores = output[0][0].detach().numpy()
scores = softmax(scores)
scores_dict = {
    'roberta_neg': scores[0],
    'roberta_neu': scores[1],
    'roberta_pos': scores[2]
}

print(scores_dict)
```

This code snippet tokenizes the given text and leverages the model to make a prediction on it. The outcomes are presented in the form of a dictionary, offering an intuitive way to understand the model's classification results.

5. Results

The initial findings from the exploratory analysis of the dataset reveal key insights, including the distribution of sentiment classes and the most frequent words associated with each class. An important step prior to data processing involves analyzing the distribution by sentiment tagging. Notably, the frequencies of negative and positive sentiments are comparably similar, with reviews tagged as neutral displaying a marginally higher frequency.

The table below offers a concise overview of the dataset following its importation using the pandas library and the `pd.read` function:

Table 2. Amazon Product Review Data

Id	ProductId	UserId	Summary	Text
1	B001E4KFG0	A3SGXH7AUHU8GW	Good Quality Dog Food	I have bought several of the Vitality canned d...
2	B00813GRG4	A1D87F6ZCVE5NK	Not as Advertised	Product arrived labeled as Jumbo Salted Peanut...
3	B000LQOCHO	ABXLMWJIXXAIN	"Delight" says it all	This is a confection that has been around a fe...
4	B000UA0QIQ	A395BORC6FGVXV	Cough Medicine	If you are looking for the secret ingredient i...
5	B006K2ZZ7K	A1UQRSCLF8GW1T	Great taffy	Great taffy at a great price. There was a wid...
568450	B001EO7N10	A28KG5XORO54AY	Will not do without	Great for sesame chicken..this is a good if no...
568451	B003S1WTCU	A3I8AFVPEE8KI5	disappointed	I'm disappointed with the flavor.

				The chocolat...
568452	B004I613EE	A121AA1GQV751 Z	Perfect for our maltipoo	These stars are small, so you can give 10-15 o...

This preliminary analysis sets the stage for deeper data processing and modeling efforts, providing a foundational understanding of the dataset's sentiment composition.

The dataset features key headers including user ID, product ID, summary, and product review text. The subsequent step in data preparation involves filtering words based on their syntactic roles. Only words functioning as proper nouns, nouns, verbs, adverbs, and adjectives are retained, while terms fulfilling other syntactic roles are excluded.

To complete the preprocessing phase, accents and punctuation marks are removed from the words. Additionally, any text fragments that become empty as a result of the preceding filtering process are also eliminated. This meticulous preparation ensures that the dataset is optimized for the subsequent stages of analysis.

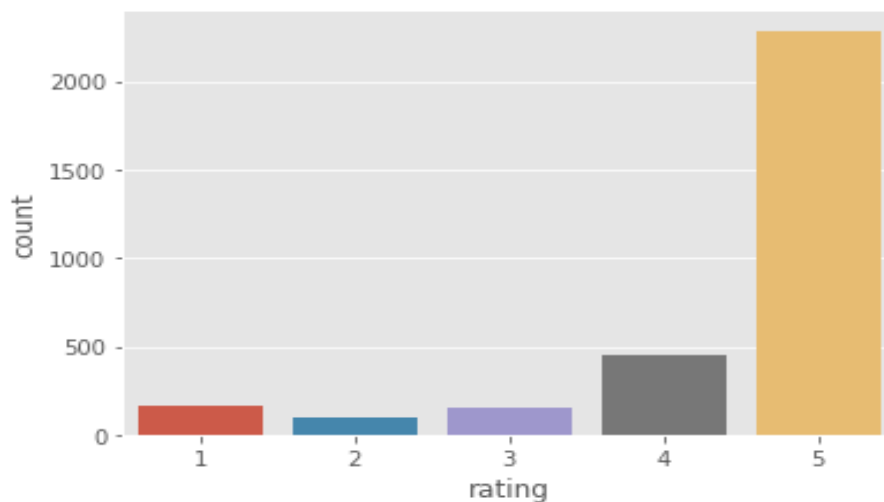


Figure 13. Rating Given to Amazon Products

This figure illustrates the distribution of ratings for Amazon products, showing that a significant majority (72%) of products received a five-star rating, indicating general customer satisfaction. Specifically, more than 2,000 products were rated as five stars.

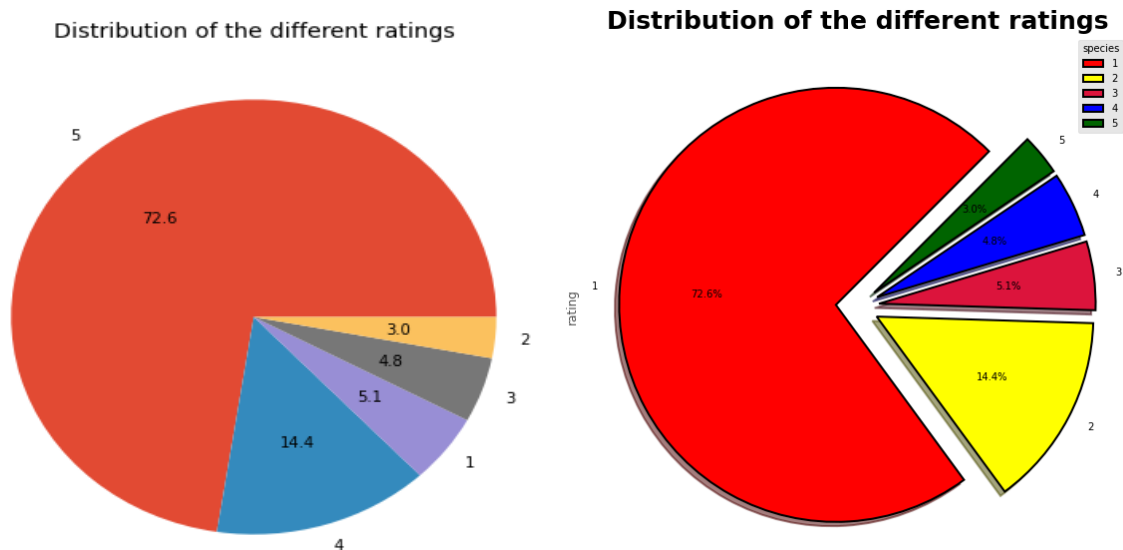


Figure 14. Distribution of Different Ratings

This figure categorizes the five possible customer ratings, highlighting the overwhelming prevalence of five-star ratings among the products reviewed.

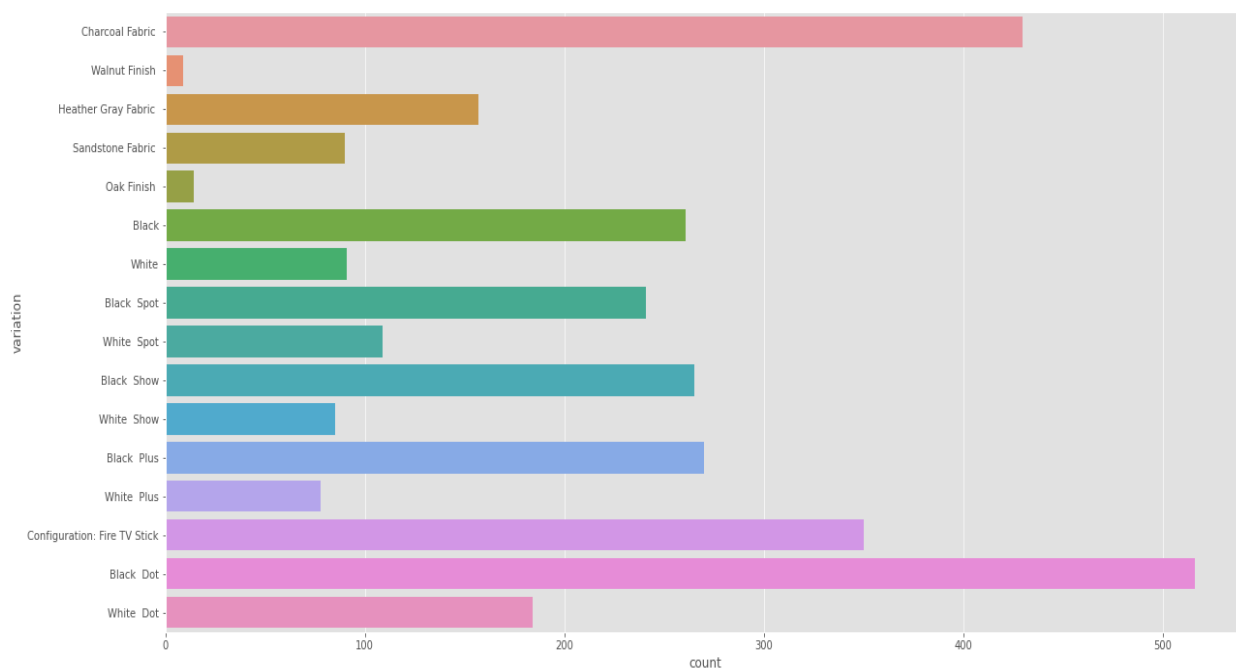


Figure 15. Product Variation

Analysis identified 16 distinct products, with 'Black Dot' and 'Configuration TV Stock' being the most prevalent. The `sns.countplot(x='feedback', data=df) plt.show()` function facilitated the

visualization of product variation.

Table 3: Product Variation

Black Dot	516
Charcoal Fabric	430
Configuration: Fire TV Stick	350
Black Plus	270
Black Show	265
Black	261
Black Spot	241
White Dot	184
Heather Gray Fabric	157
White Spot	109
White	91
Sandstone Fabric	90
White Show	85
White Plus	78
Oak Finish	14
Walnut Finish	9

Name: variation, dtype: int64

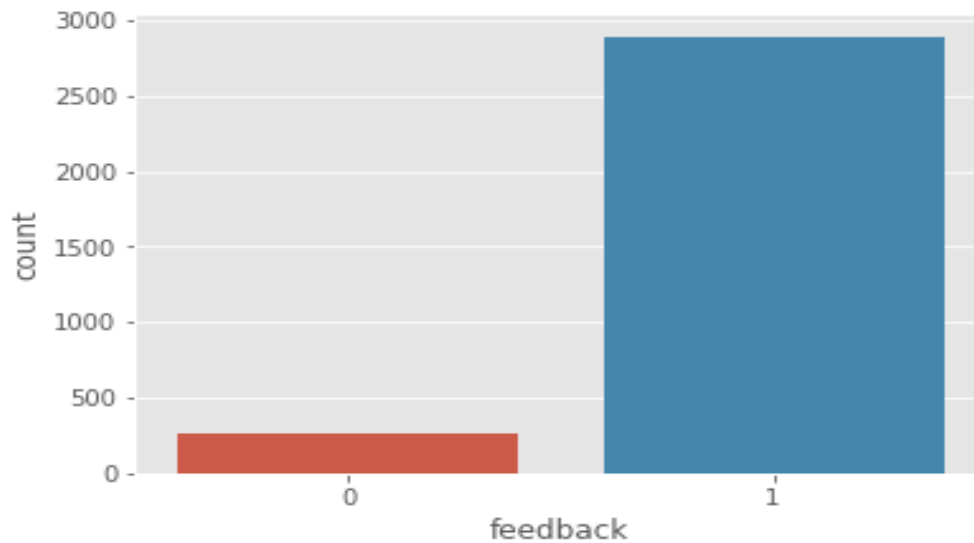


Figure 16. User Feedback Classification

Feedback from users was divided into two categories, with the first category having fewer than 500 counts and the second exceeding 2,500 counts. This classification was achieved using the function `(fig = plt.figure(figsize=(7,7)) tags = df['feedback'].value_counts())`.

[illegible]

Analysis of the most frequently mentioned words in reviews utilized the lemmatizer=WordNetLemmatizer() function. The words "Love," "Great," "Music," and "Device" emerged as the most commonly cited terms in the dataset.

To assess the performance of the models, I utilized a range of metrics including Accuracy, Sensitivity, F1-Score, and the ROC curve for each version. These metrics were compiled for each model, with composite metrics such as variance and standard deviation also being calculated. The accuracy for each model was reported at every epoch.

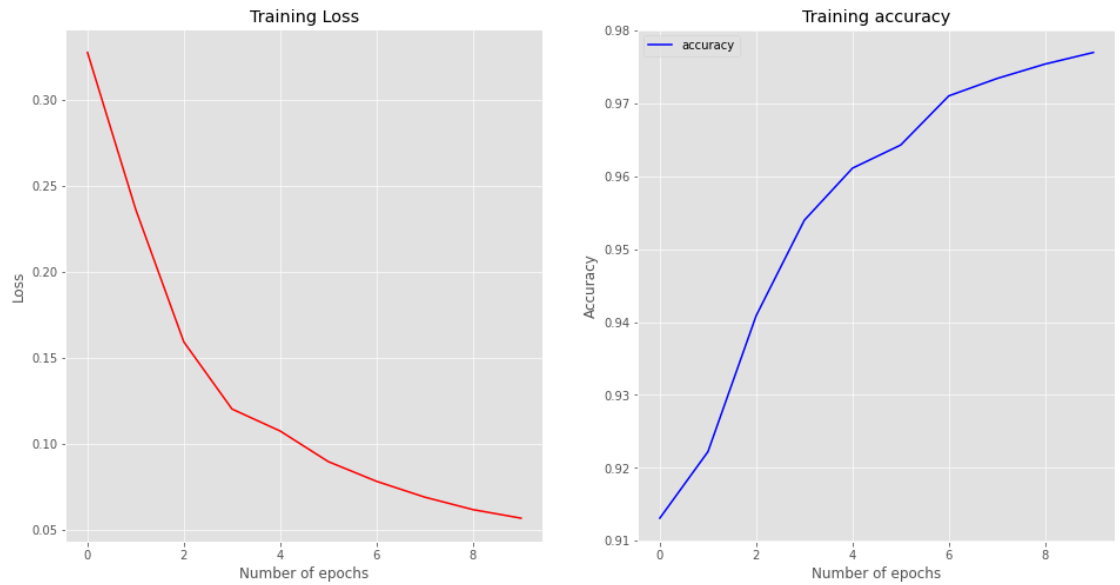


Figure 18. Evaluation of the Roberta Model



Figure 19. Evaluation of the Bi-LSTM Implementation

Among the models evaluated, the Bi-LSTM emerged as the superior performer. The ROC curve for the Bi-LSTM is depicted in Figure 5.10, where the dark blue curve represents the average of the curves across different versions, and the shaded area illustrates the standard deviation of this average curve. Tables 4 and 5 aggregate metrics for the Bi-LSTM model, detailing the lowest, highest, average, mean, and standard deviation for

F1-Score, Accuracy, and Sensitivity. Notably, Sensitivity exhibited a higher standard deviation than the other metrics, indicating variability.

Table 4. BI-LSTM Classification Accuracy

	precision	recall	f1-score	support
0	0.86	0.75	0.80	766
1	0.96	0.98	0.97	4187
accuracy			0.94	4953
macro avg	0.91	0.86	0.88	4953
weighted avg	0.94	0.94	0.94	4953

The F1 Score, a crucial metric in machine learning for evaluating binary and multi-class classification models, was employed to gauge training performance. It synthesizes precision and recall into a singular metric, offering a holistic view of model efficacy. The F1 Score is particularly valuable for its consideration of both false positives and false negatives, making it ideal for scenarios where a balance between precision and recall is necessary. For the Bi-LSTM model, the achieved metrics were an accuracy of 0.94, precision of 0.86, recall of 0.75, and an F1-Score of 0.80.

Table 5. Roberta Classification Accuracy

	precision	recall	f1-score	support
0	0.77	0.80	0.79	766
1	0.96	0.96	0.96	4187
accuracy			0.93	4953
macro avg	0.87	0.88	0.87	4953
weighted avg	0.93	0.93	0.93	4953

According to the data presented, the Roberta model achieved an accuracy of 0.93, with precision at 0.77, recall at 0.80, and an F1-Score of 0.97.

5.2. Analysis and Self Reflection

This study leveraged sentiment analysis algorithms to pinpoint the most frequently occurring terms in feedback, identifying common words in both positive and negative reviews. By training a Bi-directional LSTM model with multiple layers and a batch size of 64, and creating a specialized stop-word dictionary through tokenization and

embedding, the study successfully filtered out common terms. This process highlighted key phrases like "disappointed," "great," "good," "well," "would," "nice," "what's," and "don't."

A significant reduction in execution time was observed when utilizing a GPU environment compared to a CPU-only setup, underscoring the importance of hardware in model training efficiency. The Bi-LSTM model achieved a 94% accuracy rate after 10 epochs, slightly outperforming the aROBERTa model's 93% accuracy. However, aROBERTa demonstrated faster training times, with 414 seconds compared to Bi-LSTM's 500 seconds.

The challenge of emotion detection from text highlights the need for ongoing research to enhance model accuracy and generalization. This study contributes valuable insights into model performance across different datasets, despite facing issues like uneven sample distribution and lack of data preprocessing.

Future research should address these limitations by adjusting model weights and biases or using a more balanced dataset. Preprocessing the data thoroughly can also improve model accuracy. Additionally, exploring various linguistic features could offer deeper understanding into classification outcomes.

The study's findings underscore the effectiveness of transfer learning in boosting model performance. Fine-tuning pre-trained models for specific tasks, such as pattern classification, significantly enhances their capabilities. This approach allows models to build upon a general knowledge base, facilitating more focused and effective learning for particular domains or tasks.

6. Discussion

6.1. Interpretation of Results

This study's analysis reveals closely matched validation accuracy results between the Bi-LSTM and RoBERTa models, with a negligible difference of less than 0.1%. Despite the Bi-LSTM's larger size, its accuracy closely paralleled that of RoBERTa under the applied fine-tuning conditions, suggesting that exploiting RoBERTa's potential may require larger datasets and more extensive parameter testing. Both models demonstrated commendable performance, as depicted in Tables 4 and 5, with Bi-LSTM slightly outperforming RoBERTa.

In scenarios prioritizing hit maximization, both Bi-LSTM and RoBERTa stood out, with Bi-LSTM showing a slight edge due to its scale and unchanged accuracy in comparative tests. However, for environments with hardware constraints, smaller models like BERT could offer a viable alternative [68].

Training times varied directly with model size; Bi-LSTM was the quickest, while RoBERTa took the longest. This distinction highlights Bi-LSTM's advantage in contexts where time and resources are limited, offering a cost-effective and efficient solution. Figures 16 and 17 illustrate the comparative training durations, using RoBERTa's time as a benchmark.

The models demonstrated superior generalization on the Reddit dataset over the Amazon Product Review dataset. Despite improvements from additional data, achieving an 80% generalization benchmark remained elusive. Label-wise performance analysis revealed nuanced strengths and weaknesses in both models, with batch size adjustments notably enhancing RoBERTa's capability to predict "surprise" in Amazon datasets.

Surprisingly, the RoBERTa model, trained solely on natural language, achieved F-scores comparable to those of other models trained on code-based languages, as shown in Tables 4 and 5. This underscores the adaptability of pre-trained language models to diverse domains and tasks, affirming their utility in various classification challenges. However, performance can vary by dataset and task, underscoring the need for comprehensive evaluation and consideration of other factors in real-world applications.

The Bi-LSTM model, following a Robustly Optimized LSTM Pretraining Approach, showed significant improvement after adjustments such as larger batch sizes, extended training, and dynamic masking. This approach yielded better results than the RoBERTa model, suggesting that meticulous fine-tuning of pre-trained models is crucial for enhancing task-specific performance [72].

The study's findings indicate that weakly supervised learning effectively categorizes data, with high performance suggesting the approach's efficacy in identifying key data patterns. The comparison between Bi-LSTM and RoBERTa, before and after applying knowledge transfer, highlighted the transformative impact of fine-tuning on model performance, despite their foundational similarities.

6.2. Limitations

This study acknowledges several limitations that merit attention. A notable challenge is the unequal distribution of samples across different labels, which could undermine the generalizability of both Bi-LSTM and RoBERTa models. Future research should consider applying weights and biases to mitigate this imbalance. Additionally, our approach did

not involve preprocessing the Poem data to eliminate newline characters, potentially affecting model performance.

Furthermore, the failure of any model to surpass a 90 percent accuracy rate raises questions about their ability to generalize effectively to new, unseen data. This limitation may stem from the subjective nature of emotions and the inherent complexity of linguistic expression. As such, advancing the field of emotion detection will require innovative strategies aimed at boosting both the accuracy and generalizability of these models.

6.3. Implications

The insights derived from Chapter 4 offer significant value to professionals and researchers interested in employing the Bi-LSTM model for sentiment analysis projects. The chapter elucidates that the choice of model hinges on various factors, including hardware limitations and time constraints. For projects aiming to maximize accuracy with identical parameters and datasets, the Bi-LSTM model emerges as a more resource-efficient option, providing superior performance without unnecessary expenditure of resources. This makes it a beneficial benchmark for evaluating other neural network models, not limited to those derived from Bi-LSTM architecture.

An important consideration from the analysis of the RoBERTa model is its potential underutilization. Given RoBERTa's design for larger datasets and extensive parameter use, akin to its pre-training conditions, employing similar parameters as those used for other models might not unleash its full capabilities. Therefore, leveraging larger batch sizes and more expansive datasets could significantly enhance RoBERTa's performance.

Furthermore, the study identifies a crucial dataset limitation: an imbalance in label distribution, particularly a notable disparity in the joy category compared to others and an exceedingly low number of samples for surprise. Addressing this issue by acquiring a larger, more balanced dataset is imperative for a thorough evaluation of model efficacy across various emotional categories.

6.4. Recommendations for Future Research

This study has demonstrated that integrating the seq2seq code representation technique with pre-trained RoBERTa-based models yields commendable performance in sentiment classification. These outcomes are encouraging and offer applicable insights across various domains requiring classification, laying a robust groundwork for subsequent research and advancements in this area.

For future endeavors, the study recommends exploring diverse vectorization methods; Word2Vec is a well-known semantic model, but the academic community continuously develops new models annually. Additionally, experimenting with various pre-processing techniques is advised. Due to the limitations imposed by sample size and the unique characteristics of the documents analyzed, the full range of possible technique combinations and experiments remains unexplored. Evaluating different classification models, such as those combining transformer architectures like RoBERTa-LSTM, could also prove beneficial. As interest in this research area grows, the addition of new samples, particularly those underrepresented in current classifications, could enrich the dataset.

Several avenues for further investigation merit attention. An especially compelling line of inquiry is the investigation of how different linguistic elements impact the classification process. For example, examining the influence of poetic devices such as rhyme, alliteration, and meter on classification outcomes could provide valuable insights into the linguistic features most critical to classification. Additionally, such exploration might elucidate the underlying causes of common classification challenges, including the differentiation between emotions like happiness and dissatisfaction.

7. Conclusion

This research evaluates the efficacy of various sentiment analysis and processing models, utilizing a comprehensive array of techniques and concepts from the field of natural language processing (NLP). Techniques such as tokenizing, stemming, and lemmatization, along with term embedding and counting methods like Bag of Words and transformers, were employed. The study focused on assessing two classification models: Roberta and Bi-directional Long Short-Term Memory (Bi-LSTM), encountering challenges related to sample size and the diversity of models tested.

A novel LSTM model was proposed to enhance sentiment analysis capabilities. This model improved upon traditional LSTM architecture by incorporating a larger batch size of 64 and adding four additional layers, including a bi-directional layer, an embedding layer with tokenizer, and two dense layers with 128 units each utilizing the sigmoid activation function. The study compared this advanced LSTM model against the conventional Roberta model using the Amazon product review dataset, aiming to determine the optimal batch size and number of epochs for fine-tuning the LSTM model. Unlike previous studies which utilized smaller batch sizes and fewer epochs, this research observed overfitting from the third epoch onward with the LSTM model, leading to adjustments in the number of epochs to two. After these modifications, the Bi-LSTM model achieved a validation accuracy of 0.95, suggesting that increasing layers in the LSTM model renders Roberta less effective, contrary to the widespread belief in Roberta's superior accuracy.

Optimization and access to more powerful computing resources addressed the challenges encountered, particularly in vectorizing the solution flow of the neural network models. A new flow utilizing the Word2Vec model was developed for text sample vectorization during training and evaluation. The analysis indicated that while some linear statistical models performed adequately, deep neural networks consistently outperformed them, showing greater stability across various scenarios with minimal variance. Among these, the Deep Neural Network (DNN) model exhibited the best average performance.

The study also highlighted a tendency for the models to exhibit higher precision than sensitivity, potentially leading to a greater number of false negatives rather than false positives. Implementing an NLP model requires establishing a robust architecture and applying diverse strategies to meet the research objectives. Initial data exploration revealed the presence of semantically insignificant words that impeded model performance. The development of a cleaning technique allowed for the input of more concise information into the model by removing stop words, punctuation, and irrelevant emojis, thereby enhancing the accuracy of review text tagging.

References

- [1] Hashem, I.A.T., Yaqoob, I., Anuar, N.B., Mokhtar, S., Gani, A. and Khan, S.U., 2015. The rise of “big data” on cloud computing: Review and open research issues. *Information systems*, 47, pp.98-115.
- [2] Górriz, J.M., Ramírez, J., Ortíz, A., Martínez-Murcia, F.J., Segovia, F., Suckling, J., Leming, M., Zhang, Y.D., Álvarez-Sánchez, J.R., Bologna, G. and Bonomini, P., 2020. Artificial intelligence within the interplay between natural and artificial computation: Advances in data science, trends and applications. *Neurocomputing*, 410, pp.237-270.
- [3] Wankhade, M., Rao, A.C.S. and Kulkarni, C., 2022. A survey on sentiment analysis methods, applications, and challenges. *Artificial Intelligence Review*, 55(7), pp.5731-5780.
- [4] Mineiro, A., Báez-Montero, I.C., Moita, M., Galhano-Rodrigues, I. and Castro-Caldas, A., 2021. Disentangling pantomime from early sign in a new sign language: window into language evolution research. *Frontiers in Psychology*, 12, p.640057.
- [5] Ben-Ami, Z., Feldman, R. and Rosenfeld, B., 2014. Using multi-view learning to improve detection of investor sentiments on twitter. *Computación y Sistemas*, 18(3), pp.477-490.
- [6] Huang, L., Dou, Z., Hu, Y. and Huang, R., 2019. Textual analysis for online reviews: A polymerization topic sentiment model. *IEEE access*, 7, pp.91940-91945.

- [7] Dey, L., Chakraborty, S., Biswas, A., Bose, B. and Tiwari, S., 2016. Sentiment analysis of review datasets using naive bayes and k-nn classifier. arXiv preprint arXiv:1610.09982.
- [8] Broniarczyk, S.M. and Griffin, J.G., 2014. Decision difficulty in the age of consumer empowerment. *Journal of Consumer Psychology*, 24(4), pp.608-625.
- [9] Chiroma, H., Abdullahi, U.A., Alarood, A.A., Gabralla, L.A., Rana, N., Shuib, L., Hashem, I.A.T., Gbenga, D.E., Abubakar, A.I., Zeki, A.M. and Herawan, T., 2018. Progress on artificial neural networks for big data analytics: a survey. *IEEE Access*, 7, pp.70535-70551.
- [10] Lindsay, G.W., 2021. Convolutional neural networks as a model of the visual system: Past, present, and future. *Journal of cognitive neuroscience*, 33(10), pp.2017-2031.
- [11] Weis, J., Spilger, P., Billaudelle, S., Stradmann, Y., Emmel, A., Müller, E., Breitwieser, O., Grübl, A., Ilmberger, J., Karasenko, V. and Kleider, M., 2020. Inference with artificial neural networks on analog neuromorphic hardware. In *IoT Streams for Data-Driven Predictive Maintenance and IoT, Edge, and Mobile for Embedded Machine Learning: Second International Workshop, IoT Streams 2020, and First International Workshop, ITEM 2020, Co-located with ECML/PKDD 2020, Ghent, Belgium, September 14-18, 2020, Revised Selected Papers 2* (pp. 201-212). Springer International Publishing.
- [12] Ding, B., Qian, H. and Zhou, J., 2018, June. Activation functions and their characteristics in deep neural networks. In *2018 Chinese control and decision conference (CCDC)* (pp. 1836-1841). IEEE.
- [13] Dombi, J. and Jónás, T., 2022. Generalizing the sigmoid function using continuous-valued logic. *Fuzzy Sets and Systems*, 449, pp.79-99.
- [14] Eisenstein, J., 2019. *Introduction to natural language processing*. MIT press.
- [15] Lauriola, I., Lavelli, A. and Aiolli, F., 2022. An introduction to deep learning in natural language processing: Models, techniques, and tools. *Neurocomputing*, 470, pp.443-456.
- [16] Chowdhary, K. and Chowdhary, K.R., 2020. Natural language processing. *Fundamentals of artificial intelligence*, pp.603-649.
- [17] Schoot Uiterkamp, L., 2019. Improving text representations for NLP from bags to strings of words (Master's thesis, University of Twente).
- [18] Wadud, M.A.H., Mridha, M.F. and Rahman, M.M., 2022. Word embedding methods for word representation in deep learning for natural language processing. *Iraqi Journal of Science*, pp.1349-1361.
- [19] Bakarov, A., 2018. A survey of word embeddings evaluation methods. arXiv preprint arXiv:1801.09536.
- [20] Khan, M.U., 2022. *Embeddings for Product Data* (Doctoral dissertation).

- [21] Karani, D., 2018. Introduction to word embedding and word2vec. Towards Data Science, 1.
- [22] Santhanam, S., 2020. Context based text-generation using lstm networks. arXiv preprint arXiv:2005.00048.
- [23] Maitama, J.Z., Idris, N., Abdi, A., Shuib, L. and Fauzi, R., 2020. A systematic review on implicit and explicit aspect extraction in sentiment analysis. IEEE Access, 8, pp.194166-194191.
- [24] Azzouzi, M.E., Coatrieux, G., Bellafqira, R., Delamarre, D., Riou, C., Oubenali, N., Cabon, S., Cuggia, M. and Bouzillé, G., 2024. Automatic de-identification of French electronic health records: a cost-effective approach exploiting distant supervision and deep learning models. BMC Medical Informatics and Decision Making, 24(1), pp.1-18.
- [25] Sun, P., Yang, X., Zhao, X. and Wang, Z., 2018, November. An overview of named entity recognition. In 2018 International Conference on Asian Language Processing (IALP) (pp. 273-278). IEEE.
- [26] He, Q. and Yang, B., 2018. A corpus-based study of the correlation between text technicality and ideational metaphor in English. Lingua, 203, pp.51-65.
- [27] Zennaki, O., Semmar, N. and Besacier, L., 2019. A neural approach for inducing multilingual resources and natural language processing tools for low-resource languages. Natural language engineering, 25(1), pp.43-67.
- [28] Lin, J.C.W., Shao, Y., Djenouri, Y. and Yun, U., 2021. ASRNN: A recurrent neural network with an attention model for sequence labeling. Knowledge-Based Systems, 212, p.106548.
- [29] Bakhashwain, N. and Sagheer, A., 2021. Online Tuning of Hyperparameters in Deep LSTM for Time Series Applications. International Journal of Intelligent Engineering & Systems, 14(1).
- [30] Peng, Y., Yan, S. and Lu, Z., 2019. Transfer learning in biomedical natural language processing: an evaluation of BERT and ELMo on ten benchmarking datasets. arXiv preprint arXiv:1906.05474.
- [31] Liu, W., Wen, B., Gao, S., Zheng, J. and Zheng, Y., 2020. A multi-label text classification model based on ELMo and attention. In MATEC Web of Conferences (Vol. 309, p. 03015). EDP Sciences.
- [32] Reece, A., Cooney, G., Bull, P., Chung, C., Dawson, B., Fitzpatrick, C., Glazer, T., Knox, D., Liebscher, A. and Marin, S., 2023. The CANDOR corpus: Insights from a large multimodal dataset of naturalistic conversation. Science Advances, 9(13), p.eadf3197.

- [33] Zebari, R., Abdulazeez, A., Zeebaree, D., Zebari, D. and Saeed, J., 2020. A comprehensive review of dimensionality reduction techniques for feature selection and feature extraction. *Journal of Applied Science and Technology Trends*, 1(2), pp.56-70.
- [34] Klaiber, M., Sauter, D., Baumgartl, H. and Buettner, R., 2021, July. A systematic literature review on transfer learning for 3d-CNNs. In *2021 international joint conference on neural networks (IJCNN)* (pp. 1-10). IEEE.
- [35] Vrbančič, G. and Podgorelec, V., 2020. Transfer learning with adaptive fine-tuning. *IEEE Access*, 8, pp.196197-196211.
- [36] Wang, Y., Pan, Z., Yuan, X., Yang, C. and Gui, W., 2020. A novel deep learning based fault diagnosis approach for chemical process with extended deep belief network. *ISA transactions*, 96, pp.457-467.
- [37] Zhang, Y. and Yang, Q., 2018. An overview of multi-task learning. *National Science Review*, 5(1), pp.30-43.
- [38] Howard, J. and Ruder, S., 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.
- [39] Qasim, R., Bangyal, W.H., Alqarni, M.A. and Ali Almazroi, A., 2022. A fine-tuned BERT-based transfer learning approach for text classification. *Journal of healthcare engineering*, 2022.
- [40] Hermansson, E. and Boddien, C., 2020. Using pre-trained language models for extractive text summarisation of academic papers.
- [41] Hossain, A.B., 2022. Multi-Teacher Knowledge Distillation Using Teacher's Domain Expertise (Doctoral dissertation, North Dakota State University).
- [42] Zhou, B., Zou, L., Mostafavi, A., Lin, B., Yang, M., Gharaibeh, N., Cai, H., Abedin, J. and Mandal, D., 2022. VictimFinder: Harvesting rescue requests in disaster response from social media with BERT. *Computers, Environment and Urban Systems*, 95, p.101824.
- [43] Saabith, S., Vinothraj, T. and Fareez, M., 2021. A review on Python libraries and Ides for Data Science. *Int. J. Res. Eng. Sci*, 9(11), pp.36-53.
- [44] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M. and Davison, J., 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.
- [45] Tunstall, L., Von Werra, L. and Wolf, T., 2022. Natural language processing with transformers. "O'Reilly Media, Inc."
- [46] Liu, G. and Guo, J., 2019. Bidirectional LSTM with attention mechanism and convolutional layer for text classification. *Neurocomputing*, 337, pp.325-338.

- [47] Althelaya, K.A., El-Alfy, E.S.M. and Mohammed, S., 2018, April. Evaluation of bidirectional LSTM for short-and long-term stock market prediction. In 2018 9th international conference on information and communication systems (ICICS) (pp. 151-156). IEEE.
- [48] Tan, K.Q., 2023. Machine learning for email filtering and categorising (Doctoral dissertation, UTAR).
- [49] Zhao, R., Yan, R., Wang, J. and Mao, K., 2017. Learning to monitor machine health with convolutional bi-directional LSTM networks. *Sensors*, 17(2), p.273.
- [50] Bowden, G.J., Maier, H.R. and Dandy, G.C., 2002. Optimal division of data for neural network models in water resources applications. *Water resources research*, 38(2), pp.2-1.
- [51] Semary, N.A., Ahmed, W., Amin, K., Pławiak, P. and Hammad, M., 2023. Improving sentiment classification using a RoBERTa-based hybrid model. *Frontiers in Human Neuroscience*, 17.
- [52] Krstinić, D., Braović, M., Šerić, L. and Božić-Štulić, D., 2020. Multi-label classifier performance evaluation with confusion matrix. *Computer Science & Information Technology*, 1.
- [53] Zhou, X., Lu, P., Zheng, Z., Tolliver, D. and Keramati, A., 2020. Accident prediction accuracy assessment for highway-rail grade crossings using random forest algorithm compared with decision tree. *Reliability Engineering & System Safety*, 200, p.106931.
- [54] Avanzato, R. and Beritelli, F., 2020. Automatic ECG diagnosis using convolutional neural network. *Electronics*, 9(6), p.951.
- [55] Miao, J. and Zhu, W., 2022. Precision–recall curve (PRC) classification trees. *Evolutionary intelligence*, 15(3), pp.1545-1569.
- [56] Kwon, Y.J., Toussie, D., Azour, L., Concepcion, J., Eber, C., Reina, G.A., Tang, P.T.P., Doshi, A.H., Oermann, E.K. and Costa, A.B., 2020, November. Appropriate Evaluation of Diagnostic Utility of Machine Learning Algorithm Generated Images. In *Machine Learning for Health* (pp. 179-193). PMLR.
- [57] Agbesi, V.K., Chen, W., Yussif, S.B., Hossin, M.A., Ukwuoma, C.C., Kuadey, N.A., Agbesi, C.C., Abdel Samee, N., Jamjoom, M.M. and Al-antari, M.A., 2023. Pre-Trained Transformer-Based Models for Text Classification Using Low-Resourced Ewe Language. *Systems*, 12(1), p.1.
- [58] Roumeliotis, K.I., Tselikas, N.D. and Nasiopoulos, D.K., 2024. LLMs in e-commerce: A comparative analysis of GPT and LLaMA models in product review evaluation. *Natural Language Processing Journal*, 6, p.100056.
- [59] Rajapaksha, P., Farahbakhsh, R. and Crespi, N., 2021. Bert, xlnet or roberta: the best transfer learning model to detect clickbaits. *IEEE Access*, 9, pp.154704-154716.

- [60] Teodoro, G., Sachetto, R., Sertel, O., Gurcan, M.N., Meira, W., Catalyurek, U. and Ferreira, R., 2009, August. Coordinating the use of GPU and CPU for improving performance of compute intensive applications. In 2009 IEEE International Conference on Cluster Computing and Workshops (pp. 1-10). IEEE.
- [61] Nandi, G. and Sharma, R.K., 2020. Data Science fundamentals and practical approaches: understand why data science is the next. BPB Publications.
- [62] Kimm, H., Paik, I. and Kimm, H., 2021, December. Performance comparison of TPU, GPU, CPU on Google colab over distributed deep learning. In 2021 IEEE 14th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoc) (pp. 312-319). IEEE.
- [63] Granero-Moya, M., Karanasou, P., Karlapati, S., Schnell, B., Peinelt, N., Moinet, A. and Drugman, T., 2023. A comparative analysis of pretrained language models for text-to-speech. arXiv preprint arXiv:2309.01576.
- [64] Rácz, A., Bajusz, D. and Héberger, K., 2021. Effect of dataset size and train/test split ratios in QSAR/QSPR multiclass classification. *Molecules*, 26(4), p.1111.
- [65] Gupta, S., He, P., Meister, C. and Su, Z., 2020, November. Machine translation testing via pathological invariance. In Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (pp. 863-875).
- [66] Jain, S.M., 2022. Hugging face. In Introduction to Transformers for NLP: With the Hugging Face Library and Models to Solve Problems (pp. 51-67). Berkeley, CA: Apress.
- [67] Du, X., 2022, May. OEmoBERTa: Only Using Emotion for Emotion Recognition in Conversation. In 2022 4th International Conference on Communications, Information System and Computer Engineering (CISCE) (pp. 631-636). IEEE.
- [68] Nityasya, M.N., Wibowo, H.A., Chevi, R., Prasojo, R.E. and Aji, A.F., 2022. Which student is best? A comprehensive knowledge distillation exam for task-specific BERT models. arXiv preprint arXiv:2201.00558.
- [69] Sharma, S., Sharma, S., Athaiya, A., 2017. Activation functions in neural networks. *Towards Data Sci* 6, 310–316
- [70] Minaee, S., Kalchbrenner, N., Cambria, E., Nikzad, N., Chenaghlu, M. and Gao, J., 2021. Deep learning--based text classification: a comprehensive review. *ACM computing surveys (CSUR)*, 54(3), pp.1-40.

[71] Feng, Z., Guo, D., Tang, D., Duan, N., Feng, X., Gong, M., Shou, L., Qin, B., Liu, T., Jiang, D. and Zhou, M., 2020. Codebert: A pre-trained model for programming and natural languages. arXiv preprint arXiv:2002.08155.

[72] Han, X., Zhang, Z., Ding, N., Gu, Y., Liu, X., Huo, Y., Qiu, J., Yao, Y., Zhang, A., Zhang, L. and Han, W., 2021. Pre-trained models: Past, present and future. AI Open, 2, pp.225-250.

Appendices

1. Google Drive (Shared) -
<https://drive.google.com/drive/folders/1DNzLu6sdQXVkaHq9wxcw3rXMQLyZO5Ny?usp=sharing>
2. ZIP file containing most of the files (Due to file size restrictions on Moodle, I had to omit some files. However, they are available at the following Google Drive).
3. Notebook Name: Pranjali Sinha - Bi-directional LSTM Model.ipynb
4. Notebook Name: Pranjali Sinha - roBERTa-base Model.ipynb
5. Datasets
 - 5.1. amazon_train.csv
 - 5.2. amazon_test.csv
 - 5.3. Reviews.csv
6. Other files
 - 6.1. glove.6B.100d.txt
 - 6.2. Readme.txt