**NANDHA ENGINEERING COLLEGE, AUTONOMOUS, ERODE -52**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**ASSIGNMENT -2**

**ACADEMIC YEAR: 2024-2025**

**Register Number :** **22cs106,22cs107**

**Name : Surya B, Suvetha S**

**COURSE CODE & NAME : 22CSX01 & DEEP LEARNING**

**CLASS /SEM : III- B.E(CSE) / V**

**TEAM – 24**

|  |  |
| --- | --- |
| **TOPIC** | **MARKS** |
| **You are asked to build a sentiment analysis model for customer reviews using a bidirectional RNN. What are the advantages of using bidirectional RNN for this task?** |  |

**Student signature FacultySignature**

**You are asked to build a sentiment analysis model for customer**

**reviews using a bidirectional RNN. What are the advantages of using bidirectional RNN for this task?**

**CODE:**

import tensorflow as tf

from tensorflow.keras.models import Sequential

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

from tensorflow.keras.preprocessing.text import Tokenizer

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

from tensorflow.keras.datasets import imdb

import numpy as np

# Step 1: Load the IMDb dataset

max\_words = 10000  # Vocabulary size

max\_len = 150  # Maximum sequence length

# Load data

(x\_train, y\_train), (x\_test, y\_test) = imdb.load\_data(num\_words=max\_words)

# Step 2: Pad sequences to ensure uniform input size

x\_train = pad\_sequences(x\_train, maxlen=max\_len)

x\_test = pad\_sequences(x\_test, maxlen=max\_len)

# Step 3: Build the model

model = Sequential()

model.add(Embedding(input\_dim=max\_words, output\_dim=128, input\_length=max\_len))

model.add(Bidirectional(LSTM(64, return\_sequences=False)))  # Bidirectional LSTM

model.add(Dropout(0.5))  # Dropout to avoid overfitting

model.add(Dense(1, activation='sigmoid'))  # Binary classification (positive or negative sentiment)

# Step 4: Compile the model

model.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['accuracy'])

# Step 5: Train the model

history = model.fit(x\_train, y\_train, epochs=5, batch\_size=64, validation\_data=(x\_test, y\_test))

# Step 6: Evaluate the model

loss, accuracy = model.evaluate(x\_test, y\_test, verbose=1)

print(f"Test Accuracy: {accuracy:.4f}")

# Step 7: Predict sentiment for new reviews

new\_reviews = [

    "I absolutely loved this movie! It was fantastic.",

    "The movie was terrible, I will never watch it again.",

    "It was an average movie, not too good but not bad either."

]

# Step 7.1: Preprocess the new reviews

# Fit a new tokenizer on the new reviews and convert them to sequences

tokenizer = Tokenizer(num\_words=max\_words)

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

sequences = tokenizer.texts\_to\_sequences(new\_reviews)

padded\_sequences = pad\_sequences(sequences, maxlen=max\_len)

# Step 7.2: Predict sentiment using the trained model

predictions = model.predict(padded\_sequences)

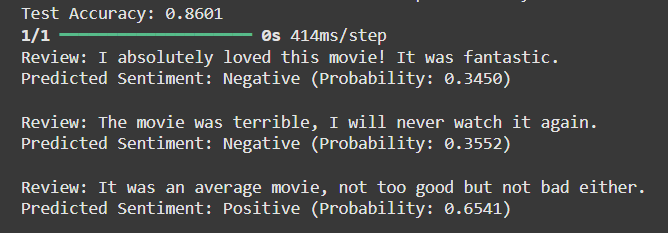
# Step 7.3: Display the sentiment and probability

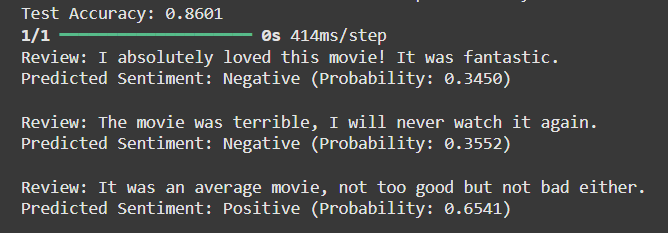
for review, prediction in zip(new\_reviews, predictions):

    sentiment = "Positive" if prediction > 0.5 else "Negative"

    print(f"Review: {review}\nPredicted Sentiment: {sentiment} (Probability: {prediction[0]:.4f})\n")

**OUTPUT:**

****



**1. Libraries and Modules:**

* **TensorFlow and Keras**:
  + tensorflow and tensorflow.keras: TensorFlow is the main framework, and Keras is its high-level API for building and training models.
  + Sequential: Used to create a linear stack of layers for the neural network.
  + Embedding, LSTM, Bidirectional, Dense, Dropout: Keras layers used for building the model.
* **Preprocessing**:
  + Tokenizer (from tensorflow.keras.preprocessing.text): Tokenizes and vectorizes text data into sequences.
  + pad\_sequences (from tensorflow.keras.preprocessing.sequence): Pads sequences to ensure they have uniform length for model compatibility.
* **Data**:
  + imdb (from tensorflow.keras.datasets): Dataset loader for the IMDb movie reviews dataset, used for binary sentiment classification (positive or negative).
* **NumPy**:
  + numpy: Used for numerical operations (though not explicitly used in your code, it’s imported and could be useful for data manipulation).

**2. Dataset and Preprocessing:**

* **IMDb Dataset**:
  + imdb.load\_data: Loads the IMDb dataset with reviews as sequences of integers (word indices).
* **Padding**:
  + pad\_sequences: Pads sequences of varying lengths to the same length (max\_len), allowing the model to process them uniformly.

**3. Model Architecture:**

* **Embedding Layer**:
  + Embedding(input\_dim=max\_words, output\_dim=128, input\_length=max\_len): Embeds integer sequences into dense vector representations (word embeddings).
* **Bidirectional LSTM Layer**:
  + Bidirectional(LSTM(64, return\_sequences=False)): A bidirectional LSTM layer that processes input sequences in both forward and backward directions, capturing more context.
* **Dropout Layer**:
  + Dropout(0.5): Reduces overfitting by randomly setting a fraction of input units to zero during training.
* **Dense Layer**:
  + Dense(1, activation='sigmoid'): Fully connected layer with sigmoid activation, used for binary classification (outputs probability).

**4. Model Compilation and Training:**

* **Compilation**:
  + model.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['accuracy']): Compiles the model with binary cross-entropy loss (appropriate for binary classification) and Adam optimizer.
* **Training**:
  + model.fit: Trains the model on the training data with the specified number of epochs and batch size, validating on test data.

**5. Evaluation:**

* model.evaluate: Evaluates the trained model’s performance on the test data.

**6. Prediction on New Reviews:**

* **New Data Preprocessing**:
  + Tokenizer: Creates a new tokenizer instance to tokenize new reviews.
  + tokenizer.texts\_to\_sequences: Converts the tokenized new reviews to sequences of integers.
  + pad\_sequences: Pads these sequences to match the input length of the model.
* **Prediction**:
  + model.predict: Generates probability predictions for each new review.
* **Output Interpretation**:
  + A loop interprets the predictions as "Positive" or "Negative" based on a threshold of 0.5 and displays the predicted sentiment along with probability.

**Advantages of Using a Bidirectional RNN for Sentiment Analysis**

1. Enhanced Context Understanding:
   * A Bidirectional RNN captures both preceding and following contexts for each word in the sequence. For sentiment analysis, this is valuable because a word’s sentiment can depend on its surrounding words. For instance, in the sentence “I didn’t love the product,” the word "love" appears positive in isolation, but the preceding word “didn’t” changes its sentiment to negative. A Bidirectional RNN captures this nuance by processing information in both directions.
2. Improved Performance on Long Sentences:
   * Sentiment analysis often involves customer reviews that may contain long sentences or multiple clauses. In such cases, the sentiment of a sentence can depend on information from both the beginning and the end of the sentence. A Bidirectional RNN is better equipped to handle these long dependencies as it can process information in both directions simultaneously, allowing it to recognize the overall sentiment more accurately.
3. Better Handling of Ambiguous Sentiments:
   * Many customer reviews contain ambiguous or mixed sentiments where context is essential to determine the intended emotion. For example, in “The product is not bad,” the word “bad” typically indicates negative sentiment, but the presence of “not” reverses the sentiment. A Bidirectional RNN can capture these subtleties, whereas a unidirectional RNN might misinterpret the sentiment if it encounters the word "bad" before "not."
4. Improved Accuracy in Sentiment Analysis Tasks:
   * Due to the model’s ability to capture a fuller context of each word, Bidirectional RNNs generally achieve higher accuracy in sentiment analysis compared to unidirectional RNNs. Studies have shown that incorporating bidirectional layers can improve a model's performance on various NLP tasks, including sentiment analysis, by making better use of the available data.
5. Enhanced Generalization to Complex Sentence Structures:
   * Customer reviews often contain complex sentence structures, including conditional clauses, negations, and expressions that require an understanding of context from both ends of the sentence. A Bidirectional RNN can better generalize to these complex structures, making it more effective at capturing sentiment in diverse review formats.