```
#@title Licensed under the Apache License, Version 2.0 (the "Licen
# you may not use this file except in compliance with the Licen
# 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, sof
# distributed under the License is distributed on an "AS IS" BAS
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
# See the License for the specific language governing permissions
# 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__)
```

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

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

```
tokenizer = info.features['text'].encoder
```

```
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(trai
test_dataset = test_dataset.padded_batch(BATCH_SIZE, tf.compat.v1.data.get_output_shapes(test_da
```

```
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(tokenizer.vocab_size, 64),
    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()
```

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


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


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()


plot_graphs(history,  'accuracy')


plot_graphs(history,  'loss')
```