*Experiment No. 5 *

Aim: Perform Sentiment Analysis in the network graph using RNN.

```
import numpy as np
import pandas as pd
import networkx as nx
import tensorflow as tf
import matplotlib.pyplot as plt
import seaborn as sns
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, LSTM, Dense, SpatialDropout1D
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import confusion_matrix
```

```
# Sample dataset (Replace with actual dataset)
data = {"text": ["I love this product!", "This is the worst experience ever.", "Amazing service and friendly staff.", "I hate waiting in
        "sentiment": ["positive", "negative", "positive", "negative", "positive"]}
df = pd.DataFrame(data)
```

```
# Convert text data into a network graph
G = nx.Graph()
for text in df['text']:
    words = text.lower().split()
    for i in range(len(words) - 1):
        G.add_edge(words[i], words[i+1])
```

```
# Text Preprocessing
max_words = 5000  # Vocabulary size
max_len = 20       # Max length of each sentence

tokenizer = Tokenizer(num_words=max_words, oov_token="<OOV>")
tokenizer.fit_on_texts(df['text'])
X = tokenizer.texts_to_sequences(df['text'])
X = pad_sequences(X, maxlen=max_len, padding='post')
```

```
# Encode sentiment labels
encoder = LabelEncoder()
y = encoder.fit_transform(df['sentiment'])
y = tf.keras.utils.to_categorical(y, num_classes=2)
```

```
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Build RNN Model
model = Sequential([
    Embedding(input_dim=max_words, output_dim=128, input_length=max_len),
    SpatialDropout1D(0.2),
    LSTM(100, dropout=0.2, recurrent_dropout=0.2, return_sequences=False),
    Dense(2, activation='softmax')
])

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

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument `input_length` is deprecated. Just
  warnings.warn(
```

```
# Train Model
epochs = 5
batch_size = 4
model.fit(X_train, y_train, epochs=epochs, batch_size=batch_size, validation_data=(X_test, y_test), verbose=1)
```

```
Epoch 1/5
1/1 ━━━━━━━━━━━━━━━━━━━━ 6s 6s/step - accuracy: 0.2500 - loss: 0.7056 - val_accuracy: 0.0000e+00 - val_loss: 0.7353
Epoch 2/5
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 124ms/step - accuracy: 0.7500 - loss: 0.6765 - val_accuracy: 0.0000e+00 - val_loss: 0.7932
Epoch 3/5
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 140ms/step - accuracy: 0.7500 - loss: 0.6559 - val_accuracy: 0.0000e+00 - val_loss: 0.8575
Epoch 4/5
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 142ms/step - accuracy: 0.7500 - loss: 0.6217 - val_accuracy: 0.0000e+00 - val_loss: 0.9304
Epoch 5/5
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 135ms/step - accuracy: 0.7500 - loss: 0.6088 - val_accuracy: 0.0000e+00 - val_loss: 1.0112
<keras.src.callbacks.history.History at 0x79d993f32cd0>
```

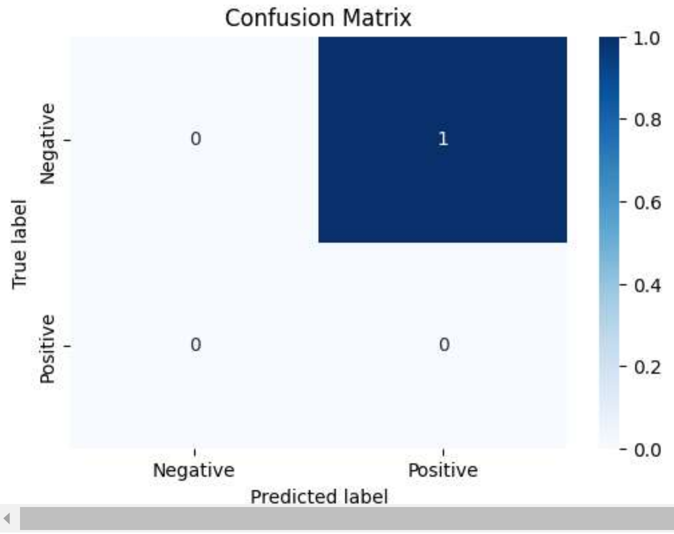
```
# Evaluate Model
loss, accuracy = model.evaluate(X_test, y_test, verbose=1)
print(f"Test Accuracy: {accuracy * 100:.2f}%")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - accuracy: 0.0000e+00 - loss: 1.0112
Test Accuracy: 0.00%
```

```
# Predict on test data
y_pred = model.predict(X_test)
y_pred_classes = np.argmax(y_pred, axis=1)
y_true = np.argmax(y_test, axis=1)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
```

```
# Confusion Matrix
cm = confusion_matrix(y_true, y_pred_classes)
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=["Negative", "Positive"], yticklabels=["Negative", "Positive"])
plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.title('Confusion Matrix')
plt.show()
```


Confusion Matrix
True label
Negative
Positive
Predicted label
Negative
Positive