```
#@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");

Open in Colab

```
import numpy as np

import json
import tensorflow as tf

from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

!wget --no-check-certificate \
    https://storage.googleapis.com/laurencemoroney-blog.appspot.com/sarcasm.json \
    -O /tmp/sarcasm.json

vocab_size = 1000
embedding_dim = 16
max_length = 120
trunc_type='post'
padding_type='post'
oov_tok = "<OOV>"
training_size = 20000


with open("/tmp/sarcasm.json", 'r') as f:
    datastore = json.load(f)


sentences = []
labels = []
urls = []
for item in datastore:
    sentences.append(item['headline'])
    labels.append(item['is_sarcastic'])

training_sentences = sentences[0:training_size]
testing_sentences = sentences[training_size:]
training_labels = labels[0:training_size]
training_labels = labels[0:training_size]
testing_labels = labels[training_size:]

tokenizer = Tokenizer(num_words=vocab_size, oov_token=oov_tok)
tokenizer.fit_on_texts(training_sentences)

word_index = tokenizer.word_index

training_sequences = tokenizer.texts_to_sequences(training_sentences)
training_padded = pad_sequences(training_sequences, maxlen=max_length, padding=padding_type,

testing_sequences = tokenizer.texts_to_sequences(testing_sentences)
testing_padded = pad_sequences(testing_sequences, maxlen=max_length, padding=padding_type, tr

model = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, embedding_dim, input_length=max_length),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(32)),
    tf.keras.layers.Dense(24, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
model.summary()

num_epochs = 50
training_padded = np.array(training_padded)
training_labels = np.array(training_labels)
testing_padded = np.array(testing_padded)
testing_labels = np.array(testing_labels)
history = model.fit(training_padded, training_labels, epochs=num_epochs, validation_data=(tes
     Epoch 22/50
     625/625 [==============================] - 36s 58ms/step - loss: 0.1819 - accuracy: 0.92
     Epoch 23/50
     625/625 [==============================] - 36s 58ms/step - loss: 0.1750 - accuracy: 0.92
     Epoch 24/50
     625/625 [==============================] - 36s 58ms/step - loss: 0.1681 - accuracy: 0.93
     Epoch 25/50
     625/625 [==============================] - 36s 58ms/step - loss: 0.1648 - accuracy: 0.93
     Epoch 26/50
     625/625 [==============================] - 36s 58ms/step - loss: 0.1550 - accuracy: 0.93
     Epoch 27/50
     625/625 [==============================] - 36s 58ms/step - loss: 0.1487 - accuracy: 0.94
     Epoch 28/50
     625/625 [==============================] - 36s 58ms/step - loss: 0.1460 - accuracy: 0.94
     Epoch 29/50
     625/625 [==============================] - 36s 58ms/step - loss: 0.1377 - accuracy: 0.94
     Epoch 30/50
     625/625 [==============================] - 36s 58ms/step - loss: 0.1313 - accuracy: 0.94
     Epoch 31/50
     625/625 [==============================] - 36s 58ms/step - loss: 0.1298 - accuracy: 0.94
     Epoch 32/50
     625/625 [==============================] - 36s 58ms/step - loss: 0.1232 - accuracy: 0.95
     Epoch 33/50
     625/625 [==============================] - 36s 58ms/step - loss: 0.1166 - accuracy: 0.95
     Epoch 34/50
     625/625 [==============================] - 36s 58ms/step - loss: 0.1115 - accuracy: 0.95
     Epoch 35/50
     625/625 [==============================] - 36s 58ms/step - loss: 0.1084 - accuracy: 0.95
     Epoch 36/50
     625/625 [==============================] - 36s 58ms/step - loss: 0.0990 - accuracy: 0.96
     Epoch 37/50
     625/625 [==============================] - 36s 58ms/step - loss: 0.1024 - accuracy: 0.95
     Epoch 38/50
     625/625 [==============================] - 36s 58ms/step - loss: 0.1000 - accuracy: 0.96
     Epoch 39/50
     625/625 [==============================] - 36s 58ms/step - loss: 0.0915 - accuracy: 0.96
     Epoch 40/50
     625/625 [==============================] - 36s 58ms/step - loss: 0.0894 - accuracy: 0.96
     Epoch 41/50
     625/625 [==============================] - 36s 58ms/step - loss: 0.0827 - accuracy: 0.96
     Epoch 42/50
     625/625 [==============================] - 36s 58ms/step - loss: 0.0836 - accuracy: 0.96
     Epoch 43/50
     625/625 [==============================] - 36s 58ms/step - loss: 0.0804 - accuracy: 0.96
     Epoch 44/50
     625/625 [==============================] - 36s 58ms/step - loss: 0.0763 - accuracy: 0.96
     Epoch 45/50
     625/625 [==============================] - 36s 58ms/step - loss: 0.0726 - accuracy: 0.97
     Epoch 46/50
     625/625 [==============================] - 36s 57ms/step - loss: 0.0712 - accuracy: 0.97
     Epoch 47/50
     625/625 [==============================] - 36s 57ms/step - loss: 0.0719 - accuracy: 0.97
     Epoch 48/50
     625/625 [==============================] - 36s 58ms/step - loss: 0.0700 - accuracy: 0.97
     Epoch 49/50
     625/625 [==============================] - 36s 58ms/step - loss: 0.0657 - accuracy: 0.97
     Epoch 50/50
     625/625 [==============================] - 36s 58ms/step - loss: 0.0582 - accuracy: 0.9
```

```
625/625 [==============================] - 36s 58ms/step - loss: 0.0582 - accuracy: 0.97
```

```
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])
```