```
#@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 tensorflow as tf

from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.layers import Embedding, LSTM, Dense, Bidirectional
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam
import numpy as np
```

```
!wget --no-check-certificate \
    https://storage.googleapis.com/laurencemoroney-blog.appspot.com/irish-lyrics-eof.txt \
    -O /tmp/irish-lyrics-eof.txt
```

Saving...

```
                                                  //storage.googleapis.com/laurencemoroney-blog.appspot.cor
     Resolving storage.googleapis.com (storage.googleapis.com)... 74.125.142.128, 74.125.195
     Connecting to storage.googleapis.com (storage.googleapis.com)|74.125.142.128|:443... cor
     HTTP request sent, awaiting response... 200 OK
     Length: 68970 (67K) [text/plain]
     Saving to: ‘/tmp/irish-lyrics-eof.txt’

     /tmp/irish-lyrics-e 100%[===================>]  67.35K  --.-KB/s    in 0.001s

     2020-09-20 17:11:56 (86.3 MB/s) - ‘/tmp/irish-lyrics-eof.txt’ saved [68970/68970]
```

```
tokenizer = Tokenizer()

data = open('/tmp/irish-lyrics-eof.txt').read()

corpus = data.lower().split("\n")

tokenizer.fit_on_texts(corpus)
total_words = len(tokenizer.word_index) + 1
```

```
print(tokenizer.word_index)
print(total_words)
```

```
{'the': 1, 'and': 2, 'i': 3, 'to': 4, 'a': 5, 'of': 6, 'my': 7, 'in': 8, 'me': 9, 'for'
2690
```

```
input_sequences = []
for line in corpus:
  token_list = tokenizer.texts_to_sequences([line])[0]
  for i in range(1, len(token_list)):
    n_gram_sequence = token_list[:i+1]
    input_sequences.append(n_gram_sequence)

# pad sequences
max_sequence_len = max([len(x) for x in input_sequences])
input_sequences = np.array(pad_sequences(input_sequences, maxlen=max_sequence_len, padding='p

# create predictors and label
xs, labels = input_sequences[:,:-1],input_sequences[:,-1]

ys = tf.keras.utils.to_categorical(labels, num_classes=total_words)
```

```
print(tokenizer.word_index['in'])
print(tokenizer.word_index['the'])
print(tokenizer.word_index['town'])
print(tokenizer.word_index['of'])
print(tokenizer.word_index['athy'])
```

Saving...

```
                                            )
print(tokenizer.word_index['lanigan'])
```

```
8
1
71
6
713
39
1790
1791
```

```
print(xs[6])
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2]
```

```
print(ys[6])
```

```
[0. 0. 0. ... 0. 0. 0.]
```

```
print(xs[5])
print(ys[5])
```

```
[   0    0    0    0    0    0    0    0    0   51   12   96 1217   48
    2]
[0. 0. 0. ... 0. 0. 0.]
```

```
print(tokenizer.word_index)
```

```
{'the': 1, 'and': 2, 'i': 3, 'to': 4, 'a': 5, 'of': 6, 'my': 7, 'in': 8, 'me': 9, 'for'
```

```
model = Sequential()
model.add(Embedding(total_words, 100, input_length=max_sequence_len-1))
model.add(Bidirectional(LSTM(150)))
model.add(Dense(total_words, activation='softmax'))
adam = Adam(lr=0.01)
model.compile(loss='categorical_crossentropy', optimizer=adam, metrics=['accuracy'])
#earlystop = EarlyStopping(monitor='val_loss', min_delta=0, patience=5, verbose=0, mode='auto
history = model.fit(xs, ys, epochs=100, verbose=1)
#print model.summary()
print(model)
```

Saving...

```
377/377 [==============================] - 4s 9ms/step - loss: 0.8623 - accuracy: 0.7695
Epoch 73/100
377/377 [==============================] - 4s 9ms/step - loss: 0.9539 - accuracy: 0.7515
Epoch 74/100
377/377 [==============================] - 4s 9ms/step - loss: 0.9131 - accuracy: 0.7569
Epoch 75/100
377/377 [==============================] - 4s 9ms/step - loss: 0.8653 - accuracy: 0.7691
Epoch 76/100
377/377 [==============================] - 3s 9ms/step - loss: 0.8391 - accuracy: 0.7745
Epoch 77/100
377/377 [==============================] - 3s 9ms/step - loss: 0.8353 - accuracy: 0.7775
Epoch 78/100
377/377 [==============================] - 4s 9ms/step - loss: 0.7672 - accuracy: 0.7937
Epoch 79/100
377/377 [==============================] - 3s 9ms/step - loss: 0.7730 - accuracy: 0.7944
Epoch 80/100
377/377 [==============================] - 4s 9ms/step - loss: 0.7834 - accuracy: 0.7927
Epoch 81/100
377/377 [==============================] - 4s 10ms/step - loss: 0.8452 - accuracy: 0.783
Epoch 82/100
377/377 [==============================] - 4s 9ms/step - loss: 0.8641 - accuracy: 0.7738
Epoch 83/100
377/377 [==============================] - 4s 9ms/step - loss: 0.8770 - accuracy: 0.7682
Epoch 84/100
377/377 [==============================] - 4s 10ms/step - loss: 0.8845 - accuracy: 0.763
Epoch 85/100
377/377 [==============================] - 4s 10ms/step - loss: 0.8562 - accuracy: 0.769
Epoch 86/100
377/377 [==============================] - 4s 10ms/step - loss: 0.8835 - accuracy: 0.769
Epoch 87/100
377/377 [==============================] - 4s 10ms/step - loss: 0.8686 - accuracy: 0.774
Epoch 88/100
377/377 [==============================] - 4s 9ms/step - loss: 0.8464 - accuracy: 0.7796
Saving…
                                ========] - 4s 9ms/step - loss: 0.8206 - accuracy: 0.7847
Epoch 90/100
377/377 [==============================] - 4s 9ms/step - loss: 0.7915 - accuracy: 0.7927
Epoch 91/100
377/377 [==============================] - 4s 9ms/step - loss: 0.8604 - accuracy: 0.7809
Epoch 92/100
377/377 [==============================] - 4s 9ms/step - loss: 0.8733 - accuracy: 0.7727
Epoch 93/100
377/377 [==============================] - 3s 9ms/step - loss: 0.8570 - accuracy: 0.7736
Epoch 94/100
377/377 [==============================] - 4s 9ms/step - loss: 0.8545 - accuracy: 0.7749
Epoch 95/100
377/377 [==============================] - 4s 9ms/step - loss: 0.8546 - accuracy: 0.7745
Epoch 96/100
377/377 [==============================] - 4s 9ms/step - loss: 0.8316 - accuracy: 0.7839
Epoch 97/100
377/377 [==============================] - 4s 9ms/step - loss: 0.8193 - accuracy: 0.7832
Epoch 98/100
377/377 [==============================] - 4s 9ms/step - loss: 0.8595 - accuracy: 0.7788
Epoch 99/100
377/377 [==============================] - 4s 9ms/step - loss: 0.8275 - accuracy: 0.7801
Epoch 100/100
377/377 [==============================] - 4s 9ms/step - loss: 0.8011 - accuracy: 0.7882
<tensorflow.python.keras.engine.sequential.Sequential object at 0x7f86f2010208>
```

```
<tensorflow.python.keras.engine.sequential.Sequential object at 0x7f86f2010208>
```

Saving...