```
#@title Licensed under the Apache License, Version 2.0 (the "License")
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
# https://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
```

Licensed under the Apache License, Version 2.0 (the "License");

Single Layer LSTM

```
from __future__ import absolute_import, division, print_function, unicode_literals


import tensorflow_datasets as tfds
import tensorflow as tf
print(tf.__version__)
```

```
2.3.0
```

```
import tensorflow_datasets as tfds
```

Saved successfully!

```
2.3.0
```

```
# Get the data
dataset, info = tfds.load('imdb_reviews/subwords8k', with_info=True, as_supervised=True)
train_dataset, test_dataset = dataset['train'], dataset['test']
```

```
Downloading and preparing dataset imdb_reviews/subwords8k/1.0.0 (download: 80.23 MiB, ge
```

```
Dl Completed...: 100%  1/1 [00:09<00:00, 9.15s/ url]
Dl Size...: 100%  80/80 [00:09<00:00, 8.78 MiB/s]
```

```
#build in encoder
tokenizer = info.features['text'].encoder
```

```
Shuffling and writing examples to /root/tensorflow_datasets/imdb_reviews/subwords8k/1.0
```

```
BUFFER_SIZE = 10000
BATCH_SIZE = 64

train_dataset = train_dataset.shuffle(BUFFER_SIZE)
train_dataset = train_dataset.padded_batch(BATCH_SIZE, tf.compat.v1.data.get_output_shapes(tr
test_dataset = test_dataset.padded_batch(BATCH_SIZE, tf.compat.v1.data.get_output_shapes(test
```

```
Dataset imdb_reviews downloaded and prepared to /root/tensorflow_datasets/imdb_reviews/s
```

```
model = tf.keras.Sequential([
    #built in vocab size of our build in tokenizer
    tf.keras.layers.Embedding(tokenizer.vocab_size, 64),
    #long-short-term memory with 64 units
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64)),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])


model.summary()
```

Saved successfully!

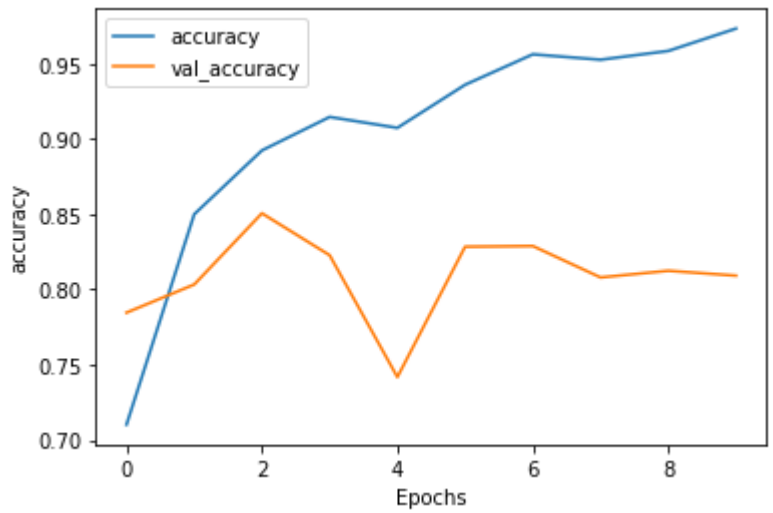
```
_________________________________________________________________
tput Shape              Param #
=================================================================
embedding (Embedding)        (None, None, 64)          523840
_________________________________________________________________
bidirectional (Bidirectional (None, 128)               66048
_________________________________________________________________
dense (Dense)                (None, 64)                8256
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 65
=================================================================
Total params: 598,209
Trainable params: 598,209
Non-trainable params: 0
_________________________________________________________________
```

```
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])


NUM_EPOCHS = 10
history = model.fit(train_dataset, epochs=NUM_EPOCHS, validation_data=test_dataset)
```

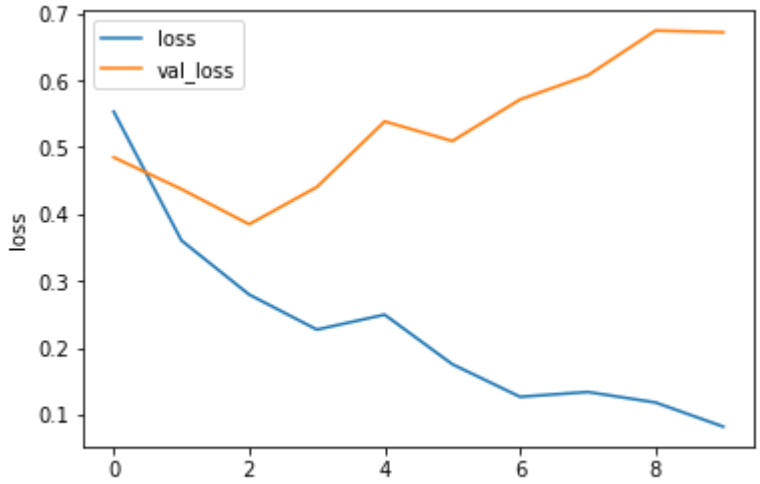
```
Epoch 1/10
391/391 [==============================] - 68s 174ms/step - loss: 0.5532 - accuracy: 0.7
Epoch 2/10
391/391 [==============================] - 69s 176ms/step - loss: 0.3605 - accuracy: 0.8
Epoch 3/10
391/391 [==============================] - 69s 175ms/step - loss: 0.2795 - accuracy: 0.8
Epoch 4/10
391/391 [==============================] - 69s 177ms/step - loss: 0.2272 - accuracy: 0.9
Epoch 5/10
391/391 [==============================] - 70s 178ms/step - loss: 0.2495 - accuracy: 0.9
Epoch 6/10
391/391 [==============================] - 69s 177ms/step - loss: 0.1754 - accuracy: 0.9
Epoch 7/10
391/391 [==============================] - 69s 177ms/step - loss: 0.1266 - accuracy: 0.9
Epoch 8/10
391/391 [==============================] - 69s 176ms/step - loss: 0.1339 - accuracy: 0.9
Epoch 9/10
391/391 [==============================] - 69s 177ms/step - loss: 0.1182 - accuracy: 0.9
Epoch 10/10
391/391 [==============================] - 71s 181ms/step - loss: 0.0821 - accuracy: 0.9
```

```
import matplotlib.pyplot as plt


def plot_graphs(history, string):
  plt.plot(history.history[string])
  plt.plot(history.history['val_'+string])
  plt.xlabel("Epochs")
  plt.ylabel(string)
  plt.legend([string, 'val_'+string])
  plt.show()
```

Saved successfully!




```
plot_graphs(history, 'loss')
```



Saved successfully!