```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
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, SimpleRNN, LSTM, Dense, Dropout
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report

# Set random seed
np.random.seed(42)
tf.random.set_seed(42)

# Create sample dataset (replace with your own dataset)
def load_data():
    # Sample data - in real application, load your dataset
    texts = [
        "I absolutely loved this movie, it was fantastic!",
        "This product is amazing, best purchase ever",
        "The service was terrible and the staff was rude",
        "I really hated the experience, would not recommend",
        "The movie was okay, nothing special but not bad either",
        "This restaurant is great, I'll definitely come back",
        "The quality of the product is poor",
        "I'm quite satisfied with my purchase",
        "The customer service was outstanding",
        "This is the worst experience I've ever had"
    ]

    # Labels: 0=negative, 1=neutral, 2=positive
    labels = [2, 2, 0, 0, 1, 2, 0, 1, 2, 0]

    # Generate more data by adding variations
    expanded_texts = texts.copy()
    expanded_labels = labels.copy()

    modifiers = ["really", "very", "extremely", "somewhat", "kind of"]
    for _ in range(90):  # Add 90 more samples
        idx = np.random.randint(0, len(texts))
        text = texts[idx]
        label = labels[idx]

        # Add random modifier
        if np.random.random() > 0.5:
            words = text.split()
            insert_pos = np.random.randint(1, len(words))
            words.insert(insert_pos, np.random.choice(modifiers))
            text = " ".join(words)

        expanded_texts.append(text)
        expanded_labels.append(label)

    return np.array(expanded_texts), np.array(expanded_labels)

# Load and prepare data
texts, labels = load_data()
X_train, X_test, y_train, y_test = train_test_split(texts, labels, test_size=0.2)

# Tokenize text
max_words = 5000
max_len = 100

tokenizer = Tokenizer(num_words=max_words)
tokenizer.fit_on_texts(X_train)
X_train_seq = tokenizer.texts_to_sequences(X_train)
X_test_seq = tokenizer.texts_to_sequences(X_test)

# Pad sequences
X_train_pad = pad_sequences(X_train_seq, maxlen=max_len)
X_test_pad = pad_sequences(X_test_seq, maxlen=max_len)

# Build RNN model
model = Sequential([
    Embedding(max_words, 128, input_length=max_len),
    LSTM(64, return_sequences=True),  # Use LSTM instead of SimpleRNN for better performance
    Dropout(0.3),
    LSTM(32),  # Second LSTM layer
    Dropout(0.3),
    Dense(3, activation='softmax')  # 3 classes: negative, neutral, positive
```

```
])

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

# Train model
history = model.fit(
    X_train_pad, y_train,
    epochs=10,
    batch_size=32,
    validation_split=0.2,
    verbose=1
)

# Evaluate model
loss, accuracy = model.evaluate(X_test_pad, y_test)
print(f"Test accuracy: {accuracy:.4f}")

# Make predictions
y_pred = model.predict(X_test_pad)
y_pred_classes = np.argmax(y_pred, axis=1)

# Generate confusion matrix
cm = confusion_matrix(y_test, y_pred_classes)
plt.figure(figsize=(8, 6))
plt.imshow(cm, interpolation='nearest', cmap='Blues')
plt.title('Confusion Matrix')
plt.colorbar()
plt.xlabel('Predicted')
plt.ylabel('True')
plt.xticks([0, 1, 2], ['Negative', 'Neutral', 'Positive'])
plt.yticks([0, 1, 2], ['Negative', 'Neutral', 'Positive'])

# Add labels in each cell
for i in range(3):
    for j in range(3):
        plt.text(j, i, str(cm[i, j]), ha='center', va='center')

plt.tight_layout()
plt.savefig('confusion_matrix.png')
plt.close()

# Print classification report
print("\nClassification Report:")
print(classification_report(y_test, y_pred_classes,
                           target_names=['Negative', 'Neutral', 'Positive']))

# Plot training history
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Training')
plt.plot(history.history['val_accuracy'], label='Validation')
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Training')
plt.plot(history.history['val_loss'], label='Validation')
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.tight_layout()
plt.savefig('training_history.png')
plt.close()

# Test with new examples
def predict_sentiment(text):
    sequence = tokenizer.texts_to_sequences([text])
    padded = pad_sequences(sequence, maxlen=max_len)
    prediction = model.predict(padded)[0]
    class_idx = np.argmax(prediction)
    sentiment = ['Negative', 'Neutral', 'Positive'][class_idx]
    return sentiment, prediction[class_idx]

# Test examples
test_examples = [
    "The food was delicious and the service was excellent",
    "This was a complete waste of money",
    "The product works as expected, nothing extraordinary"
```

```
]

print("\nPredictions on new examples:")
for text in test_examples:
    sentiment, confidence = predict_sentiment(text)
    print(f"Text: '{text}'")
    print(f"Sentiment: {sentiment} (confidence: {confidence:.4f})")
    print()

print("Analysis complete.")
```

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

Layer (type)	Output Shape	Param #
embedding (Embedding)	?	0 (unbuilt)
lstm (LSTM)	?	0 (unbuilt)
dropout (Dropout)	?	0
lstm_1 (LSTM)	?	0 (unbuilt)
dropout_1 (Dropout)	?	0
dense (Dense)	?	0 (unbuilt)

```
 Total params: 0 (0.00 B)
 Trainable params: 0 (0.00 B)
 Non-trainable params: 0 (0.00 B)
Epoch 1/10
2/2 ━━━━━━━━━━━━━━━━━━━━ 18s 2s/step - accuracy: 0.3333 - loss: 1.0985 - val_accuracy: 0.3750 - val_loss: 1.0777
Epoch 2/10
2/2 ━━━━━━━━━━━━━━━━━━━━ 3s 176ms/step - accuracy: 0.4375 - loss: 1.0705 - val_accuracy: 0.3750 - val_loss: 1.0532
Epoch 3/10
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 268ms/step - accuracy: 0.4375 - loss: 1.0397 - val_accuracy: 0.3750 - val_loss: 1.0282
Epoch 4/10
2/2 ━━━━━━━━━━━━━━━━━━━━ 1s 256ms/step - accuracy: 0.4375 - loss: 1.0052 - val_accuracy: 0.3750 - val_loss: 1.0061
Epoch 5/10
2/2 ━━━━━━━━━━━━━━━━━━━━ 1s 264ms/step - accuracy: 0.4375 - loss: 0.9772 - val_accuracy: 0.3750 - val_loss: 0.9730
Epoch 6/10
2/2 ━━━━━━━━━━━━━━━━━━━━ 1s 164ms/step - accuracy: 0.4375 - loss: 0.9723 - val_accuracy: 0.3750 - val_loss: 0.9185
Epoch 7/10
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 186ms/step - accuracy: 0.4583 - loss: 0.9290 - val_accuracy: 0.5000 - val_loss: 0.8658
Epoch 8/10
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 158ms/step - accuracy: 0.6562 - loss: 0.8393 - val_accuracy: 0.6250 - val_loss: 0.8111
Epoch 9/10
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 157ms/step - accuracy: 0.6875 - loss: 0.8011 - val_accuracy: 0.6250 - val_loss: 0.7494
Epoch 10/10
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 160ms/step - accuracy: 0.6771 - loss: 0.7376 - val_accuracy: 0.6250 - val_loss: 0.6919
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - accuracy: 0.8000 - loss: 0.6997
Test accuracy: 0.8000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 360ms/step

Classification Report:
              precision    recall  f1-score   support

    Negative       0.90      1.00      0.95         9
     Neutral       0.00      0.00      0.00         4
    Positive       0.70      1.00      0.82         7

    accuracy                           0.80        20
   macro avg       0.53      0.67      0.59        20
weighted avg       0.65      0.80      0.71        20

/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-define
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-define
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-define
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))

Predictions on new examples:
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 345ms/step
Text: 'The food was delicious and the service was excellent'
Sentiment: Positive (confidence: 0.5338)

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step
Text: 'This was a complete waste of money'
Sentiment: Positive (confidence: 0.8031)

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step
Text: 'The product works as expected, nothing extraordinary'
Sentiment: Positive (confidence: 0.7214)

Analysis complete.
```