# Performance Comparison of LSTM and BiLSTM Models for Health Tweet Classification

CSC4093- Neural Network and Deep Learning

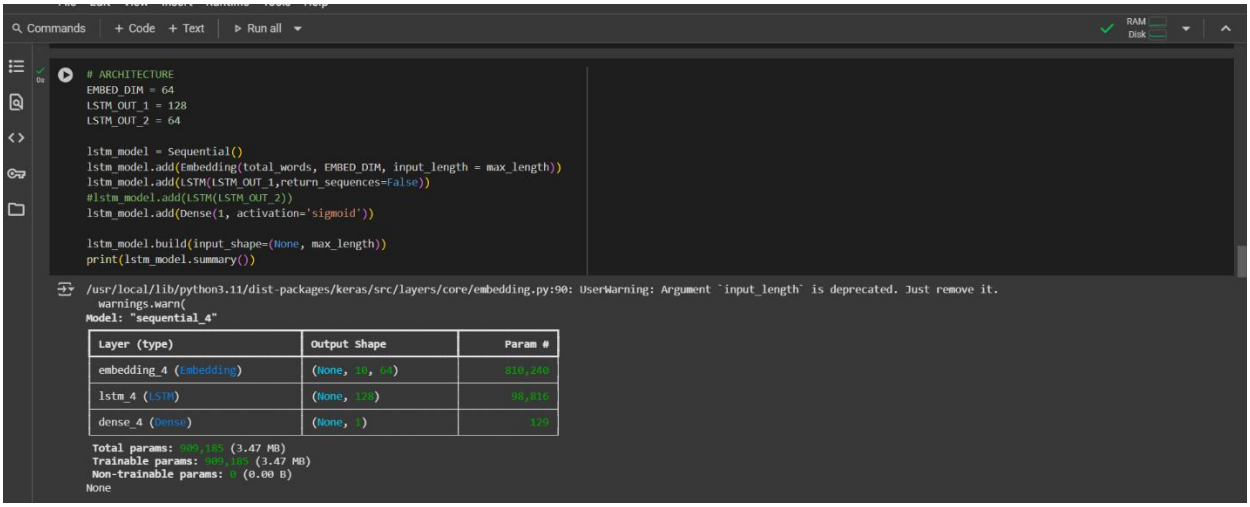
M.SAYANTHINY

S/19/948

1.

screenshots of the outputs.

LSTM

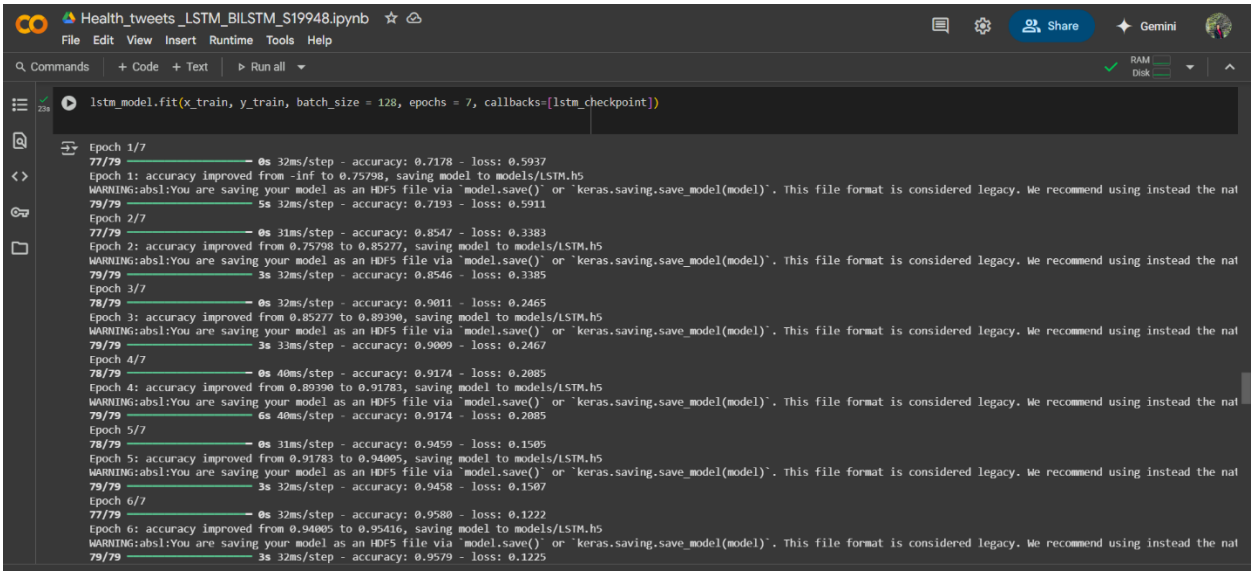


```
# ARCHITECTURE
EMBED_DIM = 64
LSTM_OUT_1 = 128
LSTM_OUT_2 = 64

lstm_model = Sequential()
lstm_model.add(Embedding(total_words, EMBED_DIM, input_length = max_length))
lstm_model.add(LSTM(LSTM_OUT_1,return_sequences=False))
#lstm_model.add(LSTM(LSTM_OUT_2))
lstm_model.add(Dense(1, activation='sigmoid'))

lstm_model.build(input_shape=(None, max_length))
print(lstm_model.summary())
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument `input_length` is deprecated. Just remove it.
  warnings.warn(
Model: "sequential_4"
```

| Layer (type) | Output Shape | Param # |
|---|---|---|
| embedding_4 (Embedding) | (None, 10, 64) | 810,240 |
| lstm_4 (LSTM) | (None, 128) | 98,816 |
| dense_4 (Dense) | (None, 1) | 129 |

```
Total params: 909,185 (3.47 MB)
Trainable params: 909,185 (3.47 MB)
Non-trainable params: 0 (0.00 B)
None
```



```
lstm_model.fit(x_train, y_train, batch_size = 128, epochs = 7, callbacks=[lstm_checkpoint])
```

```
Epoch 1/7
77/79 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - accuracy: 0.7178 - loss: 0.5937
Epoch 1: accuracy improved from -inf to 0.75798, saving model to models/LSTM.h5
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the nat
79/79 ━━━━━━━━━━━━━━━━━━━━ 5s 32ms/step - accuracy: 0.7193 - loss: 0.5911
Epoch 2/7
77/79 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 0.8547 - loss: 0.3383
Epoch 2: accuracy improved from 0.75798 to 0.85277, saving model to models/LSTM.h5
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the nat
79/79 ━━━━━━━━━━━━━━━━━━━━ 3s 32ms/step - accuracy: 0.8546 - loss: 0.3385
Epoch 3/7
78/79 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - accuracy: 0.9011 - loss: 0.2465
Epoch 3: accuracy improved from 0.85277 to 0.89390, saving model to models/LSTM.h5
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the nat
79/79 ━━━━━━━━━━━━━━━━━━━━ 3s 33ms/step - accuracy: 0.9009 - loss: 0.2467
Epoch 4/7
78/79 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step - accuracy: 0.9174 - loss: 0.2085
Epoch 4: accuracy improved from 0.89390 to 0.91783, saving model to models/LSTM.h5
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the nat
79/79 ━━━━━━━━━━━━━━━━━━━━ 6s 40ms/step - accuracy: 0.9174 - loss: 0.2085
Epoch 5/7
78/79 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 0.9459 - loss: 0.1505
Epoch 5: accuracy improved from 0.91783 to 0.94005, saving model to models/LSTM.h5
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the nat
79/79 ━━━━━━━━━━━━━━━━━━━━ 3s 32ms/step - accuracy: 0.9458 - loss: 0.1507
Epoch 6/7
77/79 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - accuracy: 0.9580 - loss: 0.1222
Epoch 6: accuracy improved from 0.94005 to 0.95416, saving model to models/LSTM.h5
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the nat
79/79 ━━━━━━━━━━━━━━━━━━━━ 3s 32ms/step - accuracy: 0.9579 - loss: 0.1225
```

```
77/79 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - accuracy: 0.9580 - loss: 0.1222
Epoch 6: accuracy improved from 0.94005 to 0.95416, saving model to models/LSTM.h5
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the nat
79/79 ━━━━━━━━━━━━━━━━━━━━ 3s 32ms/step - accuracy: 0.9579 - loss: 0.1225
Epoch 7/7
78/79 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - accuracy: 0.9681 - loss: 0.0951
Epoch 7: accuracy improved from 0.95416 to 0.96147, saving model to models/LSTM.h5
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the nat
79/79 ━━━━━━━━━━━━━━━━━━━━ 3s 33ms/step - accuracy: 0.9679 - loss: 0.0953
<keras.src.callbacks.history.History at 0x78473fc62410>
```

```
[50] # LSTM Training Accuracy
lstm_train_pred = lstm_model.predict(x=x_train)
lstm_train_y_pred = (lstm_train_pred >= 0.5) * 1

true = 0
for i, y in enumerate(y_train):
    if y == lstm_train_y_pred[i]:
        true += 1

print('LSTM Training Accuracy')
print('Correct Prediction:{}'.format(true))
print('Wrong Prediction:{}'.format(len(lstm_train_y_pred) - true))
print('Accuracy: {:.2f}%'.format(true/ len(lstm_train_y_pred) * 100))
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step
LSTM Training Accuracy
Correct Prediction:9744
Wrong Prediction:247
Accuracy: 97.53%
```

Health_tweets _LSTM_BILSTM_S19948.ipynb

File Edit View Insert Runtime Tools Help

Share Gemini

Commands + Code + Text Run all

```
Correct Prediction: 9786
Wrong Prediction: 205
Accuracy: 97.95%
```

```
[ ] #lstm_Model testing
```

```
lstm_pred = lstm_model.predict(x=x_test)
lstm_y_pred = (lstm_pred >= 0.5) * 1

#y_pred = model.predict(x_test)

true = 0
for i, y in enumerate(y_test):
    if y == lstm_y_pred[i]:
        true += 1


print('Correct Prediction: {}'.format(true))
print('Wrong Prediction: {}'.format(len(lstm_y_pred) - true))
print('Accuracy: {}'.format(true/len(lstm_y_pred)*100))
```

```
105/105 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step
Correct Prediction: 2652
Wrong Prediction: 679
Accuracy: 79.61573101170819
```

## BILSTM

Health_tweets _LSTM_BILSTM_S19948.ipynb

File Edit View Insert Runtime Tools Help

Share Gemini

Commands + Code + Text Run all

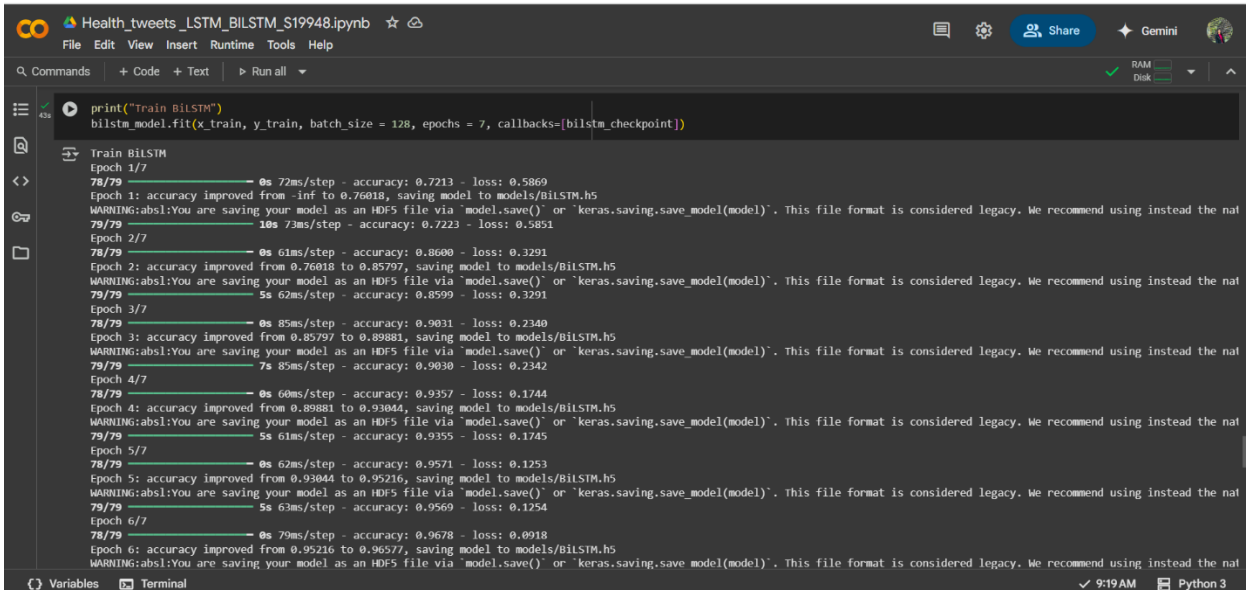
```
[ ] Start coding or generate with AI.
```

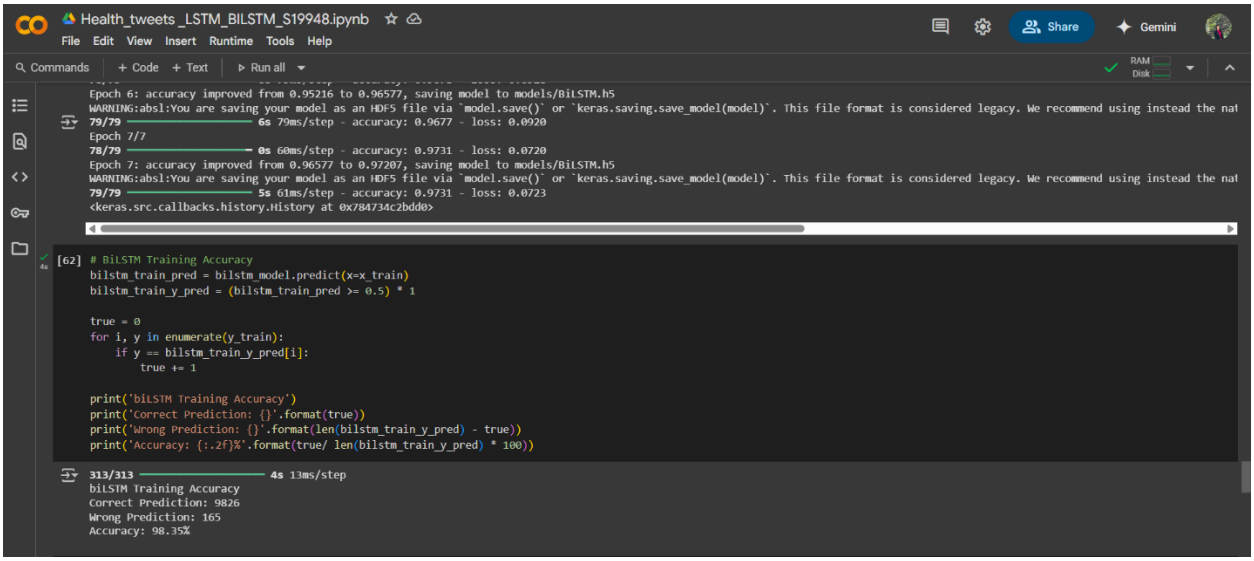
```
# ARCHITECTURE
EMBED_DIM = 64
biLSTM_OUT_1 = 128
#LSTM_OUT_2 = 64

bilstm_model= Sequential()
bilstm_model.add(Embedding(total_words, EMBED_DIM, input_length = max_length))
bilstm_model.add(Bidirectional(LSTM(biLSTM_OUT_1,return_sequences=False)))
#bilstm_model.add(Bidirectional(LSTM(LSTM_OUT_2)))
bilstm_model.add(Dense(1, activation='sigmoid'))

bilstm_model.build(input_shape=(None, max_length))
print(bilstm_model.summary())
```

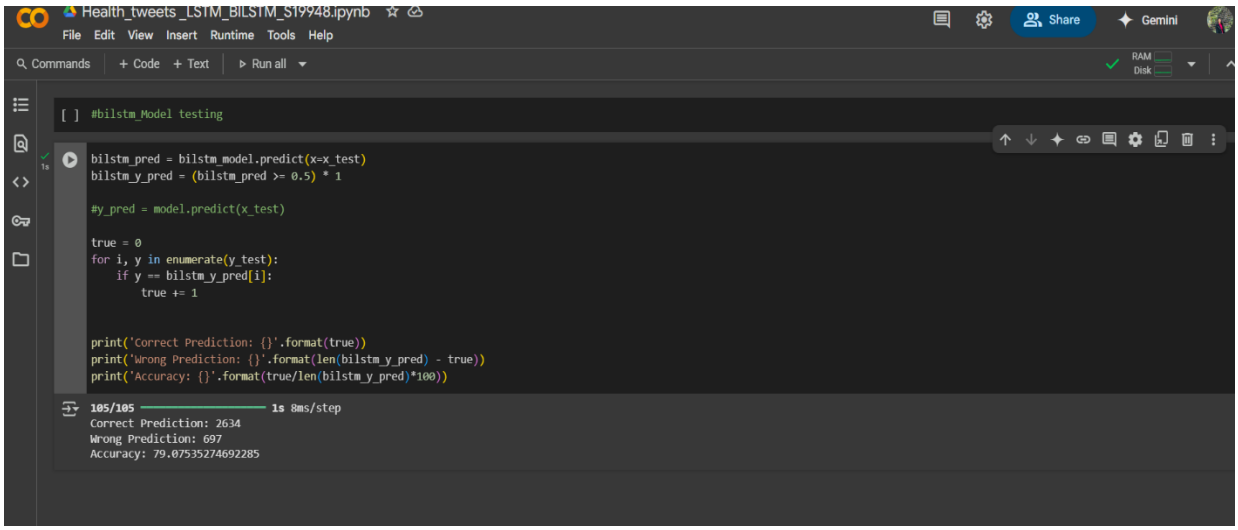
```
Model: "sequential_7"
```

| Layer (type) | Output Shape | Param # |
|---|---|---|
| embedding_7 (Embedding) | (None, 10, 64) | 810,240 |
| bidirectional_4 (Bidirectional) | (None, 256) | 197,632 |
| dense_7 (Dense) | (None, 1) | 257 |

```
Total params: 1,008,129 (3.85 MB)
Trainable params: 1,008,129 (3.85 MB)
Non-trainable params: 0 (0.00 B)
None
```


Health_tweets _LSTM_BILSTM_S19948.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Run all
RAM
Disk
print("Train BiLSTM")
bilstm_model.fit(x_train, y_train, batch_size = 128, epochs = 7, callbacks=[bilstm_checkpoint])
Train BiLSTM
Epoch 1/7
78/79 ━━━━━━━━━━━━━━━━━━━━ 0s 72ms/step - accuracy: 0.7213 - loss: 0.5869
Epoch 1: accuracy improved from -inf to 0.76018, saving model to models/BiLSTM.h5
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the nat
79/79 ━━━━━━━━━━━━━━━━━━━━ 10s 73ms/step - accuracy: 0.7223 - loss: 0.5851
Epoch 2/7
78/79 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - accuracy: 0.8600 - loss: 0.3291
Epoch 2: accuracy improved from 0.76018 to 0.85797, saving model to models/BiLSTM.h5
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the nat
79/79 ━━━━━━━━━━━━━━━━━━━━ 5s 62ms/step - accuracy: 0.8599 - loss: 0.3291
Epoch 3/7
78/79 ━━━━━━━━━━━━━━━━━━━━ 0s 85ms/step - accuracy: 0.9031 - loss: 0.2340
Epoch 3: accuracy improved from 0.85797 to 0.89881, saving model to models/BiLSTM.h5
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the nat
79/79 ━━━━━━━━━━━━━━━━━━━━ 7s 85ms/step - accuracy: 0.9030 - loss: 0.2342
Epoch 4/7
78/79 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step - accuracy: 0.9357 - loss: 0.1744
Epoch 4: accuracy improved from 0.89881 to 0.93044, saving model to models/BiLSTM.h5
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the nat
79/79 ━━━━━━━━━━━━━━━━━━━━ 5s 61ms/step - accuracy: 0.9355 - loss: 0.1745
Epoch 5/7
78/79 ━━━━━━━━━━━━━━━━━━━━ 0s 62ms/step - accuracy: 0.9571 - loss: 0.1253
Epoch 5: accuracy improved from 0.93044 to 0.95216, saving model to models/BiLSTM.h5
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the nat
79/79 ━━━━━━━━━━━━━━━━━━━━ 5s 63ms/step - accuracy: 0.9569 - loss: 0.1254
Epoch 6/7
78/79 ━━━━━━━━━━━━━━━━━━━━ 0s 79ms/step - accuracy: 0.9678 - loss: 0.0918
Epoch 6: accuracy improved from 0.95216 to 0.96577, saving model to models/BiLSTM.h5
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the nat
Variables
Terminal
9:19 AM
Python 3


Health_tweets _LSTM_BILSTM_S19948.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Run all
RAM
Disk
Epoch 6: accuracy improved from 0.95216 to 0.96577, saving model to models/BiLSTM.h5
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the nat
79/79 ━━━━━━━━━━━━━━━━━━━━ 6s 79ms/step - accuracy: 0.9677 - loss: 0.0920
Epoch 7/7
78/79 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step - accuracy: 0.9731 - loss: 0.0720
Epoch 7: accuracy improved from 0.96577 to 0.97207, saving model to models/BiLSTM.h5
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the nat
79/79 ━━━━━━━━━━━━━━━━━━━━ 5s 61ms/step - accuracy: 0.9731 - loss: 0.0723
<keras.src.callbacks.history.History at 0x784734c2bdd0>
[62] # BiLSTM Training Accuracy
bilstm_train_pred = bilstm_model.predict(x=x_train)
bilstm_train_y_pred = (bilstm_train_pred >= 0.5) * 1
true = 0
for i, y in enumerate(y_train):
if y == bilstm_train_y_pred[i]:
true += 1
print('biLSTM Training Accuracy')
print('Correct Prediction: {}'.format(true))
print('Wrong Prediction: {}'.format(len(bilstm_train_y_pred) - true))
print('Accuracy: {:.2f}%'.format(true/ len(bilstm_train_y_pred) * 100))
313/313 ━━━━━━━━━━━━━━━━━━━━ 4s 13ms/step
biLSTM Training Accuracy
Correct Prediction: 9826
Wrong Prediction: 165
Accuracy: 98.35%


Health_tweets _LSTM_BILSTM_S19948.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Run all
RAM
Disk
[ ] #bilstm_Model testing
bilstm_pred = bilstm_model.predict(x=x_test)
bilstm_y_pred = (bilstm_pred >= 0.5) * 1
#y_pred = model.predict(x_test)
true = 0
for i, y in enumerate(y_test):
if y == bilstm_y_pred[i]:
true += 1
print('Correct Prediction: {}'.format(true))
print('Wrong Prediction: {}'.format(len(bilstm_y_pred) - true))
print('Accuracy: {}'.format(true/len(bilstm_y_pred)*100))
105/105 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step
Correct Prediction: 2634
Wrong Prediction: 697
Accuracy: 79.07535274692285

2.

This project focuses on building and comparing two types of recurrent neural network (RNN) models Long Short-Term Memory (LSTM) and Bidirectional LSTM (BiLSTM) to classify tweets as personal health mentions or non-personal mentions.

**a.**

The following table summarizes the performance metrics of both the LSTM and BiLSTM models used for the classification of personal health mention tweets:

| Model | Training Accuracy (%) | Testing Accuracy (%) |
|---|---|---|
| LSTM | **97.53 %**<br>correct: 9744 \| incorrect: 247 | **79.61 %**<br>correct: 2652 \| incorrect: 679 |
| | | |
| BILSTM | **98.35 %**<br>correct: 9826 \| incorrect: 165 | **79.07 %**<br>correct: 2634 \| incorrect: 697 |

**b.**

In this experiment, I create two sequential models LSTM and BILSTM, both models were built using the same architecture parameters

- Embedding size: 64
- LSTM/BiLSTM output size: 128

However, the total number of trainable parameters differed between the two models:

- LSTM model: 909,185 parameters (3.47 MB)
- BiLSTM model: 1,008,129 parameters (3.85 MB)

The BiLSTM model has a higher number of parameters because it processes the input sequence in both forward and backward directions, effectively doubling the number of bidirectional weights compared to the LSTM layer in the model. Both models used same data and,

- Optimizer: Adam
- Loss function: Binary Cross entropy
- Metric: Accuracy
- Training settings: batch size = 128, epochs = 7, callbacks = [lstm_checkpoint]

Based on this, BiLSTM model got **98.35 %** training accuracy it is slightly higher than LSTM model training accuracy **97.53 %.** Because BiLSTM learned the training data better via the its bidirectional context processing.

While Testing accuracy Both models perform almost equally on the test set, as the performance gap is less than 1%. Although LSTM (79.61 %) slightly outperforms BiLSTM (79.07 %) on test accuracy, his suggests that BiLSTM may have started overfitting the training data and failed to generalize as well on unseen data. the difference is not practically significant. So, neither model can be confidently said to outperform the other in terms of generalization ability for this task.

In general, BiLSTM had more parameters, which increases its model complexity and capacity to learn detailed patterns from the data because they capture both past and future contexts of a sequence. However, in this task BILSTM model did not performed better then LSTM on the testing data. For this have some possible reasons,

- BILSTM may have memorized patterns in the training data cause reducing generalization capability.
- The dataset may not have required the additional context provided by the bidirectional layer, meaning the LSTM model was sufficient to capture necessary patterns.
- Both LSTM and BiLSTM models were trained with the same batch size (128) and epochs (7) for fair comparison. However, BiLSTM generally needs more training or fine-tuning to fully utilize its bidirectional features. Using identical settings may have limited its performance advantage over LSTM.

In my perspective, In conclusion, in this task LSTM model achieved slightly higher test accuracy (by approximately 0.6%) compare to BILSTM. This outcome can be created to dataset's nature, model complexity, and the identical training settings applied to both models without specific tuning for BiLSTM.

Therefore, based on the observed performance, the LSTM model is the preferred choice for this task.