SENTIMENTAL ANALYSIS ON MOVIE REVIEWS USING RNN WITH LSTM

A Major Project Report submitted for the partial fulfillment of the requirement for the award of the degree of

MASTER OF COMPUTER APPLICATIONS

SUBMITTED BY

S. Mohammad Sazid

Roll No: 22691F00F5

Under the Guidance of

Dr R. Maruthamuthu M.C.A., Ph.D.,

Assistant Professor

DEPARTMENT OF COMPUTER APPLICATIONS

MADANAPALLE INSTITUTE OF TECHNOLOGY & SCIENCE
MADANAPALLE

UGC AUTONOMOUS

(Approved by AICTE, New Delhi & affiliated to JNTUA, Anantapuramu)
2021-2022

MADANAPALLE INSTITUTE OF TECHNOLOGY & SCIENCEMADANAPALLE

(UGC-AUTONOMOUS)

(Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuram)

DEPARTMENT OF COMPUTER APPLICATIONS

BONAFIDE CERTIFICATE

This is to certify that the project work entitled "SENTIMENTAL ANALYSIS ON MOVIE REVIEWS USING RNN WITH LSTM" is a bona fide work carried out by S. Mohammad Sazid Roll No.:22691F00F5, submitted in the partial fulfillment of the requirements for the award of the degree of Master of Computer Applications in Madanapalle Institute of Technology & Science, Madanapalle, affiliated to Jawaharlal Nehru Technological University Anantapur, Anantapuramu during the academic year 2023-2024.

PROJECT GUIDE

Dr. R. Maruthamuthu M.C.A., Ph.D.,
Assistant Professor

HEAD OF THE DEPARTMENT

Dr. N. Naveen Kumar M.C.A., Ph.D.,
Associate Professor

Submitted for viva voice examination held on ________________

Internal Examiner

Date:

External Examiner

Date:

DECLERATION

I, **S. Mohammad Sazid (Roll No.:22691F00F5)** hereby declare that the project **entitled “**SENTIMENTAL ANALYSIS ON MOVIE REVIEWS USING RNN WITH LSTM” is done by me under the guidance of **Dr. R. Maruthamuthu M.C.A., Ph.D.,** submitted in partial fulfillment of the requirements for the award of degree of Master of Computer Applications at MADANAPALLE INSTITUTE OF TECHNOLOGY & SCIENCE Madanapalle, affiliated to Jawaharlal Nehru Technological University Anantapur, Anantapuramu during the academic year 2023-2024. This work has not been submitted by anybody towards the award of any degree.

Date: Signature of the Student

Place: Madanapalle

ACKNOWLEDGEMENT

First, I must thank the almighty who has granted me knowledge, wisdom, strength and courage to serve in this world and to carry out my project work in a successfully way.

Next, I have to thank my parents who encouraged me to undergo this MCA course and do thisproject in a proper way.

I express my sincere thanks to **Dr. N. Vijaya Bhaskar Chowdary Ph.D.,** Secretary& Correspondent, Madanpalle Institute of Technology & Science for his continuous encouragement to wards practical education and constant support in all aspects which includesthe provision of very good infrastructure facilities in the institute.

It is my duty to thank our Principal, **Dr. C. Yuvaraj M.E, Ph.D.,** Madanapalle Institute of technology & science, for his guidance and support at the time of my course and project.

I also wish to express my thanks to our vice principal, **Dr.P.Ramanathan Ph.D.,** for his continuous support though out our MCA career.

It is my foremost duty to thank Head of the Department **Dr. N. Naveen Kumar M.C.A., Ph.D.,**who gave me constant support during the project time and continuous encouragement to words the completion of the project successfully.

I thank my faculty guide **Dr. R. Maruthamuthu M.C.A., Ph.D.,** for his continuous support by conducting periodical reviews and guidance until the completion of the project in as successful manner.

S. Mohammad Sazid

ABSTRACT

Sentiment analysis is the process of determining the emotional tone conveyed in text and plays a key role in understanding public opinion and consumer sentiment. In this project, I propose a sentiment analysis framework for movie reviews using recurrent neural networks (RNNs) with long short-term memory cells (LSTMs). We use a dataset consisting of labeled movie reviews, with each review tagged with a corresponding sentiment (positive or negative). Our model architecture uses LSTM units. LSTM units are a type of RNN that can capture long-range dependencies in sequential data, making them particularly suitable for analyzing text sequences. Input movie reviews are pre-processed, tokenized, and encoded as a numeric vector using techniques such as word embedding. These embeddings are fed to an LSTM network, learns which to extract meaningful representations of review sentiment from continuous input. During training, optimize model parameters using techniques such as stochastic gradient descent (SGD) and Adam optimization, and apply methods such as early stopping to prevent overfitting. Evaluate model performance using metrics such as precision, precision, recall, and F1 score on a separate test dataset. Experimental results demonstrate the effectiveness of the LSTM-based sentiment analysis framework in accurately classifying the emotions conveyed in movie reviews. The model achieves competitive performance compared to traditional machine learning approaches and demonstrates the ability to generalize to unseen data.

Keywords : Sentiment analysis, LSTM, Word embedding, Stocastic Gradient Descent, Adam Optimization.

Table of Contents

List of Figures

List of Abbreviations

S.NO	Abbrevation	TITLE	P.NO
1	NLP	Natural Language Processing	1
2	RNN	Recurrent Neural Networks	1
3	LSTM	Long Short Term Memory	1
4	CNN	Convolutional Neural Networks	1
65	UML	Unified Modelling Language	17

CHAPTER-1

INTRODUCTION

1.1 About the Project

Sentiment analysis of movie reviews is a fascinating field within natural language processing (NLP) that aims to classify the sentiment expressed in textual data, typically into positive, negative, or neutral categories. Recurrent Neural Networks (RNNs) with Long Short-Term Memory (LSTM) units have emerged as powerful tools for this task due to their ability to capture sequential dependencies in data. In this study, we delve into employing RNNs with LSTM architecture for sentiment analysis of movie reviews. The objective is to develop a model capable of automatically discerning the sentiment conveyed within reviews, aiding in tasks such as movie recommendation systems, market analysis, and understanding audience preferences. The use of LSTM cells in the RNN architecture allows the model to effectively capture long-range dependencies and handle sequential data, making it particularly well-suited for analyzing text. By learning from sequences of words in movie reviews, the LSTM-based model can discern nuanced patterns indicative of sentiment, thereby providing more accurate classification results. Our approach involves preprocessing the raw text data, including steps such as tokenization, padding, and potentially embedding representations to convert words into numerical vectors that can be fed into the LSTM network. The LSTM layers in the network then process these sequences of word embeddings, capturing contextual information and extracting features relevant to sentiment analysis. Furthermore, we employ techniques such as dropout regularization to prevent overfitting and optimize model generalization. The model is trained using labeled movie review datasets, where each review is associated with a sentiment label. Through iterative training and validation, the LSTM-based sentiment analysis model learns to distinguish between positive, negative, and neutral sentiments with increasing accuracy. Finally, we evaluate the performance of the trained model using various metrics such as accuracy, precision, recall, and F1-score on unseen test data. Additionally, we may explore techniques for interpreting model decisions, such as attention mechanisms, to understand which parts of the input text contribute most to the sentiment classification.

1.2 LITERATURE S U R V E Y:

Writing overview is the foremost critical step in computer program advancement prepare. Some time recently creating the device it is fundamental to decide the time figure, economy and company quality. Once these things are fulfilled at that point following steps are to decide which working framework and dialect can be utilized for creating the apparatus. Once the software engineers begin building the instrument the software engineers ned parcel of outside bolster. This bolster cab be gotten from senior software engineers from books and from websites. Some time recently building the framework the above consideration are taken into consideration for creating the proposed framework.

1. “Learning Word Vectors for Sentiment Analysis" by Andrew L. Maas et al. (2011):

This think about emphasizes the significance of conveyed word representations, known as word embeddings, for opinion examination. It presents the thought of learning word vectors from expansive content corpora, which capture semantic implications and connections between words. By leveraging these word embeddings, the creators illustrate critical enhancements in assumption classification precision over conventional bag-of-words approaches. Furthermore, their work highlights the potential of unsupervised learning strategies to improve the quality of word embeddings, eventually driving to more nuanced and successful estimation examination models.

2. “Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank" by Richard Socher et al. (2013):

This seminal work presents the Recursive Neural Tensor Organize (RNTN) for assumption investigation. It centers on understanding opinion compositionality by speaking to sentences as parse trees and illustrates the adequacy of profound learning strategies in capturing nuanced estimation. By leveraging syntactic structures, RNTN can adeptly capture the relevant extremity of words in several expressions. Besides, the consider underscores the prevalent execution of RNTN in taking care of complex etymological marvels such as invalidations and differentiating opinions inside sentences. This progression highlights the significant capability of profound learning models in semantic translation and opinion acumen.

3. "Attention-Based LSTM for Aspect-Level Sentiment Classification" by Xin Li et al. (2017):

This paper presents consideration instruments inside LSTM systems for aspect-level assumption classification. In spite of the fact that not particularly centered on motion picture audits, it underscores the significant part of consideration instruments in capturing fine-grained opinion data from content information. By powerfully weighting the significance of diverse words and expressions in a sentence, consideration instruments empower LSTM models to observe nuanced assumption qualities related with particular perspectives. This approach permits for a more exact and context-aware opinion examination, tending to the restrictions of conventional strategies in taking care of polysemous words and setting shifts. In addition, the ponder illustrates the potential of joining consideration components with profound learning structures to improve interpretability and execution in complex NLP assignments.

4. "Sentiment Analysis of Movie Reviews using Recurrent Neural Networks with TensorFlow" by Ahmad Varasteh (2018):

This paper gives a comprehensive and down to earth usage of opinion investigation utilizing Repetitive Neural Systems (RNNs), particularly Long Short-Term Memory (LSTM) systems, on motion picture audit datasets. It fastidiously subtle elements preprocessing steps, counting tokenization, stemming, and taking care of of out-of-vocabulary words, to plan crude content information for demonstrate utilization. The piece of the demonstrate engineering digs into the complexities of inserting layers, LSTM cells, and completely associated layers, explaining the method of reasoning behind each plan choice. Preparing procedures such as angle clipping, learning rate planning, and dropout regularization are investigated to optimize the model's execution and moderate overfitting. Moreover, the paper examines assessment measurements such as precision, accuracy, review, F1-score, and AUC-ROC, giving a strong system for surveying show viability. This consider not as it were serves as a profitable direct for executing comparative ventures but too contributes to the broader talk on progressed strategies in estimation investigation.

5."A Comparative Study on Sentiment Analysis of Movie Reviews Using Deep Learning Techniques" by S. Sumathi and R. Gomathi (2019):

This ponder compares the execution of different profound learning models, counting LSTM-based designs, for assumption examination of motion picture surveys. It gives bits of knowledge into the adequacy of LSTM compared to other approaches and examines the affect of diverse hyperparameters.

The investigate fastidiously assesses the models employing a run of measurements, highlighting the qualities and restrictions of each approach in capturing opinion subtleties. Furthermore, it investigates the part of pre-trained word embeddings, such as Word2Vec and GloVe, in upgrading demonstrate execution. The think about moreover looks at the impact of hyperparameter tuning, counting learning rates, clump sizes, and the number of LSTM layers, advertising a nitty gritty investigation of how these components contribute to te generally precision and strength of opinion classification.

6. “Deep Sentiment Analysis with LSTM Networks: An Extended Study of Text-Based Emotion Detection in Movie Reviews" by Damjan Bojadzievski and Bojan Furlan (2016):

A few farther client confirmation procedures for telecare medication data framework TMIS have been proposed within the writing. But most existing strategies have impediments such as powerless to different assaults, need of functionalities, and wastefulness. As of late, Amin and Biswas proposed a three- calculate confirmation and key assention method for TMIS. But their conspire is wasteful and has a few security downsides. The assaults such as privileged-insider, client pantomime, and solid answer assaults are conceivable on their plot. It too has blemish in secret word upgrade stage

7. "A Survey on Sentiment Analysis and Opinion Mining Techniques" by N. Jindal and B. Liu (2008):

In spite of the fact that not particular to LSTM-based approaches, this overview gives a comprehensive outline of assumption examination strategies up to its distribution date. It covers different techniques, counting machine learning and lexicon-based approaches, giving profitable setting for understanding the advancement of assumption examination inquire about. The study moreover digs into cross breed models, which combine numerous strategies to improve execution, and talks about the challenges and future headings within the field. This comprehensive asset serves as a foundational reference for analysts and professionals looking to get a handle on the breadth of estimation investigation progressions. Also, it highlights the part of highlight building in moving forward demonstrate execution and the integration of opinion examination with other NLPassignments.

8."Attention-Based LSTM for Aspect-Level Sentiment Classification" by Xin Li et al. (2017):

Whereas this paper centers on aspect-level estimation classification instead of motion picture audits particularly, it presents consideration components inside LSTM systems. Consideration instruments can improve the interpretability of LSTM-based opinion investigation models by highlighting important words or expressions within the input content. This approach permits the demonstrate to concentrate on the foremost important parts of the input when making a estimation expectation, hence moving forward exactness. Besides, the utilize of consideration instruments encourages superior understanding and visualization of how the show forms and weighs distinctive perspectives of the content. This makes the models not as it were more precise but moreover more straightforward and reasonable to end-users and analysts alike. Moreover, the paper investigates different sorts of consideration components, such as self-attention and progressive consideration, illustrating their particular benefits. It too examines the computational effectiveness of consideration instruments compared to conventional LSTM models. The integration of these instruments has cleared the way for more modern and fine-grained opinion examination applications.

9."Hierarchical Attention Networks for Document Classification" by Zichao Yang et al. (2016):

This paper presents Various leveled Consideration Systems (HAN) which can be exceptionally important for estimation investigation. The various leveled structure and consideration component in HAN permit the show to center on the foremost enlightening words and sentences in a archive, making it a valuable approach for analyzing motion picture audits.

10. "Deep Learning for Sentiment Analysis: A Survey" by Youngjin Kim, Kyunghyun Cho, Yoshua Bengio (2017):

This study gives a comprehensive outline of different profound learning procedures connected to opinion examination, counting RNNs and LSTMs. It talks about diverse designs, techniques, and applications, making it a profitable asset for understanding the broader setting and progressions with in the field.

CHAPTER-2

SYSTEM ANALYSIS

2.1 EXISTING SYSTEM:

The existing systems for sentiment analysis of movie reviews typically involve various machine learning and deep learning techniques.

1. Machine Learning-Based Systems:

Traditional Feature-Based Models: These systems rely on handcrafted features such as word frequencies, n-grams, or syntactic patterns to train classifiers like Support Vector Machines (SVM) or Naive Bayes.

2. Deep Learning-Based Systems:

FNNs comprise of numerous layers of neurons, where each neuron is associated to each neuron within the consequent layer. They are prepared utilizing backpropagation with angle plummet to classify motion picture audits into opinion categories.

CNN-based models regularly utilize convolutional layers to extricate highlights from printed inputs, taken after by pooling layers and completely associated layers for classification.

2.2 DISADVANTAGES OF EXISTING SYSTEM:

Limitations of Machine Learning Models:

1. Limited Contextual Understanding
2. Difficulty with Sequential Data
3. Manual Feature Engineering

Limitations of Feedforward Neural Networks (FNNs):

1. Fixed Input Size
2. Lack of Temporal Understanding
3. Difficulty with Textual Data

Limitations of Convolutional Neural Networks (CNNs):

1. Limited Sequential Modeling
2. Difficulty with Variable-Length Inputs
3. Limited Contextual Understanding

2.3 PROPOSED SYSTEM:

The proposed framework for estimation examination of motion picture audits utilizing Repetitive Neural Systems (RNNs) with Long Short-Term Memory (LSTM) engineering points to create an effective and exact demonstrate for classifying the assumption communicated in motion picture audits. The framework will start with preprocessing the literary information, counting tokenization and word inserting representation. The LSTM-based RNN will be outlined to capture successive conditions and relevant data characteristic in motion picture surveys. To optimize demonstrate execution and anticipate overfitting, procedures such as dropout regularization will be utilized. The demonstrate will be prepared utilizing labeled motion picture audit datasets and assessed utilizing measurements like exactness, accuracy, review, and F1-score. Also, we are going investigate strategies for deciphering demonstrate choices, such as consideration components, to get it the commitment of diverse words to the opinion classification. By leveraging profound learning techniques, our proposed framework points to supply important experiences into group of onlookers opinions towards motion pictures, encouraging applications like proposal frameworks and advertise examination.

2.4 ADVANTAGES OF PROPOSED SYSTEM:

Effective Sequential Data Handling: RNNs with LSTM effectively process sequential data like movie reviews, capturing temporal dependencies.

Long-Term Dependency Learning: LSTM units learn and retain information over longer sequences, understanding overall sentiment.

Dynamic Input Length Handling: RNNs with LSTM accommodate variable-length inputs without preprocessing.

Contextual Understanding: LSTM's memory state enables better context comprehension by retaining information from earlier parts of the text.

Interpretability with Attention: Enhanced with attention mechanisms, the model highlights key phrases, improving interpretability.

2.5 HARDWARE REQUIREMENTS

• **Processor**	-	Intel I5/Ryzen 5
• **RAM**	-	8GB (Minimum requried)
• **Hard Disk**	-	512GB

SOFTWARE REQUIREMENTS

• **Programming language**	**:**	Python
• **Libraries used**	**:**	Numpy, Matplolib, Pandas, NLTK, Tensor flow, Keras.
• **IDE**	**:**	Jupyter Notebook
• **Operating System**	**:**	Windows 11

2.6 FEASIBILITY STUDY

The possibility of the project is examined in this stage and commerce proposition is put forward with an awfully common arrange for the venture and a few fetched gauges. Amid framework investigation the possibility think about of the proposed framework is to be carried out. Typically to guarantee that the proposed framework isn't a burden to the company. For possibility investigation, a few understanding of the major necessities for the framework is fundamental.
Three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- OPERATIONAL FEASIBILITY

ECONOMICAL FEASIBILITY:

This consider is carried out to check the money related influence that the system will have on the organization. The whole of stores that the company can allocate to the ask around and advancement of the system is limited. The employments must be guarded. In this way, the made system remained interior the budget, and this was finished since most of the propels utilized are wholeheartedly available. Because it were the customized things had to be gotten. Gear costs, such as GPU utilization (within the occasion that not utilizing free organizations like Google Colab), may cause additional costs. Cloud organizations, within the occasion that passing on the illustrate inside the cloud, as well ought to consider costs related with encouraging and serving the appear. Moreover, preparing expansive models on-premise may require critical frank hypothesis in hardware and back. Operational costs, such as control and cooling, have to be besides be figured in. Executing cost-effective measures like optimizing code and utilizing viable calculations can offer help in reducing resource utilization. Leveraging open-source libraries and frameworks can inside and out lower program costs. Conducting a cost-benefit examination will offer help in understanding the long-term monetary reasonability of the expand.

TECHNICAL FEASIBILITY:

This consider is carried out to check the specialized achievability, that's , the specialized necessities of the framework. Any framework created must not have a tall request on the accessible specialized assets. This will lead to tall requests being put on the client. The created framework must have unassuming prerequisites, as as it were negligible or no changes are required for actualizing this framework. Profound Learning Models:

Executing RNN (Recurrent Neural Networks) with LSTM (Long Short-Term Memory) is actually doable. These models handle consecutive information and are well-suited for common dialect handling assignments. Python Libraries:

Utilize Python libraries such as TensorFlow, Keras, and NLTK (Characteristic Dialect Toolkit) for demonstrate advancement. GPU Back:

Preparing profound learning models can be computationally seriously. Utilizing GPU bolster (e.g., Google Colab) guarantees quicker preparing times. Furthermore, it is pivotal to guarantee that the system's integration into the existing foundation is smooth and does not disturb progressing operations.

OPERATIONAL FEASIBILITY:

The perspective of consider is to check the level of acknowledgment of the framework by the client. This incorporates the method of preparing the client to utilize the framework effectively. The client must not feel debilitated by the framework, instep must acknowledge it as a need. The level of acknowledgment by the clients exclusively depends on the strategies that are utilized to teach the client about the framework and to create him commonplace with it. His level of certainty must be raised so that he is additionally able to form a few valuable feedback, which is invited, as he is the ultimate client of the framework.

Information Collection:

Accumulate a labeled dataset of motion picture surveys (e.g., IMDB surveys). This dataset ought to cover a assorted run of estimations.

Information Preprocessing:

Clean and preprocess the content information by tokenizing, evacuating halt words, and changing over words to vectors (utilizing procedures like Word2Vec or GloVe).

Show Preparing:

Prepare the RNN-LSTM demonstrate on the dataset. Test with hyperparameters (e.g., learning rate, bunch estimate) for ideal execution.

Induction:

Convey the prepared show to anticipate opinion names for unused motion picture surveys.

CHAPTER-3

SYSTEM DESIGN

3.1 INPUT DESIGN

The input plan could be a basic component that guarantees the information encouraged into the show is well-structured and significant. The input plan prepare starts with information collection, where a expansive dataset of motion picture audits, such as the IMDB surveys dataset, is accumulated. This dataset ordinarily comprises of person audits labeled with their comparing sentiments (positive, negative, or impartial). Each review may be a content archive that must be preprocessed some time recently being utilized as input for the LSTM show. Information preprocessing includes a few steps to change over crude content into a reasonable arrange for the neural organize. To begin with, tokenization is applied, which breaks down the audits into person words or tokens. This step is vital for taking care of content information because it allows the demonstrate to prepare and understand the input. Taking after tokenization, halt word evacuation is performed to dispense with common but uninformative words just "like," the "and," and "is," which don't contribute altogether to the opinion of the audit.

Following, the remaining words are changed over into numerical representations utilizing word implanting procedures such as Word2Vec, GloVe, or embeddings given by TensorFlow or Keras. These embeddings change words into thick vectors that capture their semantic meaning and connections. This step is fundamental because it permits the LSTM show to prepare literary information in a numerical arrange that it can get it and learn from.

To guarantee consistency within the input information, cushioning or truncation is connected to the tokenized surveys. Since motion picture surveys change in length, cushioning arrangements with zeros or truncating them to a fixed length ensures that all input sequences are of the same length, which is vital for bunch handling in neural systems. This settled length is regularly chosen based on the dissemination of audit lengths within the dataset, guaranteeing that most audits are not one or the other as well brief nor too much long.

Also, the input plan may include creating bunches of information to move forward the effectiveness of show preparing. Bunching permits the show to handle numerous surveys at the same time, speeding up the training process and making way better utilize of computational resources. The estimate of these clusters is decided based on the accessible memory and computational control.

By carefully planning the input pipeline through these preprocessing steps, the venture guarantees that the LSTM show gets clean, well-structured, and important information. This careful arrangement of input information is pivotal for preparing an successful estimation investigation show that can precisely translate and classify the sentiment of movie reviews.

3.2 OBJECTIVES

The input design targets center on making a organized and productive information pipeline that upgrades the model's capacity to memorize and generalize from the given information. The essential objective is to preprocess and change crude motion picture audit writings into a numerical organize reasonable for an LSTM show whereas keeping up the semantic astuteness of the surveys. This includes standardizing the length of input arrangements through cushioning or truncation, permitting the show to prepare bunches of information productively. Steady input lengths avoid issues related to variable grouping sizes and guarantee that each clump contains surveys of a uniform estimate, which is basic for the compelling preparing of RNNs and LSTMs. The objective is to convert the content information into a arrange that jam the relevant meaning of words and expressions, empowering the LSTM show to understand and learn from the basic sentiment patterns within the audits. By centering on the foremost significant parts of the content, the input plan guarantees that the show isn't diverted by uninformative words or images, subsequently making strides its capacity to precisely classify assumptions. The preprocessing pipeline ought to be outlined to handle expansive datasets productively, empowering fast changes and negligible computational overhead.
This implies guaranteeing that the prepared information is within the redress arrange and structure required by the show, allowing seamless information stream from preprocessing to demonstrate preparing and assessment. By accomplishing these input plan targets, the venture guarantees that the LSTM demonstrate is given with high-quality, semantically wealthy, and consistently organized information, which is basic for creating an successful and precise estimation investigation framework for motion picture audits.

3.3 OUTPUT DESIGN:

The yield plan is centered around giving clear, enlightening, and significant comes about from the model's opinion expectations. The essential objective is to guarantee that the yield is effortlessly interpretable by clients, advertising both high-level rundowns and point by point experiences into the assumption investigation.

The primary component of the yield plan is the assumption name, which categorizes each motion picture audit as positive, negative, or impartial. This name is went with by a assumption score, a numerical esteem ordinarily extending from to 1, demonstrating the model's certainty in its forecast. For occurrence, a score of 0.85 for a positive assumption recommends a tall certainty level, while a score closer to 0.5 may demonstrate equivocalness.

OBJECTIVES OF OUTPUT DESIGN:

The objectives of output design are:

- Ensure the model accurately classifies movie reviews into sentiment categories (e.g., positive, negative, neutral).
- Achieve high accuracy, precision, recall, and F1-score metrics
- Highlight improvements or areas where the LSTM model excels.
- Ensure low latency in processing and response time to enhance user experience.
- Design the output interface to be intuitive and easy to interpret for end-users.

3.4 MODULES DESCRIPTION:

six modules:

- ➢ Data Collection and Preprocessing
- ➢ Word Embedding
- ➢ Model Architecture and Design Building
- ➢ Model Training
- ➢ Model Evaluation
- ➢ Model Deployment

1. Data Collection and Preprocessing

Collect a comprehensive dataset of motion picture audits from sources like IMDb, Spoiled Tomatoes, or Kaggle. Guarantee the dataset incorporates different estimations and adequate illustrations for each course. Preprocess the content information by expelling HTML labels, accentuation, and extraordinary characters. Utilize libraries like BeautifulSoup for HTML parsing and regex for design coordinating to clean the text. Perform tokenization to part the content into person words or subwords. Utilize tokenization libraries such as NLTK or SpaCy to handle distinctive dialects and edge cases viably. Change over content to lowercase and evacuate halt words to standardize the input. This step makes a difference in diminishing the dimensionality of the information and progressing the execution of the show.

2. Word Embedding

Select a word inserting strategy such as Word2Vec, GloVe, or FastText based on the extend necessities and dataset characteristics. Consider pre-trained embeddings for superior generalization.
Execute the chosen implanting strategy to create word vectors for the motion picture survey lexicon. Utilize libraries like Gensim for Word2Vec or download pre-trained embeddings for GloVe and FastText.
Construct an implanting lattice utilizing the word vectors and outline each word within the lexicon to its comparing implanting vector. Initialize obscure words with irregular vectors or utilize a extraordinary token.
Guarantee the implanting network is consistent with the profound learning system being utilized (e.g., TensorFlow, PyTorch). This network will be utilized to change over input content arrangements into numerical organize for the demonstrate.

3. Model Architecture and Design

Define the design of the LSTM-based sentiment investigation show, counting the number of LSTM layers, covered up units, and dropout regularization rates. Consider including Bidirectional LSTMs for progressed setting understanding.
Plan the input layer to acknowledge groupings of word embeddings created from the inserting network. Utilize an Implanting layer in TensorFlow or PyTorch to handle the implanting lookup.
Arrange the yield layer for parallel classification (positive or negative assumption) or multi-class classification (positive, negative, impartial estimation) based on the venture necessities.
Compile the show with an suitable misfortune work and optimizer. For parallel classification, utilize twofold cross-entropy, and for multi-class classification, utilize categorical cross-entropy. Select optimizers like Adam or RMSprop.

4. Model Training

Split the dataset into training, validation, and test sets. Guarantee the part is stratified to preserve the conveyance of classes in each set.

Prepare the LSTM demonstrate utilizing the preparing information and screen execution on the approval set to avoid overfitting. Actualize early halting and model checkpointing to spare the most excellent demonstrate.

Explore with diverse hyperparameters (e.g., learning rate, bunch estimate, number of ages) to optimize model execution. Utilize methods like framework look or arbitrary rummage around for precise tuning.

On the off chance that the dataset is imbalanced, consider utilizing strategies like oversampling, undersampling, or lesson weights to address the awkwardness amid preparing.

5. Model Evaluation

Evaluate the prepared demonstrate on the test set utilizing measurements such as exactness, exactness, review, and F1-score to survey its execution. Visualize the comes about utilizing perplexity frameworks and ROC bends.

Conduct mistake investigation to recognize common misclassifications and regions for change. Look at misclassified illustrations to get it show confinements and information subtleties.

Compare the comes about with standard models or existing estimation examination strategies to gage the viability of the LSTM-based approach. Utilize less difficult models like calculated relapse or SVM for baselines.

Based on the assessment and blunder investigation, iteratively refine the demonstrate and preprocessing steps. Consider extra information enlargement or progressed show models in case required.

6. Model Deployment

Send the prepared model as a estimation examination apparatus for motion picture audits. Serialize the prepared demonstrate to a record arrange consistent with sending situations (e.g., HDF5, TensorFlow SavedModel).

Select a sending environment such as cloud stages (e.g., AWS, GCP) or edge gadgets. Guarantee the environment underpins the desired libraries and systems.

Approve the conveyed demonstrate in real-world scenarios and assemble client criticism for encourage advancements. Screen execution and client intuitive to distinguish any issues.
Execute persistent checking and upkeep to guarantee the demonstrate remains compelling over time. Arrange for intermittent retraining with modern information to keep the demonstrate upgraded.

3.5 UML DIAGRAMS

UML(Unified Model Language) speaks to Bound together Modeling Dialect. UML is an institutionalized all around useful appearing lingo within the circumstance of article found programming planning. The a la mode is managed, and develop to be made by way of, the Protest Administration Gather.

The reason is for UML to turn out to be an regular lingo for making designs of thing organized PC programming. In its show body UML is contained two vital components:

a Meta-show and documentation. Later on, a number of sorts of approach or contraption can also moreover be introduced to; or related with, UML.

The Bound together Modeling Dialect could be a well known tongue for showing, Visualization, Constructing and archiving the curios of programming system, and for trade illustrating and uncommon non-programming systems. The UML may be a basic piece of creating gadgets found programming and the item advancement strategy. The UML makes utilize of commonly graphical documentations to specific the arrange of programming wanders.

3.5.1 Activity Diagram

Activity charts are graphical representation of work stream of step astute exercises and activities with bolster for choice, emphasis and concurrency, within the bound together demonstrating dialect, movement charts can be utilized to portray the commerce and operational step-by-step work streams of components in a framework.

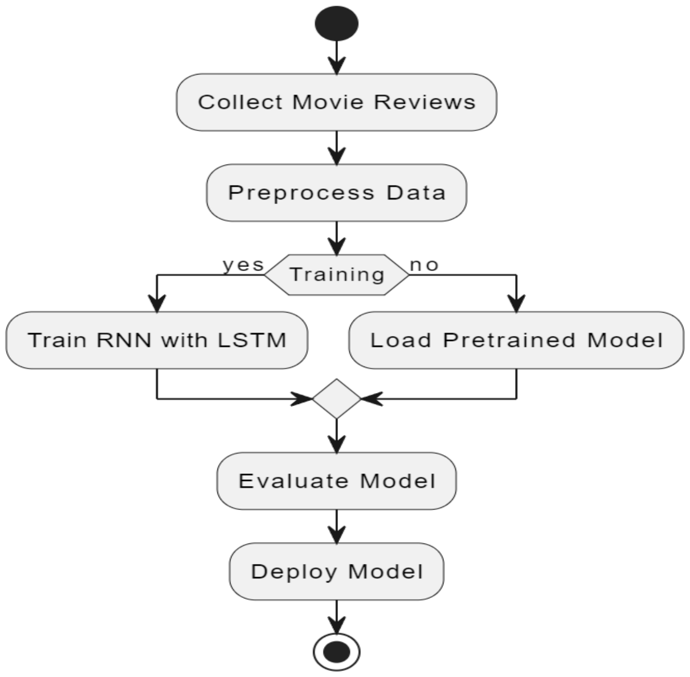


Fig 3.5.1 Activity Diagram

3.5.2 USE CASE DIAGRAM:

A use case chart within the Unified Modeling Language (UML) could be a sort of behavioural graph characterized by and made from a Use-case investigation. Its reason is to display a graphical outline of the usefulness given by a framework in terms of performing artists, their objectives (represented as utilize cases), and any conditions between those utilize cases. The most reason of a utilize case graph is to appear what framework capacities are performed for which performing artist. Parts of the on-screen characters within the framework can be portrayed.

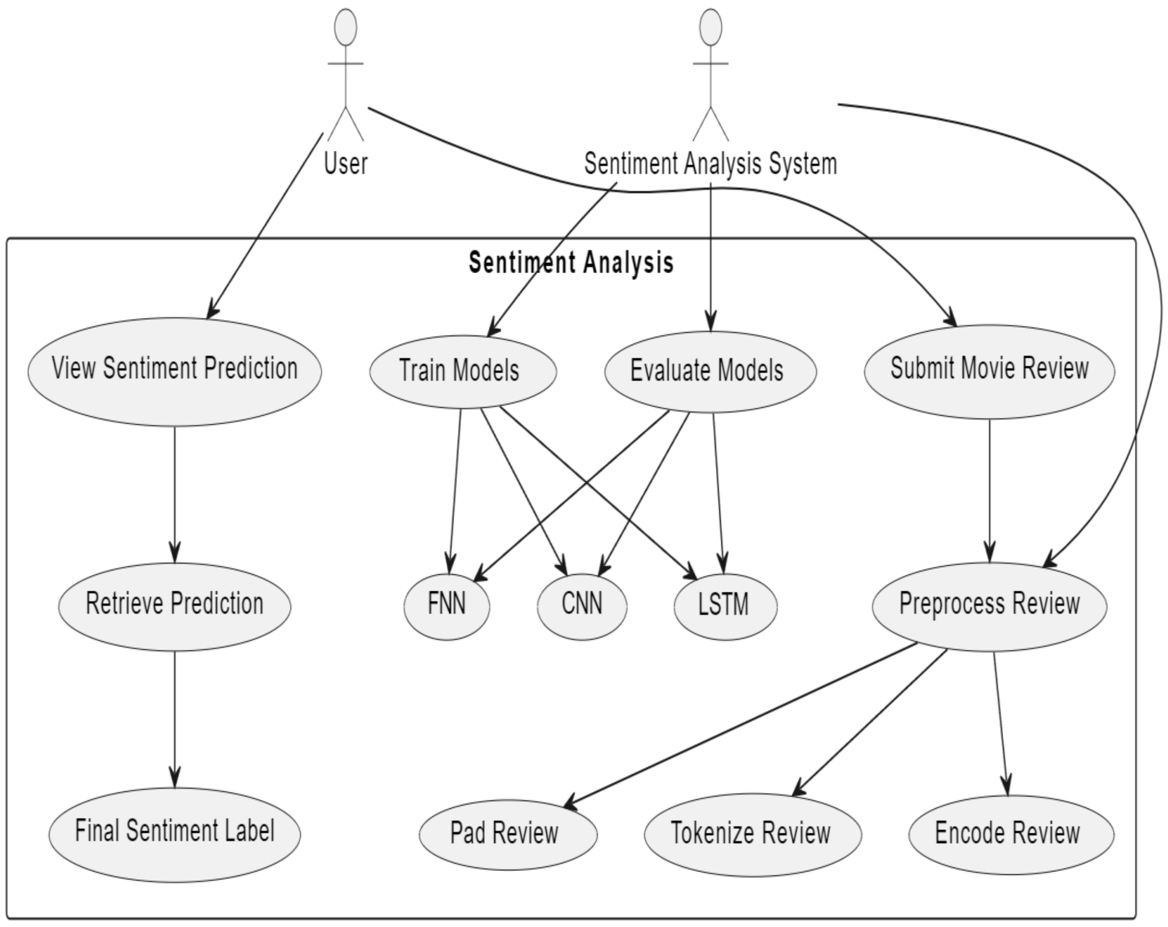


Fig 3.5.2 Use Case Diagram

3.5.3 SEQUENCE DIAGRAM:

A sequence diagram in Bound together modeling Dialect (UML) may be a kind of interaction graph that appears how forms work with one another and in what arrange. It may be a build of a Message Sequence Chart. Arrangement charts are now and then called occasion graphs, occasion scenarios, and timing graphs.

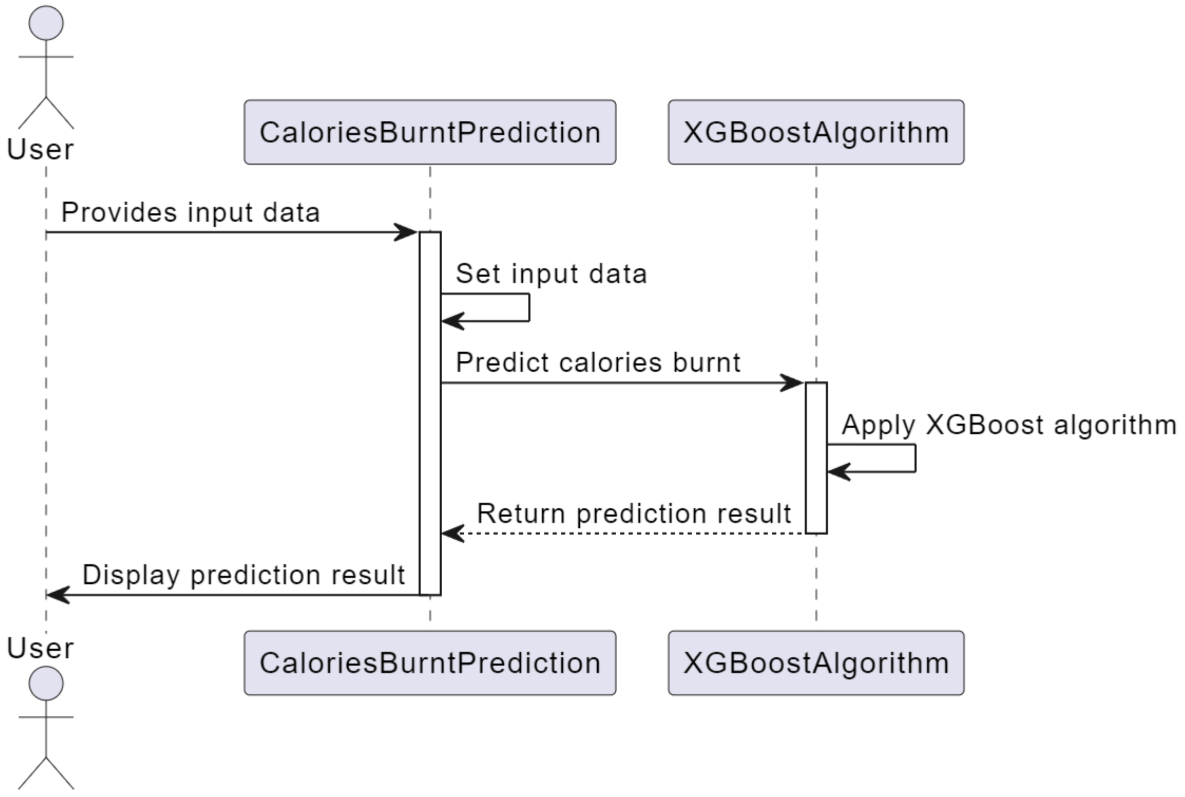


Fig 3.5.3 Sequence Diagram

3.5.4 Deployment Diagram

Deployment diagram speaks to the sending see of a framework. It is related to the component chart.

Since the components are sent utilizing the arrangement graphs. A arrangement graph comprises of hubs. Hubs are nothing but physical hardware's utilized to send the applications.

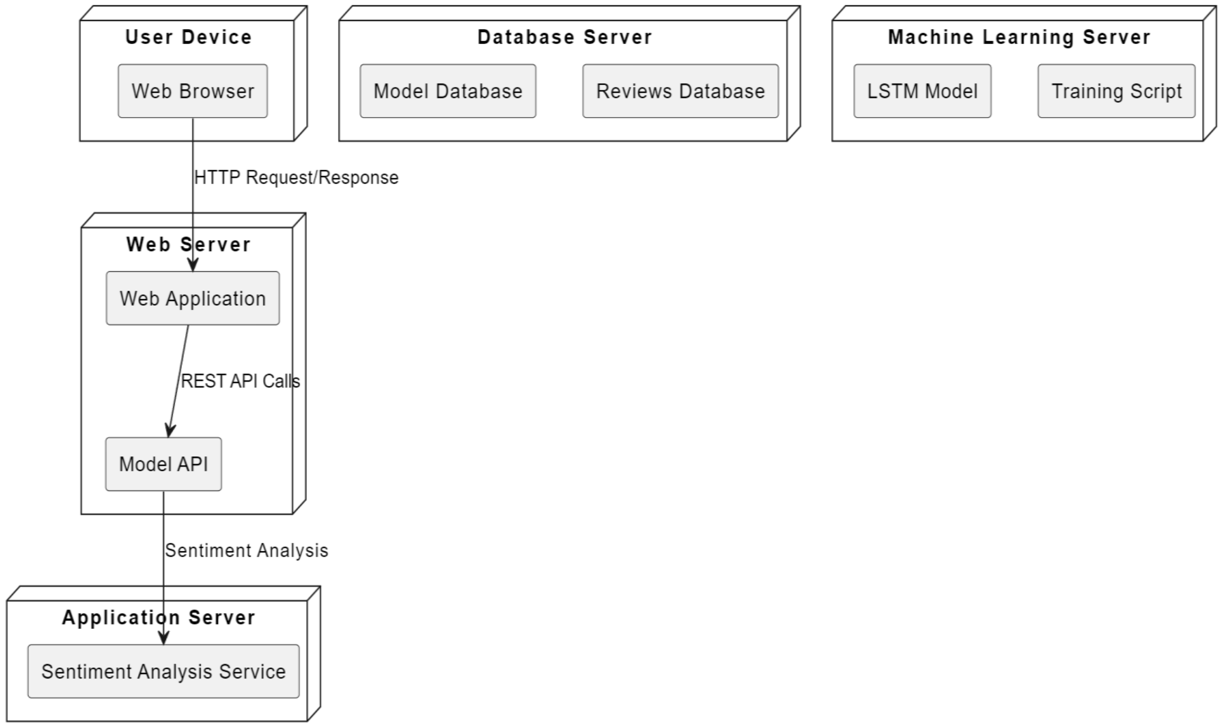


Fig 3.5.4 Deployment Diagram

3.5.5 CLASS DIAGRAM

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information. There are four approaches for identifying classes.

1. Noun phrase approach.
2. Common class pattern approach.
3. Use case driven sequence or collaboration approach.
4. Classes, Responsibilities and Collaboration approach.

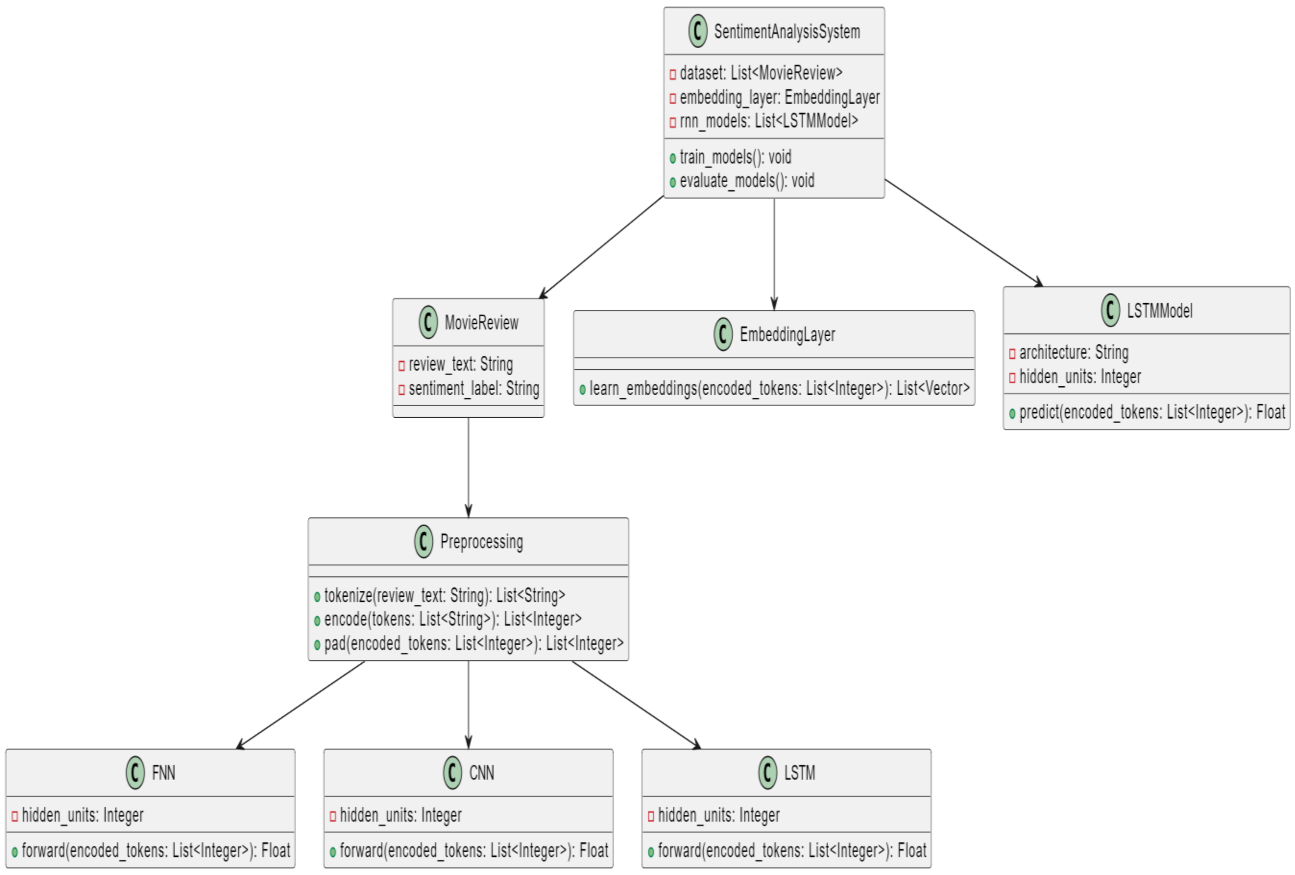

SentimentAnalysisSystem
dataset: List<MovieReview>
embedding_layer: EmbeddingLayer
rnn_models: List<LSTMModel>
train_models(): void
evaluate_models(): void
MovieReview
review_text: String
sentiment_label: String
EmbeddingLayer
learn_embeddings(encoded_tokens: List<Integer>): List<Vector>
LSTMModel
architecture: String
hidden_units: Integer
predict(encoded_tokens: List<Integer>): Float
Preprocessing
tokenize(review_text: String): List<String>
encode(tokens: List<String>): List<Integer>
pad(encoded_tokens: List<Integer>): List<Integer>
FNN
hidden_units: Integer
forward(encoded_tokens: List<Integer>): Float
CNN
hidden_units: Integer
forward(encoded_tokens: List<Integer>): Float
LSTM
hidden_units: Integer
forward(encoded_tokens: List<Integer>): Float

Fig 3.5.5 Class Diagram

3.5.6 COMPONENT DIAGRAM

Component diagrams are utilized to depict the physical artifacts of a framework. This artifact incorporates records, executables, libraries etc. So the reason of this graph is diverse, Component graphs are utilized amid the execution stage of an application. But it is ready well in development to imagine the usage points of interest. At first the system is planned utilizing diverse UML charts and after that when the artifacts are prepared component charts are utilized to urge an thought of the execution.

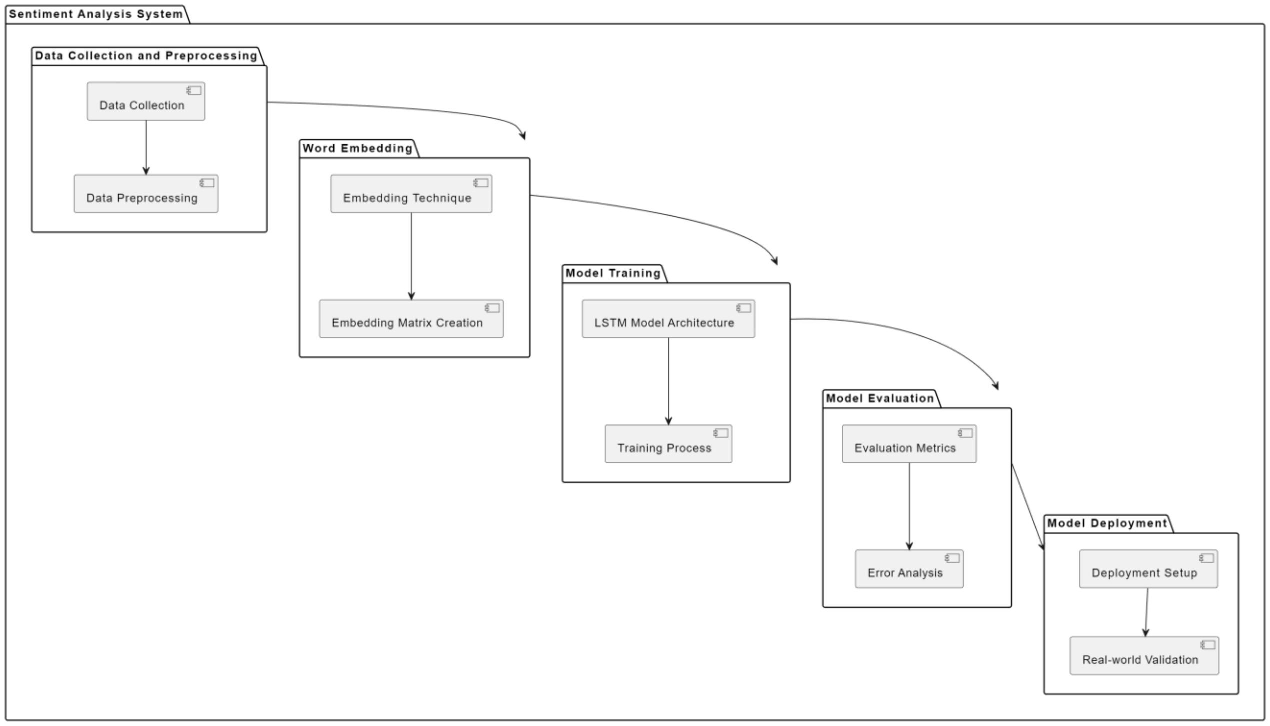


Fig 3.5.6 Component Diagram

CHAPTER-4

SYSTEM IMPLEMENTATION

4.1 ALGORITHMS:

There are a few key calculations are utilized to preprocess the information, prepare the demonstrate, and make forecasts. The primary pivotal step includes information preprocessing, where normal dialect preparing (NLP) procedures such as tokenization, halt word expulsion, and word implanting are connected. Tokenization breaks down the motion picture audits into person words or tokens, whereas halt word evacuation dispenses with common but uninformative words. Word implanting procedures like Word2Vec or GloVe change over these tokens into thick vector representations that capture semantic meaning.

For the center show, the extend utilizes Repetitive Neural Systems (RNNs) with Long Short-Term Memory (LSTM) units. RNNs are well-suited for consecutive information as they have the capability to preserve a hidden state that captures data from past steps within the grouping. Be that as it may, conventional RNNs battle with long-term conditions due to vanishing slope issues. LSTMs address this issue by consolidating memory cells and gating components (input, disregard, and yield doors) that control the stream of data, permitting the arrange to hold and utilize long-term conditions more successfully.

The show preparing prepare includes forward and in reverse engendering. Amid forward engendering, the input information (preprocessed motion picture surveys) is passed through the LSTM layers, producing covered up states and yield forecasts at each time step. The misfortune work, ordinarily parallel cross-entropy for twofold assumption classification, is computed by comparing the predicted sentiments with the genuine names. In reverse proliferation through time (BPTT) is at that point utilized to upgrade the show weights. This includes calculating angles of the misfortune work with regard to the demonstrate parameters and applying an optimization calculation like Adam to play down the misfortune.

Furthermore, the extend may consolidate consideration components to upgrade the model's execution and interpretability. Consideration instruments permit the model to center on the foremost pertinent parts of the input grouping when making forecasts. By relegating distinctive weights to distinctive tokens, the show can highlight vital words or phrases that essentially contribute to the by and large assumption of the audit.

At last, the prepared show is sent to foresee opinion names for unused, concealed motion picture surveys. This includes passing the preprocessed audits through the LSTM organize and utilizing the learned weights to create opinion scores. These scores are at that point changed over into assumption names (positive, negative, or impartial) based on a predefined edge. The combination of these calculations guarantees a vigorous and precise opinion examination framework that can successfully decipher and classify motion picture audits.

4.2 LANGUAGE SELECTION

History of Python

Python underpins different programming ideal models, counting procedural, object-oriented, and useful programming. Its broad standard library and community-contributed modules make it flexible and capable for a wide run of applications, from web advancement to information science. The language's plan logic emphasizes code meaningfulness, utilizing critical space to delimit code pieces. Python's ease of learning and utilize has made it a prevalent choice for apprentices and experienced engineers alike. Nonstop overhauls and a solid community guarantee Python remains at the bleeding edge of programming advancement.

Overview of Python

Python brags a endless standard library that gives instruments suited to numerous assignments, from web improvement to logical computing. Furthermore, Python's broad environment incorporates a riches of third-party bundles accessible through the Python Bundle File (PyPI). This vigorous bolster organize encourages quick advancement and arrangement of complex applications. Python's energetic writing and memory administration capabilities assist disentangle coding, decreasing the require for boilerplate code. Customary upgrades and a solid, dynamic community offer assistance guarantee that Python proceeds to advance and meet desires of present day engineers.

Features of Python

Python is an object-oriented dialect that bolsters epitome, legacy, polymorphism, and energetic official. It permits for different legacy, not at all like Java. Python's basic language structure emphasizes coherence and diminishes the taken a toll of program support. It has broad bolster for modules and bundles, empowering measured programs and code reuse. Python underpins multithreading, permitting distinctive parts of a program to run at the same time, which can make strides execution in certain applications.

Python and Internet

Python's integration capabilities too make it an perfect choice for back-end improvement, interfacing consistently with databases like MySQL, PostgreSQL, and SQLite. Its capacity to handle HTTP demands, oversee sessions, and interface with APIs disentangles the creation of strong web administrations. Besides, Python's bolster for offbeat programming through systems like asyncio and libraries such as aiohttp upgrades its execution in dealing with concurrent web demands. The language's flexibility amplifies to front-end advancement as well, with instruments like Jinja2 for templating. Python's dynamic community contributes to a wealthy environment of plugins and expansions, giving arrangements for nearly any web advancement challenge. The combination of these highlights guarantees Python remains a beat choice for engineers looking to make adaptable, viable web applications.

Python and World Wide Web

The World Wide Web is an open-ended data recovery framework planned for dispersed situations. Python can be utilized to form energetic and intuitively web pages, joining consistently with web advances such as HTML, CSS, and JavaScript. With Python, designers can construct web applications that are competent of supporting activitys, illustrations, recreations, and other uncommon impacts. Python has made the web more energetic and intelligently, permitting for the improvement of strong web applications. Python's information science libraries (such as Pandas, NumPy, and Matplotlib) are fundamental for analyzing and visualizing web-related information. From following client behavior to making intuitively charts, Python makes it conceivable.

Python Environment

The Python environment incorporates a wide run of apparatuses and libraries that are portion of the Python Standard Library (PSL). These instruments and libraries give functionalities for different errands such as record dealing with, framework calls, and web conventions. The Python environment moreover incorporates the Python Bundle File (PyPI), which has thousands of third-party modules and bundles that expand Python's capabilities.

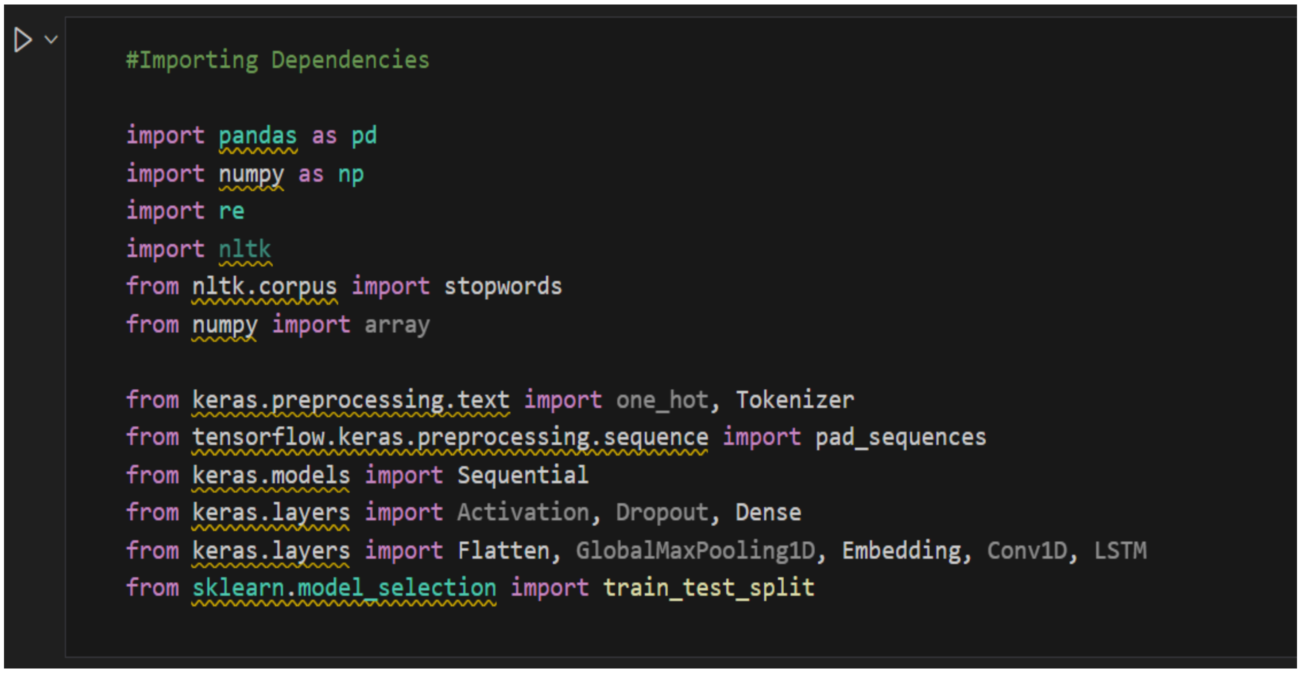
Python Architecture

Python gives a convenient, strong, and high-performance environment for advancement. It accomplishes transportability by compiling the source code into bytecode, which is at that point translated by the Python Virtual Machine (PVM) on each stage. Python guarantees code soundness through thorough compile-time and runtime checking and programmed memory administration, which makes a difference in composing strong and viable code.

Python Virtual Machine

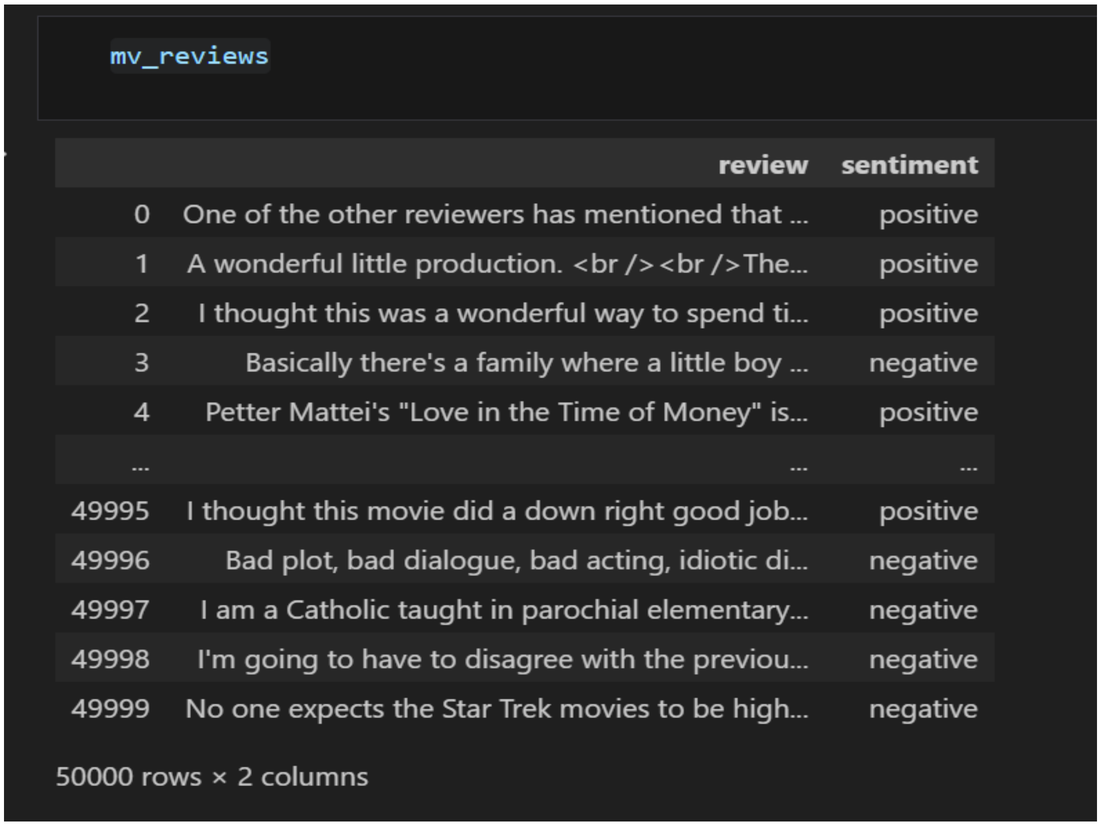
When Python code is executed, the Python translator compiles the source code into machine code (bytecode) for a speculative machine known as the Python Virtual Machine (PVM). The PVM executes the bytecode, guaranteeing that Python code can run on any stage without adjustment. This virtual machine show understands the issue of compactness, permitting Python code to be composed and compiled on one machine and executed on any other machine that has the PVM introduced.

4.3 SCREEN SHOTS



```
#Importing Dependencies

import pandas as pd
import numpy as np
import re
import nltk
from nltk.corpus import stopwords
from numpy import array

from keras.preprocessing.text import one_hot, Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Activation, Dropout, Dense
from keras.layers import Flatten, GlobalMaxPooling1D, Embedding, Conv1D, LSTM
from sklearn.model_selection import train_test_split
```

Fig 3.4.1 Importing requried libraries



```
mv_reviews
```

	review	sentiment
0	One of the other reviewers has mentioned that ...	positive
1	A wonderful little production. <br /><br />The...	positive
2	I thought this was a wonderful way to spend ti...	positive
3	Basically there's a family where a little boy ...	negative
4	Petter Mattei's "Love in the Time of Money" is...	positive
...	...	...
49995	I thought this movie did a down right good job...	positive
49996	Bad plot, bad dialogue, bad acting, idiotic di...	negative
49997	I am a Catholic taught in parochial elementary...	negative
49998	I'm going to have to disagree with the previou...	negative
49999	No one expects the Star Trek movies to be high...	negative

50000 rows × 2 columns

Fig 3.4.2 Dataset Description

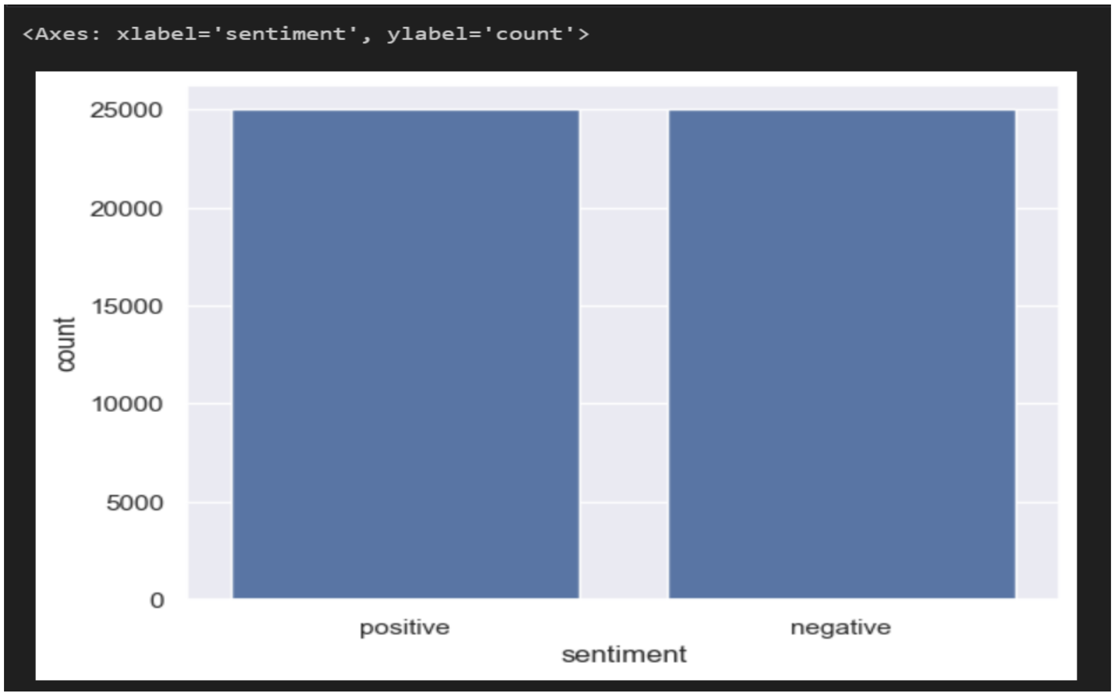
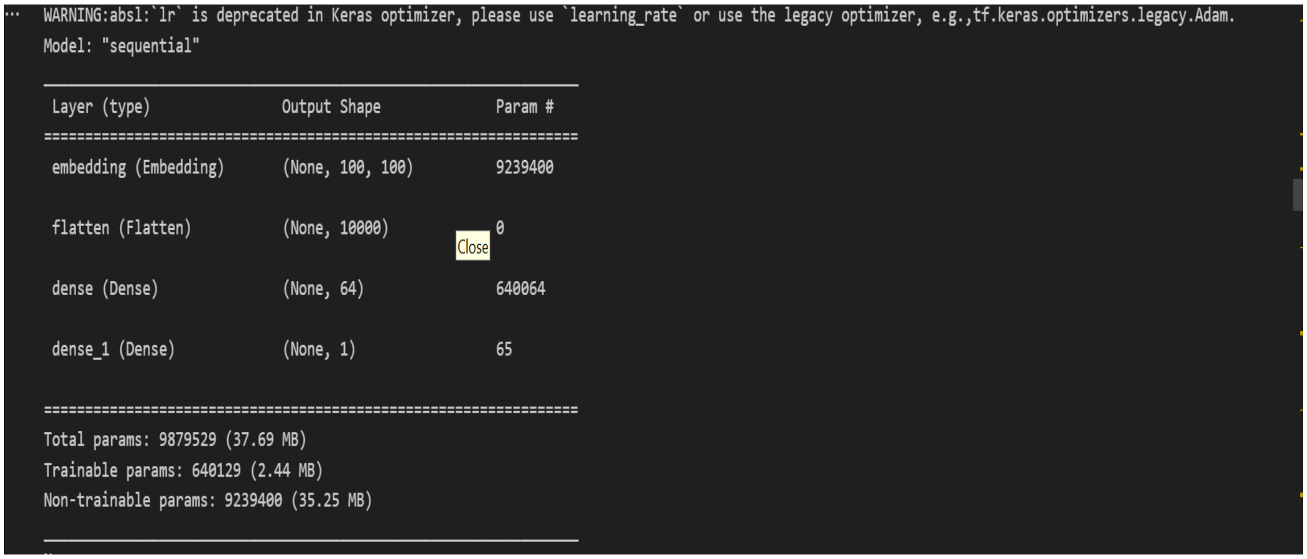
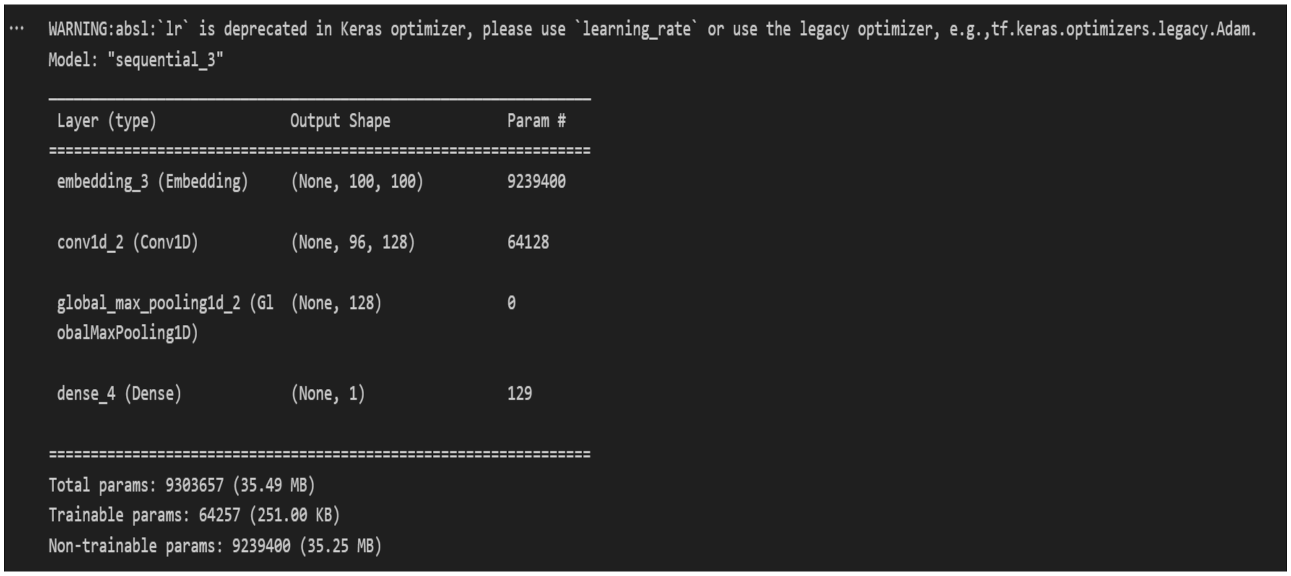


Fig 4.3.3 sentiment values distribution in dataset



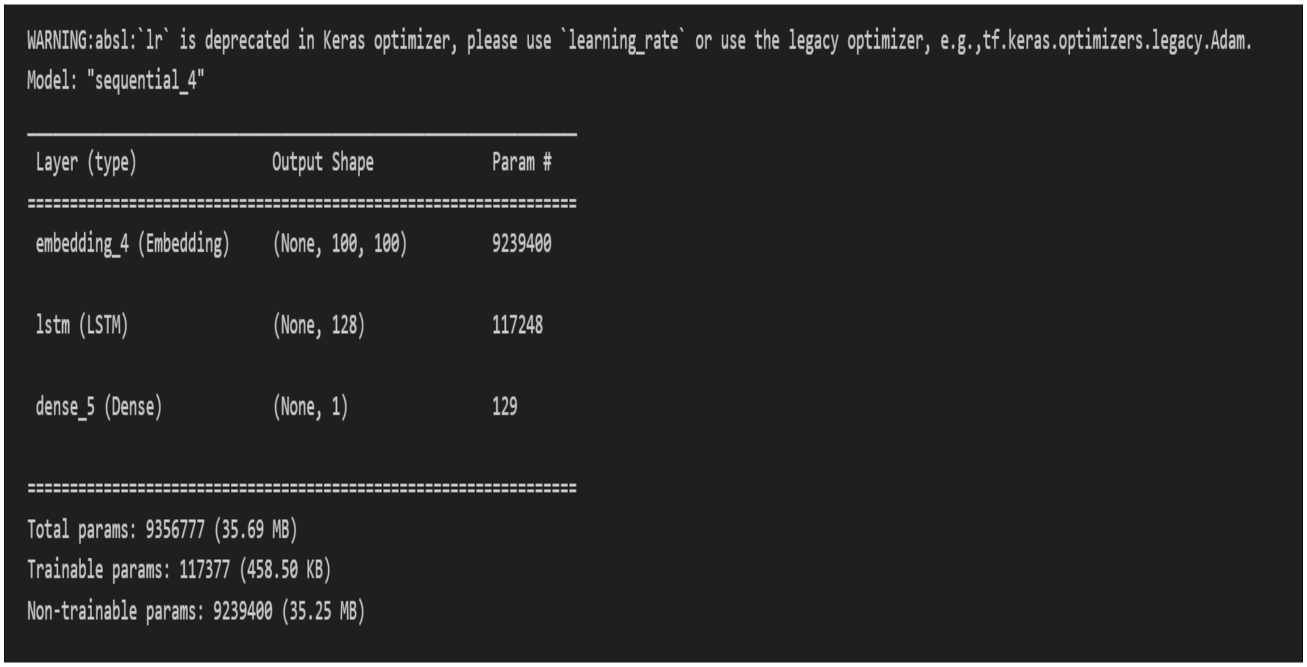
```
WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning_rate` or use the legacy optimizer, e.g.,tf.keras.optimizers.legacy.Adam.
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 100, 100)          9239400

 flatten (Flatten)           (None, 10000)             0

 dense (Dense)               (None, 64)                640064

 dense_1 (Dense)             (None, 1)                 65

=================================================================
Total params: 9879529 (37.69 MB)
Trainable params: 640129 (2.44 MB)
Non-trainable params: 9239400 (35.25 MB)
_________________________________________________________________
```

Fig 4.3.4 FNN Model label description



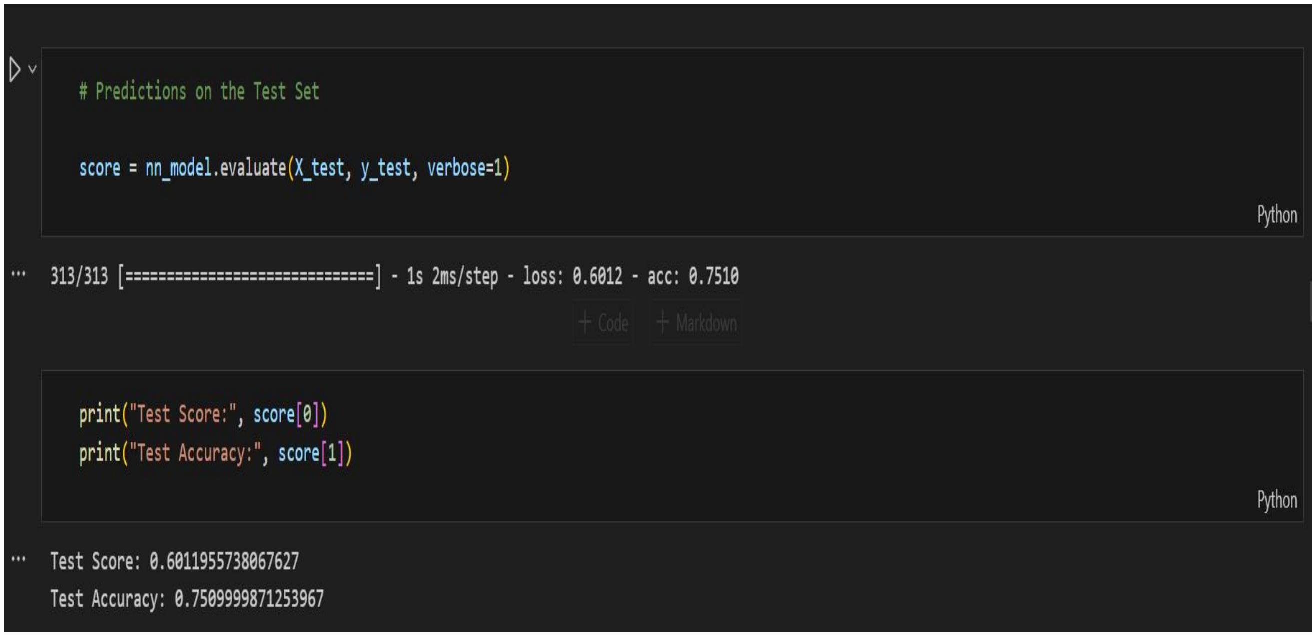
```
···  WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning_rate` or use the legacy optimizer, e.g.,tf.keras.optimizers.legacy.Adam.
     Model: "sequential_3"
     _________________________________________________________________
      Layer (type)                Output Shape              Param #   
     =================================================================
      embedding_3 (Embedding)     (None, 100, 100)          9239400   
                                                                      
      conv1d_2 (Conv1D)           (None, 96, 128)           64128     
                                                                      
      global_max_pooling1d_2 (Gl  (None, 128)               0         
      obalMaxPooling1D)                                               
                                                                      
      dense_4 (Dense)             (None, 1)                 129       
                                                                      
     =================================================================
     Total params: 9303657 (35.49 MB)
     Trainable params: 64257 (251.00 KB)
     Non-trainable params: 9239400 (35.25 MB)
```

Fig 4.3.5 CNN Model label description



```
WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning_rate` or use the legacy optimizer, e.g.,tf.keras.optimizers.legacy.Adam.
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 embedding_4 (Embedding)     (None, 100, 100)          9239400   
                                                                 
 lstm (LSTM)                 (None, 128)               117248    
                                                                 
 dense_5 (Dense)             (None, 1)                 129       
                                                                 
=================================================================
Total params: 9356777 (35.69 MB)
Trainable params: 117377 (458.50 KB)
Non-trainable params: 9239400 (35.25 MB)
```

Fig 4.3.6 LSTM Model label description



```
# Predictions on the Test Set

score = nn_model.evaluate(X_test, y_test, verbose=1)
```

```
313/313 [==============================] - 1s 2ms/step - loss: 0.6012 - acc: 0.7510
```

```
print("Test Score:", score[0])
print("Test Accuracy:", score[1])
```

```
Test Score: 0.6011955738067627
Test Accuracy: 0.7509999871253967
```

Fig 4.3.7 FNN Test Accuracy

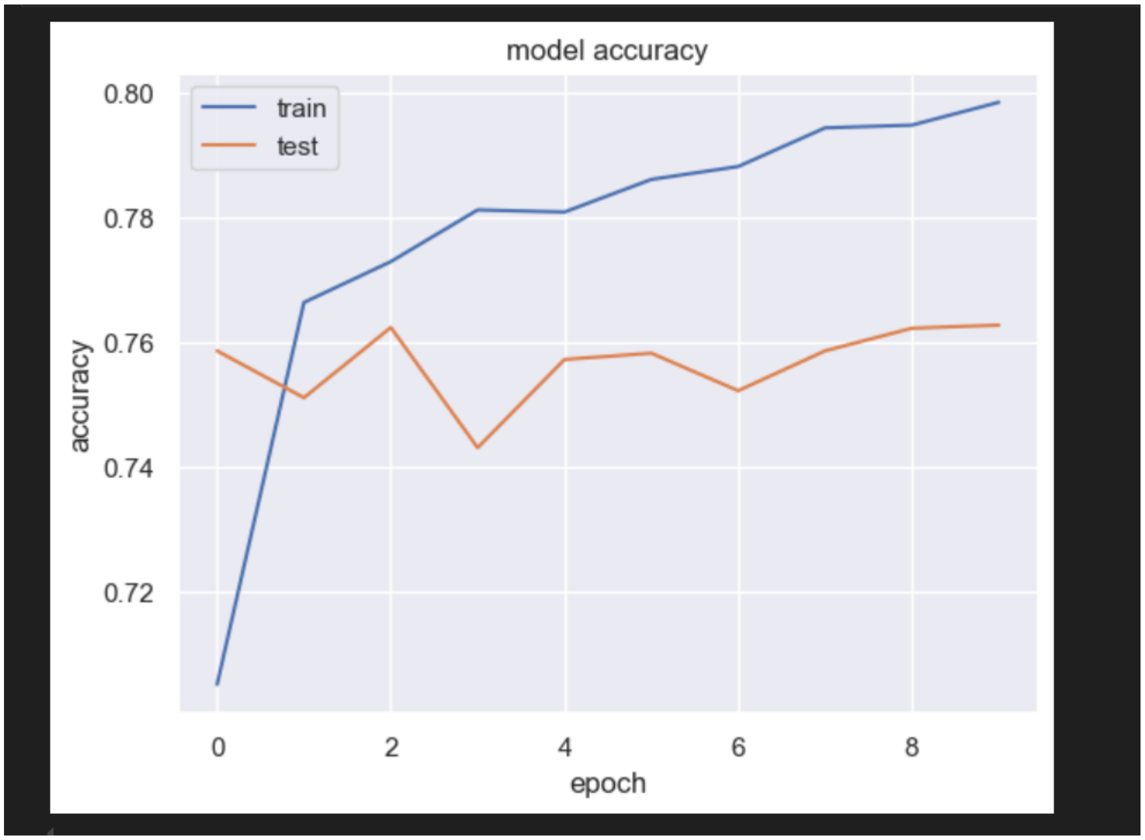


Fig 4.3.8 FNN Accuracy Graph

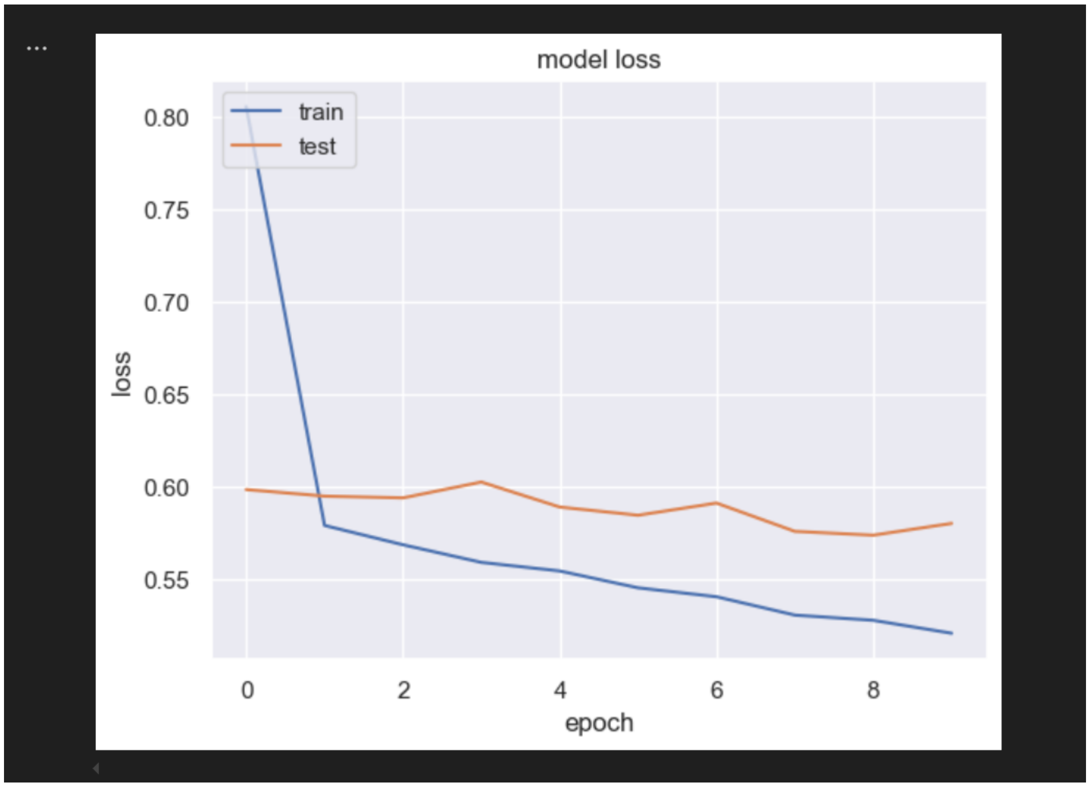
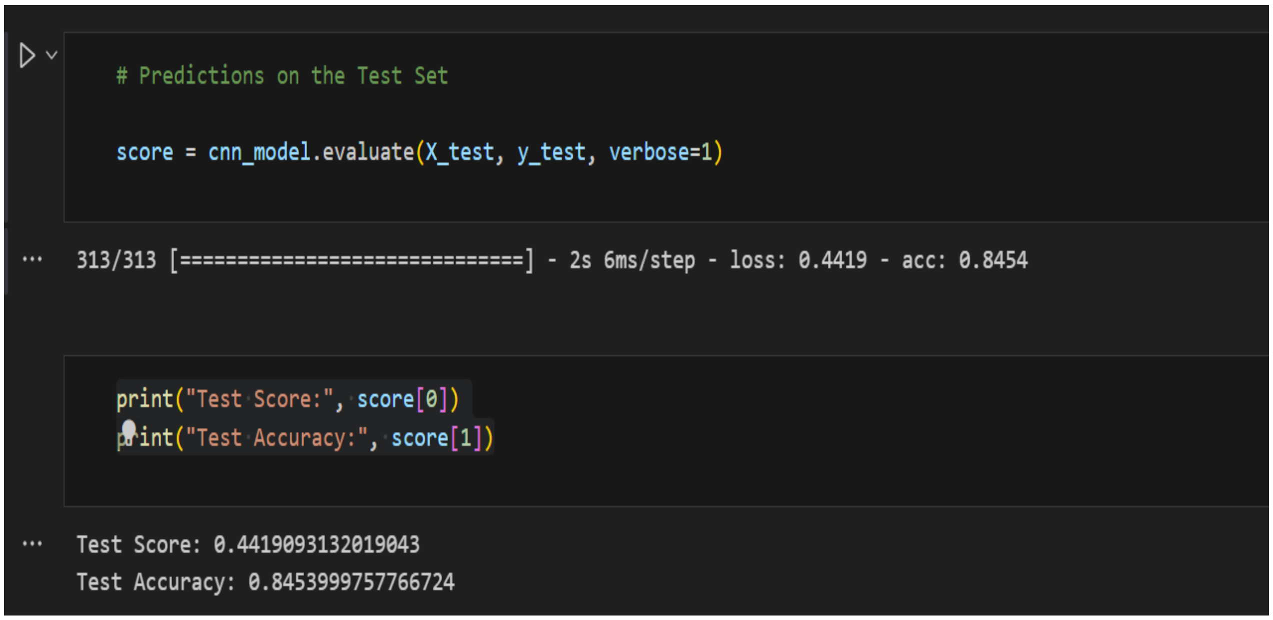


Fig 3.3.9 FNN loss function graph



```
# Predictions on the Test Set

score = cnn_model.evaluate(X_test, y_test, verbose=1)
```

```
313/313 [==============================] - 2s 6ms/step - loss: 0.4419 - acc: 0.8454
```

```
print("Test Score:", score[0])
print("Test Accuracy:", score[1])
```

```
Test Score: 0.4419093132019043
Test Accuracy: 0.8453999757766724
```

Fig 4.3.8 CNN Test Accuracy

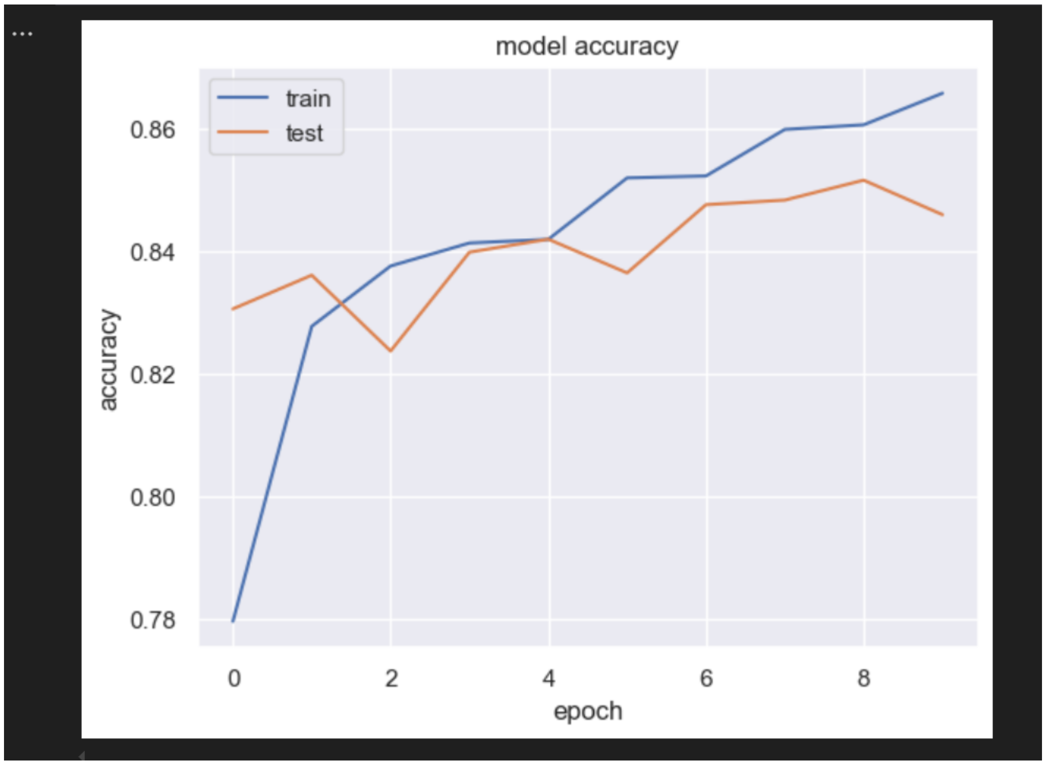


Fig 4.3.9 CNN Model Test Accuracy graph

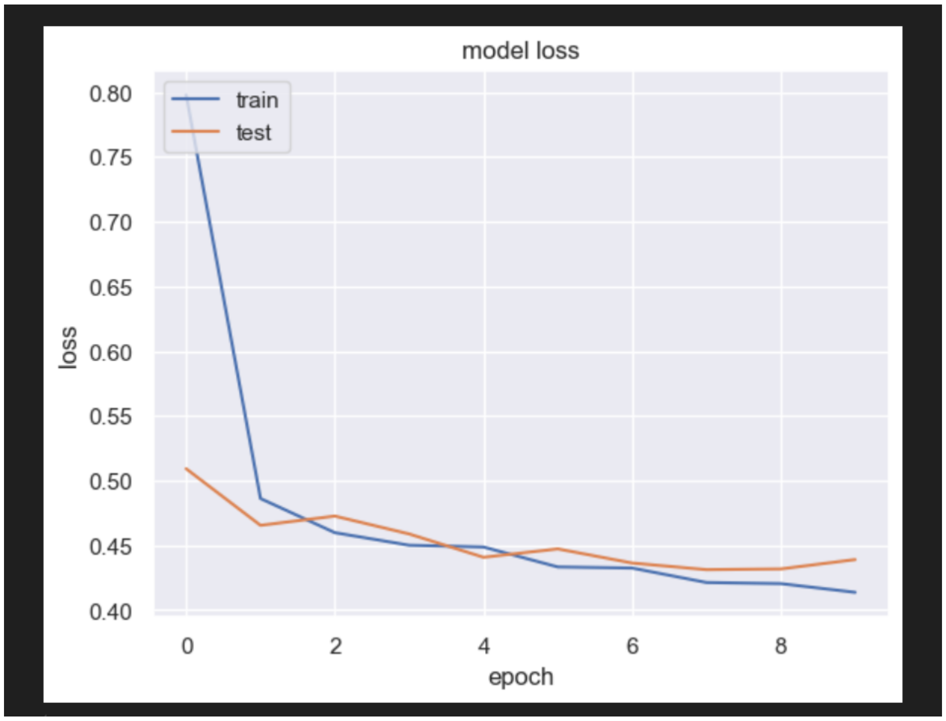
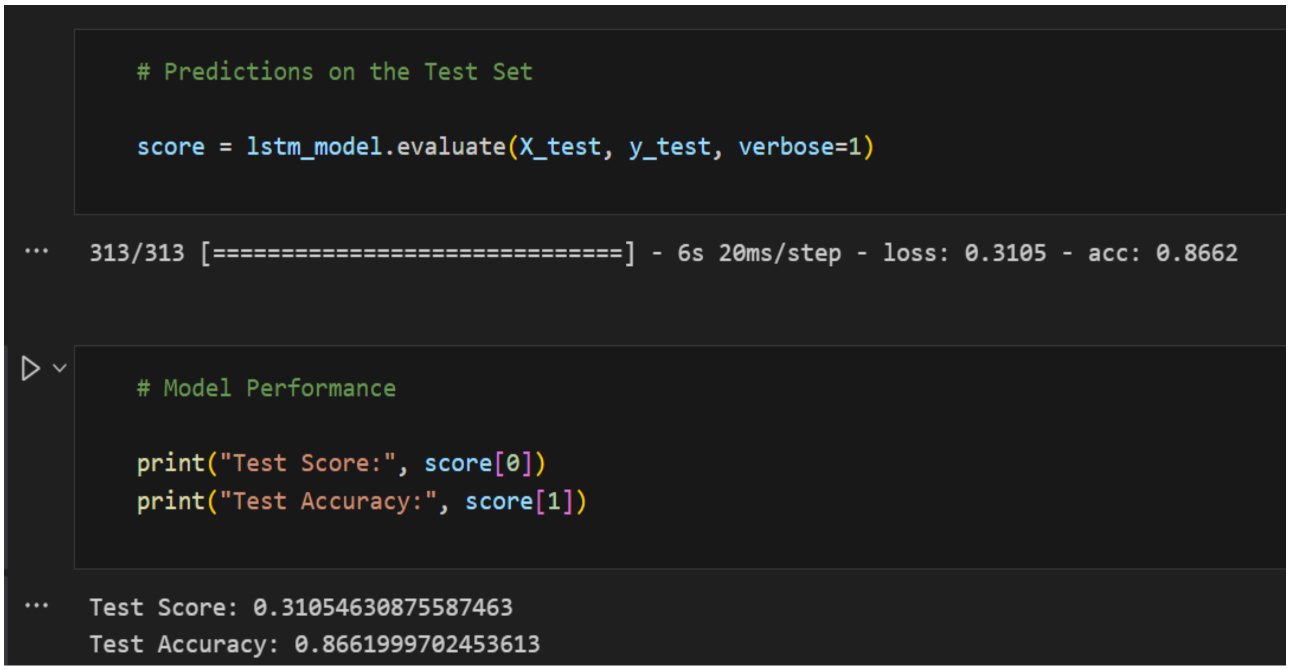


Fig 4.3.6 CNN loss function graph



```
# Predictions on the Test Set

score = lstm_model.evaluate(X_test, y_test, verbose=1)
```

```
313/313 [==============================] - 6s 20ms/step - loss: 0.3105 - acc: 0.8662
```

```
# Model Performance

print("Test Score:", score[0])
print("Test Accuracy:", score[1])
```

```
Test Score: 0.31054630875587463
Test Accuracy: 0.8661999702453613
```

Fig 4.3.7 LSTM Test Accuracy

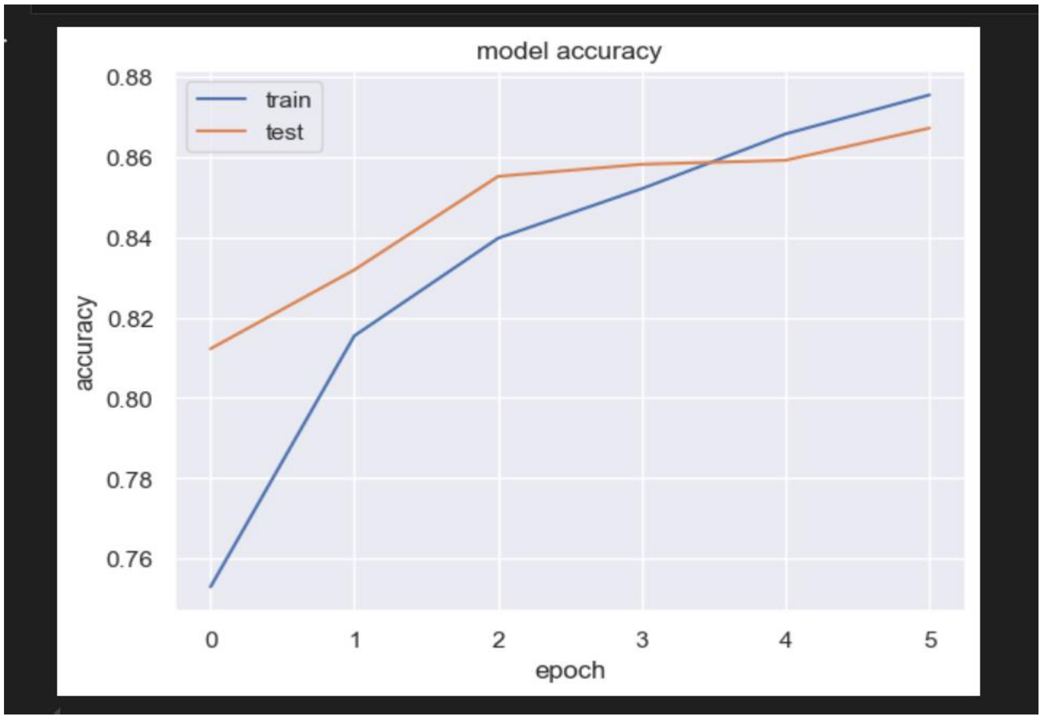


Fig 4.3.8 LSTM Test Accuracy grap

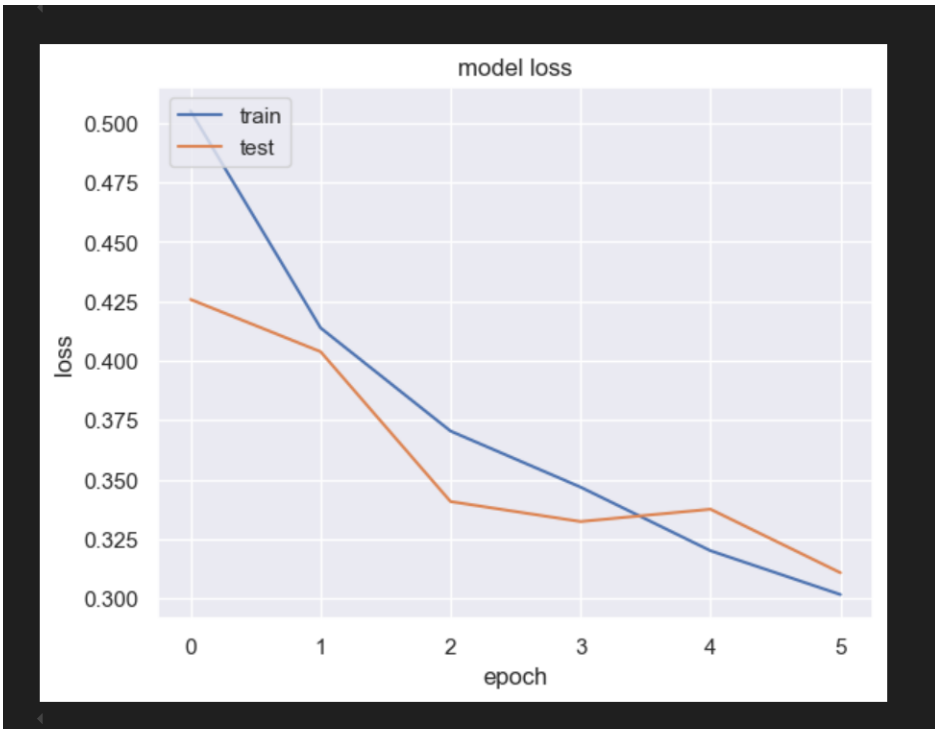
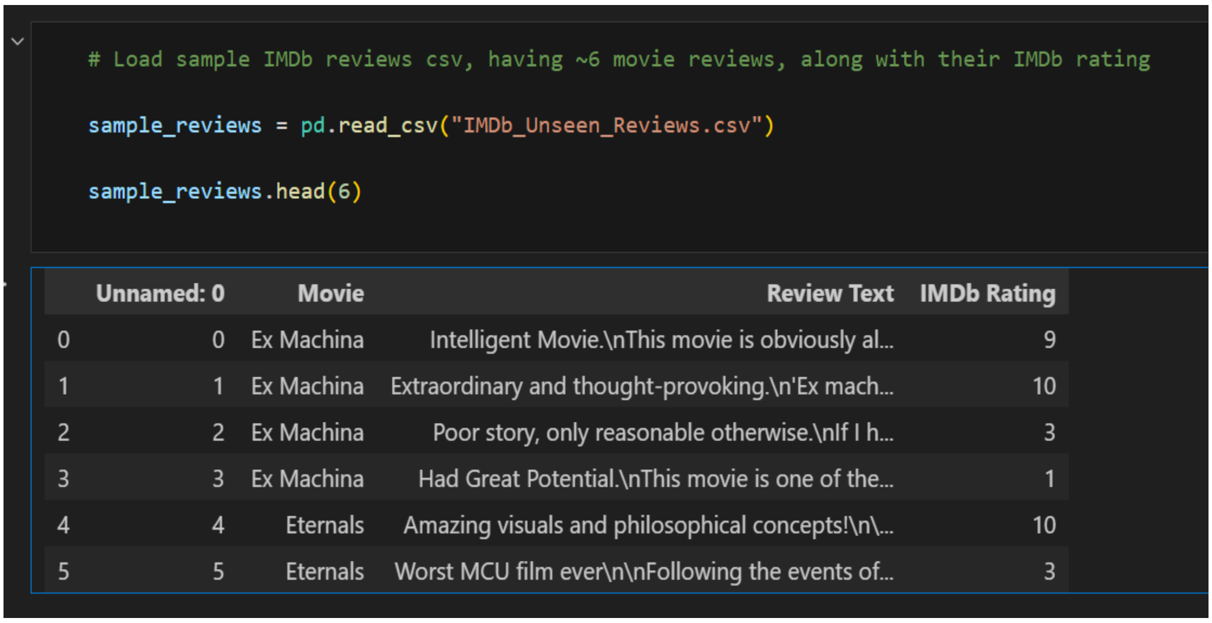


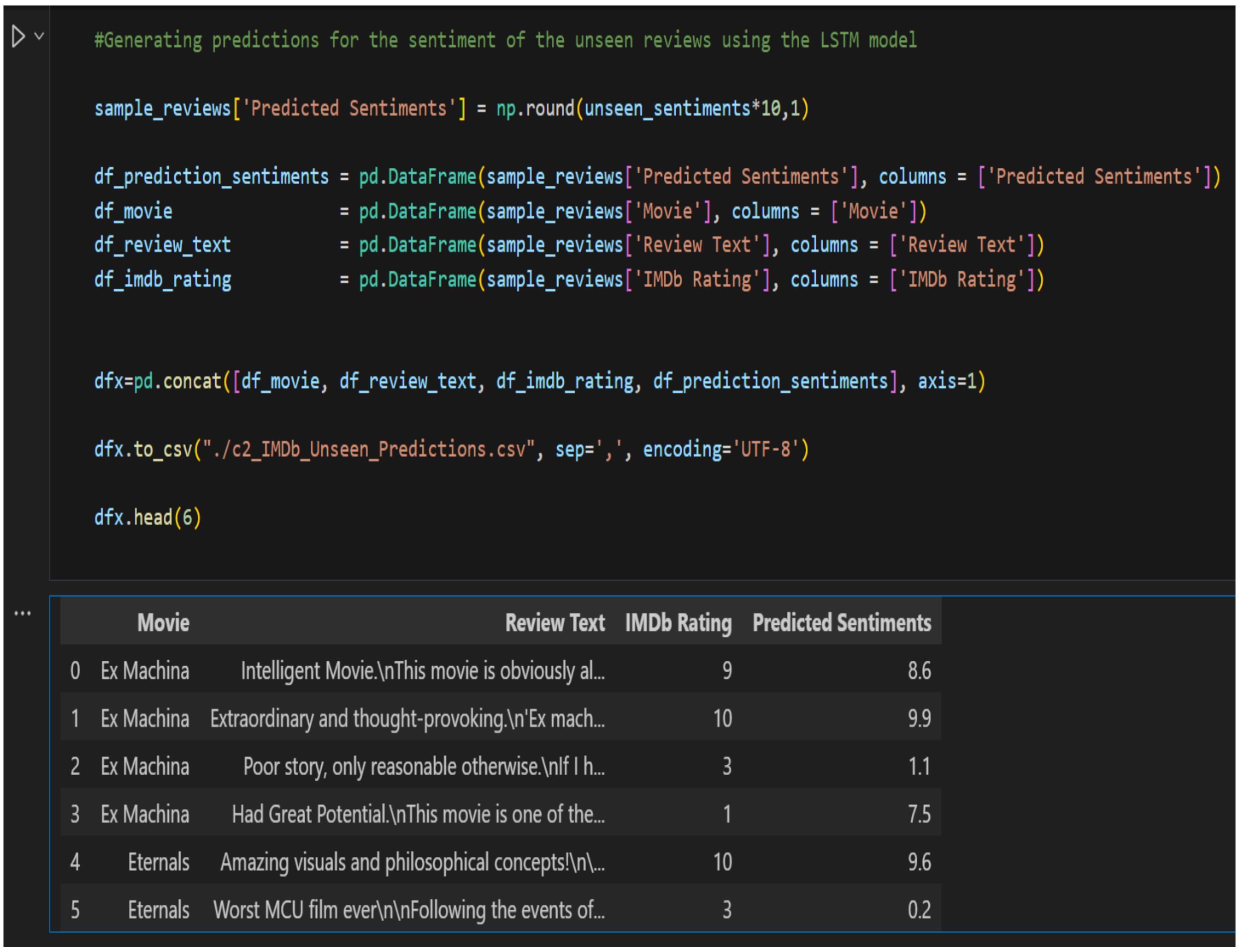
Fig 4.3.9 LSTM loss function graph



```
# Load sample IMDb reviews csv, having ~6 movie reviews, along with their IMDb rating

sample_reviews = pd.read_csv("IMDb_Unseen_Reviews.csv")

sample_reviews.head(6)
```

	Unnamed: 0	Movie	Review Text	IMDb Rating
0	0	Ex Machina	Intelligent Movie.\nThis movie is obviously al...	9
1	1	Ex Machina	Extraordinary and thought-provoking.\n'Ex mach...	10
2	2	Ex Machina	Poor story, only reasonable otherwise.\nIf I h...	3
3	3	Ex Machina	Had Great Potential.\nThis movie is one of the...	1
4	4	Eternals	Amazing visuals and philosophical concepts!\n\...	10
5	5	Eternals	Worst MCU film ever\n\nFollowing the events of...	3

Fig 4.3.10 Unseen data input to model



```
#Generating predictions for the sentiment of the unseen reviews using the LSTM model

sample_reviews['Predicted Sentiments'] = np.round(unseen_sentiments*10,1)

df_prediction_sentiments = pd.DataFrame(sample_reviews['Predicted Sentiments'], columns = ['Predicted Sentiments'])
df_movie                 = pd.DataFrame(sample_reviews['Movie'], columns = ['Movie'])
df_review_text           = pd.DataFrame(sample_reviews['Review Text'], columns = ['Review Text'])
df_imdb_rating           = pd.DataFrame(sample_reviews['IMDb Rating'], columns = ['IMDb Rating'])


dfx=pd.concat([df_movie, df_review_text, df_imdb_rating, df_prediction_sentiments], axis=1)

dfx.to_csv("./c2_IMDb_Unseen_Predictions.csv", sep=',', encoding='UTF-8')

dfx.head(6)
```

```
     Movie                                         Review Text  IMDb Rating  Predicted Sentiments
0  Ex Machina        Intelligent Movie.\nThis movie is obviously al...            9                   8.6
1  Ex Machina  Extraordinary and thought-provoking.\n'Ex mach...           10                   9.9
2  Ex Machina        Poor story, only reasonable otherwise.\nIf I h...            3                   1.1
3  Ex Machina        Had Great Potential.\nThis movie is one of the...            1                   7.5
4    Eternals    Amazing visuals and philosophical concepts!\n\...           10                   9.6
5    Eternals    Worst MCU film ever\n\nFollowing the events of...            3                   0.2
```

Fig 4.3.10 Predictive system of LSTM Model

4.4 Source code

```
import pandas as pd
import numpy as np
import re
import nltk
from nltk.corpus import stopwords
from numpy import array

from keras.preprocessing.text import one_hot, Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Activation, Dropout, Dense
from keras.layers import Flatten, GlobalMaxPooling1D, Embedding, Conv1D, LSTM
from sklearn.model_selection import train_test_split

mv_reviews = pd.read_csv("IMDB Dataset.csv")

mv_reviews
```

output :

```
         review                                            sentiment
0        One of the other reviewers has mentioned that ...  positive
1        A wonderful little production. <br /><br />The...  positive
2        I thought this was a wonderful way to spend ti...  positive
3        Basically there's a family where a little boy ...  negative
4        Petter Mattei's "Love in the Time of Money" is...  positive
...      ...                                                ...
49995    I thought this movie did a down right good job...  positive
49996    Bad plot, bad dialogue, bad acting, idiotic di...  negative
49997    I am a Catholic taught in parochial elementary...  negative
49998    I'm going to have to disagree with the previou...  negative
49999    No one expects the Star Trek movies to be high...  negative
50000 rows × 2 columns
```

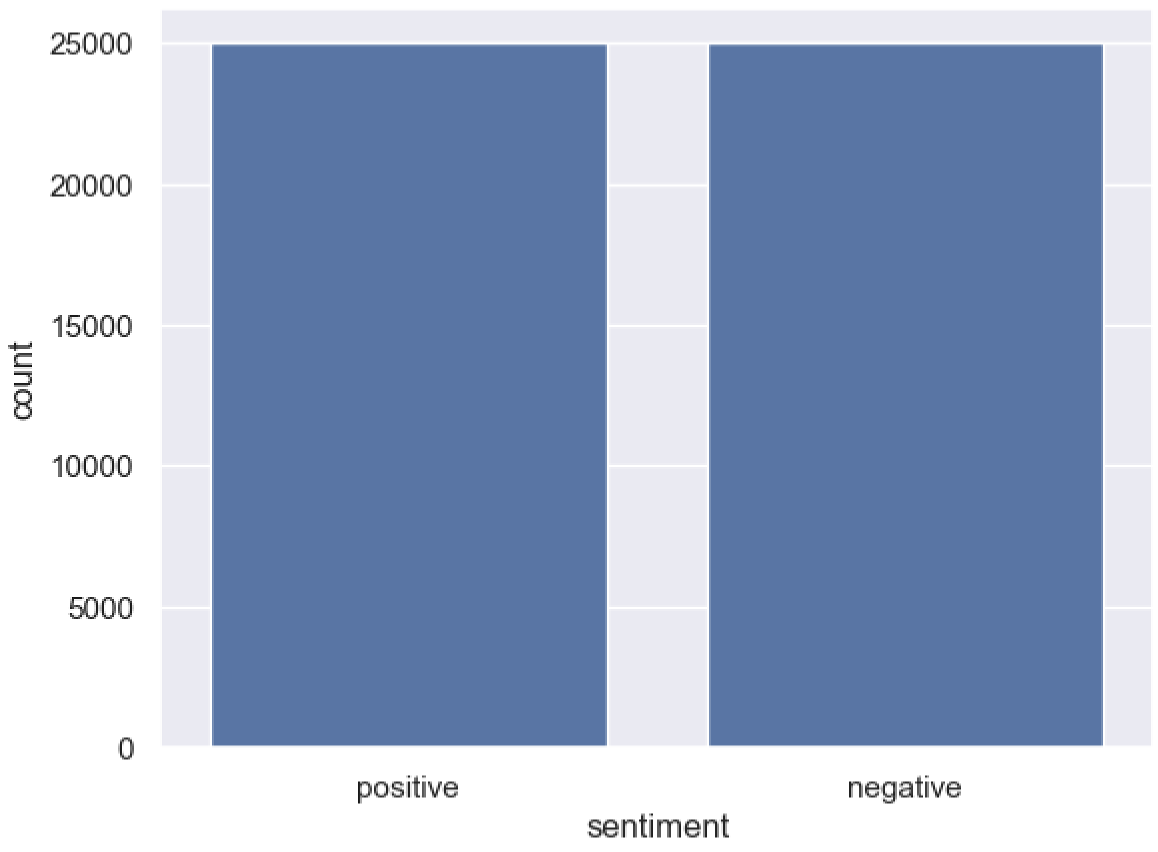
```
mv_reviews.shape

(50000, 2)
```

```
mv_reviews.isnull().values.any()
```

```
False
```

```
import seaborn as sns
sns.set()
sns.countplot(x='sentiment', data=mv_reviews)
```

output:



```
mv_reviews["review"][1]
```

output:

'A wonderful little production. <br /><br />The filming technique is very unassuming- very old-time-BBC fashion and gives a comforting, and sometimes discomforting, sense of realism to the entire piece. <br /><br />The actors are extremely well chosen- Michael Sheen not only "has got all the polari" but he has all the voices down pat too! You can truly see the seamless editing guided by the references to Williams\' diary entries, not only is it well worth the watching but it is a terrificly written and performed piece. A masterful production about one of the great master\'s of comedy and his life. <br /><br />The realism really comes home with the little things: the fantasy of the guard which, rather than use the traditional \'dream\' techniques remains solid then disappears. It plays on our knowledge and our senses, particularly with the scenes concerning

```
Orton and Halliwell and the sets (particularly of their flat with Halliwell\'s murals decorating
every surface) are terribly well done.'

TAG_RE = re.compile(r'<[^>]+>')

def remove_tags(text):
    '''Removes HTML tags: replaces anything between opening and closing <> with empty space'''

    return TAG_RE.sub('', text)

import nltk
nltk.download('stopwords')

def preprocess_text(sen):
    '''Cleans text data up, leaving only 2 or more char long non-stepwords composed of A-Z & a-z
only
    in lowercase'''

    sentence = sen.lower()

    # Remove html tags
    sentence = remove_tags(sentence)

    # Remove punctuations and numbers
    sentence = re.sub('[^a-zA-Z]', ' ', sentence)

    # Single character removal
    sentence = re.sub(r"\s+[a-zA-Z]\s+", ' ', sentence) # When we remove apostrophe from the word
"Mark's", the apostrophe is replaced by an empty space. Hence, we are left with single character
"s" that we are removing here.

    # Remove multiple spaces
    sentence = re.sub(r'\s+', ' ', sentence)  # Next, we remove all the single characters and replace it
by a space which creates multiple spaces in our text. Finally, we remove the multiple spaces from
our text as well.

    # Remove Stopwords
    pattern = re.compile(r'\b(' + r'|'.join(stopwords.words('english')) + r')\b\s*')
    sentence = pattern.sub('', sentence)

    return sentence

#Initializing preprocessing_text function on mv_reviews

X = []
sentences = list(mv_reviews['review'])
for sen in sentences:
    X.append(preprocess_text(sen))

# Looking for Sample cleaned up movie review
```

```
X[5]

# As we shall use Word Embeddings, stemming/lemmatization is not performed as a
preprocessing step here

'probably time favorite movie story selflessness sacrifice dedication noble cause preachy boring
never gets old despite seen times last years paul lukas performance brings tears eyes bette davis
one truly sympathetic roles delight kids grandma says like dressed midgets children makes fun
watch mother slow awakening happening world roof believable startling dozen thumbs movie '

# Converting sentiment labels to 0 & 1

y = mv_reviews['sentiment']

y = np.array(list(map(lambda x: 1 if x=="positive" else 0, y)))

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42)

# The train set will be used to train our deep learning models
# while test set will be used to evaluate how well our model performs

word_tokenizer = Tokenizer()
word_tokenizer.fit_on_texts(X_train)

X_train = word_tokenizer.texts_to_sequences(X_train)
X_test = word_tokenizer.texts_to_sequences(X_test)

vocab_length = len(word_tokenizer.word_index) + 1

vocab_length

# Padding all reviews to fixed length 100

maxlen = 100

X_train = pad_sequences(X_train, padding='post', maxlen=maxlen)
```

```
X_test = pad_sequences(X_test, padding='post', maxlen=maxlen)

# Load GloVe word embeddings and create an Embeddings Dictionary

from numpy import asarray
from numpy import zeros

embeddings_dictionary = dict()
glove_file = open('a2_glove.6B.100d.txt', encoding="utf8")

for line in glove_file:
    records = line.split()
    word = records[0]
    vector_dimensions = asarray(records[1:], dtype='float32')
    embeddings_dictionary [word] = vector_dimensions
glove_file.close()

embedding_matrix = zeros((vocab_length, 100))
for word, index in word_tokenizer.word_index.items():
    embedding_vector = embeddings_dictionary.get(word)
    if embedding_vector is not None:
        embedding_matrix[index] = embedding_vector

embedding_matrix.shape
(92394, 100)

#Neural Network Architecture
from keras import regularizers
from keras.optimizers import Adam
from keras.callbacks import EarlyStopping



nn_model = Sequential()
embedding_layer = Embedding(vocab_length, 100, weights=[embedding_matrix],
input_length=maxlen , trainable=False)
```

```
nn_model.add(embedding_layer)

nn_model.add(Flatten())
nn_model.add(Dense(64, activation='relu', kernel_regularizer=regularizers.l2(0.01)))
nn_model.add(Dense(1, activation='sigmoid'))

nn_model.compile(optimizer=Adam(lr=0.001), loss='binary_crossentropy', metrics=['acc'])


print(nn_model.summary())

nn_model_history = nn_model.fit(X_train, y_train, batch_size=128, epochs=10, verbose=1,
validation_split=0.2, callbacks=[EarlyStopping(patience=2)])
```

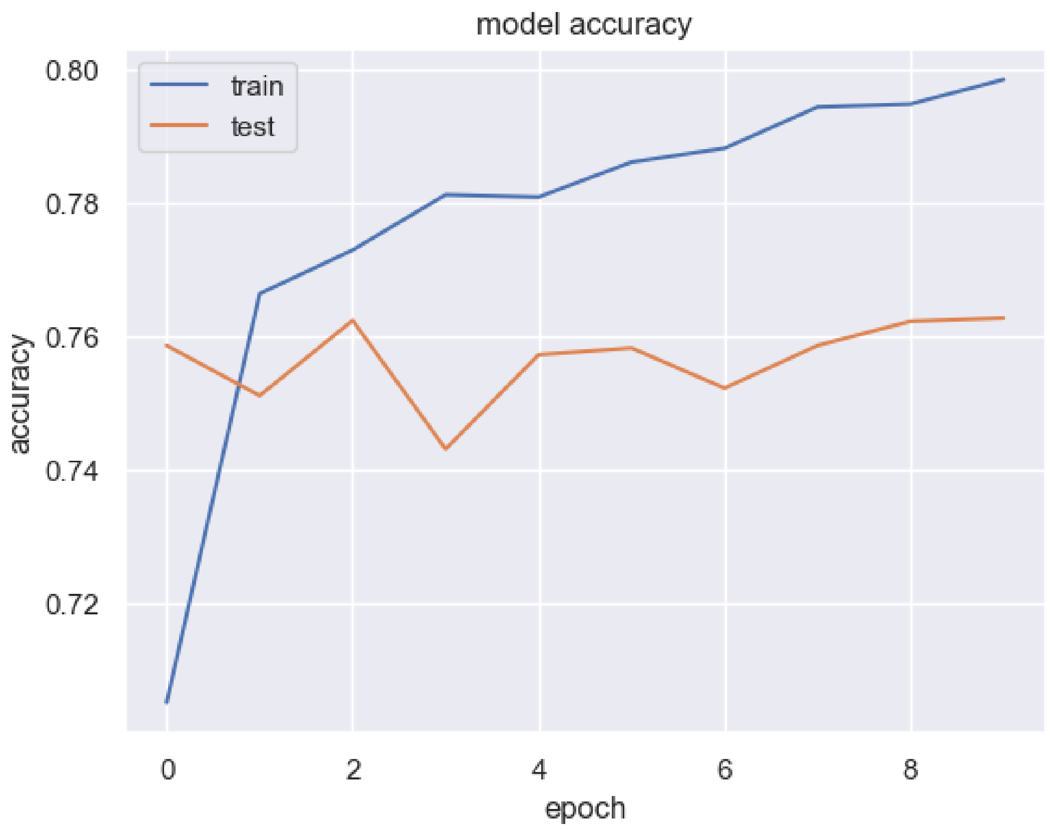
output:

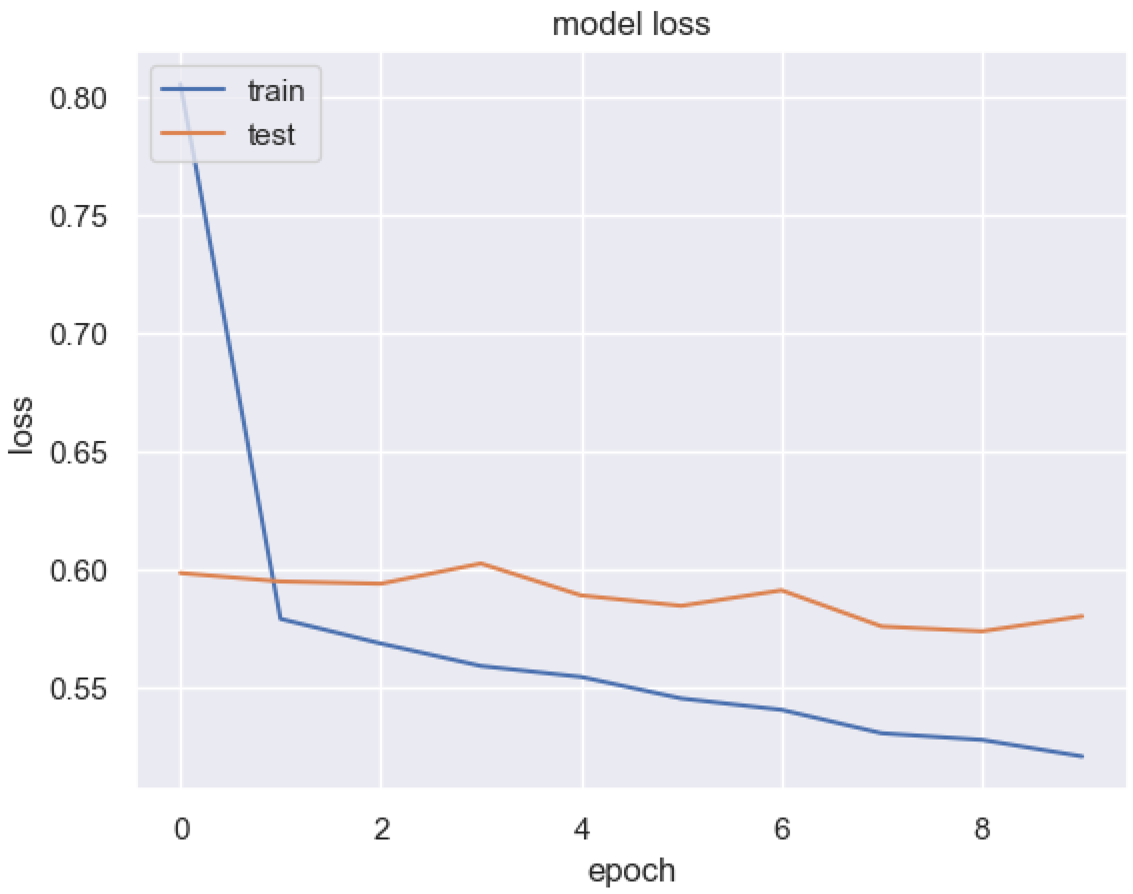
```
WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning_rate` or use the legacy
optimizer, e.g.,tf.keras.optimizers.legacy.Adam.
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 100, 100)          9239400

 flatten (Flatten)           (None, 10000)             0

 dense (Dense)               (None, 64)                640064

 dense_1 (Dense)             (None, 1)                 65

=================================================================
Total params: 9879529 (37.69 MB)
Trainable params: 640129 (2.44 MB)
Non-trainable params: 9239400 (35.25 MB)
_________________________________________________________________
None
```

```
Epoch 1/10
250/250 [==============================] - 4s 14ms/step - loss: 0.8055 - acc: 0.7053 -
val_loss: 0.5982 - val_acc: 0.7586
Epoch 2/10
250/250 [==============================] - 4s 14ms/step - loss: 0.5788 - acc: 0.7664 -
val_loss: 0.5946 - val_acc: 0.7511
Epoch 3/10
250/250 [==============================] - 4s 14ms/step - loss: 0.5683 - acc: 0.7729 -
val_loss: 0.5937 - val_acc: 0.7624
Epoch 4/10
250/250 [==============================] - 3s 13ms/step - loss: 0.5588 - acc: 0.7812 -
val_loss: 0.6023 - val_acc: 0.7431
Epoch 5/10
250/250 [==============================] - 3s 13ms/step - loss: 0.5541 - acc: 0.7808 -
val_loss: 0.5887 - val_acc: 0.7573
Epoch 6/10
250/250 [==============================] - 3s 13ms/step - loss: 0.5450 - acc: 0.7861 -
val_loss: 0.5843 - val_acc: 0.7582
Epoch 7/10
250/250 [==============================] - 3s 13ms/step - loss: 0.5402 - acc: 0.7881 -
val_loss: 0.5909 - val_acc: 0.7523
Epoch 8/10
250/250 [==============================] - 3s 13ms/step - loss: 0.5302 - acc: 0.7943 -
val_loss: 0.5756 - val_acc: 0.7586
Epoch 9/10
250/250 [==============================] - 3s 13ms/step - loss: 0.5274 - acc: 0.7947 -
val_loss: 0.5735 - val_acc: 0.7623
Epoch 10/10
250/250 [==============================] - 3s 14ms/step - loss: 0.5204 - acc: 0.7984 -
val_loss: 0.5799 - val_acc: 0.7628
```

```
# Predictions on the Test Set

score = nn_model.evaluate(X_test, y_test, verbose=1)
```

```
313/313 [==============================] - 1s 2ms/step - loss: 0.6012 - acc: 0.7510

print("Test Score:", score[0])
print("Test Accuracy:", score[1])

Test Score: 0.6011955738067627
Test Accuracy: 0.7509999871253967

import matplotlib.pyplot as plt

plt.plot(nn_model_history.history['acc'])
plt.plot(nn_model_history.history['val_acc'])

plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train','test'], loc='upper left')
plt.show()

plt.plot(nn_model_history.history['loss'])
plt.plot(nn_model_history.history['val_loss'])

plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train','test'], loc='upper left')
plt.show()
```


model accuracy
train
test
accuracy
0.80
0.78
0.76
0.74
0.72
0
2
4
6
8
epoch


model loss
train
test
loss
0.80
0.75
0.70
0.65
0.60
0.55
0
2
4
6
8
epoch

```
#Convolutional Neural Network architecture

from keras import regularizers
from keras.optimizers import Adam
from keras.callbacks import EarlyStopping
from keras.layers import Conv1D, GlobalMaxPooling1D

cnn_model = Sequential()

embedding_layer = Embedding(vocab_length, 100, weights=[embedding_matrix],
input_length=maxlen, trainable=False)
cnn_model.add(embedding_layer)

cnn_model.add(Conv1D(128, 5, activation='relu', kernel_regularizer=regularizers.l2(0.01)))
cnn_model.add(GlobalMaxPooling1D())
cnn_model.add(Dense(1, activation='sigmoid'))

cnn_model.compile(optimizer=Adam(lr=0.001), loss='binary_crossentropy', metrics=['acc'])

print(cnn_model.summary())

cnn_model_history = cnn_model.fit(X_train, y_train, batch_size=128, epochs=10, verbose=1,
validation_split=0.2, callbacks=[EarlyStopping(patience=2)])
```

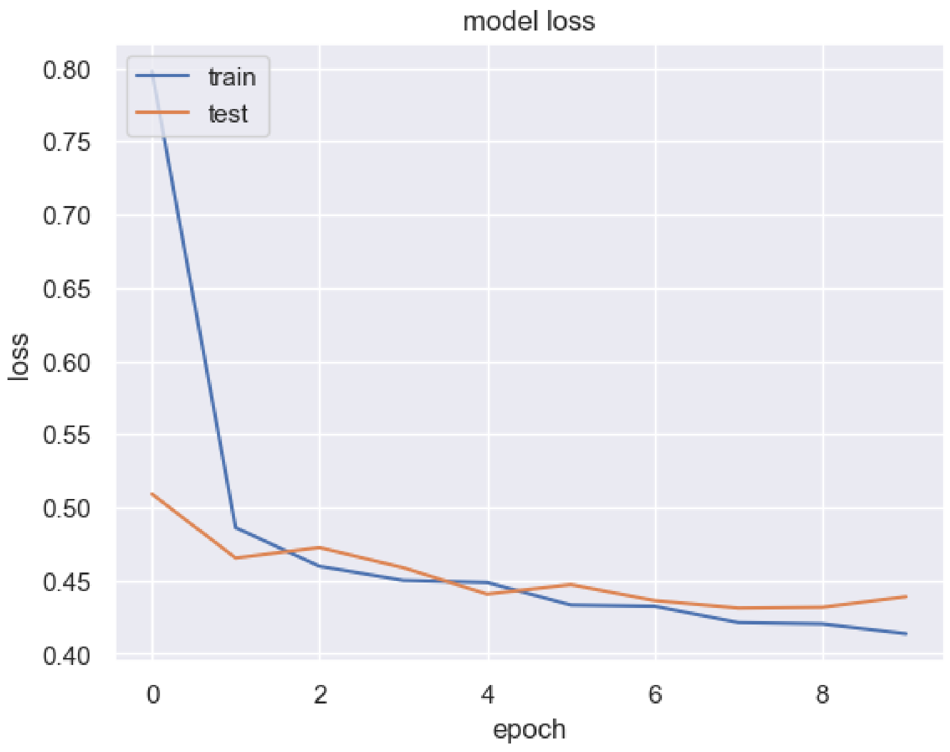
```
WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning_rate` or use the legacy
optimizer, e.g.,tf.keras.optimizers.legacy.Adam.
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_3 (Embedding)     (None, 100, 100)          9239400


 conv1d_2 (Conv1D)           (None, 96, 128)           64128
 global_max_pooling1d_2 (Gl  (None, 128)              0
 obalMaxPooling1D)

 dense_4 (Dense)            (None, 1)                129

=================================================================
Total params: 9303657 (35.49 MB)
Trainable params: 64257 (251.00 KB)
Non-trainable params: 9239400 (35.25 MB)
_________________________________________________________________
None
Epoch 1/10
250/250 [==============================] - 10s 35ms/step - loss: 0.7979 - acc: 0.7797 -
val_loss: 0.5092 - val_acc: 0.8306
Epoch 2/10
250/250 [==============================] - 8s 33ms/step - loss: 0.4862 - acc: 0.8278 -
val_loss: 0.4655 - val_acc: 0.8361
Epoch 3/10
250/250 [==============================] - 8s 32ms/step - loss: 0.4598 - acc: 0.8376 -
val_loss: 0.4726 - val_acc: 0.8238
Epoch 4/10
250/250 [==============================] - 8s 32ms/step - loss: 0.4501 - acc: 0.8414 -
val_loss: 0.4588 - val_acc: 0.8399
Epoch 5/10
250/250 [==============================] - 8s 32ms/step - loss: 0.4487 - acc: 0.8420 -
val_loss: 0.4408 - val_acc: 0.8420
Epoch 6/10
250/250 [==============================] - 9s 35ms/step - loss: 0.4334 - acc: 0.8520 -
val_loss: 0.4473 - val_acc: 0.8365
Epoch 7/10
250/250 [==============================] - 9s 34ms/step - loss: 0.4325 - acc: 0.8523 -
val_loss: 0.4364 - val_acc: 0.8476
Epoch 8/10
250/250 [==============================] - 9s 35ms/step - loss: 0.4213 - acc: 0.8599 -
val_loss: 0.4313 - val_acc: 0.8484
```

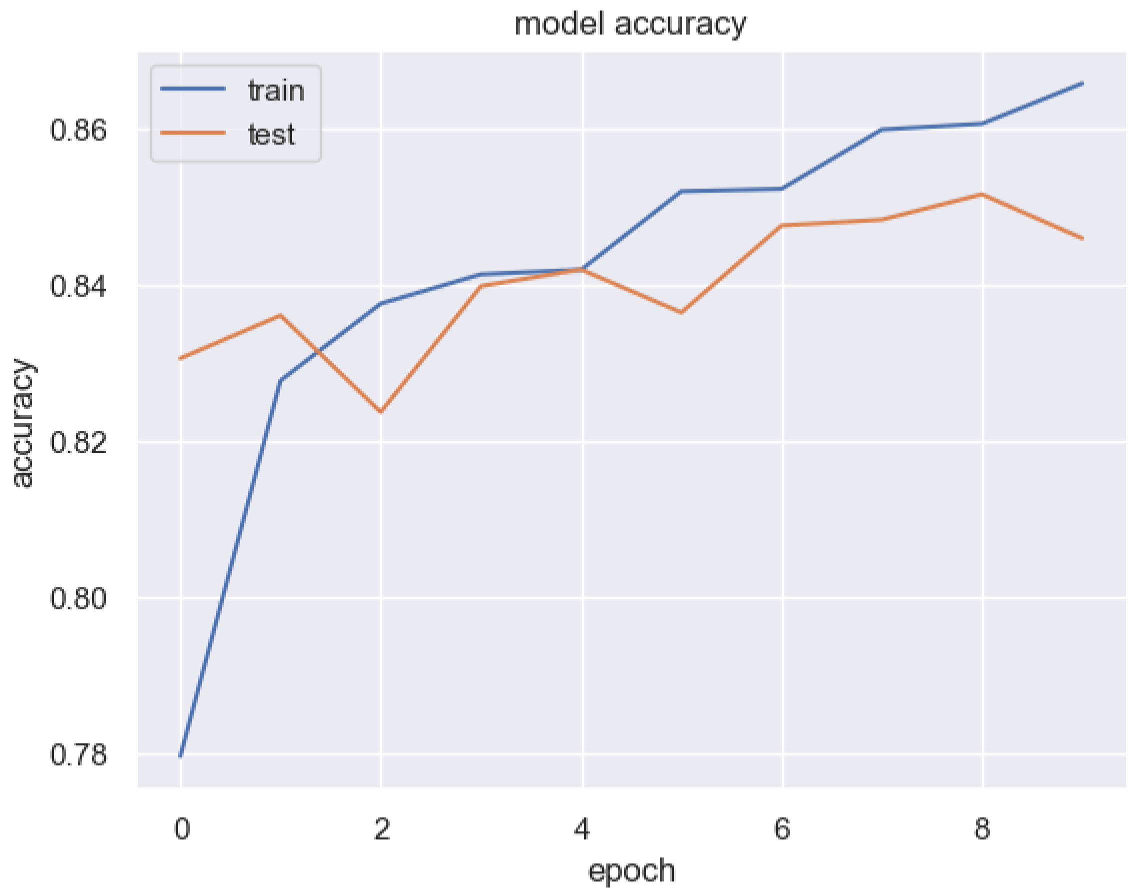
```
Epoch 9/10
250/250 [==============================] - 9s 34ms/step - loss: 0.4205 - acc: 0.8607 -
val_loss: 0.4318 - val_acc: 0.8516
Epoch 10/10
250/250 [==============================] - 9s 35ms/step - loss: 0.4138 - acc: 0.8658 -
val_loss: 0.4390 - val_acc: 0.8460
```

```
print("Test Score:", score[0])
print("Test Accuracy:", score[1])
```

```
Test Score: 0.4419093132019043
Test Accuracy: 0.8453999757766724
```

```
import matplotlib.pyplot as plt

plt.plot(cnn_model_history.history['acc'])
plt.plot(cnn_model_history.history['val_acc'])

plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train','test'], loc = 'upper left')
plt.show()

plt.plot(cnn_model_history.history['loss'])
plt.plot(cnn_model_history.history['val_loss'])

plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train','test'], loc = 'upper left')
plt.show()
```



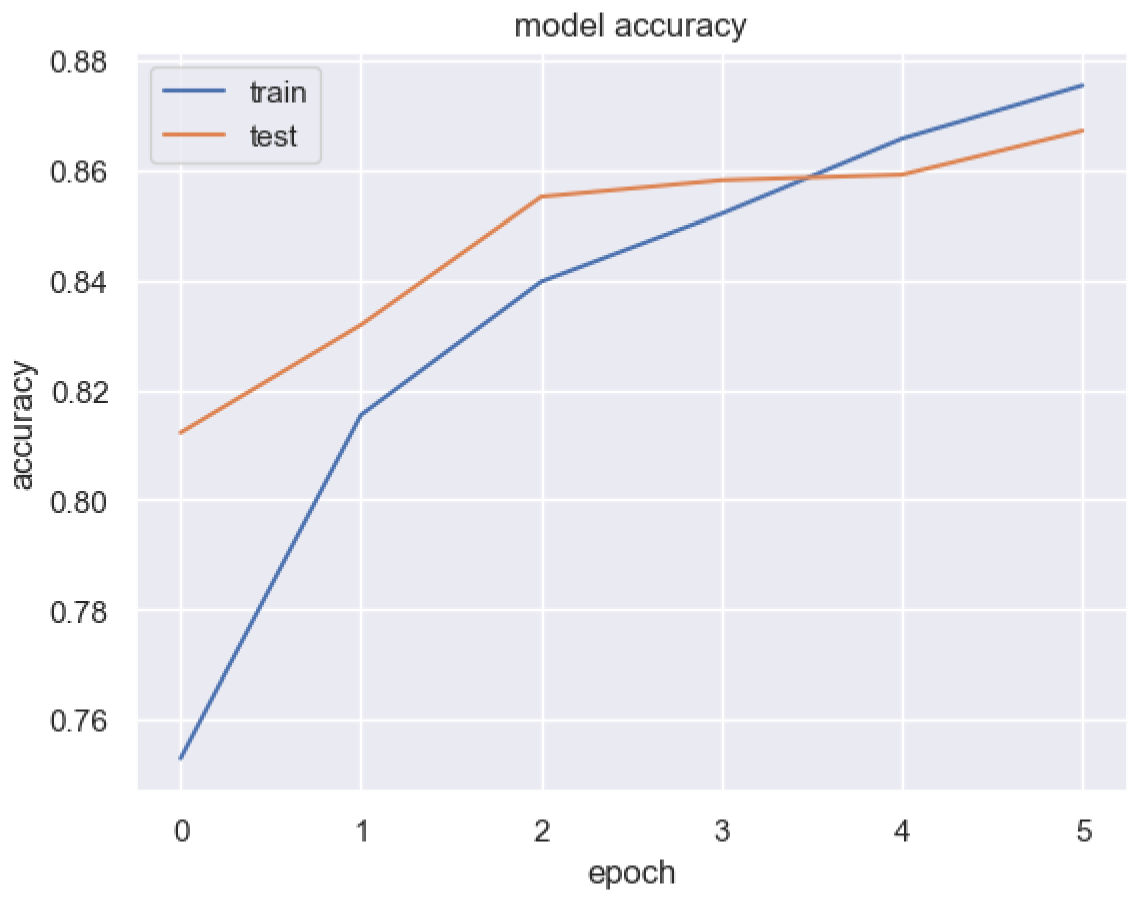


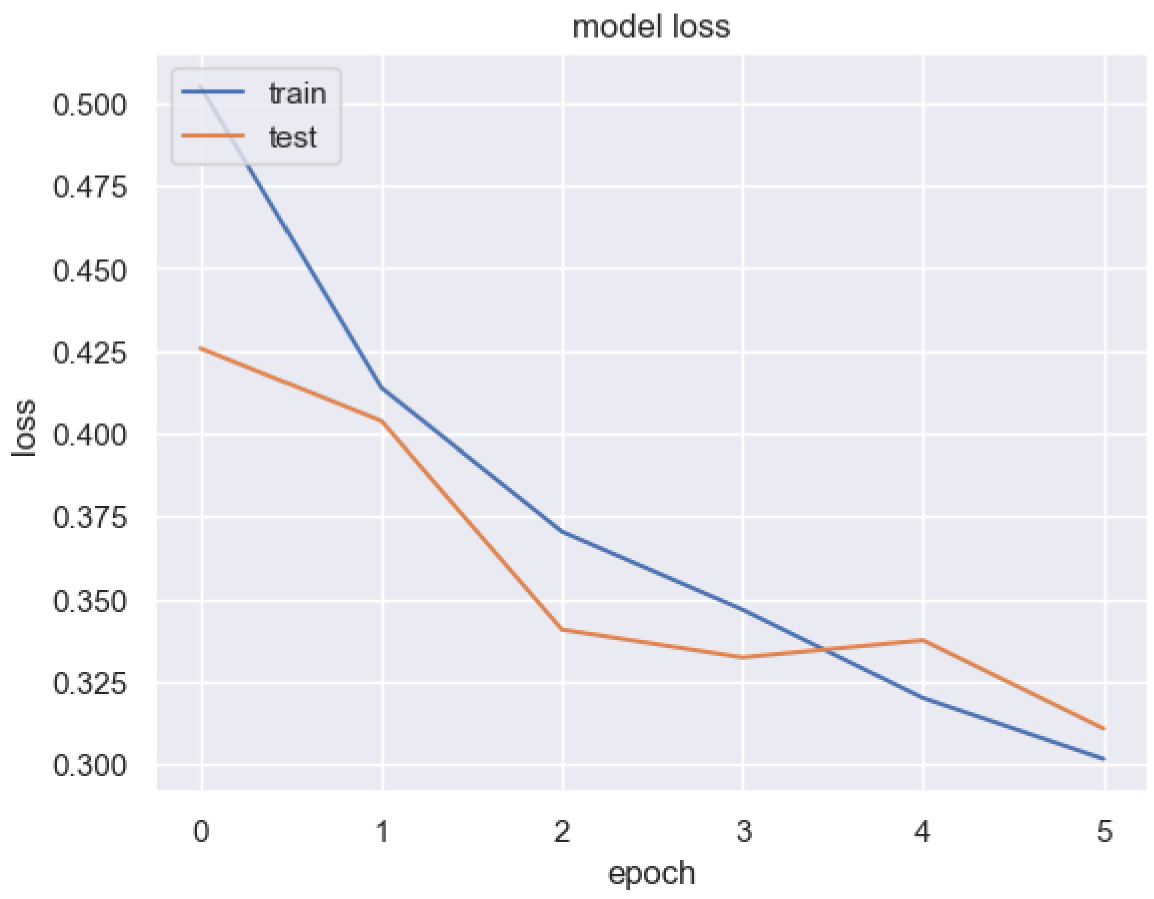
```
# Recurrent Neural Network architecture

from keras import regularizers
from keras.optimizers import Adam
from keras.callbacks import EarlyStopping
from keras.layers import LSTM


lstm_model = Sequential()

embedding_layer = Embedding(vocab_length, 100, weights=[embedding_matrix],
input_length=maxlen, trainable=False)
lstm_model.add(embedding_layer)

lstm_model.add(LSTM(128))

lstm_model.add(Dense(1, activation='sigmoid'))

lstm_model.compile(optimizer=Adam(lr=0.001), loss='binary_crossentropy', metrics=['acc'])

print(lstm_model.summary())

lstm_model_history = lstm_model.fit(X_train, y_train, batch_size=128, epochs=6, verbose=1,
validation_split=0.2)
```

output :

```
WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning_rate` or use the legacy
optimizer, e.g.,tf.keras.optimizers.legacy.Adam.
Model: "sequential_4"
_________________________________________________________________
 Layer (type)              Output Shape              Param #
=================================================================
 embedding_4 (Embedding)     (None, 100, 100)          9239400


 lstm (LSTM)                (None, 128)               117248


 dense_5 (Dense)             (None, 1)                 129


=================================================================
Total params: 9356777 (35.69 MB)
Trainable params: 117377 (458.50 KB)
Non-trainable params: 9239400 (35.25 MB)
_________________________________________________________________
None
Epoch 1/6
250/250 [==============================] - 60s 230ms/step - loss: 0.5050 - acc: 0.7530
- val_loss: 0.4258 - val_acc: 0.8123
```

```
Epoch 2/6
250/250 [==============================] - 62s 246ms/step - loss: 0.4138 - acc: 0.8155
- val_loss: 0.4038 - val_acc: 0.8319
Epoch 3/6
250/250 [==============================] - 58s 231ms/step - loss: 0.3704 - acc: 0.8397
- val_loss: 0.3408 - val_acc: 0.8551
Epoch 4/6
250/250 [==============================] - 58s 233ms/step - loss: 0.3469 - acc: 0.8521
- val_loss: 0.3324 - val_acc: 0.8581
Epoch 5/6
250/250 [==============================] - 63s 253ms/step - loss: 0.3201 - acc: 0.8657
- val_loss: 0.3376 - val_acc: 0.8591
Epoch 6/6
250/250 [==============================] - 60s 239ms/step - loss: 0.3017 - acc: 0.8754
- val_loss: 0.3108 - val_acc: 0.8671

# Predictions on the Test Set

score = lstm_model.evaluate(X_test, y_test, verbose=1)

313/313 [==============================] - 6s 20ms/step - loss: 0.3105 - acc: 0.8662

# Model Performance

print("Test Score:", score[0])
print("Test Accuracy:", score[1])

Test Score: 0.310546308755587463
Test Accuracy: 0.8661999702453613

# Model Performance Charts


plt.plot(lstm_model_history.history['acc'])
plt.plot(lstm_model_history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train','test'], loc='upper left')
plt.show()

plt.plot(lstm_model_history.history['loss'])
plt.plot(lstm_model_history.history['val_loss'])

plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train','test'], loc='upper left')
plt.show()
```





```
# Saving the model as a h5 file for possible use later

lstm_model.save(f"./c1_Lstm_model_acc_{round(score[1], 3)}.h5", save_format='h5')
```

```
C:\Users\mohammad sazid\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\engine\training.py:3079: UserWarning: You are saving your model as an HDF5 file via `model.save()`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(
```

```
# Load sample IMDb reviews csv, having ~6 movie reviews, along with their IMDb rating

sample_reviews = pd.read_csv("IMDb_Unseen_Reviews.csv")

sample_reviews.head(6)
```

```
Unnamed: 0  Movie  Review Text   IMDb Rating
0      0       Ex Machina    Intelligent Movie.\nThis movie is obviously al...     9
```

```
1       1       Ex Machina      Extraordinary and thought-provoking.\n'Ex mach...  10
2       2       Ex Machina      Poor story, only reasonable otherwise.\nIf I h...  3
3       3       Ex Machina      Had Great Potential.\nThis movie is one of the...  1
4       4       Eternals        Amazing visuals and philosophical concepts!\n\...  10
5       5       Eternals        Worst MCU film ever\n\nFollowing the events of...  3
```

```
# Preprocess review text with earlier defined preprocess_text function

unseen_reviews = sample_reviews['Review Text']

unseen_processed = []
for review in unseen_reviews:
  review = preprocess_text(review)
  unseen_processed.append(review)

# Tokenising instance with earlier trained tokeniser
unseen_tokenized = word_tokenizer.texts_to_sequences(unseen_processed)

# Pooling instance to have maxlength of 100 tokens
unseen_padded = pad_sequences(unseen_tokenized, padding='post', maxlen=maxlen)

# Passing tokenised instance to the LSTM model for predictions
unseen_sentiments = lstm_model.predict(unseen_padded)

unseen_sentiments
1/1 [==============================] - 0s 480ms/step
array([[0.8567016 ],
       [0.9899078 ],
       [0.10935952],
       [0.7451177 ],
       [0.96416646],
       [0.01640477]], dtype=float32)
#Generating predictions for the sentiment of the unseen reviews using the LSTM model

sample_reviews['Predicted Sentiments'] = np.round(unseen_sentiments*10,1)
```

```
df_prediction_sentiments = pd.DataFrame(sample_reviews['Predicted Sentiments'], columns =
['Predicted Sentiments'])
df_movie              = pd.DataFrame(sample_reviews['Movie'], columns = ['Movie'])
df_review_text         = pd.DataFrame(sample_reviews['Review Text'], columns = ['Review
Text'])
df_imdb_rating         = pd.DataFrame(sample_reviews['IMDb Rating'], columns = ['IMDb
Rating'])
dfx=pd.concat([df_movie, df_review_text, df_imdb_rating, df_prediction_sentiments], axis=1)
dfx.to_csv("./c2_IMDb_Unseen_Predictions.csv", sep=',', encoding='UTF-8')
dfx.head(6)
```

```
     Movie  Review Text   IMDb Rating  Predicted Sentiments
0    Ex Machina    Intelligent Movie.\nThis movie is obviously al...       9      8.6
1    Ex Machina    Extraordinary and thought-provoking.\n'Ex mach...  10     9.9
2    Ex Machina    Poor story, only reasonable otherwise.\nIf I h...       3      1.1
3    Ex Machina    Had Great Potential.\nThis movie is one of the...       1      7.5
4    Eternals      Amazing visuals and philosophical concepts!\n\...  10     9.6
5    Eternals      Worst MCU film ever\n\nFollowing the events of...  3      0.2
```

CHAPTER -5
SYSTEM TESTING

5.1 TESTING DESCRIPTION

The reason of testing is to find blunders. Testing is the method of attempting to find each conceivable blame or shortcoming in a work item. It gives a way to check the usefulness of components, sub-assemblies, assemblies and/or a wrapped up item It is the method of working out program with the aim of guaranteeing that the software package meets its prerequisites and client desires and does not come up short in an unsatisfactory way. There are different sorts of test. Each test sort addresses a particular testing prerequisite.

5.2 Types of Tests

Unit testing

Unit testing includes the plan of test cases that approve that the inner program rationale is working legitimately, which program inputs create substantial yields. All choice branches and inside code stream ought to be approved. It is the testing of person computer program units of the application .it is done after the completion of an person unit before integration. This is often a auxiliary testing, that depends on information of its development and is obtrusive. Unit tests perform essential tests at component level and test a particular trade handle, application, and/or framework arrangement. Unit tests guarantee that each interesting way of a commerce prepare performs precisely to the recorded determinations and contains clearly characterized inputs and anticipated comes about.

Integration testing

Integration tests are outlined to test coordinates program components to decide on the off chance that they really run as one program. Testing is occasion driven and is more concerned with the essential result of screens or areas. Integration tests illustrate that in spite of the fact that the components were separately fulfillment, as appeared by effectively unit testing, the combination of components is rectify and reliable. Integration testing is particularly pointed at uncovering the issues that emerge from the combination of components.

Functional test

Functional tests give efficient shows that capacities tried are accessible as indicated by the commerce and specialized necessities, framework documentation, and client manuals. Useful testing is centered on the taking after things:

Substantial Input : distinguished classes of substantial input must be acknowledged.

Invalid Input : distinguished classes of invalid input must be rejected.

Capacities : recognized capacities must be worked out.

Yield : recognized classes of application yields must be worked out.

Systems/Procedures: meddle frameworks or strategies must be conjured.

System Testing

System testing guarantees that the complete coordinates package meets prerequisites. It tests a setup to guarantee known and unsurprising comes about. An illustration of framework testing is the arrangement arranged framework integration test. System testing is based on handle depictions and streams, emphasizing pre-driven handle joins and integration focuses.

White Box Testing

White Box Testing could be a strategy where the analyzer has nitty gritty information of the inside workings, structure, and code of the computer program. This approach permits for a intensive examination of the framework, guaranteeing that all pathways, branches, and inner rationale are working accurately. It is especially valuable for distinguishing covered up mistakes that are not obvious through dark box testing strategies. By understanding the inside plan, analyzers can make more comprehensive test cases that cover edge cases and potential security vulnerabilities.

Black Box Testing

Black Box Testing is testing the computer program without any information of the internal workings, structure or dialect of the module being tried. Black box tests, as most other sorts of tests, must be composed from a authoritative source archive, such as detail or prerequisites archive, such as determination or prerequisites record. It may be a testing in which the program beneath test is treated, as a black box. You cannot “see” into it. The test gives inputs and reacts to yields without considering how the computer program works.

Unit Testing:

Unit testing is ordinarily conducted as portion of a combined code and unit test stage of the program lifecycle, in spite of the fact that it isn't exceptional for coding and unit testing to be conducted as two particular stages. Test procedure and approachField testing will be performed physically and useful tests will be composed in detail. Test destinations all field sections must work legitimately.

Integration Testing

Program integration testing is the incremental integration testing of two or more coordinates computer program components on a single stage to deliver disappointments caused by interface absconds.

The assignment of the integration test is to check that components or program applications, e.g. components in a computer code or – one step up – computer program applications at the company level connected without blunder.

Test Comes about: All the test cases specified over passed effectively. No abandons experienced.

Acceptance Testing

Client Acknowledgment Testing could be a basic stage of any venture and requires noteworthy interest by the conclusion client. It too guarantees that the framework meets the useful necessities.

Test Comes about:

All the test cases specified over passed effectively. No surrenders experienced.

CHAPTER-6

CONCLUSION & FUTURE ENHANCEMENT

6.1 CONSLUSION

Security and privacy are two fundamental concerns to set up a secure confirmation system in shrewd therapeutic framework. The paper is the development of an ECC-based appropriate system for shrewd restorative framework in cloud environment. In this paper, we have examined six diverse stages such as enlistment stage, healthcare center transfer stage, understanding information transfer stage, treatment stage, check up stage. The paper has appeared the security examination of the displayed system. we have illustrated that the proposed system oversees way better security and protection highlights and attributes compared to related systems within the comparable environment.

6.2 FUTURE ENHANCEMENT

Long-term upgrade of the project is

• Extend the demonstrate to bolster numerous dialects, empowering assumption investigation on motion picture surveys composed in several dialects by utilizing multilingual embeddings or language-specific models.

• Investigate the utilize of transformer-based models like BERT or GPT for opinion investigation, which have appeared prevalent execution on numerous NLP assignments compared to conventional RNN-LSTM models.

REFERENCES

[1] Maas, A.L., Daly, R.E., Pham, P.T., Huang, D., Ng, A.Y., Potts, C. (2011). "Learning Word Vectors for Sentiment Analysis." In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT). https://www.aclweb.org/anthology/P11-1015/

[2] Tang, D., Qin, B., Liu, T. (2015). "Learning Sentiment-Specific Word Embedding for Twitter Sentiment Classification." In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL). https://www.aclweb.org/anthology/P15-1167/

[3] Kim, Y. (2014). "Convolutional Neural Networks for Sentence Classification." In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). https://www.aclweb.org/anthology/D14-1181/

[4] Sundermeyer, M., Schlüter, R., Ney, H. (2012). "LSTM Neural Networks for Language Modeling." In Thirteenth Annual Conference of the International Speech Communication Association (INTERSPEECH). https://www.iscaspeech.org/archive/interspeech_2012/i12_0194.html

[5] Karpathy, A., Johnson, J., Fei-Fei, L. (2015). "Visualizing and Understanding Recurrent Networks." In Proceedings of the 31st International Conference on Machine Learning (ICML). https://arxiv.org/abs/1506.02078

[6] Tang, D., Qin, B., Feng, X., Liu, T. (2016). "Effective LSTMs for Target-Dependent Sentiment Classification." In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL). https://www.aclweb.org/anthology/P16-1149/

[7] Hochreiter, S., Schmidhuber, J. (1997). "Long Short-Term Memory." Neural Computation. https://www.mitpressjournals.org/doi/abs/10.1162/neco.1997.9.8.1735

[8] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y. (2014). "Learning Phrase Representations using RNN Encoder-Decoder for Statistical

Machine Translation." In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). https://www.aclweb.org/anthology/D14-1179/

Here are two additional references for sentiment analysis on movie reviews using RNNs with LSTMs:

[9] Zhang, X., Zhao, J., LeCun, Y. (2015)"Character-level Convolutional Networks for Text Classification." In Proceedings of the 28th International Conference on Neural Information Processing Systems (NIPS). https://papers.nips.cc/paper/5782-character-level-convolutional-networks-for-text-classification

[10] Lai, S., Xu, L., Liu, K., Zhao, J. (2015) "Recurrent Convolutional Neural Networks for Text Classification." In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI). https://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9745