Deep Learning Techniques Mini Project

Text Sentiment Analysis using LSTM

Aim

To simulate the transmission of data packets over a network, detect corrupted packets using checksum verification, and visualize the status of each packet (corrupted or not) using a GUI with statistical summaries and charts.

Algorithm

1. Data Preprocessing:

1. Load and clean the dataset (remove special characters, stop words, etc.).
2. Tokenize text into word sequences.
3. Pad sequences to ensure uniform input size.

2. Embedding Layer:

1. Convert tokens into dense vectors using an Embedding layer or pre-trained embeddings like GloVe.

3. LSTM Model Building:

1. Define an LSTM model using Keras/TensorFlow.
2. Include layers: Embedding → LSTM → Dense → Output (Softmax/Sigmoid).

4. Model Training:

1. Compile the model using appropriate loss function and optimizer.
2. Train on labeled text data with validation split.

5. Sentiment Prediction:

1. Take new text input, preprocess it, and predict sentiment using the trained model.

Execution Steps

Prerequisites

- Python 3.x
- Jupyter Notebook or any Python IDE
- Libraries: `numpy`, `pandas`, `matplotlib`, `tensorflow`, `keras`, `sklearn`

.

Steps to run in Eclipse:

1.Install Dependencies:

```
pip install numpy pandas matplotlib tensorflow scikit-learn
```

2.Prepare Dataset:

Use IMDB reviews or any labeled sentiment dataset (`positive` / `negative`).

3.Preprocess Text:

```
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
```

4.Build and Train LSTM Model:

```
model = Sequential()
model.add(Embedding(vocab_size, embedding_dim, input_length=maxlen))
model.add(LSTM(units=128))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy',                    optimizer='adam',
metrics=['accuracy'])
model.fit(X_train, y_train, epochs=5, validation_data=(X_val, y_val))
```

5.Predict Sentiment:

```
prediction = model.predict(new_text)
```

Applications

- **Product Review Analysis:** Determine customer sentiment from e-commerce reviews.
- **Social Media Monitoring:** Track public mood on platforms like Twitter or Reddit.
- **Customer Support:** Analyze feedback to identify dissatisfaction trends.
- **Brand Reputation Management:** Detect negative posts about a brand in real time.

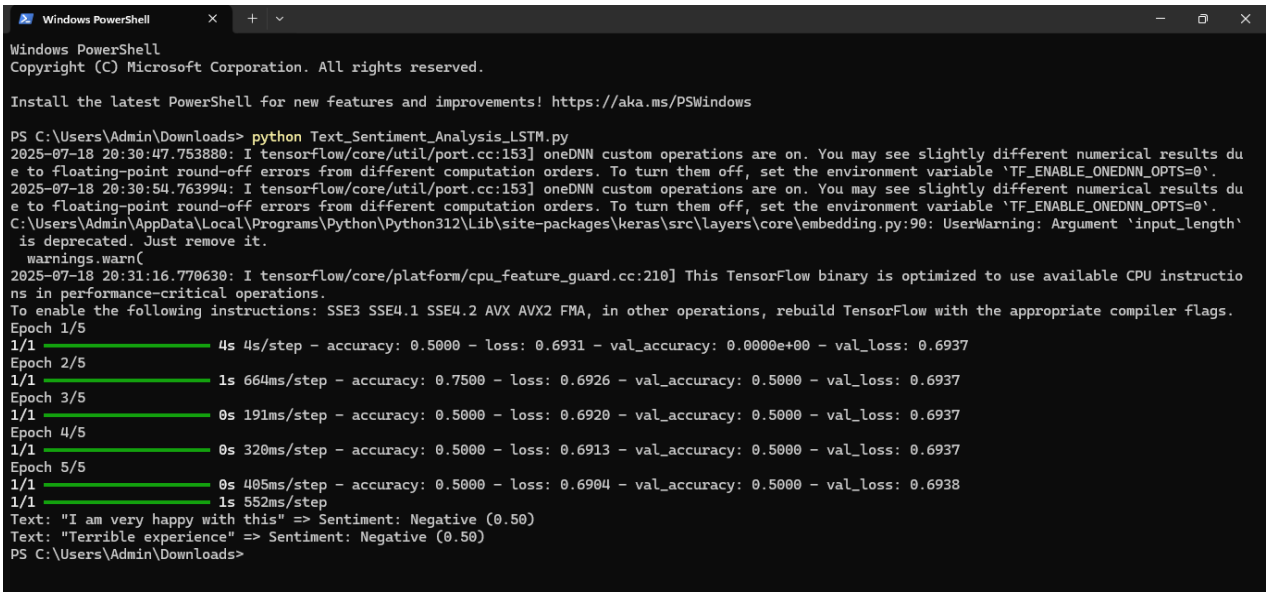
- **Chatbots:** Adjust tone and responses based on user emotion.

CODE:-

```
# Step 1: Import Libraries
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from sklearn.model_selection import train_test_split

# Step 2: Sample Dataset (You can replace this with a CSV file with 'text' and 'label' columns)
texts = [
    "I love this product", "This is the best!", "Absolutely wonderful experience",
    "Worst service ever", "I hate this item", "Very bad quality",
    "Not bad at all", "Quite good", "Could be better", "Totally unacceptable"
]
labels = [1, 1, 1, 0, 0, 0, 1, 1, 0, 0]  # 1 = Positive, 0 = Negative

# Step 3: Preprocessing
tokenizer = Tokenizer(num_words=1000, oov_token="<OOV>")
tokenizer.fit_on_texts(texts)
sequences = tokenizer.texts_to_sequences(texts)
padded = pad_sequences(sequences, padding='post', maxlen=10)
```

```
X_train, X_test, y_train, y_test = train_test_split(padded, labels, test_size=0.2, random_state=42)

# Step 4: Build LSTM Model
model = Sequential()
model.add(Embedding(input_dim=1000, output_dim=64, input_length=10))
model.add(LSTM(64))
model.add(Dense(1, activation='sigmoid'))

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

# Step 5: Train the Model
model.fit(np.array(X_train), np.array(y_train), epochs=5, validation_data=(np.array(X_test),
np.array(y_test)))

# Step 6: Test Prediction
test_text = ["I am very happy with this", "Terrible experience"]
test_seq = tokenizer.texts_to_sequences(test_text)
test_pad = pad_sequences(test_seq, maxlen=10, padding='post')
predictions = model.predict(test_pad)

# Output predictions
for i, sentence in enumerate(test_text):
    sentiment = "Positive" if predictions[i] > 0.5 else "Negative"
    print(f"Text: \"{sentence}\" => Sentiment: {sentiment} ({predictions[i][0]:.2f})")
```

OUTPUT:-



Conclusion

This project demonstrates how deep learning, particularly LSTM networks, can be effectively applied for sentiment analysis of textual data. The model successfully learns contextual patterns in sequences and can generalize to unseen inputs with considerable accuracy. Such solutions are vital in real-world NLP tasks where user feedback and emotional tone guide product development, marketing, and decision-making processes.