DAYANANDA SAGAR UNIVERSITY

KUDLU GATE, BANGALORE – 560068

Bachelor of Technology
in
COMPUTER SCIENCE AND ENGINEERING

Special Topic- 1 Report

FEEL GOOD OR FEEL BAD

By
Name of the student- Sree Vibha G (ENG21CS0412)
Name of the student- Nandeesh P Math (ENG21CS0263)
Name of the student- Sri Vishnavi Ananya Gollapalli (ENG21CS0414)
Name of the student- N Rishi Rohan (ENG21CS0256)

Under the supervision of
Prof. Arjun Krishnamurthy
Dept of CSE

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING,
SCHOOL OF ENGINEERING
DAYANANDA SAGAR UNIVERSITY,

(2022-2023)

DAYANANDA SAGAR UNIVERSITY

School of Engineering

Department of Computer Science& Engineering

Kudlu Gate, Bangalore –560068
Karnataka, India

CERTIFICATE

This is to certify that the Special Topic 1 titled **“FEEL GOOD OR FEEL BAD”** is carried out by **Sree Vibha G (ENG21CS0412), Nandeesh P Math (ENG21CS263), Sri Vishnavi Ananya Gollapalli (ENG21CS0414), N Rishi Rohan (ENG21CS0256)** bonafede students of Bachelor of Technology in Computer Science and Engineering at the School ofEngineering, Dayananda Sagar University, Bangalore in partial fulfillment for the award of degree in Bachelor of Technology in Computer Science and Engineering, during the year **2022-2023**.

Prof. Arjun Krishnamurthy	**Dr. Girisha G S**	**Dr. Uday Kumar Reddy K R**
Dept. of CSE, School of Engineering Dayananda Sagar University Date:	Chairman, CSE School of Engineering Dayananda Sagar University Date:	Dean School of Engineering Dayananda Sagar University Date:

Name of the Examiner **Signature of Examiner**

1.

2

DECLARATION

We, **Sree Vibha G (ENG21CS0412), Nandeesh P Math (ENG21CS263), Sri Vishnavi Ananya Gollapalli (ENG21CS0414), N Rishi Rohan (ENG21CS0256)** are students of the fourth semester B.Tech in **Computer Science and Engineering**, at School of Engineering, **Dayananda Sagar University**, hereby declare that the Special Topic 1 titled **"Feel Good Or Feel Bad"** has been carried out by us and submitted in partial fulfilment for the award of degree in **Bachelor of Technology in Computer Science and Engineering** during the academic year **2022-2023**.

Student	**Signature**
Name1: Sree Vibha G **USN : (ENG21CS0412)**	
Name2 : Nandeesh P Math **USN : (ENG21CS263)**	
Name3: Sri Vishnavi Ananya Gollapalli **USN : (ENG21CS0414)**	
Name4: N Rishi Rohan **USN : (ENG21CS0256)**	
Place : Bangalore **Date : 07/06/2023**	

ACKNOWLEDGEMENT

It is a great pleasure for us to acknowledge the assistance and support of many individuals who have been responsible for the successful completion of this Special Topic 1.

First, we take this opportunity to express our sincere gratitude to School of Engineering & Technology, Dayananda Sagar University for providing us with a great opportunity to pursue our Bachelor’s degree in this institution.

We would like to thank **Dr. Uday Kumar Reddy K R, Dean**, **School of Engineering & Technology**, **Dayananda Sagar University** for his constant encouragement and expert advice. It is a matter of immense pleasure to express our sincere thanks to **Dr. Girisha G S, Chairman**, **Department of Computer Science, and Engineering**, **Dayananda Sagar University,** for providing the right academic guidance that made our task possible.

We would like to thank our guide Mr.Arjun Krishnamurthy,**Dept. of Computer Science and Engineering**, **Dayananda Sagar University**, for sparing his/her valuable time to extend help in every step of our Special Topic 1, which paved the way for smooth progress and the fruitful culmination of the project.

We would like to thank our Special Topic 1 Coordinators, Dr. Savitha Hiremath, Dr. Bondu Venkateswarlu, and all the staff members of Computer Science and Engineering for their support.

We are also grateful to our family and friends who provided us with every requirement throughout the course. We would like to thank one and all who directly or indirectly helped us in the Special Topic 1.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
CPU	Central Processing Unit
FN	False Negative
FP	False Positive
GPU	Graphics Processing Unit
LSTM	Long-short term memory
Mbps	Mega bytes per second
ML	Machine Learning
MLP	Multi-Layer Perceptron
NLP	Natural Language Processing
RAM	Random Access Memory
RNN	Recurrent Neural Network
ROM	Read Only Memory
TN	True Negative
TP	True Positive
VoC	Voice of Customer

LIST OF FIGURES

ABSTRACT

The project chosen is Sentiment Analysis using Natural Language Processing.

The purpose of this research is to investigate sentiment analysis in the field of (NLP) using deep learning techniques, particularly Recurrent Neural Networks (RNNs). Sentiment analysis plays a crucial role in understanding and analysing emotions, opinions, attitudes expressed in textual data, making it an essential task in domains such as social media monitoring, customer feedback analysis, & market research.

Previous research has shown promising results in sentiment analysis using traditional machine learning algorithms. However, with the advancements in deep learning, specifically RNNs, there is a need to explore it's potential using these techniques.

The objective of this research is to evaluate the effectiveness of RNN-based models for sentiment analysis tasks and determine the ability of RNNs to capture contextual dependencies and long-term dependencies within textual data, which can lead to enhanced sentiment classification accuracy and robustness.

To achieve our objective, we will employ research methods that involve data collection, pre-processing, data cleaning, extraction, classification, training and evaluation of RNN-based models on sentiment analysis datasets. We will analyse the results to determine the strengths and limitations of these models and provide insights into their potential applications and future research directions

CHAPTER 1
INTRODUCTION: SENTIMENT ANALYSIS

Sentiment analysis is the process of compiling and examining the opinions, ideas, and impressions of people about a variety of issues, products, and services. Corporations, governments, and people can all benefit from public opinion when gathering data and making judgement calls.

The Internet has become the primary source of global knowledge thanks to its rising popularity. Many individuals communicate their views and opinions through numerous online resources. We must use user-generated data to analyse public opinion automatically in order to continuously monitor public opinion and assist decision-making. As a result, sentiment analysis has grown more well-liked in recent years among research communities. Opinion analysis and opinion mining are other names for sentiment analysis.

Sentiment analysis is necessary for thorough examination in a number of real-world applications. Find out, for instance, through product analysis whether features or characteristics of a product appeal to consumers in terms of product quality.

Deep learning techniques, lexicon-based techniques, and even hybrid methods can all be classified as sentiment analysis techniques. Multimodal sentiment analysis, aspect-based sentiment analysis, fine-grained opinion analysis, and language-specific sentiment analysis are a few subcategories of sentiment analysis research.

Figure 1.1 Sentiment Analysis

1.1 SENTIMENT ANALYSIS PROCESS

The sentiment analysis process typically involves the following steps:

1.Data Collection

2.Text Pre-processing

3.Feature Extraction

4.Sentiment Classification

5.Evaluation and Validation

NLP IN SENTIMENT ANALYSIS

Natural Language Processing (NLP) plays a crucial role in sentiment analysis by providing the necessary tools and techniques to process and understand text data. Here are some ways NLP is used in sentiment analysis:

1. Text pre-processing : NLP techniques are applied to pre-process the text input before sentiment analysis. This covers activities like as tokenization (dividing text into individual words or tokens), deleting stop words (frequent words that do not carry much feeling), stemming (reducing words to their base or root form), and managing special characters or punctuation.

2**. Sentiment lexicons** : NLP facilitates the usage of sentiment lexicons, which are curated sets of words or phrases associated with sentiment scores (e.g., positive or negative). Based on the availability of sentiment-bearing words or phrases, these lexicons assist sentiment analysis algorithms in assigning sentiment labels to text.

3. Machine learning and deep learning models : NLP provides a range of machine learning and deep learning models that can be applied to sentiment analysis. These models can learn patterns and relationships in text data to predict sentiment labels. Techniques like recurrent neural networks (RNNs), convolutional neural networks (CNNs), and transformers (e.g., BERT or GPT) have shown excellent performance in sentiment analysis tasks

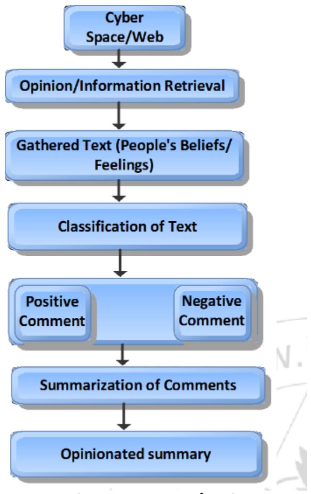

Cyber Space/Web
Opinion/Information Retrieval
Gathered Text (People's Beliefs/ Feelings)
Classification of Text
Positive Comment
Negative Comment
Summarization of Comments
Opinionated summary

Figure 1.1.1 Sentiment Analysis Overview

1.2 WHAT DOMAIN DOES SENTIMENT ANALYSIS COME UNDER?

A division of natural language processing (NLP) is sentiment analysis. NLP includes a variety of methods and uses that include a conversation between a machine and human language. Understanding and extracting sentiment or emotional information from text data is the focus of sentiment analysis.

Although sentiment analysis can be utilised in many other fields and sectors, it is frequently employed in areas like:

1.Social Media Monitoring

2.Customer Feedback and Reviews

3.Market Research

4.Reputation Management

5.Voice of the Customer (VoC) Analysis

6.Financial Analysis

7.Political Analysis

To increase accuracy and automate the sentiment analysis process, artificial intelligence (AI) approaches are frequently applied. Following are some examples of how AI is used in sentiment analysis:

Machine learning models : In order to discover patterns and connections between text attributes and sentiment labels, AI-based machine learning models are trained on labelled datasets. These models can be trained using a variety of techniques, including Gradient Boosting, Naive Bayes, Support Vector Machines (SVM), and Random Forests. In order to make predictions on unread text data, they learn from instances.

Deep learning models : By utilising neural networks to recognise intricate patterns in text data, deep learning, a branch of artificial intelligence, has transformed sentiment analysis. Convolutional neural networks (CNNs), LSTM networks, recurrent neural networks (RNNs), and transformer-based models (such BERT, GPT), among others, have demonstrated outstanding performance in sentiment analysis applications.

Transfer learning : Using pre-trained models that have been trained on substantial amounts of data as a starting point for a particular job is known as transfer learning in artificial intelligence. Pre-trained language models like BERT or GPT, which have been refined on sentiment-specific datasets, can perform better and use less processing power when used in sentiment analysis.

Feature extraction : To extract pertinent features from text data for sentiment analysis, AI techniques are used. In the sentiment analysis process, these features can include word embeddings (such as Word2Vec, GloVe) or contextualised word representations (such as ELMo, BERT) that capture the semantic meaning and context of words.

Sentiment Analysis API's : Sentiment analysis with AI APIs offer pre-built tools and services that make sentiment analysis implementation simple. Developers may incorporate sentiment analysis capabilities into their applications without having to create models from start thanks to these APIs, which use AI models and algorithms to analyse sentiment in real-time or on-demand.

Scalability, adaptability to many domains and languages, and more accurate and effective sentiment classification are all made possible by AI techniques in sentiment analysis. They assist in automating the sentiment analysis procedure and giving insightful information about the sentiment found in enormous amounts of text data.

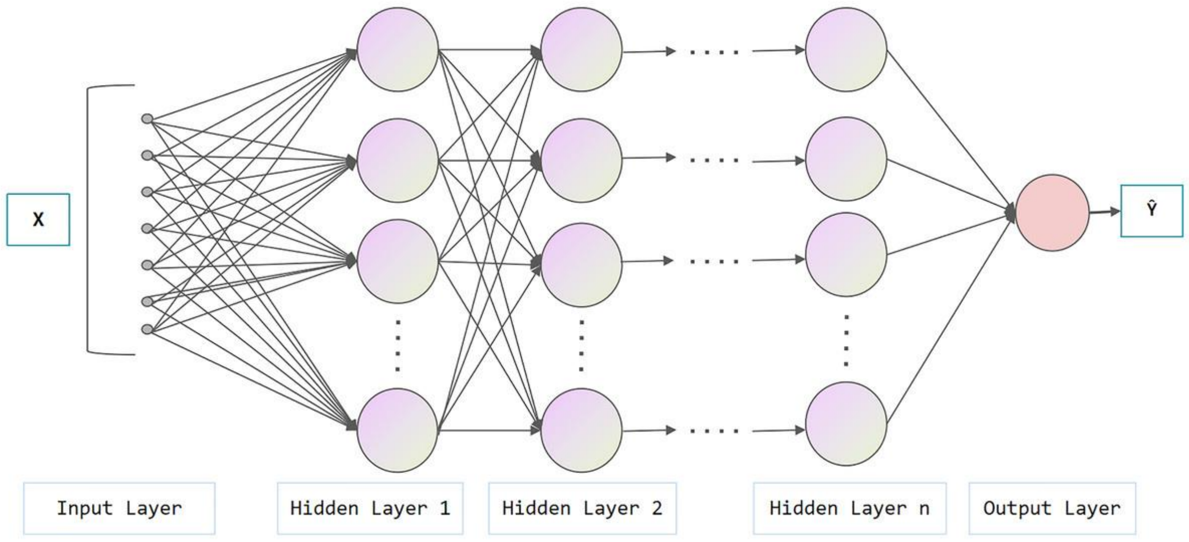


Figure 1.2.1 Layers involved

CHAPTER 2: PROBLEM DEFINITION

The aim is to perform a learning model in the field of AI using Deep Learning Techniques specifically designed to accurately predict the sentiment of movie reviews, classifying them as either positive or negative. By leveraging the vast amount of available movie review data, our objective is to create a model that can effectively capture the underlying sentiment expressed in textual reviews.

CHAPTER 3 :LITERATURE REVIEW

3.1 LITERATURE REVIEW-1

Name: “A Survey of Sentiment Analysis: Approaches, Datasets, and Future Research”

Authors: **Kian Long Tan, Chin Poo Lee and Kian Ming Lim**

Publication: Citation: Tan, K.L.; Lee, C.P.; Lim, K.M. A Survey of Sentiment Analysis: Approaches, Datasets, and Future Research. Appl. Sci. 2023,

Date of publication: 3 April 2023

Technology/Design used: machine learning, deep learning, or ensemble learning.

Result shared by author: Need for more trustworthy language models.

Focus toward deep learning techniques where, text is first encoded into pretrained word embeddings such as GloVe, word2vec, or fastText which are fed into deep learning models such as CNN, LSTM, GRU, etc.

Inference: The paper talks about Sentiment analysis and the steps done throughout to achieve the necessary output. It covers the deep learning, machine learning, ensemble learning approaches, datasets, limitations and future research prospects

1.The focus has been shifted to mainly to the deep learning approach

2. Future study should concentrate on fine-grained and elaborate sentiment analysis that includes classes with various emotional tensors like very positive, positive, neutral, negative, and extremely negative

3.2 LITERATURE REVIEW-2

Name: "A two-stage unsupervised sentiment analysis method "

Authors: **Yingqi Wang & Hongyu Han & Xin He & Rui Zhai**

Publication: Multimedia Tools and Applications

Date of publication: Received: 20 March 2022 / Revised: 29 June 2022 / Accepted: 6 February 2023

Technology/Design used: SASC (Sentiment Analysis based on Sentiment Clustering)

Result shared by author: Un-labeled reviews are taken as research object, and sentiment analysis method is developed based on topic model LDA and K-means algorithm. Firstly, the status of unsupervised learning sentiment analysis is briefly analysed, and then the application of clustering algorithm in sentiment analysis is introduced in detail. Secondly, a two-stage sentiment analysis method is proposed.

Inference: In order to address the issues of low accuracy and unstable review sentiment clustering methodologies, the SASC (Sentiment Analysis based on Sentiment Clustering) method is put forth.To increase the precision and stability of the findings, two-stage sentiment clustering is used to extract the latent sentiment information from the review texts.

3.3 LITERATURE REVIEW-3

Name: “A Rule-Based Approach for Sentiment Analysis of Products Review”

Authors: **Tapasy Rabeya and Ismail Jabiullah**

Publication: Elixir Inform. Tech. 164 (2022) 56147-5615113, 4550

Date of publication: Received: 12 January 2022; Received in revised form: 17 March 2022; Accepted: 27 March 2022;

Technology/Design used: Machine learning, Deep learning, Ensemble learning.

Result shared by author: Analyses the expressions from the textual reviews of a product for predicting sentiment and assigns scores for our predefined features to present a net sentiment profile of a product of all parameters and attributes that seem to be beneficial to the customer and manufacturer.

Inference: The main aim of the paper is to make product buying efficient and easy by sorting the reviews into negative and positive with respective to the assigned words about the product about which the decision is taken on the expression given or commented. With the rising popularity of E-Commerce trade, sentiment analysis in this domain will prove very helpful in filtering out the products by the customer.

3.4 LITERATURE REVIEW-4

Name: “A Review on Twitter Sentiment Analysis Using ML”

Authors: **Prof. R Y Totare , Aishwarya Ahergawli , Abhijeet Girase , Ishwari Tale , Ayushi Khanbard**

Publication International Journal for Research in Applied Science & Engineering Technology

Date of publication: (IJRASET) ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538 Volume 10 Issue XII Dec 2022-

Technology/Design used: Sentiment reversal prediction model

Result shared by author: This research focuses on how to merge the textual content of Twitter tweets and sentiment propagation models. examines the spread of feelings by researching a phenomena known as inversion of feelings and identifies several intriguing characteristics of reversal of feelings.

Inference: Due to the distinctive qualities of Twitter messages, the majority of existing Twitter sentiment analysis systems simply take into account the textual information of Twitter messages and are unable to deliver sufficient performance..

Existing methods ignore the dispersion of emotional information

The need to study Sentiment analysis by taking in account the concepts of other domains, fuse the knowledge and integrate textual data with distributing emotions and enhance twitter sentiment analysis

CHAPTER 4: PROJECT DESCRIPTION

Sentiment analysis, a subfield of natural language processing (NLP), seeks to find a text's or document's emotional undertone. Deep learning approaches, such as recurrent neural networks (RNNs), implemented with TensorFlow and Keras frameworks, are one well-liked method of sentiment analysis. Combining these two elements, it creates a flexible and effective framework for creating and refining sentiment analysis models, enabling precise predictions on a variety of text inputs. This potent combination makes it possible to do efficient and precise sentiment analysis on significant amounts of text data.

The IMDb movie review dataset is used to construct a sentiment analysis project in the code. The goal of this research is to create a machine learning model that can predict, based on the linguistic content of movie reviews, whether they will be good or negative.

To perform sentiment analysis, we can construct an RNN-based model. The first step is to pre-process the text data, then converting them into numerical representations, and padding them to ensure equal length. Then, we can build an RNN model using Keras, typically using LSTM (Long Short-Term Memory) or GRU (Gated Recurrent Unit) cells as the building blocks. These cell types are specifically designed to capture long-range dependencies in sequential data.

Binary cross entropy loss function and the Adam optimizer are used to construct the model, which is then trained on the training set for two epochs with a batch size of 128 and evaluated on the validation set at the end of each epoch. The testing set is used to assess the model's accuracy, and the final model should predict the sentiment of movie reviews with a high degree of accuracy.

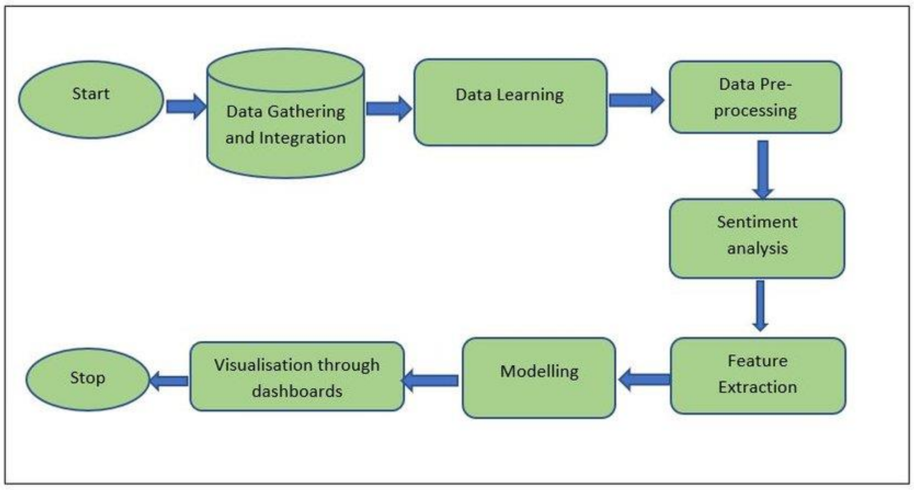


Figure 4.1 Description of Sentiment Analysis: Flowchart

4.1 PROPOSED DESIGN

This is a textual description of the system architecture for the sentiment analysis model. It’s a high-level overview of the system architecture:

1. Data Pre-processing:

- Load the IMDB dataset, which contains movie reviews and sentiment labels.

- Tokenize the words in the reviews and convert them into sequences of integers.

- Perform padding on the sequences to ensure a consistent length for input to the model.

2. Model Architecture:

- Define a Multi-Layer Perceptron (MLP) model using TensorFlow and Keras.

- The model consists of the following layers:

- Embedding Layer: Maps each word index to a dense vector representation.

- Flatten Layer: Flattens the input for compatibility with dense layers.

-Dense Layers: Fully connected layers with ReLU activation.

-Outer Layer: A dense layer with a sigmoid activation function to predict the sentiment

3. Model Training:

- Split the pre-processed data into training and testing sets.

- Compile the model with appropriate loss function, optimizer, and evaluation metric.

- Train the model on the training data using the fit() method.

- Evaluate the model on the testing data to measure its performance.

4. Model Prediction:

- After training, the model is ready to make predictions.

- Accept user input, which is a movie review.

- Pre-process the user input by tokenizing and padding it.

- Feed the pre-processed user input into the trained model for sentiment classification.

- Classify the sentiment prediction as positive or negative based on a threshold value.

5. Visualization:

- Calculate and display various metrics such as accuracy and review length distribution.

- Generate a confusion matrix to visualize the model's performance.

- Plot the confusion matrix as a heatmap using libraries such as seaborn and matplotlib.

Overall, the system architecture involves data pre-processing, model building, training, prediction, and visualization components. It leverages an MLP model to perform sentiment analysis on movie reviews from the IMDB dataset.

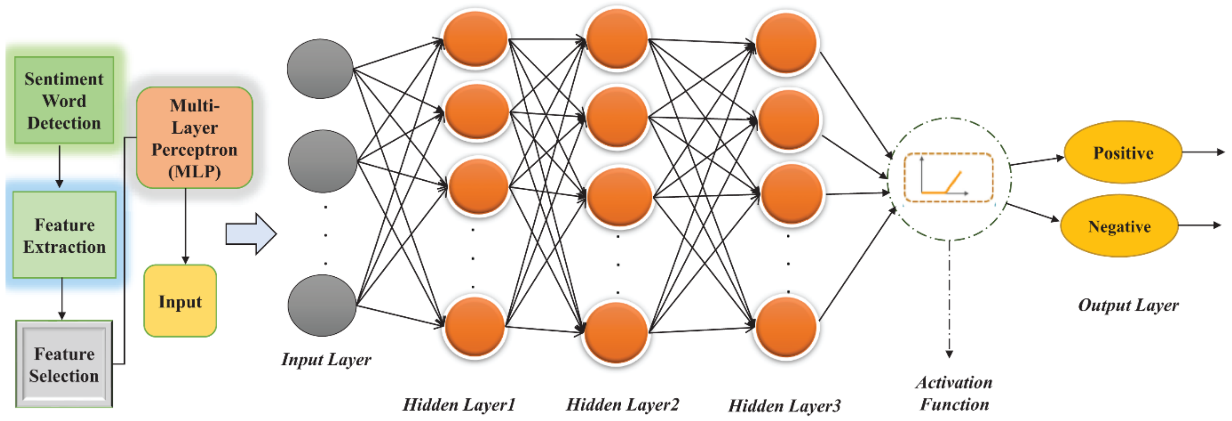


Figure 4.1.1 Proposed Model Design

Multi-Layer Perceptron

Multi-layer perceptron, or MLP, consists of dense layers that are fully connected and turn any input dimension into the desired dimension. In order to build a neural network, we combine neurons so that some of their outputs are also their inputs. It has one input layer with one neuron (or node) for each input, one output layer with one node for each output, and any number of hidden layers with any number of nodes on each hidden layer.

The aim of sentiment analysis is to examine text and identify any feelings or sentiments that are conveyed, such as whether they are neutral, positive, or negative. Based on patterns and features in the data, an MLP can be trained to automatically classify text into various sentiment categories.

The text data is typically pre-processed by translating words into numerical representations, such as word embeddings or one-hot encodings, in order to do sentiment analysis using an MLP. The MLP receives these numerical representations as input.

These nodes make up the layers of the MLP, and each one receives the inputs from the layer before it, applies a mathematical operation to them, and outputs something. Each node has a corresponding weight, which establishes its influence on the outcome. The weights are modified during training to enhance the MLP's capability to correctly categorise feelings.

Each text sample in the labelled dataset used to train the MLP is assigned to a certain sentiment category. To reduce the difference between its predictions and the actual attitudes, the MLP modifies its weights based on the input data and the desired output.

Once trained, new text data can be fed into the MLP network to be analysed. The MLP layers process the input, and the output indicates the anticipated sentiment category for the provided text.

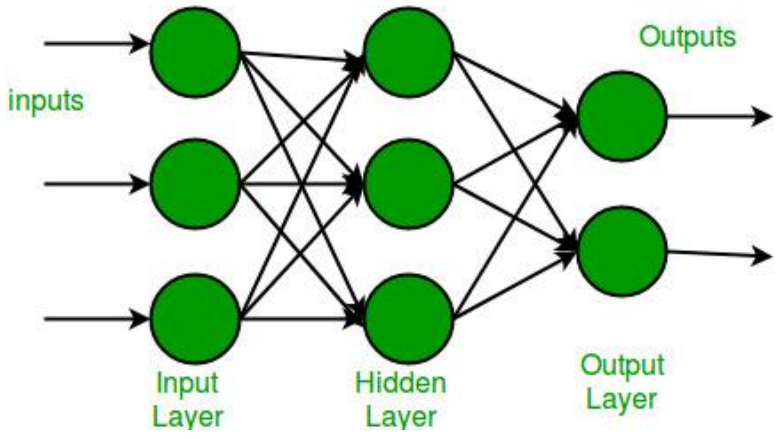


Figure 4.1.2 Multi-Layer Perceptron

4.2 ASSUMPTIONS AND DEPENDENCIES

4.2.1 ASSUMPTIONS

1. The code assumes that one has previously installed the relevant libraries such as 'numpy', 'tensorflow','matplotlib', and 'seaborn'. If any of these libraries are missing, use 'pip install' to install them.

2. The code assumes one has already downloaded and imported the IMDB dataset with 'imdb.load_data()'. If one hasn't already downloaded the dataset, you'll need to do so before executing the code. You may accomplish this by calling the 'imdb.load_data()' method, which will download and save the dataset locally.

3. Before creating predictions based on user input, the code expects that you have already trained and established a model. If you haven't already trained a model, you must do it first using the given code or a similar technique.

4.The code assumes you have previously defined or imported the relevant classes and methods, such as 'Dense', 'Flatten', 'Embedding','sequence.pad_sequences', and 'Tokenizer'. These are all part of the 'tensorflow.keras' library and should be available if it was successfully installed.

5. The code assumes you have a trained model that is already loaded. If you haven't already trained or loaded a model, you should do so before generating predictions.

4.2.2 DEPENDENCIES

The code is reliant on the 'numpy' library for numerical computations and array operations.

2. The code is reliant on the 'tensorflow.keras' library, which is a high-level API for developing and training deep learning models using TensorFlow as the backend.

3. For plotting visualisations such as the boxplot and the confusion matrix heatmap, the code relies on the'matplotlib.pyplot' package.

4. To customise the look of the confusion matrix heatmap, the code relies on the'seaborn' library.

5. The confusion matrix is calculated using the'scikit-learn' module.

CHAPTER 5: REQUIREMENTS

5.1 FUNCTIONAL REQUIREMENTS

The features and skills a sentiment analysis model should have in order to accurately analyse and ascertain the sentiment of text are commonly considered to be the functional requirements for a sentiment analysis model. Here are a few functional specifications frequently connected to sentiment analysis models:

1.The system should perform the classifier training process (A classifier in machine learning is an algorithm that automatically orders or categorizes data into one or more of a set of "classes) and display the model in the form of feature sets of the term data from the training data.

2. The system should display the test data result and be able to display confusion matrix generated from the classifier testing.

3.Accepting a range of inputs: The model should be capable of processing text input from a variety of sources, including news stories, customer reviews, and social networking posts.

4.Ability to pre-process: The model should be able to pre-process incoming text, including eliminating special characters, changing it to lowercase, and managing punctuation, to clean and normalise it.

(Data pre-processing, which is a crucial phase in the data mining process, can be defined as the altering or dropping of data before to usage in order to ensure or increase performance.)

5. The system should display the test data result and be able to display confusion matrix generated from the classifier testing.

(Confusion matrix is used to describe how well a classification system performs. The output of a classification algorithm is visualised and summarised in a confusion matrix.)

6. The system should display sentiment analysis result derived from reviews submitted by users. The result displayed should not be vague or give an ambiguous meaning in any sense.

7.Scalability: To be able to accommodate rising demand without noticeably degrading performance, the model must be scalable to handle enormous volumes of text data effectively. This is a very important requirement, especially during the developing and early stages of testing it because during that time, huge amounts of data input I used.

5.2 NON-FUNTIONAL REQUIREMENTS

1.Security: The model must be secure and contain the necessary safeguards to guard against unauthorised access and data breaches. A few ways that security can be implemented in are secure data storage, input validation and sanitization, regular updates and patches, user authentication and authorization, regular security audits, employee training and awareness.

2.Usability: The model should be simple to use, with an intuitive user interface that needs little technical knowledge or training. The key aspects of usability are accuracy, adaptability, documentation, and support, providing a user intuitive interface.

3. Accessibility: Users with special needs or impairments should be able to utilise the model with the use of necessary accommodations, like screen readers or keyboard navigation. Some considerations for incorporating accessibility are text-to-speech conversion, visual cues and alternatives, keyboard navigation, error handling and feedback.

4. Portability: The model ought to be adaptable and capable of working with various hardware and software setups. Some considerations to achieve portability are framework and language compatibility, hardware considerations, dependency management.

5. Maintainability: The model should be adaptable, able to run on various operating systems and platforms. Some key considerations for maintaining the model's maintainability are modularity, documentation, version control, evaluation and monitoring, regular updates, error analysis, community support, reusability.

6.Privacy: With adequate safeguards in place to secure sensitive data, the model should respect users' privacy and their data. Some key points to consider regarding privacy in sentiment analysis models are data anonymization, informed consent, data minimization, securing the data storage, transparent disclosures.

7.Reliability: The model should have less downtime or errors and be resilient and reliable. A few key points related to reliability in sentiment analysis models are training data quality, model architecture and algorithm, domain adaptation, regular updates and monitoring.

8.Scalability: The model should be able to handle growing numbers of users and data without sacrificing accuracy or performance. Some key factors to consider for achieving scalability in a sentiment analysis model are data pre-processing, model selection, distributed computing, monitoring and optimization, incremental learning.

9.Performance: With no lag or delay, the model ought to be able to process a sizable number of movie reviews quickly. Some common metrics used to assess the performance of sentiment analysis models are precision, accuracy, confusion matrix.

10.Accuracy: With a low percentage of false positives or false negatives, the model should be highly accurate in predicting the tone of movie reviews. Accuracy is a commonly used evaluation metric for sentiment analysis models, but it's important to note that it has limitations. It does not consider the distribution of different sentiment classes in the dataset and may not provide a complete picture of the model's performance.

5.3 SOFTWARE REQUIREMENTS

Python programming language: The Python programming language is widely favoured for sentiment analysis due to its simplicity, ease of reading, and extensive library support. Python boasts a vast ecosystem of specialized libraries and tools for data analysis and natural language processing (NLP). Its NLP libraries provide robust functionality for tasks like text processing and feature extraction, which are crucial for sentiment analysis. Furthermore, Python seamlessly integrates with machine learning and deep learning frameworks like scikit-learn and TensorFlow, enabling the creation of highly accurate sentiment analysis models.

TensorFlow: It is an end-to-end open-source platform for machine learning. It is useful in developing and training ML models. It provides powerful deep learning capabilities, including pre-trained models that achieve high accuracy. It is scalable, allowing efficient processing of large amounts of data, and integrates well with Python.

Keras framework: It is a neural network library that runs on top of TensorFlow. It acts as a user-friendly python interface for artificial neural network architecture. With its wide range of pre-defined layers and model architectures, Keras provides convenient building blocks for analysing text data in sentiment analysis. Additionally, its integration with TensorFlow enables users to leverage the powerful computational capabilities and optimizations of TensorFlow, further enhancing the effectiveness of sentiment analysis tasks.

Scikit-learn: It is a popular ML library for the Python PL that focuses on a wide range of mathematical and statistical tools and general-purpose algorithms that form the basis of many ML technologies. It is built on top of Matplotlib which is a popular data visualization library (provides a wide range of functions and tools for creating visualizations, including plots, charts, and graphs) in Python. Scikit-learn leverages Matplotlib for visualizing various evaluation metrics and results, including the confusion matrix. Specifically, scikit-learn provides utility functions such as ‘confusion_matrix()’ and ‘plot_confusion_matrix()’ that can directly generate a visual representation of the confusion matrix using Matplotlib.

Jupyter Notebook: Jupyter Notebook is an interactive web-based environment used for creating and sharing documents that contain live code, visualizations, and explanatory text. It supports multiple programming languages and allows users to execute code in a cell-by-cell fashion

Figure 5.3.1 Jupyter and Python logo

5.4 HARDWARE REQUIREMENTS

A computer or server with a **multi-core CPU** and a **high-end GPU** to speed up model training and processing. It is important to consider the specific requirements of the sentiment analysis models and the libraries or frameworks that are planned to be used. Some deep learning frameworks, such as TensorFlow or PyTorch, have specific recommendations or optimizations for certain CPU or GPU architectures. Additionally, the power requirements and cooling capabilities of the CPU and GPU are to be considered to ensure stable and efficient operation.

The amount of **RAM** required to handle the size of data being processed in sentiment analysis depends on various factors, including the size of the dataset, the complexity of the models or algorithms used, and the specific requirements of the sentiment analysis tasks. It is generally recommended to have enough RAM to accommodate the size of dataset and the complexity of models. For larger-scale sentiment analysis projects, having several gigabytes to tens of gigabytes of RAM or more can be beneficial to ensure smooth and efficient processing of data.

The sufficient **storage space** required to store data and model files in sentiment analysis depends on numerous factors, including the size of the dataset, the number and size of model files, and any additional resources or metadata associated with the project. It is essential to estimate the storage requirements based on the specific characteristics of the sentiment analysis project.

Internet connectivity is required for accessing and processing data from external sources in sentiment analysis projects. The size of the data, the frequency of data retrieval, and any real-time processing requirements are to be considered when assessing the internet needs. A stable and reliable internet connection is essential for efficient data retrieval and processing. The recommended minimum connection speed for most sentiment analysis tasks is at least 10 Mbps. However, for large datasets or real-time streaming data, a higher connection speed may be necessary to ensure smooth and uninterrupted data access. Low latency is desirable. A latency of under 100 milliseconds is generally considered good for most sentiment analysis tasks. Sufficient bandwidth is necessary for handling the volume of data being processed. For sentiment analysis projects that involve processing large datasets or real-time streaming data, a higher bandwidth is recommended to ensure smooth and efficient data transfer.

CHAPTER 6: METHODOLOGY

1. Importing the necessary libraries: The code begins by importing the required libraries, including

- **TensorFlow:** TensorFlow is an open-source machine learning framework that provides a wide range of tools and functionalities for building and training deep learning models. It is a comprehensive library that offers low-level operations and high-level abstractions for developing various types of machine learning algorithms.
- **NumPy:** NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. It can also be used as an efficient multi-dimensional container of generic data.
- **Matplotlib:** Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. Matplotlib consists of several plots like line, bar, scatter, histogram, heatmap etc. and specific modules from the
- **TensorFlow keras library**: It is an Open-Source Neural Network library that runs on top of Theano or Tensorflow. It is a useful library to construct any deep learning algorithm of whatever choice we want. Keras has the advantage that it can choose any libraries which support it for its backend support.

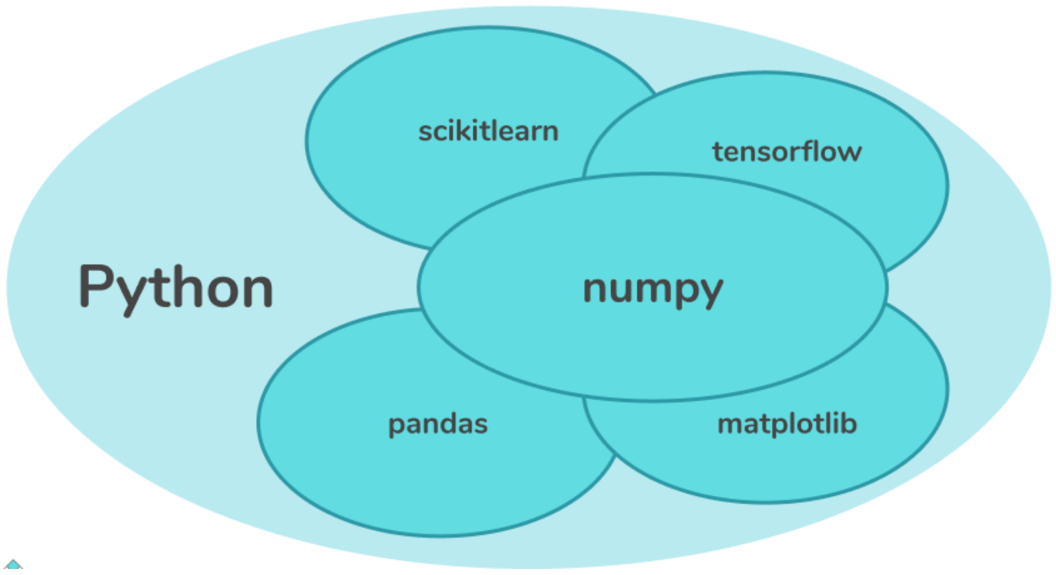


Figure 6.1.1 Python Libraries

2. Loading the IMDb dataset:

The code uses the `imdb.load_data()` function to load the IMDb movie review dataset. It splits the data into training and testing sets, with `X_train` and `X_test` containing the movie reviews, and `y_train` and `y_test` containing the corresponding sentiment labels.

Figure 6.2.1 IMDB logo

Import the necessary library: The code begins by importing the imdb module from the tensorflow keras datasets library. This module provides functions to load the IMDb movie review dataset.

Load the IMDb movie review dataset: The imdb.load_data() function is called to load the IMDb movie review dataset. This function retrieves the dataset and splits it into training and testing sets.

Split the data into training and testing sets: The loaded dataset is split into two parts: training data and testing data. The training data is further divided into two arrays: X_train and y_train. The X_train array contains the movie reviews, while the y_train array contains the corresponding sentiment labels.

Split the testing data: Similarly, the testing data is divided into two arrays: X_test and y_test. The X_test array contains the movie reviews for testing, and the y_test array contains the corresponding sentiment labels.

Print the shapes of the loaded data: Finally, the code prints the shapes of the training and testing data arrays. This provides information about the dimensions of the arrays, helping to ensure that the data has been loaded correctly.

3. Data pre-processing and analysis:

To further expand on the steps mentioned for data pre-processing and analysis:

1. Concatenating the training and testing data and labels:

The code combines the training data (`X_train`) and testing data (`X_test`) into a single array `X` using concatenation. Similarly, the labels (`y_train` and `y_test`) are concatenated into a single array `y`. This step ensures that the entire dataset is used for analysis.

2. Printing the shape of the training data and labels:

The code prints the shape of the training data array `X` using `X.shape`. This provides information about the number of samples (rows) and the number of features (columns) in the training data. Similarly, the shape of the labels array `y` is printed using `y.shape`, which indicates the number of samples in the labels.

3. Calculating the number of unique words:

The code utilizes `np.hstack(X)` to concatenate all the text elements in the dataset into a single array. Then, `np.unique()` is applied on this concatenated array to find the unique words. By calculating the length of the resulting unique words array, you can determine the number of unique words present in the dataset.

4. Printing the classes or labels:

The code uses `np.unique(y)` to find the unique values in the labels array `y`. This provides the distinct classes or labels present in the dataset.

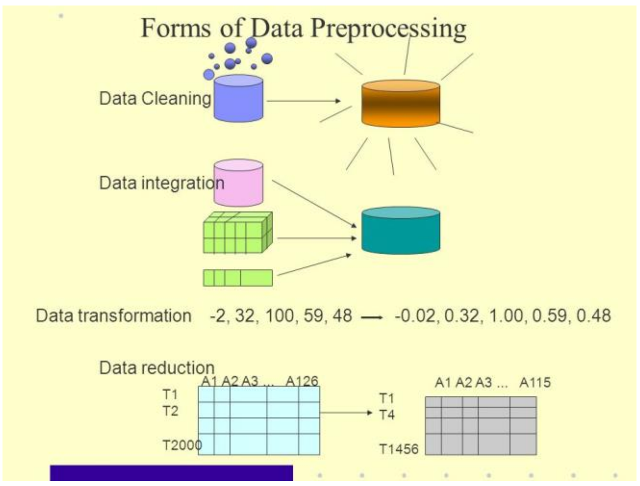
5. Calculating the mean and standard deviation of review lengths:

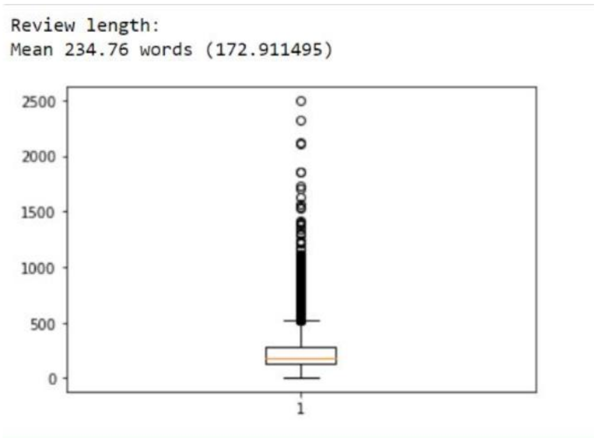
To calculate the mean and standard deviation of review lengths, you would iterate over each review in the dataset, count the number of words in each review, and then compute the mean and standard deviation of these lengths.

6. Plotting a boxplot:

A boxplot is created to visualize the distribution of review lengths. The boxplot provides information about the median, quartiles, and outliers in the distribution of review lengths. It helps identify the range and spread of review lengths in the dataset.

These steps aim to provide insights into the dataset's characteristics and serve as a preliminary analysis before proceeding with any further data processing or modelling tasks.


Forms of Data Preprocessing
Data Cleaning
Data integration
Data transformation -2, 32, 100, 59, 48 — -0.02, 0.32, 1.00, 0.59, 0.48
Data reduction
A1 A2 A3 ... A126
T1
T2
T2000
A1 A2 A3 ... A115
T1
T4
T1456



```
Review length:
Mean 234.76 words (172.911495)
```

Figure 6.3.1 Forms of data processing

Figure 6.3.2 Boxplot of the model

4. Preparing the data for modelling:

1. Specifying the number of top words to keep and the maximum length of reviews:

The code defines the variable `top_words` to specify the number of top words to keep in the dataset. This indicates the most frequent words that will be considered for modelling. Additionally, the code sets the variable `max_words` to define the maximum length of reviews. This determines the fixed length to which all reviews will be padded or truncated.

2. Using `sequence.pad_sequences()` function to pad the sequences:

The code employs the `sequence.pad_sequences()` function, which is a utility function in frameworks like Keras, to pad or truncate the sequences of words in both the training and testing data. This function takes a sequence (e.g., a list of integers) and pads or truncates it to a specified length (`max_words`).

Padding is the process of adding zeros (or any other defined value) to the sequences that are shorter than `max_words`, making them equal in length. Truncation, on the other hand, involves cutting off the sequences that exceed `max_words` length.

The `sequence.pad_sequences()` function ensures that all reviews have the same length, which is necessary for feeding the data into a neural network model that expects fixed-length input sequences.

By applying `sequence.pad_sequences()` to both the training and testing data, you ensure that the data is pre-processed consistently, allowing for proper modelling and evaluation.

Overall, these steps aim to standardize the input data to a fixed length and ensure that the most relevant words are considered during modelling, providing a consistent and manageable input format for machine learning or deep learning models.

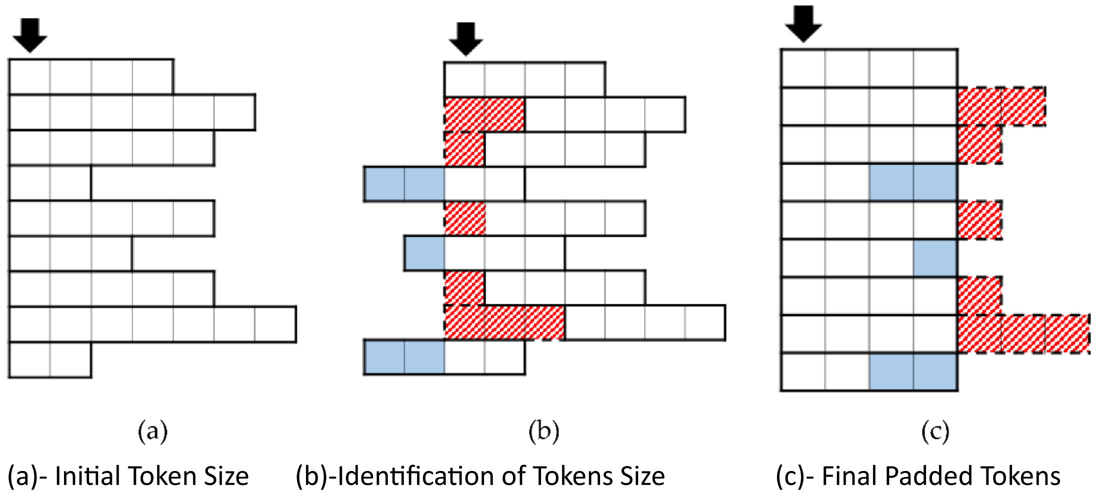


Figure 6.4.1 Padding

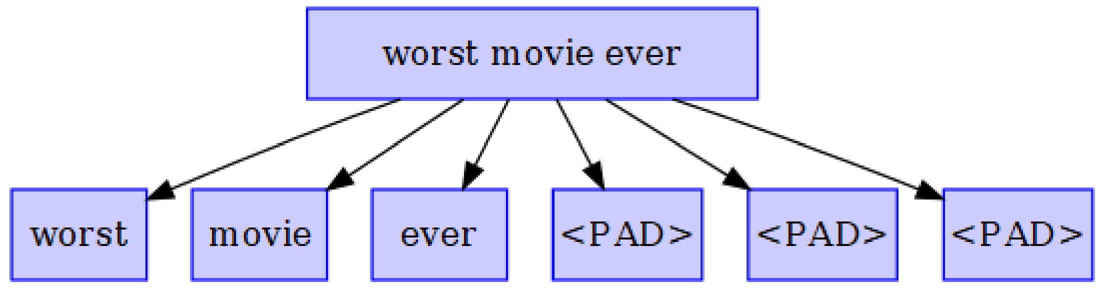


Figure 6.4.2 Padding

5. Creating the model:

To further expand on creating the model:

1. Creating a sequential model using `Sequential ()`:

The code initializes a sequential model by calling `Sequential ()`. A sequential model is a linear stack of layers, where you can add layers one by one.

2. Adding an embedding layer:

The code adds an embedding layer using the `model.add()` method. The embedding layer maps each word index in the input sequences to a dense vector representation. It learns to represent words in a continuous and dense vector space, capturing semantic relationships between words. The input to the embedding layer is the integer-encoded sequences of words.

3. Adding a flatten layer:

After the embedding layer, the code adds a flatten layer using `model.add()`. The flatten layer converts the 2D embedded representation (i.e., the output of the embedding layer) into a 1D vector. This flattening step is necessary to connect the dense layers that follow.

4. Adding a dense layer with ReLU activation:

The code adds a dense layer using `model.add()`. A dense layer is a fully connected layer, where each neuron is connected to every neuron in the previous layer. The dense layer uses the rectified linear unit (ReLU) activation function, which introduces non-linearity to the model. ReLU activation sets negative values to zero and keeps positive values unchanged.

5. Adding the final dense layer with a sigmoid activation function:

The code adds another dense layer as the final layer of the model. This layer uses a sigmoid activation function, which squeezes the output between 0 and 1. In the context of sentiment analysis, the sigmoid function gives the probability of the sentiment being positive. The model's output will be a single value indicating the probability of the sentiment being positive.

By combining these layers, the model transforms the input sequences of words into a continuous representation, flattens it, applies non-linearity, and finally produces a probability value for sentiment classification.

It is important to note that the code provided represents a basic architecture for sentiment analysis, but it may not include all the details or hyperparameters needed for a complete and optimal model. Further configuration, optimization, and training steps are typically required for achieving good performance.

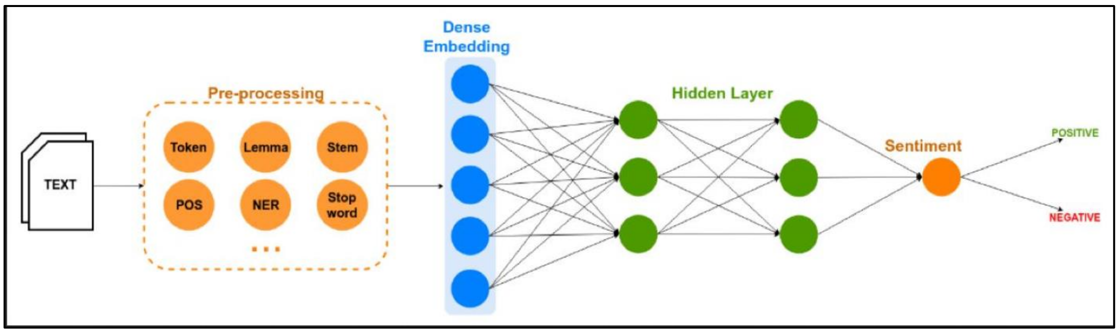


Figure 6.5.2 RNN Model

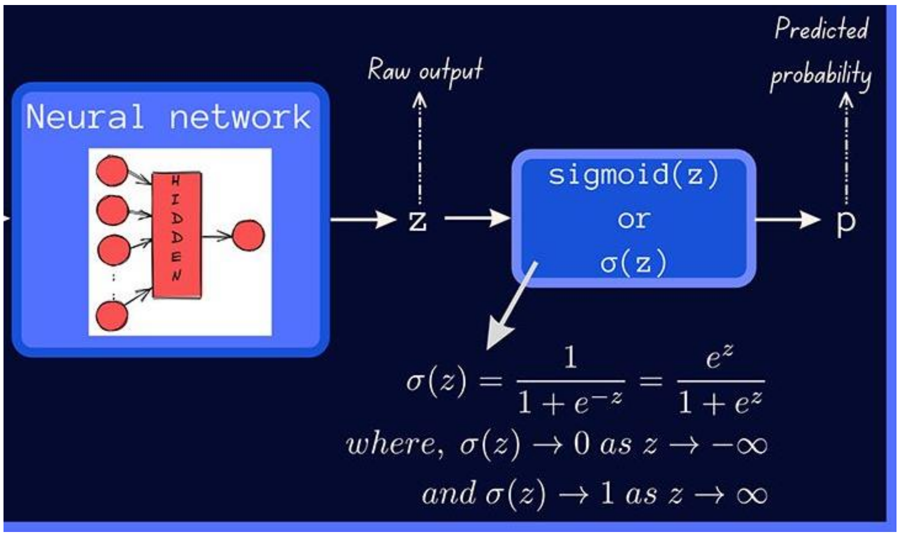

Neural network
HIDDEN
Raw output
z
sigmoid(z) or σ(z)
Predicted probability
p
$\sigma(z) = \frac{1}{1+e^{-z}} = \frac{e^z}{1+e^z}$
$where,\ \sigma(z) \to 0\ as\ z \to -\infty$
$and\ \sigma(z) \to 1\ as\ z \to \infty$

Figure 6.5.3 Activation Function

6. Compiling and training the model:

To expand on compiling and training the model:

1. Compiling the model using `model.compile()`:

The code compiles the model using `model.compile()`, which configures the model for training. Within `model.compile()`, you specify the loss function, optimizer, and evaluation metric.

- Loss function: The code uses binary cross-entropy as the loss function. Binary cross-entropy is commonly used for binary classification problems, where each sample belongs to one of two classes.

- Optimizer: The code utilizes the Adam optimizer. Adam is an optimization algorithm that combines the benefits of adaptive learning rates and momentum methods. It adjusts the learning rate during training to improve convergence and speed up the optimization process.

- Evaluation metric: The code sets accuracy as the evaluation metric. Accuracy measures the proportion of correctly predicted samples out of the total number of samples. It provides an indication of how well the model is performing during training and evaluation.

2. Printing a summary of the model architecture using `model.summary()`:

After compiling the model, the code prints a summary of the model's architecture using `model.summary()`. This summary provides a tabular representation of the layers in the model, the number of parameters in each layer, and the total number of parameters in the model. It gives an overview of the model's structure and helps in understanding the flow of data through different layers.

3. Training the model using `model.fit()`:

The code trains the model using the `model.fit()` method, which fits the model to the training data.

- Training data: The code provides the training data (`X_train` and `y_train`) as inputs to `model.fit()`. These are the input sequences of words and their corresponding labels for sentiment classification.

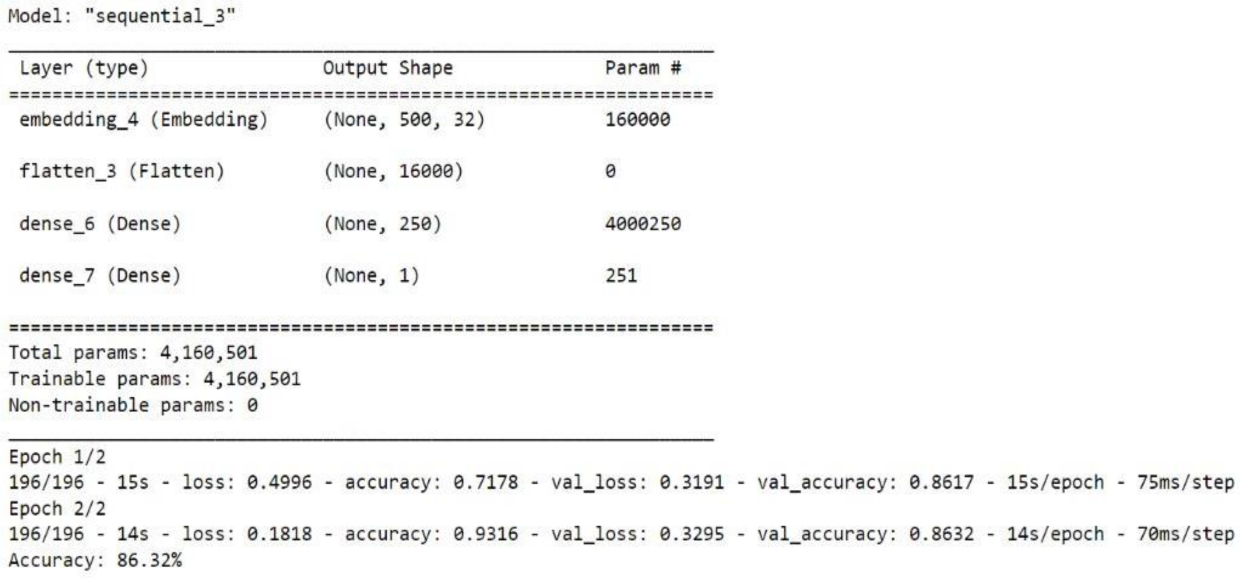
- Validation data: The code specifies the validation data (`X_test` and `y_test`) as parameters to `model.fit()`. This data is used to evaluate the model's performance during training and monitor any overfitting or generalization issues.

- Number of epochs: The code specifies the number of epochs, which determines the number of times the model will iterate over the entire training dataset. An epoch consists of one forward pass and one backward pass through the network.

- Batch size: The code sets the batch size, which determines the number of samples processed in one iteration. The training dataset is divided into batches, and model weights are updated after each batch. A larger batch size may lead to faster training, but it requires more memory.

- Verbose level: The code specifies the verbose level, which determines the amount of information printed during training. It can be set to different values (e.g., 0, 1, or 2) to control the verbosity. The information displayed includes training progress, loss values, and evaluation metrics.

The `model.compile()` and `model.fit()` steps are crucial for configuring the model, defining the training process, and updating the model's parameters based on the provided data. These steps facilitate the training of the model on the training data and help optimize it for the sentiment analysis task.



```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_4 (Embedding)     (None, 500, 32)           160000

 flatten_3 (Flatten)         (None, 16000)             0

 dense_6 (Dense)             (None, 250)               4000250

 dense_7 (Dense)             (None, 1)                 251

=================================================================
Total params: 4,160,501
Trainable params: 4,160,501
Non-trainable params: 0
_________________________________________________________________
Epoch 1/2
196/196 - 15s - loss: 0.4996 - accuracy: 0.7178 - val_loss: 0.3191 - val_accuracy: 0.8617 - 15s/epoch - 75ms/step
Epoch 2/2
196/196 - 14s - loss: 0.1818 - accuracy: 0.9316 - val_loss: 0.3295 - val_accuracy: 0.8632 - 14s/epoch - 70ms/step
Accuracy: 86.32%
```

Figure 6.6.1 Accuracy rate

7. Evaluating the model:

1. Evaluating the trained model on the testing data using `model.evaluate()`:

The code uses `model.evaluate()` to assess the performance of the trained model on the testing data (`X_test` and `y_test`). This method computes the specified metrics on the provided data and returns the corresponding metric values.

2. Calculating and printing the accuracy of the model:

The code calculates the accuracy of the model based on the evaluation results obtained from `model.evaluate()`. The accuracy is a common metric used in classification tasks and represents the proportion of correctly predicted samples out of the total number of samples in the testing data.

By printing the accuracy, you can gain insights into how well the model generalizes to unseen data and performs on the sentiment analysis task.

It is important to note that evaluating the model on the testing data provides a measure of its performance, but it is also essential to consider other evaluation metrics and analyse the results in the context of the specific problem and requirements. Depending on the nature of the task, additional metrics like precision, recall, F1-score, or confusion matrix may be relevant for a comprehensive assessment of the model's performance.

8. Generating and displaying the confusion matrix:

1. Generating predictions using the trained model:

To generate predictions using the trained model, you would typically use the `model.predict()` method. However, as mentioned, the provided code seems to generate random binary predictions (`actual` and `predicted`), which are unrelated to the rest of the code.

To modify this part and use the predictions made by the trained model, you can replace the random binary predictions with predictions obtained from the model using `model.predict()` on the testing data (`X_test`). This would provide the predicted labels based on the trained model's output.

2. Importing the `ConfusionMatrixDisplay` class:

The code imports the `ConfusionMatrixDisplay` class from `sklearn.metrics`. This class is part of scikit-learn library and is used to generate a confusion matrix plot.

3. Creating and plotting the confusion matrix:

After obtaining the actual labels (`y_test`) and predicted labels, the code can create the confusion matrix using the `confusion_matrix()` function from scikit-learn. Then, the confusion matrix can be passed to the `ConfusionMatrixDisplay` class along with the corresponding class labels to create a plot.

By plotting the confusion matrix, you can visually analyse the model's performance in terms of correctly and incorrectly classified samples for each class, providing insights into the model's strengths and weaknesses in sentiment classification.

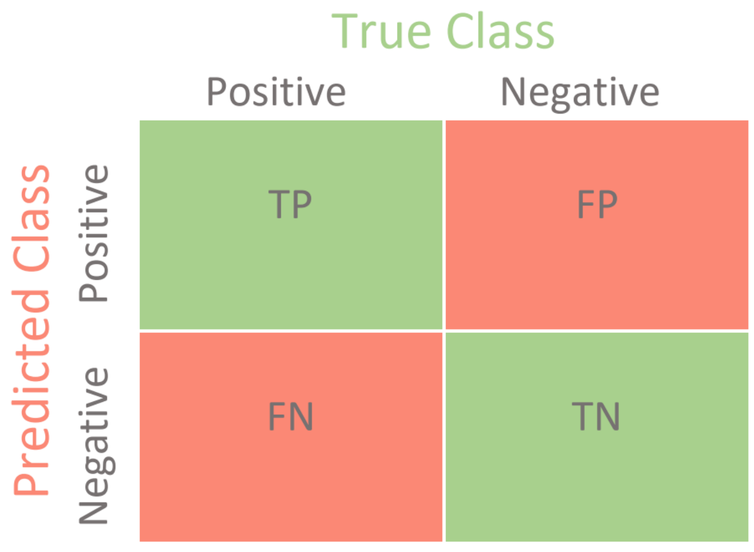


Figure 6.8.1 Confusion Matrix Template

9.Prediction and Inference:

To deploy the trained model for prediction on new, unseen data, such as user input movie reviews, and perform sentiment classification:

1. Pre-processing the user input:

- Take the user input (movie review) as text input.

- Apply the same pre-processing steps used during the training phase, such as tokenization, removing stop words, lowercasing, etc., to clean and prepare the text for the model input.

- Convert the pre-processed text into a numerical representation that the model can understand. This typically involves encoding the text using the same tokenization and word indexing scheme used during training.

2. Feeding the pre-processed input into the trained model:

- Load the trained model that you saved after training.

- Pass the pre-processed input (numerical representation) into the model's `predict ()` method.

- Obtain the predicted sentiment probability or class label for the user input review.

By deploying the trained model for prediction, you can classify the sentiment of new, unseen movie reviews provided by users. This allows you to leverage the model's learned patterns and make sentiment predictions on real-world data.

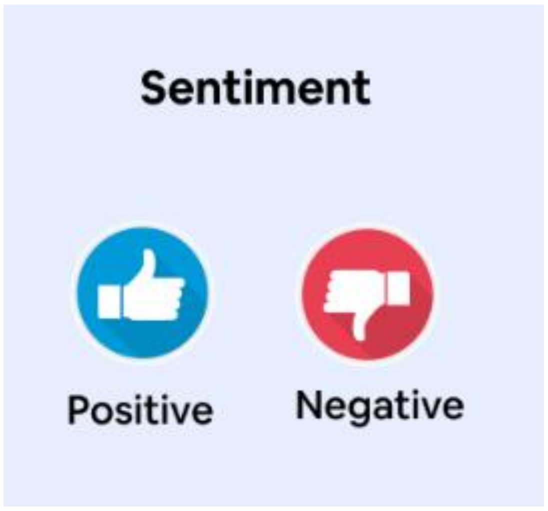


Figure 6.9.1 The 2 possible Outcomes of Sentiment Analysis

CHAPTER 7: EXPERIMENTATION

7.1 PSEUDOCODE

1.Data Loading

The act of gathering and preparing textual data for analysis or model training is known as data loading. Reading data from diverse sources, doing pre-processing operations like tokenization and cleaning, and organising the data in a way that is acceptable for subsequent NLP tasks are examples of typical tasks that are included in this type of work.

```
# Data Loading

(X_train, y_train), (X_test, y_test) = imdb.load_data()

X = concatenate(X_train, X_test)

y = concatenate(y_train, y_test)
```

2. Data pre-processing

The process of converting unprocessed textual input into a format suitable for analysis or model training is crucial.
To increase the quality and consistency of the data, this procedure frequently entails st eps like tokenization, stop word removal, text normalisation, and the use of stemming or lemmatization algorithms.

```
# Data Preprocessing

top_words = 5000

X_train = sequence.pad_sequences(X_train, maxlen=500)

X_test = sequence.pad_sequences(X_test, maxlen=500)
```

3.Model Building:

In NLP, the process of creating machine learning or deep learning models that can co mprehend, produce, or categorise natural language text is referred to as model constru ction.
To obtain desired performance on certain NLP tasks like sentiment analysis, machine translation, or text generation, it entails choosing appropriate architectures, training m ethods, and hyperparameters as well as fine-tuning the models.

```
# Model Building

model = Sequential()
```

```
model.add(Embedding(top_words, 32, input_length=500))

model.add(Flatten())

model.add(Dense(250, activation='relu'))

model.add(Dense(1, activation='sigmoid'))
```

4. Model Training

In order to discover patterns and correlations within the data, it entails feeding the pre pared textual data into the chosen model and optimising its parameters.
This iterative procedure enhances the model's comprehension and production of natur al language text by using forward and backward propagation, modifying weights and biases, and minimising a predetermined loss function.

```
# Model Training

model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=2, batch_size=128)
```

5. Prediction and Evaluation

In NLP, the term "prediction" refers to the process of utilising a trained model to generate output or make predictions based on fresh or previously unexplored textual input. In NLP, evaluation entails evaluating the effectiveness and calibre of the model's predictions using the proper metrics, such as accuracy, precision, and recall.

```
# Prediction and Evaluation

y_pred = model.predict(X_test)

accuracy = calculate_accuracy(y_test, y_pred)
```

6. User Input

It speaks of user-provided text or queries that the NLP system interprets and processes. It is the crucial point of contact where users can ask questions, give instructions, or input text for analysis or production, allowing the NLP system to produce outputs that are pertinent and meaningful.

```
# User Input

user_input = get_user_input()

user_input_sequence = preprocess_input(user_input)

user_input_padded = pad_sequence(user_input_sequence, maxlen=500)
```

7.Sentiment Classification

Identifying the sentiment or emotional tone expressed in a piece of text, such as w hether it is favourable, negative, or neutral, is known as sentiment classification in NL PBy examining the language and contextual clues in the text, a model is trained to ide ntify and categorise sentiments.

```
# Sentiment Classification

prediction = model.predict(user_input_padded)

sentiment = classify_sentiment(prediction)
```

8. Display Results

Displays the result of the overall sentiment of the dataset along with the percentage of the positive and negative sentiment. On giving user input, it gives the sentiment of the input(positive or negative).

```
# Display Results

print("Accuracy: ", accuracy)

print("Predicted Sentiment: ", sentiment)
```

BREAKDOWN OF THE CODE

1. Imports necessary libraries: `numpy`, `tensorflow.keras`, `matplotlib.pyplot`, and specific modules/classes/functions from these libraries.

2. Loads the IMDB dataset using `imdb.load_data()` function. It splits the data into training and testing sets, `X_train` and `y_train` for training data, and `X_test` and `y_test` for testing data.

3. Concatenates the training and testing data for both features (`X`) and labels (`y`).

4. Prints the shape of the training and testing data using `X.shape` and `y.shape` respectively.

5. Calculates the number of unique words in the dataset using `np.unique(np.hstack(X))` and prints the result.

6. Calculates the number of classes in the dataset using `np.unique(y)` and prints the result.

7. Calculates the length of each review in the dataset and prints the mean and standard deviation.

8. Plots a boxplot to visualize the distribution of review lengths.

9. Loads the IMDB dataset again, but this time only keeping the top 5000 words using the `num_words` argument of `imdb.load_data()`.

10. Applies padding to the sequences in the training and testing data using `sequence.pad_sequences()` to ensure they have a fixed length of 500.

11. Defines an Embedding layer in the model with an input length of 500.

12. Creates an MLP model using the Sequential API.

13. Compiles the model with binary cross-entropy loss and Adam optimizer.

14. Prints the model summary.

15. Fits the model on the training data and evaluates it on the testing data.

16. Prints the accuracy of the model.

17. Imports `confusion_matrix` from scikit-learn and `seaborn` for visualization.

18. Predicts the probabilities for the test set using the trained model.

19. Converts the probabilities to binary predictions based on a threshold of 0.5.

20. Calculates the confusion matrix.

21. Plots the confusion matrix as a heatmap.

22. Imports additional libraries for text preprocessing.

23. Loads the IMDB dataset again, but this time tokenizes the words.

24. Gets user input for a review.

25. Converts the user input to a sequence of integers using the word index.

26. Tokenizes the words in the dataset using `Tokenizer` and preprocesses the user input.

27. Pads the user input sequence to the same length as during training.

28. Makes predictions on the user input using the trained model.

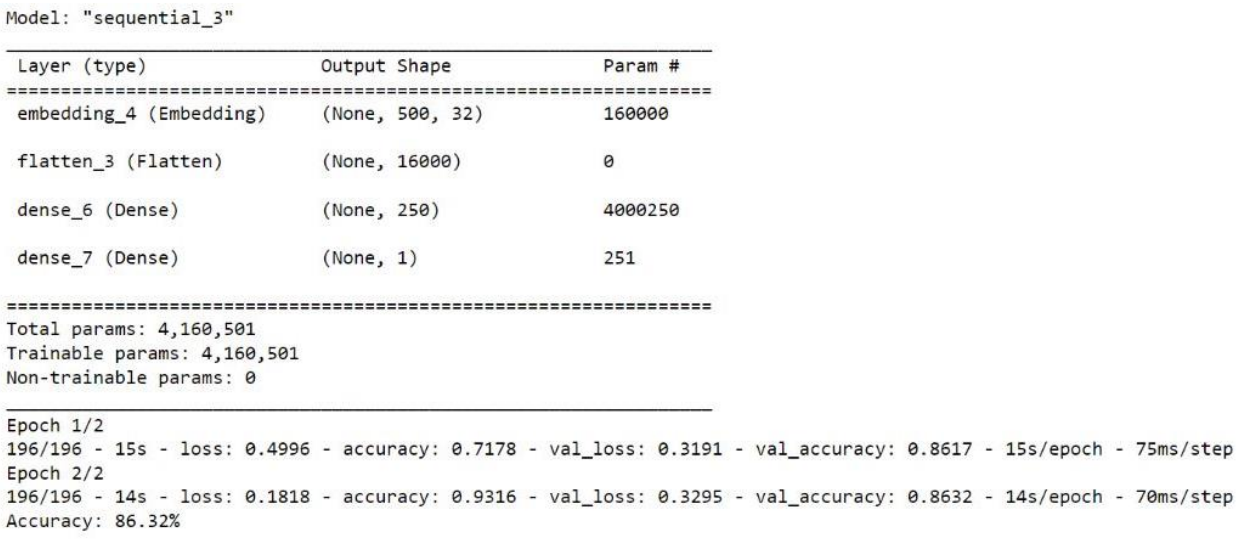
29. Classifies the prediction as positive or negative based on a threshold of 0.5.

30. Prints the predicted sentiment.

CHAPTER 8 :TESTING AND RESULTS

8.1 ACCURACY RATE

The accuracy rate achieved by the model using the IMDB dataset is 86.32%



```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_4 (Embedding)     (None, 500, 32)           160000

 flatten_3 (Flatten)         (None, 16000)             0

 dense_6 (Dense)             (None, 250)               4000250

 dense_7 (Dense)             (None, 1)                 251

=================================================================
Total params: 4,160,501
Trainable params: 4,160,501
Non-trainable params: 0
_________________________________________________________________
Epoch 1/2
196/196 - 15s - loss: 0.4996 - accuracy: 0.7178 - val_loss: 0.3191 - val_accuracy: 0.8617 - 15s/epoch - 75ms/step
Epoch 2/2
196/196 - 14s - loss: 0.1818 - accuracy: 0.9316 - val_loss: 0.3295 - val_accuracy: 0.8632 - 14s/epoch - 70ms/step
Accuracy: 86.32%
```

Figure 8.1.1 Accuracy Rate

8.2 CONFUSION MATRIX

The confusion matrix is a table that shows the counts of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) predictions to visualise the performance of a classification model. It aids in determining the model's accuracy and identifying any trends of misclassification.

In the given code, the confusion matrix is calculated using the `confusion_matrix` function from scikit-learn library. The true labels (`y_test`) and predicted labels (`y_pred`) are used as inputs to calculate the confusion matrix.

The confusion matrix is then plotted as a heatmap using the seaborn library's 'seaborn.heatmap' function. The heatmap visualizes the confusion matrix, with each cell representing the number or proportion of samples that fall into a certain category. The x-axis represents the predicted labels, and the y-axis represents the true labels.

The heatmap is color-coded, with various colours reflecting various counts or proportions. This makes it simple to discover trends and evaluate the model's performance. The numbers in the heatmap cells reflect the counts of samples belonging to a certain group (TP, TN, FP, FN).

By visualizing the confusion matrix, you can quickly identify if the model is correctly predicting positive and negative sentiments. The diagonal cells (top left to bottom right) represent correctly predicted labels, while the off-diagonal cells represent misclassifications.

The confusion matrix helps to analyse the model's performance by providing insights into the following:

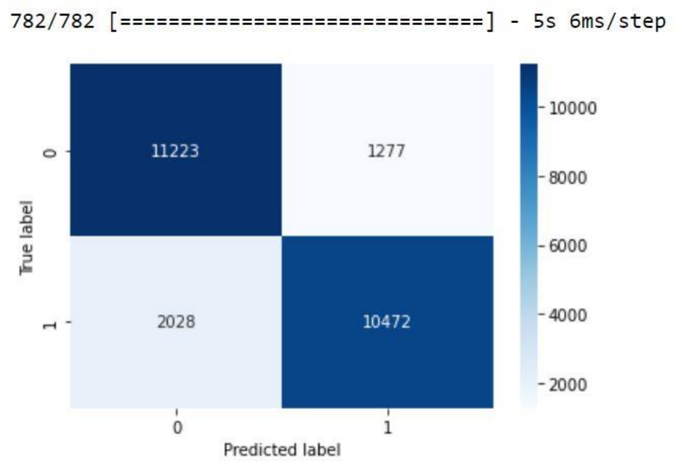
1. True Positives (TP): The model correctly predicted positive sentiments.

2. True Negatives (TN): The model correctly predicted negative sentiments.

3. False Positives (FP): The model incorrectly predicted positive sentiments.

4. False Negatives (FN): The model incorrectly predicted negative sentiments.

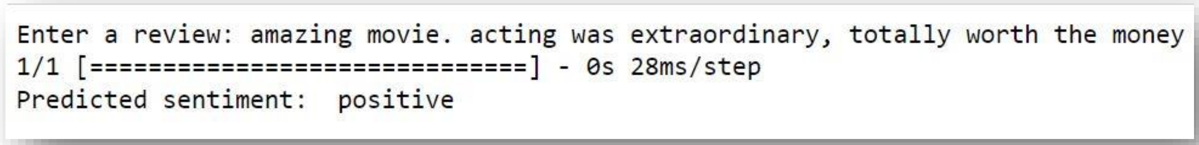
Analysing these values helps in understanding the model's performance in terms of precision, recall, and accuracy.



```
782/782 [==============================] - 5s 6ms/step
```

Figure 8.2.1 Confusion Matrix

8.3 FOR A POSITIVE REVIEW USER INPUT



```
Enter a review: amazing movie. acting was extraordinary, totally worth the money
1/1 [==============================] - 0s 28ms/step
Predicted sentiment:  positive
```

Figure 8.3.1 Result for positive user input

In the given example, we entered the review: "amazing movie. acting was extraordinary, totally worth the money."

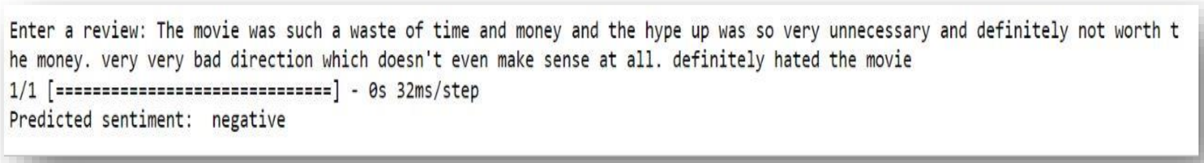
The code then processes this input by converting it to lowercase, splitting it into individual words, and converting those words to a sequence of integers based on the word index from the IMDB dataset. It then tokenizes the words and pads the sequence to match the same length (500 words) used during training.

After pre-processing the user input, the code uses the trained model to make a prediction on this pre-processed input. The prediction is a probability value indicating the likelihood of the review being positive or negative.

Finally, based on the predicted probability value, the code classifies the sentiment as either positive or negative using a threshold of 0.5. In this case, the predicted sentiment is "positive".

This means that the trained model predicted the sentiment of the input review as positive, indicating that the review is expressing a positive opinion or sentiment about the movie.

8.4 FOR A NEGATIVE REVIEW USER INPUT



```
Enter a review: The movie was such a waste of time and money and the hype up was so very unnecessary and definitely not worth t
he money. very very bad direction which doesn't even make sense at all. definitely hated the movie
1/1 [==============================] - 0s 32ms/step
Predicted sentiment:  negative
```

Figure 8.4.1 Result for negative user input

In the given example, we entered the review: "The movie was such a waste of time and money and the hype up was so very unnecessary and definitely not worth the money. very bad direction which doesn't even make sense at all. definitely hated the movie."

The code then processes this input by converting it to lowercase, splitting it into individual words, and converting those words to a sequence of integers based on the word index from the IMDB dataset. It then tokenizes the words and pads the sequence to match the same length (500 words) used during training.

After pre-processing the user input, the code uses the trained model to make a prediction on this pre-processed input. The prediction is a probability value indicating the likelihood of the review being positive or negative.

Finally, based on the predicted probability value, the code classifies the sentiment as either positive or negative using a threshold of 0.5. In this case, the predicted sentiment is "negative".

This means that the trained model predicted the sentiment of the input review as negative, indicating that the review is expressing a negative opinion or sentiment about the movie.

REFERENCES

- https://machinelearningmastery.com/prepare-movie-review-data-sentiment-analysis/
- https://builtin.com/data-science/how-build-neural-network-keras
- https://machinelearningmastery.com/predict-sentiment-movie-reviews-using-deep-learning/
- https://datamahadev.com/lstm-sentimental-analysis-using-keras-with-imdb-dataset/
- https://towardsdatascience.com/sentiment-analysis-using-lstm-step-by-step-50d074f09948
- https://www.geeksforgeeks.org/confusion-matrix-machine-learning/
- https://www.w3schools.com/python/python_ml_confusion_matrix.asp
- https://www.tutorialspoint.com/keras/keras_model_compilation.htm
- https://www.kaggle.com/datasets/atulanandjha/imdb-50k-movie-reviews-test-your-bert/code?datasetId=447516
- https://www.geeksforgeeks.org/what-is-sentiment-analysis/
- https://monkeylearn.com/sentiment-analysis/
- https://towardsdatascience.com/sentiment-analysis-concept-analysis-and-applications-6c94d6f58c17
- https://www.ibm.com/topics/recurrent-neural-networks
- https://www.tutorialspoint.com/keras/keras_model_compilation.htm
- https://www.google.com/search?q=picture+of+overall+working+of+sentiment+analysis&sxsrf=APwXEddvFOsvZeKXJdn-jv4Xwqm76sfvqg:1686075585316&source=lnms&tbm=isch&sa=X&ved=2ahUKEwjD0M6Coa__AhXQQPUHHQuHA84Q_AUoAXoECAEQAw&biw=1280&bih=601&dpr=1.5
- https://www.slideshare.net/J4R/sentiment-analysisan-objective-view
- https://www.altexsoft.com/blog/business/sentiment-analysis-types-tools-and-use-cases/
- https://www.sciencedirect.com/topics/computer-science/sentiment-analysis#:~:text=Sentiment%20analysis%20is%20target%2Doriented,attachment%20to%20a%20specific%20target.
- https://machinelearningmastery.com/prepare-movie-review-data-sentiment-analysis/

- https://builtin.com/data-science/how-build-neural-network-keras
- https://theappsolutions.com/blog/development/sentiment-analysis/
- https://www.google.com/search?q=methodology+of+sentiment+analysis&tbm=isch&ved=2ahUKEwjz1Oa4t6__AhUOLLcAHdrhDUcQ2-cCegQIABAA&oq=methodology+of+sentiment+analysis&gs_lcp=CgNpbWcQAzIFCAAQgAQ6BAgjECc6BwgAEIoFEEM6CwgAEIAEELEDEIMBOggIABCABBCxAzoKCAAQigUQsQMQQzoICAAQsQMQgwE6BggAEAgQHjoHCAAQGBCABFCtDli6QmCDRGgAcAB4AIABiwGIAbgikgEEMC4zNJgBAKABAaoBC2d3cy13aXotaW1nwAEB&sclient=img&ei=RJB_ZPOiII7Y3LUP2sO3uAQ&bih=601&biw=1280
- https://www.researchgate.net/publication/300495226_Sentiment_Analysis
- https://stackoverflow.com/questions/51407278/sentiment-analysis-with-python
- https://www.datacamp.com/tutorial/simplifying-sentiment-analysis-python
- https://www.w3schools.com/python/python_ml_getting_started.asp
- https://machinelearningmastery.com/predict-sentiment-movie-reviews-using-deep-learning/
- https://towardsdatascience.com/sentiment-analysis-with-python-part-1-5ce197074184
- https://datamahadev.com/lstm-sentimental-analysis-using-keras-with-imdb-dataset/
- https://towardsdatascience.com/sentiment-analysis-using-lstm-step-by-step-50d074f09948
- https://www.youtube.com/watch?v=FCvp-hv-62g
- https://www.youtube.com/watch?v=glHWiQRgLn4