REVIEWTAG: TAGGING AMAZON NEGATIVE PRODUCT REVIEW WITH DEEP LEARNING

by

Priyanka Kumari, MSCS

THESIS

Presented to the Faculty of

The University of Houston-Clear Lake

In Partial Fulfillment

Of the Requirements

For the Degree

MASTER OF SCIENCE

in Computer Science

THE UNIVERSITY OF HOUSTON-CLEAR LAKE

MAY, 2023

REVIEWTAG: TAGGING AMAZON NEGATIVE PRODUCT

REVIEW WITH DEEP LEARNING

by

Priyanka Kumari

APPROVED BY

_____

Kewei Sha, PhD, Chair

_____

Yalong Wu , PhD, Committee Member

_____

Wei Wei, PhD, Committee Member

RECEIVED/APPROVED BY THE COLLEGE OF SCIENCE AND ENGINEERING:

_____

David Garrison, PhD, Associate Dean

_____

Miguel A. Gonzalez, PhD, Dean

Dedication

I dedicate this thesis to my family. Their unwavering love and support have been a constant source of inspiration throughout my academic journey. I am grateful for their encouragement and belief in me, even during the most challenging times. Their sacrifices and understanding have allowed me to pursue my passions and reach this point. This thesis is a testament to their love and a small token of my appreciation for all that they have done for me.

Acknowledgements

My academic journey has been enriched by the unwavering guidance and support of my thesis advisor, Dr. Kewei Sha, to whom I am immensely grateful. Dr. Sha invested a significant amount of time and effort into reviewing my thesis, providing invaluable suggestions that greatly improved my work, and sharing his vast wisdom and expertise. I cannot express enough gratitude for his contributions.

I also extend my gratitude to Dr. Yalong Wu, Committee Member, for his valuable time, constructive discussions, insightful comments, and support. Additionally, I would like to thank Dr. Wei Wei for serving as a committee member and providing constructive feedback and guidance. Her suggestions significantly enhanced my understanding of the field and improved the quality of my work.

These professors and their expertise and dedication have been instrumental in helping me to complete this work. Their unwavering support and encouragement have inspired me to reach my full potential, and I am truly grateful for their patience and understanding throughout this exciting but challenging journey.

I also want to acknowledge the contributions of all those who have supported me in this journey, including my colleagues, friends, and family. Their assistance and motivation were essential in bringing this project to a successful conclusion.

Finally, I would like to thank the institution for its support and resources, including the library and computer facilities, which were critical for my research and the preparation of this thesis.

ABSTRACT

REVIEWTAG: TAGGING AMAZON NEGATIVE PRODUCT

REVIEW WITH DEEP LEARNING

Priyanka Kumari
University of Houston-Clear Lake, 2023

Thesis Chair: Dr. Kewei Sha

The success of Amazon sellers hinges on high ratings and meeting customer needs with exceptional products and services. However, the large scale of negative reviews pose significant challenges that require careful analysis to identify underlying reasons of buyers concerns. We aim to develop an automated tagging system named ReviewTag to address this challenge. The system uses deep learning models and Natural Language Processing (NLP) techniques to swiftly categorize negative reviews into two broader categories i.e., product issues and seller issues.

The system provides further insight into customers' specific issues using subtopic tagging, allowing Amazon sellers to identify areas for improvement and make data-driven decisions to meet evolving customer expectations.

We employ five deep learning models to perform topic and subtopic tagging. These models include Bidirectional Encoder Representations from Transformers (BERT), Distilled Bidirectional Encoder Representations from Transformers (DistilBERT), Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), and Recurrent Neural Network (RNN). Based on the evaluation with a prototype implementation of

ReviewTag, the BERT model demonstrates high precision, recall, and F1-scores of 0.97, 0.96, and 0.96, respectively, for topic tagging. Additionally, the BERT and CNN models show impressive precision, recall, and F1-scores of about 0.92, for subtopic tagging. These results demonstrate the effectiveness of deep learning models for automatically tagging negative product reviews on Amazon. It helps Amazon sellers take action to improve their product ratings.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I:

INTRODUCTION

1.1 Background and Significance

Amazon is a technology and online retail company that has succeeded in the e-commerce industry through advanced technology and innovative practices. The e-commerce industry encompasses the buying and selling of products and services using online platforms. This industry has seen substantial growth in recent years, with many businesses and consumers opting in for e-commerce as a convenient and efficient shopping method [1]. Amazon platform's customer feedback system enables shoppers to leave ratings and reviews, which offer valuable insights into the quality of products and services available on the platform. Amazon sellers can derive various advantages from customer feedback, whether it is positive or negative. Positive feedback can provide social proof of the quality of the sellers products or services, which can help build trust with potential customers and increase sales. It helps to increase a sellers visibility on the Amazon platform.

On the other hand, negative feedback not only helps other customers make informed buying decisions, but it also benefits the sellers by identifying areas for improvement and addressing any issues with their products or services. It helps sellers effectively to tackle the issues and elevate their product ranking on Amazon. The e-commerce platforms, such as Amazon, rely on customer ratings and reviews to assess the quality of products and services offered by sellers. Higher-rated products are more likely to the top of search results, resulting in increased sales and revenue for sellers. This creates a strong incentive for sellers to continuously improve their offerings to meet the needs of customers and maintain a high rating. To stay competitive in the e-commerce industry,

sellers must prioritize providing exceptional products and services that attract and retain customers, ultimately leading to revenue growth [2].

1.2 Motivation and Research Challenges

Amazon sellers are individuals or businesses that use Amazon's platform to sell their products and services to customers. They can operate independently or as a part of Amazon's Fulfillment by Amazon (FBA) program, where their products are stored in Amazon's warehouses, and Amazon handles the fulfillment and shipping process [3]. Amazon sellers offer various products and services, including clothing, shoes, and jewelry, playing a significant role in the e-commerce industry, and contributing to Amazon's success as a leading online marketplace. However, maintaining their product rankings can be difficult due to factors like analyzing vast amounts of customer feedback, particularly negative reviews.

Especially for high-volume sellers, analyzing customer feedback, particularly negative reviews, can be time-consuming and energy-intensive, requiring careful attention to maintain product rankings. Amazon sellers must meticulously read and analyze each customer's feedback to identify areas for improvement or issues that require attention to improving product descriptions, addressing complaints, and providing prompt and reliable customer service.

Negative customer reviews can significantly impact Amazon sellers businesses, affecting their reputation and revenue as it can give potential customers the impression that the seller's products or services are unsatisfactory, ultimately leading to lost sales. In addition, negative reviews can adversely affect a seller's search rankings, making it harder for them to be discovered by new customers [4]. Amazon sellers must prioritize offering excellent products and services while actively managing customer feedback to maintain a positive reputation and attract more customers.

The primary objective of this thesis is to develop a system called ReviewTag, which will briefly categorize negative reviews into categories and sub-categories, enabling sellers to identify areas of concern quickly. Implementing this approach can save sellers time while improving their understanding of customer needs, resulting in higher product ratings and tremendous success on the Amazon platform. The method for categorizing reviews is called "topic tagging". It involves tagging reviews into two primary categories: product issues and seller issues. Product issues relate to problems with the product itself, while seller issues relate to problems with the seller. The system will use "subtopic tagging" to provide further insight into the issues to break down each category into sub-categories. For product issues, subtopics will include design issues, quality issues, and product description issues. For seller issues, subtopics consist of product authentication issues and delivery and return issues.

1.3 Research Design and Results

1.3.1 Research Design

The automated system, ReviewTag based on deep learning algorithms is designed to assist sellers by classifying customer reviews into predetermined tags using NLP techniques. This system uses text tagging, a fundamental task in NLP, to provide insights into the reasons behind negative ratings. The system utilizes four automated techniques: Topic Tagging (TT), SubTopic Tagging with Known Topics (STWKT), SubTopic Tagging without Topic (STWOT), and SubTopic Tagging with Predicted Topics (STWPT). The techniques use deep learning models like Bidirectional Encoder Representations from Transformers (BERT), Distilled Bidirectional Encoder Representations from Transformers (DistilBERT), Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), and Recurrent Neural Network (RNN) to ensure accuracy and efficiency.

"Topic Tagging (TT)" is a technique of classifying reviews into broad categories, such as product issues and seller issues, to help sellers identify areas for improvement. Product issues relate to problems with the product itself, while seller issues relate to problems with the sellers service. Using topic tagging, sellers can easily understand what customers say about their products and services, enabling them to make better decisions and improve their business.

"SubTopic Tagging" is a technique used to classify reviews into specific subtopics within the broad categories of product issues and seller issues. This approach involves three different ways to tag subtopics: subtopic tagging with known topic, subtopic tagging without topic, and subtopic tagging with predicted topics. By usng known subtopics, sellers can get more in-depth insights into specific areas that need improvement and act accoordingly.

SubTopic Tagging with Known Topics (STWKT) is a technique of classifying reviews using predetermined subtopics within the broad categories of product issues and seller issues. This approach involves tagging the reviews with the known subtopics within each category, such as design issues, quality issues, and product description issues for product issues and product authentication issues, delivery and return issues for seller issues.

SubTopic Tagging without Topic (STWOT) is an alternative technique to subtopic tagging that uses machine learning algorithms to identify and classify subtopics without predetermined topic knowledge because topic information may not be available. This technique involves analyzing the text data of reviews and tagging them into subtopics.

SubTopic Tagging with Predicted Topics (STWPT) technique aims to improve the accuracy of subtopic tagging when topic information is not available. It uses the "Topic Tagging" method to predict the topic, then, the subsequent "SubTopic Tagging" process further categorizes the review, allowing sellers to pinpoint specific areas for improvement.

ReviewTag classifies customer reviews into predetermined tags and provide insights into the reasons of negative ratings. It includes topic tagging and subtopic tagging. These techniques help sellers identify areas for improvement, discover new areas, and make informed decisions to improve their products and services based on customer reviews.

1.3.2 Research Result

We evaluated the performance of the deep learning system using precision, recall, and F1 score. They compared the predicted categories with the actual categories assigned to the reviews by topic tagging. The findings showed that BERT had the highest F1 score of 0.96, with a precision of 0.97 and a recall of 0.96. DistilBERT, CNN, and LSTM achieved a similar F1 score of 0.95, with precision and recall scores ranging from 0.95 to 0.96. RNN had the lowest F1 score of 0.83, with a precision of 0.87 and a recall of 0.80.

The results show that BERT achieved the highest precision, recall, and F1 score for all subtopic tagging levels, ranging from 0.90 to 0.92. DistilBERT and CNN also achieved high scores, ranging from 0.85 to 0.92. However, LSTM and RNN models achieved lower scores, with RNN performing the worst. The research findings demonstrate the effectiveness of the deep learning models for automated tagging of negative product reviews on Amazon. Based on customer feedback, the system can provide valuable insights for sellers to improve their products and services.

1.4 Organization of Thesis

The thesis is divided into six chapters. Chapter I introduces the research topic, its background, and significance. It also outlines the motivation and challenges, design, and results. Chapter II reviews the relevant literature. Chapter III describes the process of acquiring the Amazon review dataset, creating the Amazon review tagged dataset, and tokenizing the dataset. Chapter IV presents the architecture of ReviewTag. Chapter V

describes the approaches to identify topics and subtopics. Finally, Chapter VI summarizes the research findings, limitation, and recommendations for future research studies.

CHAPTER II:

RELATED WORK

ReviewTag is an automated tagging system that uses deep learning models and Natural Language Processing (NLP) techniques to categorize negative reviews on Amazon into product and seller issues and subtopic tagging to provide more specific insight into customer concerns. Several related works share similarities with ReviewTag regarding using NLP and deep learning techniques to extract insights from online content.

One such related work is Topic modeling, a powerful technique to uncover hidden structures in large document collections [5,6]. It can help differentiate the usage of words with different meanings and link words with similar contexts. Topic modeling methods include Vector Space Model (VSM) [7], Latent Semantic Indexing (LSI) [8], Probabilistic Latent Semantic Analysis (PLSA) [9], and Latent Dirichlet Allocation (LDA). These methods have applications in text categorization, tag recommendation, keyword extraction, and similarity search in text mining and information retrieval, much like topic and subtopic tagging.

Another related work is blog mining, which involves extracting valuable information from blog data, such as tags and multidimensional data [10]. A tag-topic model for blog mining is introduced in one paper based on the Author-Topic model [11] and Latent Dirichlet Allocation [12]. This model determines the most likely tags and words for a given topic in a collection of blog posts, like how ReviewTag categorizes reviews into product and seller issues and subtopics.

The study on identifying and evaluating documented serious illness communication (SIC) relates to ReviewTag's use of NLP techniques to analyze text data [13]. However, this study focuses on identifying and characterizing documented serious illness

communication with oncology patients to measure healthcare quality, whereas ReviewTag categorizes negative reviews on Amazon.

Finally, a related work used the Sentence-Level Topic Model (SLTM) method to extract product features from smartphone reviews on Amazon [14, 15]. The SLTM method has been used previously to extract features from online reviews and can extract both explicit and implicit features from reviews. The SLTM method uses NLP and machine learning techniques to extract features from online reviews, like ReviewTag's use to categorize reviews into product and seller issues and subtopics.

In summary, these related works demonstrate the potential of NLP and machine learning techniques to extract insights from online content, much like ReviewTag. However, each related work has its unique focus, such as topic modeling, blog mining, SIC, and feature extraction. At the same time, ReviewTag specifically targets negative reviews on Amazon to help sellers make data-driven decisions.

CHAPTER III:

DESCRIPTION OF DATASET AND PROBLEM MODELING

Amazon reviews are essential to an Amazon seller. They offer valuable insights into how customers view their products, with negative reviews incredibly significant. Negative reviews can help sellers identify areas of improvement, maintain a favorable product ranking on Amazon, and address any issues or concerns that customers may have. By addressing negative feedback, sellers can improve their products and customer service, increasing sales and higher customer satisfaction.

3.1 Amazon Review Acquisition

The dataset used for this thesis, obtained from the University of California San Diego (UCSD), includes a vast collection of 142.8 million product reviews and metadata from Amazon [16, 17, 18]. These reviews were written by customers between May 1996 and Oct 2018, spanning a considerable duration and offering a diverse range of opinions and preferences. Notably, the dataset comprises various product categories, with Clothing, Shoes, and Jewelry being the most prevalent.

The thesis required a dataset with an overall rating and review text to train ReviewTag for sellers to analyze the products on Amazon. The UCSD-provided Amazon dataset was considered suitable for analysis due to its extensive attribute, overall rating, and review text.

3.2 Amazon Review Tagged Dataset

Amazon sellers must analyze many negative reviews to understand why their products receive poor ratings. Developing ReviewTag involves selecting reviews with one or two-star ratings, which indicates Negative Product Reviews, and then manually analyzing it. Manual tagging is preferred over automated tagging techniques because automated tagging relies on algorithms that may need to fully understand the context and

nuances of natural language, which can lead to incorrect or incomplete tagging. On the other hand, manual tagging allows human analysts to read and comprehend each review carefully, considering the overall sentiment, the issues raised, and any relevant subtopics. Although manual tagging may take longer than automated tagging, it tags reviews with specific criteria, such as product or service issues.

ReviewTag categorizes negative reviews into two primary categories: product issues and seller issues. The tags for product issues relate to problems with the product itself, while the tags for seller issues relate to problems with the seller. Each category is further broken-down using subtopics to provide a deeper understanding of the issues. For example, the tags for product issues may include design issues, quality issues, and product description issues, while those for seller issues may include product authentication issues, and delivery and return issues. Quality issues refer to problems with the material used, such as unpleasant odors in clothes or shoes. Design issues pertain to size or fit issues and may also encompass jewelry. Product description issues involve errors or inaccuracies in the product's details, such as incorrect sizing information. Meanwhile, the seller issues category is also divided into subtopics. These subtopics include product authentication and delivery and return issues. Product authentication issues denote the sale of counterfeit products, which can be a significant problem for both sellers and customers. Delivery and return issues refer to problems with the shipping process, such as delays in delivery or difficulties in returning products.

The system can identify patterns and themes that help sellers improve their products and services by analyzing thousands of reviews and using the tagging process. Figure 3.1 summarizes the hierarchy of tags in this work.

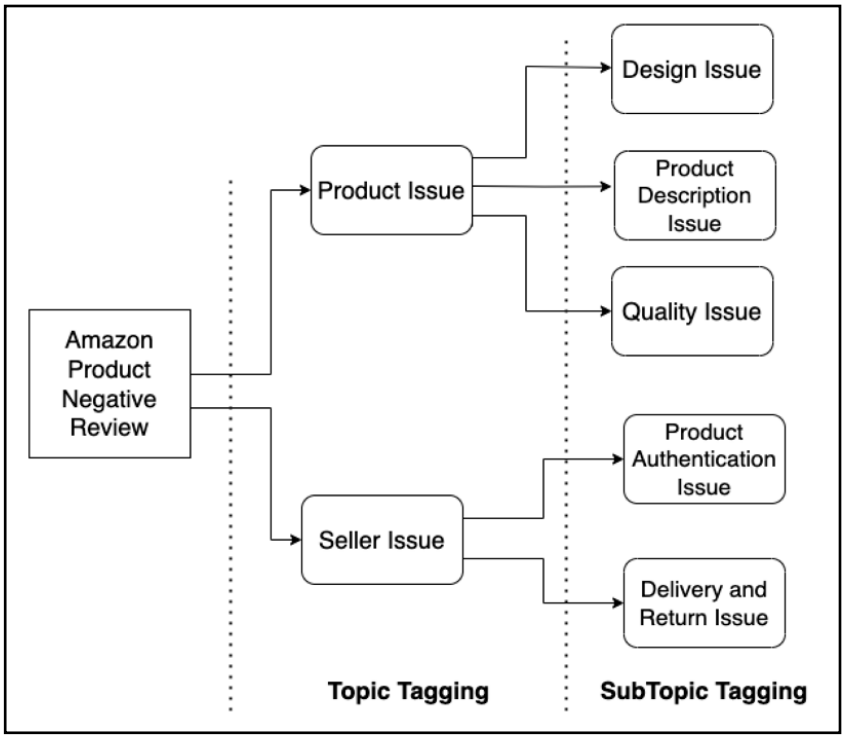

Figure 3.1: Categories of topics and subtopics

The UCSD Amazon dataset presents a challenge in obtaining an adequate number of negative reviews due to an imbalance between positive and negative datasets, which results in a lower percentage of negative reviews than positive ones. Specifically, the positive dataset has 26 million entries, while the negative dataset only has 4 million. Additionally, manual tagging of negative reviews can be difficult and time-consuming, further adding to the challenge of collecting adequate negative reviews.

Despite these difficulties, the entire negative review dataset within the UCSD Amazon dataset contains 12053 entries. For each entry, it will have a tag of a topic and a tag of SubTopics. The table 2.1 shows the count of reviews that have been tagged with two topics: Product Issue and Seller Issue. There are 7932 reviews tagged as Product Issue and 4121 reviews tagged as Seller Issue.

Table 2.1: Count of Topics

Topic	**Count**
Product Issue	7932
Seller Issue	4121

Table 2.2 presents a detailed breakdown of the count of tags for subtopics. The Product Issue topic has three subtopics: Quality Issue with 2012 reviews tagged, Design Issues with 2785 reviews tagged, and Product Description Issue with 3135 reviews tagged. However, collecting high-quality data for Seller Issues presents a challenge because issues related to product authenticity may require customers to possess specific knowledge about the product, such as its manufacturing process or the source of materials used. Similarly, delivery and return issues may require customers to have experienced certain situations, such as receiving damaged goods or undergoing extended wait times for delivery. The Seller Issue topic has two subtopics: Product Authenticity Issue with 2399 reviews tagged, and Delivery and Return Issue with 1722 reviews tagged.

Table 2.2: Count of SubTopics

Topic	SubTopic	**Count**
Product Issue	Quality Issues	2012
Product Issue	Design Issues	2785
Product Issue	Product Description Issues	3135
Seller Issue	Product Authenticity Issues	2399
Seller Issue	Delivery and Return Issues	1722

3.3 Amazon Review Data Transformation

The Amazon Negative Product Review-Manual Tagging has two columns: reviewText and tags, where reviewText includes the textual content of Amazon product reviews. However, deep learning algorithms require numerical inputs, so we must convert the reviewText data into a numerical format through tokenization. Tokenization breaks down the text data into individual tokens or words and assigns each token a numerical value that can be used as input for deep learning models.

NLP tasks, such as topic tagging on Amazon product reviews, often rely on popular tokenizer libraries. There are two commonly used tokenizer libraries for natural language processing: NLTK Word Tokenizer [20] for Vocab Tokenization and Hugging Face Tokenizer [21, 22] for Ids and Masks Tokenization. Both tokenizers are designed to capture as much information as possible from the input review text. However, Vocab Tokenization is typically used for LSTM, RNN, and CNN models, while Ids and Masks Tokenizers are used for BERT and DistilBERT models.

3.3.1 Vocab Tokenization:

The NLTK Word Tokenizer is a Python module in the Natural Language Toolkit (NLTK) library that provides functionality for tokenizing text into individual words [19]. The NLTK Word Tokenizer is a natural language processing tool that leverages "basic English" to break down raw text data into individual words, phrases, and sentences [20]. The primary objective of a tokenizer is to convert unstructured text data into structured data that can be readily analyzed and processed by machine learning models. The tokenizer typically removes punctuation marks and special characters and segments the text into individual words. This process helps to organize the text data and make it more manageable and structured for efficient analysis by machine learning models. In the upcoming section, we will perform text tokenization.

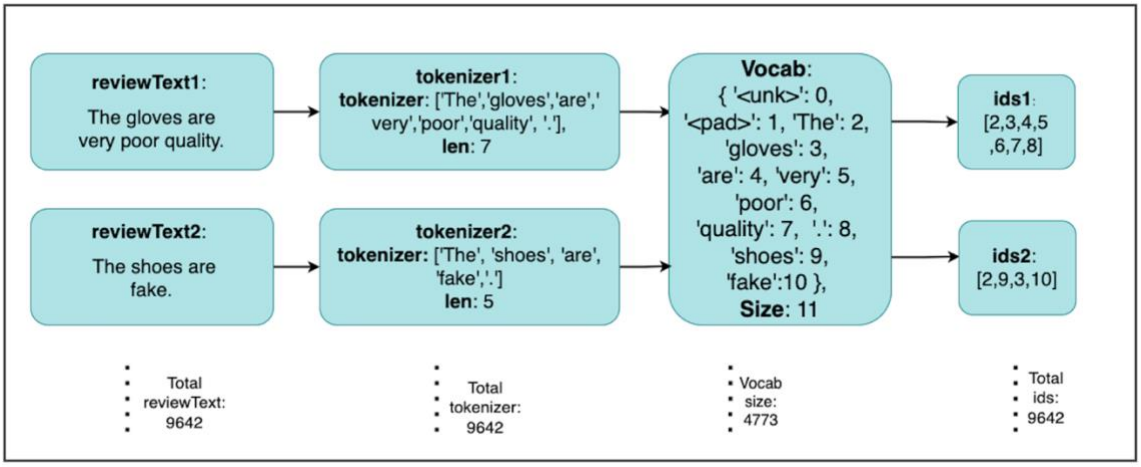

Figure 3.2: Text Tokenization

The process of tokenizing text involves several steps, as shown in Figure 3.2. The four main steps involved in creating a tokenized text are reviewText, tokenizer, vocab, and ids. First, to analyze text data, we need to use a tokenizer to break it down into smaller units of meaning, such as words or parts of words. Tokenization involves taking the text and dividing it into individual tokens. Figure 3.2 depicts the tokenization process, where the sentence "The gloves are of very poor quality." is transformed into individual tokens, ['The', 'gloves', 'are', 'of', 'very', 'poor', 'quality', '.']. The figure also shows reviewText1, reviewText2, ... reviewText9642, passed through a tokenizer to generate tokenizer1, tokenizer2...tokenizer9642.

Once we have broken down the text into individual tokens, we can generate a vocabulary or vocab comprising all the unique tokens. Figure 3.2 shows an example vocab that consists of eleven words, including two special reserved words. The first reserved word, "<unk>", is a placeholder for unknown words that the tokenizer cannot include in the vocab. The second reserved word, "<pad>", is used to pad token sequences to a fixed

length. The remaining nine words represent the actual tokens in the text. Our task is to tokenize the reviewText, which contains 9642 instances, likely to have repeated words, making the overall vocab smaller than the total number of tokens. In this example, the vocab has 4773 words, with the first few reserved for special purposes, such as unknown and padding words. These special reserved words have specific identifiers that enable us to represent the text as a sequence of integers.

Finally, each token is assigned a unique identifier or id to represent the text as a sequence of integers. In this figure 3.2, we have assigned the Ids [2, 3, 4, 5, 6, 7, 8] to the tokens in our vocab, starting from the third word, "The", which has an id of 2, and ending with the token "fake", which has an id of 10. These ids represent the reviewText data, such as reviewText1, reviewText2, ... reviewText9642, which is converted into a sequence of ids1, ids2... ids9642.

We focus is on padding input ids and creating batches, as shown in Figure 3.3, which illustrates the flow from input ids to batches. Deep learning models require inputs to be of equal length, while naturally, input ids can have varying lengths as different texts may have different lengths. To address this, we will pad the input ids with zeros to make them uniform, with a maximum length of 256. If an id is shorter than 256, we add zeros; if it is longer, we truncate it. This process is called padding, and we will apply it to all 9642 input ids, resulting in a padded dataset.

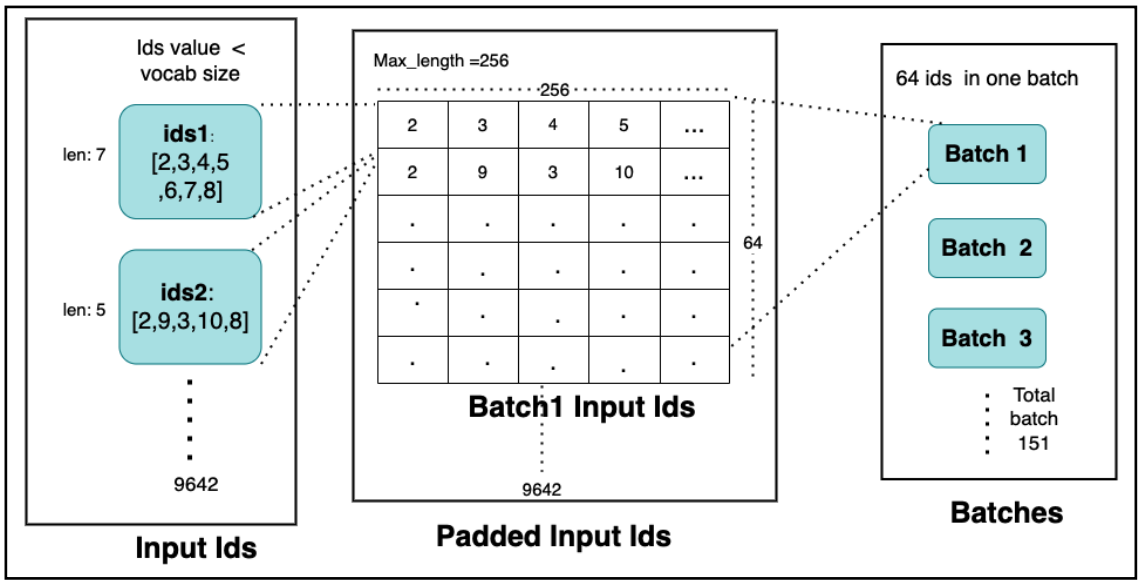

Figure 3.3: Padding Ids and batch

In the previous section, we discussed the importance of padding input ids and creating batches for deep learning models. In Figure 3.3, we divide the padded ids into 150 batches, with each batch consisting of 64 padded ids. Batches offer several advantages:

• It efficiently uses memory during training by avoiding loading the entire dataset simultaneously.

• It speeds up the learning process by enabling the model to update parameters more frequently by simultaneously processing smaller subsets of data.

• It helps to avoid overfitting by exposing the model to a more diverse set of examples in each iteration.

3.3.2 Ids and Masks Tokenizers

Tokenization is critical in preparing text data for machine learning models in NLP. It involves breaking down a text into its constituent words or sub words to enable the model to analyze it. The Hugging Face Tokenizer is a Python package for natural language processing that provides tokenization functionality for various state-of-the-art transformer

models, including BERT and DistilBERT [21, 22]. The library that offers state-of-the-art tokenization techniques based on transformer models. One of its critical features is the ability to fine-tune the techniques to perform topic tagging, a crucial capability for deep learning models. We can explore customer issues and preferences by applying Tokenizers to Amazon's Negative Product Reviews and identify recurring patterns or concerns.

Ids and Masks is dedicated to discussing input ids and masks for transformer models, such as BERT and DistilBERT batches. Figure 3.4 illustrates the primary components involved, which include reviewText, ids, and masks.

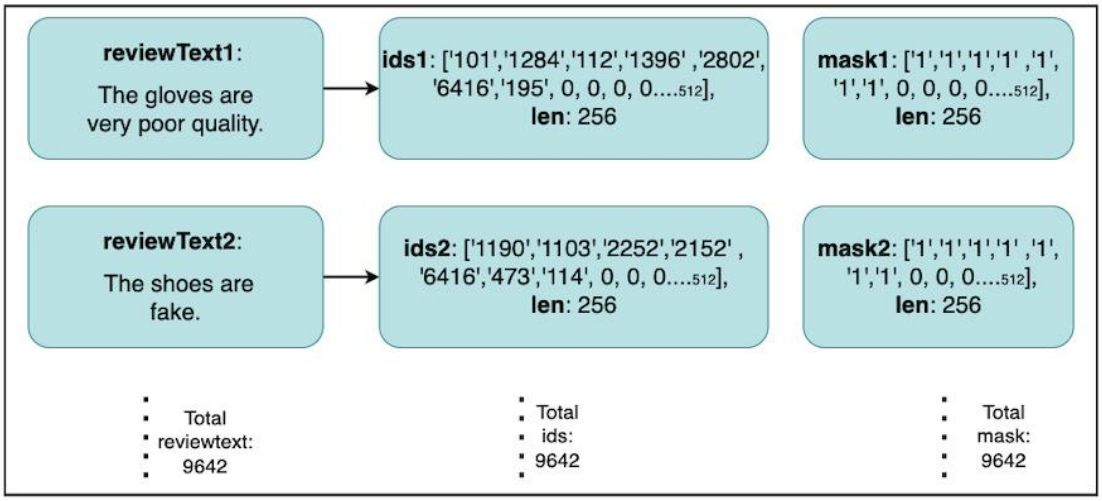

Figure 3.4: Ids and Masks

The mask tokenizer adds a special "mask" token to the input sequence, instructing the model to ignore specific input tokens during training. Transformer models require inputs to have equal length, thus we use a tokenizer such as Roberta-base or DistilBERT-base-uncased to convert reviewText into ids and masks. The total number of reviewText is 9642, resulting in 9642 ids and masks.

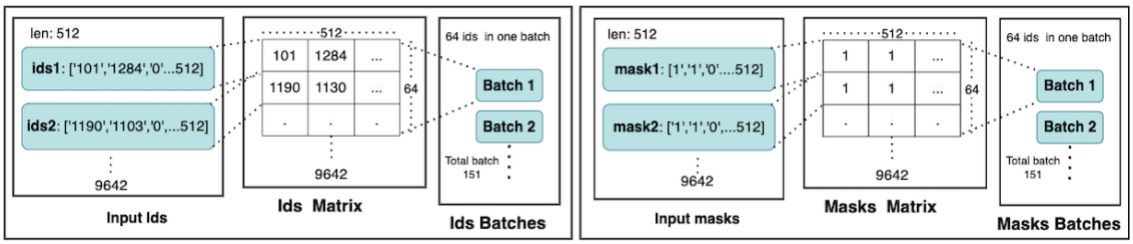

Figure 3.5: Ids Batches(left) and Masks Batches(right)

Figure 3.5, the left-hand side, illustrates the input ID batches, an ID matrix, and ID batches. The tokenizer assigns each token in the input text a unique ID, enabling the model to comprehend the text. The right-hand side displays the Masks batches, which consist of input masks, a mask matrix, and mask batches. A unique mask token is appended to the input sequence during tokenization, indicating the model ignores specific input tokens during training.

The 9642 input IDs and masks into 150 batches, each comprising 64 masks. Batch processing offers several advantages, such as efficient memory utilization, faster learning, and a reduced risk of overfitting. We can train our deep learning models effectively and achieve more accurate results by utilizing batch processing.

CHAPTER IV:

THEORETICAL FOUNDATIONS OF DEEP LEARNING

The thesis proposes developing ReviewTag that can aid sellers in managing customer reviews efficiently. The last chapter involved the preparation of the Amazon Review Tagged Dataset Chapter 3.2, which was then pre-processed using a tokenizer in the Amazon Review Data Transformation Chapter 3.3. In this chapter, we will introduce the concepts of NLP, such as hyperparameters, model architecture, and deep learning models, along with overview of the Automated Tagging System.

4.1 Automated Tagging System

We are developing an automated tagging system for negative reviews of Amazon products, utilizing state-of-the-art deep learning techniques such as BERT, DistilBERT, LSTM, RNN, and CNN. This system aims to enhance the visibility of Amazon sellers by effectively identifying and tagging negative reviews based on their topics. The system offers topic tagging and three distinct subtopic tagging methods: subtopic tagging without topic, subtopic tagging with topic, and subtopic tagging with predicted topics.

4.1.1 Topic Tagging (TT)

Topic tagging helps sellers categorize reviews into broader categories like product issues and seller issues, which assists them in identifying areas that need improvement. Product issues usually stem from problems with the product, while seller issues arise from problems with the seller's service. Figure 4.1 depicts implementing TT system using a deep learning model called the topic classifier. The system takes Amazon's negative reviews as input, and the topic classifier analyzes the text of each review and predicts the related product or seller issue based on the words and phrases used in the review.

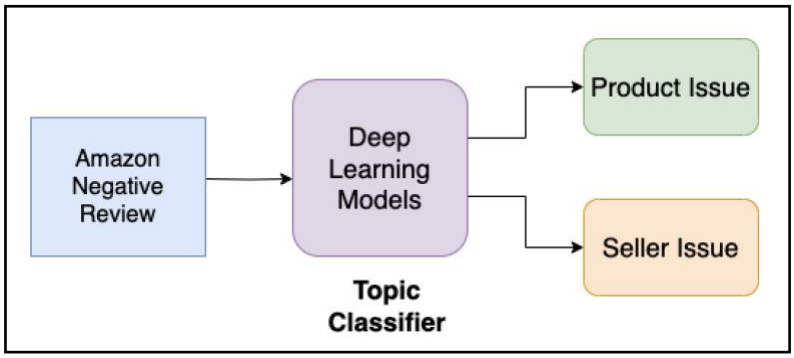

Figure 4.1: Topic Tagging (TT)

4.1.2 SubTopic Tagging without Known Topic (STWOT)

SubTopic Tagging without a Topic can classify subtopics such as design issues, quality issues, and product description issues for product issues and product authentication issues, delivery and return issues for seller issues without predetermined topic knowledge.

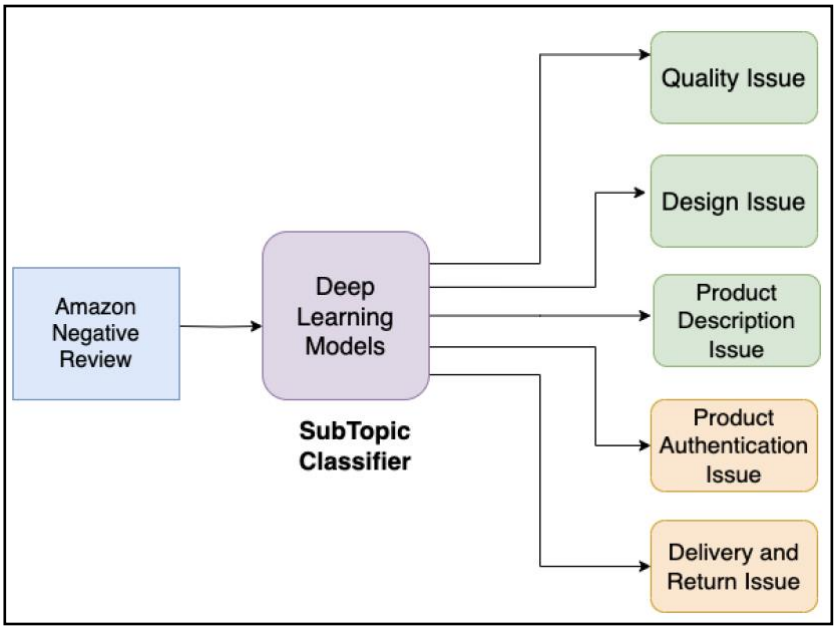

Figure 4.2: SubTopic Tagging without Topic (STWOT)

The implementation of STWKT system using a deep learning model called a subtopic classifier is depicted in Figure 4.2. After taking in Amazon product reviews as input, the subtopic classifier analyzes the text of each review. It predicts the related subtopic issues based on the words and phrases used in the review.

4.1.3 SubTopic Tagging with Known Topic (STWKT)

STWKT involves further categorizing reviews into predetermined subtopics under the broader categories of product and seller issues. This method entails labeling the reviews with specific subtopics within each category, such as design issues, quality issues, and product description issues for product issues, product authentication issues, delivery and return issues for seller issues.

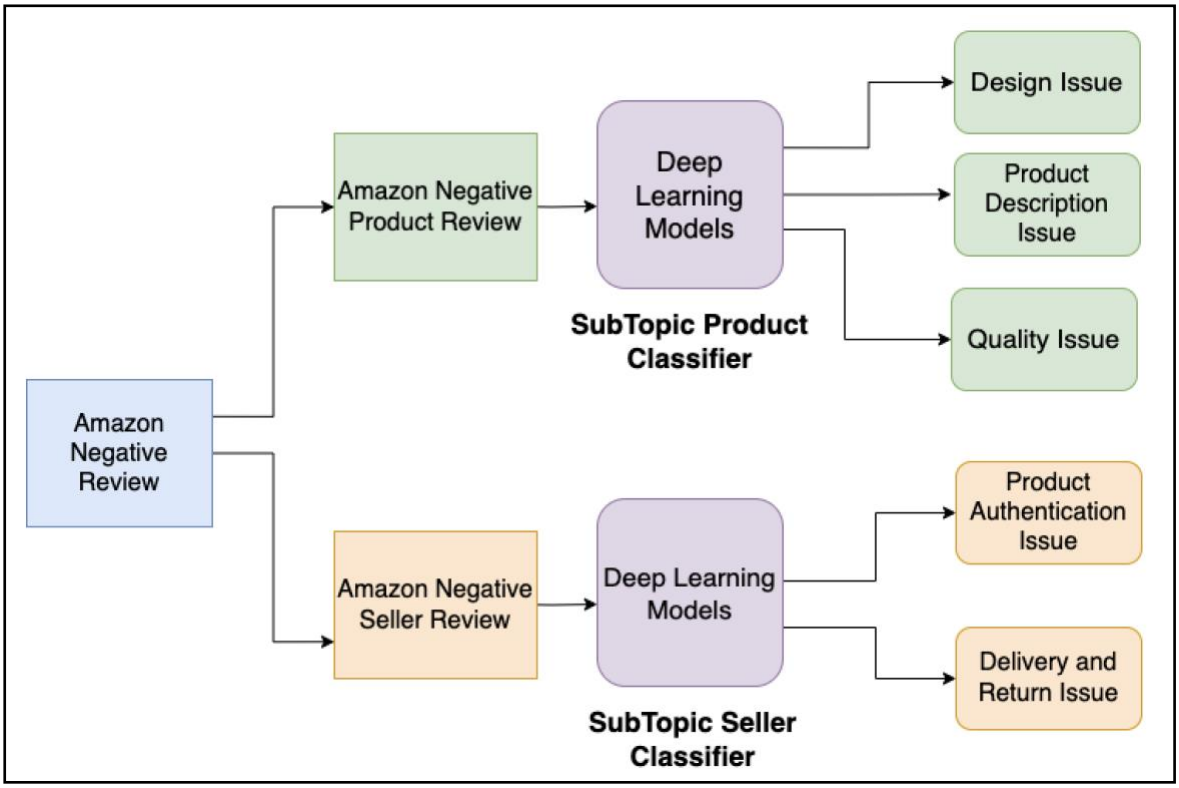

Figure 4.3: SubTopic Tagging with Known Topic (STWKT)

In Figure 4.3, the implementation of the STWOT system employs a deep learning model called the SubTopic product and SubTopic seller classifiers for analyzing product issues and seller issues reviews, respectively. Amazon negative reviews are separated into two categories: Amazon negative product reviews and Amazon negative seller reviews. The system takes Amazon's negative product review as input, and the SubTopic product classifiers analyze the text of each review to predict the related subtopics of product issues. Similarly, the system takes Amazon's negative seller review as input, and the SubTopic seller classifiers analyze the text of each review to predict the related subtopics of seller issues.

4.1.4 SubTopic Tagging with Predicted Topics (STWPT)

The "SubTopic Tagging with Predicted Topics" technique can be an effective tool for Amazon sellers without labeled datasets for negative reviews of their products. This technique predicts the negative review's topic and the subtopic based on the predicted topic and the negative review.

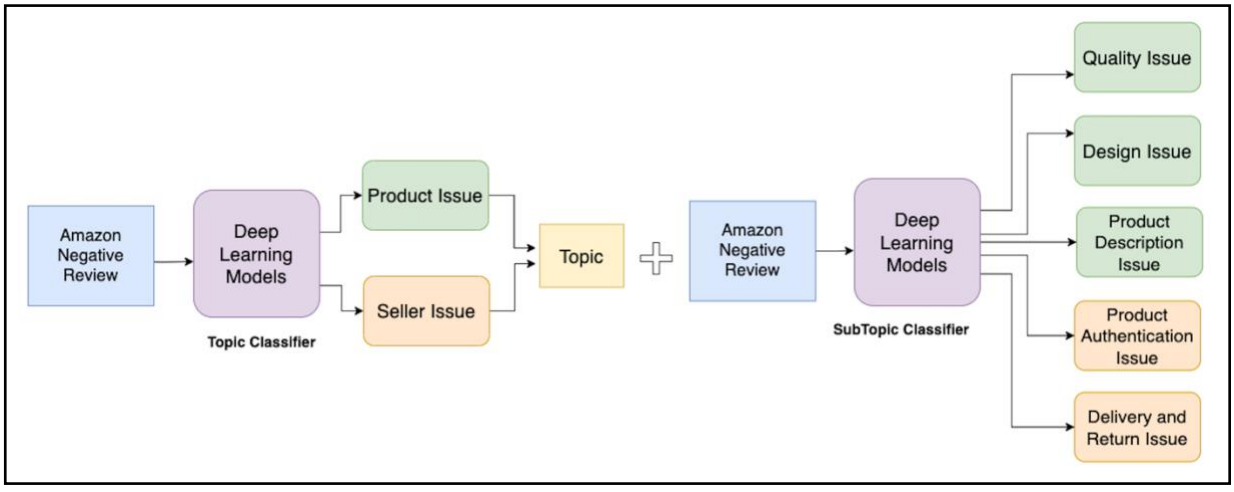

Figure 4.4: SubTopic Tagging with Predicted Topic (STWPT)

In Figure 4.4, Amazon negative review is first passed to a topic classifier. The topic classifier predicts the topic of the review. The predicted topic and the Amazon negative review are concerted and passed to a subtopic classifier. The subtopic classifier uses the predicted topic to predict the review's subtopic accurately.

4.2 Introduction of Neural Network

Neural Networks are a powerful machine learning techniques used extensively in Natural Language Processing NLP to perform various tasks, such as topic tagging. One key aspect of neural networks is using hyperparameters, which are model parameters the user sets before training the model. These hyperparameters, including the loss function and optimizer, significantly impact the model's performance during training and can affect its final performance [23]. In this context, understanding the role of hyperparameters in NLP is crucial for achieving optimal results in various NLP tasks.

4.2.1 Loss Function

A loss function is a mathematical function that measures the difference between the predicted output of a model and the actual output. The goal of a deep learning model is to minimize this difference, also known as the "loss" so that the predicted output is as close to the actual output as possible. In natural language processing, many tasks aim to predict a given input text's correct label or category. For example, in topic tagging for Amazon reviews, the task is to predict which topics the review is about, such as "quality", "design", "delivery service", etc.

We need a loss function to train a model for this task, which measures the difference between the predicted and actual labels. The loss function guides the model to learn the correct parameters during training by minimizing the difference between the predicted and actual labels. For Transformer models such as BERT and DistilBERT, a cross-entropy loss

is the most used loss function for topic tagging [24]. This loss function penalizes the model for making incorrect predictions by computing the negative log-likelihood of the actual label, given the predicted probability distribution over all possible labels. For CNN, LSTM, and RNNs models, the same cross-entropy loss function can also be used for topic tagging. However, different architectures may require different loss function modifications or additional regularization techniques to prevent overfitting.

4.2.2 Optimizer

An optimizer is an algorithm used during the training of machine learning models to minimize the loss function by updating the model parameters based on the loss gradients for those parameters. The goal of an optimizer is to find the set of model parameters that will result in the lowest possible value of the loss function.

In the deep learning models such as BERT, DistilBERT, CNN, LSTM, and RNN related to Amazon review topic tagging, the optimizer is typically chosen as Adam, which is an adaptive optimization algorithm that adjusts the learning rate of each weight during training based on the previous gradients for that weight. Adam (Adaptive Moment Estimation) is a popular optimizer used in deep learning that is well-suited for large datasets and complex models like Amazon Review [25]. Adam is widely used in natural language processing tasks due to its ability to handle sparse gradients and noisy data, which are common in text data.

During the training process, the optimizer iteratively updates the model parameters to minimize the loss function by computing the loss gradients for each parameter and updating the parameter values in the direction that decreases the loss. Adam optimizer utilizes exponentially decaying averages of past gradients and squared gradients to update the model weights.

4.2.3 Learning Rate

The learning rate is the size of the "steps" taken by the optimization algorithm during training toward the steepest descent of the loss function. If the learning rate is too high, the optimizer can overshoot the minimum of the loss function and prevent the model from converging [26]. Conversely, a learning rate that is too low can result in slow convergence or the optimizer getting stuck in local minima. In this thesis, we choose different learning rates to achieve optimal results in the deep learning model.

For the BERT and DistilBERT models used in this thesis, we have set the learning rate of 2 to the power of -5. This is a relatively low learning rate, which helps to prevent the models from overfitting the training data. However, it also means that the models will take longer to learn.

In contrast, for our CNN, LSTM, and RNN models, we have chosen a learning rate 5e-4, 5e-4 = 5 x 10 ^ (-4) = 5 / (10^4), which is equal to 0.0005. It is relatively low and suitable for models with a large number of parameters or for datasets that have high levels of noise.

4.2.4 Dropout

Neural network models often employ regularization techniques to prevent overfitting, and one popular method is dropout [27]. This approach involves randomly disabling a percentage of neurons during training to encourage the network to learn more robust features that are not dependent on individual neurons. Dropout has proven to be effective in mitigating overfitting in neural network models.

It is worth noting that the dropout rate determines the proportion of neurons that are randomly dropped out during training. A higher dropout rate can lead to more substantial regularization effects, which can help to avoid overfitting. However, a higher

dropout rate can also increase the difficulty of training the network and reduce its accuracy on the training data. Experts typically suggest starting with a dropout rate of 0.3 and then experimenting with various values to determine the most effective dropout rate for the trained network.

This thesis used dropout in a range of models, including BERT, and DistilBERT, CNN, LSTM, and RNN. The CNN, LSTM, and RNN models were trained using a dropout rate of 0.5, while the BERT and DistilBERT models utilized a dropout rate of 0.3.

4.3 Embedded Layer

In NLP, a neural network uses an embedding layer to convert text data into a numerical format it can process. The network learns dense embeddings and vector representations of text with a fixed length and are continuous-valued. These embeddings can capture complex relationships between words and be used for various NLP tasks, such as sentiment analysis and named entity recognition [28].

4.3.1 Input-Output Embedded Layer

The embedding layer is an essential component of many deep learning models, including CNN, LSTM, and RNN, and its primary function is to convert word tokens into dense vector representations. The input to the embedding layer is typically a sequence of integer-encoded word tokens mapped to high-dimensional vectors. In Chapter 3.3.1.2, if we consider a reviewText1 like "The gloves are very poor quality" and tokenize each word into an integer, we could generate the input token sequence [2, 3, 4, 5, 6, 7, 8]. These tokens would then be passed as input to the embedding layer.

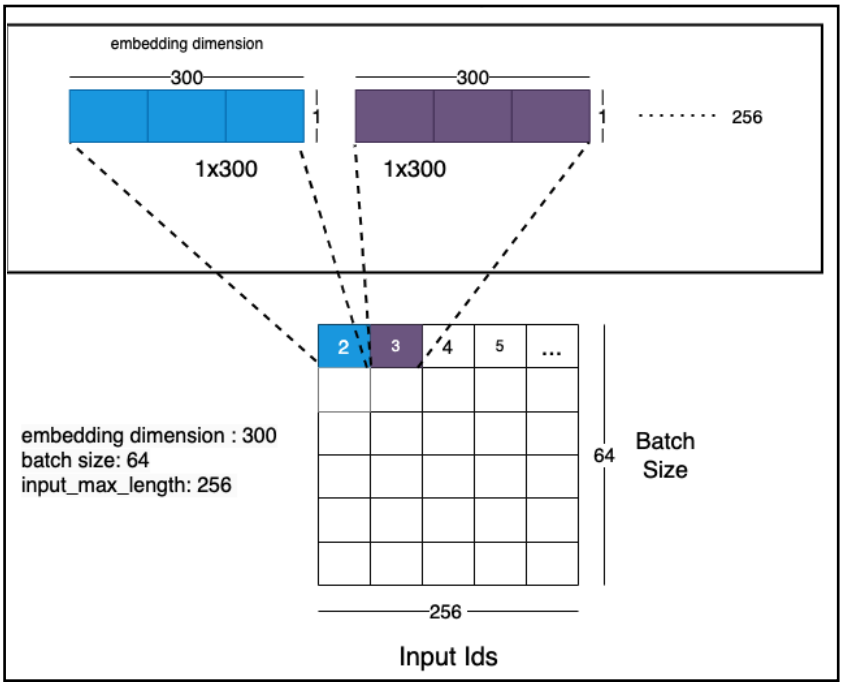

Figure 4.5: Embedded Layer

The output of the embedding layer is a sequence of dense vector representations, with each vector corresponding to a specific word in the input sequence. Each vector has a fixed length, and the dimensionality of the vectors is typically a hyperparameter that can be tuned during model training. The embedding layer aims to learn a set of vector representations that capture the semantic relationships between words in the input sequence. In Figure 4.5, the embedding layer is configured with a batch size of 64 and a maximum input length of 256 [29]. Each input consists of a 1x300 vector, where the dimensions represent related words. For instance, the word "gloves" is associated with 300 related words, including hand, leather, finger, mittens, winter, sports, fashion, latex, motorcycle, and work. These words are assigned a vector representation at position 2 with a shape of 1x300.

4.3.2 fastText

fastText is an open-source library developed by Facebook AI Research, designed to process and classify text data efficiently, particularly in natural language processing (NLP) [30]. It uses a neural network-based approach to learn the embeddings of words or short phrases, called n-grams, in a continuous vector space. The embeddings represent words semantic and syntactic meanings and can be used as input features for various downstream NLP tasks, such as topic tagging. fastText can handle out-of-vocabulary words by breaking them down into smaller sub word units or character n-grams, which are then represented with their embeddings, enabling the model to generalize better to unseen words and improve overall classification accuracy.

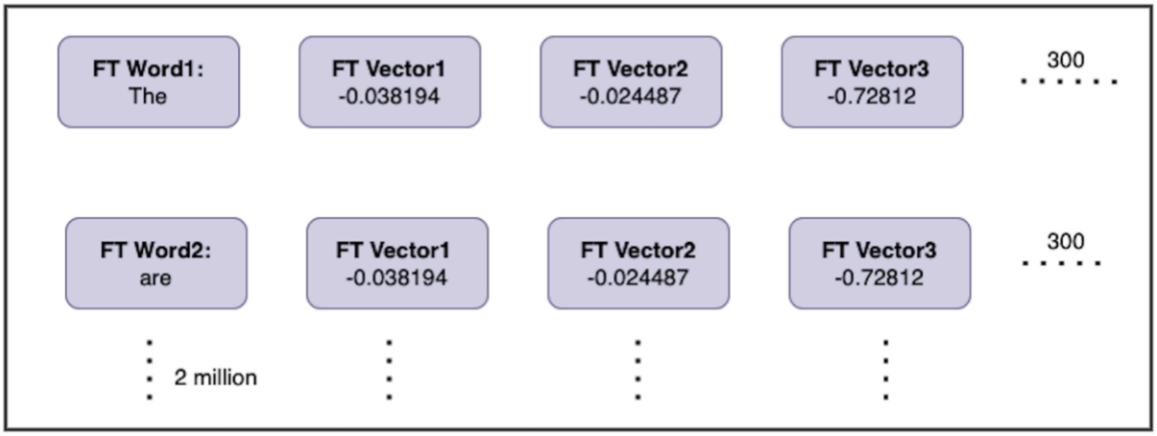

Figure 4.6: fastText (FT)

The fastText model is a pre-trained word embedding model that learns embeddings of words or n-grams in a continuous vector space. It is trained on a massive dataset of text, Common Crawl, consisting of over 600 billion tokens from various sources, including web pages, news articles, and social media posts [31]. The model outputs 2 million word vectors, each with a dimensionality of 300, because of this pre-training process. These pre-trained word vectors can be used as an embedding layer in neural networks for various

NLP tasks, such as topic tagging. They are a great starting point for training deep learning models on other tasks, as they allow for improved performance with less training data and time. Figure 4.6 illustrates the output of the fastText model, which consists of 2 million word vectors with a dimensionality of 300, called fastText embedding. The word is represented by FTWord1, and its corresponding vector is represented by FT vector1, FT vector2, FT vector3, ... FT vector300. The original website represented " FastText " as "fastText".

The vector comprises 300 dimensions, each representing a unique aspect of a word's meaning. The first dimension may indicate the word's part of speech, the second its semantic representation, and the third its sentiment. The values assigned to each dimension are real numbers, representing the degree of the word's association with that particular aspect of meaning. For instance, the value in the first dimension might be -0.038194, indicating that "fastText" is slightly more likely to be a noun than a verb based on the vector's analysis.

4.3.3 fastText Embeddings Layer

Figure 4.7 provides an overview of the three primary components of the system: Vocab, fastText, and embedding. The vocabulary was created using a text tokenizer, resulting in a size of 4773 for the training dataset, as explained in Chapter 3.3.1.1. Additionally, Chapter 4.3.2 presents fastText as a 2 million by 300 word vector.

To initialize the embedding matrix, a random matrix of size (4777, 300) is created, matching the vocabulary size and fastText dimensionality. The matrix is initialized with all zeros. Next, the fastText embedding is read line by line, extracting each word from FT Word and FT Vector representation. The function checks if the word is present in Vocab and updates its FT vector in the initialized embedding matrix. The resulting embedding

matrix has each row representing the embedding vector for the corresponding word in fastText, producing an output like Chapter 4.3.3

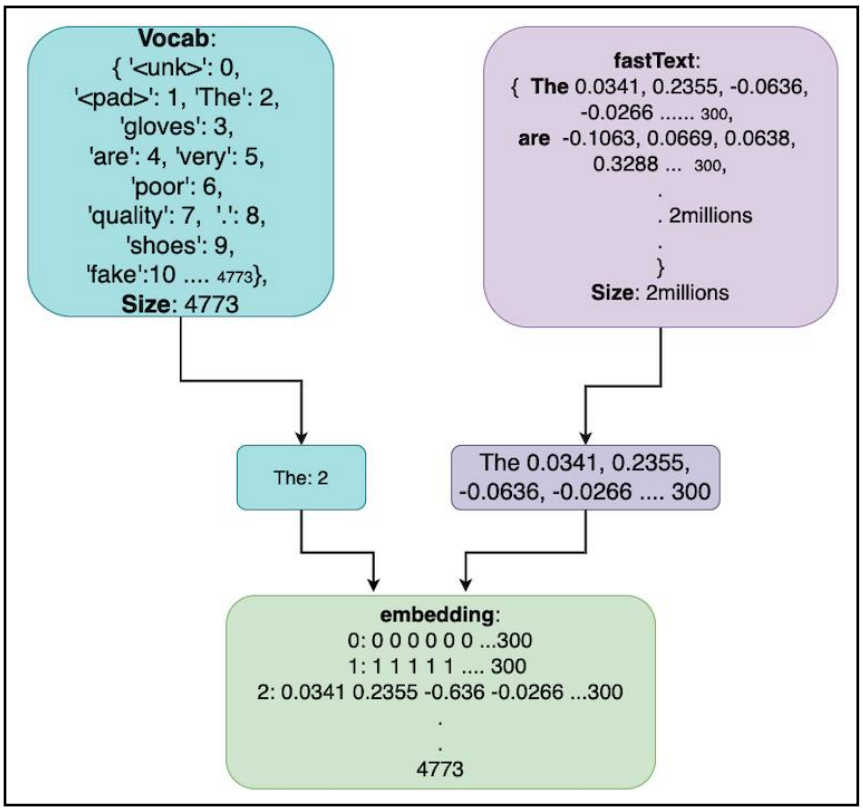

Figure 4.7: Vocab and Fast Text to Embedding

4.5 Fully Connected Layer

Topic tagging is a popular natural language processing task involving assigning one or more topic labels to a text document. Neural networks such as LSTM, RNN, and CNN have been used for topic tagging successfully [32,33]. In this context, a fully connected layer is typically added to the end of the neural network architecture to perform the final classification task. Chapter 4.6 will discuss how a deep learning model can use topic tagging to predict two topics. The size of the fully connected layer generating the model's predicted layers may vary depending on the number of layers in the model.

4.6 Introduction to Deep Learning Models

The thesis proposes developing ReviewTag system, which assists sellers in effectively managing customer reviews. The system leverages various Natural Language Processing (NLP) models, including BERT, Distiller, CNN, LSTM, and RNN [34]. ReviewTag enables sellers to efficiently manage customer reviews and enhance their products' ranking on Amazon.

4.6.1 BERT

Google developed BERT (Bidirectional Encoder Representations from Transformers) in 2018 as a powerful natural language processing model. It is a deep neural network that uses self-supervised learning to pre-train on a large corpus of text data, allowing it to learn contextual relations between words in a text [35, 36]. The transformer architecture is the foundation of BERT, which is designed to handle sequential data, such as text. It has multiple layers of self-attention mechanisms that enable it to capture long-range dependencies between words in a text. BERT is a bidirectional model that can consider both left and right context when making predictions. This helps BERT understand the context and meaning of words in a sentence. BERT can be fine-tuned on specific downstream NLP tasks, such as sentiment analysis or named entity recognition, by training on a smaller, labeled dataset. Fine-tuning allows BERT to achieve state-of-the-art performance on a wide range of NLP tasks, even when training data is limited.

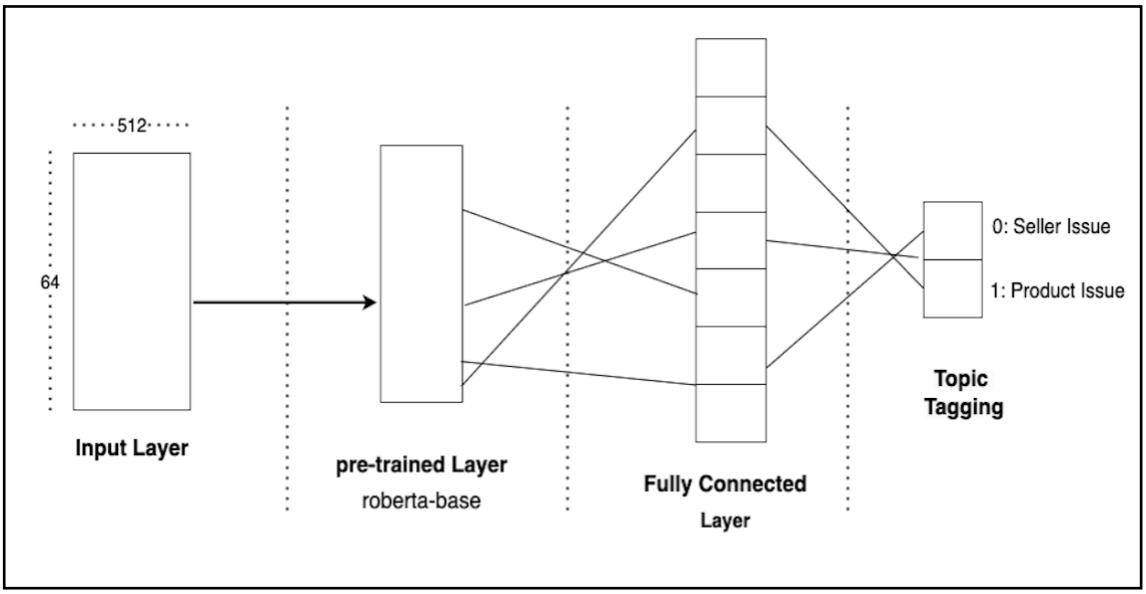

Figure 4.8: BERT

In Figure 4.8 BERT model is pre-trained on a large corpus of Amazon Negative Product Review data and can be fine-tuned for various NLP tasks [37]. The BERT Model consists of several layers:

- **Input Layer:** The BERT Model takes inputs: ids, and mask. These inputs are encoded representations of the input text obtained using a tokenizer from the Chapter 3.3.2.
- **pre-trained Layer:** The pre-trained layer refers to a neural network pre-trained on large amounts of text data. When input text is fed into the Roberta model, the pre-trained layer processes it and produces a sequence of hidden states for each input token. The RoBERTa model has a deep architecture comprising multiple self-attention layers and feed-forward neural networks.
- **Fully connected Layer:** The fully connected layer is a linear layer that takes the output of the **pre-trained Layer** and maps it to the desired output dimensionality.

During the forward pass, the input ids, and mask are passed through the pre-trained layer, generating a hidden state sequence. Finally, the output of the dropout layer is passed through the linear output layer to produce the final output of the model, which is a vector of size two representing the probabilities of the two classes in the topic tagging task.

4.6.2 DistilBERT

Hugging Face introduced DistilBERT in 2019 as a smaller and faster alternative to BERT, a powerful natural language processing model developed by Google. DistilBERT uses a distillation technique to train a smaller model to replicate BERT's behavior, resulting in a model with fewer parameters while maintaining high accuracy and performance [38, 39, 40]. DistilBERT still uses the transformer architecture and is pre-trained using self-supervised learning on a large corpus of text data, making it a bidirectional model that considers both the left and right context of each word in a sentence when making predictions.

DistilBERT performs well on various NLP tasks such as text classification, question answering, and named entity recognition. Many researchers and practitioners in the NLP community have adopted DistilBERT due to its smaller size and faster training and inference times, without sacrificing much accuracy or performance.

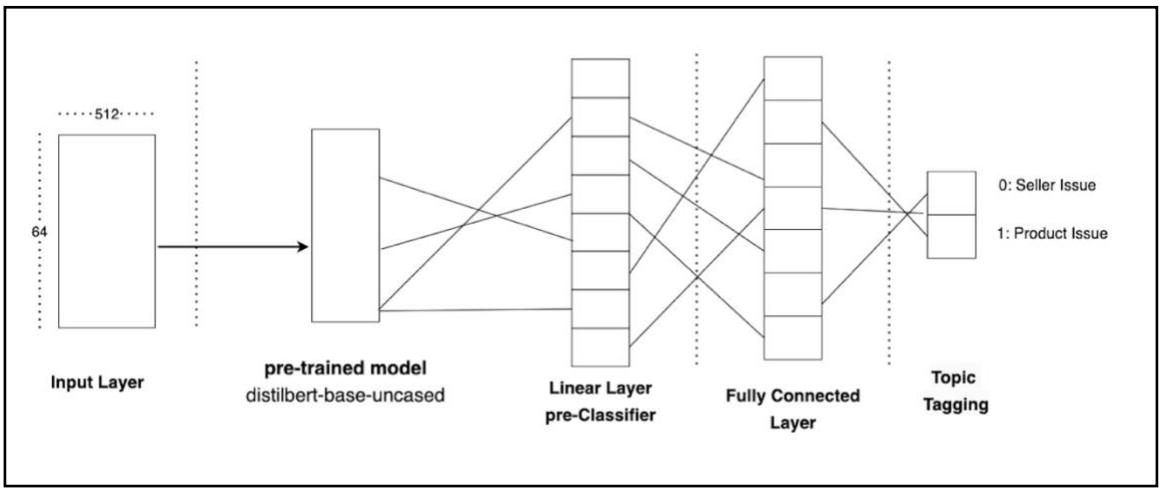

Figure 4.9: DistilBERT

Figure 4.9 DistilBERT model is pre-trained on a large corpus of Amazon Negative Product Review data and can be fine-tuned for various NLP tasks. DistilBERT Model consists of several layers:

- **Input Layer:** The DistilBERT Model takes inputs: ids, and mask. These inputs are encoded representations of the input text obtained using a tokenizer from the Chapter 3.3.2.
- **pre-trained Layer:** The pre-trained "distilbert-base-uncased" weights are used to initialize the DistilBERT model. It processes the input tensors to obtain the hidden state output. The first token of the sequence is used to obtain a pooled representation of the input sequence from the hidden_state tensor.
- **Linear Layer pre-classifier:** Linear layer pre-classifier is responsible for extracting features from data by taking the output of the pre-trained layer and passing it to a fully connected layer.

- **Fully connected Layer:** The fully connected layer is a linear layer that takes the output of linear layer pre-classifier and maps it to the desired output dimensionality. In this case, the output dimensionality is two, corresponding to topic tagging.

During the forward pass, the input ids, and mask are passed through the pre-trained layer, generating a hidden state sequence. Finally, the output of the dropout layer is passed through the linear output layer to produce the final output of the model, which is a vector of size two representing the probabilities of the two classes in the topic tagging task.

4.6.3 CNN

CNN for Sentence Classification is a NLP technique that uses convolutional neural networks to classify sentences into different categories or labels. The technique involves training a CNN on a large dataset of sentences, where each sentence is assigned a label or category [41, 42].

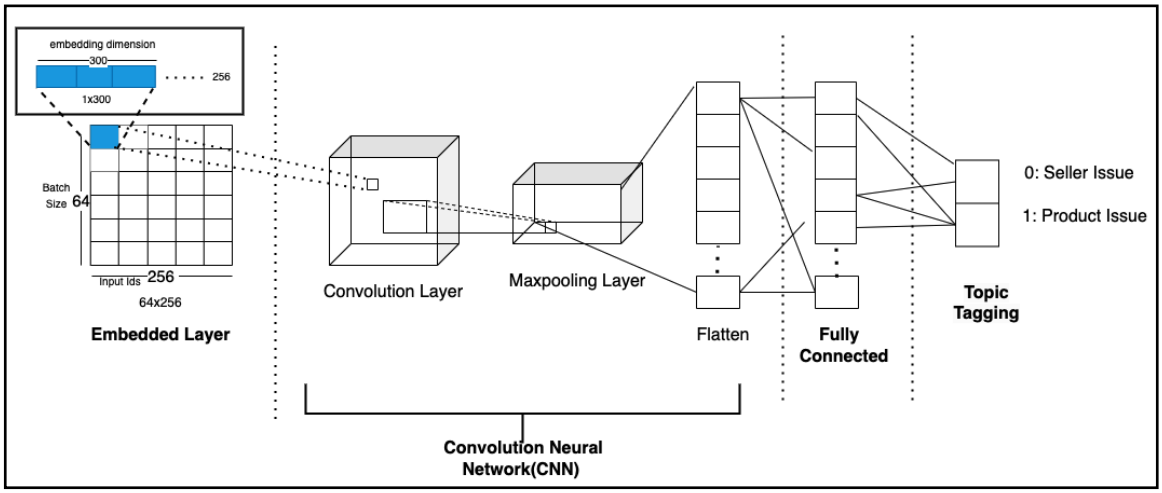

Figure 4.10: CNN

Figure 4.10 shows the architecture of the CNN model [43], which is composed of multiple layers:

- **Input layer:** The input layer receives the input data through token IDs and their lengths. The token IDs represent the index of the words in the vocabulary, and their lengths indicate how many tokens are in each sequence in Chapter 3.3.1.
- **Embedding layer:** The embedding layer converts the token IDs into dense vectors of fixed size (embedding dimension) that capture the semantic meaning of the words in the input sequence of each sequence (batch_size, max_len, embedding_dim) of (64, 256,300). The embeddings are learned during training in Chapter 4.3.3. The input sequence is reshaped to have dimensions (batch_size, embedding_dim, max_len) of (64, 300, 256) and then fed into the convolutional layers.
- **Convolutional Layers:** The next layer consists of multiple parallel convolutional layers with different filter sizes (filter_sizes) and the number of filters (num_filters). Each convolutional layer applies a set of filters with the same kernel size to the input embeddings, producing a feature map for each filter. The output shape of each feature map is (batch_size, num_filters[i], L_out), where L_out is the output length after applying the convolution operation.
- **Max Pooling:** Max pooling is applied to each feature map over the time dimension (L_out) to obtain a fixed-length representation of the most important features. This operation reduces the feature maps' dimensionality and helps extract the most relevant features in the data. The output shape of each pooled feature map is (batch_size, num_filters[i], 1).
- **Flatten Layer:** A flatten layer (nn.Flatten) is added to convert the 3D tensor output of the pooling layers into a 2D tensor that can be fed into the fully connected layer.

- **Fully Connected Layer:** The fully connected layer is a linear layer that takes the output of the pre-trained layer and maps it to the desired output dimensionality. In this case, the output dimensionality is two, corresponding to topic tagging.

4.6.4 LSTM

LSTM is a type of RNN type that overcomes the vanishing gradient problem in traditional RNNs [38, 39]. The vanishing gradient problem arises when the gradients used to update the network's weights become very small during backpropagation, making it difficult for the network to learn long-term dependencies. A set of gates that control the flow of information through the network is introduced by LSTMs to solve this issue. These gates, including an input gate, an output gate, and a forget gate, enable the network to selectively update, output, or forget information at each time step.

At each time step, the input gate determines which information from the current input should be used to update the memory state. Meanwhile, the forget gate decides which information from the previous memory state should be forgotten. The output gate then determines which information from the updated memory state should be outputted to the next layer or as the final prediction. The cell state is the memory state in an LSTM, which can retain information for extended periods. The input and forget gates enable the network to selectively update or forget information in the cell state. LSTMs have been shown to be effective at modeling sequential data and have achieved state-of-the-art results on many tasks in natural language processing, speech recognition, machine translation, and image captioning.

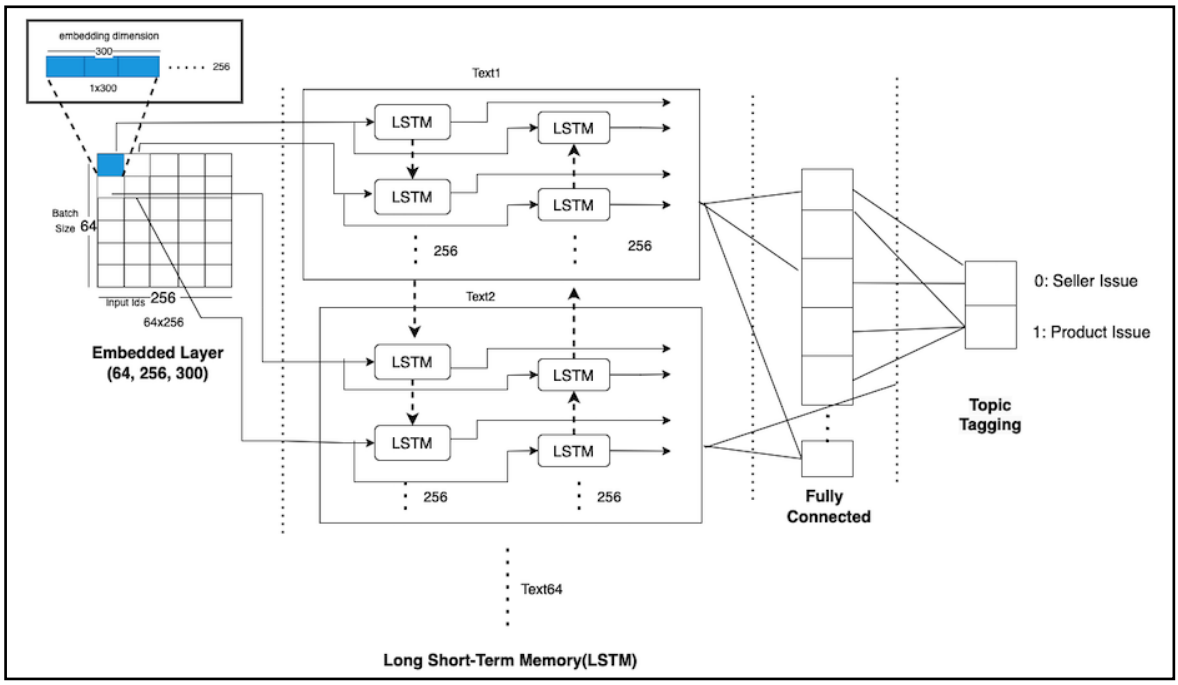

Figure 4.11: LSTM

Figure 4.11 illustrates the architecture of the LSTM model [40], which is composed of multiple layers:

- **Input layer:** The input layer receives the input data through token IDs and their lengths. The token IDs represent the index of the words in the vocabulary, and their lengths indicate how many tokens are in each sequence in Chapter 3.3.1.
- **Embedding layer:** The embedding layer converts the token IDs into dense vectors of fixed size (embedding dimension) that capture the semantic meaning of the words in the input sequence. The embeddings are learned during training in Chapter 4.3.3.
- **LSTM layer:** The LSTM layer processes the embedded input sequences using the LSTM algorithm. The LSTM layer has a certain number of hidden units (hidden_dim) and can be stacked in multiple layers (n_layers) to increase the model's capacity.

- **Fully Connected Layer:** The fully connected layer is a linear layer that takes the output of the pre-trained layer and maps it to the desired output dimensionality. In this case, the output dimensionality is two, corresponding to topic tagging.

4.6.5 RNN

RNN is a type of neural network designed for handling sequential data, such as time series or natural language. RNNs can process sequences of inputs by maintaining an internal state that depends on the previous inputs, and they can share weights across different time steps. This allows them to learn and model temporal dependencies in the data using recurrent connections [47, 48]. The LSTM network is the most common type of RNN that addresses the problem of vanishing gradients in traditional RNNs.

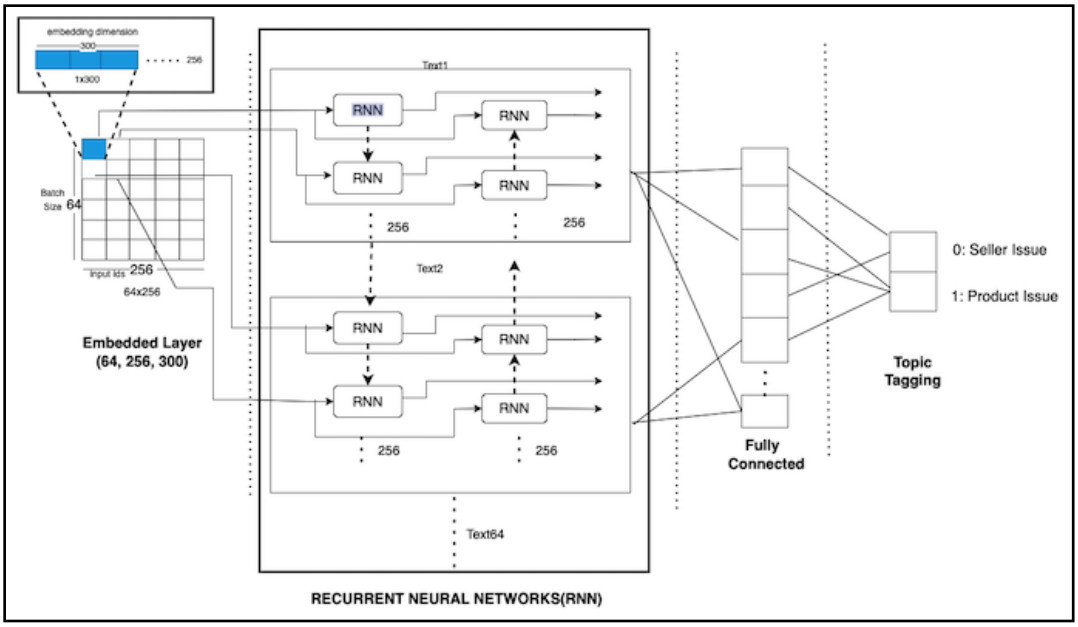

Figure 4.12: RNN

Figure 4.12 illustrates the architecture of the RNN model [37], which is composed of multiple layers:

- **Input layer:** The input layer receives the input data through token IDs and their lengths. The token IDs represent the index of the words in the vocabulary, and their lengths indicate how many tokens are in each sequence Chapter 3.3.1.
- **Embedding layer:** The embedding layer converts the token IDs into dense vectors of fixed size (embedding dimension) that capture the semantic meaning of the words in the input sequence. The embeddings are learned during training in Chapter 4.3.3.
- **RNN layer:** RNN layer takes the embedded sequences as input and produces a sequence of hidden states. The RNN layer used in this implementation is an instance of the nn.RNN class. The layer can be configured to have multiple layers and be bidirectional.
- **Fully Connected Layer:** The fully connected layer is a linear layer that takes the output of the pre-trained layer and maps it to the desired output dimensionality. In this case, the output dimensionality is two, corresponding to topic tagging.

CHAPTER V:

PERFORMANCE EVALUATION

The thesis proposes developing ReviewTag, which can assist sellers in efficiently managing customer reviews by tagging negative product reviews on Amazon. Chapter 3 explains the process of creating the Amazon Review Tagged Dataset through manual tagging and further pre-processing the data using tokenizer in the Amazon Review Data Transformation. Chapter 4 introduces the concepts of NLP, such as hyperparameters, model architecture, and deep learning models, along with an overview of the Automated Tagging System. This chapter evaluates the performance of the automated tagging system using advanced deep learning for topic tagging (TT) system, and three different subtopics tagging (ST): STWOT, STWKT, and STWPT.

Using Amazon Review Tagged Dataset contains 12,053 entries with 7,932 reviews tagged as Product Issues and 4,121 reviews tagged as Seller Issues. The Product Issue category has three subtopics: Design Issues, Quality Issues, and Product Description Issues, while the Seller Issue category has two subtopics: Product Authenticity Issues and Delivery and Return Issues.

The training and test datasets are prepared for different systems, including TT, STWPT, STWOT, and STWKT, with an 80:20 ratio. For TT, STWPT, and STWOT, the dataset is divided into 9637 entries for training and 2,416 entries for testing. Meanwhile, the STWKT system uses two sets of data: Amazon's negative product reviews and negative seller reviews. The datasets are divided into training and test sets, with the first dataset containing 7,932 entries of Amazon negative product reviews, with 6,346 for training and 1,586 for testing. The second dataset comprises 4,121 entries of Amazon negative seller reviews, with 3,297 for training and 824 for testing.

After preparing the training dataset, we use advanced deep learning techniques such as BERT, DistilBERT, LSTM, RNN, and CNN models for training purposes. Following that, we will assess the performance of our trained model by utilizing the test dataset. The evaluation related to the test dataset is shown below.

5.1 TT Performance Evaluation

Topic Tagging (TT) performance evaluation is a crucial step in assessing the effectiveness of topic tagging techniques in categorizing reviews into specific categories. The evaluation process typically involves comparing the predicted categories with the actual categories assigned to the reviews by manual tagging. Precision, recall, and F1 score are metrics used to evaluate the performance of the topic classifier.

Table 5.1: TT Performance Evaluation

Model	Precision	Recall	F1- Score
BERT	0.97	0.96	0.96
DistilBERT	0.95	0.95	0.95
CNN	0.96	0.95	0.95
LSTM	0.96	0.95	0.95
RNN	0.87	0.80	0.83

Table 5.1 shows the topic tagging evaluation results for various models, including BERT, DistilBERT, CNN, LSTM, and RNN, based on their precision, recall, and F1 score.

BERT and DistilBERT are both transformer-based language models that have been pre-trained on a massive dataset of text and code. BERT is a larger model than DistilBERT, which means that it has more parameters and requires more training data and computation

time. In the experiment, BERT achieved the highest precision score of 0.97, along with a recall and F1 score of 0.96. DistilBERT is a smaller model that has been distilled from BERT to be more efficient while maintaining its accuracy. Despite its smaller size, DistilBERT still performed well in the experiment, achieving precision, recall, and an F1 score of 0.95.

CNN, LSTM, and RNN are types of deep networks commonly used for text classification. CNN sentence classifiers can learn local patterns in text, while LSTMs can learn long-range dependencies between words. In this experiment, LSTM and CNN models achieved a precision of 0.96, along with a recall and F1 score of 0.95 Compared to LSTMs and CNNs, RNN models may be less efficient due to their limited ability to capture long-range dependencies in text data. In this experiment, RNN models achieved precision, recall, and an F1 score of 0.87, 0.80, and 0.83, respectively.

Based on our experiment, BERT, achieved the highest precision score of 0.97 and a recall and F1 score of 0.96. DistilBERT, CNN, and LSTM achieved a similar F1 score of 0.95, with precision and recall scores ranging from 0.95 to 0.96. On the other hand, RNN achieved a precision of 0.87, a recall of 0.80, and an F1 score of 0.83.

5.2 ST Performance Evaluation

Our automated tagging system for Amazon negative product reviews employs deep learning models to identify and tag negative reviews based on their topics accurately. The system employs three different types of subtopics tagging: SubTopic Tagging without a Topic, SubTopic Tagging with Known Topics, and subtopic tagging with predicted topics.

Our system's comprehensive analysis of negative reviews, utilizing three different types of subtopics tagging, enables sellers to make informed decisions and improve their businesses by addressing specific issues highlighted in the reviews. For subtopic tagging,

we evaluated the precision, recall, and F1 scores of various models, including BERT, DistilBERT, CNN, LSTM, and RNN.

5.2.1 STWOT Performance Evaluation

SubTopic Tagging without Known Topics (STWOT) is an approach that utilizes predetermined subtopics within the broad categories of product and seller issues to classify reviews further. The method involves tagging reviews with known subtopics within each category, such as product design issues, quality issues, and description issues for product issues and product authentication issues, delivery and return issues for seller issues. This approach provides more detailed insights into specific areas that require improvement.

Table 5.2, BERT achieved the highest precision, recall, and F1 score of 0.90, while DistilBERT, despite its smaller size, performed well with a precision and F1 score of 0.85 and a recall of 0.86. The CNN models achieved a precision of 0.89, with a recall and F1 score of 0.88. The LSTM models obtained a precision, recall, and F1 score of 0.87. However, the RNN models had a precision score of 0.75 and a recall and F1 score of 0.73, indicating lower performance than the experiment's other models.

Table 5.2: STWOT Performance Evaluation

Model	Precision	Recall	F1- Score
BERT	0.90	0.90	0.90
DistilBERT	0.85	0.86	0.85
CNN	0.89	0.88	0.88
LSTM	0.87	0.87	0.87
RNN	0.75	0.73	0.73

While STWOT allows for identifying emerging subtopics without pre-defined labels, it may lower precision, recall, and F1 scores. In contrast, STWKT, with predetermined topic tagging help to improve precision, recall, and F1 scores for subtopic tagging.

5.2.2 STWKT Performance Evaluation

SubTopic Tagging with Known Topic (STWKT) method is a powerful approach to classify reviews further using predetermined subtopics within the broad categories of product and seller issues. This method involves tagging reviews with the known subtopics within each category, such as design issues, quality issues, and product description issues for product issues and product authentication issues, delivery and return issues for seller issues.

Table 5.3, BERT achieved the highest precision, recall, and F1 score of 0.91, while DistilBERT, despite its smaller size, performed well with precision, recall, and F1 score of 0.89. The CNN models achieved a precision of 0.91, with a recall and F1 score of 0.90. The LSTM models obtained a precision, recall, and F1 score of 0.90. However, the RNN models had a precision, recall, and F1 score of 0.86, indicating lower performance than the experiment's other models.

Table 5.3: STWKT Performance Evaluation

Model	Precision	Recall	F1- Score
BERT	0.91	0.91	0.91
DistilBERT	0.89	0.89	0.89
CNN	0.91	0.90	0.90
LSTM	0.90	0.90	0.90

RNN	0.86	0.86	0.86

5.2.3 STWPT Performance Evaluation

SubTopic Tagging with Predicted Topics (STWPT) technique can be an effective tool for Amazon sellers who lack labeled datasets for negative reviews of their products. This technique predicts the negative review's topic and subtopic based on the predicted topic and the negative review itself.

Table 5.4, BERT achieved the highest precision score and F1 score of 0.92 and a recall of 0.91, while DistilBERT, despite its smaller size, performed well with a precision score and F1 score of 0.90 and a recall of 0.89. The CNN models achieved a precision of 0.89, with a recall and F1 score of 0.92. The LSTM models obtained a precision, recall, and F1 score of 0.91. However, the RNN models had a precision score of 0.85 and a recall and F1 score of 0.84, indicating lower performance than the experiment's other models.

Table 5.4: STWPT Performance Evaluation

Model	Precision	Recall	F1- Score
BERT	0.92	0.91	0.92
DistilBERT	0.90	0.89	0.90
CNN	0.92	0.92	0.92
LSTM	0.91	0.91	0.91
RNN	0.85	0.84	0.84

5.2.4 ST Performance Evaluation

Our automated tagging system for negative Amazon product reviews utilizes deep learning models. It employs three different type of subtopic tagging to identify and tag negative reviews based on their topics accurately. The performance evaluation of our subtopic tagging system is presented in Figure 5.1, 5.2, and 5.3. Precision, recall, and F1-score metrics are used to evaluate the performance of subtopic tagging without topic, subtopic tagging with known topic, and subtopic tagging with the predicted topic for BERT, DistilBERT, CNN, LSTM, and RNN models.

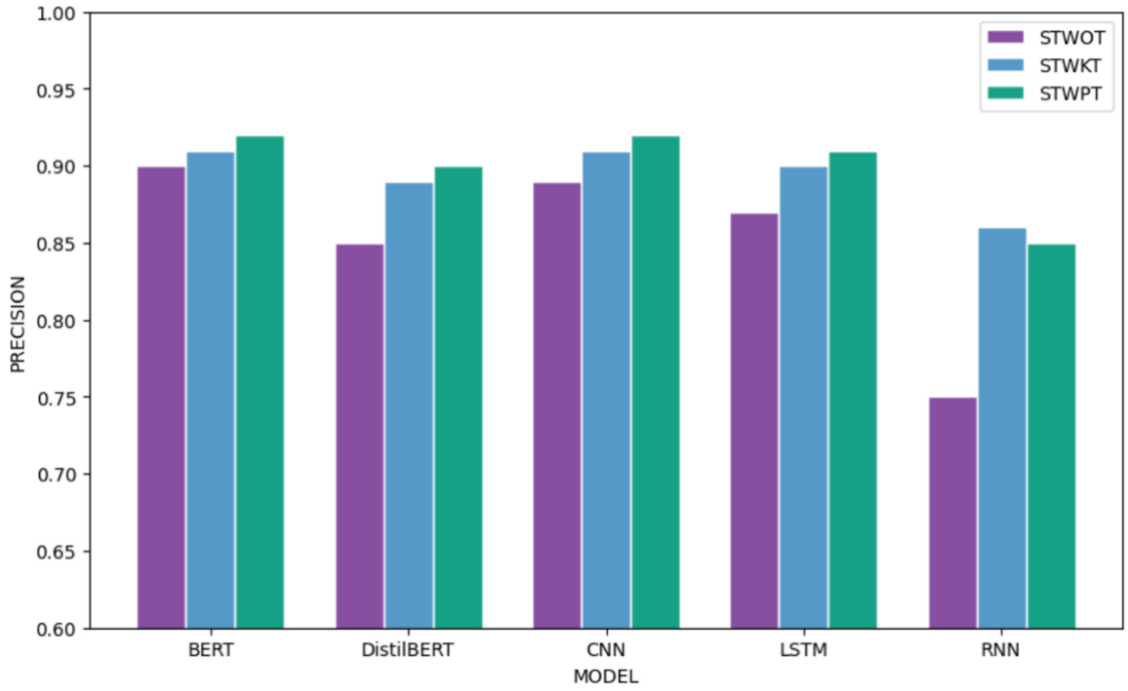

Figure 5.1: Precision of SubTopic Tagging

Figure 5.1 shows the precision of subtopic tagging for each model. BERT achieves the highest precision score for all subtopic tagging levels, ranging from 0.90 to 0.92.

DistilBERT and CNN also achieve high precision scores, ranging from 0.85 to 0.92. LSTM and RNN models achieve lower precision scores, with RNN achieving the lowest.

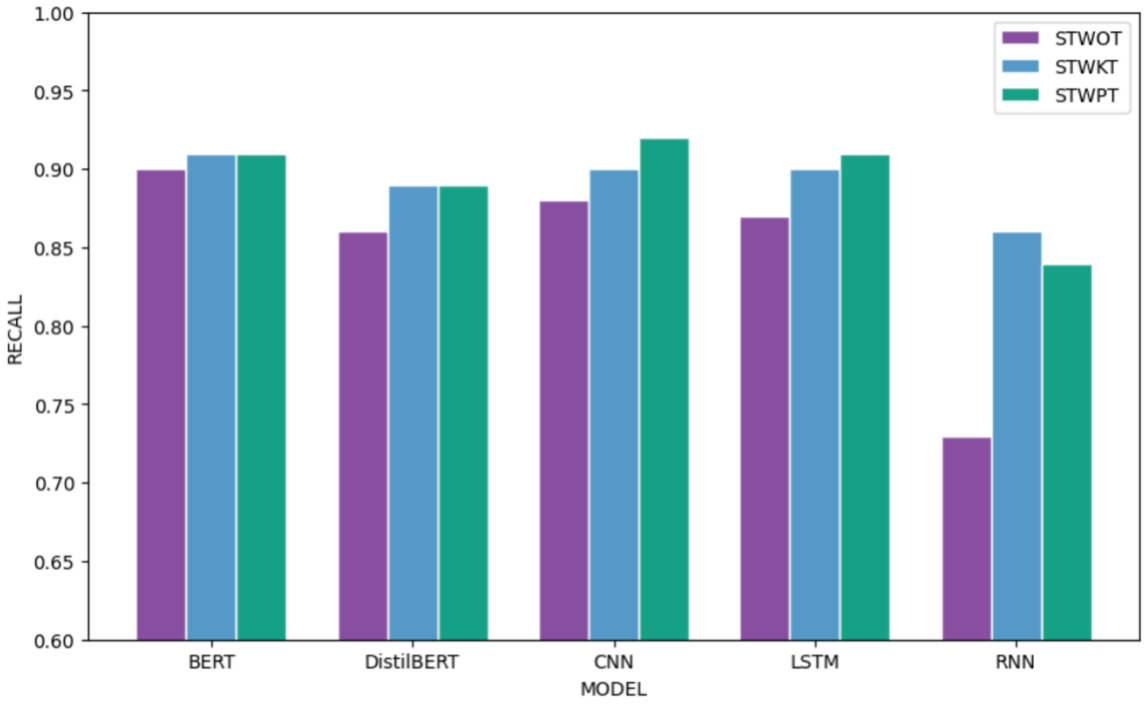

Figure 5.2: Recall of SubTopic Tagging

Figure 5.2 shows the recall of subtopic tagging for each model. BERT achieves the highest recall score for all subtopic tagging levels, ranging from 0.90 to 0.91. DistilBERT and CNN also achieve high recall scores, ranging from 0.86 to 0.92. LSTM and RNN models achieve lower recall scores, with RNN achieving the lowest.

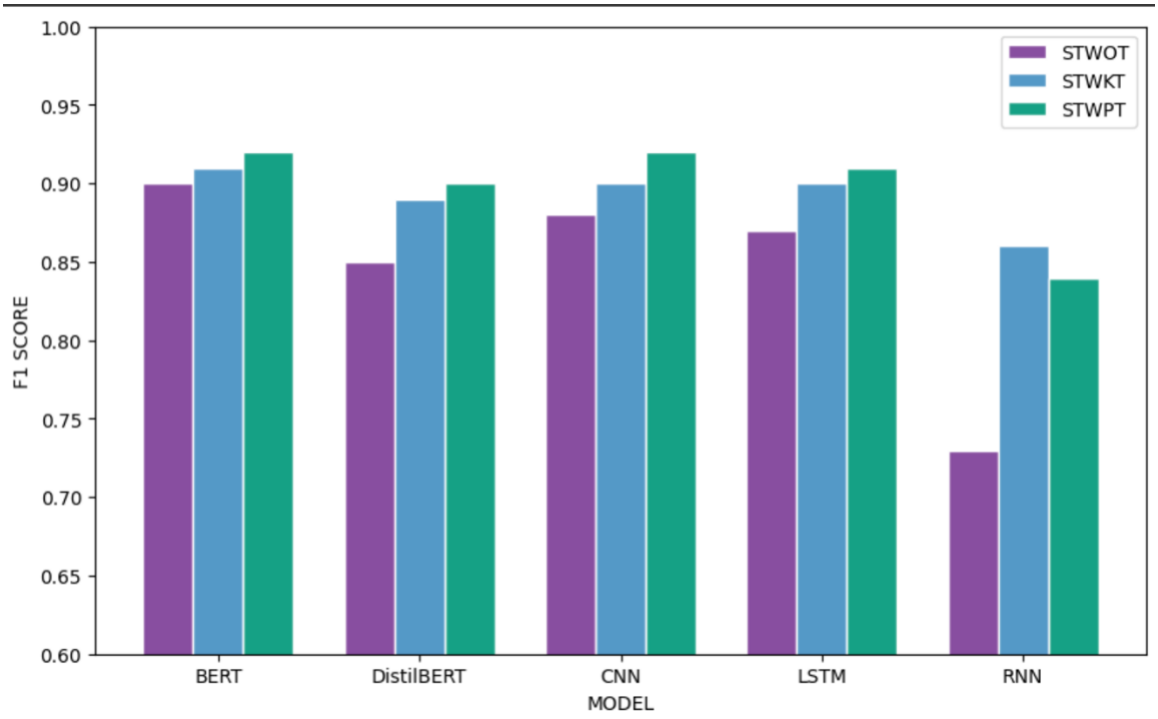

Figure 5.3: F1 Score of SubTopic Tagging

Figure 5.3 shows the F1 score of subtopics tagging for each model. BERT achieves the highest F1 score for all subtopic tagging levels, ranging from 0.90 to 0.92. DistilBERT and CNN also achieve high F1 scores, ranging from 0.85 to 0.92. LSTM and RNN models achieve lower F1 scores, with RNN achieving the lowest.

Based on our experiment, BERT achieved the highest precision, recall, and F1 scores across all subtopic tagging levels, ranging from 0.90 to 0.92. DistilBERT and CNN also achieved high scores, ranging from 0.85 to 0.92. On the other hand, LSTM and RNN models obtained lower scores, with RNN achieving the lowest precision, recall, and F1 scores. The results suggest that transformer-based models such as BERT and DistilBERT outperform traditional neural networks such as LSTM and RNN in subtopic tagging tasks.

CHAPTER VI:

CONCLUSION AND FUTURE WORK

Amazon sellers face a significant hurdle in maintaining high ratings for success when they receive negative customer reviews. To help sellers quickly identify the root cause of negative reviews, ReviewTag using deep learning models such as BERT, DistilBERT, LSTM, RNN, and CNN to categorize negative reviews into broader topics, such as product issues and seller issues. ReviewTag provides subtopic tagging, which offers more insight into customers' specific concerns.

We categorized negative Amazon product reviews using topic and subtopic tagging to evaluate the system's effectiveness. Our findings showed that BERT, a transformer-based architecture and pre-trained on vast amounts of text data, is an exceptional deep learning model for automated tagging in terms of topic tagging. BERT achieved a precision, recall, and F1-score of 0.97, 0.96, and 0.96, respectively. Moreover, when predicting the subtopic, BERT also displayed high precision, recall, and F1-score of 0.92, 0.91, and 0.92, respectively. On the other hand, the CNN sentence classifier model, which extracts features from sentences, such as word usage, order, and relationships, showed similar precision, recall, and F1-score of 0.92, 0.92, and 0.92.

One limitation of ReviewTag is the availability of limited datasets for each class, which can affect the reliability of the model's tagging for specific topics and subtopics. Additionally, the manual labeling of reviews for training the model can take time, hindering its scalability in handling a large volume of reviews.

While the proposed Tagging Amazon Product Negative Review with Deep Learning System offers significant value to Amazon sellers, there is potential for further development and expansion. The system could incorporate sentiment analysis to provide an overall sentiment score for customer reviews, enabling sellers to quickly identify

positive and negative feedback. The system could also integrate with Amazon's product ranking algorithm to provide insights into how customer reviews impact product ranking and sales. Additionally, the system could expand to analyze positive reviews, providing sellers with a comprehensive understanding of customer feedback.

REFERENCES

[1] "About Amazon - Our Company." Amazon. Accessed September 9, 2021- https:/www.aboutamazon.com

[2] Rosário, A., & Raimundo, R. (2021). Consumer marketing strategy and E-commerce in the last decade: a literature review. Journal of theoretical and applied electronic commerce research, 16(7), 3003-3024.

[3] Amazon's Fulfillment by Amazon (FBA): Fulfillment services for your ecommerce business

[4] Wan, M., & McAuley, J. (2016, December). Modeling ambiguity, subjectivity, and diverging viewpoints in opinion question answering systems. In 2016 IEEE 16th international conference on data mining (ICDM) (pp. 489-498). IEEE.

[5] B. V. Barde and A. M. Bainwad, "An overview of topic modeling methods and tools," 2017 International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, 2017, pp. 745-750, doi: 10.1109/ICCONS.2017.8250563.

[6] Rubayyi Alghamdi and Khalid Alfalqi, "A Survey of Topic Modeling in Text Mining", International Journal of Advanced Computer Science and Applications, vol. 6, no. 1, 2015.

[7] Jorg Becker and Dominik Kuropka, "Topic-based Vector Space Model", Business Information Systems Proceedings of BIS 2003.

[8] Deerwester Scott, Susan T. Dumis, George W. Furnas, Thomas K. Landaur and Richard Harshman, "Indexing by Latent Semantic Analysis", Journal of the American Society for Information Science 1986–1998.

[9] N. Bassiou and C. Kotropoulos, "RPLSA: A novel updating scheme for Probabilistic Latent Semantic Analysis", Department of Informatics Aristotle University of Thessaloniki Box 451 Thessaloniki 541 24 Greece, April 2010.

[10] Tsai, F. S. (2011). A tag-topic model for blog mining. Expert Systems with Applications, 38(5), 5330-5335.

[11] Wen, Q., Qiang, M., Xia, B., & An, N. (2019). Discovering regulatory concerns on bridge management: An author-topic model based approach. Transport Policy, 75, 161-170.

[12] Yoon, Y. S., Lee, J., & Park, K. (2019, October). Extracting Promising Topics on Smart Manufacturing Based on Latent Dirichlet Allocation (LDA). In 2019 International Conference on Information and Communication Technology Convergence (ICTC) (pp. 1237-1242). IEEE.

[13] Davoudi, A., Tissot, H., Doucette, A., Gabriel, P. E., Parikh, R., Mowery, D. L., & Miranda, S. P. (2022). Using natural language processing to classify serious illness communication with oncology patients. In AMIA Annual Symposium Proceedings (Vol. 2022, p. 168). American Medical Informatics Association.

[14] Huda, A. F., & Baizal, Z. A. (2021, October). Feature Extraction Amazon Customer Review to Determine Topic on Smartphone Domain. In 2021 13th International Conference on Information & Communication Technology and System (ICTS) (pp. 342-347). IEEE.

[15] Yuhan Zhang and Haiping Xu. SLTM: A Sentence Level Topic Model for Analysis of Online Reviews. (Seke):449-453, 2016.

[16] J. McAuley, "Amazon product data," University of California San Diego (UCSD).

[17] McAuley, J., Shi, Q., Targett, C., & van den Hengel, A. (2015). Image-based recommendations on styles and substitutes. In Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 43-52).

[18] He, R., & McAuley, J. (2016). Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In Proceedings of the 25th International Conference on World Wide Web (pp. 507-517).

[19] Loper, E., & Bird, S. (2002). Nltk: The natural language toolkit. arXiv preprint cs/0205028

[20] NLTK 3.6.3 documentation. (2021). Tokenization. Retrieved from https://www.nltk.org/api/nltk.tokenize.html#module-nltk.tokenize.

[21] Transformer Tokenizer, Hugging Face, 2019: https://huggingface.co/tokenizers/

[22] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., ... & Brew, J. (2019). HuggingFace's transformers: State-of-the-art natural language processing. arXiv preprint arXiv:1910.03771.

[23] S. Kim, J. Do and M. Kim, "Pseudo-Supervised Learning for Semantic Multi-Style Transfer," in IEEE Access, vol. 9, pp. 7930-7942, 2021, doi: 10.1109/ACCESS.2021.3049637.

[24] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2818-2826).

[25] D. P. Kingma and J. L. Ba, "ADAM: A Method for Stochastic Optimization," arXiv preprint arXiv:1412.6980, 2014

[26] M. Zhang, Y. Sun, and R. Jin, "Optimizing Learning Rates for Deep Neural Networks," in Proceedings of the 2019 International Conference on Artificial Intelligence and Computer Engineering, 2019, pp. 111-115.

[27] Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." Journal of Machine Learning Research 15.1 (2014): 1929-1958.

[28] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. Proceedings of the International Conference on Learning Representations (ICLR).

[29] E. Tighe, O. Aran, and C. Cheng, "Exploring Neural Network Approaches in Automatic Personality Recognition of Filipino Twitter Users," De La Salle University Manila, 2023.

[30] Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. Transactions of the association for computational linguistics, 5, 135-146.

[31] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, and A. Joulin. Advances in Pre-Training Distributed Word Representations. Facebook AI Research, 2018.

[32] Dieng, A. B., Ruiz, F. J., & Blei, D. M. (2020). Topic modeling in embedding spaces. Transactions of the Association for Computational Linguistics, 8, 439-453.

[33] Ma, W., & Lu, J. (2017). An equivalence of fully connected layer and convolutional layer. arXiv preprint arXiv:1712.01252.

[34] Tang, Y., Chen, M., & He, X. (2019). Multi-label classification of customer reviews based on deep learning. IEEE Access, 7, 114454-114465. doi: 10.1109/access.2019.2939636.

[35] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.

[36] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

[37] "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding" by Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), pages 4171–4186, 2019.

[38] Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper, and lighter. arXiv preprint arXiv:1910.01108. Link: https://arxiv.org/abs/1910.01108

[39] Sun, Y., Wang, S., Li, Z., Feng, Y., Chen, X., & Zhang, H. (2021). DistilBERT for Question Answering. In Proceedings of the 2021 International Conference on Education Technology Management (ICETM 2021) (pp. 420-424). Atlantis Press. Link: https://www.atlantis-press.com/proceedings/icetm-21/125957688

[40] Tariq, U., Naseem, I., & Razzak, I. (2021). A Comparative Study of BERT and DistilBERT on Text Classification. Journal of Data Science and Applications, 4(2), 79-88. Link: http://www.jdsap.org/jdsap/article/view/123

[41] Kim, Y. (2014). Convolutional neural networks for sentence classification. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) (pp. 1746-1751).

[42] Chen, Y. (2015). Convolutional neural network for sentence classification (Master's thesis, University of Waterloo).

[43] Zhang, Y., & Wallace, B. (2016). A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. arXiv preprint arXiv:1510.03820.

[44] S. Hu, G. Zou, S. Lin, L. Wu, C. Zhou, B. Zhang, and Y. Chen, "Recurrent Transformer for Dynamic Graph Representation Learning with Edge Temporal States", Washington University in St. Louis, 2023

[45] [Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Ju ̈rgen Schmidhuber. Long short-term memory. Neural Computation, 9(8):1735–1780, 1997.

[46] Zhang, X., Zhao, J., & LeCun, Y. (2015). Character-level convolutional networks for text classification. In Advances in neural information processing systems (pp. 649-657).

[47] Dieng, A. B., Ruiz, F. J., & Blei, D. M. (2020). Topic modeling in embedding spaces. Transactions of the Association for Computational Linguistics, 8, 439-453.

[48] Liu, P., Qiu, X., & Huang, X. (2016). Recurrent neural network for text classification with multi-task learning. arXiv preprint arXiv:1605.05101.

[49] Zulqarnain, M., Ghazali, R., Hassim, Y. M. M., & Rehan, M. (2020). A comparative review on deep learning models for text classification. Indones. J. Electr. Eng. Comput. Sci, 19(1), 325-335.