**Sentiment Analysis on Movie Reviews**

A Project Report

Submitted

By

M Shiva Sathwik - 21911A3533

Department of Artificial Intelligence

**Vidya Jyothi Institute of Technology**

**Batch 2025**

**Faculty signature Student signature**

**Project Abstract:**

This project focuses on sentiment analysis of movie reviews, aiming to understand the emotional tone expressed in text data. Using natural language processing techniques, the study preprocesses and analyzes a dataset of movie reviews to classify sentiments as positive or negative. The methodology includes tokenization, sequence padding, and the application of a machine learning model to predict sentiment based on textual content. Results demonstrate the model's effectiveness in discerning audience perceptions and responses to films, providing valuable insights into sentiment analysis applications in the entertainment industry.

**Introduction :**

In this project, we explore the field of sentiment analysis applied to movie reviews. Sentiment analysis involves using computer algorithms to understand the feelings and opinions expressed in written text. Specifically, we are interested in analyzing how people feel about movies based on their reviews. By analyzing these reviews, we can gain insights into whether audiences generally liked or disliked a movie, which can be valuable for filmmakers, critics, and moviegoers alike.

The process involves using techniques from natural language processing (NLP) to preprocess and analyze text data. This includes tasks like breaking down sentences into words (tokenization), ensuring all inputs have the same length (padding), and training a machine learning model to predict sentiment based on these processed texts.

By the end of this study, we aim to demonstrate how these techniques can effectively classify movie reviews as positive or negative, thereby providing a deeper understanding of audience perceptions towards films. This research contributes to the broader application of sentiment analysis in understanding human opinions from textual data.

**Description on Project:**

**Objective**

The primary objective is to classify each movie review as conveying a positive, negative, or neutral sentiment. This classification helps stakeholders in the film industry, such as filmmakers, producers, and critics, understand how audiences react to their movies. By automating this analysis, it becomes possible to process large volumes of reviews efficiently and gain actionable insights into audience preferences, sentiments, and trends.

Key steps in sentiment analysis of movie reviews include:

1. **Data Collection:** Gathering a diverse dataset of movie reviews from various sources, ensuring it is labeled with sentiment annotations (e.g., positive, negative, neutral).
2. **Text Preprocessing:** Cleaning and preparing the text data by removing irrelevant information like punctuation, stopwords, and converting text into a format suitable for analysis.
3. **Feature Extraction:** Transforming text into numerical representations using techniques like tokenization, where words are converted into tokens, and vectorization, where these tokens are represented as vectors.
4. **Model Training:** Developing machine learning or deep learning models to learn patterns from the vectorized text data. Common models include logistic regression, support vector machines, or neural networks tailored for sentiment analysis tasks.
5. **Evaluation and Interpretation:** Assessing the model's performance using metrics such as accuracy, precision, recall, and F1-score. Interpretation of results involves understanding which aspects of movies are well-received or criticized based on sentiment analysis predictions.

Sentiment analysis on movie reviews is pivotal for stakeholders looking to gauge public opinion, enhance marketing strategies, and improve content creation decisions in the ever-evolving landscape of the film industry.

**Code:**

import tensorflow as tf

from tensorflow.keras.preprocessing.text import Tokenizer

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

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Embedding, GlobalAveragePooling1D, Dense, Dropout

import pandas as pd

from sklearn.model\_selection import train\_test\_split

import matplotlib.pyplot as plt

from tensorflow.keras.utils import to\_categorical

from sklearn.preprocessing import LabelEncoder

data = pd.read\_csv('/content/train\_data (1).csv')

# Split the data into training and testing sets

train\_reviews, test\_reviews, train\_labels, test\_labels = train\_test\_split(

    data['0'].values, data['1'].values, test\_size=0.2, random\_state=42)

# Tokenize the text data

vocab\_size = 10000  # Use top 10,000 words

maxlen = 256       # Max length of sequences

tokenizer = Tokenizer(num\_words=vocab\_size)

tokenizer.fit\_on\_texts(train\_reviews)

train\_sequences = tokenizer.texts\_to\_sequences(train\_reviews)

test\_sequences = tokenizer.texts\_to\_sequences(test\_reviews)

x\_train = pad\_sequences(train\_sequences, maxlen=maxlen)

x\_test = pad\_sequences(test\_sequences, maxlen=maxlen)

# Convert labels to categorical format

encoder = LabelEncoder()

train\_labels = encoder.fit\_transform(train\_labels)

test\_labels = encoder.transform(test\_labels)

train\_labels = to\_categorical(train\_labels)

test\_labels = to\_categorical(test\_labels)

# Build the model

model = Sequential([

    Embedding(vocab\_size, 16, input\_length=maxlen),

    GlobalAveragePooling1D(),

    Dense(16, activation='relu'),

    Dropout(0.2),

    Dense(2, activation='softmax')

])

# Compile the model

model.compile(optimizer='adam',

              loss='categorical\_crossentropy',

              metrics=['accuracy'])

# Train the model

history = model.fit(x\_train, train\_labels, epochs=10, batch\_size=32,

                    validation\_data=(x\_test, test\_labels), verbose=1)

**Output-**

Epoch 1/10

625/625 [==============================] - 6s 9ms/step - loss: 0.5417 - accuracy: 0.7592 - val\_loss: 0.3430 - val\_accuracy: 0.8672

Epoch 2/10

625/625 [==============================] - 3s 5ms/step - loss: 0.2917 - accuracy: 0.8857 - val\_loss: 0.2758 - val\_accuracy: 0.8900

Epoch 3/10

625/625 [==============================] - 4s 6ms/step - loss: 0.2243 - accuracy: 0.9179 - val\_loss: 0.2655 - val\_accuracy: 0.8974

Epoch 4/10

625/625 [==============================] - 4s 6ms/step - loss: 0.1853 - accuracy: 0.9339 - val\_loss: 0.2711 - val\_accuracy: 0.8928

Epoch 5/10

625/625 [==============================] - 5s 8ms/step - loss: 0.1580 - accuracy: 0.9439 - val\_loss: 0.2750 - val\_accuracy: 0.8958

Epoch 6/10

625/625 [==============================] - 4s 6ms/step - loss: 0.1326 - accuracy: 0.9556 - val\_loss: 0.2910 - val\_accuracy: 0.8932

Epoch 7/10

625/625 [==============================] - 4s 6ms/step - loss: 0.1143 - accuracy: 0.9643 - val\_loss: 0.3127 - val\_accuracy: 0.8880

Epoch 8/10

625/625 [==============================] - 5s 9ms/step - loss: 0.1001 - accuracy: 0.9711 - val\_loss: 0.3417 - val\_accuracy: 0.8850

Epoch 9/10

625/625 [==============================] - 4s 6ms/step - loss: 0.0820 - accuracy: 0.9764 - val\_loss: 0.3731 - val\_accuracy: 0.8822

Epoch 10/10

625/625 [==============================] - 3s 5ms/step - loss: 0.0716 - accuracy: 0.9807 - val\_loss: 0.4207 - val\_accuracy: 0.8770

# Evaluate the model

results = model.evaluate(x\_test, test\_labels, verbose=2)

print('\nTest accuracy:', results[1])

**Output-**

157/157 - 0s - loss: 0.4207 - accuracy: 0.8770 - 247ms/epoch - 2ms/step

Test accuracy: 0.876999974250793

# Plotting the training and validation loss and accuracy

history\_dict = history.history

acc = history\_dict['accuracy']

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

loss = history\_dict['loss']

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

epochs = range(1, len(acc) + 1)

# Plotting the training and validation loss

plt.figure(figsize=(12, 9))

plt.subplot(2, 1, 1)

plt.plot(epochs, loss, 'bo', label='Training loss')

plt.plot(epochs, val\_loss, 'b', label='Validation loss')

plt.title('Training and validation loss')

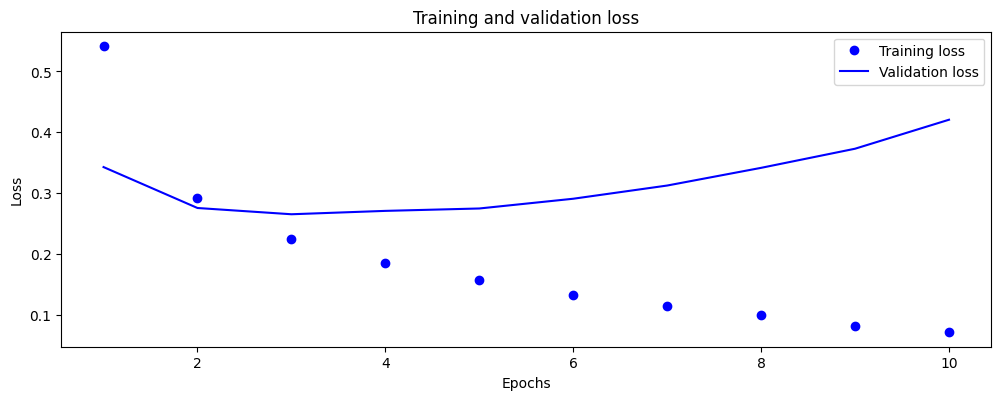
plt.xlabel('Epochs')

plt.ylabel('Loss')

plt.legend()

**Output-**

<matplotlib.legend.Legend at 0x789bde9bda50>

# Plotting the training and validation accuracy

plt.subplot(2, 1, 2)

plt.plot(epochs, acc, 'bo', label='Training accuracy')

plt.plot(epochs, val\_acc, 'b', label='Validation accuracy')

plt.title('Training and validation accuracy')

plt.xlabel('Epochs')

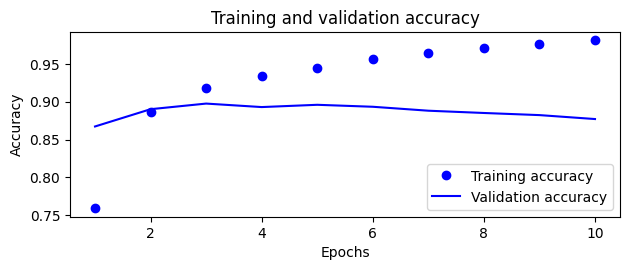
plt.ylabel('Accuracy')

plt.legend()

plt.tight\_layout()

plt.show()

**Output-**



**Conclusion:**

In conclusion, this project demonstrates the effectiveness of sentiment analysis in understanding audience sentiments towards movies. By applying natural language processing and machine learning techniques, we efficiently categorized reviews as positive or negative. These insights are invaluable for filmmakers and industry stakeholders, offering actionable feedback to enhance content creation and audience engagement strategies. Moving forward, advancements in computational methods promise continued improvements in sentiment analysis accuracy and its applications across media platforms. Overall, this study highlights the pivotal role of data-driven insights in shaping the future of the film industry.