Autonomous Decision Making on User Complaints with NLP based Deep Learning Models

By

Mariam Sabir

(Registration No: 00000320328)

Thesis Supervisor: Dr. Arslan Shaukat

DEPARTMENT OF COMPUTER & SOFTWARE ENGINEERING

COLLEGE OF ELECTRICAL & MECHANICAL ENGINEERING

NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY

ISLAMABAD

September 2023

Annex A

THESIS ACCEPTANCE CERTIFICATE

Certified that final copy of MS/MPhil thesis written by **NS Mariam Sabir** Registration No. **00000320328**, of College of E&ME has been vetted by undersigned, found complete in all respects as per NUST Statutes/Regulations, is free of plagiarism, errors and mistakes and is accepted as partial fulfillment for award of MS/MPhil degree. It is further certified that necessary amendments as pointed out by GEC members of the scholar have also been incorporated in the thesis.

Signature : _____

Name of Supervisor: **Dr Arslan Shaukat**

Date: 07-09-2023

Signature of HOD: _____
(Dr Usman Qamar)
Date: 07-09-2023

Signature of Dean: _____
(Brig Dr Nasir Rashid)
Date: 07 SEP 2023

Dedicated to my exceptional parents: ***Muhammad Sabir & Yasmin Sabir****, my siblings specially Sarmad and friends specially Qurat-ul-ain whose tremendous support and cooperation led me to this accomplishment*

Acknowledgements

All praise and glory to Almighty Allah (the most glorified, the highest) who gave me the courage, patience, knowledge and ability to carry out this work and to persevere and complete it satisfactorily. Undoubtedly, HE eased my way and without HIS blessings I can achieve nothing.

I would like to express my sincere gratitude to my advisor Dr. Arslan Shaukat for his continual assistance, motivation, dedication and invaluable guidance in my quest for knowledge. I am blessed to have such a co-operative advisor and kind mentor for my research.

Along with my advisor, I would like to acknowledge my entire thesis committee: Dr. Ali Hassan and Dr. Wasi Haider for their cooperation and prudent suggestions.

My acknowledgement would be incomplete without thanking the biggest source of my strength, my family. I am profusely thankful to my beloved parents who raised me when I was not capable of walking and continued to support me throughout in every department of my life.

Finally, I would like to express my gratitude to all my friends for always standing by my side and the individuals who have encouraged and supported me through this entire period.

Abstract

With the advancement in technological domain, the volume of text data, particularly user complaints in public sector, is increasing rapidly. This huge amount of data requires more accurate and automated system for text classification, as manual sorting of this data is time-consuming and error-prone. Addressing this challenge, our research proposes a deep learning-based text classification system that utilizes three models; Convolutional Neural Networks (CNN), a hybrid of Bidirectional Long Short-Term Memory (BiLSTM) and CNN, and the state-of-the-art Bidirectional Encoder Representations from Transformers (BERT) model. These models will interpret the context of text and classify complaints automatically into respective departments. After preprocessing, GloVe embeddings and BERT Tokenizer are used to extract crucial information as features from the complaints, and feed to the models for further pattern learning and classification. Proposed models achieved over 80% classification accuracy on the local complaint dataset, and more than 86% on the internationally acclaimed Consumer Complaint Dataset. Among the models, BERT model outperformed, delivering better performance with accuracy of 82% and 89.5% on the Local and Consumer Complaint Datasets respectively. The proposed methodology will significantly improve complaint management efficiency and serve as a foundation for future improvements in automated text classification.

Key Words: *Deep Learning Models, Complaints Classification, Classification Techniques, GloVe, RNN, CNN, LSTM, Bi-LSTM, BERT, Natural Language Processing*

Table of Contents

List of Figures

List of Tables

CHAPTER 1: INTRODUCTION

In today's digital era, consumers extensively use online platforms to express their feedback concerns and complaints to respective departments and agencies as depicted in Figure 1.1. These platforms have transformed the way public services are delivered, making them more efficient and accessible. However, the massive influx of unstructured text data poses a significant challenge in managing and effectively categorizing the complaints as because manual sorting of the data is time-consuming and likely to have errors. Additionally, Natural Language Processing NLP faces a big challenge when dealing with complaints as these can be hard to understand because they're often missing context, might not follow grammar rules, or could even have spelling mistakes. Consequently, existing research in this domain has not yet utilized the models that are able capture the essential semantic and textual information of complaint data. And historically, text classification research has relied only on traditional classifiers like k-nearest neighbor (KNN), k-mean, support vector machines (SVM), Randon Forest, and naive bayes (NB). These methods, although useful, still fail to extract the contextual relationships between words in a text. For this reason, the need for robust deep learning models coupled with Natural Language Processing (NLP) techniques to accurately classify these consumer complaints has never been more urgent as NLP based deep learning models offers a more advanced way to understand the contextual and semantic information from user complaints and classify it into its respective classes.

Our Proposed System: This research focuses on developing an autonomous system capable of interpreting the context of complaints and categorize them into respective departments by using three deep learning models; Convolutional Neural Network (CNN), a hybrid of CNN and Bidirectional Long Short-Term Memory (BiLSTM), and the state-of-the-art Bidirectional Encoder Representations from Transformers (BERT) model.

Unlike traditional classifiers, Convolutional Neural Network (CNN) is specifically designed to understand the local correlation in the text, making it excellent at short-text classification as it captures correlations through convolution and pooling operations. Furthermore, a special kind of Recurrent Neural Network (RNN), known as Bidirectional Long Short-Term Memory (BiLSTM), has been found to be particularly effective as it has both forward and backward directions that allow it to excel in extracting contextual features from the text. Still, despite these advances, rendering a single extraction method proves to be less effective in some cases.

For this reason, this study proposes a hybrid model combining the strengths of Convolutional Neural Networks (CNN) and Bidirectional LSTM. The proposed model capitalizes on the benefits of both CNN, known for its ability to process local features, and Bidirectional LSTM, recognized for managing long-term dependencies in the text. The features extracted by these two models are combined to form a crucial unified representation and provide meaningful information for further classification. And hence, this hybrid model proves to be an improved autonomous classification model for classification of user complaints.

CNN and hybrid of BiLSTM-CNN perform well but a language model is considered more effective, if it understands the context of whole sentence by differentiating the meaning of a word based on the surrounding text as this feature is very essential when dealing with text-based complaints. In this context, this research study utilizes third deep learning model called Bidirectional Encoder Representations from Transformers (BERT) for classifying consumer complaints. The model's deep understanding of language and its ability to process text bidirectionally, allows it to understand the context of each word in relation to the entire sentence. This capability makes it particularly effective in handling often complex language used in consumer complaints. However, our proposed models, Convolutional Neural Network (CNN), CNN with Bidirectional LSTM and BERT outperformed previous traditional models.

In this study, we present a novel methodology for complaint classification. Firstly, we employ Natural Language Processing (NLP) techniques for preprocessing to refine the raw text. Subsequently, GloVe embeddings and the BERT Tokenizer are applied to extract preliminary feature vectors from the text. These feature vectors are fed to deep learning models to learn new patterns better contextual information thus improving the accuracy of complaint categorization. Moreover, this research stands as a novel effort that uses deep learning models for classifying complaints based on their contextual and semantic information. Hence, deep learning models with NLP are essential tools for classifying and managing consumer complaints in today's digital and customer-centric world.

Fig 1.1: Customer Feedback in Business Sector [38]

1.1 Motivation

Increasing consumer complaints requires continuous advancement in solutions to ensure customer satisfaction. Hence, an accurate, fast, reliable and scalable solution is required in this regard. Therefore, the selected research topic utilizes deep learning models in NLP as an alternative to the conventional customer care strategies. Deep learning models possesses the potential to address the aforementioned constraints. Need of Deep learning in the field of customer care can be highlighted by the following concerns.

1.1.1. The increasing dataset of consumer complaints:

Increase in the use of online banking causes rapid growth in the volume of consumer complaints. This leads towards the difficulty in organization of these volumes, hence, causes difficulty while managing them.

1.1.2. The growing need of accurate & scalable solution for consumer complaints classification

For the effective and timely response of consumer's complaints, it is essential to classify them. Classification helps prioritize consumer's complaints which leads to their effective investigation. Organizing these complaints also enables authorities to identify the trend in customer complaints hence resolving them for better services.

1.1.3. The growing potential of deep learning models for classification tasks.

Deep learning model validates its effectiveness to obtain accuracy in text classification which makes them favorable to utilize them for consumer complaints classification. By utilizing them, the process of classification become automatic hence, dealing with large data set could become easier and effective.

This research work will evaluate if deep learning models can effectively classify complaints into respective classes. For this purpose, various deep learning models will be considered, and critical analysis will be made after evaluating them. In literature, different optimization algorithms are utilized based on the nature of objective. So, a researcher should be vigilant in choosing learning models for text classification that proves to be both suitable for their data set and project requirements. Moreover, this research work will evaluate the performance of different deep learning models by training and testing on two datasets. One dataset comprises of 10,000 user complaints collected from a public education sector and the other dataset contains 1,20,000 complaints collected from an international online dataset repository.

1.2 Problem Statement

Due to fast growth in the use of online and mobile transactions, a rapid increase in the volume of consumer complaints is observed which causes difficulty in managing these complaints effectively. Effective management requires accurate and timely classification of consumer complaints so that organizations can be able to prioritize and investigate them accordingly. Classifying them also enables organizations to identify the trends which lead them to improve customer care service. This can be done by tuning deep learning models. Deep learning models have been validated to obtain high level of accuracy in the field of text classification. This makes them a useful approach for consumer complaints classification to automate the process of classification for handling large datasets. For this purpose, various methodologies need to be combined with machine learning techniques along with their effective tuning according to the nature of problem needed to be solved. In that way, classification of the complaints can be made into the respective department after successful prediction. Typically, machine learning classifiers are utilized to classify user complaints but there is a need of efficient and effective deep learning models which can provide more scalable and time saving approach to accurately predict the class (department) and classify the complaints accordingly. Therefore, this research work aims to provide valuable insights into the use of deep learning models for consumer complaints classification. The proposed model is designed to develop and deploy deep learning models which can offer improved accuracy and efficiency of their consumer complaints management.

1.3 Aims and Objectives

This research work aims to train various deep learning models for the classifications of consumer complaints and critically analyze them to observe the effectiveness of each model so that a suitable model can be selected which could be utilized to classify consumer complaints for the respective organization.

To achieve this aim, the following objectives have been identified.

- To combine literature techniques having different feature extraction to address our concerned problem.
- To provide timely, accurate and scalable solutions to the companies for better management of consumer complaints which will ensure customer satisfaction towards the organization.

To obtain the mentioned objectives, a three-step methodology is proposed. In the first step, a

large dataset that can be classified into multiple classes needs to be collected. After that, the collected data needs to be pre-processed before processing into deep learning model such as cleaning and transforming it into a suitable format. Finally, this data is subjected to deep learning models for testing and training. Different performance metrics needed to evaluate the performance of the models will be established.

1.4 Structure of Thesis

This work is structured as follows:

Chapter 2 describes technical background and different techniques used researchers in the past for Text Classification.

Chapter 3 gives review of the literature and the significant work done by researchers in past for the task of Complaints Classification. It also explains different techniques used to extract the features and classification by researchers and the datasets that can be used.

Chapter 4 consists of the proposed methodology in detail. It includes three main modules: pre-processing, vectorization or feature extraction and the deep learning models.

Chapter 5 consists of all the experimental results in detail with respective tables, figures, and performance measures.

Chapter 6 concludes the research work and reveals future scope of it.

CHAPTER 2: TECHNICAL BACKGROUND

One of the most transformative technological domains of recent decades is Natural Language Processing (NLP). It is redefining the ways in which we interact with machines. The primary aim is to allow computers to comprehend, interpret, and generate human language in a valuable and meaningful way. This field of study and application emerged in the mid-20th century, around the same time as the advent of modern computers.

2.1. Natural Language Processing

Natural Language Processing (NLP) is a subfield of evolving artificial intelligence (AI) that makes the interaction between humans and computers possible as depicted in Figure 2.1. It focuses on developing classifiers and models to enable computers and machines to interpret, and communicate in a language in a way that is meaningful and useful. NLP has gained significant attention and has seen remarkable advancements in recent years, driven by the availability of large-scale datasets, computational power, and innovative machine learning techniques [16].

One of the key challenges in NLP is dealing with the ambiguity of natural language. NLP models must be able to understand the context in which a word is used in order to disambiguate its meaning. Another challenge in NLP is dealing with the complexity exist in human language. Human language is full of nuances and subtleties that can be difficult for computers to capture [17]. NLP models must be able to learn to represent the meaning of words and phrases in a way that is robust to these complexities.

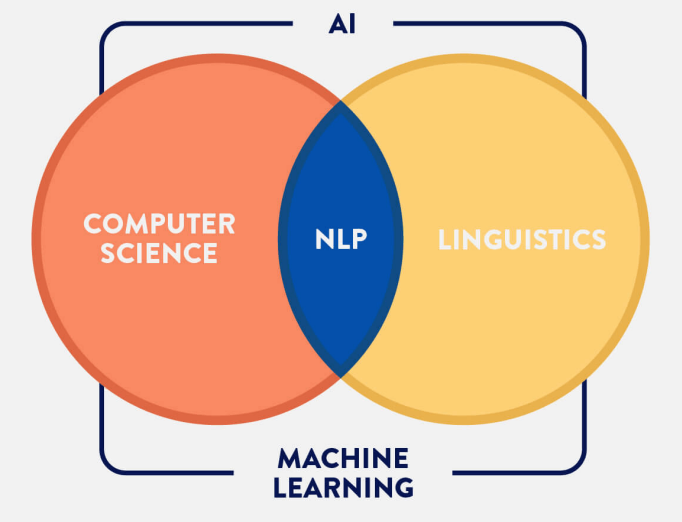


Fig 2.1: Relationship of NLP and other Fields [39]

2.1.1 History of NLP:

The field of NLP has its roots in the early days of artificial intelligence (AI). In the 1950s and 1960s, researchers began to develop computer programs that could process natural language. However, these early programs were limited in their capabilities. In the 1970s and 1980s, there was a renewed interest in NLP. This was due in part to the development in the field of machine learning classifiers that were better adopted for NLP tasks. In the 1990s, NLP saw significant progress. This was due in part to the availability of large corpora of text data, which could be used to train NLP models. In the 2000s, NLP has continued to advance. This is due in part to the development of deep learning algorithms, which have presented to be very effective for natural language processing tasks [18].

2.1.2 Linguistic Foundations

NLP relies on linguistic theories and concepts to understand and process human language effectively. It draws from various linguistic subfields, including morphology, syntax, semantics, and pragmatics. Morphology involves the study of word formation and the analysis of word structure, inflection, and derivation. Syntax deals with the study of sentence structure and the rules governing the arrangement of words. Semantics in text emphases on the context of words, and phrases. while pragmatics examines how context influences language interpretation and usage.

2.1.3. Tasks in NLP:

NLP encompasses a wide range of tasks [19] precisely depicted in Figure 2.2. Some of the common tasks include:

Text classification: This is the task of assigning a category to a piece of text. For example, a text classification model could be used to classify customer complaints into different categories, such as "product defect," "service issue," or "billing error."

Text summarization: This generates a precise version of a piece of text that retains the most important information. For example, a text summarization model could be used to generate a summary of a news article or a research paper.

Machine translation: This is the task of translating text from one language to another. For example, a machine translation model could be used to translate a website from English to Spanish.

Question answering: This is the task of answering modeled in natural language. For example, a question answering model could be used to answer questions about a factual topic, such as "What is the capital of France?"

Named entity recognition: This is the task of identifying entities in corpus, such as people, organizations, and locations. For example, a named entity recognition model could be used to identify the names of people in a news article.

Sentiment analysis: This is the task of extracting sentiment from the text, such as whether it is positive, negative, or neutral. For example, a sentiment analysis model could be used to determine whether a customer review of a product is positive or negative.

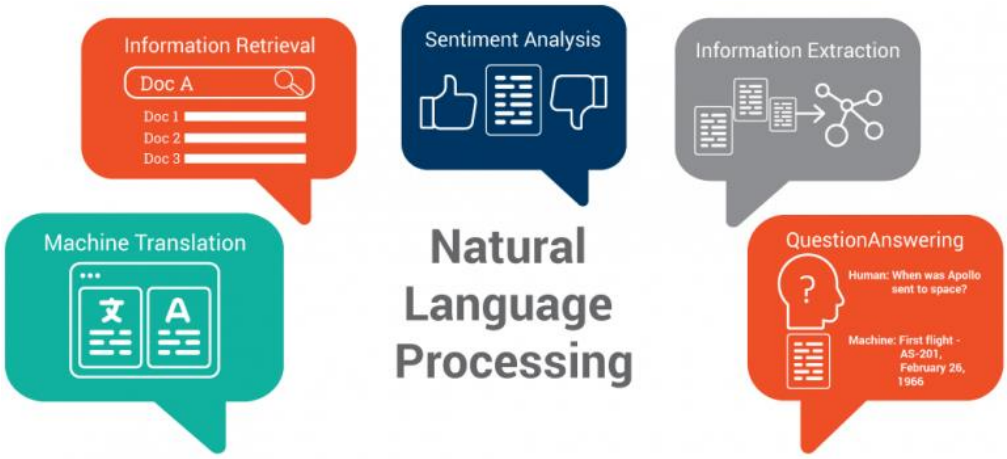


Fig 2.2: Tasks in Domain of NLP [40]

2.1.4 Preprocessing Techniques

To prepare text data for analysis, NLP involves several preprocessing techniques [20] as given below:

Tokenization: Breaking text into individual words or tokens.

Stop Word Removal: Filtering out commonly occurring words (e.g., "and," "the") that do not carry significant meaning.

Lemmatization: Reducing words to their base or dictionary form (e.g., "running" to "run").

Part-of-Speech (POS) Tagging: Assigning grammatical tags (e.g., noun, verb) to words.

Named Entity Recognition (NER): Identifying and classifying named entities such as names, locations, and organizations.

Sentiment Analysis: Determining the emotional tone of a text (e.g., positive, negative, neutral).

2.1.5 NLP Applications:

Industrial applications of NLP can be mostly categorized into 3 categories: Conversational systems, Text Analytics, Machine translation

Conversational Systems: Using a speech or text-based interface, we may converse with an automated computer in natural language using a conversational system. A company firm helps to automate complex procedures with 24X7 assistance to its users. Virtual Assistants and Chatbots are the two most frequent types of conversational gadgets. Banks, e-commerce, social networking, and other self-provider factor-of-income systems all use these devices to serve their clients with a wide range of services.

Machine Translation: Device translation is the venture of automatically translating one natural language into another, retaining the means of the input text [19]. Maximum famous software for device translation is Google translator. Other machine translation software programs are also utilized in speech translation and teaching. Now, we will observe some industrial packages in the following area regions: Healthcare, car, Finance, manufacturing, retail, education, and customer service.

Text Analytics: Text Analytics additionally called textual content mining pursuits to extract meaningful content from text, either in files, emails, or brief-form communications such as tweets and SMS texts. Most commonplace use cases of textual content analytics on social media analytics.

2.1.6 Challenges in NLP:

NLP poses several challenges due to the complexity of human language. Some common challenges include:

Ambiguity: Words or phrases with multiple meanings or interpretations.

Syntax and Grammar: Variations in sentence structure, grammar rules, and syntactic conventions across different languages.

Named Entity Ambiguity: Identifying and disambiguating named entities with similar names or acronyms

Contextual Understanding: Capturing and leveraging contextual information for accurate language understanding and generation

Data Limitations: The need for large, diverse, and annotated datasets to train and evaluate NLP models effectively.

2.1.7. Deep learning in NLP:

Deep learning has had a major impact on NLP. Deep learning models have shown to achieve state-of-the-art results on a wide range of NLP tasks, such as text classification, text summarization, and machine translation [21]. Deep learning models are able to achieve these results by learning to represent the meaning of words and phrases in a way that is robust to the ambiguity and complexity of natural language. Deep learning models are able to learn these representations by being trained on large datasets of text data as explained in detail in Section 4.

2.2 Text Pre-Processing

Pre-processing techniques [20] play a crucial role in deep learning-based text classification in Natural Language Processing (NLP) as it involves transforming raw text data into a format suitable for analysis and modeling. Deep learning has revolutionized text classification in NLP by leveraging powerful neural network architectures to learn complex representations directly from raw text data. Pre-processing techniques are employed to transform raw text into a suitable format for deep learning models. These techniques ensure data consistency, handle variations in text, and enhance the performance and efficiency of text classification tasks. It aims to clean and structure the text, removing noise, irrelevant information, and inconsistencies. This can lead to better performance on downstream tasks, such as text classification, text summarization, and machine translation. This step can help to reduce the size of the dataset. This can be important for NLP models that are trained on large datasets. By removing stop words and noise, the size of the dataset can be reduced without significantly affecting the performance of the model. Pre-processing techniques can also help to improve the speed of NLP models. By normalizing text and removing stop words, NLP models can process text more quickly. This can be important for real-time applications, such as chatbots and search engines. It covers essential steps such as tokenization, stop word removal, stemming, part-of-speech tagging, padding, word embeddings, data augmentation, and strategies to handle OOV words named entity recognition. Understanding these techniques is essential for ensuring accurate and effective NLP analysis and modeling. However, it's important to note that the pre-processing steps may vary depending on the specific requirements of the task and the characteristics of the text data. Moreover, this technical background provides a comprehensive overview of the pre-processing techniques commonly used in NLP.

2.2.1. Tokenization:

Tokenization is the process of splitting text into individual words or tokens. It involves breaking down sentences, paragraphs, or entire documents into smaller units, such as words or sub words. Tokenization can be achieved using simple techniques like splitting on whitespace or more advanced methods like using language-specific rules, regular expressions, or machine learning algorithms [22] or more advanced tokenization methods such as using regular expressions or libraries like NLTK or spaCy. Specifically, the **texts_to_sequences** method is used to convert the text data into sequences of integers based on the vocabulary learned from the training data

2.2.2. Conversion of Text into Lower Case:

In NLP, converting text into lowercase is a widely used preprocessing step with significant advantages. Firstly, it unifies words that have the same meaning but different capitalizations (e.g., "Bank" and "bank" are treated as one word). This simplifies the vector space model, reducing the number of dimensions and making text analysis more efficient. Secondly, lowercase conversion enhances text normalization, ensuring consistent formats across the dataset and improving various NLP tasks like sentiment analysis and text classification. Additionally, it reduces the vocabulary size by treating variations of a word as a single term, reducing computational complexity during training and inference. However, a potential drawback is the loss of semantic meaning for some terms, like "US" (United States) and "us" (pronoun). Thus, striking a balance between lowercase conversion and case sensitivity is essential for optimizing NLP model performance.

2.2.3. Stop Word Removal:

Stop words are common words that do not carry significant meaning and often appear frequently in text. Examples of stop words include "the," "and," "is," and "a." In NLP, removing stop words is a common pre-processing step as they can add noise to the analysis without providing valuable information. Stop word removal helps reduce the dimensionality of the data and improves the efficiency and accuracy of subsequent NLP tasks. This can be done using a pre-defined list of stopwords or using libraries like NLTK.

2.2.4. Removal of Punctuation

The removal of punctuation is an important preprocessing technique in NLP tasks to ensure consistent and standardized text representation. By eliminating punctuation marks, such as commas, periods, and exclamation points, each word is treated equally, regardless of its context within the sentence. However, caution must be exercised during this process to preserve the meaning of contraction words like "don't" that might lose their significance when separated. Decisions on which punctuations to exclude should be carefully made based on specific use cases. While Python's string.punctuation provides a standard list of punctuation symbols, customizing the list according to the context can enhance the accuracy and contextual understanding of NLP models. Striking the right balance between removing punctuation for consistent representation and preserving the meaning of essential linguistic constructs is essential for effective text preprocessing in NLP applications.

2.2.5. Stemming and Lemmatization:

Stemming and lemmatization are techniques used to reduce words to their base or root form. Stemming involves removing affixes from words, such as plurals or verb conjugations, to derive the stem or root. It relies on heuristic rules and pattern matching algorithms. Lemmatization, on the other hand, maps words to their base or dictionary form using linguistic rules and morphological analysis. It ensures that the resulting word is a valid word with a semantic meaning.

2.2.6. Padding and Sequence Length:

Ensure that all input sequences have the same length by padding shorter sequences with zeros or truncating longer sequences. This is necessary to create consistent input dimensions for the CNN model.

2.2.7. Truncation:

Truncation is a crucial preprocessing step in NLP, involving the removal of parts of a text that exceed a certain length or from the beginning or end of a sequence. Its primary purpose is to standardize the length of input sequences, especially in tasks using neural network models like BERT, GPT, or LSTM. With fixed-length inputs, models can be processed efficiently, enabling training and evaluation on data batches. There are two common types of truncations: right

truncation, which retains essential information at the beginning of the text, and left truncation, which preserves recent or relevant details at the end. The choice between left and right truncation depends on the specific NLP task and contextual significance. However, care should be taken not to remove critical information that could impact the model's performance. Truncation, when combined with other preprocessing steps like tokenization and lowercasing, ensures consistent and standardized input sequences, leading to enhanced model generalization and performance across diverse NLP tasks.

2.2.8. Part-of-Speech (POS) Tagging:

Part-of-speech tagging involves assigning grammatical tags to words in a sentence based on their role and context. Common POS tags include noun, verb, adjective, adverb, and pronoun. POS tagging helps in understanding the syntactic structure of sentences, identifying relationships between words, and disambiguating word senses. POS-tagged data is useful for tasks like text classification, information extraction, and machine translation.

2.2.9. Named Entity Recognition (NER):

Named Entity Recognition is the process of identifying and classifying named entities in text, such as names of people, organizations, locations, dates, and quantities. NER plays a vital role in information extraction, question answering, and text summarization. It involves using machine learning algorithms or rule-based approaches to identify and classify named entities accurately.

2.2.10. Normalization and Spell Checking:

Normalization involves transforming text to a standardized format, reducing variations due to capitalization, punctuation, and special characters. This process converts text to a standard form. This involves tasks such as lowercasing all text, removing punctuation, and stemming words. Normalization can help to improve the accuracy of NLP models by making text more consistent. It ensures consistent representation and comparability of text data. Additionally, spell checking techniques can be applied to correct spelling errors in text, enhancing the accuracy of subsequent NLP tasks.

2.2.11. Handling Out-of-Vocabulary (OOV) Words:

In deep learning for text classification, encountering words that are not present in the pre-trained word embeddings can pose a challenge. Techniques such as replacing OOV words with

a special token or learning contextualized word representations can be employed to handle OOV words effectively.

A few basic pre-processing steps in NLP are mentioned below in the Figure 2.3.

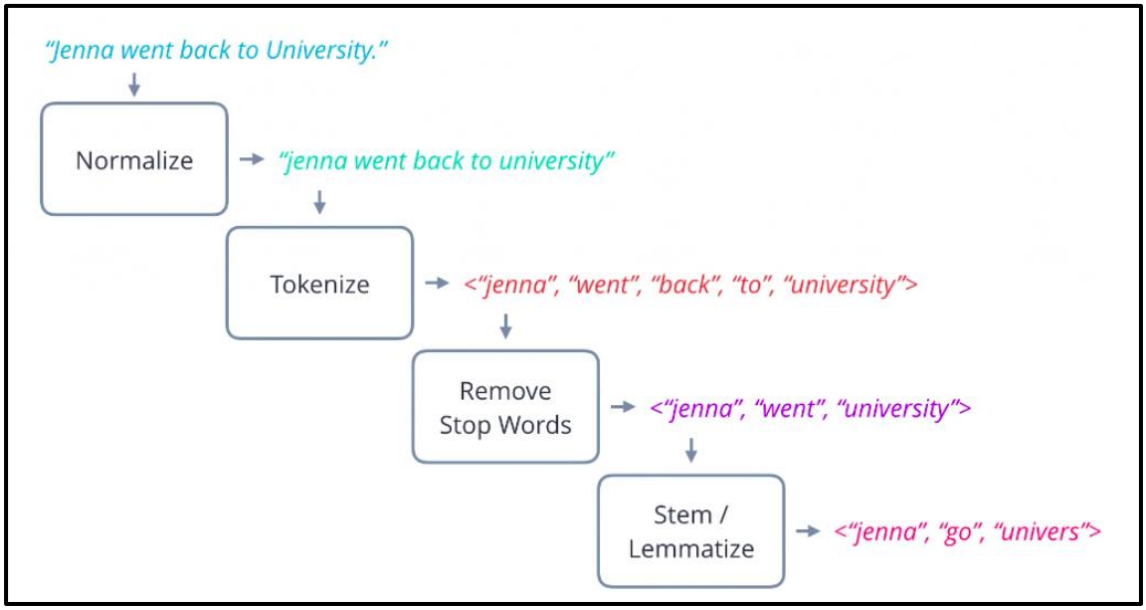


Fig 2.3: Example of Pre-processing [41]

2.3. Feature Extraction Techniques used in Deep learning Models

Deep learning has appeared as a powerful technique for text classification in NLP, leveraging neural network architectures to learn complex representations from raw text data. Feature extraction techniques are employed to transform text into meaningful numerical representations that deep learning models can process. Effective feature extraction is crucial for capturing relevant information and patterns, enabling accurate text classification meaning this step is important in text classification, as it allows the machine learning model to learn the important patterns in the text that are relevant to the classification task.

2.3.1. Word Embeddings:

This is the process of representing words as vectors of real numbers. The vectors are learned from corpus, and they are able to extract the semantic meaning of words appeared in text. Word embeddings can be used to create feature vectors for text classification by simply taking the feature vectors of each word in the document. They encode contextual meaning and help the deep learning models understand the underlying meanings in the text. Pre-trained word embeddings, such as Word2Vec, GloVe, or FastText [23] can be utilized to initialize the

embedding layer of deep learning models or fine-tuned during the training process.

2.3.1.1 GloVe Embeddings

GloVe (Global Vectors for Word Representation) is a widely adopted word embedding technique in Natural Language Processing (NLP). It represents words as dense vectors in a continuous vector space, capturing semantic relationships based on their co-occurrence statistics in large text corpora. Word vectors place words in a space where similar words group together and different words are farther apart. The key idea is to factorize the word co-occurrence matrix, allowing GloVe to learn fixed-size vectors for each word. Words with similar contexts appear closer in the embedding space, facilitating the discovery of meaningful semantic relationships [24].

GloVe is a popular word embedding technique similar to Word2Vec, and it's used to convert text data into numerical vectors which can be easily processed by deep learning models [23]. Word embeddings transforms words into vectors by using pre-trained word vectors file from the GloVe (Global Vectors for Word Representation) project, developed by the Stanford NLP Group. Each word has corresponding n-dimensional vector, implying that each word is described by n features in the embedding space. These vectors capture semantic and syntactic information about the words, and words have similar vectors that are similar in meaning. These vectors are then used as input for machine and deep learning models for NLP tasks.

For this process, each sentence is converted into words and unique indexes are assigned to each word. And embedding provides a d-dimensional vector for each index as shown in the following Figure 2.4.

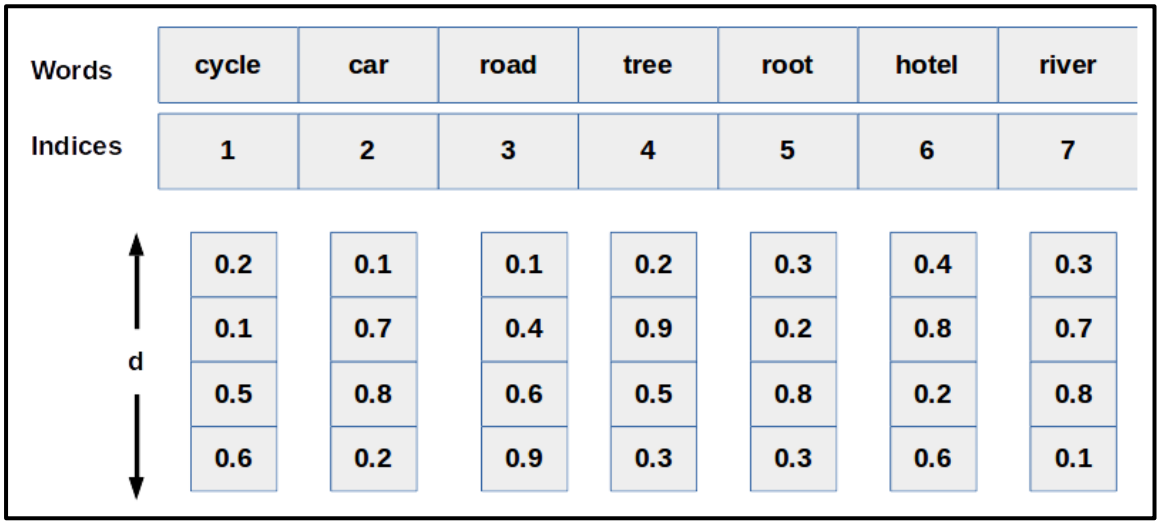


Fig 2.4: Vector Representation using GloVe [42]

2.3.2. Attention Mechanisms:

Attention mechanisms enable deep learning models to target relevant parts of the text while making class predictions. They assign weights to different words or sub words in the text, emphasizing the most informative elements. Attention mechanisms focus on important features and capture fine-grained details, improving the accuracy of text classification. Transformer-based models, such as variants like BERT, extensively use attention mechanisms.

2.3.3. Transfer Learning and Fine-tuning:

Transfer learning is the process of leveraging pre-trained models on large-scale datasets to extract features that are relevant for a specific task. Pre-trained models, such as transformer and its variant like GPT and BERT, can be fine-tuned on task-specific data to improve text classification performance. Fine-tuning makes the model to adapt to the specific nuances of the target classification task.

2.3.4. Ensemble Methods:

Ensemble methods combine several models to improve the overall performance and generalization of the text classification system. Ensemble techniques, such as model averaging or stacking, can be applied to deep learning models for text classification. By aggregating the predictions of multiple models, ensemble methods enhance robustness and accuracy.

2.4. Deep Learning Models

Deep learning focuses on training multiple neural networks with hidden layers to learn hierarchical representations of data. It is inspired by the structure and function of the human brain, where interconnected neurons processes information. Deep learning models, also known as deep neural networks, consist of multiple layers of interconnected artificial neurons called nodes or units. Each layer receives inputs, performs computations on them, and produces outputs that are passed to the next layer. The layers are typically organized in a sequential manner, with the initial layers learning low-level features and the subsequent layers learning higher-level features. The key advantage of deep learning is its ability to learn representations from text data, without the need for manual feature extraction. Deep neural networks can learn

complex patterns directly from the data, making them particularly powerful for image and speech recognition, natural language processing, and various other domains. Training a deep learning model involves feeding it with a large dataset and adjusting the weights and biases of the network's parameters through a process called backpropagation. Backpropagation uses an optimization approach, that is gradient descent, to iteratively update the model's parameters in order to minimize the cost or loss function that continuously measures the difference between actual ground truth values and predicted values.

Deep learning has shown remarkable success in past few years, largely due to advancements in computational power, the accessibility and availability of diverse datasets, and the development of more sophisticated neural network architectures. Deep learning models, such as convolutional neural networks (CNNs) for image analysis and recurrent neural networks (RNNs) for sequential data,

Bidirectional Encoder from Transformer (BERT) have achieved state-of-the-art performance on various complex tasks. However, training deep learning models can be computationally intensive and requires large amounts of labeled data. Additionally, deep learning models are often considered black boxes, making their decision-making processes less interpretable compared to traditional machine learning models. Nevertheless, deep learning continues to drive advancements in various fields and remains a highly active and exciting area of research and development. Figure 2.5 shows the hierarchy of deep learning architecture.

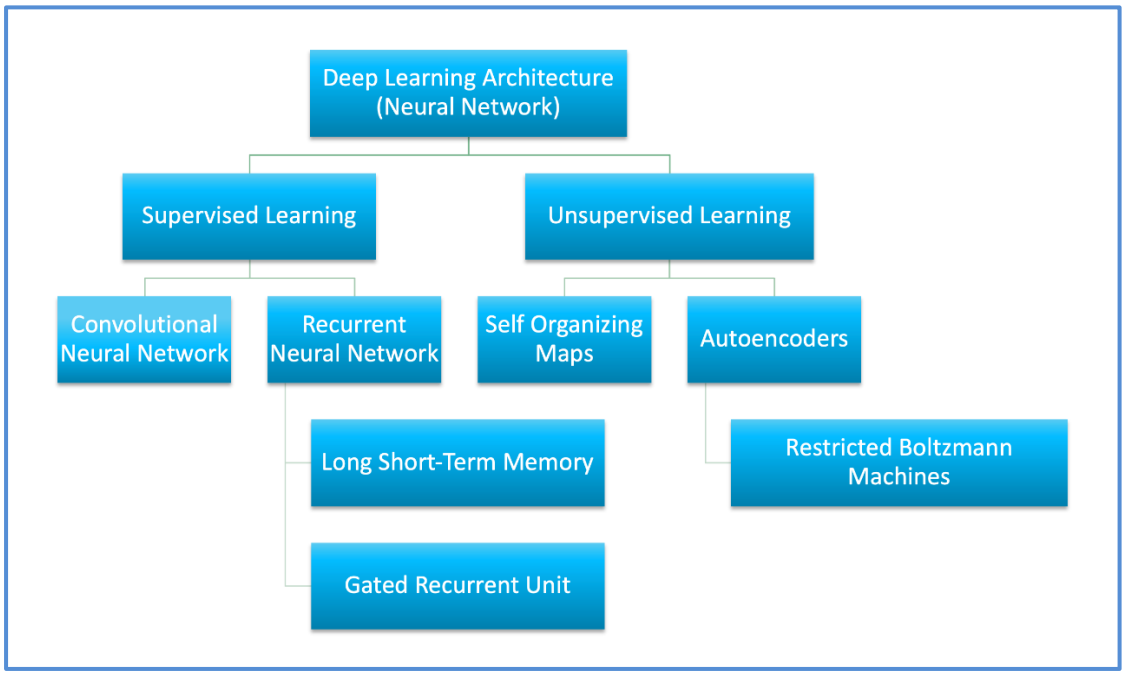


Fig 2.5: Deep Learning Architecture [43]

2.4.1. Convolutional Neural Network (CNN):

CNN is a multi-layered neural network as shown in the Figure 2.6; each layer is composed of multiple 2D surfaces, and each plane is composed of multiple independent neurons [24]. A group of local units is the next layer in the upper adjacent unit of input; this views local connection originating in perceptron.

Convolutional Neural Networks (CNN) have been widely applied in computer vision tasks, such as image recognition and text classification. However, CNNs have also shown promising results in text classification, a fundamental task in NLP. CNN are deep learning models designed to automatically learn hierarchical representations from input data. CNN architecture is built on multiple layers like convolutional layers, dropout layers, pooling layers, and fully connected layers. The core idea behind CNNs is the use of convolutional operations to extract local patterns from the corpus. Convolution is a mathematical operation that takes a 1D input and produces a 1D output in case of text. In the context of NLP, the input to a CNN is a sequence of words, and the output is a feature vector that represents the relationships between the words in the sequence.

CNNs for text classification typically have the following architecture:

1. An embedding layer that converts words into vectors of real numbers.
2. A convolutional layer that applies a convolution operation to the word vectors.
3. A pooling layer that down-samples the output of the convolutional layer.
4. A fully connected layer that classifies the text.

The embedding layer converts words into vectors of real numbers. This is done by using a technique called word embedding. Word embedding is a process of representing words as vectors of real numbers that capture the semantic meaning of words.

The convolutional layer applies a convolution operation to the word vectors. The convolution operation looks for patterns in the word vectors. These patterns can be used to represent the relationships between words in a sentence.

The pooling layer down-samples the output of the convolutional layer. This is done to reduce the size of the output and to improve the computational efficiency of the CNN.

The fully connected layer classifies the text. This is done by using a technique called classification. Classification is a process of assigning a category to a text.

Model's parameters are further optimized after extracting features by the above-mentioned layers to achieve accurate classification by using the steps as follow:

Training and Optimization:

CNNs for text classification are trained using labeled data. The model's parameters are optimized through backpropagation and gradient descent techniques, minimizing a defined loss function (e.g., cross-entropy). Dropout and batch normalization are proved to be regularization techniques to prevent overfitting.

Hyperparameter Tuning:

CNNs have several hyperparameters that influence their performance, such as the number of convolutional filters, filter sizes, pooling sizes, learning rate, and batch size. These hyperparameters need to be carefully tuned to achieve optimal performance. Techniques like grid search or random search can be used to explore the hyperparameter space.

Evaluation and Performance Metrics:

After training, the CNN model is evaluated using evaluation metrics such as accuracy, precision, recall, and F1-score. Cross-validation or hold-out validation strategies can be employed to assess the model's generalization performance.

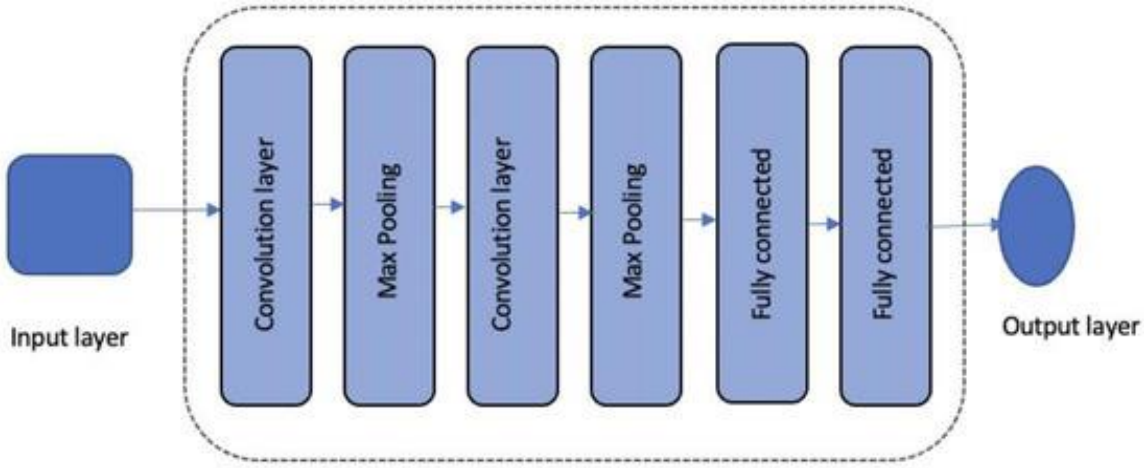


Fig 2.6: Convolutional Neural Network [44]

2.4.2. Recurrent Neural Networks (RNN)

Recurrent Neural Networks (RNNs) are a class of neural networks commonly used in Natural Language Processing (NLP) tasks due to their ability to process sequential data. Unlike traditional feedforward neural networks, they capture contextual dependencies by propagating information from previous words to the current word. Long Short-Term Memory (LSTM) [25] and Gated Recurrent Unit (GRU) are popular variants of RNNs.

The basic idea behind RNNs is to process input data one element at a time while maintaining a hidden state that represents the network's memory or context. This hidden state is updated at

each time step and depends not only on the current input but also on the previous hidden state. This mechanism allows RNNs to capture dependencies and patterns in sequences.

Architecture: The RNN consists of recurrent units, and each unit processes a single element of the input sequence at a time. In NLP, these elements are typically words or tokens. The recurrent units are interconnected in a way that allows information to flow from one time step to the next.

Input Representation: To feed text data into an RNN, you need to represent words or tokens as numerical vectors. Common approaches include word embeddings like Word2Vec, GloVe that convert words into dense vector representations. These word embeddings can be learned from scratch during training or loaded from pre-trained embeddings.

Sequence Processing: Each word or token in the input sequence is passed into the RNN one at a time, along with the previous hidden state. The RNN processes the input and updates its hidden state at each time step. The final hidden state after processing the entire sequence carries information about the entire context of the input.

Backpropagation Through Time: Training an RNN involves using the backpropagation algorithm to update the model's weights based on the error between the predicted output and the true output. However, RNNs are inherently deep due to the unfolding of the recurrent connections through time. This requires a variant of backpropagation called BPTT, which unfolds the network in time and calculates gradients at each time step. Basic Recurrent Neural Network is mentioned below in the Figure 2.7.

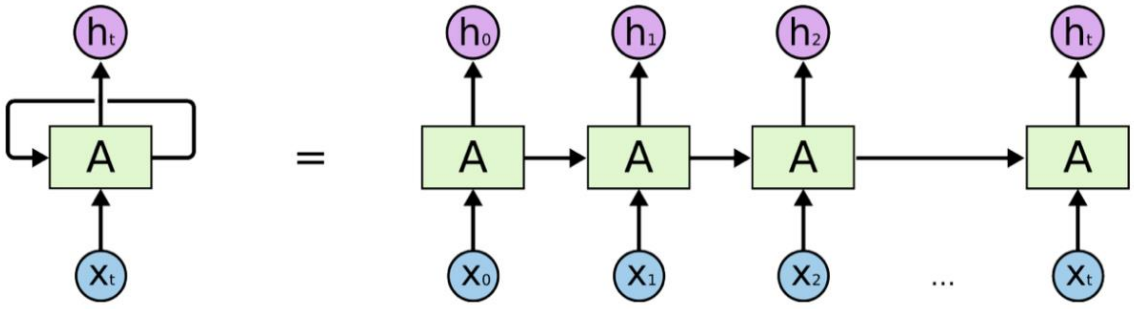


Fig 2.7: Recurrent Neural Network [45]

Although RNNs are effective for certain NLP tasks, they suffer from some limitations, such as difficulty in capturing long-range dependencies (vanishing and exploding gradients) and the inability to remember information from early time steps for long sequences. This is where Long

Short-Term Memory (LSTM) units come into play.

2.4.3. LSTM Based RNN

Long Short-Term Memory (LSTM) based Recurrent Neural Networks (RNNs) are a specific type of RNN [27] designed and its key feature is its unique memory cell structure, which allows it to store and recall information over extended periods. An LSTM-based RNN contains memory cells arranged in a layer [28]. These cells communicate with the rest of the network through carefully regulated structures known as gates. There are three types of gates: the input gate, which determines how much of the new information to let into the cell; the forget gate, deciding what information to discard; and the output gate, controlling the amount of cell information to output to the next layer [29]. These gate mechanisms allow LSTM-based RNNs to selectively remember or forget information, making them particularly effective in tasks requiring the understanding of long-term sequential dependencies, such as language translation, text generation, and speech recognition.

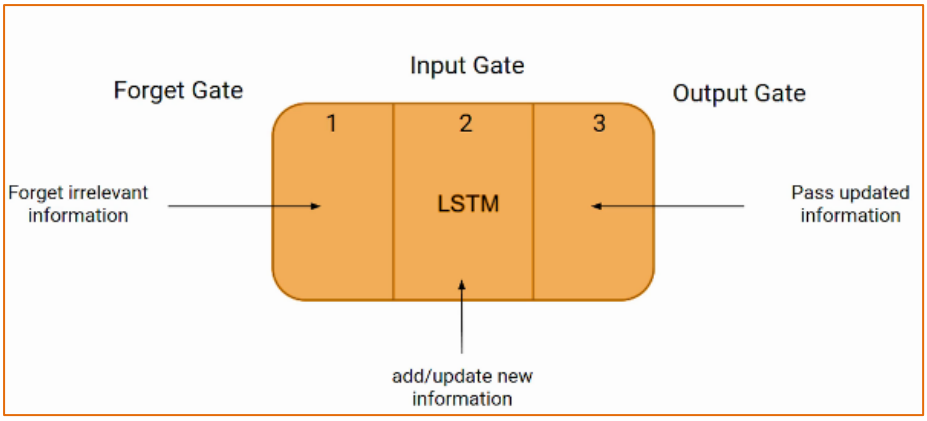


Fig 2.8: Gates Mechanism in LSTM [29]

However, in simple terms, LSTM units act as memory components within the network, capable of retaining context from the beginning of an input sequence. For example, if a sentence is 10 words long and the task is to predict the 11th word, the RNN processes all 10 words. In this way, LSTM-based RNNs have revolutionized the field of sequence prediction and time-series analysis.

2.4.4. BERT

In 2018, a significant advancement was made in the field of deep learning, particularly in the

realm of Natural Language Processing (NLP) - this was marked by the advent of successful transfer learning techniques. A method that gained notable prominence was the Bidirectional Encoder Representations for Transformers, also known as BERT. This model, because of its exceptional performance, has set a new benchmark in NLP, redefining many language-oriented operations. BERT's novelty lies in its capacity to understand the contextual meanings of words by scrutinizing both the left and right contexts and textual elements [32]. In contrast to Traditional language models, which typically analyze text in a unidirectional manner, - either forward or backward, BERT focuses on the bidirectional functionality of Transformers. This provides it a more holistic insight into the context of the language, ensuring a more robust understanding of how words are used within a specific context. Figure 2.9 depicts how BERT works with stacked layers making it state-of-the-art model.

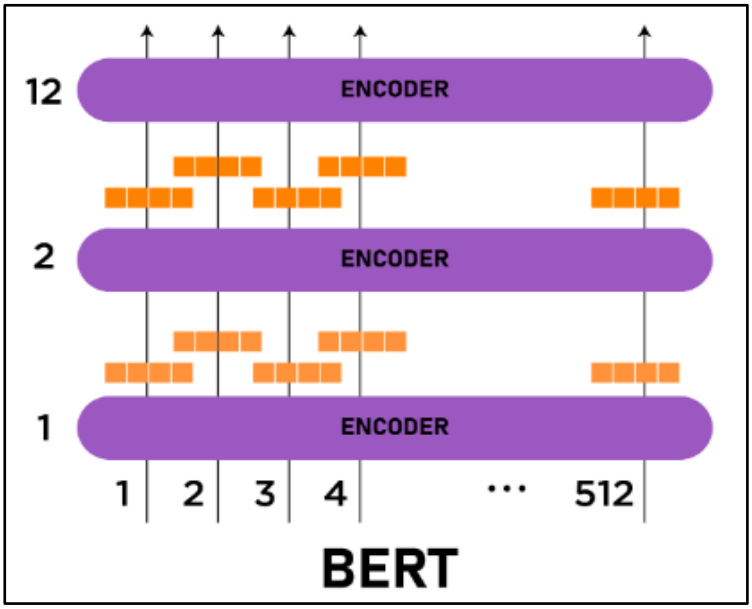


Fig 2.9: Stacked Layers of BERT [46]

2.4.5.1 Architecture

BERT consists of multiple encoder layers called Transformer Blocks, and each block contains two main components: the self-attention mechanism and the feedforward network. The self-attention mechanism captures dependencies and relationships between the words in the input. This helps BERT understand how different words relate to each other and form meaningful connections within the text. After the self-attention step, a feed-forward network is applied to further transform the representations of the words. This network introduces non-linear transformations to capture complex patterns and structures within the input. So basically, the feed-forward networks enable non-linear transformations. This way the output of each layer is

passed to the next encoder layer, creating a stacked structure of multiple layers. Each layer builds upon the representations learned in the previous layer, gradually refining the understanding of the input text [32] as shown in the Figure 2.10.

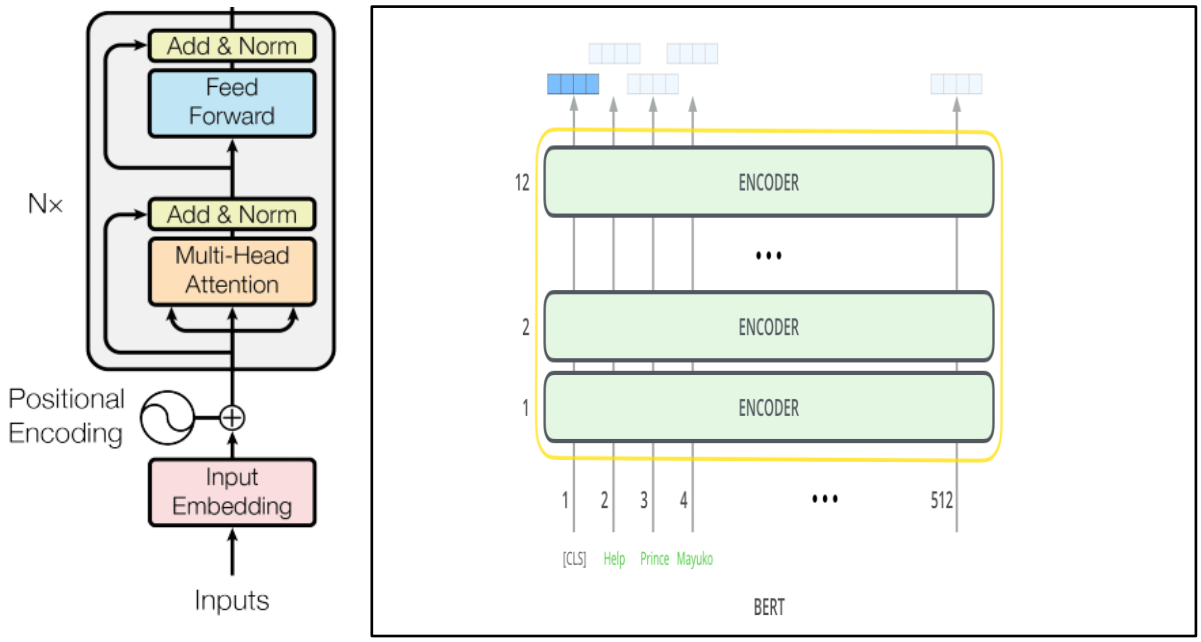


Fig 2.10: Flow inside an Encoder Layer of BERT [32]

BERT model is available in two sizes as pre-trained model shown in Figure 2.11: BERT BASE and BERT LARGE. BERT BASE consists of twelve encoder layers, while the LARGE version has twenty-four encoder layers. Compared to the initial Transformer, BERT employs larger feedforward networks with 768 and 1024 hidden units, respectively [33] Basically, the feedforward network consists of multiple layers, and each layer contains hidden units. These hidden units are intermediate representations within the network that help capture and transform the information present in the input data. The size or number of hidden units determines the dimensionality of these intermediate representations.

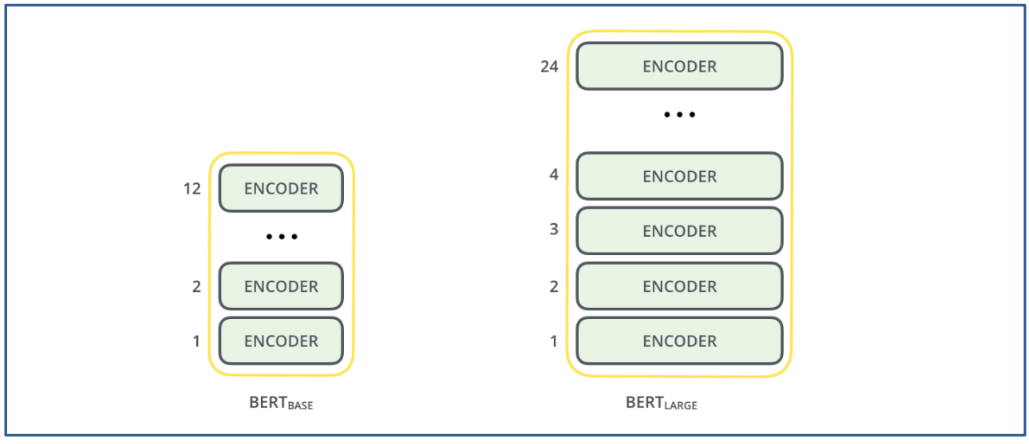


Fig 2.11: Sizes Available in BERT [33]

In BERT, two different sizes of hidden units are used: 768 for the BASE model and 1024 for the LARGE model. These values indicate the dimensionality of the hidden representations within the feedforward networks. A higher number of hidden units allows for more expressive power in capturing complex patterns and representations in the input data. To put it simply, hidden units in BERT represent intermediate features or representations within the feedforward networks. The sizes 768 and 1024 denote the dimensionality of these representations as mentioned in Figure 2.12. By increasing the number of hidden units, BERT can potentially capture more intricate linguistic patterns and contextual information, which can lead to improved performance on various language-based tasks. But the specific choice of hidden unit sizes in BERT, such as 768 and 1024, has been determined through experimentation and empirical evaluation to strike a balance between model complexity and performance. These sizes have been found to work well for a wide range of NLP tasks, showcasing the effectiveness of the chosen architectures in capturing rich semantic and syntactic information from textual data.

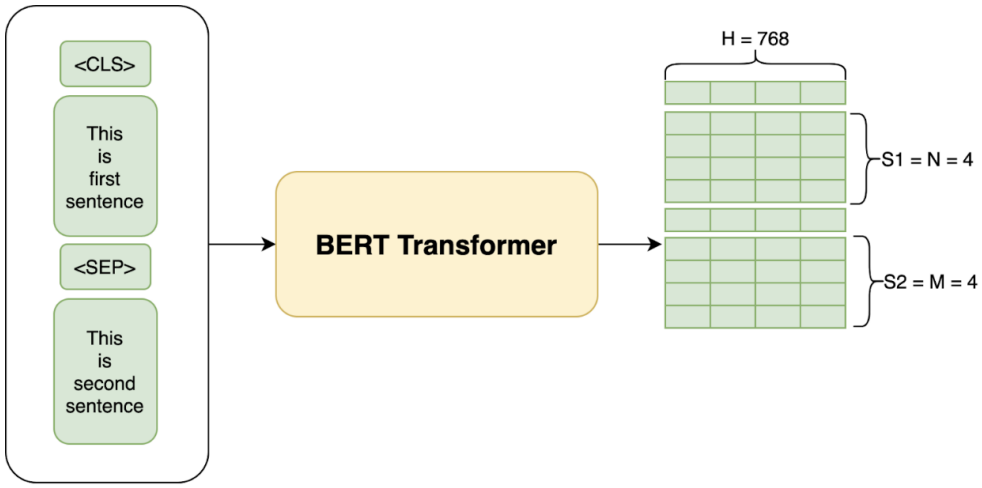


Fig 2.12: Preprocessing and Vectorization by BERT Tokenizer [47]

2.4.5.2 Training Inputs

BERT, which stands for Bidirectional Encoder Representations from Transformers, utilizes three types of embeddings as part of its architecture: WordPiece token embeddings, Segment embeddings, and Position embeddings as shown in the Figure 2.13. Each of these embeddings contributes to the model's ability to understand the semantic and syntactic context of words in sentence[34].

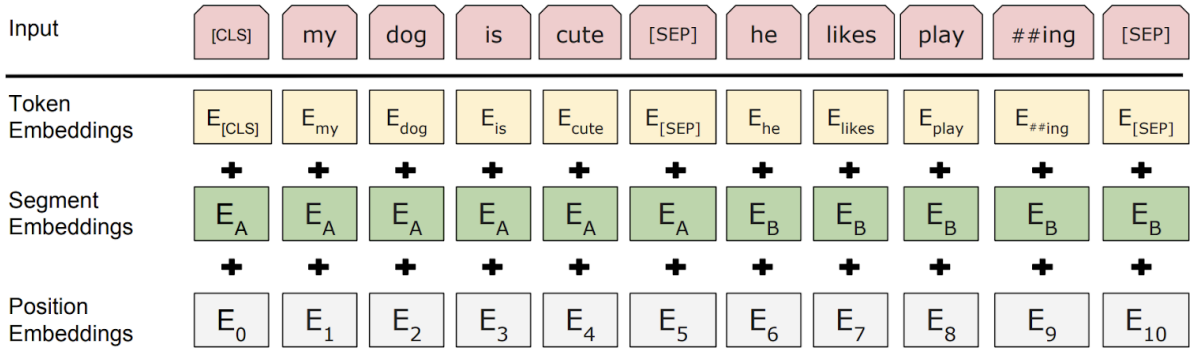


Fig 2.13: Embeddings in BERT Tokenizer [34]

WordPiece Token Embeddings: These are embeddings for sub-word tokens. A single English word might be represented as one or more sub-word tokens. For example, the word 'embeddings' could be split into two tokens 'embed' and '##dings'.

Segment Embeddings: BERT is designed to handle pairs of sentences for tasks such as question-answering or natural language inference. For this reason, it needs to be able to distinguish between the first sentence and the second sentence. To accomplish this, BERT adds a sentence (or segment) embedding to each token indicating whether it belongs to the first sentence or the second one. There are only two segment embeddings, one for the first sentence and one for the second.

Position Embeddings: As BERT doesn't have recurrence or convolution which would inherently handle sequence ordering, it needs a mechanism to incorporate the order of words into its understanding of a sentence. Position embeddings are added to give the model information about the position of the words in a sentence. In BERT, these are learned, not hardcoded.

For each token, these embeddings are added together to form a single input representation [35]. So, the input at each position in the input sequence is the sum of the corresponding token, segment, and position embedding. These combined embeddings are then passed into the Transformer encoder. The Transformer architecture uses self-attention mechanisms and feeds these embeddings through layers of self-attention and feed-forward neural networks. Through training, the model learns to generate contextualized representations for each token.

2.4.5.3 Pre-training Model:

However, by using the above parameters, the model is pre-trained in two main steps: masked language modeling and next sentence prediction [35].

Masked Language Modeling: In this step, a certain percentage of the input tokens (words or sub-words) are randomly replaced with a special token called [MASK]. The model is then trained to predict the original tokens that were replaced with [MASK]. By doing this, BERT learns to understand the syntax, meaning, and relationships between different words in a sentence. It essentially learns to fill in the blanks in a sentence, even when some words are missing or replaced.

Sentence Prediction. In this next step, the model is trained to determine whether two given sentences appear consecutively in the original text or not. The model learns to understand the contextual flow of information and distinguish between sentences that are logically connected and those that are not. By going through these two pre-training steps, BERT gains a deep understanding of the syntax, and relationships within sentences and the coherence between consecutive sentences. It learns to predict missing words and comprehend the overall structure of text. This pre-training process equips BERT with valuable language understanding capabilities.

2.4.5.4 Role of Special Token

Another distinctive aspect of BERT is the presence of a special token, [CLS], added as the first input token. BERT takes a sequence of words as input, which is processed through the stacked layers. Each layer applies self-attention to capture dependencies and relationships within the input, followed by a feed-forward network for further transformation [36]. The output of each layer is then passed to the subsequent encoder.

At last, the output of BERT is received that is vectors of size equal to hidden_size (768 in case of BASE model). For tasks such as sentence classification, the focus is often on the output of the first position, to which the special [CLS] token was supplied. This position's output encapsulates the overall representation of the entire input sequence, providing a contextualized embedding that can be utilized for downstream tasks.

By pre-training BERT on massive datasets and making the pre-trained models readily available, the BERT framework has significantly reduced the time, effort, and resources required for training language processing models from scratch.

CHAPTER 3: LITERATURE REVIEW

This research work encompasses literature from multiple sources including journals, conferences, websites and case studies, focusing on User Complaints Classification. The main objective of this section is to review the existing knowledge related to User Complaints Classification. I have reviewed previous work that has been done to classify text in different domains by checking databases like IEEE Xplore, ResearchGate, Springer Link, Science Direct and other notable journals for this domain. For the purpose of searching, the keywords used are text classification, machine learning models for complaints classification, deep learning models, classification techniques, machine learning techniques for text classification, RNN, CNN, Bi-LSTM, BERT. The span of publication considered for this research work is from 2001 to 2022.

In recent years, the advancement in natural language processing (NLP) has captured global research attention. The aim is to optimize language models for better comprehension of text so efforts from researchers all around the world focus on refining language models, leading to varied methodologies. Some researchers work on contextual features and embeddings, while others introduce novel models and techniques. Prior research suggests that efforts have been made to introduce techniques that understand rules of language better specially in the field of customer services. In this section, we will briefly discuss the contribution of multiple researchers and their proposed techniques.

1. Dien Tran Thanh [1], and Bui Huu Loc proposed a new approach to classify articles by the aid of natural language processing (NLP) and machine learning (ML) as with the abrupt increase of data sources usually referred as text classification, have posed an important and challenging role in numerous fields. Therefore, the key objective of designing the proposed model to classify the articles by using pre-processing that includes removing of stop words, stemming the words, and creating a vocabulary and feature extraction. Features are extracted using bag-of-words, TF-IDF, and n-grams. And, lastly, classification of articles is achieved by using three classifiers SVM, naive Bayes, and decision tree by evaluating on a dataset of 10,000 articles from the Vnexpress newsletters. The experimental results show an accuracy of approximately 92%, which is capable of comparative analysis to the state-of-the-art results but the model was trained on small dataset of 10,000 articles having maximum 10 classes so the model might not generate promising results on large dataset belonging to different knowledge fields.

2. CNN, a feedforward network with multiple convolutional layers along with pooling continues increasing but in contrast many researchers are proposing more efficient models like Lai et.al. [2] proposed a new method for text classification using recurrent convolutional neural networks (RCNNs), which applied a recurrent network aimed to capture information in terms of context instead of traditional capturing of keywords. This approach uses RNNs that is good at modeling sequential data, while CNNs are good at extracting features from data.

3. CNN models rapidly gained popularity in the natural language processing (NLP) domain for text classification but sometimes only scrutinization doesn't yield satisfactory results so Xuesong Tong [3] came up with a novel approach of integrating two models, negative elements removal (NER) module to remove negative elements from the complaint text and a character-level CNN to extract features from the data and to classify data into respective classes. The proposed model was trained on Consumer Finance Complaints and it shows an accuracy of 91% by classifying Consumer Complaints into 3 classes but as it was trained for only 3 classes so it might not perform well for the system having more classes/ domains.

4. Tran Thanh Dien, Nguyen Thanh-Hai [4] proposed a deep learning method for automatic topic classification implemented for an online submission system. The system is designed to classify documents into one of a predefined classes/topic. The authors evaluated their approach on five different datasets (including English, Turkish, and Vietnamese) and the largest data set contained around 10,000 documents and 10 different topics/classes. The proposed deep learning approach consists of two stages: the first one is feature extraction and the 2nd stage is feature classification. In the first stage, the authors use a convolutional neural network (CNN) to extract features from the document text. The CNN is trained on a large corpus of text that has been manually labeled with topics. In the classification stage, long short-term memory (LSTM) network is trained on the features extracted by the CNN to classify the document into one of the predefined topics. The authors achieved an accuracy around 95%. Authors' approach is a scalable but it is possible that the approach would not perform well on a larger dataset and on different topic taxonomy from the one used to train the model.

5. China is paying more attention to non-technical literacy of individuals such as perspective on life, and the notion of sustainable development specially in university education. In the

past recent years, they evaluate individuals by conducting interviews or questionnaires that often contains long text from students. These paragraphs, often contain unstructured and long text that gets hard to classify by classifiers so Miao Zang [5] proposed a multi-label classification model called MLformer that uses Longformer to do mainly two tasks. It extracts features to get global semantic and context of the text that makes it easy to classify with 80% accuracy on the dataset about engineering sustainability that is in Chinese originally. The experimental results shows that the model outperforms other conventional deep learning models like BERT. But the major drawback of this model is its small dataset. It was trained only on 763 samples, in which only 20% of the data contained long text (of 500 to 900 Chinese characters) so it will not show promising result on new testing data.

6. As population is increasing day by day, the increase in the use of transportation services demands a system that is capable of improving quality of service by analyzing railway complaints by passengers. This conducted research work proposes a deep learning model for railway complaint categorization. Aforementioned model by [6] uses a bidirectional long short-term memory (LSTM) network with an attention mechanism for feature extraction from the text. The attention mechanism allows the model to focus on the most important words in a complaint gathered from COMS application of Indian Railways and classify it in one of the 14 classes. This resulted in improved accuracy upto 93% along with F1-score of 0.93 but the major drawback of the proposed model includes the use of a small dataset and the lack of evaluation on new/updated real-world data.

7. Technology plays significant role in the times of crisis like Singh, M. [7] used sentiment analysis to track the public's response to the pandemic on Twitter using BERT model that achieved an accuracy of 93%. In the conducted research work sentimental analysis on COVID-19 is presented. The mentioned strategy has used five metrices for performance evaluation that are Average Likes over the period, Intensity Analysis, Average Re-tweets over the period, Wordcloud, Polarity, and Subjectivity. The seven intensity categories he used to analyze the tweets are neutral to strongly negative. The pandemic has had a profound impact on social life. Their result shows that the sentiment of tweets about the coronavirus was generally negative, with the mean sentiment score being -0.03. However, the sentiment of tweets from India was more positive than the sentiment of tweets from the rest of the world. And this trend varied over time. His work showed how sentiment analysis can be used to identify areas where public health interventions are needed. But the author

acknowledged the limitations to their study that the tweets were collected from Twitter, which is a platform that is used by a relatively small percentage of the population.

8. The exponential expansion of biomedical literature, particularly in the context of the COVID-19 pandemic, presents a substantial obstacle to manual curation and interpretation. This is primarily due to the accelerated pace at which new research findings are being published and accepted. LitCovid is a comprehensive database of scholarly papers pertaining to COVID-19, hosted within the PubMed platform. This database currently encompasses a vast collection of over 100,000 articles, and witnesses millions of user accesses on a monthly basis from individuals across the globe. During this particular challenge, Qingyu Chen [8] introduced a novel approach, known as LitMC-BERT, for the multi-label classification of biomedical literature. The classification of papers into various labels, such as Diagnosis, Prevention, Treatment, etc., is achieved through the utilization of a transformer-based model. This model is designed to learn and comprehend the associations between different labels, enabling accurate classification. The model demonstrated a level of accuracy amounting to 84% during the performance evaluation conducted on a separate sample of 3000 articles in the LitCovid database.

9. The use of technology is always been in trend and a rapid growth can be seen in the field of NLP specifically tasks related to classification or prediction to make systems more capable and efficient but these machine learning models offer no clarification for the predictions they made. Existing models by [9] tend to focus on predicting the output or make prediction on the basis of the relation between input and output. However, these approaches work like a black box so Hui Liu came up with a novel approach of generative explanation framework called GEF that tends to classify text and generate fine-grained human-readable explanation for the prediction the model made. Hui Liu used LSTM and CNN as base models for classification and integrate it with GEF for generating explanations for the decision taken. The author additionally joined CVAE+GEF to generate even better explanations. The testing is performed on two datasets. the models achieved an improved accuracy that surpasses all the baseline models on these datasets by generating precise and accurate explanation.

10. Vasudeva Raju S [10] proposed a novel approach for topic modelling by using on BERT-based embeddings. The data is obtained from Consumer Financial Protection Bureau

(CFPB) data. BERT, or Bidirectional Encoder Representations from Transformers, uses BERT-based embeddings that has ability of capturing the semantic relationships between words in a way that is not possible with traditional topic modelling methods. They evaluated their approach on a dataset of CFPB complaints.

11. The paper by [11] proposed a novel approach that utilizes the hybrid deep learning model for the automatic classification of citizen e-petitions. The model generated by using aforementioned technique combines the strengths of convolutional neural networks (CNNs) and bidirectional long short-term memory (Bi-LSTMs) to extract both local and sequential features from the text of the petitions. a dataset of e-petitions is used to train the model. the data set is obtained from the Chinese bulletin board system (BBS). The results of the experiments depicts that the proposed model achieved a weighted F1 score equal to figure 0.8267. aforementioned figure is higher than the F1 scores of the baseline models. The paper's findings suggest that hybrid deep learning models and decision support systems can be used to improve the efficiency and make the method of e-petition classification more effective in future.

12. In this paper by [12], Machine learning techniques were used to analyze customer reviews about the restaurant management, food and beverage. The review holds information in text form but Machine learning use numerical data. For that it applies NLP and preprocessed the data after that it apply different vectorization technique which provide information in numerical format. This article applies different classifiers, train the model and after that test the model and check the accuracy. The main feature of this article gives comparison chart of accuracy by different classifiers as well as with different vectorization techniques.

13. Business enterprises are currently reconsidering the prospect of transitioning their operations to the online realm. In doing so, they are also reassessing various strategies that may prove advantageous for their business endeavors. These strategies encompass the identification and prioritization of customer concerns, the analysis of customer satisfaction, and the enhancement of customer service. In order to address this objective, Zhuyi Rao [13] introduced a methodology rooted in data mining principles, which entails employing a fusion of text mining and machine learning methodologies, including Bayesian network for feature extraction and K-mean clustering for categorization. The methodology was assessed using a dataset of user complaints sourced from GitHub, on Chinese language

data. The evaluation results demonstrated that the method attained a level of accuracy of 90%. One limitation of this model is its effectiveness in classifying user complaints that are written in a specific language, because it was trained on a relatively smaller dataset.

14. In continuation, Qurat-ul-Ain [14] proposed NLP-based model such as Logistic Regression, Random Forest, K-nearest neighbor and SVM using TF-IDF for the classification of complaints. The models used same approach, a combination of natural language processing and machine learning techniques to extract features from the text of the complaints first and then classify them into different categories. Model achieved an accuracy that lies between 80-83% for dataset 1 and 72-80% for local dataset.

Author Name & Publish Date	Dataset	Classifiers	Accuracy
Tran Thanh Dien, Bui Huu Loc,Nguyen Thai-Nghe [1] (2019)	Scientific Articles, VNEXPRESS NEWSLETTERS	SVM	90.1%
		Naive Bayes	87.6%
		KNN	46.7%
Siwei Lai, Liheng Xu, Kang Liu, Jun Zhao [2] (2015)	20Newsgroups Fudan set ACL Anthology Network Stanford Sentiment Treebank	RCNN	96.49
			95.20
			49.19
			47.21
Xuesong Tong, Bin Wu, Shuyang Wang, and Jinna Lv [3] (2018)	Consumer Finance Complaints (English), Consumer Express Complaints (Chinese)	Character-level Convolutional Network	91.0 7
			86.85
Tran Thanh Dien, Nguyen Thanh-Hai [4] (2020)	Scientific_Articles Turkish_News_Article VnExpress_Newsletters	MLP	0.977
		SVM	0.965
		Decision Tree	0.819
Miao Zang [5] (2023)	Dataset in Chinese on 'Engineering Sustainability'	MLformer	0.80

Meenu Gupta [5] (2021)	Data from COMS app of Indian Railways	LSTM	93%
Mrityunjay Singh &· Amit Kumar Jakhar [6] (2021)	COVID-19 Tweets selected from Twitter, using Twitter scraper and the tweepy APIs for extracting tweets & data scraping	BERT	93%
Qingyu Chen [7] (2022)	LitCovid BioCreative HoC	LitMC-BERT	0.8022
		LitMC-BERT	0.6854
Hui Liu [8] (2019)	Skytrax User Review dataset	CNN+GEF	7 9.07%
		LSTM+GEF	77.9%
Vasudeva Raju S [9] (2022)	Dataset from Consumer Financial Protection Bureau (CFPB)	FinBERT	C-V: 0.3327 U-Mass: -12.652
Fengmei Sun [10] (2022)	Shanghai e-petition BBS.	BERT	82%

Anuradha Tutika, M Y V Nagesh [11] (2019)	superdatascience.com	3 techniques are used for these models that are Countvectorizer, TFIDF, Hashing vectorizer respectively	
		K-NN	1. 75% 2. 78% 3. 62%
		Logistic Regression	1. 80% 2. 88% 3. 68%
		SVM	1. 70% 2. 68% 3. 69%
Zhuyi Rao, Yunxiang Zhang [12] (2022)	Chinese natural language processing data set from GitHub	K-mean	90%
Qurat-ul-ain [13] (2022)	Dataset I: Consumer Complaint from Kaggle Dataset II: Local complaints dataset	**Models for dataset I**	**Count Vector**
		Random forest	83%
		Support Vector Machines	83%
		Logistic Regression	82%
		K--Nearest Neighbor classifier	78%

3.1. Research Gaps

In recent years, there are different types of features extraction techniques and selection of model utilized by researchers which are explained in this section. Count Vector and TF-IDF features rely on number of word occurrences. Features extracted by embeddings like Word2vec, Bag-of-words provide information regarding

occurrence matrices. Likewise, Researchers have applied multiple machine learning algorithms and hybrid deep learning models for better understanding of text. But from the prior work, it is clear that a significant research gap still exists due to the limited work done in the field of complaints classification, Thus, this scarcity in literature makes it more challenging to develop a comprehensive and generalized system for complaints classification.

Hence, the contributions of this paper can be summarized as:

- An in-depth investigation of the feature techniques, word embeddings and models that can be used for complaints classification.

- It addresses the significance of contextual understanding in text, highlighting the shortcomings of traditional models in extracting the context-based meanings in complaints, which can often be vague or ambiguous.

- A system is proposed to enhance the classification of consumer complaints. By integrating the capabilities of CNN and Bidirectional LSTM, and introducing BERT model, the system emphasis on the significance of contextual understanding in text and ensures accurate routing of complaints to their respective departments, promoting rapid and effective response measures

3.2. Data Sets

The primary objective of this study is to assess the performance of the proposed deep learning models and find the best approach for our problem by testing models on multiple datasets. For this purpose, experiments have been conducted using identical settings on three datasets: Consumer complaints dataset I, Consumer complaints dataset II and Local dataset III.

3.2.1. Dataset I:

Dataset I used in this work, is the US Consumer Finance Complaints dataset available on Kaggle. This dataset is a collection of 555957 consumer complaints about financial products and services. It was created by the Consumer Financial Protection Bureau (CFPB). The data is classified into eleven categories, each representing a unique class such as credit card, retail banking, credit card, and credit reporting, mortgage, or debt collection etc. in the bank department. The key features in the dataset are described in the following graphs:

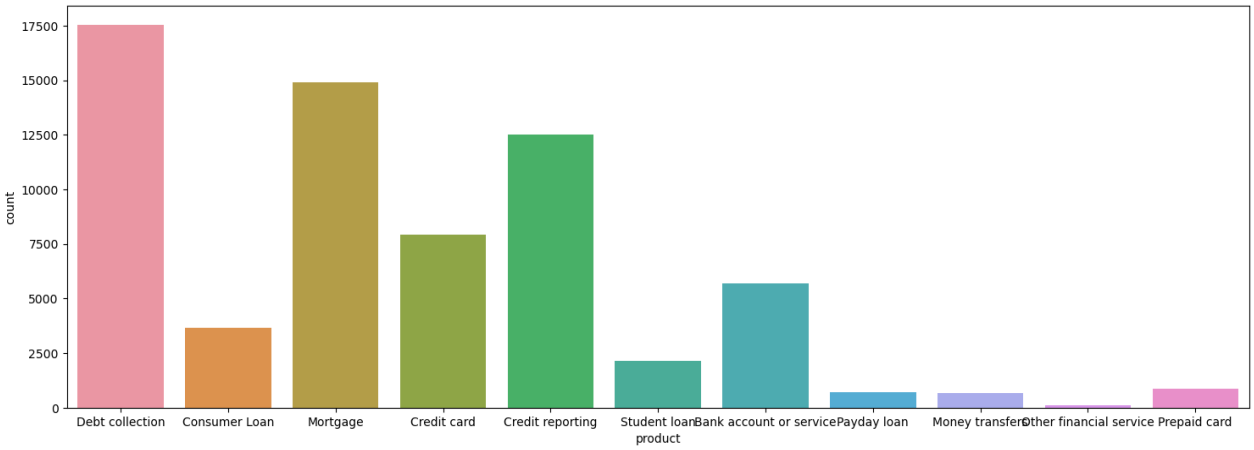


Fig 3.1: Number of Samples in Each Class

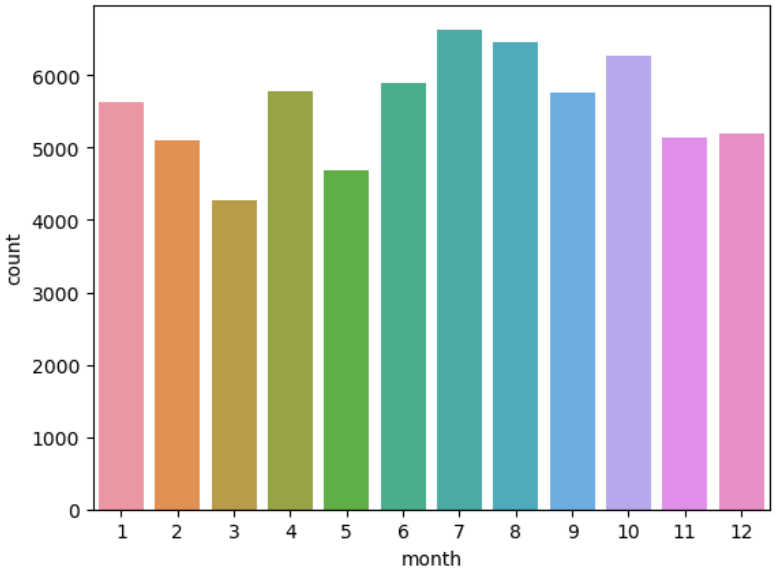


Fig 3.2: Received complaints on the basis of months

3.2.2. Dataset II:

Dataset II used in this work, is the dataset with the name “Consumer Complaints” available on Kaggle and GitHub. This dataset is a collection of 1,62,421 consumer complaints about financial products and services. It contains information about the complaints filed by the users. The data is classified into five categories, each representing a unique class such as credit card, retail banking, credit reporting, mortgage, and debt collection. The distribution of data samples is given below in Figure 3.3.

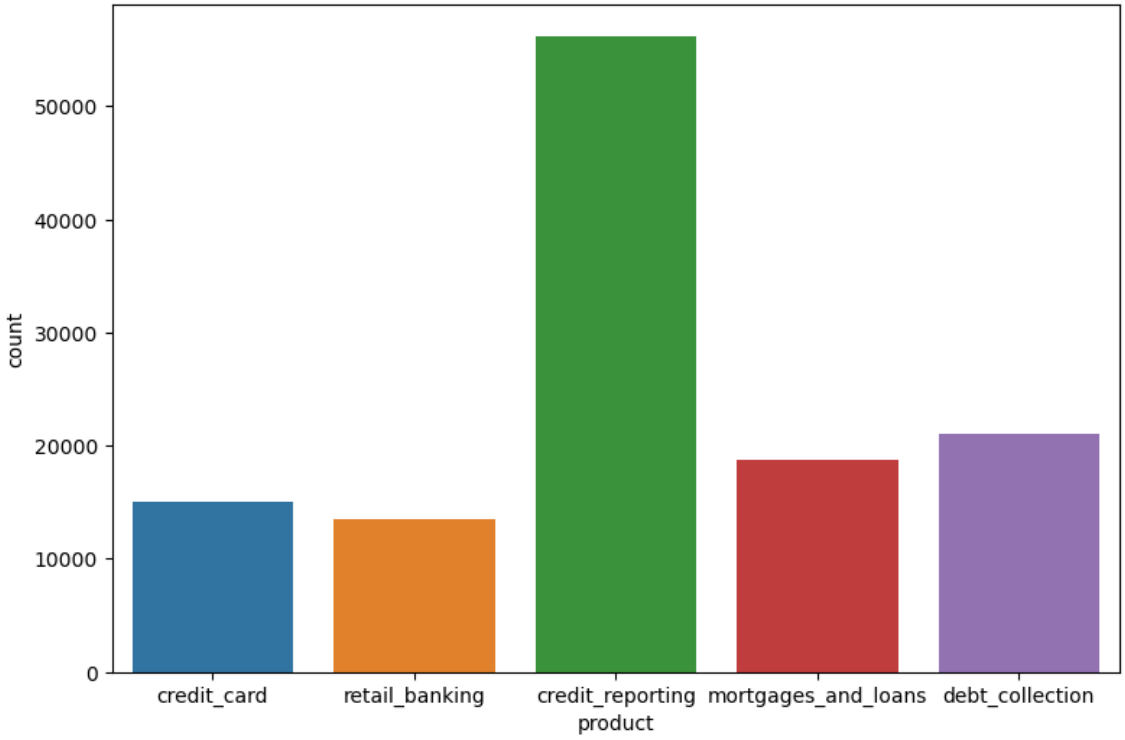


Fig 3.3: Number of Samples in each Class

	narrative	product
0	purchase order day shipping amount receive pro...	credit_card
1	forwarded message date tue subject please inve...	credit_card
2	forwarded message cc sent friday pdt subject f...	retail_banking
3	payment history missing credit report speciali...	credit_reporting
4	payment history missing credit report made mis...	credit_reporting

Fig 3.4: First Four Samples & respective Class

3.2.3. Dataset III:

Dataset III refers to a local dataset that has been personally gathered and cleaned by using initial preprocessing. This Local Data set contain 10 distinct classes, and each class shows a different domain such as Scholarships, Academics, Attestation, Accreditation, Sports, Information & Technology, Equivalency of different degrees and domains, Quality Assurance Division, Quality Assurance Agency, and Research and Development. Each sample in the dataset belongs to one of the respective classes. The dataset comprises 10,002 complaints that were gathered from the organization's official website. It is worth noting that some complaints might overlap across multiple classes/divisions, as they show various instances of similar issues being reported. The classes are converted to labels from 1-10 as shown in the Figure 3.5 below:

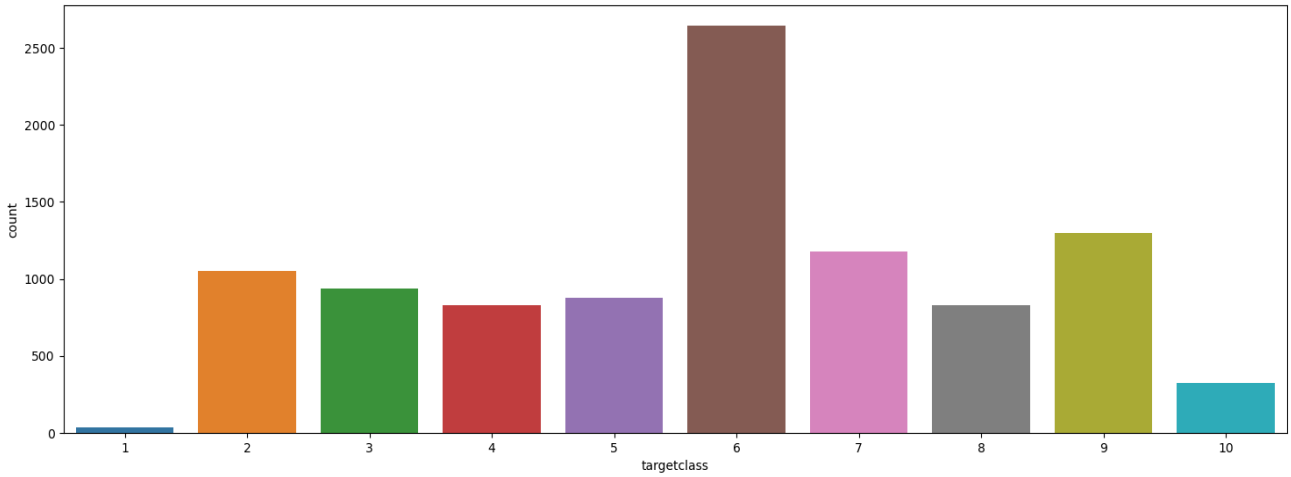


Fig 3.5: Number of Samples in each Class

	complaints	targetclass
0	need to upload my profile in HEC PCD List as D...	1
1	My name is Zarrin Basharat and I have done my ...	1
2	I have a friend named Basharat Ali from Skardu...	1
3	I am working as Assistant Manager in PESCO. Th...	1
4	Dear sir, Please apprise about ranking status...	1

Fig 3.6: First Four Samples & Respective Class

CHAPTER 4: METHODOLOGY

In the proposed system, three distinct models have been implemented. Text samples are subjected to preprocessing techniques. Subsequently, GloVe embeddings and BERT Tokenizer have been utilized for feature extraction. BERT tokenizer is employed additionally only for BERT model to extract features because BERT model works on a specific data format prepared by its tokenizer. These processed feature vectors are then passed into three distinct models. The first model is a Convolutional Neural Network (CNN), specifically designed to recognize local patterns in the extracted features. The second model is a hybrid of BiLSTM-CNN, integrating both temporal and spatial analysis. Finally, the third model is Bidirectional Encoder Representations from Transformer (BERT). This state-of-the-art model uses the features extracted by BERT tokenizer and channels these feature vectors to BERT model for deeper contextual understanding. The framework of our proposed methodology is shown in the Figure 4.1

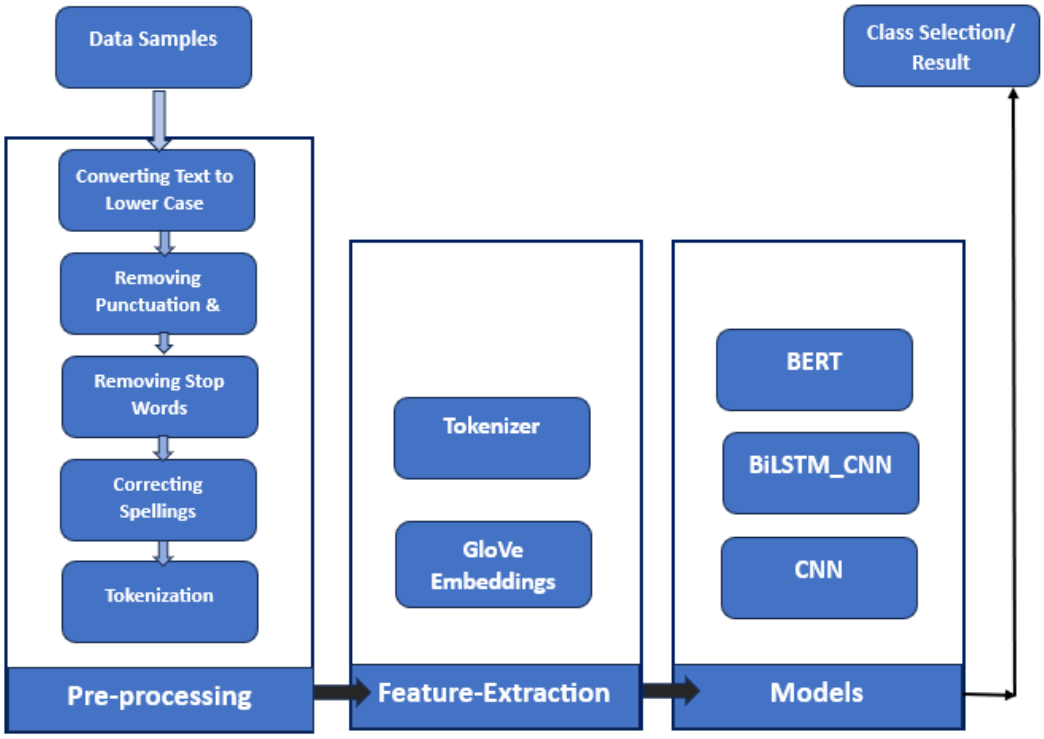


Fig 4.1: Framework of our Proposed Architecture

4.1. Initial Preprocessing

Removing duplicates can help maintain a clean and organized dataset, ensuring that each data entry is unique and contributes to accurate analysis and results. However, the data undergoes an extensive cleaning process. The original dataset might contain repeated occurrences like similar complaints. Duplicate instances are defined as those with identical department and complaint descriptions, even though the date or time may differ.

The decision to remove or keep these duplicates is contentious. On one hand, removing duplicates helps prevent overfitting during the model training process. This is particularly true when the training set contains many similar instances. For less frequent events, the model can properly account for the recurrence of the same incident. On the other hand, high duplicate counts reflect that certain type of complaints occur more frequently. To optimize performance, duplicates were removed, but it should be kept in mind that it's uncertain whether this is the best approach.

This issue of duplicate handling needs further research, but it falls outside the scope of this study.

4.2. Pre-processing data

Preprocessing aids in cleaning the raw text data by removing irrelevant characters, special symbols, and unwanted formatting artifacts. This step minimizes noise, ensuring the subsequent NLP models are fed with high-quality and consistent data. The techniques used to preprocess the data are mentioned below:

- Converting text to lower case.
- Removing punctuation and unwanted symbols and characters.
- Removing stop words.
- Correcting spellings
- Lemmatizing
- Tokenization

4.2.1. Conversion of Text to Lower Case.

In the first step, we applied the process of converting all text to lowercase. This preprocessing step involved changing all the letters in the text to their corresponding lowercase forms. The conversion to lowercase aids in creating a standardized and normalized text corpus, minimizing variations due to letter case and preventing the duplication of words based on their capitalization.

As a result, the NLP model can learn and recognize patterns from the language more accurately, leading to improved performance in various NLP tasks. Conversion to lower text of the data samples is shown in the Figure 4.2

Preprocessed Text
xxxx has claimed i owe them 2700 for xxxx years despite the proof of payment i sent them canceled check and their ownpaid invoice for 2700 they continue to insist i owe them and collection agencies are after me how can i stop this harassment for a bill i already paid four years ago
due to inconsistencies in the amount owed that i was told by m t bank and the amount that was reported to the credit reporting agencies

Fig 4.2: Samples after converting to lower text

4.2.2.Removing Stop Words and Punctuation

The removal of stop words and punctuation marks is an important preprocessing step in natural language processing (NLP) to enhance the quality and efficiency of text analysis while maintaining the original meaning of the text. Stop words are common words that occur frequently in a language but typically carry little semantic meaning (e.g., "the," "is," "and"). Punctuation marks include characters like commas, periods, question marks, and exclamation points.

However, we carefully cleaned the dataset by getting rid of unnecessary punctuation and symbols like [!"#$%&'()*+,-./:;<=>?@[]^_`{|}~], which are commonly found in text datasets. Figure 4.3 shows the text sample after removing stopwords and punctuation marks.

Preprocessed Text
xxxx has claimed i owe them 2700 for xxxx years despite the proof of payment i sent them canceled check and their ownpaid invoice for 2700 they continue to insist i owe them and collection agencies are after me how can i stop this harassment for a bill i already paid four years ago
due to inconsistencies in the amount owed that i was told by m t bank and the amount that was reported to the credit reporting agencies i was advised to write a good will letter in order to address the issue and request the negative entry be removed from my credit report all together

Fig 4.3: Samples after removing stop words & punctuation

4.2.3.Normalization

The text is being normalized in the next step as the objective is to normalize the words in a given text based on three dictionaries sourced from online to improve consistency and accuracy in text processing. These dictionaries, presumably loaded from the text files, contain mappings of non-standard words to their normalized versions. This step replaces any non-standard word it encounters with its normalized counterpart according to these dictionaries as shown in the Figure 4.4.

Preprocessed Text
has claimed i owe them 2700 for years despite the proof of payment i sent them canceled check and their ownpaid invoice for 2700 they continue to insist i owe them and collection agencies are after me how can i stop this harassment for a bill i already paid four years ago
due to inconsistencies in the amount owed that i was told by am t bank and the amount that was reported to the credit reporting agencies i was advised to write a good will letter in order to address the issue and request the negative entry be removed from my credit report all together

Fig 4.4: Samples after normalization

4.2.4.Correcting Spelling

Correcting spelling in Natural Language Processing (NLP) is a common task, that identifies words that have been misspelled and correct them with the most suitable candidate. In our proposed methodology, Dictionary Lookup method has been used. In this method, every word in the text is compared against a dictionary of correctly spelled words. If a word is not found in the dictionary, it is considered misspelled. This step uses TextBlob library, which is equipped

with robust Natural Language Processing (NLP) functionalities, to correct spelling mistakes in the textual data. Specifically, each word is traversed in the data sample column. The resulting spelling-corrected text replaces the original narrative in the Data Frame. This preprocessing step, significantly enhances the quality of text data by reducing spelling inconsistencies, thereby paving the way for more accurate subsequent analyses.

4.2.5. Lemmatization

Lemmatization, often used for normalizing words in a text corpus. It is a more complex process, does a better job than stemming at using the context of the word in a sentence and its Part of Speech (POS) to determine the base or 'lemma' of the word. It accurately transforms words to their canonical form. For example, it converts 'are' to 'be', and 'better' to 'good'. This makes it especially useful when the exact form of the word is crucial for further analysis. This step uses TextBlob's Word class to lemmatize the consumer complaint. The lambda function splits each narrative into individual words, lemmatizes each word, then joins them back together into a single string. Samples received after lemmatization step are mentioned in the Figure 4.5 below:

Preprocessed Text
claimed owe 2700 year despite proof payment sent canceled check ownpaid invoice 2700 continue insist owe collection agency stop harassment bill already paid four year ago
due inconsistency amount owed told bank amount reported credit reporting agency advised write good letter order address issue request negative entry removed credit report together

Fig 4.5: Samples after Lemmatization

4.2.6 Tokenization

In this preprocessing step, we tokenized the dataset by breaking each statement down into individual words. The NLTK package has been employed to perform this task by segmenting the text into meaningful units, making it more manageable and suitable for analysis. Tokenization is an important step that enhances the quality and efficiency of NLP tasks, as it allows the NLP model to understand and process language more effectively. By breaking the text into smaller units, we enable the model to capture meaningful information and relationships between words, which are used for accurate analysis and prediction. In our case,

a tokenizer, from the Keras library, is used to tokenize text data. This process returns a dictionary where the keys are the unique words found in the data, and the values are the corresponding indices. Basically, this dictionary is used to convert words into their corresponding numeric representations to prepare text data for a deep learning model. The tokenized dataset is shown in the Figure 4.5 below.

Preprocessed Text
{'credit': 1, 'account': 2, 'payment': 3, 'loan': 4, 'would': 5, 'bank': 6, 'time': 7, 'report': 8, 'debt': 9, 'told': 10, 'call': 11, 'received': 12, 'card': 13, 'mortgage': 14, 'company': 15, 'information': 16, 'called': 17, 'day': 18, 'month': 19, 'letter': 20, 'year': 21, 'pay': 22, 'never': 23, 'get': 24, 'sent': 25, 'paid': 26, 'number': 27, 'back': 28, 'said': 29, 'could': 30**,**

Fig 4.6: Samples after Tokenization

In our data samples, 50956 unique words are found as given below

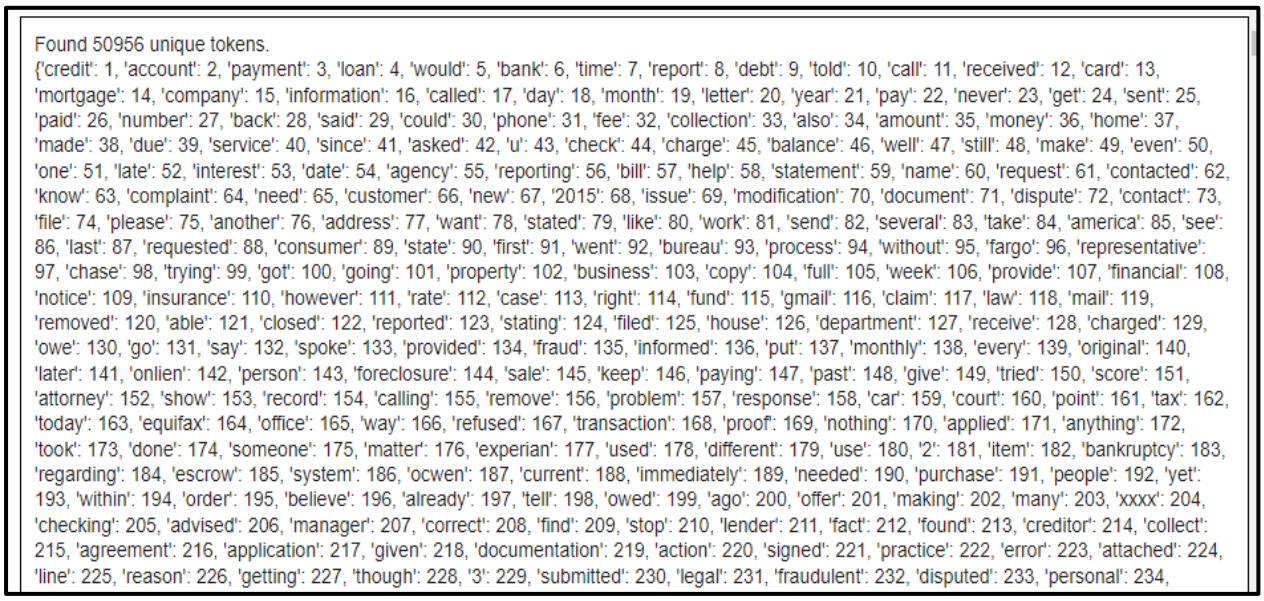


Fig 4.6: Total Samples after Tokenization

4.3. Feature Extraction

After identifying the sentence with the most words or the longest length, and converting the sentence to maximum length that is 394 in one case by either doing padding or truncating, word embeddings are loaded from a pre-trained GloVe (Global Vectors for Word Representation) model. The model is trained on a massive dataset containing one billion tokens (words) and utilizes a vocabulary of 400 thousand distinct words. The GloVe (Global Vectors for Word

Representation) algorithm is used for training, resulting in embedding vector sizes of 50, 100, 200, and 300 dimensions.

GloVe is a popular word embedding technique similar to Word2Vec, and it has been used to convert text data into numerical vectors which can be easily processed by deep learning models. Word embeddings transformed words into vectors by using 'glove.6B.300d.txt' file that is a pre-trained word vectors file from the GloVe (Global Vectors for Word Representation) project, developed by the Stanford NLP Group. The '300d' refers to the dimensionality of the vectors. Each word is represented by a 300-dimensional vector, implying that each word is described by 300 features in the embedding space. These vectors capture semantic and syntactic information about the words, and words with similar meanings tend to have similar vectors. These vectors are then fed as input to the model. A word embedded vector representation for a word token 'loan' and index value 4 in our embedding matrix is given below in Figure 4.7.

```
[('loan', 4)]
```

```
embedding_matrix[4]  #embedded vector for token 'loan'

array([ 0.13474999,  0.063568  , -0.37950999, -0.080729  ,  0.34176001,
       -0.0053481 ,  0.80001003, -0.78824002, -0.47262999, -0.89402002,
        0.086908  , -0.051773  ,  0.57349998, -0.26681   , -0.0043554 ,
       -0.68673003,  0.54759002, -0.47711   ,  0.12997   , -0.50748003,
        0.073666  , -0.56199998,  0.19243   , -0.022735  ,  0.15757   ,
        0.68008   , -0.48374999,  0.14399   , -0.69022   ,  0.26741001,
       -0.53082001, -0.29096001,  0.32907999,  0.12313   , -1.14779997,
       -0.51828998, -0.018956  ,  0.02077   , -0.0015803 , -0.053114  ,
       -0.10982   , -0.83578998, -0.46337   ,  0.85992002,  0.57225001,
       -0.33202001, -0.23357999,  0.80937999, -0.43586999,  0.35385001,
        0.0055405 ,  0.068909  ,  0.13897   ,  0.16237999,  0.038382  ,
       -0.16306999, -0.022701  ,  0.14324   , -0.25878   , -0.47661999,
        0.25588   , -0.23389   , -0.14936   , -0.51688999,  0.44329   ,
        0.49015   ,  0.25725001, -0.45041999,  0.66949999, -0.32949001,
        0.34626999,  0.91883999,  0.29245001,  0.20112   ,  0.19018   ,
        0.17321999, -0.3028    ,  0.34022999,  0.49851999, -0.66885   ,
        0.046698  , -0.19625001, -0.18028   , -0.41620001,  0.25986001,
       -0.078103  , -0.33655   ,  0.2077    , -0.69224   ,  0.13703001,
       -0.078832  ,  0.26159   , -0.81893998, -0.35289001,  0.93794   ,
       -0.1078    , -0.0944    , -0.11647   ,  0.16785   , -0.32293001,
        0.26176   , -0.37233001,  0.20868   ,  0.040312  , -0.10222   ,
        0.03121   , -0.09023   ,  0.10476   , -0.042395  ,  0.43564999,
        0.18483   , -0.37215999, -0.21328001, -0.38079   ,  0.39579999,
       -0.32354999,  0.36971   ,  0.036001  ,  0.27676001, -0.19016001,
```

Fig 4.7: Vector Embedding for Word Loan

Now these embedded vectors of same length (CNN needs input samples of same length) are ready to fed to the CNN input layer for classification.

4.4. Data Splitting for model training

Following the extraction of relevant features, data is partitioned into two subsets that is 80/20 split for this model; 80% of the dataset was utilized for training the model, while the remaining 20% was reserved for model testing. The distribution of data points across each class, is shown in the Figure 4.8.

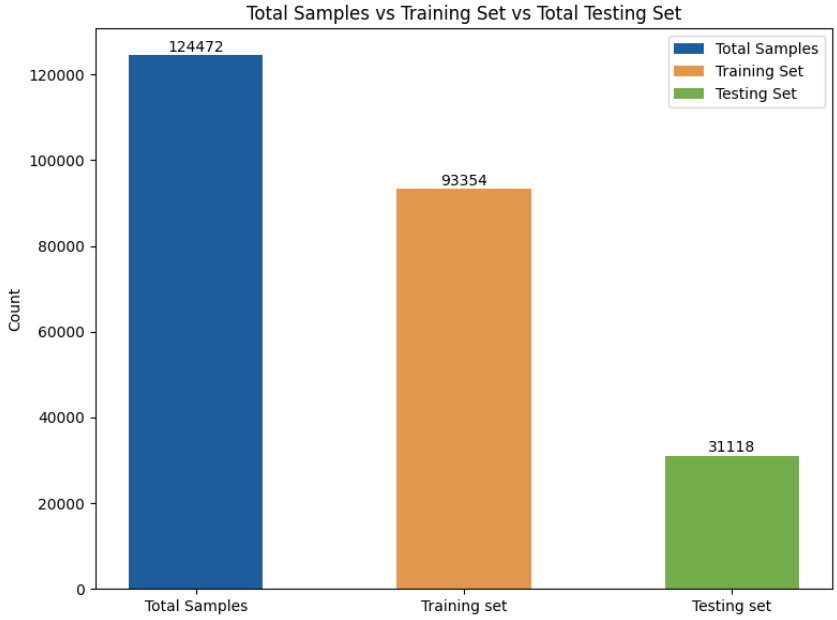


Fig 4.8: Training & Testing Subset

4.5. Classification Model

Initially, the dataset is divided into different subsets. Training and Testing dataset. Our method employs a supervised learning approach where the model is trained using pre-labeled data. Consequently, the model learns and assimilates the features from the training data. When presented with new, unseen data, the deep learning model makes predictions based on the knowledge it has previously acquired.

4.5.1. Convolutional Neural Network (CNN) Model

The model begins with an Embedding layer. Our proposed methodology used GloVe to pre-train embedding to initialize the weights of input embedding layer. GloVe is used to convert the input words (indexed by 'word_index') into dense vectors of a fixed size

(specified by EMBEDDING_DIM that is 300 in our case). This layer's weights are initialized with the pre-existing 'embedding matrix' that is set of vector array, and the 'trainable' parameter is set to True, allowing the layer's weights to be updated during training. The first layer, an Embedding layer, transforms the input into dense vectors of 300 dimensions. This layer can process input sequences of length 394, yielding an output shape of (None, 394, 300).

A Dropout layer follows the Embedding layer, dropping out 30% of the nodes to prevent overfitting. This is followed by a Conv1D layer with 128 filters and a kernel size of 5, employing a ReLU (Rectified Linear Unit) activation function which results in the output shape being (None, 390, 128). Then, a MaxPooling1D layer is applied with pool size of 5, further reducing the spatial dimensions of the output from the Conv1D layer in shape being (None, 78, 128).

Another Dropout layer is used, followed by a BatchNormalization layer, which helps in faster convergence of the model by maintaining the mean activation close to 0 and the activation standard deviation close to 1.

The process of Conv1D, MaxPooling1D, Dropout, and BatchNormalization is repeated to allow the model to learn more complex features. After these layers, the model applies a Flatten layer to convert the 2D matrix data into a vector that can be used as input to the Dense layers. This sequence of Conv1D and MaxPooling1D layers is implemented, again with 128 filters, a kernel size of 5, and a pool size of 5. This results in an output shape of (None, 74, 128) for the Conv1D layer and (None, 14, 128) for the MaxPooling1D layer. After another Dropout and BatchNormalization, the output shape remains (None, 14, 128).

A Flatten layer follows these, converting the 2D tensor into a 1D tensor, changing the output shape to (None, 1792). Subsequently, a Dense layer with 128 units is used, adjusting the output shape to (None, 128). This dense layer is applied using the ReLU activation function, which serves to learn the non-linear combinations of the features.

The final layer of the model is another Dense layer with 11 units, corresponding to the number of output classes. This layer uses the softmax activation function to output a probability distribution over the 11 classes.

Finally, the model is compiled with the 'rmsprop' optimizer and the 'categorical_crossentropy' loss function, which is suitable for multiclass classification. The metric used for evaluation is accuracy. This comprehensive structure allows the model to effectively classify consumer complaints into their respective categories.

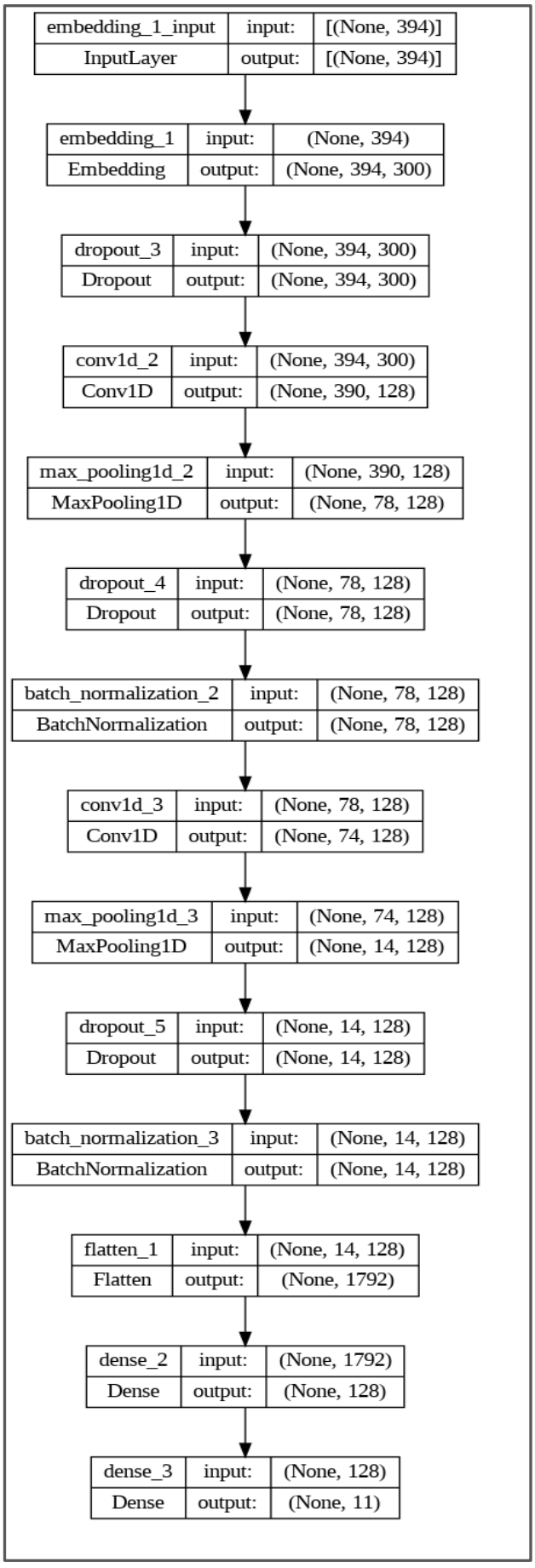
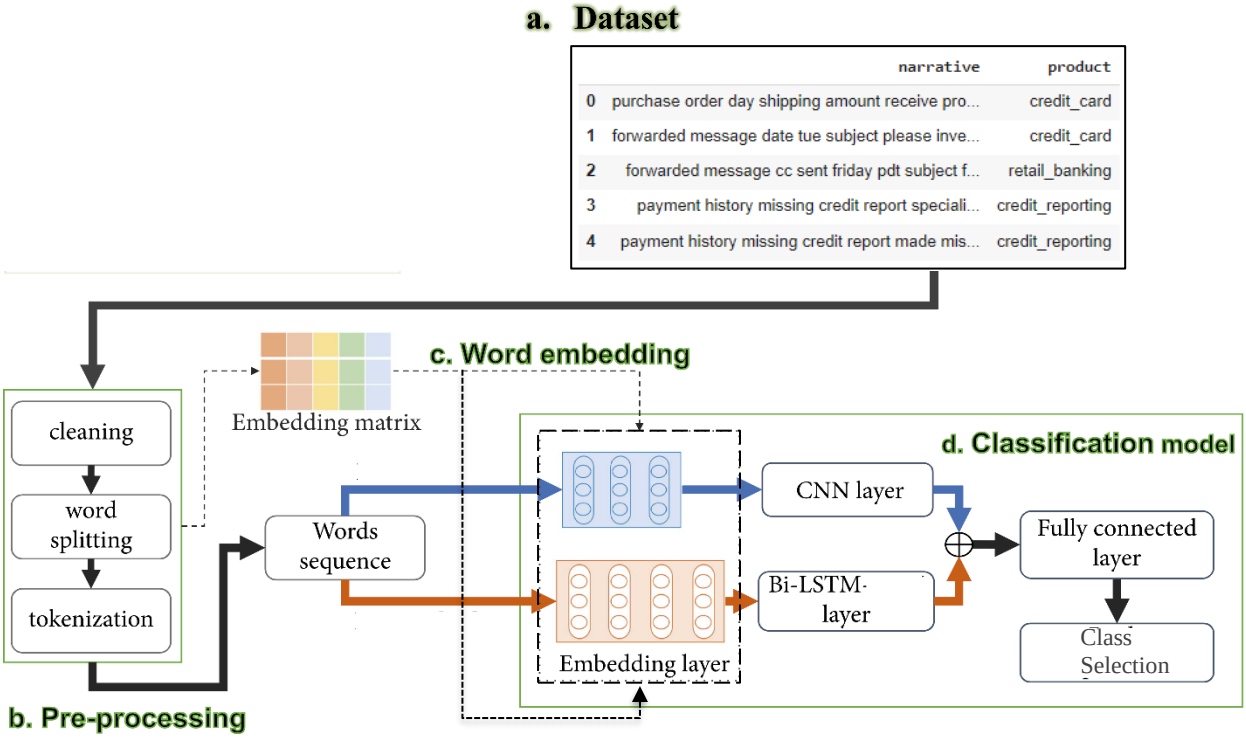


Fig 4.9: Layers of our CNN Model

4.5.2. Hybrid BiLSTM-CNN Model

Our hybrid Convolutional Neural Network (CNN) and Bidirectional Long Short-Term Memory (BiLSTM)is a model for categorizing consumer complaints in natural language processing (NLP) is constructed using a Sequential model from the Keras libra ry.



Our hybrid Convolutional Neural Network (CNN) and Bidirectional Long Short-Term Memory (BiLSTM) model begins with an Input layer, designed to accommodate sequences of fixed length 394.

The first layer, called the `Input` layer, is where our model begins. It receives input sequences each of length 394. These sequences can be understood as sentences where each word is represented by a unique integer, which is an index into a vocabulary.

Subsequently, an `Embedding` layer is incorporated to transform each integer in the sequence into a dense vector of fixed size, which we specify as `EMBEDDING_DIM`. Here, the initial weights of the `Embedding` layer are set to a pre-trained `embedding_matrix`, but we allow these weights to be updated during training (`trainable=True`).

The model then branches into two paths. The first path consists of a set of CNN layers that are structured to recognize local features in the input data. The `Conv1D` layers apply 128 filters, each spanning a window of 5 words, and use a ReLU activation function. To prevent overfitting, `Dropout` layers randomly set a fraction of the input units to 0 during training, and `BatchNormalization` layers are used to normalize the activations of the previous layer. `MaxPooling1D` layers reduce the temporal dimension by taking the maximum value over 5-word windows. After two such CNN blocks, a `Flatten` layer is used to transform the 3D output to 2D.

In parallel to the CNN branch, the input sequences are also processed by a BiLSTM layer in the second path. The `Bidirectional` LSTM layer with 100 units is capable of capturing the temporal dependencies in both forward and backward directions in the sequence, making it especially suitable for understanding the context in our data. A `Dropout` layer is also used in this branch to mitigate overfitting.

The outputs of both branches are then merged together using a `Concatenate` layer. This layer combines the feature representations learned from both the CNN and BiLSTM branches, providing a comprehensive feature set for the subsequent layers.

Following this, we have a `Dense` layer with 128 units and ReLU activation that performs a transformation of the input data into a 128-dimensional space. Finally, a `Dense` layer with 10 units and softmax activation function computes the probabilities for each of the 10 classes.

The model is compiled with the RMSprop optimizer and the categorical cross-entropy loss function, typically used for multi-class classification problems. Then, the model is trained using mini-batch gradient descent with a batch size of 64 for a total of 11 epochs.

This model presents a powerful approach to sequence classification tasks, leveraging the ability of CNNs to recognize local patterns and the capability of BiLSTMs to understand broader context. By combining these strengths, the model can understand both the individual semantic meanings of the words and their contextual relations in a sequence.

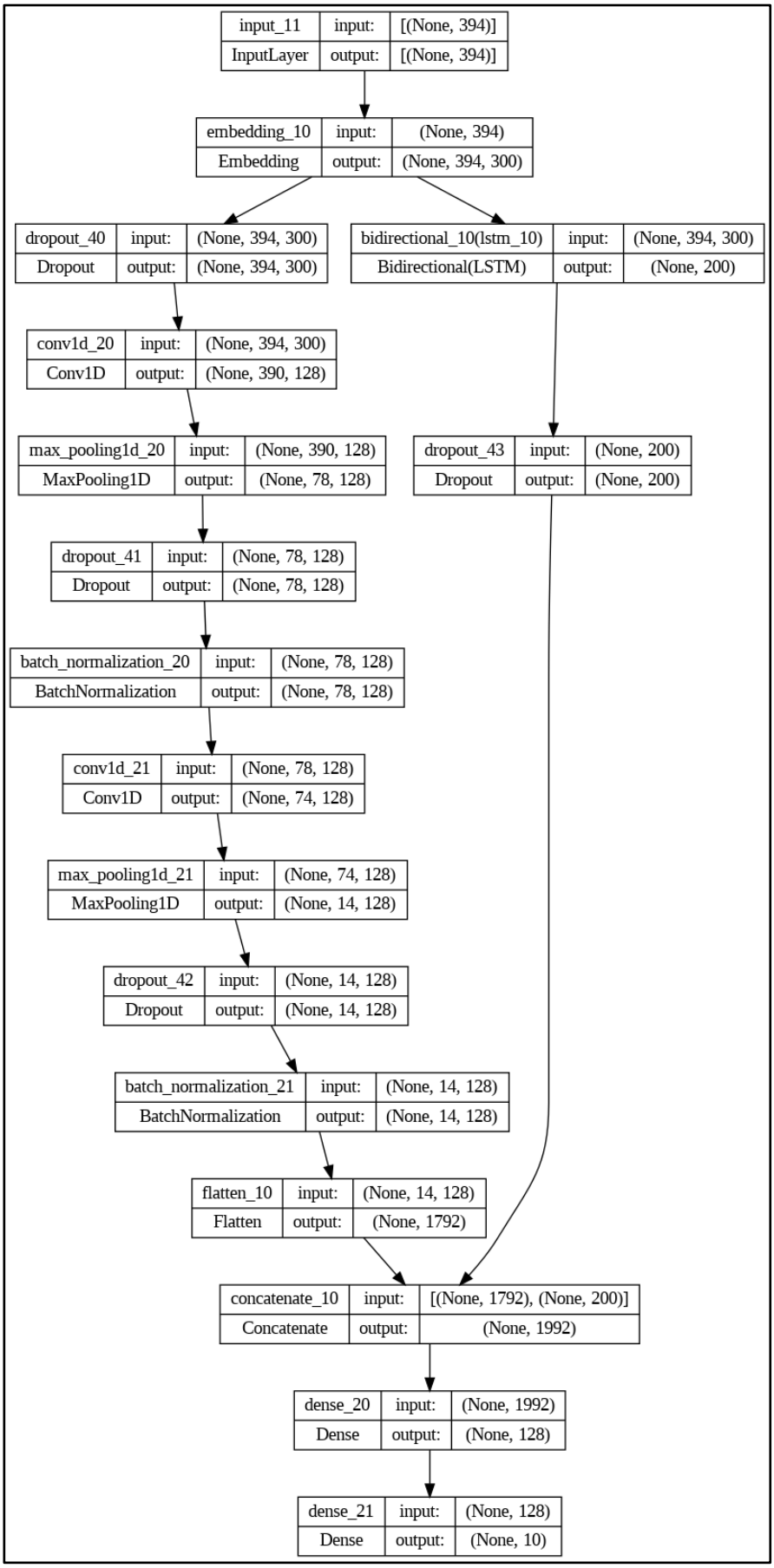


Fig 4.10: Layers of Hybrid Model

4.5.3. Model Evaluation

After training model, we evaluated model by using 10-fold cross-validation. This approach decreases the risk of model overfitting by partitioning the dataset randomly into ten subsets or "folds." Therefore, examining how the model's performance varies with the volume of training data is a practical way to fine tune the model. In this context, learning curves become invaluable tools. By observing how the model behaves on increased training data, we can fine tune and maximize model's performance. Although, further insights into the model's performance are obtained using a classification report, as elaborated in Section 5. This report provides precision, recall, and F1-score. These metrics collectively offer a comprehensive picture of the model's performance, assisting us in fine-tuning the model for better results.

4.5.4. BERT Model

Hugging Face Transformers library, an important tool in NLP, relies on TensorFlow or PyTorch for model loading. The significant advantage of the Hugging Face Transformers library is its ability to leverage pre-trained language models, eliminating the need for extensive and expensive computational resources. Most models are available directly in the library in both PyTorch and TensorFlow. This makes it function almost like an API, providing an efficient, easily tunable.

Our proposed model includes Hugging Face Transformers library with PyTorch and encompasses following steps:

1. **Dataset Loading:** This involves ingesting the raw data required for the machine learning process.
2. **Data Preprocessing:** Bert preprocessing is different from the traditional approaches used in other deep learning models. In this step, the raw data is cleaned and transformed into a suitable format called Tensor that can be used for model training.
3. **Utilizing BERT Pre-Trained Model and Tokenizer**: This step involves loading the necessary pre-trained model, such as BERT, and the corresponding tokenizer for text manipulation.
4. **Training and Evaluation**: The preprocessed data is then used to train the model, after which its performance is evaluated
5. **Prediction**: Finally, the trained model is employed to make predictions on unseen data.

4.5.4.1 Loading the Dataset II

	product	narrative
0	credit_card	purchase order day shipping amount receive pro...
1	credit_card	forwarded message date tue subject please inve...
2	retail_banking	forwarded message cc sent friday pdt subject f...
3	credit_reporting	payment history missing credit report speciali...
4	credit_reporting	payment history missing credit report made mis...
5	credit_reporting	payment history missing credit report made mis...
6	credit_reporting	va date complaint experian credit bureau invol...
7	credit_reporting	account reported abbreviated name full name se...
8	credit_reporting	account reported abbreviated name full name se...
9	credit_reporting	usdoexxxx account reported abbreviated name fu...

Fig 4.11: Loading Dataset II

4.5.4.2 Data Converting Class Labels into Categorical Data

```
dict = {'credit_card' : 0, 'credit_reporting' : 1, 'debt_collection' : 2, 'mortgages_and_loans': 3, 'retail_banking': 4}

df = df.replace({'product': dict})
```

4.5.4.3 Data Splitting for Model Training

Following the extraction of relevant features, data is partitioned into two subsets that is 90/10 split for this model; 90% of the dataset was utilized for training the model, while the remaining 10% was reserved for model testing. The distribution of data points across each class, is shown in the Figure 4.12

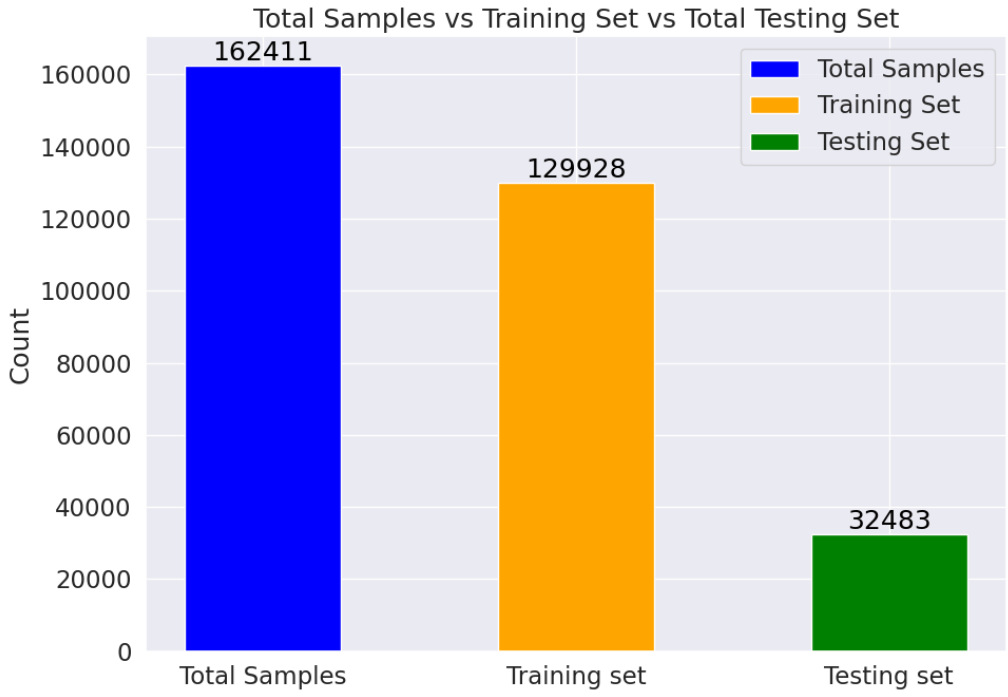


Fig 4.12: Total Samples in Training & Testing Subset

Figure 4.13 shows the total samples in training & testing subset class-wise

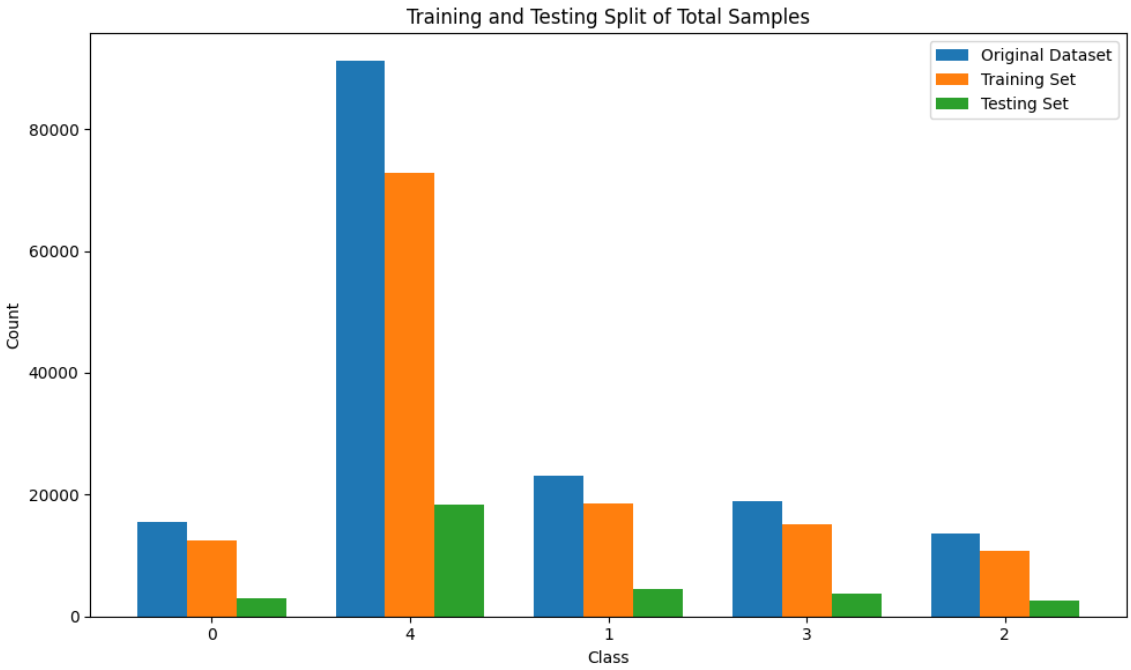


Fig 4.13: Total Training & Testing Samples in each Class

Input Data Modeling

4.5.4.4 WordPiece Tokenization

This breaks down words into known subwords, or into individual characters if the word is not in the tokenizer's vocabulary. In the context of BERT (Bidirectional Encoder Representations from Transformers), a particular kind of tokenization is employed, known as WordPiece tokenization.

In traditional tokenization, a sentence or piece of text is split into individual words. However, WordPiece tokenization takes this a step further by breaking words into subwords or even single characters if the words are not in its vocabulary. This makes BERT robust in handling out-of-vocabulary words and allows it to understand the semantic meaning of new words based on their subwords. For instance, a word like "unhappiness" may be broken down into the tokens: ["un", "##happiness"], where "##" indicates that "happiness" is a part of the original word.

Text sample after tokenization is shown in the Figure 4.14 below

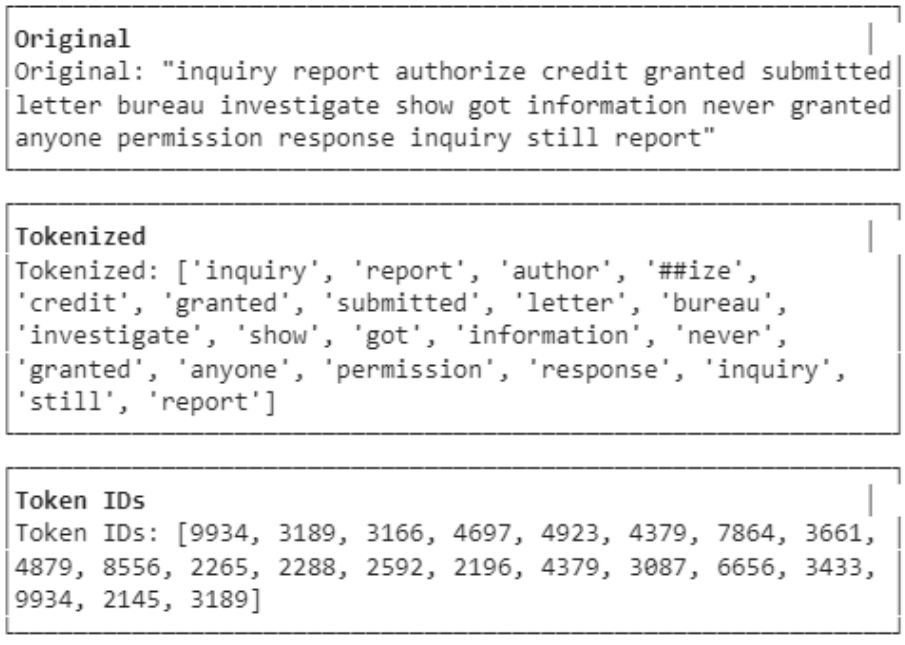


Fig 4.14: Sample after WordPiece Tokenization

4.5.4.5 Addition of Special Token, Padding and Truncation

BERT requires special tokens to be added to the input. The '[CLS]' token is added at the beginning of every sentence, and the '[SEP]' token is added at the end. If dealing with two

sentences, '[SEP]' is also added between them. BERT requires input with the same length so sentences are padded or truncated to make the sentences of the same length. Masking is also done to prevent the model from attending to the padding tokens. Masking is placing 1 for original words and 0 for padded labels.

```
After Tokenization, Padding and Truncating all sentences to 512

Padding token: "[PAD]", ID: 0

Done.
┌──────────────────────────────────────────────────────────┐
│Original: "inquiry report authorize credit granted submitted│
│letter bureau investigate show got information never granted│
│anyone permission response inquiry still report"          │
└──────────────────────────────────────────────────────────┘
Token IDs: [ 101 9934 3189 3166 4697 4923 4379 7864 3661 4879 8556 2265 2288 2592
 2196 4379 3087 6656 3433 9934 2145 3189  102    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
```

Fig 4.15: Sample after Addition of Special Token, Padding and Truncation

4.5.4.6 Masked Values

In pre-training, input sequences are typically padded to a fixed length to create batches of equal size. Padding is necessary because transformer-based models process input data in parallel, and all sequences in a batch must have the same length. To pad sequences, a special padding token is used, which can be represented by a numerical value, such as 0. So, masking replaces 1 for original word and 0 for padded sequences to distinguish.

```
Number of attention masks: 129928
Length of each attention mask: 512


Masked Values (Sentence 3): [1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0]
```

Fig 4.16: Sentence 3 after Masking

4.5.4.7 Conversion to Input IDs and Tensors:

Each token is then mapped to a unique ID from the tokenizer's vocabulary. And then sequences of input IDs are converted into tensors. This is necessary because both TensorFlow and PyTorch models operate on tensors. Tensors can have three or more dimensions, making them useful for representing complex data structures. The ability to work with tensors allows these libraries to perform complex computations efficiently and on a variety of hardware, from CPUs to GPUs.

```
Input IDs: tensor([[  101,  3728,  8182,  ...,     0,     0,     0],
        [  101,  7316,  4070,  ...,     0,     0,     0],
        [  101,  2741,  2195,  ...,     0,     0,     0],
        ...,
        [  101,  6406,  7593,  ...,     0,     0,     0],
        [  101,  2988, 27118,  ...,     0,     0,     0],
        [  101,  2972,  3291,  ...,  2655,  2067,  2733]])
Attention Masks: tensor([[1, 1, 1,  ..., 0, 0, 0],
        [1, 1, 1,  ..., 0, 0, 0],
        [1, 1, 1,  ..., 0, 0, 0],
        ...,
        [1, 1, 1,  ..., 0, 0, 0],
        [1, 1, 1,  ..., 0, 0, 0],
        [1, 1, 1,  ..., 1, 1, 1]])
Labels: tensor([1, 1, 2, 3, 0, 1, 1, 1, 1, 1, 1, 4, 2, 1, 1, 1])
```

Fig 4.17: Creation of Tensor

4.5.4.8 Model Building

Pre-trained BERT is loaded by using the Hugging Face's Transformers library. A pre-trained BERT model has already been trained on a large corpus of text data. The benefit of using a pre-trained model like BERT is that you can leverage this understanding for your specific NLP task without having to train a model from scratch, which can save significant computational resources and time.

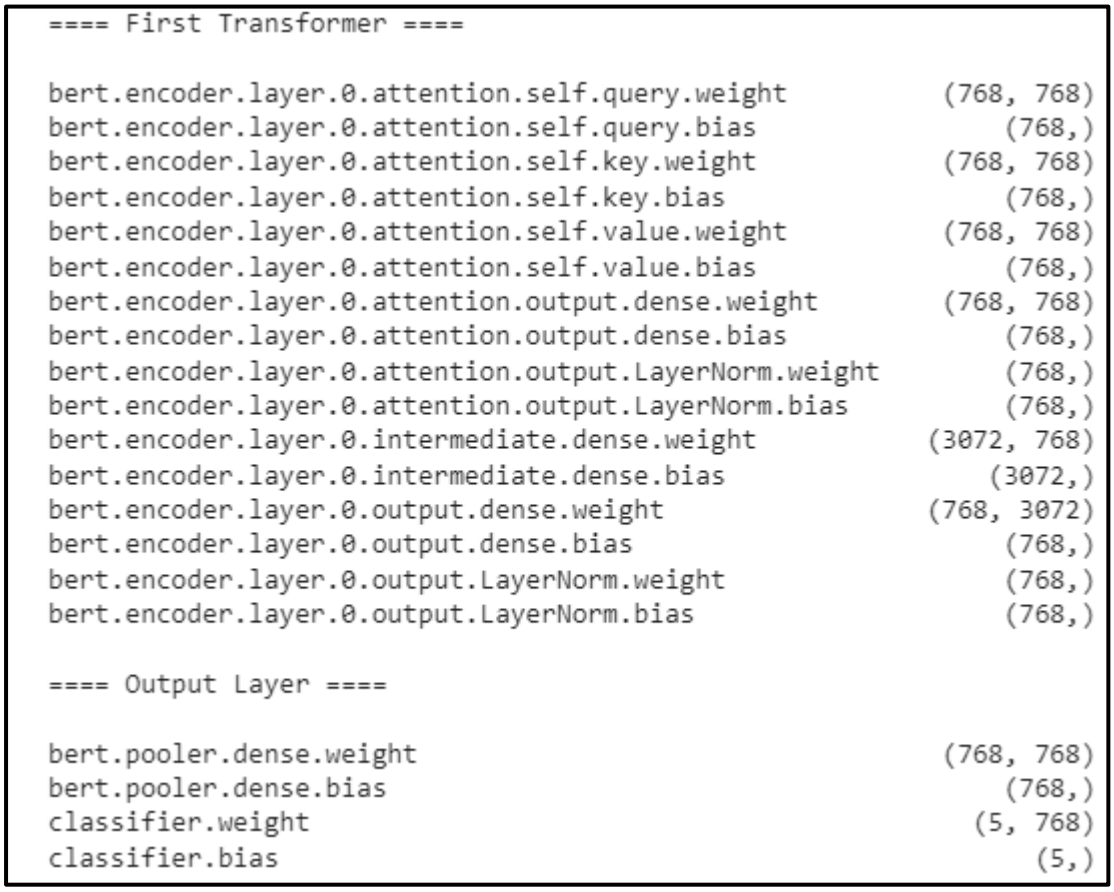
```
==== Embedding Layer ====

bert.embeddings.word_embeddings.weight                  (30522, 768)
bert.embeddings.position_embeddings.weight                (512, 768)
bert.embeddings.token_type_embeddings.weight                (2, 768)
bert.embeddings.LayerNorm.weight                            (768,)
bert.embeddings.LayerNorm.bias                              (768,)

==== First Transformer ====

bert.encoder.layer.0.attention.self.query.weight          (768, 768)
bert.encoder.layer.0.attention.self.query.bias                (768,)
bert.encoder.layer.0.attention.self.key.weight            (768, 768)
bert.encoder.layer.0.attention.self.key.bias                  (768,)
bert.encoder.layer.0.attention.self.value.weight          (768, 768)
bert.encoder.layer.0.attention.self.value.bias                (768,)
bert.encoder.layer.0.attention.output.dense.weight        (768, 768)
bert.encoder.layer.0.attention.output.dense.bias              (768,)
bert.encoder.layer.0.attention.output.LayerNorm.weight        (768,)
bert.encoder.layer.0.attention.output.LayerNorm.bias          (768,)
bert.encoder.layer.0.intermediate.dense.weight           (3072, 768)
bert.encoder.layer.0.intermediate.dense.bias                (3072,)
bert.encoder.layer.0.output.dense.weight                 (768, 3072)
bert.encoder.layer.0.output.dense.bias                        (768,)
bert.encoder.layer.0.output.LayerNorm.weight                  (768,)
bert.encoder.layer.0.output.LayerNorm.bias                    (768,)
```

Fig 4.18: BERT Layers



```
==== First Transformer ====

bert.encoder.layer.0.attention.self.query.weight          (768, 768)
bert.encoder.layer.0.attention.self.query.bias                (768,)
bert.encoder.layer.0.attention.self.key.weight            (768, 768)
bert.encoder.layer.0.attention.self.key.bias                  (768,)
bert.encoder.layer.0.attention.self.value.weight          (768, 768)
bert.encoder.layer.0.attention.self.value.bias                (768,)
bert.encoder.layer.0.attention.output.dense.weight        (768, 768)
bert.encoder.layer.0.attention.output.dense.bias              (768,)
bert.encoder.layer.0.attention.output.LayerNorm.weight        (768,)
bert.encoder.layer.0.attention.output.LayerNorm.bias          (768,)
bert.encoder.layer.0.intermediate.dense.weight           (3072, 768)
bert.encoder.layer.0.intermediate.dense.bias                (3072,)
bert.encoder.layer.0.output.dense.weight                 (768, 3072)
bert.encoder.layer.0.output.dense.bias                        (768,)
bert.encoder.layer.0.output.LayerNorm.weight                  (768,)
bert.encoder.layer.0.output.LayerNorm.bias                    (768,)

==== Output Layer ====

bert.pooler.dense.weight                                  (768, 768)
bert.pooler.dense.bias                                        (768,)
classifier.weight                                           (5, 768)
classifier.bias                                                 (5,)
```

Fig 4.18: BERT Layers

4.5.4.9 Model Compilation

Our model employed AdamW optimizer to facilitate the training of our deep learning neural network. The AdamW optimizer, an extension of the Adam optimizer, was chosen for its ability to effectively handle weight decay, also known as L2 regularization, within its adaptive learning rate framework. By initializing the optimizer with a learning rate (lr) of 2e-5 and an epsilon value (eps) of 1e-8 to ensure numerical stability, we ensured a controlled and stable training process. During training iterations, the model's parameters were updated based on the gradients computed through backpropagation. This approach was vital for achieving optimal convergence and minimizing overfitting while learning from the given dataset.

4.5.4.10 Model Training

Model training is a core step in the machine learning process. It's when the model "learns" patterns from the data to make accurate predictions or decisions. Even though BERT is already trained on a large corpus of text, it needs to adjust to the specificities of our dataset and task. This adjustment is done during the training phase. The workflow of the training process includes following steps:

1. **Data Input**: The BERT model is fed with input data, that is tokenized text converted into tensors.
2. **Forward Pass**: The tensors pass through the model, and the model makes predictions based on its current state (defined by the weights and biases it learned during pre-training).
3. **Loss Calculation**: A loss function calculates the difference between the model's predictions and the actual values (the correct answers). This function quantifies how well (or poorly) the model is performing.
4. **Backpropagation**: This is a process by which the model learns from its errors. The gradient of the loss function is calculated concerning the model's parameters, and this information is propagated back through the model. This provides insight into which parameters should be adjusted to reduce the loss.
5. **Weight Update:** The model's weights are updated using an optimization algorithm, typically something like stochastic gradient descent (SGD) or one of its variants (like Adam). In our proposed model Adam has been used. The weights are adjusted in the direction that minimally reduces the loss.
6. **Iteration**: Steps 2-5 are repeated for a certain number of iterations or epochs (complete passes through the dataset) until the model's performance on the data is satisfactory.

After training, the model's performance is evaluated on a separate validation dataset to ensure that it has generalized well to unseen data and has not caused overfitting.

4.5.4.11 Model Evaluation

Model evaluation is the process of determining how well your model is performing, not on the training data it learned from, but on unseen data. This step allows us to assess the model's ability to generalize, that is, its effectiveness at handling data it has not encountered during the training phase. The ultimate goal is to create a model that performs well not just on the training data but on unseen data as well, as this indicates the model will likely perform effectively in real-world scenarios.

CHAPTER 5: EXPERIMENTAL RESULTS

The primary goal of this study is to assess the performance of the proposed deep learning models and find the best approach for our problem by testing models on multiple datasets. For this purpose, experiments have been conducted using identical settings on three datasets: Local dataset I, Consumer complaints dataset II and Consumer complaints dataset III.

5.1 CNN

To train CNN model, data has been split into two subsets: 80% and 20%. The model has been trained on 80% of the data, and the remaining 20% is reserved for testing the model's performance. After the models have been trained on the extracted features from the 80% training set, they are then tested on the 20% testing set. This allows the evaluation of the models' ability to generalize their learning to new, unseen data. After applying the GloVe embeddings for feature extraction and employing CNN model, we acquired following results.

5.1.1. Datasets used:

Dataset I, dataset II, and dataset III comprises of 555957 samples, 1,62,421 samples, and 10002 samples respectively. Each sample belong to one of the distinct classes. For training and testing, these samples are split into two subsets: 80% & 20%.

5.1.2. Classification Report

A classification report provides valuable metrics such as precision, recall, f1-score, and the number of instances in each class after the model is implemented. This tool allows us to evaluate the overall accuracy of the model. Figure 5.1 shows the classification reports obtained from testing set.

In case of Figure 5.1(a) that is report of dataset I, the model showcased strong ability in distinguishing classes such as 'Prepaid card', 'Bank account or service', 'Credit card', 'Credit reporting', and 'Mortgage' display particularly high F1-scores, ranging between 0.80 to 0.94. This suggests that the model is well-tuned for these classes and can distinguish between them effectively. The model struggled to correctly classify 'Payday loan' with both precision and recall being 0. This indicates a complete inability of the model to recognize this class, potentially due to the limited data available for this category (with a support of only 27). Another class, 'Money transfers', although having higher metrics than 'Payday loan', still showed a relatively low F1-score of 0.49. Overall, model is indicating commendable performance with certain areas showing lack of data samples of certain class. Similar behavior is seen for dataset III but due to better distribution of samples in dataset II, model performed

well by presenting F1-scores, ranging between 0.83 to 0.86.

	precision	recall	f1-score	support
Debt collection	0.90	0.70	0.79	1428
Consumer Loan	0.77	0.66	0.71	920
Mortgage	0.86	0.76	0.81	1982
Credit card	0.84	0.89	0.86	3132
Credit reporting	0.86	0.86	0.86	4388
Student loan	0.76	0.42	0.54	166
Bank account or service	0.95	0.92	0.94	3730
Payday loan	0.00	0.00	0.00	27
Money transfers	0.47	0.32	0.38	182
Other financial service	0.83	0.66	0.74	215
Prepaid card	0.91	0.83	0.87	532
micro avg	0.87	0.83	0.85	16702
macro avg	0.74	0.64	0.68	16702
weighted avg	0.87	0.83	0.85	16702
samples avg	0.83	0.83	0.83	16702

(a)CNN on Data I

	precision	recall	f1-score	support
1	1.00	0.38	0.55	8
8	0.85	0.79	0.82	263
3	0.73	0.50	0.59	235
7	0.51	0.85	0.64	207
9	0.68	0.75	0.72	219
4	0.86	0.87	0.87	662
5	0.78	0.67	0.72	295
2	0.79	0.72	0.76	206
10	0.70	0.69	0.70	325
6	0.76	0.74	0.75	81
accuracy			0.75	2501
macro avg	0.77	0.70	0.71	2501
weighted avg	0.77	0.75	0.75	2501

(b) CNN on Data III

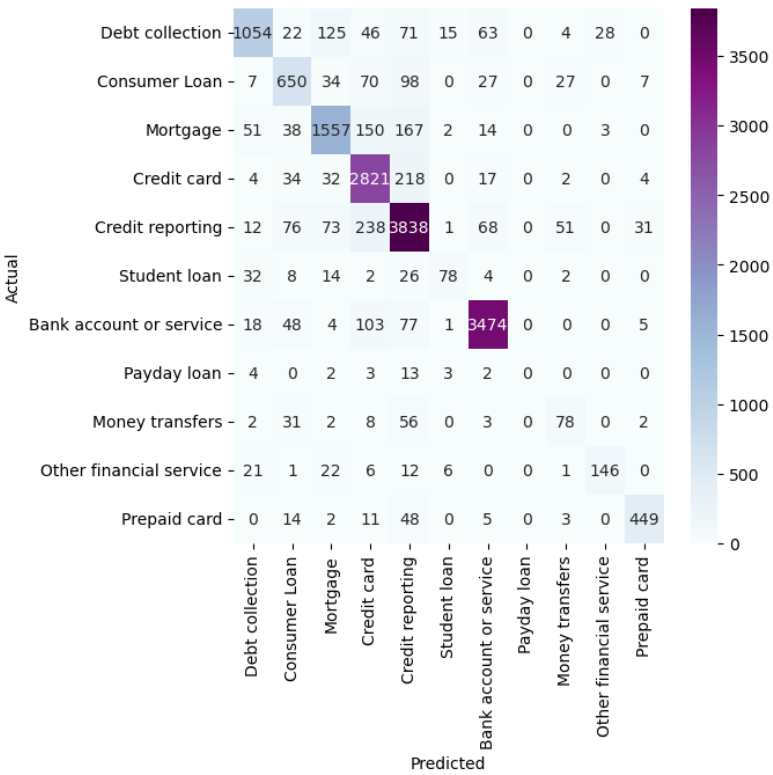
	precision	recall	f1-score	support
credit_card	0.86	0.71	0.78	3746
retail_banking	0.85	0.93	0.89	14060
credit_reporting	0.85	0.72	0.78	5264
mortgages_and_loans	0.90	0.79	0.84	4681
debt_collection	0.90	0.81	0.85	3367
micro avg	0.86	0.83	0.85	31118
macro avg	0.87	0.79	0.83	31118
weighted avg	0.86	0.83	0.84	31118
samples avg	0.83	0.83	0.83	31118

(c) CNN on Data II

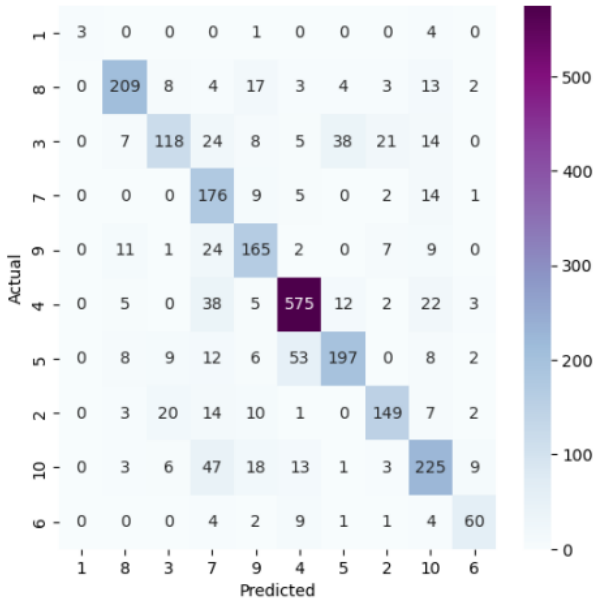
Fig 5.1: Classification reports of dataset I, dataset II, & Local dataset III

5.1.5. Confusion Matrix

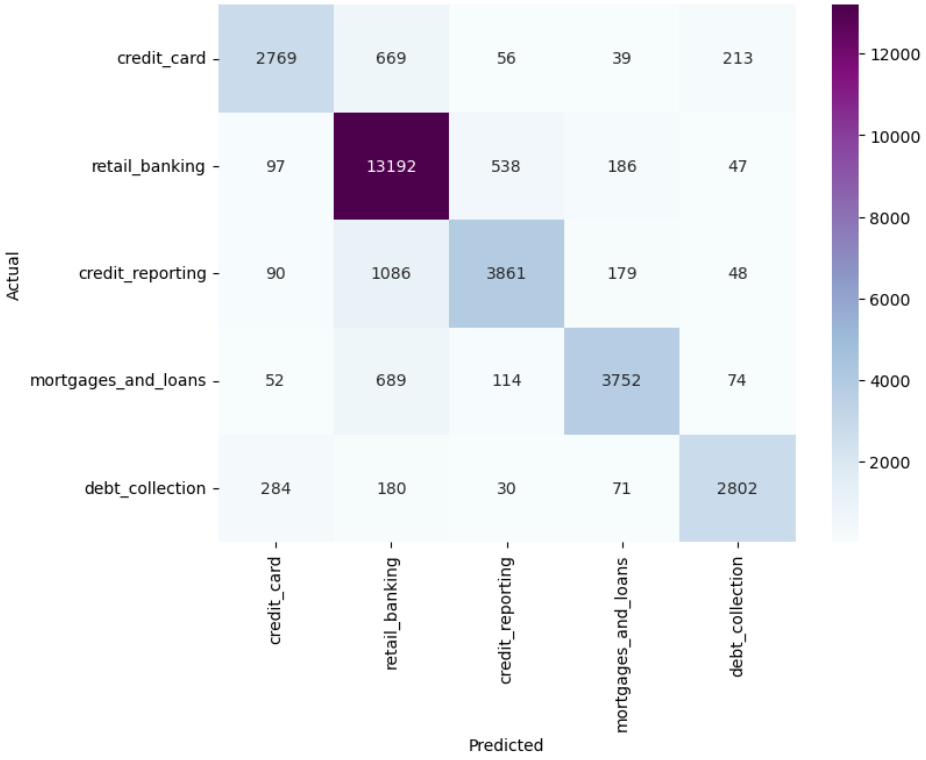
The confusion matrix assists in gaining a more precise understanding of each class data, along with evaluation of a classification model's performance. This matrix compares the true value with the value predicted by proposed model. The diagonal elements of the matrix represent the true positive values, indicating correct classifications, while the off-diagonal elements depict misclassifications. Confusion matrices given below shows the performance of CNN model, where columns represent true classes, and rows depict predicted classes.



(a)Confusion matrix of Data I



(b) Confusion matrix of Data III

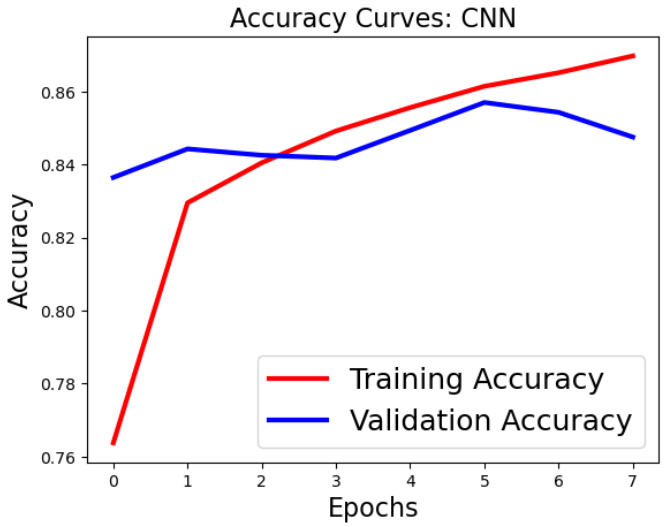


(c) Confusion matrix of Data II

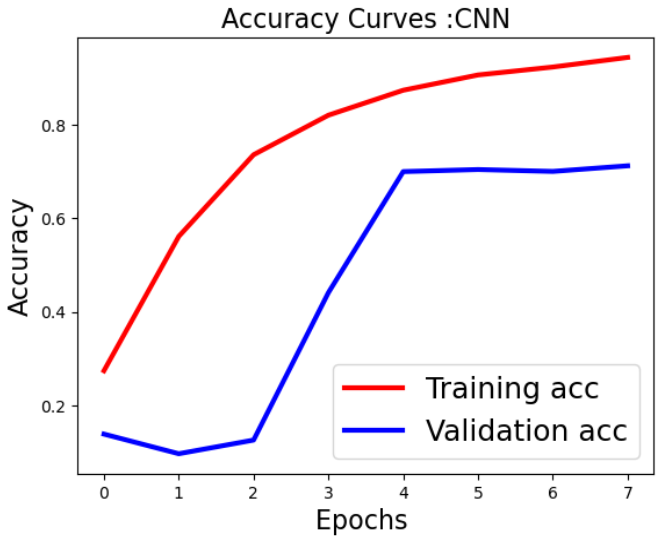
Fig 5.2: Confusion Matrix of dataset I, dataset II, & Local dataset III

5.1.6. Training & Validation Accuracy

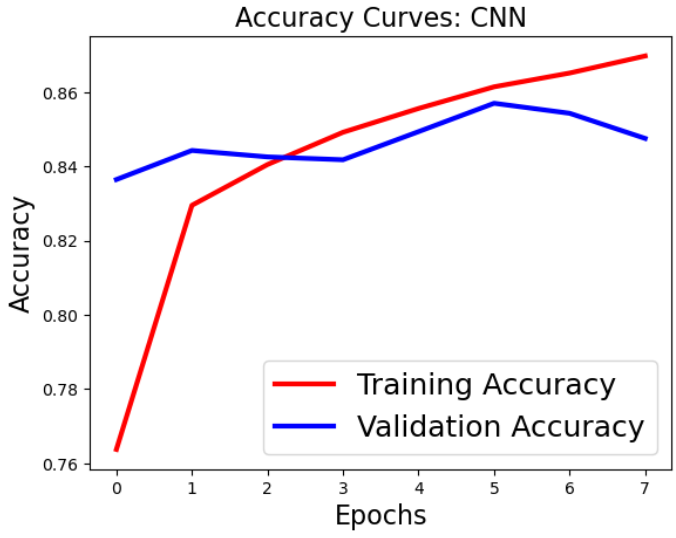
In the Figure 5.3(a), we observed a notable trend in the model's performance through the training and validation accuracy graph. As the number of epochs increased, the training accuracy also converges, reaching about 90% by the 8th epoch. In parallel, the validation accuracy, shows more gradual ascent but managed to converge around 86% by the same epoch count. To avoid overfitting, early stopping with patience of 5. This ensures that the model captures patterns during its training phase without overfitting, thereby performing well on test data. Similar trend is seen for dataset II and dataset III as illustrated in Figure 5.3(b)&(c).



(a) Accuracy Graph of Data I



(b) Accuracy Graph of Data III

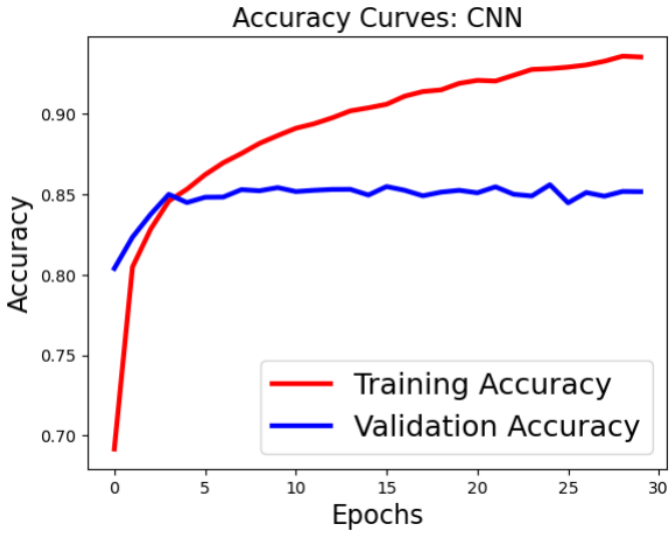


(c) Accuracy Graph of Data II

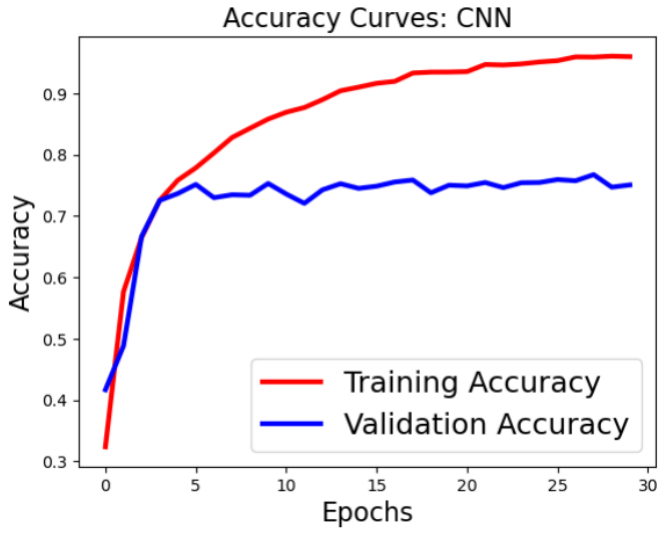
Fig 5.3: Training & Validation Accuracy of dataset I, II, & Local dataset III

5.1.7. Training & Validation Accuracy for 30 Epochs

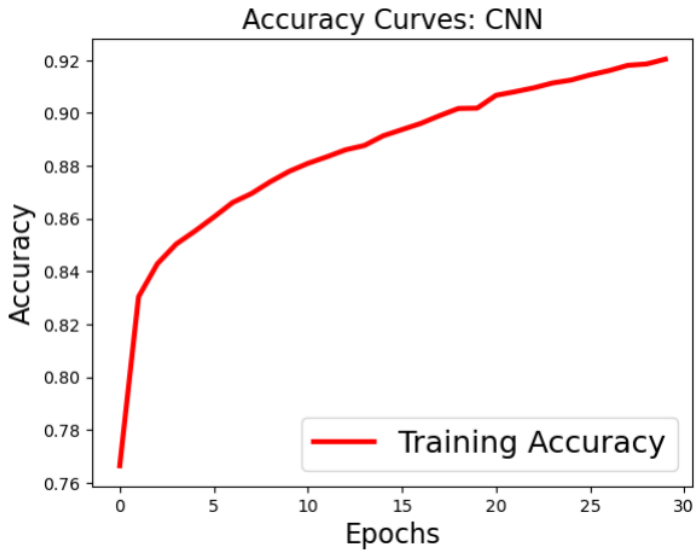
To see the trend, number of epochs has been increased and the training accuracy also converges, reaching about 98% by the 28th epoch. Conversely, the validation accuracy showed a more stable behavior, demonstrating only minor increases after stabilizing at 85% for dataset 1, as illustrated in Figure 5.4(a). So, to avoid overfitting, early stopping with patience of 5 has been utilized that ensures that the model captures patterns during its training phase without overfitting, thereby performing well on test data.



(a) Accuracy Graph of Data I



(b) Accuracy Graph of Data III

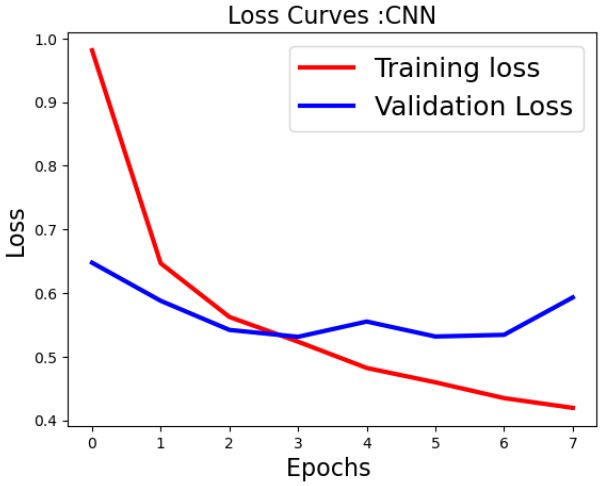


(c) Accuracy Graph of Data II

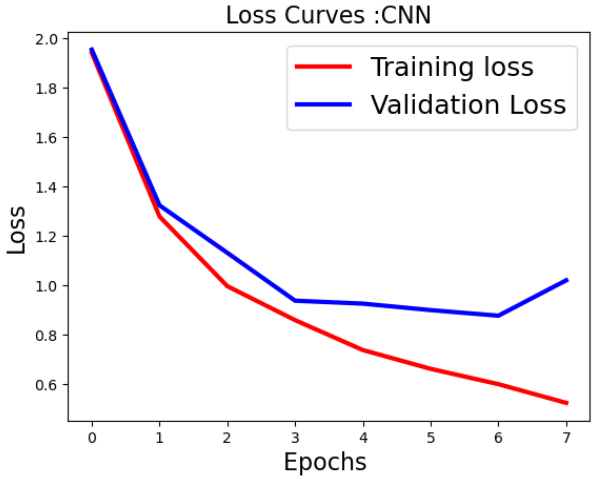
Fig 5.4: Training & Validation Accuracy for 30 Epochs on dataset I, II, & Local III

5.1.8. Training & Validation Loss

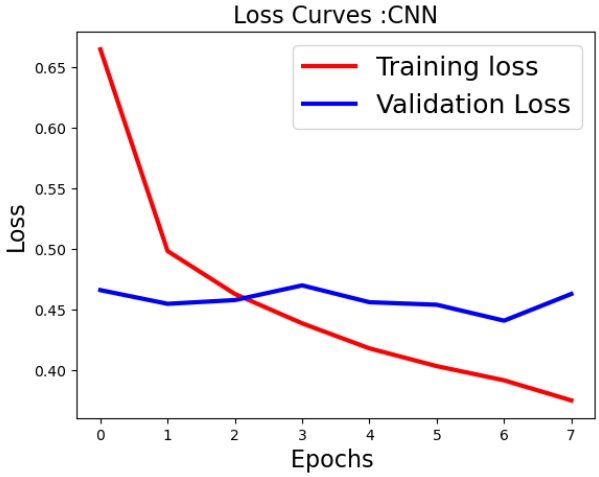
In Figure 5.5, the training goes on (represented by the number of epochs on the x-axis), the loss also called cost (on the y-axis) shows how well our model is performing. In the graph 5.5(a), up to the 4th epoch, both training and validation loss decrease together. After that, while the training loss keeps getting smaller, the validation loss levels-off around 0.5. It demonstrates minor decrease after 4^{th} epoch and stabilizes at 8^{th} epoch. This means our model will start to overfit after this point. To address this, we added early stopping with patience of 5 that means after 5 epochs without improvement, training will stop to make sure our model stays general and doesn't overfit. Overall, our model's performance looks promising in for all datasets.



(a) Loss Graph of Data I



(b) Loss Graph of Data III



(c) Loss Graph of Data II

Fig 5.5: Training & Validation Loss of 30 Epochs dataset I, II, & Local III

The primary goal of this study is to assess the performance of the proposed deep learning models and find the best approach for our problem by testing models on multiple datasets. For this purpose, experiments have been conducted using identical settings on three datasets: Local dataset I, Consumer complaints dataset II and Consumer complaints dataset III.

5.2 Hybrid CNN and BiLSTM Model

To train our hybrid model, data has been split into two subsets: 80% and 20%. The model has been trained on 80% of the data, and the remaining 20% is reserved for testing the model's performance. After the models have been trained on the extracted features from the 80% training set, they are then tested on the 20% testing set. This allows the evaluation of the models' ability to generalize their learning to new, unseen data. After applying the GloVe embeddings for feature extraction and employing our hybrid model, we acquired following results.

5.2.1. Dataset I:

This dataset comprises of 555957 samples, each belonging to one of eleven distinct classes. These samples are split into two subsets: 80% & 20%.

5.2.4. Classification Report

A classification report provides valuable metrics such as precision, recall, f1-score, and the number of instances in each class after the model is implemented. This tool allows us to evaluate the overall accuracy of the model. Below are the classification reports obtained from testing first deep learning model. Following are the testing classification reports produced by proposed models. In case of dataset I given in Figure below, the model showcased strong ability in distinguishing classes such as 'Prepaid card', 'Bank account or service', 'Credit card', 'Credit reporting', and 'Mortgage' display particularly high F1-scores, ranging between 0.80 to 0.94. This suggests that the model is well-tuned for these classes and can distinguish between them effectively. The model struggled to correctly classify 'Payday loan' with both precision and recall being 0. This indicates a complete inability of the model to recognize this class, potentially due to the limited data available for this category (with a support of only 27). Another class, 'Money transfers', although having higher metrics than 'Payday loan', still showed a relatively low F1-score of 0.49. Overall, model is indicating commendable performance with certain areas showing lack of data samples of certain class. Similar behavior is seen for dataset III but due to better distribution of samples in dataset II, model performed well by presenting F1-scores, ranging between 0.83 to 0.86.

	precision	recall	f1-score	support
Debt collection	0.90	0.71	0.79	1428
Consumer Loan	0.75	0.66	0.70	920
Mortgage	0.88	0.74	0.80	1982
Credit card	0.93	0.82	0.87	3132
Credit reporting	0.89	0.83	0.86	4388
Student loan	0.58	0.65	0.61	166
Bank account or service	0.95	0.94	0.94	3730
Payday loan	0.00	0.00	0.00	27
Money transfers	0.46	0.54	0.49	182
Other financial service	0.79	0.70	0.74	215
Prepaid card	0.89	0.77	0.83	532
micro avg	0.89	0.81	0.85	16702
macro avg	0.73	0.67	0.69	16702
weighted avg	0.89	0.81	0.85	16702
samples avg	0.81	0.81	0.81	16702

(a)CNN on Data I

	precision	recall	f1-score	support
1	1.00	0.12	0.22	8
8	0.84	0.84	0.84	263
3	0.67	0.54	0.60	235
7	0.66	0.72	0.69	207
9	0.75	0.72	0.73	219
4	0.87	0.92	0.89	662
5	0.79	0.66	0.72	295
2	0.89	0.60	0.72	206
10	0.82	0.65	0.73	325
6	0.81	0.73	0.77	81
micro avg	0.80	0.74	0.77	2501
macro avg	0.81	0.65	0.69	2501
weighted avg	0.80	0.74	0.77	2501
samples avg	0.74	0.74	0.74	2501

(b) CNN on Data III

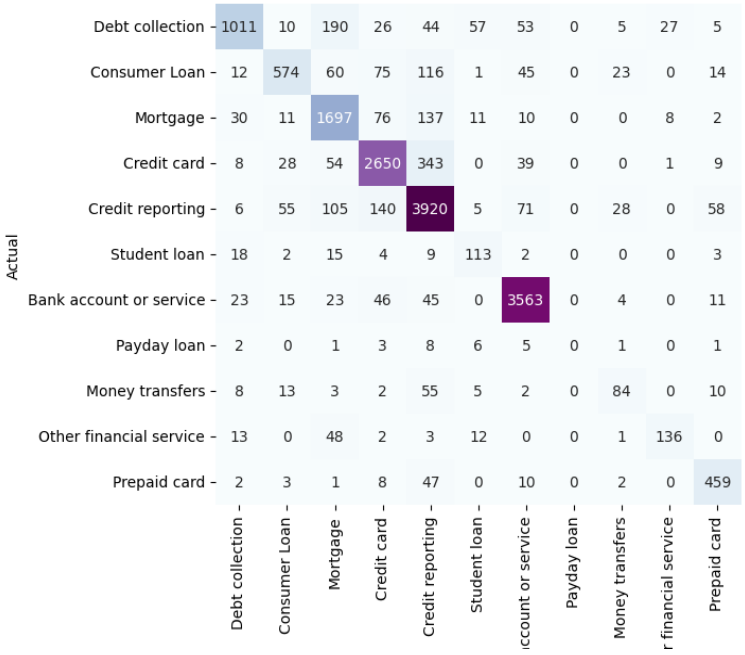
	precision	recall	f1-score	support
credit_card	0.80	0.80	0.80	3746
retail_banking	0.88	0.90	0.89	14060
credit_reporting	0.84	0.73	0.78	5264
mortgages_and_loans	0.86	0.83	0.85	4681
debt_collection	0.90	0.82	0.86	3367
micro avg	0.87	0.84	0.85	31118
macro avg	0.86	0.82	0.84	31118
weighted avg	0.87	0.84	0.85	31118
samples avg	0.84	0.84	0.84	31118

(c) CNN on Data II

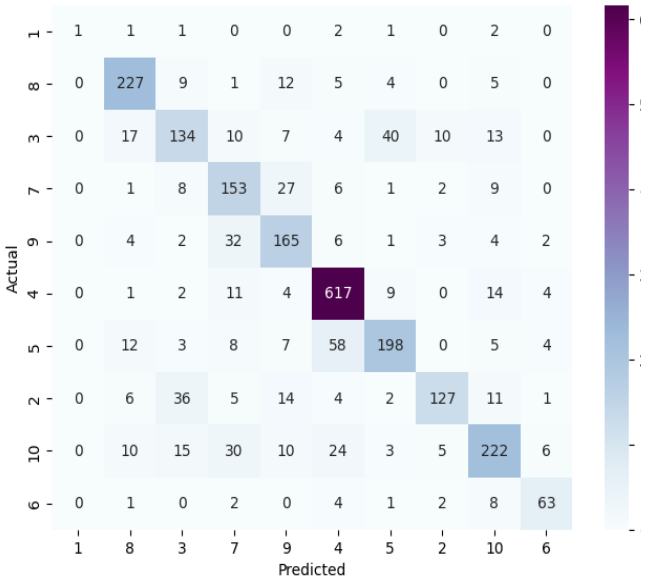
Fig 5.1: Classification reports of dataset I, dataset II, & Local dataset III

5.2.5. Confusion Matrix

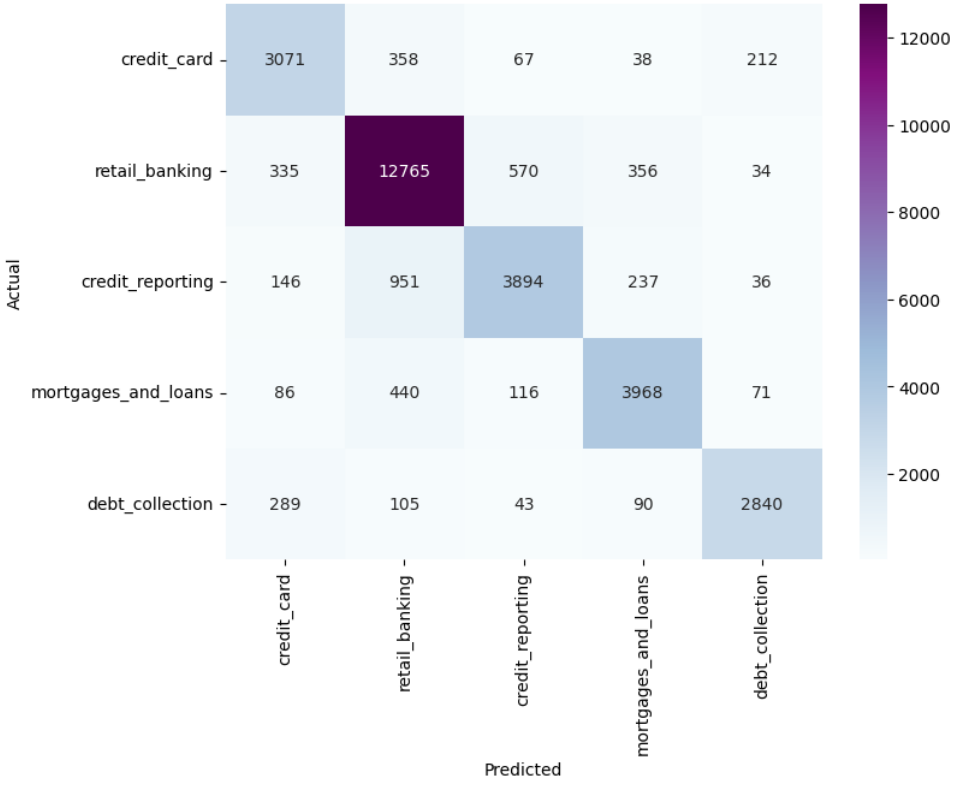
The confusion matrix assists in gaining a more precise understanding of each class data, along with evaluation of a classification model's performance. This matrix compares the true value with the value predicted by proposed model. Confusion matrices given below shows the performance of CNN model, where columns represent true classes ranging from 1 to 11, and rows depict predicted classes, also between 1 to 11.



(a)Confusion matrix of Data I



(b) Confusion matrix of Data III

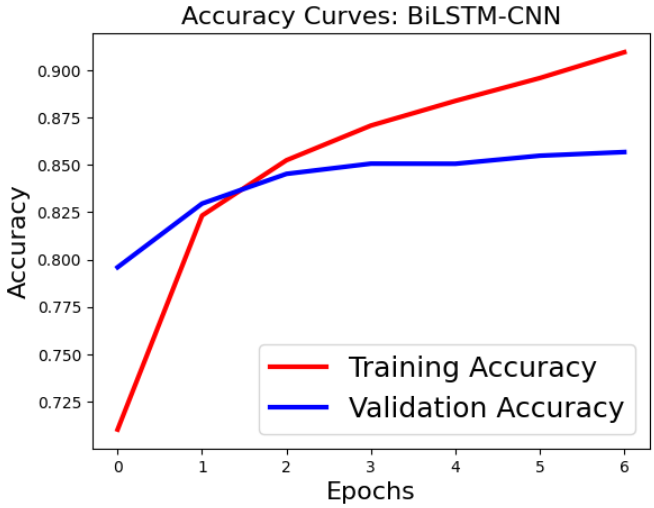


(c) Confusion matrix of Data II

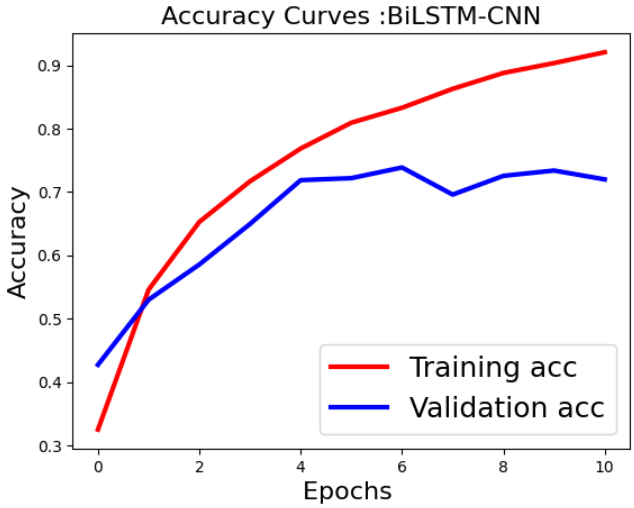
Fig 5.2: Confusion Matrix of dataset I, dataset II, & Local dataset III

5.2.6. Training & Validation Accuracy

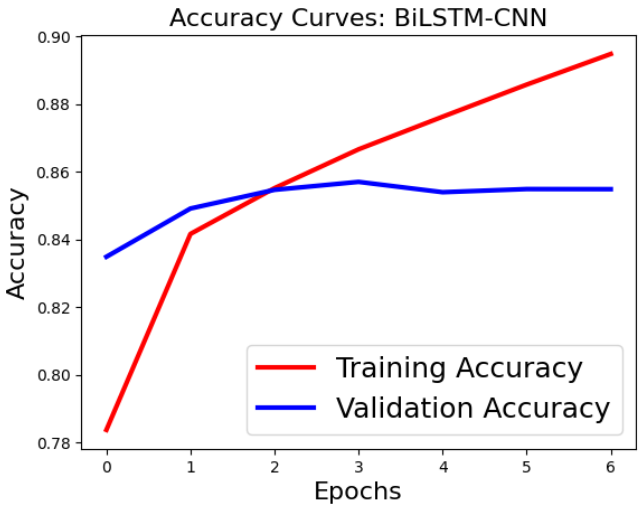
In the Figure 5.3(a), we observed a notable trend in the model's performance through the training and validation accuracy graph. As the number of epochs increased, the training accuracy also converges, reaching about 90% by the 6th epoch. In parallel, the validation accuracy, shows more gradual ascent but managed to converge around 85.5% by the same epoch count. To avoid overfitting, early stopping with patience of 3. This ensures that the model captures patterns during its training phase without overfitting, thereby performing well on test data. Similar trend is seen for dataset II and dataset III as illustrated in Figure 5.3(b)&(c).



(a) Accuracy Graph of Data I



(b) Accuracy Graph of Data III

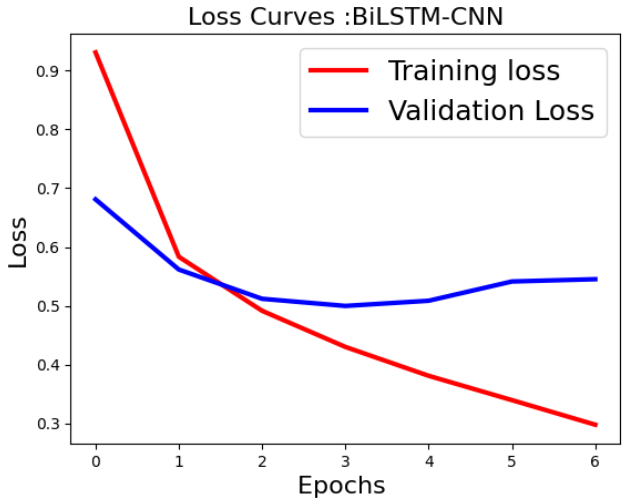


(c) Accuracy Graph of Data II

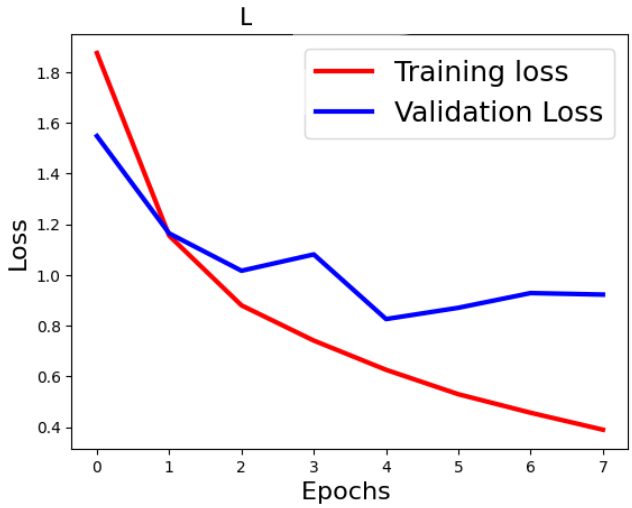
Fig 5.3: Training & Validation Accuracy of dataset I, II, & Local dataset II

5.2.8. Training & Validation Loss

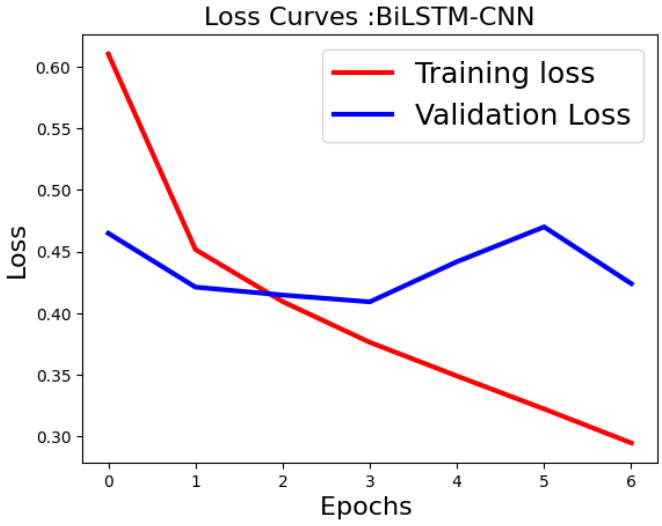
In Figure 5.5, the training goes on (represented by the number of epochs on the x-axis), the loss also called cost (on the y-axis) shows how well our model is performing. In the graph 5.5(a), up to the 3rd epoch, both training and validation loss decrease together. After that, while the training loss keeps getting smaller, the validation loss levels-off around 0.5. It demonstrates minor decrease after 4th epoch and stabilizes at 6th epoch. This means our model will start to overfit after this point. To address this, we added early stopping with patience of 3 that means after 3 epochs without improvement, training will stop to make sure our model stays general and doesn't overfit. Overall, our model's performance looks promising in for all datasets.



(a) Loss Graph of Data I



(b) Loss Graph of Data III



(c) Loss Graph of Data II

Fig 5.5: Training & Validation Loss of dataset I, II, & Local III

5.3 BERT Results

5.3.1. Dataset II & III:

These datasets comprise of 1,62,421 samples, and 1,002 samples respectively. Each belonging to one of five distinct classes in case of dataset II and nine classes (one class was excluded as there were not enough samples of that class) for dataset III. These samples are split into two subsets: 90% & 10%.

5.3.2. Classification Report

A classification report provides valuable metrics such as precision, recall, f1-score, and the number of instances in each class after the model is implemented to evaluate the overall accuracy of the model. Classification reports given below shows commendable results as in Figure 5.13(a) that is report of dataset II, the model showcased strong ability in distinguishing in almost every class by displaying particularly high F1-scores, ranging between 0.81 to 0.94. This suggests that the model is well-tuned for these classes and can distinguish between them effectively. Overall, model is indicating promising performance for dataset III as well.

	precision	recall	f1-score	support
0	0.82	0.80	0.81	3060
1	0.92	0.95	0.94	18409
2	0.83	0.79	0.81	4536
3	0.88	0.82	0.85	3782
4	0.88	0.90	0.89	2696
accuracy			0.89	32483
macro avg	0.87	0.85	0.86	32483
weighted avg	0.89	0.89	0.89	32483

MCC Score: 0.8307118747493625

Testing Accuracy of BERT:89.43%

(a)BERT on Data II

	precision	recall	f1-score	support
0	0.83	0.90	0.86	199
1	0.76	0.72	0.74	197
2	0.74	0.75	0.75	183
3	0.76	0.83	0.79	162
4	0.88	0.88	0.88	523
5	0.77	0.78	0.78	218
6	0.83	0.76	0.80	157
7	0.83	0.81	0.82	290
8	0.85	0.72	0.78	65
accuracy			0.82	1994
macro avg	0.81	0.80	0.80	1994
weighted avg	0.82	0.82	0.82	1994

Accuracy of BERT 81.59478435305918

(b) BERT on Data III

Fig 5.13: Classification reports of dataset II, & Local dataset III

5.3.3. Confusion Matrix

The confusion matrix assists in gaining a more precise understanding of each class data, along with evaluation of a classification model's performance. This matrix compares the true value with the value predicted by proposed model. Confusion matrices in Figure 5.14 shows that the

rate of misclassification is quite small compared to correctly predicted values e.g., in case of class 1 mentioned in Figure 5.14, out of total 18409 samples, 17456 were classified correctly that shows a high accuracy. Similarly other classes ranging shows same trend by predicting maximum correct values in both dataset II and dataset III.

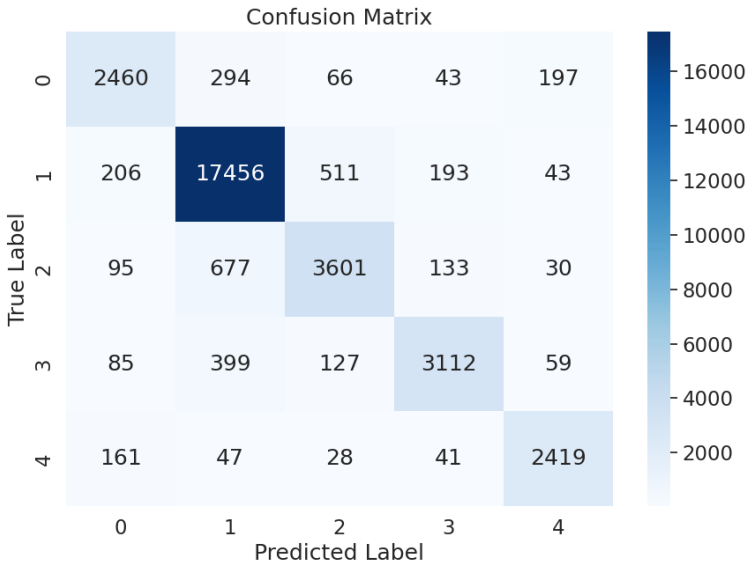


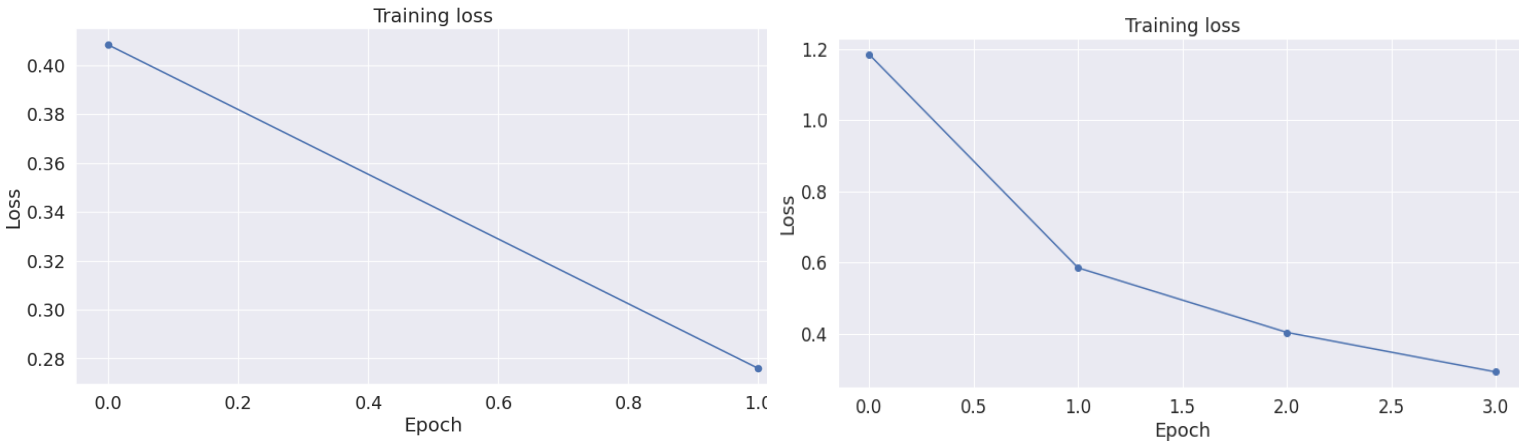
Fig 5.14: Confusion Matrix of dataset II

	1	2	3	4	5	6	7	8	9
1	**180**	5	0	4	2	5	1	2	0
2	6	**141**	3	3	4	25	8	7	30
3	1	3	**138**	20	7	0	0	12	2
4	4	1	6	**134**	12	2	3	8	2
5	7	4	17	0	**461**	18	2	11	3
6	8	3	3	3	9	**170**	1	1	0
7	5	19	3	5	2	0	**120**	3	0
8	7	9	16	6	7	0	8	**236**	1
9	0	1	1	1	8	0	1	6	**47**

Fig 5.15: Confusion Matrix of dataset III

5.3.4. Training Loss

In Figure 5.16(b), the trend of training loss is promising as loss is decreasing prominently. The training loss levels-off around 0.3 at 3rd epoch without over-fitting. This value of loss is so small, indicating the accuracy of model. Overall, BERT model's performance outperforms the other models as seen in the both graphs.



(a) Training Loss of Data II

(b) Training Loss of Data III

Fig 5.16: Training Loss of dataset II, & Local dataset III

5.3.5. Training Accuracy

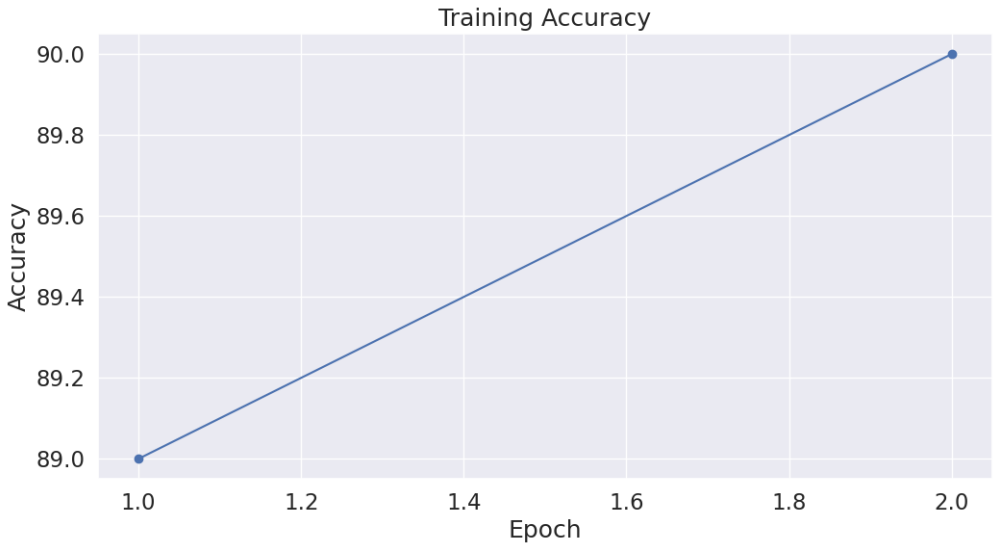
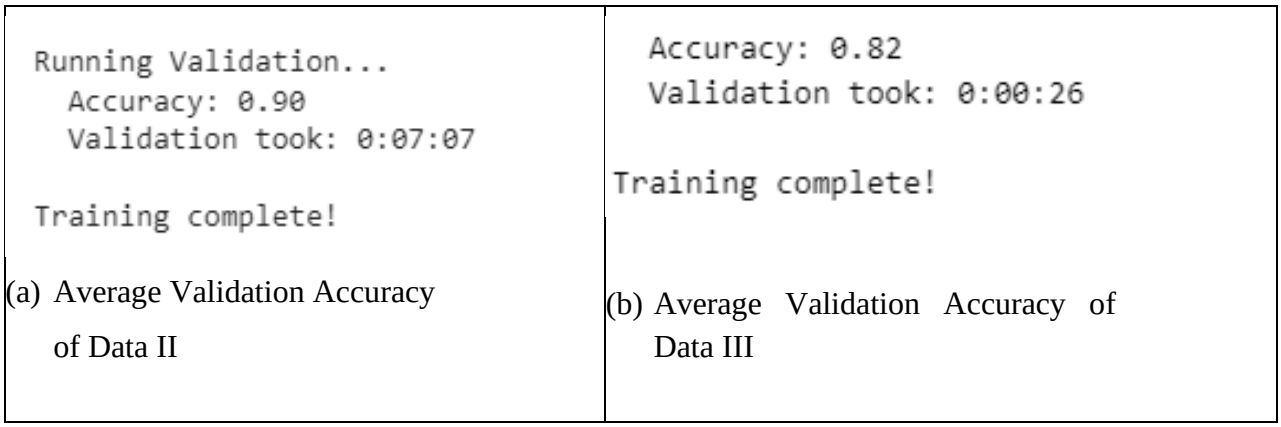


Fig 5.17: Training Accuracy of dataset II

5.3.6. Validation Accuracy

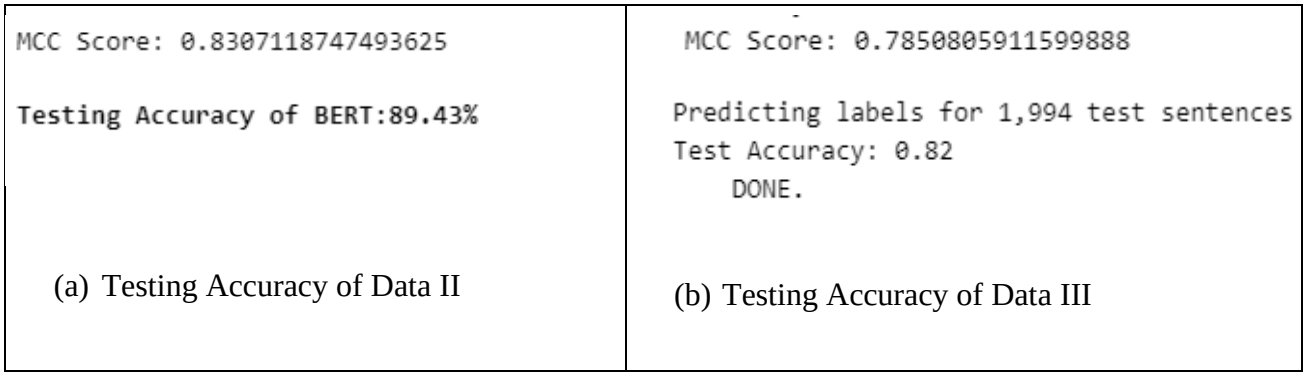


(a) Average Validation Accuracy of Data II

(b) Average Validation Accuracy of Data III

Fig 5.18: Validation Accuracy of Dataset II & Local Dataset II

5.3.7. Testing Accuracy



(a) Testing Accuracy of Data II

(b) Testing Accuracy of Data III

Fig 5.19: Testing Accuracy of Dataset II & Local Dataset II

5.3.8. Testing on Unseen Data

Excitingly, I provided an unseen sentence to the BERT model of class 4, that is retail banking, and it accurately predicted the sentence as belonging to class 4

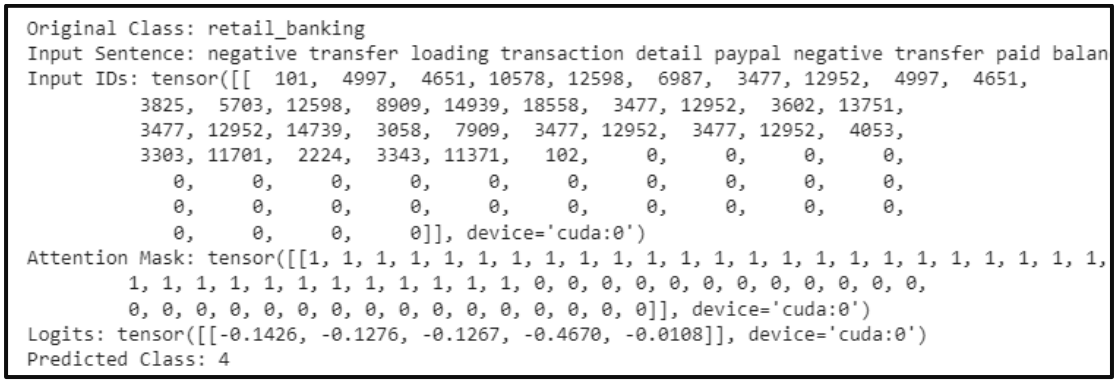


Fig 5.20: Testing on Unseen Sentence Sample

5.4 Performance Measures of our Models

Table 5.1: Testing & Average Cross-Validation Accuracy of all model on dataset I

ACCURACY OF MODEL FOR DATASET I		
Serial No.	**MODEL**	**Accuracy**
1	CNN	**85%**
2	BiLSTM	**88%**
CROSS-VALIDATION ACCURACY OF MODEL		
Serial No.	**MODEL**	**Accuracy**
1	CNN	**84%**
2	BiLSTM	**85%**

Table 5.2: Testing & Average Cross-Validation Accuracy of all model on dataset II

ACCURACY OF MODEL FOR DATASET II		
Serial No.	**MODEL**	**Accuracy**
1	CNN	**85%**
2	BiLSTM	**87%**
3	BERT	**89.43%**
CROSS-VALIDATION ACCURACY OF MODEL		
Serial No.	**MODEL**	**Accuracy**
1	CNN	**85%**
2	BiLSTM	**84%**
3	BERT	**90%**

Table 5.3: Testing & Average Cross-Validation Accuracy of all model on Local dataset III

ACCURACY OF MODEL FOR DATASET III		
Serial No.	**MODEL**	**Accuracy**
1	CNN	**76%**
2	BiLSTM	**80%**
3	BERT	**82%**
CROSS-VALIDATION ACCURACY OF MODEL		
Serial No.	**MODEL**	**Accuracy**
1	CNN	**75%**
2	BiLSTM	**75%**
3	BERT	**82%**

Comparison with Prior Work on Dataset II		
Reference	**Models**	**80/20 Testing Accuracy**
Qurat-ul-ain[14]	Logistic Regression	83%
	Random Forest	82%
	Mutlinomial NB	78%
Proposed System	CNN	87%
	BiLSTM	88%
	BERT	89.43%

Table 5.4: Accuracies on Consumer Complaints Dataset II [14]

Comparison with Prior Work on Dataset I		
Reference	**Models**	**80/20 Testing Accuracy**
Oyewola, D.O[37]	1DCNN	76.66%
	LSTM	75.53%
	BiLSTM	71.98%
	TSR1DCNN	76.66%
N. T. Thomas [8]	LSTM	62.825%
Proposed System	CNN	87%
	BiLSTM	88%
	BERT	89.43%

Table 5.5: Accuracies on US Consumer Finance Complaints Dataset I [37]

In Table 5.4 and 5.5, we present a performance comparison between our proposed method and the findings from recent work. Various feature extraction techniques and models were employed by different researchers. The results mentioned in Table III shows our proposed model that is BiLSTM-CNN models performed well. Subsequently, BERT, state-of-the-art model outperformed previous models in autonomous classification of user complaints as reported in Table 5.5.

CHAPTER 6: CONCLUSION & FUTURE WORK

6.1. Conclusion

As the digital world continues to evolve, the need for advanced models in classifying and managing unstructured text data like consumer complaints in our case, undoubtedly increases. Hence, the combination of deep learning models and Natural Language Processing (NLP) techniques has a potential to give promising solutions for managing and categorizing the vast amounts of consumer complaints online. Moreover, this research has showcased the potential of Convolutional Neural Networks, hybrid CNN-BiLSTM, and BERT models in the field of classifying consumer complaints accurately. The CNN model was adept at capturing local correlations through its convolution and pooling operations, proving to be beneficial for short text classification tasks. The hybrid model, combining CNN and Bidirectional Long Short-Term Memory (BiLSTM), excelled in addressing both local and global semantic information, rendering it effective for complaints. And the game-changer, however, was the BERT model. Its unique contextual fetching power and bidirectional processing approach allowed for a deeper understanding of language context, critical for accurate complaint classification. For this reason, BERT proved to be a state-of-the-art model in handling consumer complaints by showing an accuracy of 89.5%. In the same context, the CNN model achieved 85% accuracy, while BiLSTM posted 87% on Dataset I. Additionally, on the local Dataset II, the performance remained competitive: CNN recorded 77% accuracy, the hybrid BiLSTM-CNN model reported 80%, and the BERT model outperformed with 82% testing accuracy. The results achieved on the Consumer Complaint dataset I will serve as a baseline as it is an international acclaimed dataset. In conclusion, the application of Autonomous Decision-Making approach on user complaints, powered by NLP and deep learning models, enables organizations to strive for more precise complaint categorization, leading to improved customer service and satisfaction, and ultimately, a better digital customer experience.

6.2. Future Work

Future research should consider challenges such as multilingual datasets and non-English languages, like Urdu in our case, which are getting the attention of ample number of researchers. Deep learning algorithms are used, different progressing features techniques have been selected for our problem. However, there is still room for improvement,

potentially through the use of hybrid models to get better results. The task of extracting and choosing optimal feature techniques according to the problem remains crucial. Additionally, the implementation of other ensembles using a combination of different features presents a promising area for exploration. New and more efficient model could be Implemented and designed with a little effort.

REFERENCES

[1] Dien, T.T., Loc, B.H. and Thai-Nghe, N., 2019, November. Article classification using natural language processing and machine learning. In 2019 International Conference on Advanced Computing and Applications (ACOMP) (pp. 78-84). IEEE.

[2] Lai, S., Xu, L., Liu, K. and Zhao, J., 2015, February. Recurrent convolutional neural networks for text classification. In Proceedings of the AAAI conference on artificial intelligence (Vol. 29, No. 1).

[3] X. Tong, B. Wu, S. Wang and J. (2018), "A Complaint Text Classification Model Based on Character-Level Convolutional Network," 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, pp. 507-511, doi: 10.1109/ICSESS.2018.8663873.

[4] Dien, T.T., Thanh-Hai, N. and Thai-Nghe, N., 2020. Deep Learning Approach for Automatic Topic Classification in an Online Submission System. Advances in Science, Technology and Engineering Systems Journal, 5(4), pp.700-709.

[5] Zang, M., Gao, Y., Niu, S. and Chen, X., 2023, February. Long Text Multi-label Classification. In 2023 3rd International Conference on Neural Networks, Information and Communication Engineering (NNICE) (pp. 438-442). IEEE.

[6] Gupta, M., Singh, A., Jain, R., Saxena, A. and Ahmed, S., 2021. Multi-class railway complaints categorization using Neural Networks: RailNeural. Journal of Rail Transport Planning & Management, 20, p.100265.

[7] Singh, M., Jakhar, A.K. and Pandey, S., 2021. Sentiment analysis on the impact of coronavirus in social life using the BERT model. Social Network Analysis and Mining, 11(1), p.33.

[8] Chen, Q., Du, J., Allot, A. and Lu, Z., 2022. LitMC-BERT: transformer-based multi-label classification of biomedical literature with an application on COVID-19 literature curation. IEEE/ACM transactions on computational biology and bioinformatics, 19(5), pp.2584-2595.

[9] Liu, H., Yin, Q. and Wang, W.Y., 2018. Towards explainable NLP: A generative explanation framework for text classification. arXiv preprint arXiv:1811.00196.

[10] Raju, S.V., Bolla, B.K., Nayak, D.K. and Kh, J., 2022, April. Topic modelling on consumer financial protection bureau data: An approach using BERT based embeddings. In 2022 IEEE 7th International conference for Convergence in Technology (I2CT) (pp. 1-6). IEEE.

[11] Sun, F. and Zuo, Y., 2022. Autonomous Classification and Decision-Making Support of Citizen E-Petitions Based on Bi-LSTM-CNN. Mathematical Problems in Engineering, 2022.

[12] Tutika, A. and Nagesh, M.Y.V., 2019. Restaurant reviews classification using NLP Techniques. Journal of Information and Computational Science, 9(11), pp.1669-1676.

[13] Rao, Z. and Zhang, Y., 2020, June. Research on Content of User Complaint Classification Based on Data Mining. In 2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC) (pp. 1080-1085). IEEE.

[14] Qurat-Ul-Ain, A. Shaukat and U. Saif, "NLP based Model for Classification of Complaints: Autonomous and Intelligent System," 2022 2nd International Conference on Digital Futures and Transformative Technologies (ICoDT2), Rawalpindi, Pakistan, 2022, pp. 1-6, doi: 10.1109/ICoDT255437.2022.9787456.

[15] Oliinyk, V.A., Vysotska, V., Burov, Y., Mykich, K. and Basto-Fernandes, V., 2020. Propaganda detection in text data based on NLP and machine learning. In CEUR Workshop Proceedings (Vol. 2631, pp. 132-144).

[16] Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K. and Kuksa, P., 2011. Natural language processing (almost) from scratch. Journal of machine learning research, 12(ARTICLE), pp.2493-2537.

[17] Khurana, D., Koli, A., Khatter, K. and Singh, S., 2023. Natural language processing: State of the art, current trends and challenges. Multimedia tools and applications, 82(3), pp.3713-3744.

[18] Jones, K.S., 1994. Natural language processing: a historical review. Current issues in computational linguistics: in honour of Don Walker, pp.3-16.

[19] Liddy, E.D., 2001. Natural language processing.

[20] Dudhabaware, R.S. and Madankar, M.S., 2014, December. Review on natural language processing tasks for text documents. In 2014 IEEE International Conference on Computational Intelligence and Computing Research (pp. 1-5). IEEE.

[21] North, K., Ranasinghe, T., Shardlow, M. and Zampieri, M., 2023. Deep Learning Approaches to Lexical Simplification: A Survey. arXiv preprint arXiv:2305.12000.

[22] Webster, J.J. and Kit, C., 1992. Tokenization as the initial phase in NLP. In COLING 1992 volume 4: The 14th international conference on computational linguistics.

[23] Goldberg, Y., 2016. A primer on neural network models for natural language processing. Journal of Artificial Intelligence Research, 57, pp.345-420.

[24] Pennington, J., Socher, R. and Manning, C.D., 2014, October. Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).

[25] Chua, L.O. and Roska, T., 1993. The CNN paradigm. IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, 40(3), pp.147-156.

[26] Yin, W., Kann, K., Yu, M. and Schütze, H., 2017. Comparative study of CNN and RNN for natural language processing. arXiv preprint arXiv:1702.01923.

[27] Sherstinsky, A., 2020. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. Physica D: Nonlinear Phenomena, 404, p.132306.

[28] Graves, A. and Graves, A., 2012. Long short-term memory. Supervised sequence labelling with recurrent neural networks, pp.37-45.

[29] Analytics Vidhya. (2021, March). Introduction to Long Short-Term Memory (LSTM). Analytics Vidhya.https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/

[30] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M. and Davison, J., 2020, October. Transformers: State-of-the-art natural language processing. In Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations (pp. 38-45).

[31]

[32]Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I., 2017. Attention is all you need. Advances in neural information processing systems, 30.

[33] Jalammar, J. The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning). https://jalammar.github.io/illustrated-bert/

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

[35] Jalammar, J. A Visual Guide to Using BERT for the First Time. https://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

[36] Anna Rogers, Olga Kovaleva, Anna Rumshisky; A Primer in BERTology: What We Know About How BERT Works. *Transactions of the Association for Computational Linguistics* 2020; 8 842–866.

[37]Oyewola, D.O., Omotehinwa, O.T. and Dada, E.G., 2023. Consumer complaints of consumer financial protection bureau via two-stage residual one-dimensional convolutional neural network (TSR1DCNN). Data and Information Management, p.100046.