```
#To implement a Recurrent Neural Network (RNN) for review classification on the IMDB dataset.
import tensorflow as tf
from tensorflow.keras.datasets import imdb
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.layers import Embedding, SimpleRNN, Dense
from tensorflow.keras.models import Sequential
from matplotlib import pyplot as plt
import os



# Set the model directory
model_directory = 'models'
# Create the model directory if it does not exist
if not os.path.exists(model_directory):
  os.makedirs(model_directory)
```

+ Code + Text

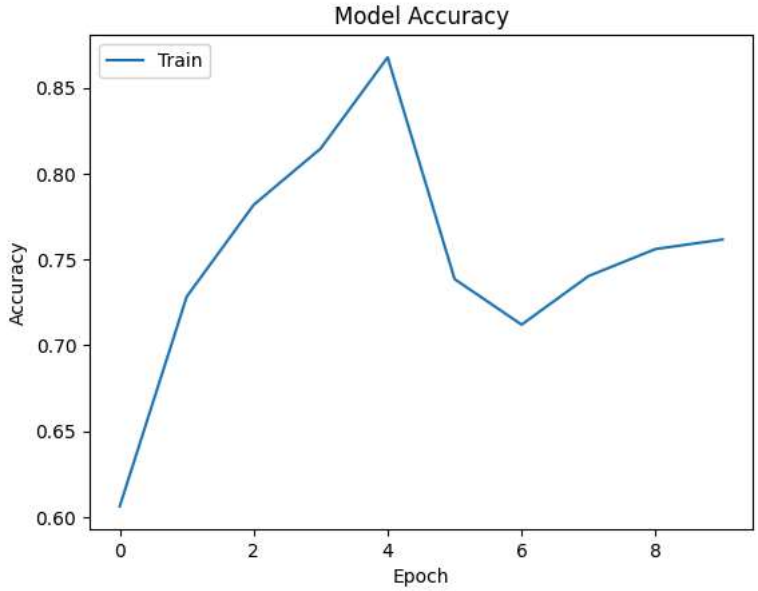
```
# Set the hyperparameters
max_features = 10000
maxlen = 200
embedding_size = 128
rnn_size = 128
batch_size = 32
epochs = 10
# Load the IMDB dataset
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=max_features)
# Pad the sequences to a fixed length
x_train = pad_sequences(x_train, maxlen=maxlen)
x_test = pad_sequences(x_test, maxlen=maxlen)
x_train[100]
# Create the RNN model
model = Sequential()
model.add(Embedding(max_features, embedding_size, input_length=maxlen))
model.add(SimpleRNN(rnn_size))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
history = model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/imdb.npz
17464789/17464789 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument `input_length` is deprecated. Just
  warnings.warn(
Epoch 1/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 78s 95ms/step - accuracy: 0.5764 - loss: 0.6595
Epoch 2/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 86s 101ms/step - accuracy: 0.7791 - loss: 0.4820
Epoch 3/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 77s 99ms/step - accuracy: 0.6743 - loss: 0.6052
Epoch 4/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 80s 97ms/step - accuracy: 0.7711 - loss: 0.4779
Epoch 5/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 83s 99ms/step - accuracy: 0.7447 - loss: 0.5038
Epoch 6/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 75s 96ms/step - accuracy: 0.8366 - loss: 0.3977
Epoch 7/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 78s 90ms/step - accuracy: 0.7935 - loss: 0.4420
Epoch 8/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 82s 90ms/step - accuracy: 0.8673 - loss: 0.3262
Epoch 9/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 82s 90ms/step - accuracy: 0.8678 - loss: 0.3232
Epoch 10/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 71s 91ms/step - accuracy: 0.8950 - loss: 0.2712
```

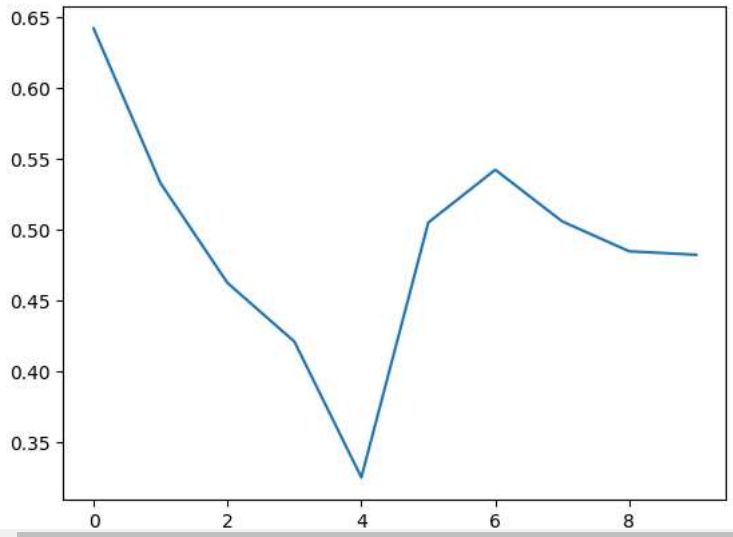
```
# Evaluate the model
loss, accuracy = model.evaluate(x_test, y_test)
print('Test loss:', loss)
print('Test accuracy:', accuracy)
# Plot the accuracy from the training history
#plt.plot(history.history['binary_crossentropy'])
plt.plot(history.history['accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
```

```
plt.show()
plt.plot(history.history['loss'])
```

```
782/782 ━━━━━━━━━━━━━━━━━━━━ 16s 18ms/step - accuracy: 0.7161 - loss: 0.5942
Test loss: 0.5874921083450317
Test accuracy: 0.7215200066566467
```




```
[<matplotlib.lines.Line2D at 0x7e4459129b40>]
```



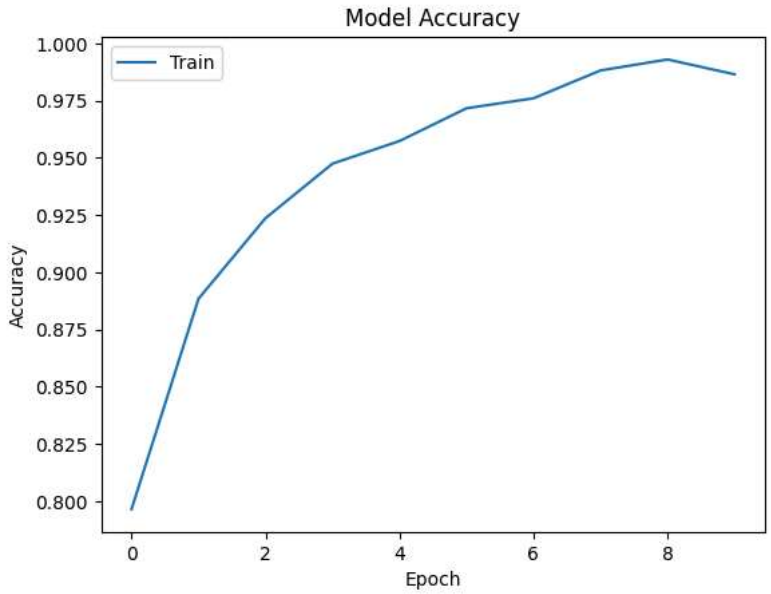

```
#8.2 - To implement a Long Short-Term Memory (LSTM) for review classification on the IMDB dataset.
import tensorflow as tf
from tensorflow.keras.datasets import imdb
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.layers import Embedding, SimpleRNN, Dense
from tensorflow.keras.layers import GRU
from tensorflow.keras.models import Sequential
from matplotlib import pyplot as plt
import os


# Set the model directory
model_directory = 'models'
# Create the model directory if it does not exist
if not os.path.exists(model_directory):
  os.makedirs(model_directory)


# Set the hyperparameters
max_features = 10000
maxlen = 200
embedding_size = 128
batch_size = 32
epochs = 10
```

```
# Load the IMDB dataset
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=max_features)
# Pad the sequences to a fixed length
x_train = pad_sequences(x_train, maxlen=maxlen)
x_test = pad_sequences(x_test, maxlen=maxlen)
x_train[100]
# Create the RNN model
model = Sequential()
model.add(Embedding(max_features, embedding_size, input_length=maxlen))
model.add(GRU(128))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
history = model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs)
```

```
Epoch 1/10
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument `input_length` is deprecated. Just
  warnings.warn(
782/782 ━━━━━━━━━━━━━━━━━━━━ 13s 12ms/step - accuracy: 0.7344 - loss: 0.5149
Epoch 2/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 19s 12ms/step - accuracy: 0.8867 - loss: 0.2863
Epoch 3/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 9s 11ms/step - accuracy: 0.9279 - loss: 0.1897
Epoch 4/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 10s 13ms/step - accuracy: 0.9518 - loss: 0.1316
Epoch 5/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 8s 11ms/step - accuracy: 0.9621 - loss: 0.1078
Epoch 6/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - accuracy: 0.9748 - loss: 0.0755
Epoch 7/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 10s 12ms/step - accuracy: 0.9824 - loss: 0.0557
Epoch 8/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 9s 11ms/step - accuracy: 0.9896 - loss: 0.0356
Epoch 9/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 11s 12ms/step - accuracy: 0.9938 - loss: 0.0217
Epoch 10/10
782/782 ━━━━━━━━━━━━━━━━━━━━ 9s 11ms/step - accuracy: 0.9857 - loss: 0.0437
```

```
# Evaluate the model
loss, accuracy = model.evaluate(x_test, y_test)
print('Test loss:', loss)
print('Test accuracy:', accuracy)
# Plot the accuracy from the training history
#plt.plot(history.history['binary_crossentropy'])
plt.plot(history.history['accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
plt.plot(history.history['loss'])
```

```
782/782 ━━━━━━━━━━━━━━━━━━━━ 6s 7ms/step - accuracy: 0.8441 - loss: 0.7915
Test loss: 0.7799365520477295
Test accuracy: 0.8448799848556519
```




```
[<matplotlib.lines.Line2D at 0x7e44529320e0>]
```