Natural Language Processing

Assignment-06

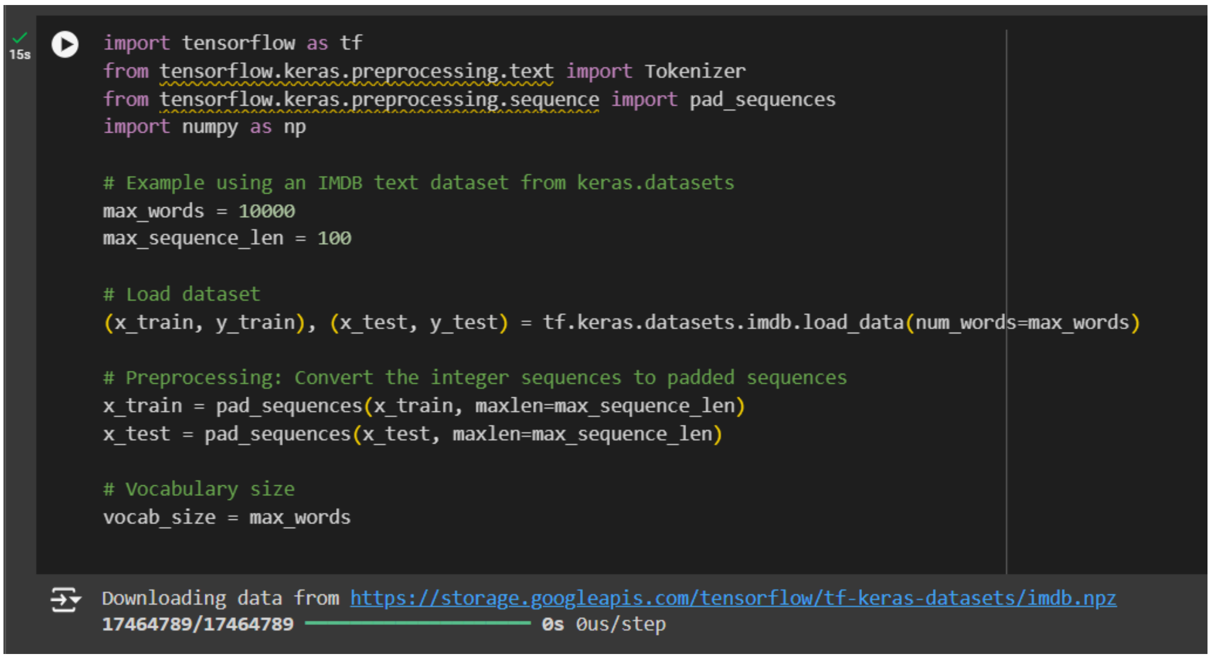
Name: Adithi Shinde

Enrollment No:2203A54032

Batch:40

1.Load data fromkeras.datasets and perform following computational analysis:- [CO3]

(a) Preprocessing of the Data



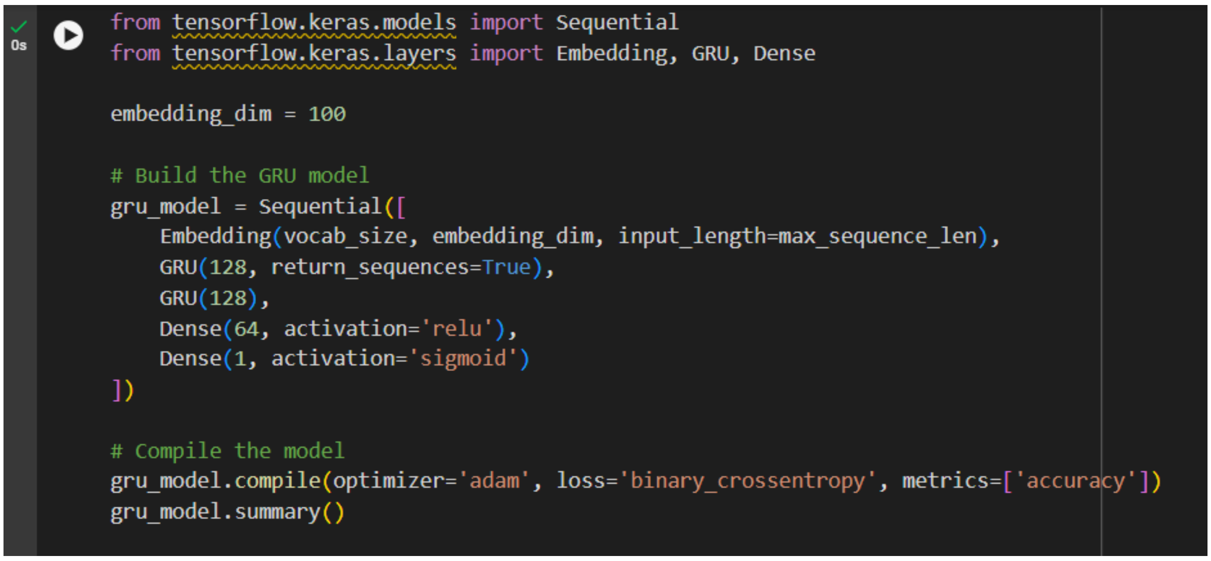
```
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
import numpy as np

# Example using an IMDB text dataset from keras.datasets
max_words = 10000
max_sequence_len = 100

# Load dataset
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.imdb.load_data(num_words=max_words)

# Preprocessing: Convert the integer sequences to padded sequences
x_train = pad_sequences(x_train, maxlen=max_sequence_len)
x_test = pad_sequences(x_test, maxlen=max_sequence_len)

# Vocabulary size
vocab_size = max_words
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/imdb.npz
17464789/17464789 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```

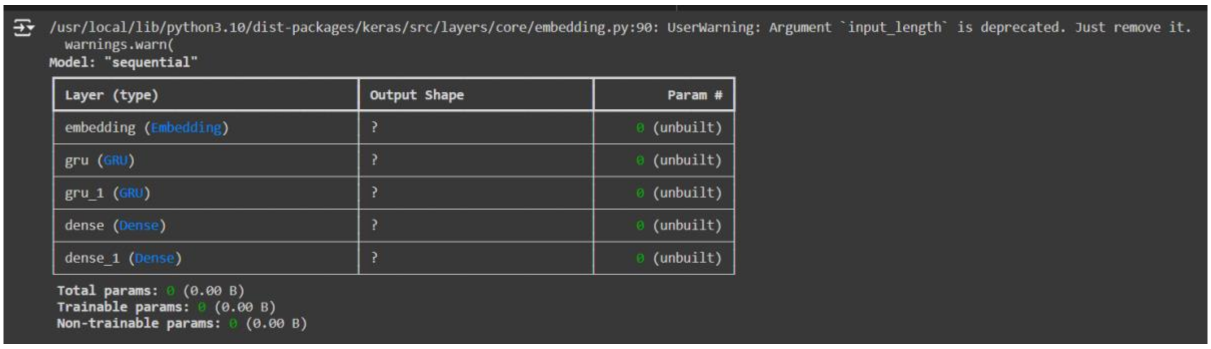
(b) Divide data into training and testing data set

This is already handled by the dataset split into x_train, y_train and x_test, y_test.

(c) Build the Gated Recurrent Units (GRU) Model

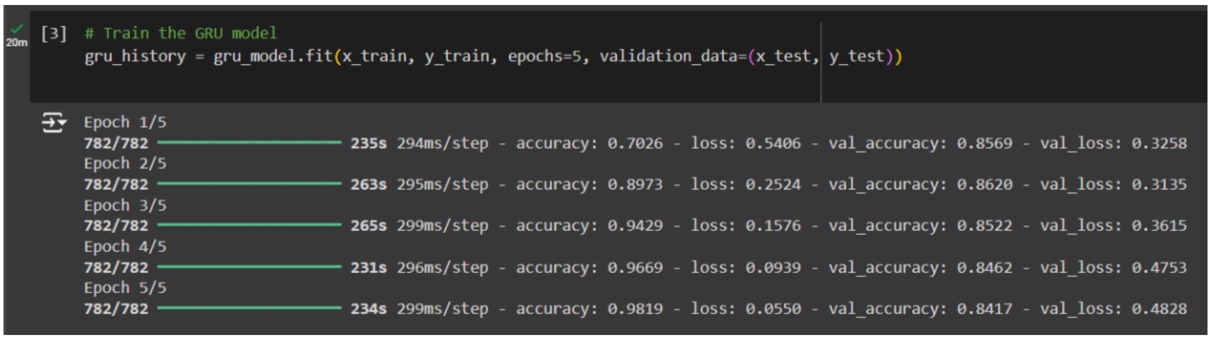


```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, GRU, Dense

embedding_dim = 100

# Build the GRU model
gru_model = Sequential([
    Embedding(vocab_size, embedding_dim, input_length=max_sequence_len),
    GRU(128, return_sequences=True),
    GRU(128),
    Dense(64, activation='relu'),
    Dense(1, activation='sigmoid')
])

# Compile the model
gru_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
gru_model.summary()
```



```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument `input_length` is deprecated. Just remove it.
  warnings.warn(
Model: "sequential"
┌──────────────────────────────┬────────────────────────┬──────────────┐
│ Layer (type)                 │ Output Shape           │      Param # │
├──────────────────────────────┼────────────────────────┼──────────────┤
│ embedding (Embedding)        │ ?                      │  0 (unbuilt) │
├──────────────────────────────┼────────────────────────┼──────────────┤
│ gru (GRU)                    │ ?                      │  0 (unbuilt) │
├──────────────────────────────┼────────────────────────┼──────────────┤
│ gru_1 (GRU)                  │ ?                      │  0 (unbuilt) │
├──────────────────────────────┼────────────────────────┼──────────────┤
│ dense (Dense)                │ ?                      │  0 (unbuilt) │
├──────────────────────────────┼────────────────────────┼──────────────┤
│ dense_1 (Dense)              │ ?                      │  0 (unbuilt) │
└──────────────────────────────┴────────────────────────┴──────────────┘
 Total params: 0 (0.00 B)
 Trainable params: 0 (0.00 B)
 Non-trainable params: 0 (0.00 B)
```

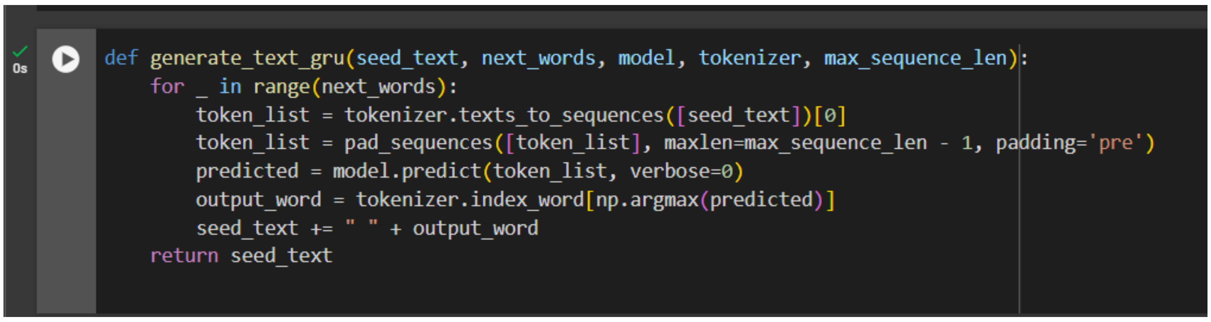
(d) Training the GRU Model



```
[3] # Train the GRU model
    gru_history = gru_model.fit(x_train, y_train, epochs=5, validation_data=(x_test, y_test))
```

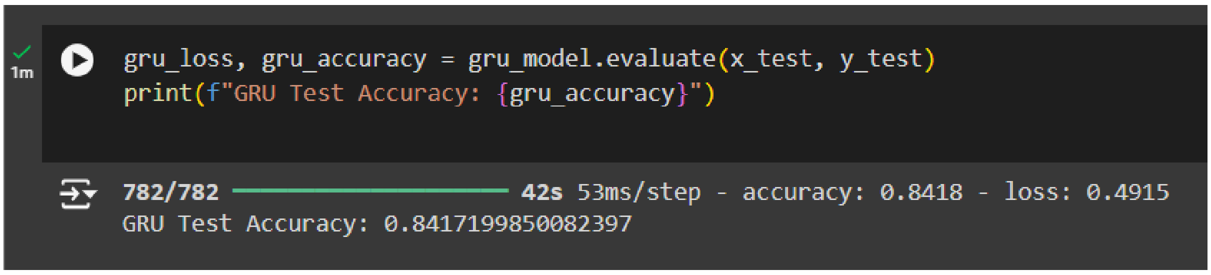
```
Epoch 1/5
782/782 ━━━━━━━━━━━━━━━━━━━━ 235s 294ms/step - accuracy: 0.7026 - loss: 0.5406 - val_accuracy: 0.8569 - val_loss: 0.3258
Epoch 2/5
782/782 ━━━━━━━━━━━━━━━━━━━━ 263s 295ms/step - accuracy: 0.8973 - loss: 0.2524 - val_accuracy: 0.8620 - val_loss: 0.3135
Epoch 3/5
782/782 ━━━━━━━━━━━━━━━━━━━━ 265s 299ms/step - accuracy: 0.9429 - loss: 0.1576 - val_accuracy: 0.8522 - val_loss: 0.3615
Epoch 4/5
782/782 ━━━━━━━━━━━━━━━━━━━━ 231s 296ms/step - accuracy: 0.9669 - loss: 0.0939 - val_accuracy: 0.8462 - val_loss: 0.4753
Epoch 5/5
782/782 ━━━━━━━━━━━━━━━━━━━━ 234s 299ms/step - accuracy: 0.9819 - loss: 0.0550 - val_accuracy: 0.8417 - val_loss: 0.4828
```

(e) Text Generation Using the Trained Model



```python
def generate_text_gru(seed_text, next_words, model, tokenizer, max_sequence_len):
    for _ in range(next_words):
        token_list = tokenizer.texts_to_sequences([seed_text])[0]
        token_list = pad_sequences([token_list], maxlen=max_sequence_len - 1, padding='pre')
        predicted = model.predict(token_list, verbose=0)
        output_word = tokenizer.index_word[np.argmax(predicted)]
        seed_text += " " + output_word
    return seed_text
```

(f) Evaluate Model’s accuracy

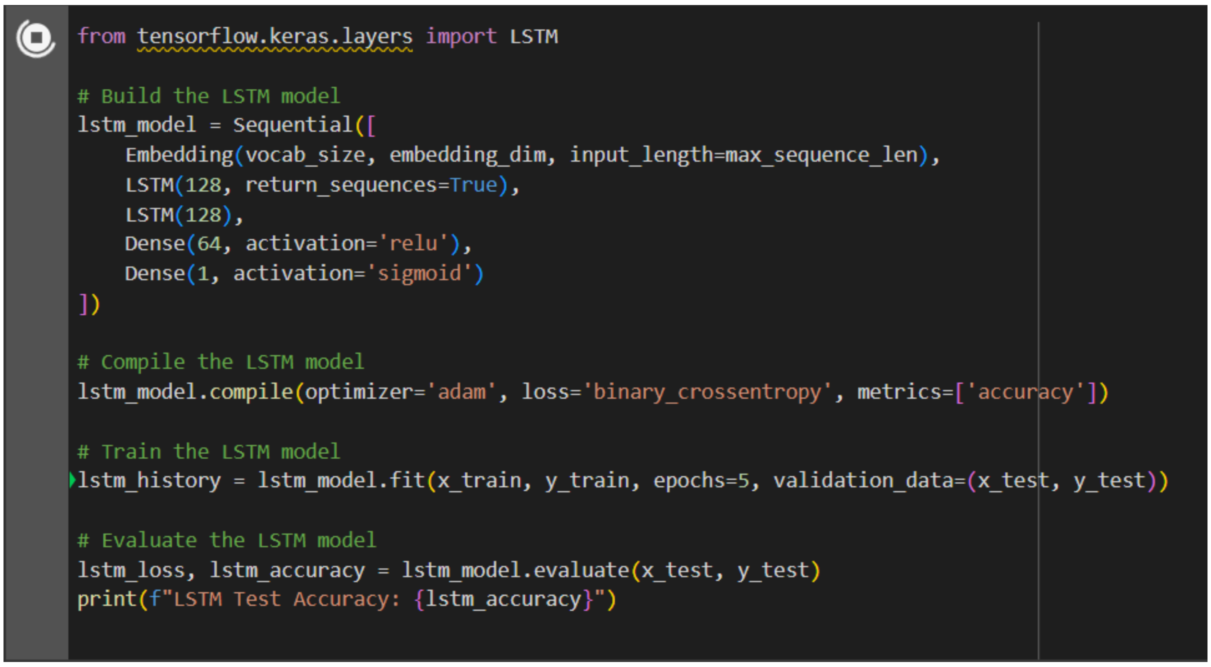


```
gru_loss, gru_accuracy = gru_model.evaluate(x_test, y_test)
print(f"GRU Test Accuracy: {gru_accuracy}")
```

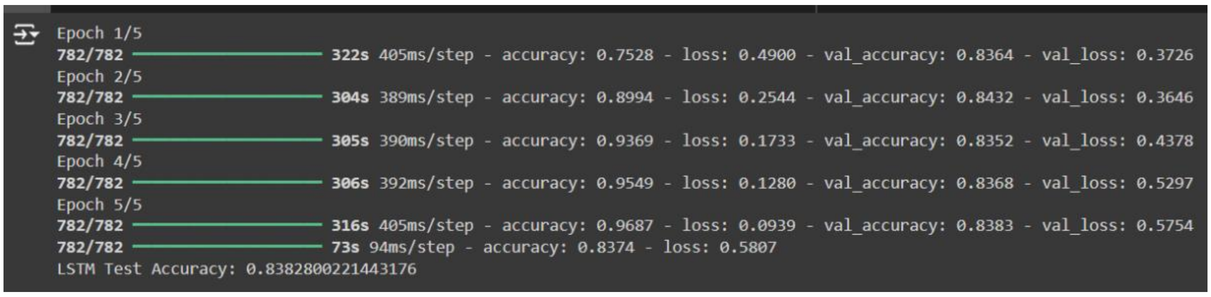
```
782/782 ━━━━━━━━━━━━━━━━━━━━ 42s 53ms/step - accuracy: 0.8418 - loss: 0.4915
GRU Test Accuracy: 0.8417199850082397
```

2.Compare accuracy of Long sort term memory and Gated recurrent Unit models for text generation using data from tensorflow.keras.datasets. [CO3]

i. Building and Training the LSTM Model

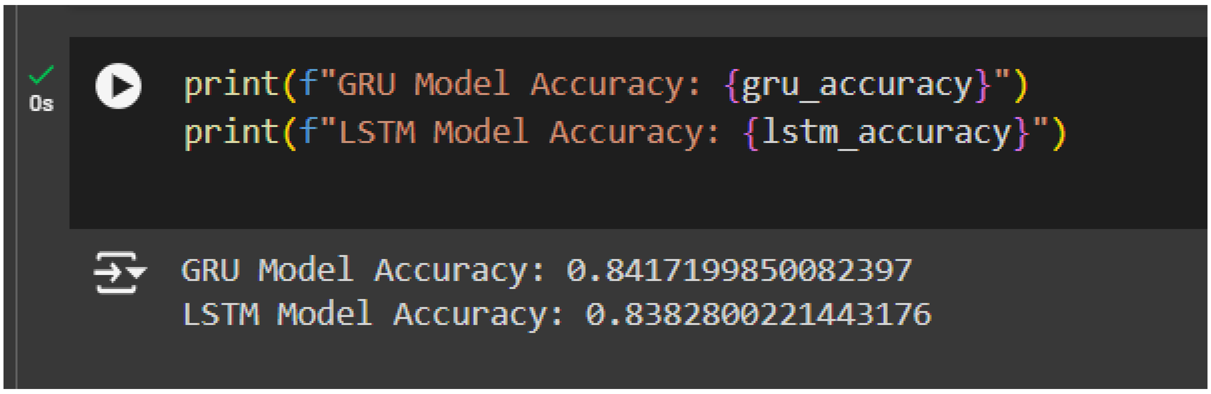


```
from tensorflow.keras.layers import LSTM

# Build the LSTM model
lstm_model = Sequential([
    Embedding(vocab_size, embedding_dim, input_length=max_sequence_len),
    LSTM(128, return_sequences=True),
    LSTM(128),
    Dense(64, activation='relu'),
    Dense(1, activation='sigmoid')
])

# Compile the LSTM model
lstm_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Train the LSTM model
lstm_history = lstm_model.fit(x_train, y_train, epochs=5, validation_data=(x_test, y_test))

# Evaluate the LSTM model
lstm_loss, lstm_accuracy = lstm_model.evaluate(x_test, y_test)
print(f"LSTM Test Accuracy: {lstm_accuracy}")
```



```
Epoch 1/5
782/782 ━━━━━━━━━━━━━━━━━━━━ 322s 405ms/step - accuracy: 0.7528 - loss: 0.4900 - val_accuracy: 0.8364 - val_loss: 0.3726
Epoch 2/5
782/782 ━━━━━━━━━━━━━━━━━━━━ 304s 389ms/step - accuracy: 0.8994 - loss: 0.2544 - val_accuracy: 0.8432 - val_loss: 0.3646
Epoch 3/5
782/782 ━━━━━━━━━━━━━━━━━━━━ 305s 390ms/step - accuracy: 0.9369 - loss: 0.1733 - val_accuracy: 0.8352 - val_loss: 0.4378
Epoch 4/5
782/782 ━━━━━━━━━━━━━━━━━━━━ 306s 392ms/step - accuracy: 0.9549 - loss: 0.1280 - val_accuracy: 0.8368 - val_loss: 0.5297
Epoch 5/5
782/782 ━━━━━━━━━━━━━━━━━━━━ 316s 405ms/step - accuracy: 0.9687 - loss: 0.0939 - val_accuracy: 0.8383 - val_loss: 0.5754
782/782 ━━━━━━━━━━━━━━━━━━━━ 73s 94ms/step - accuracy: 0.8374 - loss: 0.5807
LSTM Test Accuracy: 0.8382800221443176
```

ii. Comparison of GRU and LSTM Models



```python
print(f"GRU Model Accuracy: {gru_accuracy}")
print(f"LSTM Model Accuracy: {lstm_accuracy}")
```

```
GRU Model Accuracy: 0.8417199850082397
LSTM Model Accuracy: 0.8382800221443176
```