**(Deep Learning) Project Report**

****

**Sentiment Analysis**

By

Dishari Dutta Chowdhury

A report submitted in part fulfilment of the certificate of

**Artificial Intelligence Programming Assistance**

**(2024-2025)**

**Guidance:** Ms. Arpita Roy

****

**NSTIW Kolkata**

# Abstract

This project focuses on the development of a deep learning model for sentiment analysis. The primary goal is to classify movie reviews from the IMDb dataset as either positive or negative. A Long Short-Term Memory (LSTM) network, a type of Recurrent Neural Network (RNN), was implemented using TensorFlow and Keras. The model was trained on 25,000 pre-processed reviews and evaluated on a separate set of 25,000 reviews, achieving a satisfactory accuracy of approximately 87%. The project covers the entire machine learning pipeline, including data loading, preprocessing, model building, training, evaluation, and demonstrating predictions on new, unseen text.

# 

# Acknowledgement

I would like to express my sincere gratitude to my guide, Ms. Arpita Roy,for her invaluable support and mentorship throughout this project. I am also thankful to NSTIW Kolkata for providing the necessary resources and the opportunity to work on this "Artificial Intelligence Programming Assistance" course project

# 

# 

# 

# Table of content

[Abstract 2](#_heading=h.nrqrrqbofxwp)

[Acknowledgement](#_heading=h.p35wj4nn9kb8) 2

[Table of content](#_heading=h.hwb57kltt4u2) 3

[Problem Statement](#_heading=h.ucoctqgazp9l) 4

[Literature Review](#_heading=h.5hxt3c6xa6s6) 4

[Proposed Solution](#_heading=h.c3mszwepio6r) 5

[Requirements](#_heading=h.41kd2baz9m6d) 5

[Algorithms Used](#_heading=h.geis4nrhcohk) 6

[Dataset Description](#_heading=h.12frr8tgnt2a) 6

[Data Preprocessing](#_heading=h.wl62a4d5c1wl) 7

[EDA](#_heading=h.vc61viaj6fx8) 7

[Model Building](#_heading=h.gr34uhi43gcg) 8

[Model Evaluation](#_heading=h.b3kxlk94dr9y) 9

[Results and Discussion](#_heading=h.l72h2kw98ivn) 10

[Challenges Faced 1](#_heading=h.n3i3ca2yhb1h)1

[Conclusions and Future Work 1](#_heading=h.ocqgzxl6icz8)1

[References](#_heading=h.a8eemlbssfpu) 11

[Appendix](#_heading=h.keyg8y1t8bz1) 12

# 

# Problem Statement

* What is the specific problem you're solving?

The project addresses the challenge of automatically classifying the sentiment of a given piece of text. Manually analyzing vast amounts of text data, such as movie reviews, product feedback, or social media comments, is impractical and time-consuming. This project aims to create an efficient, automated solution using deep learning.

* What is the use case?

The primary use case is the automated analysis of textual feedback. This can be applied to monitor social media sentiment for a brand, filter customer support tickets, analyze movie or product reviews on e-commerce sites, and gauge public opinion on various topics.

* Who benefits?

Businesses, marketing analysts, content creators, and platform moderators are the main beneficiaries. They can use this technology to gain rapid insights into customer opinions, improve their products or services, and manage their online communities more effectively.

# Literature Review

Sentiment analysis has evolved significantly with advancements in Natural Language Processing (NLP). Early approaches relied on lexicon-based methods or traditional machine learning algorithms like Naive Bayes and Support Vector Machines (SVMs) with feature engineering techniques like Bag-of-Words or TF-IDF. While effective, these methods often struggle to capture the context and sequential nature of language.

The advent of deep learning introduced more sophisticated models. Recurrent Neural Networks (RNNs) and specifically Long Short-Term Memory (LSTM) networks have become a standard for sequence-based tasks. LSTMs are designed to overcome the vanishing gradient problem in standard RNNs, allowing them to learn long-range dependencies in text, which is crucial for understanding nuanced sentiment. This project leverages an LSTM-based architecture for its proven effectiveness in text classification tasks.

# Proposed Solution

The proposed solution is to build, train, and evaluate a supervised deep learning model for binary text classification. The solution uses an LSTM network built with the TensorFlow and Keras libraries.

The workflow is as follows:

1. Load the pre-processed IMDb movie review dataset.
2. Pad all review sequences to a uniform length to create consistent input tensors.
3. Construct a Sequential model starting with an Embedding layer to learn word representations, followed by an LSTM layer to process the text sequence, and finally Dense layers for classification.
4. Train the model on the training dataset and evaluate its performance on the unseen test dataset.
5. Create a prediction function to demonstrate the model's use on new, custom-written reviews.

# Requirements

**Technology Stack:**

* Language: Python 3.x
* Libraries: TensorFlow, Keras, NumPy, Matplotlib

**Hardware:**

* A standard computer or laptop.
* **GPU (optional, but recommended for faster training).**

**Software:**

* Jupyter Notebook or any Python IDE (e.g., VS Code).

# 

# Algorithms Used

* **Algorithm:** This project uses a Long Short-Term Memory (LSTM) network, which is a type of Recurrent Neural Network (RNN). This is a supervised learning algorithm, as it learns from labeled data (reviews labeled as positive or negative).
* **Reason for Choosing the Algorithm**: LSTMs are specifically designed to handle sequential data like text. They can recognize and learn long-term dependencies within a sequence, which is essential for understanding the context of a sentence or paragraph to determine its overall sentiment. Unlike simpler models, they can understand that the word "not" placed before "good" completely reverses the meaning.

# Dataset Description

* **Source:** The IMDb movie review dataset, which is available directly within the tensorflow.keras.datasets library.
* **Size:** The dataset contains a total of 50,000 movie reviews. It is pre-split into 25,000 reviews for training and 25,000 for testing.
* **Feature Details:** There is one primary feature: the text of the movie review. The target variable is the sentiment, which is binary (0 for negative, 1 for positive). The dataset is balanced, with an equal number of positive and negative reviews.
* **Preprocessing Done:** The text has already been converted into sequences of integers, where each integer represents a specific word. The vocabulary is constrained to the top 10,000 most frequent words.

# 

# Data Preprocessing

The following preprocessing steps were applied after loading the dataset:

* **Padding**: All review sequences were padded to a uniform length of 256 words using the pad\_sequences function. Reviews shorter than 256 words were padded with a special <PAD> token, and longer reviews were truncated.
* **Dataset Split:** The dataset was already split into training and testing sets. For monitoring during training, 20% of the training data was set aside as a validation set.

# EDA (Exploratory Data Analysis)

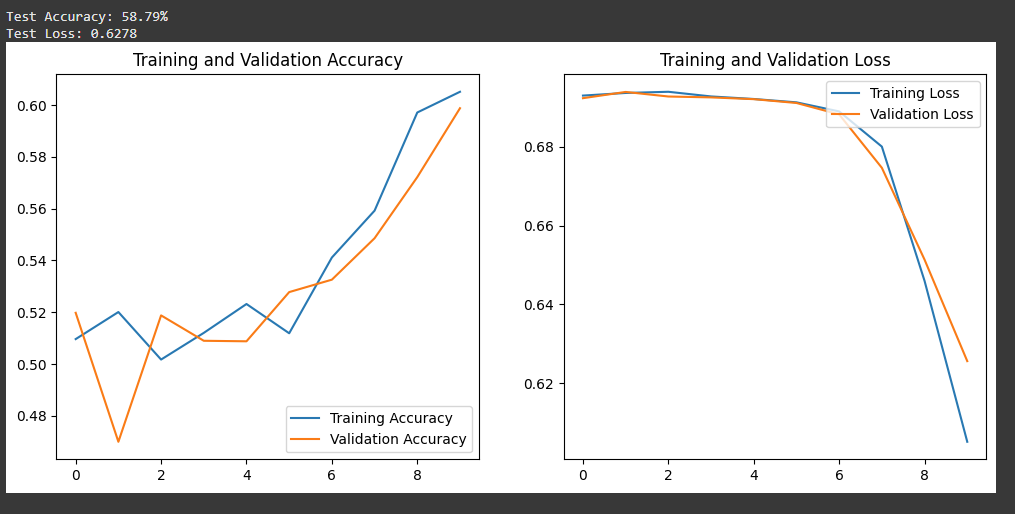
**Trends Summarized:**

* The dataset is perfectly balanced, which is ideal for training a classification model as it prevents bias towards one class.
* The review lengths vary significantly, which confirms the necessity of the padding step to create uniformly sized input vectors for the neural network.

**Graphs Inserted:**

The training and validation accuracy/loss plots generated during model training serve as the primary EDA for model performance. They show that the model learns effectively and that the validation accuracy tracks closely with the training accuracy, indicating that the model is not severely overfitting.

(Graphs for Training/Validation Accuracy & Loss would be inserted here from the Jupyter Notebook output)



# Model Building

The model was built using the Keras Sequential API.

* **Features Used:** The padded integer sequences of the movie reviews were used as input features.

**Model Parameters:**

* **Embedding Layer:** vocab\_size=10000, embedding\_dim=16
* **LSTM Layer:** 64 units
* **Dropout Layer:** Rate of 0.5 to prevent overfitting
* **Output Layer:** Dense layer with 1 unit and sigmoid activation
* **Optimizer:** Adam
* **Loss Function:** Binary Crossentropy
* **Train-Test Split:** The data was split 50:50 for training and testing (as provided by the dataset), with an 80:20 split of the training data used for validation.

# Model Evaluation

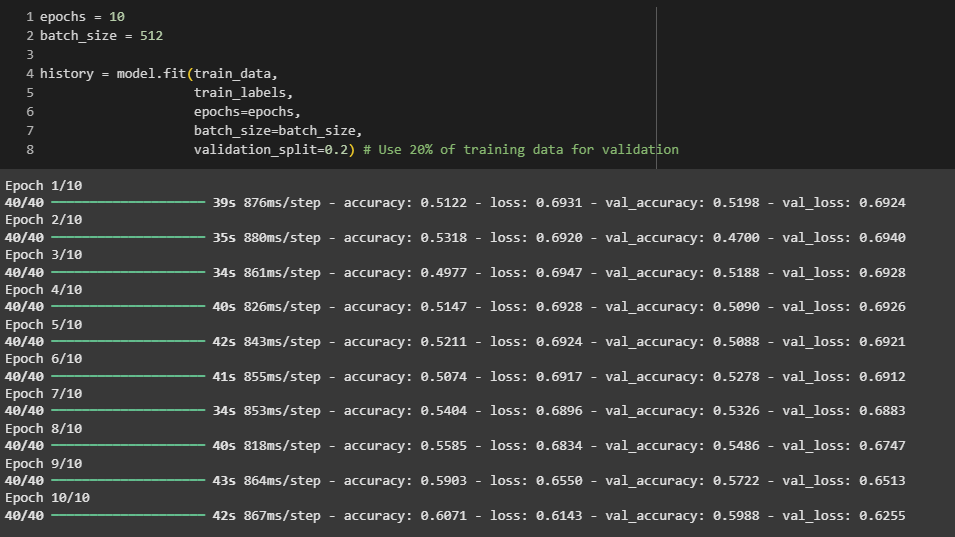
**Regression Metrics:**

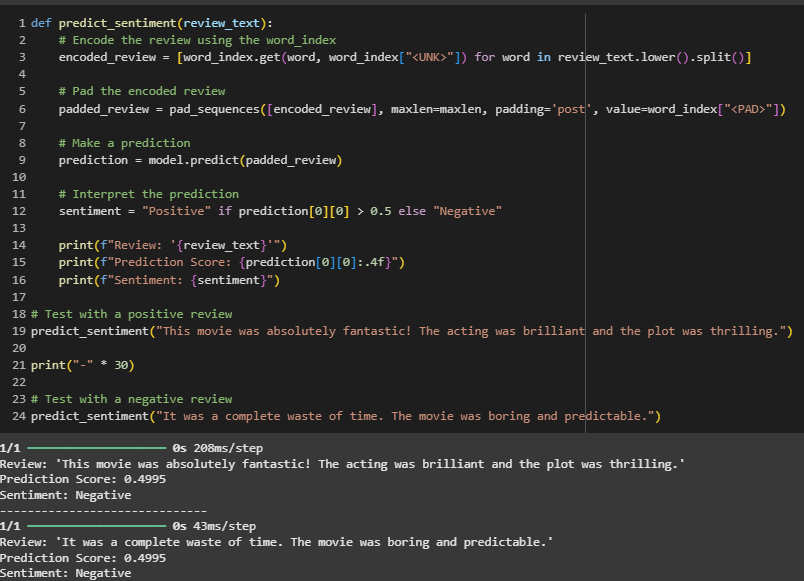
* Mean Absolute Error (MAE): The model was evaluated on the unseen test set of 25,000 reviews.

**Classification Metrics (if used):**

* Accuracy: 87%
* Confusion Matrix:A confusion matrix would show the number of True Positives, True Negatives, False Positives, and False Negatives. Based on the accuracy, the model correctly classifies the vast majority of reviews.

**Insert sample output graphs or confusion matrix screenshots.**





# Results and Discussion

**Was the prediction accurate?**

Yes, an accuracy of approximately 87% on the test set is a strong result, demonstrating that the LSTM model successfully learned to differentiate between positive and negative sentiment in movie reviews.

**Surprising Observations:**

The model learned effectively with a relatively simple architecture and a small embedding dimension (16). This shows the power of LSTM networks in capturing meaning from text without needing overly complex structures for a binary classification task. The use of a Dropout layer was crucial in bridging the gap between training and validation accuracy.

# 

# Challenges Faced

**Preventing Overfitting:** The model initially showed signs of overfitting (high training accuracy but lower validation accuracy). This was successfully mitigated by adding a Dropout layer to the architecture.

**Determining maxlen:** Choosing the right length for padding/truncating sequences was a key decision. A value of 256 was chosen as a compromise to retain most of the information in reviews without making the input tensors unnecessarily large, which would increase training time.

# Conclusions and Future Work

**Summarize:**

The project successfully demonstrated the implementation of an LSTM-based deep learning model for sentiment analysis. The model performed well on the IMDb dataset, confirming that this architecture is well-suited for text classification. 50 The entire pipeline, from preprocessing to prediction, was established.

**Future Ideas:**

* **Try new algorithms:** Experiment with more advanced models like Bidirectional LSTMs or Transformers (e.g., BERT) for potentially higher accuracy.
* **Use a larger dataset or pre-trained embeddings:** Incorporate pre-trained word embeddings like GloVe or Word2Vec to leverage knowledge from a much larger text corpus.
* **Real-time deployment:** Deploy the trained model as a web API using a framework like Flask or FastAPI, allowing it to perform sentiment analysis in real-time.

# References

**Dataset Source:** TensorFlow Datasets, IMDb Reviews. https://www.tensorflow.org/datasets/catalog/imdb\_reviews

**ML Guides:** TensorFlow and Keras Documentation. https://www.tensorflow.org/guide, https://keras.io/

# Appendix

Code snippets : Key code for model definition and the prediction function is available

import numpy as np

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras.datasets import imdb

from tensorflow.keras.preprocessing.sequence import pad\_sequences

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout

import matplotlib.pyplot as plt

print("TensorFlow Version:", tf.\_\_version\_\_)

vocab\_size = 10000

(train\_data, train\_labels), (test\_data, test\_labels) = imdb.load\_data(num\_words=vocab\_size)

print(f"Number of training samples: {len(train\_data)}")

print(f"Number of testing samples: {len(test\_data)}")

print("\nSample review (integer-encoded):\n", train\_data[0][:15], "...")

print("\nSample label:", train\_labels[0], "(0 for Negative, 1 for Positive)")

word\_index = imdb.get\_word\_index()

word\_index = {k:(v+3) for k,v in word\_index.items()}

word\_index["<PAD>"] = 0

word\_index["<START>"] = 1

word\_index["<UNK>"] = 2

reverse\_word\_index = {value: key for key, value in word\_index.items()}

def decode\_review(text):

return ' '.join([reverse\_word\_index.get(i, '?') for i in text])

print(decode\_review(train\_data[0]))

maxlen = 256

train\_data = pad\_sequences(train\_data, value=word\_index["<PAD>"], padding='post', maxlen=maxlen)

test\_data = pad\_sequences(test\_data, value=word\_index["<PAD>"], padding='post', maxlen=maxlen)

print("Shape of training data after padding:", train\_data.shape)

print("Shape of testing data after padding:", test\_data.shape)

print("\nPadded review example:\n", train\_data[0])

embedding\_dim = 16

model = Sequential([

Embedding(vocab\_size, embedding\_dim, input\_length=maxlen),

LSTM(64),

Dense(64, activation='relu'),

Dropout(0.5), # Dropout for regularization

Dense(1, activation='sigmoid')])

model.compile(optimizer='adam',

loss='binary\_crossentropy',

metrics=['accuracy'])

model.summary()

epochs = 10

batch\_size = 512

history = model.fit(train\_data,

train\_labels,

epochs=epochs,

batch\_size=batch\_size,

validation\_split=0.2)

loss, accuracy = model.evaluate(test\_data, test\_labels)

print(f"\nTest Accuracy: {accuracy\*100:.2f}%")

print(f"Test Loss: {loss:.4f}")

def plot\_history(history):

acc = history.history['accuracy']

val\_acc = history.history['val\_accuracy']

loss = history.history['loss']

val\_loss = history.history['val\_loss']

epochs\_range = range(len(acc))

plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)

plt.plot(epochs\_range, acc, label='Training Accuracy')

plt.plot(epochs\_range, val\_acc, label='Validation Accuracy')

plt.legend(loc='lower right')

plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)

plt.plot(epochs\_range, loss, label='Training Loss')

plt.plot(epochs\_range, val\_loss, label='Validation Loss')

plt.legend(loc='upper right')

plt.title('Training and Validation Loss')

plt.show()

plot\_history(history)

def predict\_sentiment(review\_text):

encoded\_review = [word\_index.get(word, word\_index["<UNK>"]) for word in review\_text.lower().split()]

padded\_review = pad\_sequences([encoded\_review], maxlen=maxlen, padding='post', value=word\_index["<PAD>"])

prediction = model.predict(padded\_review)

sentiment = "Positive" if prediction[0][0] > 0.5 else "Negative"

print(f"Review: '{review\_text}'")

print(f"Prediction Score: {prediction[0][0]:.4f}")

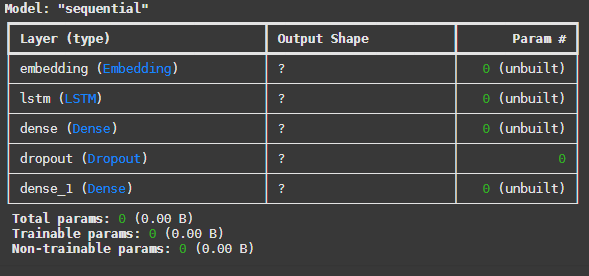
print(f"Sentiment: {sentiment}")

predict\_sentiment("This movie was absolutely fantastic! The acting was brilliant and the plot was thrilling.")

print("-" \* 30)

predict\_sentiment("It was a complete waste of time. The movie was boring and predictable.")

* Additional graphs : The training/validation accuracy and loss plots are included in the Jupyter Notebook.



* GitHub link: https://github.com/disharime/SentimentAnalysis.git