```
from keras.datasets import imdb

# Load the IMDB dataset
max_words  = 10000
(train_data, train_labels), (test_data, test_labels) =
imdb.load_data(num_words=max_words)

from tensorflow.keras.preprocessing.sequence import pad_sequences

# Truncate or pad the reviews to a length of 150 words
maxlen = 150
train_data = pad_sequences(train_data, maxlen=maxlen)
test_data = pad_sequences(test_data, maxlen=maxlen)

# Select 5000 samples for testing
test_data = test_data[:5000]
test_labels = test_labels[:5000]

# Select 10,000 samples for validation
val_data = test_data[:10000]
val_labels = test_labels[:10000]

from keras.models import Sequential
from keras.layers import Embedding, Bidirectional, LSTM, Dense,
Dropout, BatchNormalization

# Build The RNN model
rnn_model = Sequential()

rnn_model.add(Embedding(10000, 32, input_length=len(train_data[0])))
rnn_model.add(Bidirectional(LSTM(64, return_sequences=True)))
rnn_model.add(Dropout(0.5))
rnn_model.add(BatchNormalization())
rnn_model.add(Bidirectional(LSTM(32)))
rnn_model.add(Dropout(0.5))
rnn_model.add(BatchNormalization())
rnn_model.add(Dense(1, activation='sigmoid'))
rnn_model.compile(loss="binary_crossentropy", optimizer="rmsprop",
metrics=["accuracy"])

# Print model summary
print(" ")
print("RNN Model Architecture:")
print(rnn_model.summary())
print(" ")


RNN Model Architecture:
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
```

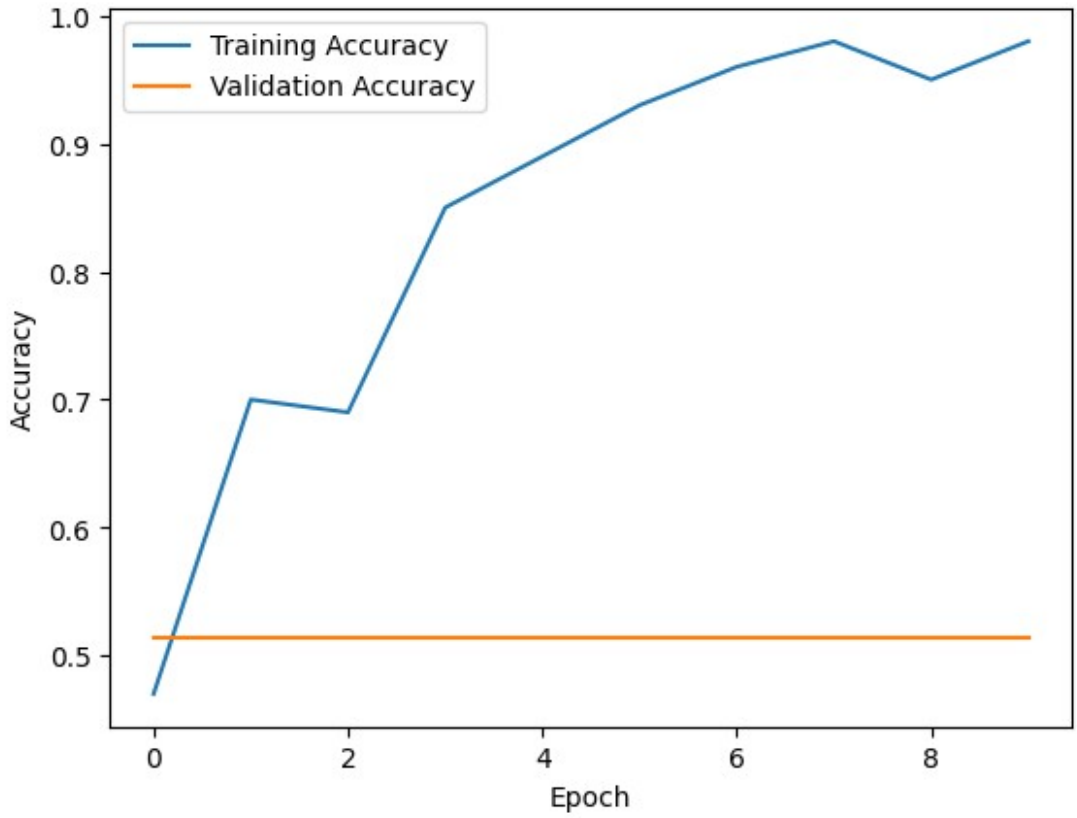
```
=================================================================
 embedding_2 (Embedding)     (None, 150, 32)           320000

 bidirectional_4 (Bidirectio  (None, 150, 128)         49664
 nal)

 dropout_4 (Dropout)         (None, 150, 128)          0

 batch_normalization_4 (Batc  (None, 150, 128)         512
 hNormalization)

 bidirectional_5 (Bidirectio  (None, 64)               41216
 nal)

 dropout_5 (Dropout)         (None, 64)                0

 batch_normalization_5 (Batc  (None, 64)               256
 hNormalization)

 dense_2 (Dense)             (None, 1)                 65

=================================================================
Total params: 411,713
Trainable params: 411,329
Non-trainable params: 384
_________________________________________________________________
None


import numpy as np

# Load GloVe word embeddings
embeddings_index = {}
with open("C:/Users/saisu/Downloads/glove.6B.100d.txt", encoding='utf-
8') as f:
    for line in f:
        values = line.split()
        word = values[0]
        coefs = np.asarray(values[1:], dtype='float32')
        embeddings_index[word] = coefs

embedding_dim = 100
embedding_matrix = np.zeros((10000, embedding_dim))
for i, word in enumerate(embeddings_index.keys()):
    if i < 10000:
        embedding_vector = embeddings_index.get(word)
        if embedding_vector is not None:
            embedding_matrix[i] = embedding_vector

# Define the model with pretrained word embeddings
```

```
rnn_model_pretrained = Sequential()
rnn_model_pretrained.add(Embedding(10000, embedding_dim,
input_length=maxlen, trainable=False))
rnn_model_pretrained.add(Bidirectional(LSTM(64,
return_sequences=True)))
rnn_model_pretrained.add(Dropout(0.5))
rnn_model_pretrained.add(BatchNormalization())
rnn_model_pretrained.add(Bidirectional(LSTM(32)))
rnn_model_pretrained.add(Dropout(0.5))
rnn_model_pretrained.add(BatchNormalization())
rnn_model_pretrained.add(Dense(1, activation='sigmoid'))
rnn_model_pretrained.compile(loss="binary_crossentropy",
optimizer="rmsprop", metrics=["accuracy"])

# Print model summary
print(" ")
print("RNN Model Pre Trained Architecture:")
print(rnn_model_pretrained.summary())
print(" ")
```

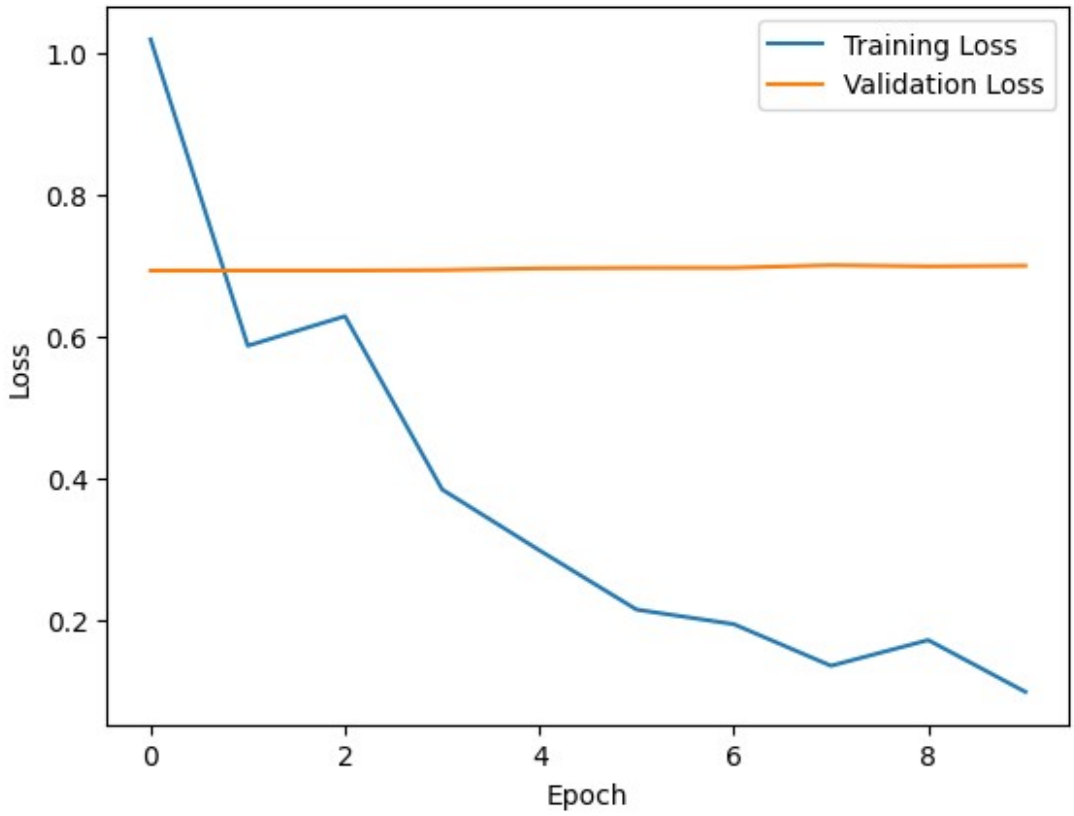
```
RNN Model Pre Trained Architecture:
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_3 (Embedding)     (None, 150, 100)          1000000

 bidirectional_6 (Bidirectio  (None, 150, 128)         84480
 nal)

 dropout_6 (Dropout)         (None, 150, 128)          0

 batch_normalization_6 (Batc  (None, 150, 128)         512
 hNormalization)

 bidirectional_7 (Bidirectio  (None, 64)               41216
 nal)

 dropout_7 (Dropout)         (None, 64)                0

 batch_normalization_7 (Batc  (None, 64)               256
 hNormalization)

 dense_3 (Dense)             (None, 1)                 65

=================================================================
Total params: 1,126,529
Trainable params: 126,145
Non-trainable params: 1,000,384
```

```
_________________________________________________________________
None


# Select the first 100 samples for training
train_data_100 = train_data[:100]
train_labels_100 = train_labels[:100]

# Train the RNN model
rnn_model_100 = rnn_model
rnn_history_100 = rnn_model_100.fit(train_data_100, train_labels_100,
epochs=10, batch_size=32, validation_data=(val_data, val_labels))

# Evaluate the model
test_loss_rnn100, test_accuracy_rnn100 =
rnn_model_100.evaluate(test_data, test_labels)

print("Test Loss : ", test_loss_rnn100)
print("Test Accuracy : ", test_accuracy_rnn100)


#Model Perfomance Evaluation
import matplotlib.pyplot as plt

print(" ")
print("Perfomance of RNN Model for 100 Training Samples : ")
print(" ")
# Plot training and validation accuracy
print("Accuracy : ")
print(" ")
plt.plot(rnn_history_100.history['accuracy'], label='Training
Accuracy')
plt.plot(rnn_history_100.history['val_accuracy'], label='Validation
Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()

# Plot training and validation loss
print(" ")
print("Loss : ")
print(" ")
plt.plot(rnn_history_100.history['loss'], label='Training Loss')
plt.plot(rnn_history_100.history['val_loss'], label='Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

```
Epoch 1/10
4/4 [==============================] - 23s 3s/step - loss: 1.0194 -
accuracy: 0.4700 - val_loss: 0.6927 - val_accuracy: 0.5142
Epoch 2/10
4/4 [==============================] - 5s 2s/step - loss: 0.5866 -
accuracy: 0.7000 - val_loss: 0.6928 - val_accuracy: 0.5142
Epoch 3/10
4/4 [==============================] - 5s 2s/step - loss: 0.6284 -
accuracy: 0.6900 - val_loss: 0.6928 - val_accuracy: 0.5142
Epoch 4/10
4/4 [==============================] - 5s 2s/step - loss: 0.3836 -
accuracy: 0.8500 - val_loss: 0.6935 - val_accuracy: 0.5142
Epoch 5/10
4/4 [==============================] - 5s 2s/step - loss: 0.2977 -
accuracy: 0.8900 - val_loss: 0.6959 - val_accuracy: 0.5142
Epoch 6/10
4/4 [==============================] - 5s 2s/step - loss: 0.2139 -
accuracy: 0.9300 - val_loss: 0.6966 - val_accuracy: 0.5142
Epoch 7/10
4/4 [==============================] - 5s 2s/step - loss: 0.1934 -
accuracy: 0.9600 - val_loss: 0.6968 - val_accuracy: 0.5142
Epoch 8/10
4/4 [==============================] - 5s 2s/step - loss: 0.1346 -
accuracy: 0.9800 - val_loss: 0.7004 - val_accuracy: 0.5142
Epoch 9/10
4/4 [==============================] - 5s 2s/step - loss: 0.1709 -
accuracy: 0.9500 - val_loss: 0.6988 - val_accuracy: 0.5142
Epoch 10/10
4/4 [==============================] - 5s 2s/step - loss: 0.0977 -
accuracy: 0.9800 - val_loss: 0.6996 - val_accuracy: 0.5142
157/157 [==============================] - 5s 31ms/step - loss: 0.6996
- accuracy: 0.5142
Test Loss :  0.699574887752533
Test Accuracy :  0.51419997215271

Perfomance of RNN Model for 100 Training Samples :

Accuracy :
```


Training Accuracy
Validation Accuracy
Accuracy
Epoch
1.0
0.9
0.8
0.7
0.6
0.5
0
2
4
6
8

```
Loss :
```

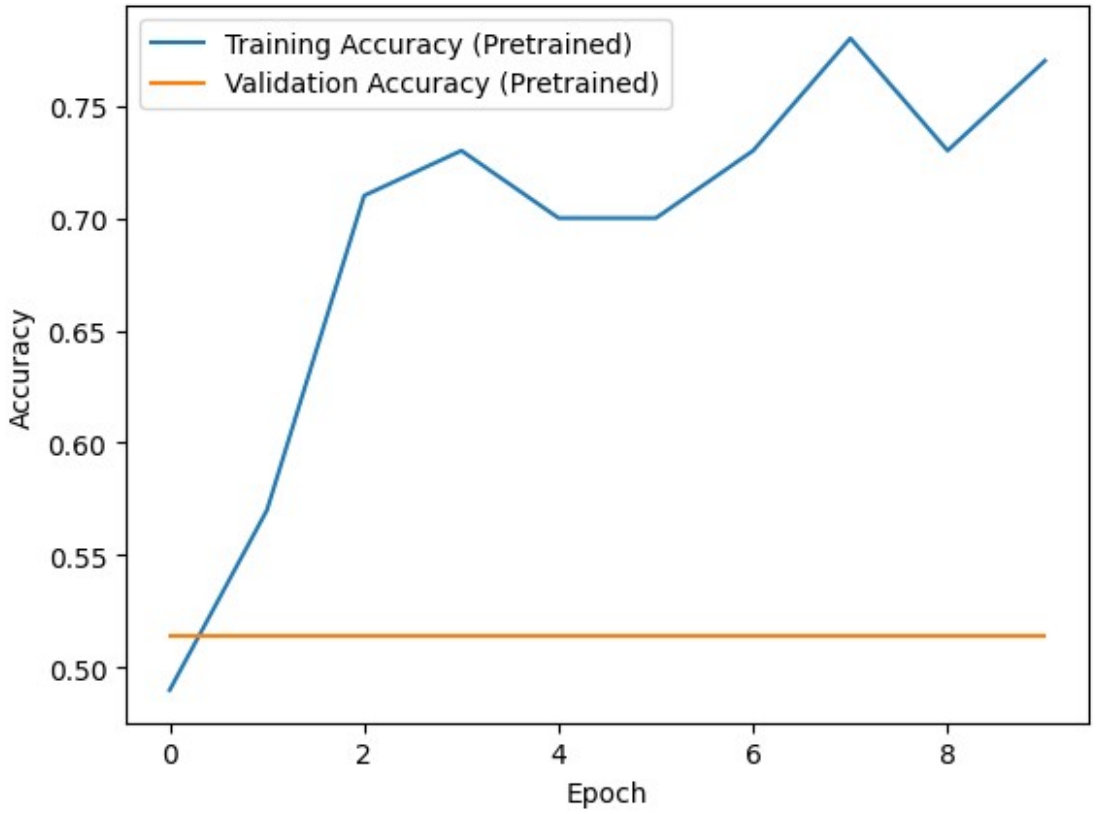



```
# Train the RNN model with pretrained embeddings
rnn_model_pretrained_100 = rnn_model_pretrained
rnn_history_pretrained_100 =
rnn_model_pretrained_100.fit(train_data_100, train_labels_100,
epochs=10, batch_size=32, validation_data=(val_data, val_labels))

# Evaluate the model on the test data
test_loss_pre_trained_rnn100, test_accuracy_pre_trained_rnn100 =
rnn_model_pretrained_100.evaluate(test_data, test_labels)

print("Test Loss : ", test_loss_pre_trained_rnn100)
print("Test Accuracy : ", test_accuracy_pre_trained_rnn100)

# Plot training and validation accuracy
print("Perfomance of Pre Trained RNN Model for 100 Training Samples :
")
print(" ")
print("Accuracy : ")
print(" ")
plt.plot(rnn_history_pretrained_100.history['accuracy'],
label='Training Accuracy (Pretrained)')
plt.plot(rnn_history_pretrained_100.history['val_accuracy'],
label='Validation Accuracy (Pretrained)')
plt.xlabel('Epoch')
```

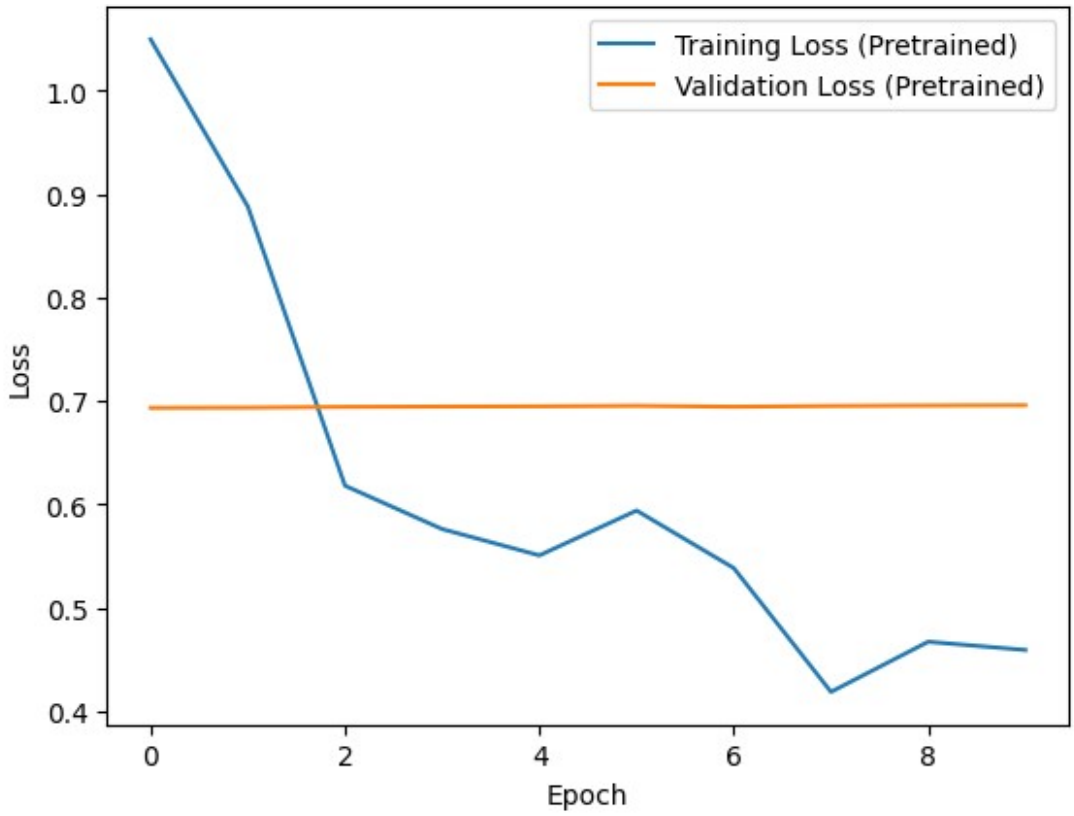
```
plt.ylabel('Accuracy')
plt.legend()
plt.show()

print(" ")
print("Loss : ")
print(" ")
# Plot training and validation loss
plt.plot(rnn_history_pretrained_100.history['loss'], label='Training
Loss (Pretrained)')
plt.plot(rnn_history_pretrained_100.history['val_loss'],
label='Validation Loss (Pretrained)')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

```
Epoch 1/10
4/4 [==============================] - 17s 3s/step - loss: 1.0498 -
accuracy: 0.4900 - val_loss: 0.6932 - val_accuracy: 0.5142
Epoch 2/10
4/4 [==============================] - 6s 2s/step - loss: 0.8880 -
accuracy: 0.5700 - val_loss: 0.6935 - val_accuracy: 0.5142
Epoch 3/10
4/4 [==============================] - 6s 2s/step - loss: 0.6179 -
accuracy: 0.7100 - val_loss: 0.6942 - val_accuracy: 0.5142
Epoch 4/10
4/4 [==============================] - 6s 2s/step - loss: 0.5760 -
accuracy: 0.7300 - val_loss: 0.6945 - val_accuracy: 0.5142
Epoch 5/10
4/4 [==============================] - 6s 2s/step - loss: 0.5505 -
accuracy: 0.7000 - val_loss: 0.6947 - val_accuracy: 0.5142
Epoch 6/10
4/4 [==============================] - 6s 2s/step - loss: 0.5939 -
accuracy: 0.7000 - val_loss: 0.6952 - val_accuracy: 0.5142
Epoch 7/10
4/4 [==============================] - 6s 2s/step - loss: 0.5382 -
accuracy: 0.7300 - val_loss: 0.6943 - val_accuracy: 0.5142
Epoch 8/10
4/4 [==============================] - 6s 2s/step - loss: 0.4184 -
accuracy: 0.7800 - val_loss: 0.6951 - val_accuracy: 0.5142
Epoch 9/10
4/4 [==============================] - 6s 2s/step - loss: 0.4669 -
accuracy: 0.7300 - val_loss: 0.6955 - val_accuracy: 0.5142
Epoch 10/10
4/4 [==============================] - 6s 2s/step - loss: 0.4590 -
accuracy: 0.7700 - val_loss: 0.6958 - val_accuracy: 0.5142
157/157 [==============================] - 5s 34ms/step - loss: 0.6958
- accuracy: 0.5142
Test Loss :  0.6957815885543823
```

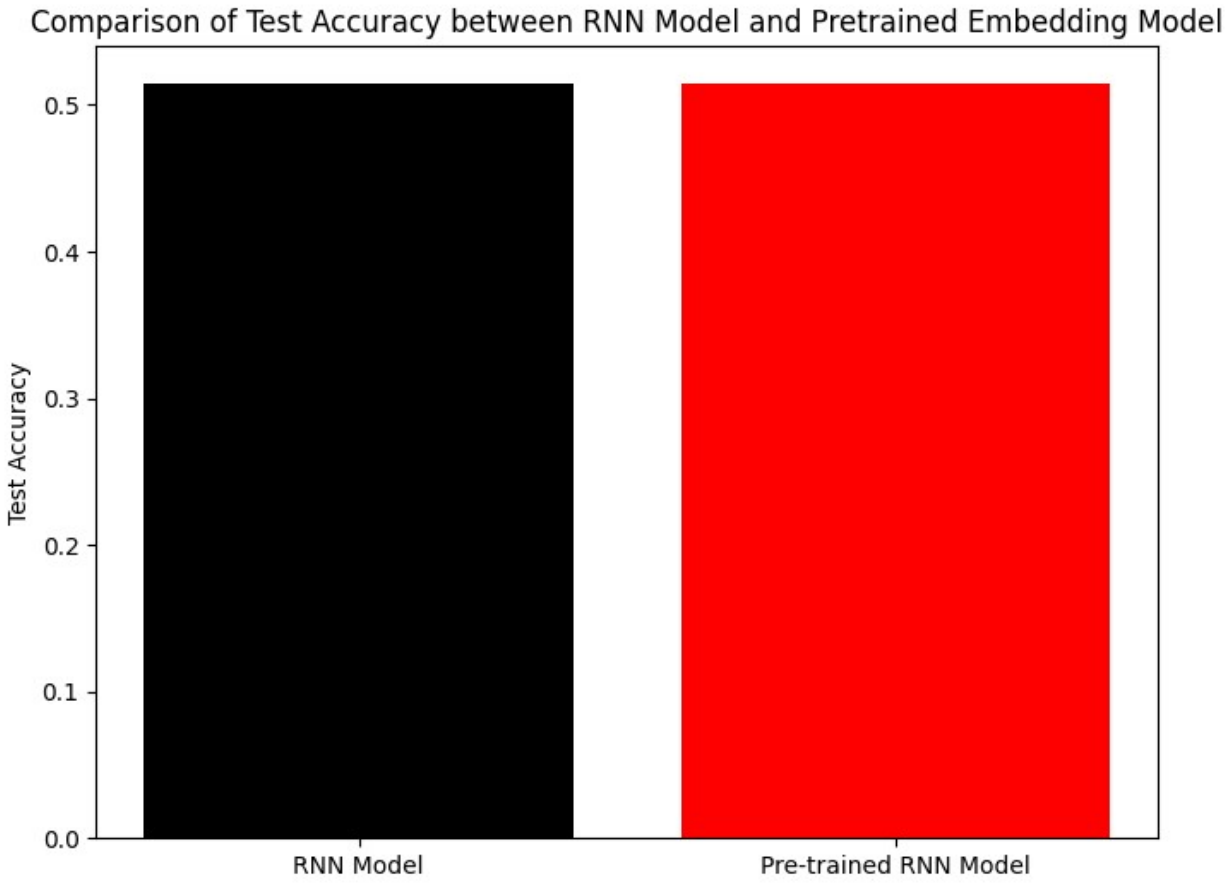
```
Test Accuracy :  0.51419997215271
Perfomance of Pre Trained RNN Model for 100 Training Samples :

Accuracy :
```



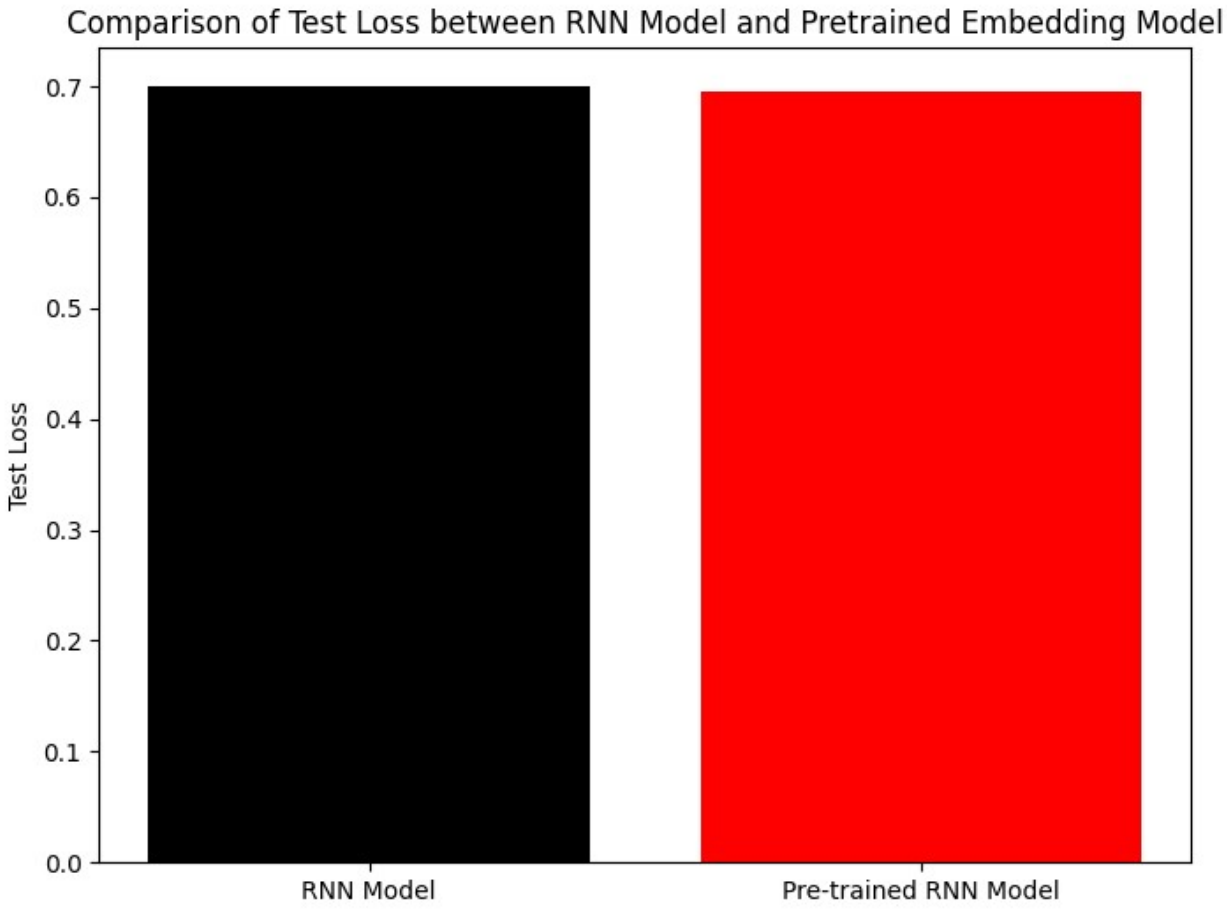

```
Loss :
```




```
# Model names for labeling
model_names = ['RNN Model', 'Pre-trained RNN Model']

# Plot comparison graph
plt.figure(figsize=(8, 6))
plt.bar(model_names, [test_accuracy_rnn100,
test_accuracy_pre_trained_rnn100], color=['black', 'red'])
plt.title('Comparison of Test Accuracy between RNN Model and
Pretrained Embedding Model')
plt.ylabel('Test Accuracy')
plt.show()
```




```
# Model names for labeling
model_names = ['RNN Model', 'Pre-trained RNN Model']

# Plot comparison graph
plt.figure(figsize=(8, 6))
plt.bar(model_names, [test_loss_rnn100, test_loss_pre_trained_rnn100],
color=['black', 'red'])
plt.title('Comparison of Test Loss between RNN Model and Pretrained
Embedding Model')
plt.ylabel('Test Loss')
plt.show()
```

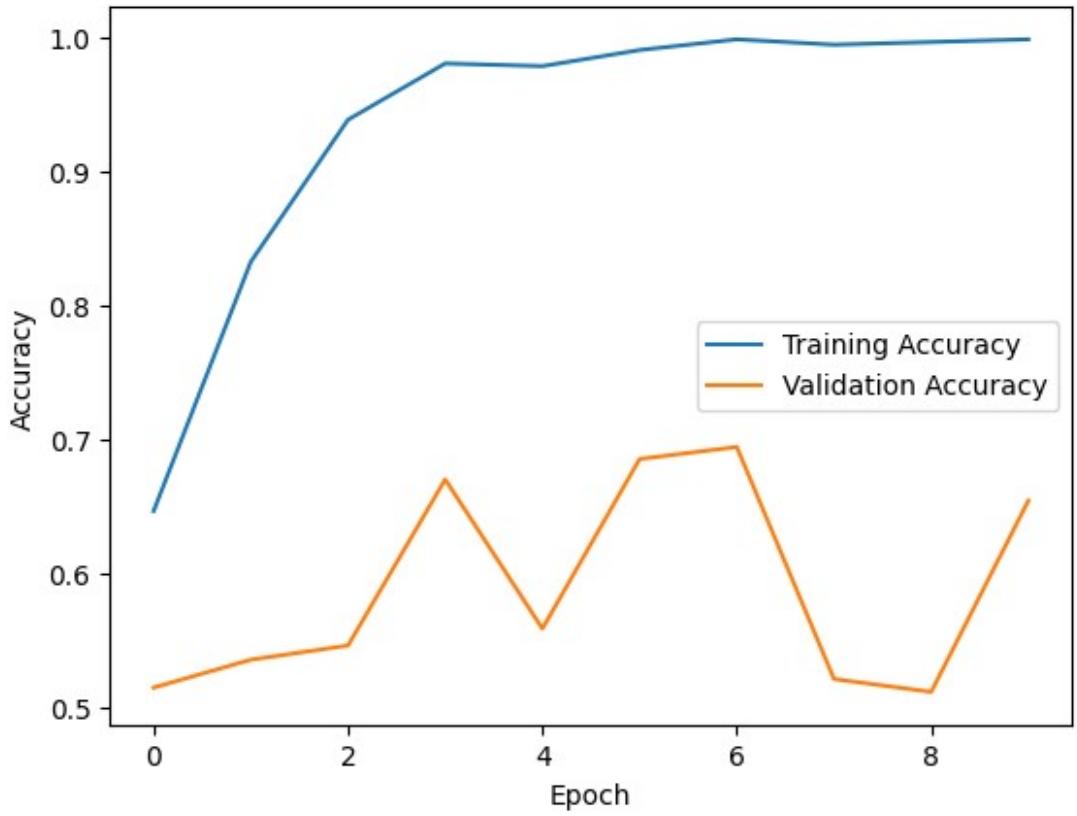



```
# Select the first 500 samples for training
train_data_500 = train_data[:500]
train_labels_500 = train_labels[:500]

# Train the RNN model
rnn_model_500 = rnn_model
rnn_history_500 = rnn_model_500.fit(train_data_500, train_labels_500,
epochs=10, batch_size=32, validation_data=(val_data, val_labels))

# Evaluate the model
test_loss_rnn500, test_accuracy_rnn500 =
rnn_model_500.evaluate(test_data, test_labels)

print("Test Loss : ", test_loss_rnn500)
print("Test Accuracy : ", test_accuracy_rnn500)


#Model Perfomance Evaluation
print(" ")
print("Perfomance of RNN Model for 500 Training Samples : ")
print(" ")
# Plot training and validation accuracy
```

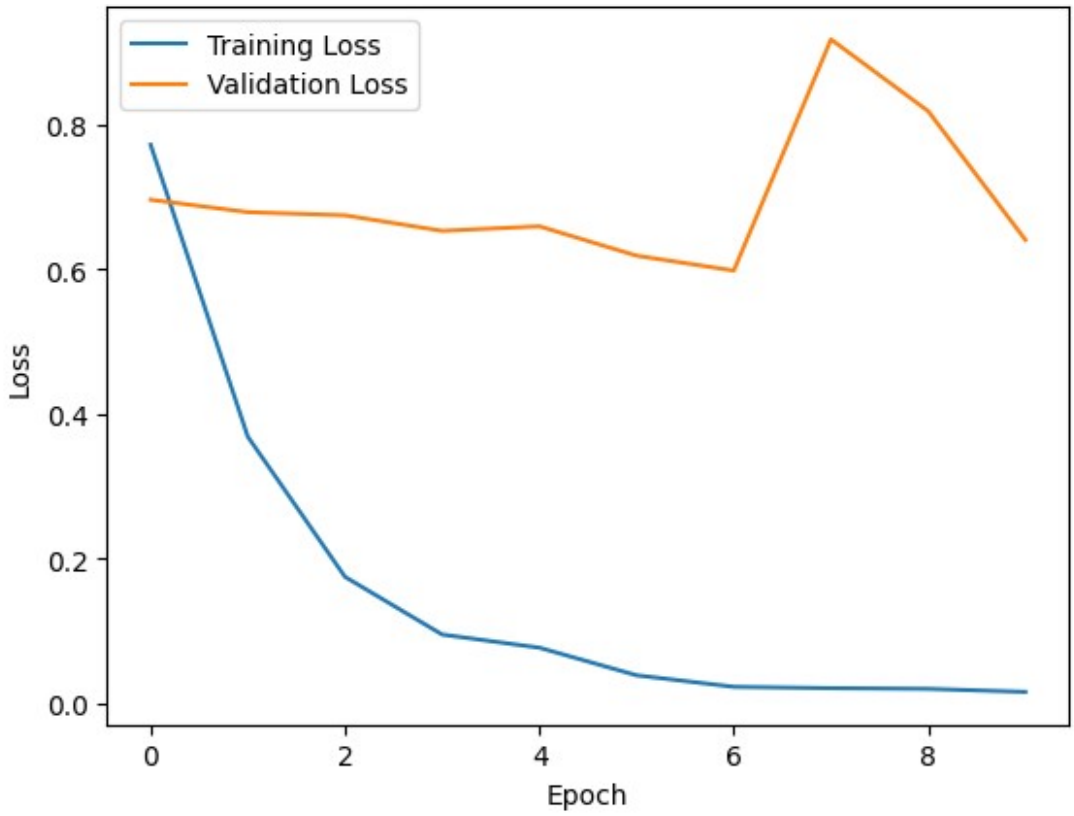
```
print("Accuracy : ")
print(" ")
plt.plot(rnn_history_500.history['accuracy'], label='Training
Accuracy')
plt.plot(rnn_history_500.history['val_accuracy'], label='Validation
Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()

# Plot training and validation loss
print(" ")
print("Loss : ")
print(" ")
plt.plot(rnn_history_500.history['loss'], label='Training Loss')
plt.plot(rnn_history_500.history['val_loss'], label='Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

```
Epoch 1/10
16/16 [==============================] - 7s 433ms/step - loss: 0.7720
- accuracy: 0.6460 - val_loss: 0.6959 - val_accuracy: 0.5142
Epoch 2/10
16/16 [==============================] - 7s 433ms/step - loss: 0.3687
- accuracy: 0.8320 - val_loss: 0.6788 - val_accuracy: 0.5350
Epoch 3/10
16/16 [==============================] - 6s 421ms/step - loss: 0.1751
- accuracy: 0.9380 - val_loss: 0.6744 - val_accuracy: 0.5456
Epoch 4/10
16/16 [==============================] - 6s 392ms/step - loss: 0.0954
- accuracy: 0.9800 - val_loss: 0.6530 - val_accuracy: 0.6696
Epoch 5/10
16/16 [==============================] - 5s 306ms/step - loss: 0.0773
- accuracy: 0.9780 - val_loss: 0.6592 - val_accuracy: 0.5582
Epoch 6/10
16/16 [==============================] - 5s 305ms/step - loss: 0.0391
- accuracy: 0.9900 - val_loss: 0.6186 - val_accuracy: 0.6848
Epoch 7/10
16/16 [==============================] - 5s 306ms/step - loss: 0.0232
- accuracy: 0.9980 - val_loss: 0.5981 - val_accuracy: 0.6938
Epoch 8/10
16/16 [==============================] - 7s 435ms/step - loss: 0.0214
- accuracy: 0.9940 - val_loss: 0.9175 - val_accuracy: 0.5206
Epoch 9/10
16/16 [==============================] - 7s 444ms/step - loss: 0.0205
- accuracy: 0.9960 - val_loss: 0.8183 - val_accuracy: 0.5110
Epoch 10/10
```

```
16/16 [==============================] - 5s 341ms/step - loss: 0.0162
- accuracy: 0.9980 - val_loss: 0.6405 - val_accuracy: 0.6538
157/157 [==============================] - 4s 23ms/step - loss: 0.6405
- accuracy: 0.6538
Test Loss :  0.6405353546142578
Test Accuracy :  0.6538000106811523

Perfomance of RNN Model for 500 Training Samples :

Accuracy :
```




```
Loss :
```

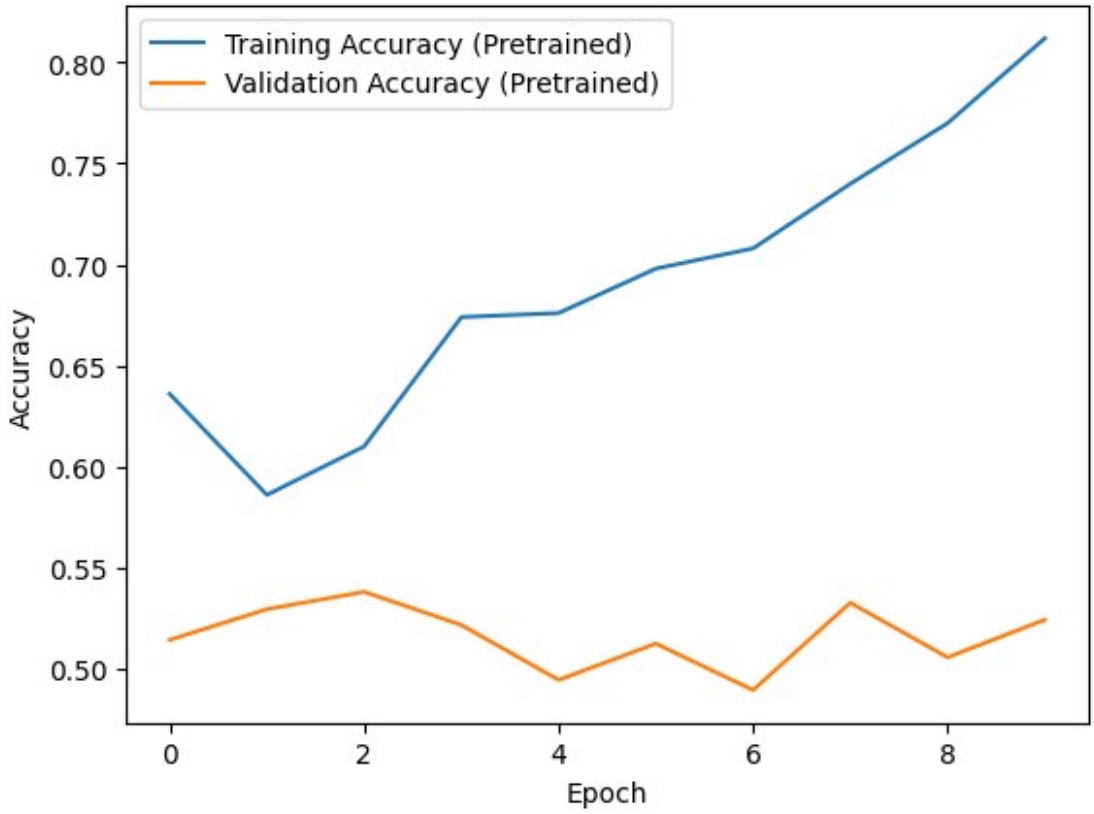



```
# Train the RNN model with pretrained embeddings
rnn_model_pretrained_500 = rnn_model_pretrained
rnn_history_pretrained_500 =
rnn_model_pretrained_500.fit(train_data_500, train_labels_500,
epochs=10, batch_size=32, validation_data=(val_data, val_labels))

# Evaluate the model on the test data
test_loss_pre_trained_rnn500, test_accuracy_pre_trained_rnn500 =
rnn_model_pretrained_500.evaluate(test_data, test_labels)

print("Test Loss : ", test_loss_pre_trained_rnn500)
print("Test Accuracy : ", test_accuracy_pre_trained_rnn500)

# Plot training and validation accuracy
print("Perfomance of Pre Trained RNN Model for 500 Training Samples :
")
print(" ")
print("Accuracy : ")
print(" ")
plt.plot(rnn_history_pretrained_500.history['accuracy'],
label='Training Accuracy (Pretrained)')
plt.plot(rnn_history_pretrained_500.history['val_accuracy'],
label='Validation Accuracy (Pretrained)')
plt.xlabel('Epoch')
```

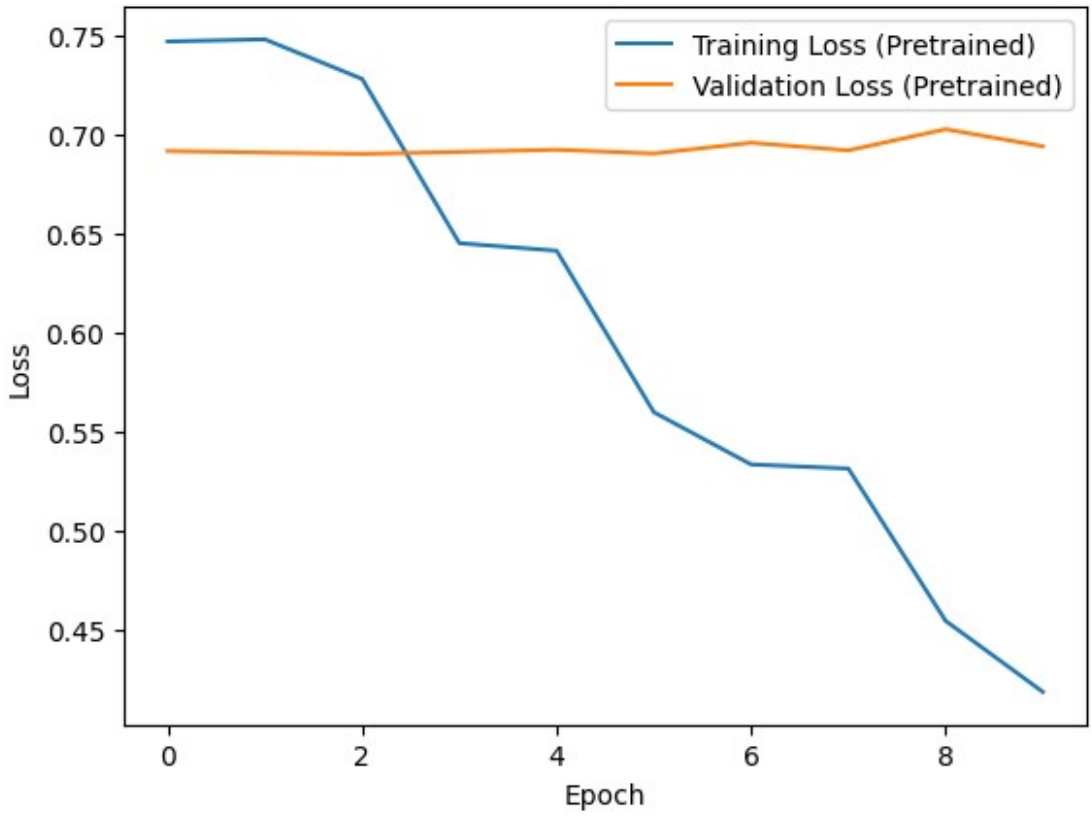
```
plt.ylabel('Accuracy')
plt.legend()
plt.show()

print(" ")
print("Loss : ")
print(" ")
# Plot training and validation loss
plt.plot(rnn_history_pretrained_500.history['loss'], label='Training
Loss (Pretrained)')
plt.plot(rnn_history_pretrained_500.history['val_loss'],
label='Validation Loss (Pretrained)')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

```
Epoch 1/10
16/16 [==============================] - 5s 338ms/step - loss: 0.7469
- accuracy: 0.6360 - val_loss: 0.6916 - val_accuracy: 0.5142
Epoch 2/10
16/16 [==============================] - 5s 340ms/step - loss: 0.7480
- accuracy: 0.5860 - val_loss: 0.6908 - val_accuracy: 0.5294
Epoch 3/10
16/16 [==============================] - 5s 344ms/step - loss: 0.7280
- accuracy: 0.6100 - val_loss: 0.6901 - val_accuracy: 0.5380
Epoch 4/10
16/16 [==============================] - 5s 334ms/step - loss: 0.6450
- accuracy: 0.6740 - val_loss: 0.6911 - val_accuracy: 0.5216
Epoch 5/10
16/16 [==============================] - 5s 331ms/step - loss: 0.6412
- accuracy: 0.6760 - val_loss: 0.6922 - val_accuracy: 0.4944
Epoch 6/10
16/16 [==============================] - 5s 325ms/step - loss: 0.5596
- accuracy: 0.6980 - val_loss: 0.6903 - val_accuracy: 0.5124
Epoch 7/10
16/16 [==============================] - 5s 325ms/step - loss: 0.5333
- accuracy: 0.7080 - val_loss: 0.6957 - val_accuracy: 0.4894
Epoch 8/10
16/16 [==============================] - 5s 326ms/step - loss: 0.5312
- accuracy: 0.7400 - val_loss: 0.6919 - val_accuracy: 0.5326
Epoch 9/10
16/16 [==============================] - 5s 325ms/step - loss: 0.4543
- accuracy: 0.7700 - val_loss: 0.7026 - val_accuracy: 0.5056
Epoch 10/10
16/16 [==============================] - 5s 342ms/step - loss: 0.4184
- accuracy: 0.8120 - val_loss: 0.6940 - val_accuracy: 0.5242
157/157 [==============================] - 4s 25ms/step - loss: 0.6940
- accuracy: 0.5242
Test Loss :  0.6939730644226074
```

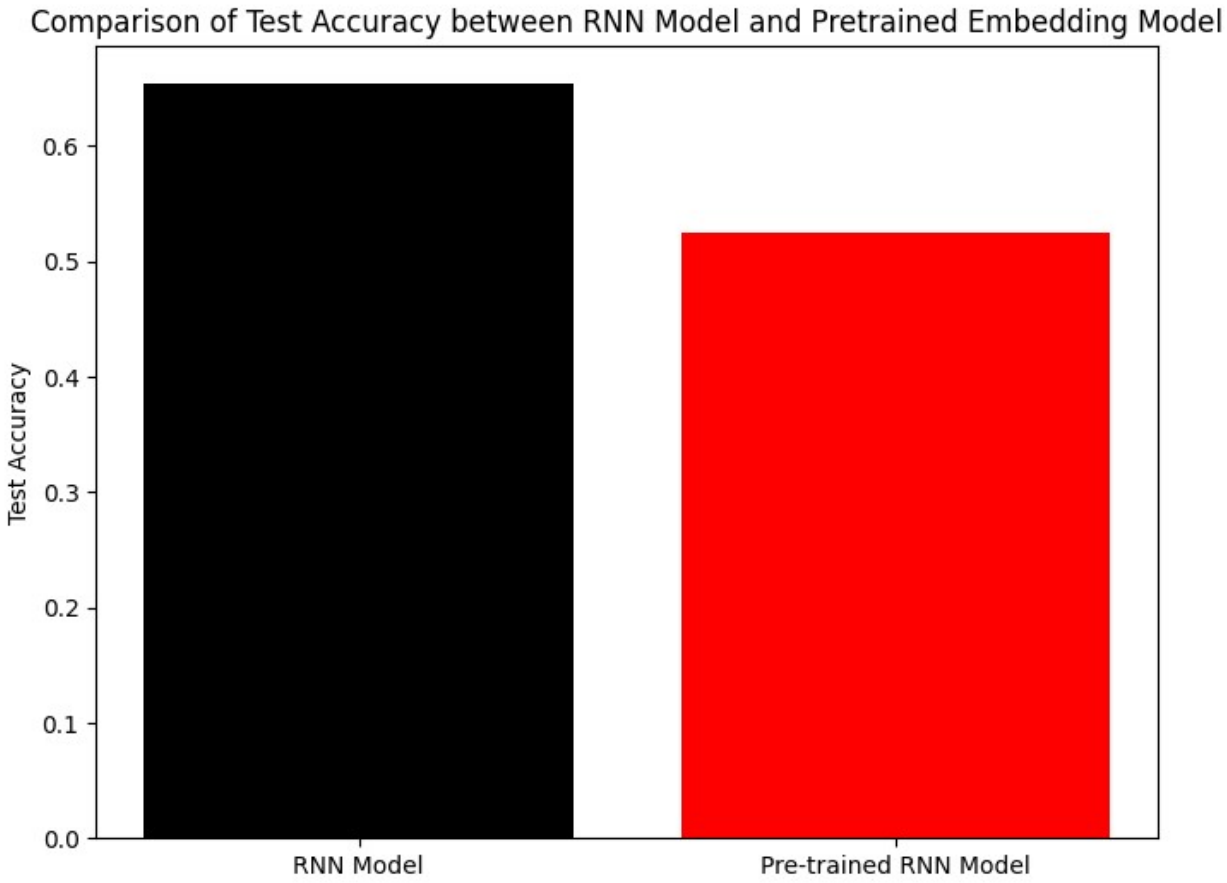
```
Test Accuracy :  0.5242000222206116
Perfomance of Pre Trained RNN Model for 500 Training Samples :

Accuracy :
```



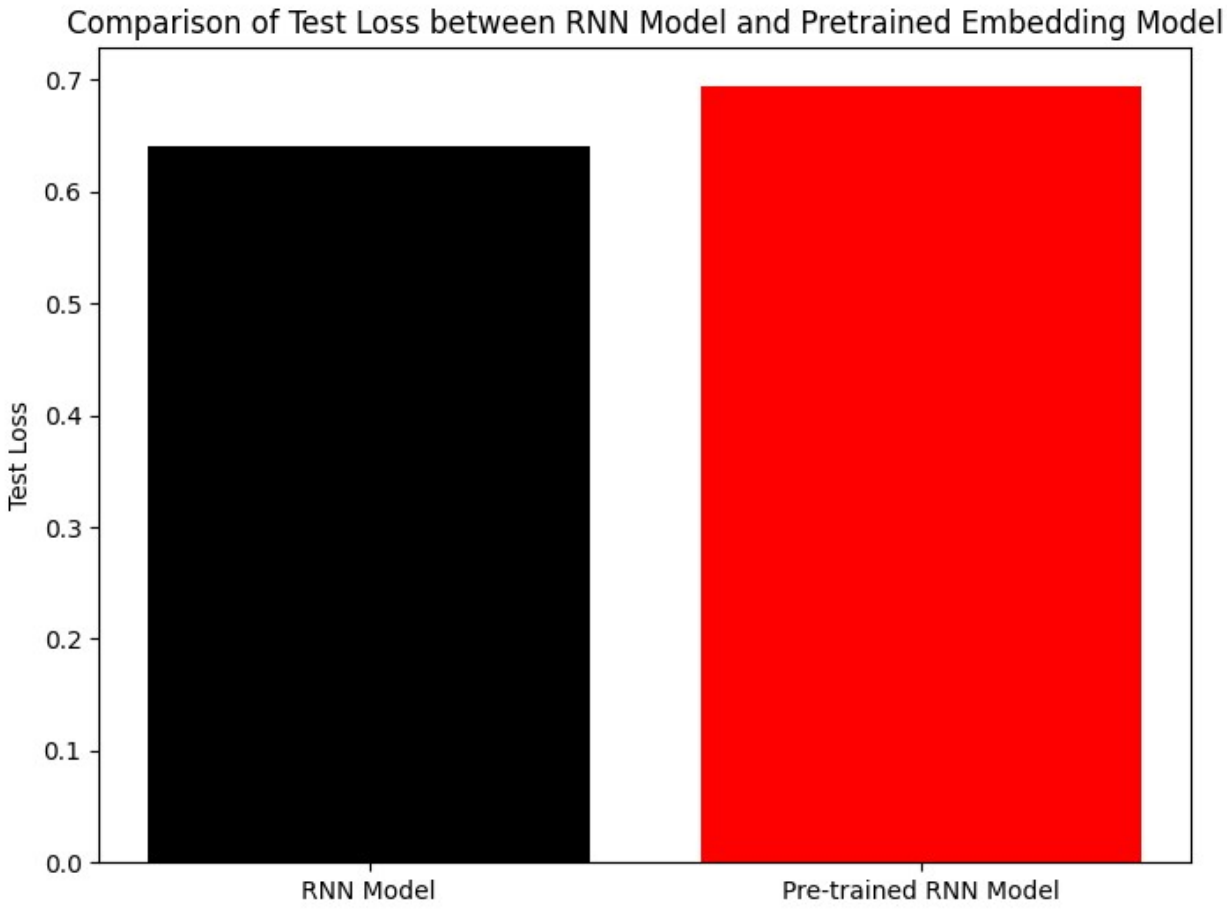

```
Loss :
```




```
# Model names for labeling
model_names = ['RNN Model', 'Pre-trained RNN Model']

# Plot comparison graph
plt.figure(figsize=(8, 6))
plt.bar(model_names, [test_accuracy_rnn500,
test_accuracy_pre_trained_rnn500], color=['black', 'red'])
plt.title('Comparison of Test Accuracy between RNN Model and
Pretrained Embedding Model')
plt.ylabel('Test Accuracy')
plt.show()
```




```
# Model names for labeling
model_names = ['RNN Model', 'Pre-trained RNN Model']

# Plot comparison graph
plt.figure(figsize=(8, 6))
plt.bar(model_names, [test_loss_rnn500, test_loss_pre_trained_rnn500],
color=['black', 'red'])
plt.title('Comparison of Test Loss between RNN Model and Pretrained
Embedding Model')
plt.ylabel('Test Loss')
plt.show()
```

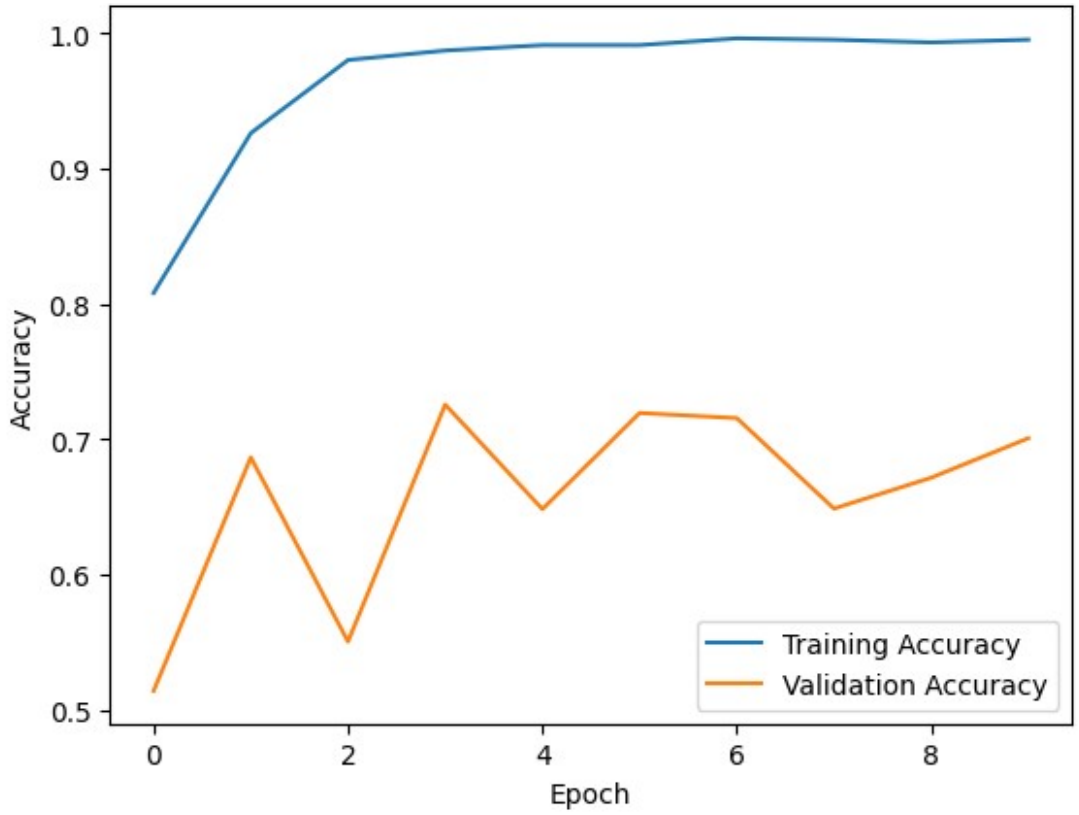



```
# Select the first 1000 samples for training
train_data_1000 = train_data[:1000]
train_labels_1000 = train_labels[:1000]

# Train the RNN model
rnn_model_1000 = rnn_model
rnn_history_1000 = rnn_model_1000.fit(train_data_1000,
train_labels_1000, epochs=10, batch_size=32,
validation_data=(val_data, val_labels))

# Evaluate the model
test_loss_rnn1000, test_accuracy_rnn1000 =
rnn_model_1000.evaluate(test_data, test_labels)

print("Test Loss : ", test_loss_rnn1000)
print("Test Accuracy : ", test_accuracy_rnn1000)


#Model Perfomance Evaluation
print(" ")
print("Perfomance of RNN Model for 1000 Training Samples : ")
print(" ")
```

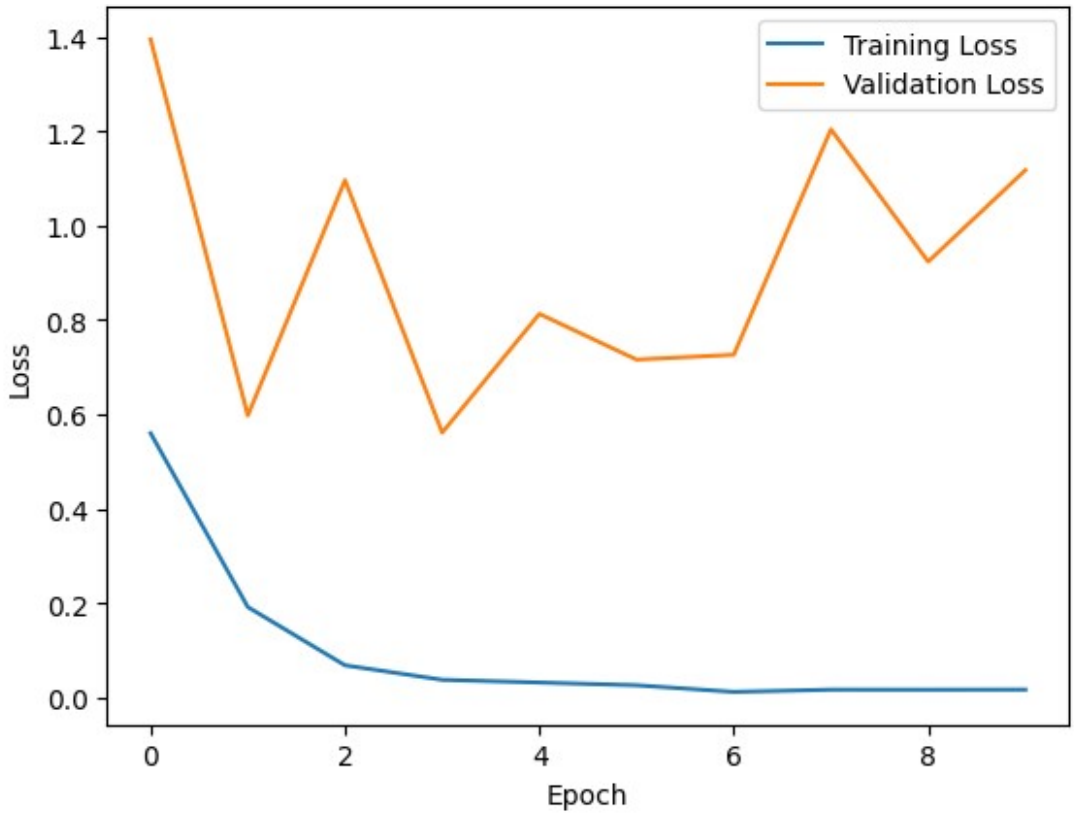
```
# Plot training and validation accuracy
print("Accuracy : ")
print(" ")
plt.plot(rnn_history_1000.history['accuracy'], label='Training
Accuracy')
plt.plot(rnn_history_1000.history['val_accuracy'], label='Validation
Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()

# Plot training and validation loss
print(" ")
print("Loss : ")
print(" ")
plt.plot(rnn_history_1000.history['loss'], label='Training Loss')
plt.plot(rnn_history_1000.history['val_loss'], label='Validation
Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

```
Epoch 1/10
32/32 [==============================] - 6s 195ms/step - loss: 0.5605
- accuracy: 0.8080 - val_loss: 1.3945 - val_accuracy: 0.5142
Epoch 2/10
32/32 [==============================] - 6s 192ms/step - loss: 0.1924
- accuracy: 0.9260 - val_loss: 0.5979 - val_accuracy: 0.6866
Epoch 3/10
32/32 [==============================] - 8s 242ms/step - loss: 0.0690
- accuracy: 0.9800 - val_loss: 1.0963 - val_accuracy: 0.5506
Epoch 4/10
32/32 [==============================] - 7s 233ms/step - loss: 0.0384
- accuracy: 0.9870 - val_loss: 0.5617 - val_accuracy: 0.7256
Epoch 5/10
32/32 [==============================] - 6s 189ms/step - loss: 0.0326
- accuracy: 0.9910 - val_loss: 0.8131 - val_accuracy: 0.6484
Epoch 6/10
32/32 [==============================] - 6s 190ms/step - loss: 0.0268
- accuracy: 0.9910 - val_loss: 0.7163 - val_accuracy: 0.7194
Epoch 7/10
32/32 [==============================] - 6s 190ms/step - loss: 0.0128
- accuracy: 0.9960 - val_loss: 0.7266 - val_accuracy: 0.7156
Epoch 8/10
32/32 [==============================] - 7s 222ms/step - loss: 0.0175
- accuracy: 0.9950 - val_loss: 1.2038 - val_accuracy: 0.6488
Epoch 9/10
32/32 [==============================] - 7s 221ms/step - loss: 0.0173
```

```
- accuracy: 0.9930 - val_loss: 0.9237 - val_accuracy: 0.6716
Epoch 10/10
32/32 [==============================] - 6s 189ms/step - loss: 0.0176
- accuracy: 0.9950 - val_loss: 1.1175 - val_accuracy: 0.7008
157/157 [==============================] - 4s 22ms/step - loss: 1.1175
- accuracy: 0.7008
Test Loss :  1.1175118684768677
Test Accuracy :  0.7008000016212463

Perfomance of RNN Model for 1000 Training Samples :

Accuracy :
```




```
Loss :
```

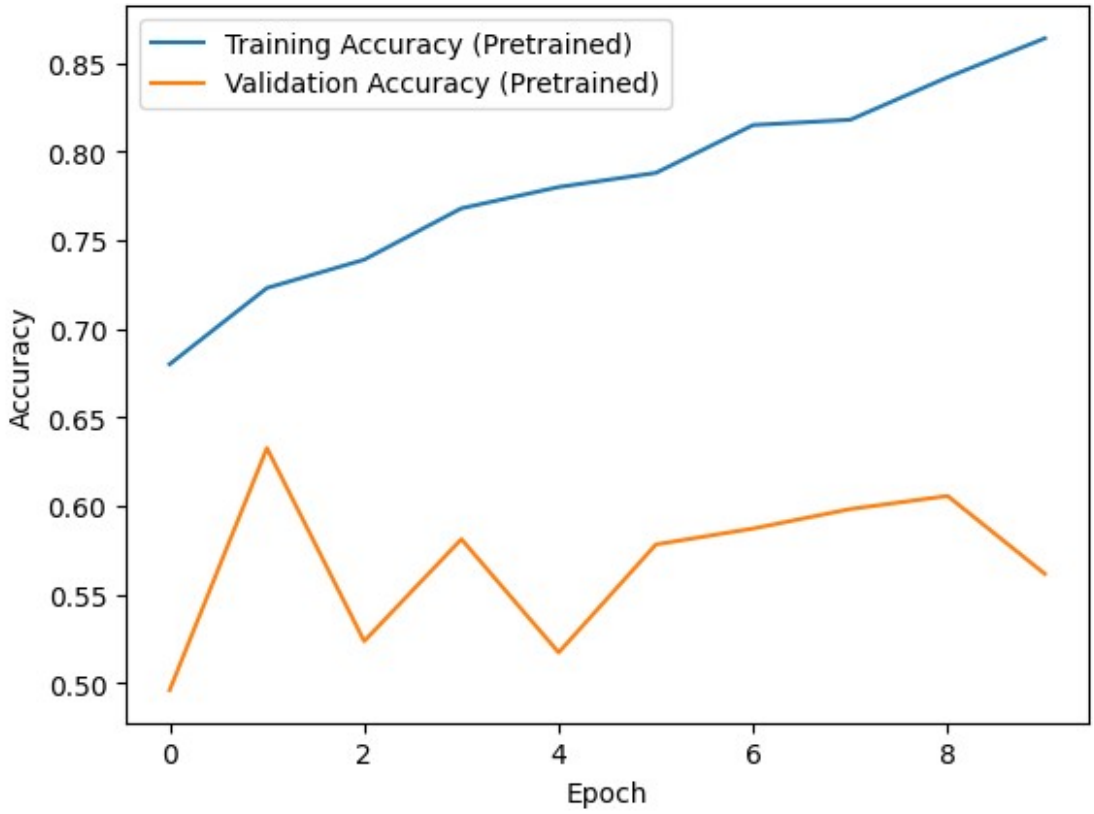



```
# Train the RNN model with pretrained embeddings
rnn_model_pretrained_1000 = rnn_model_pretrained
rnn_history_pretrained_1000 =
rnn_model_pretrained_1000.fit(train_data_1000, train_labels_1000,
epochs=10, batch_size=32, validation_data=(val_data, val_labels))

# Evaluate the model on the test data
test_loss_pre_trained_rnn1000, test_accuracy_pre_trained_rnn1000 =
rnn_model_pretrained_1000.evaluate(test_data, test_labels)

print("Test Loss : ", test_loss_pre_trained_rnn1000)
print("Test Accuracy : ", test_accuracy_pre_trained_rnn1000)

# Plot training and validation accuracy
print("Perfomance of Pre Trained RNN Model for 1000 Training Samples :
")
print(" ")
print("Accuracy : ")
print(" ")
plt.plot(rnn_history_pretrained_1000.history['accuracy'],
label='Training Accuracy (Pretrained)')
plt.plot(rnn_history_pretrained_1000.history['val_accuracy'],
label='Validation Accuracy (Pretrained)')
plt.xlabel('Epoch')
```

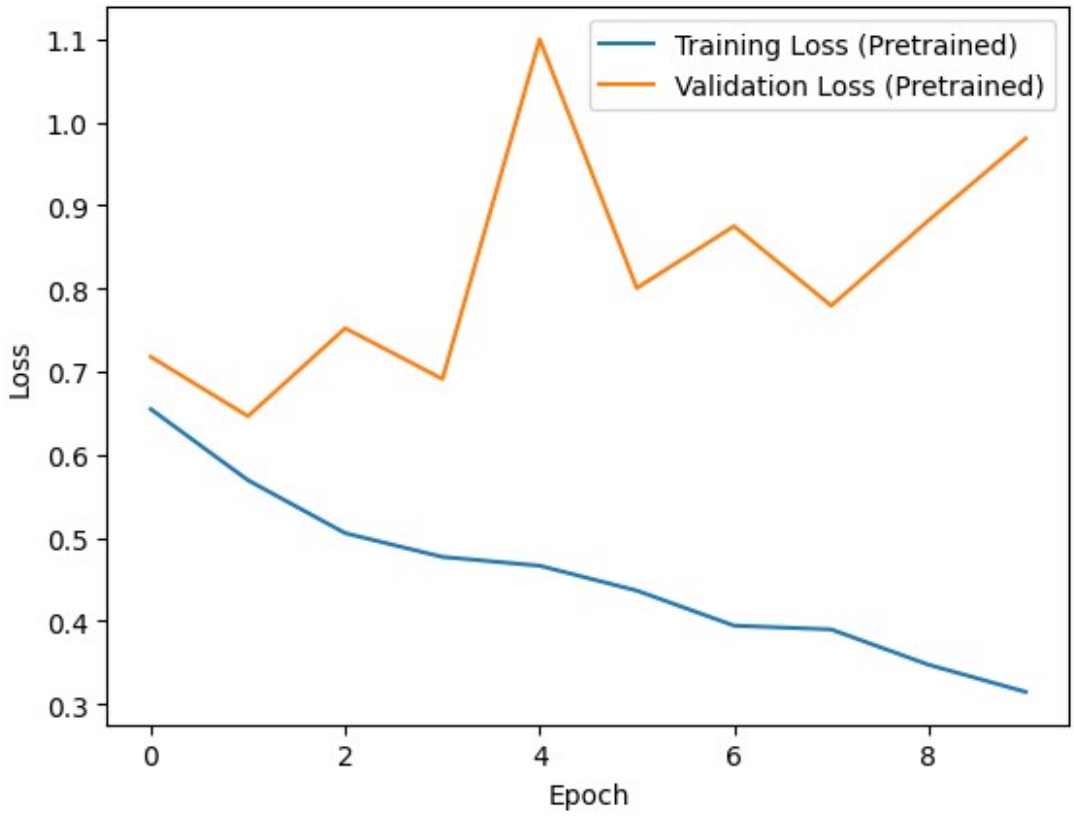
```
plt.ylabel('Accuracy')
plt.legend()
plt.show()

print(" ")
print("Loss : ")
print(" ")
# Plot training and validation loss
plt.plot(rnn_history_pretrained_1000.history['loss'], label='Training
Loss (Pretrained)')
plt.plot(rnn_history_pretrained_1000.history['val_loss'],
label='Validation Loss (Pretrained)')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

```
Epoch 1/10
32/32 [==============================] - 7s 207ms/step - loss: 0.6549
- accuracy: 0.6800 - val_loss: 0.7178 - val_accuracy: 0.4960
Epoch 2/10
32/32 [==============================] - 6s 205ms/step - loss: 0.5694
- accuracy: 0.7230 - val_loss: 0.6464 - val_accuracy: 0.6326
Epoch 3/10
32/32 [==============================] - 7s 209ms/step - loss: 0.5058
- accuracy: 0.7390 - val_loss: 0.7522 - val_accuracy: 0.5236
Epoch 4/10
32/32 [==============================] - 8s 252ms/step - loss: 0.4772
- accuracy: 0.7680 - val_loss: 0.6911 - val_accuracy: 0.5812
Epoch 5/10
32/32 [==============================] - 8s 266ms/step - loss: 0.4666
- accuracy: 0.7800 - val_loss: 1.0996 - val_accuracy: 0.5172
Epoch 6/10
32/32 [==============================] - 8s 256ms/step - loss: 0.4367
- accuracy: 0.7880 - val_loss: 0.8005 - val_accuracy: 0.5782
Epoch 7/10
32/32 [==============================] - 8s 249ms/step - loss: 0.3946
- accuracy: 0.8150 - val_loss: 0.8749 - val_accuracy: 0.5872
Epoch 8/10
32/32 [==============================] - 8s 250ms/step - loss: 0.3900
- accuracy: 0.8180 - val_loss: 0.7793 - val_accuracy: 0.5982
Epoch 9/10
32/32 [==============================] - 8s 261ms/step - loss: 0.3476
- accuracy: 0.8420 - val_loss: 0.8817 - val_accuracy: 0.6056
Epoch 10/10
32/32 [==============================] - 8s 254ms/step - loss: 0.3149
- accuracy: 0.8640 - val_loss: 0.9805 - val_accuracy: 0.5616
157/157 [==============================] - 5s 31ms/step - loss: 0.9805
- accuracy: 0.5616
Test Loss :  0.9805372953414917
```

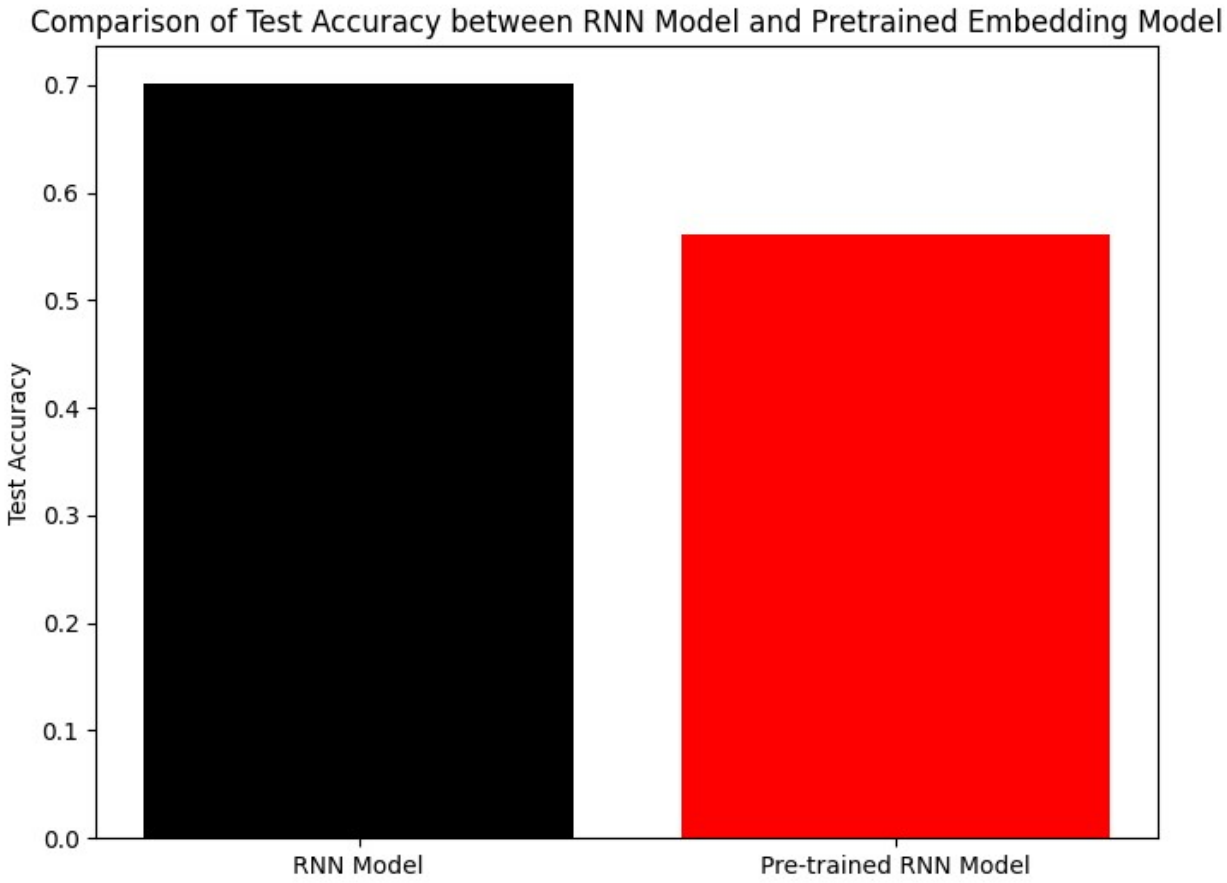
```
Test Accuracy :  0.5616000294685364
Perfomance of Pre Trained RNN Model for 1000 Training Samples :

Accuracy :
```



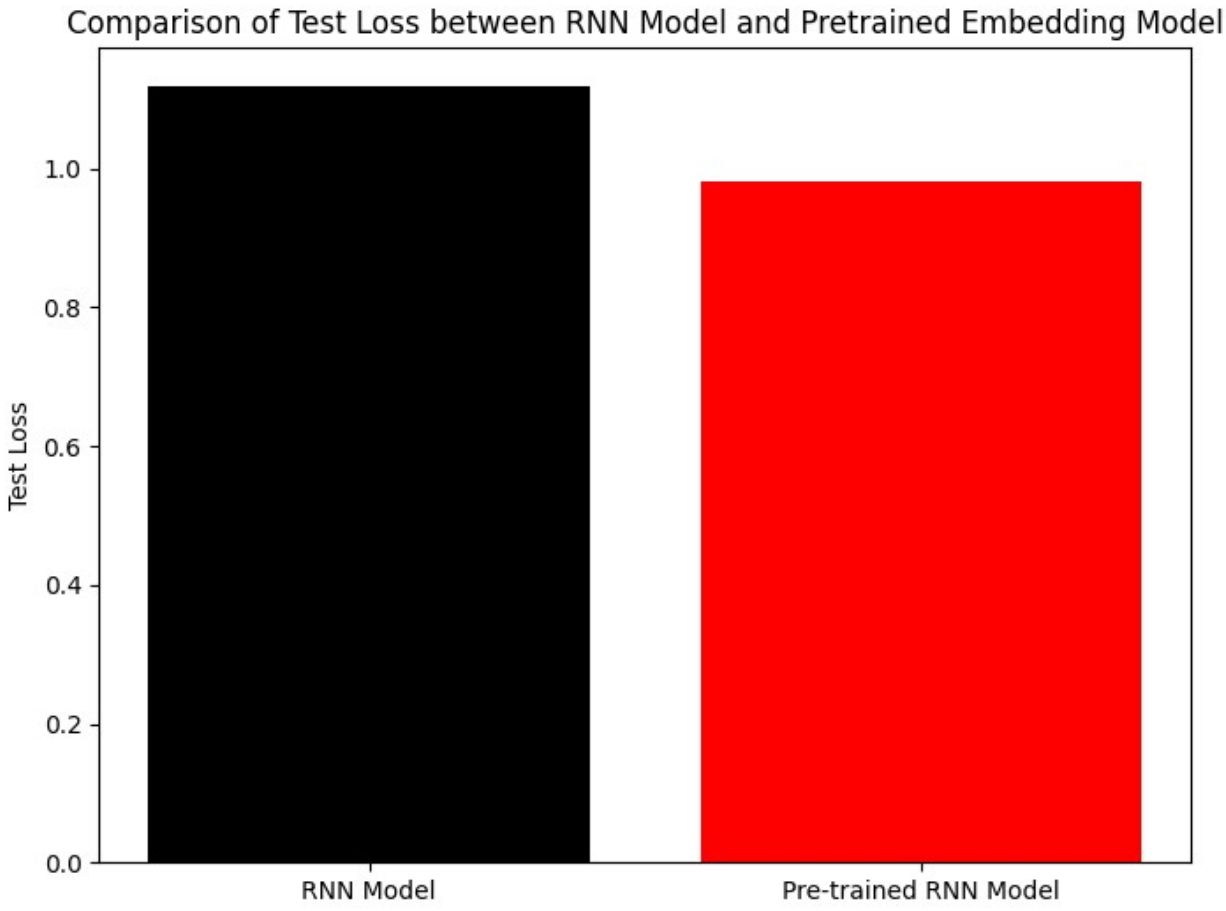

```
Loss :
```




```
# Model names for labeling
model_names = ['RNN Model', 'Pre-trained RNN Model']

# Plot comparison graph
plt.figure(figsize=(8, 6))
plt.bar(model_names, [test_accuracy_rnn1000,
test_accuracy_pre_trained_rnn1000], color=['black', 'red'])
plt.title('Comparison of Test Accuracy between RNN Model and
Pretrained Embedding Model')
plt.ylabel('Test Accuracy')
plt.show()
```




```
# Model names for labeling
model_names = ['RNN Model', 'Pre-trained RNN Model']

# Plot comparison graph
plt.figure(figsize=(8, 6))
plt.bar(model_names, [test_loss_rnn1000,
test_loss_pre_trained_rnn1000], color=['black', 'red'])
plt.title('Comparison of Test Loss between RNN Model and Pretrained
Embedding Model')
plt.ylabel('Test Loss')
plt.show()
```


Comparison of Test Loss between RNN Model and Pretrained Embedding Model
Test Loss
1.0
0.8
0.6
0.4
0.2
0.0
RNN Model
Pre-trained RNN Model