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

```
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
```

```
tokenizer = Tokenizer()

data="In the town of Athy one Jeremy Lanigan \n Battered away til he hadnt a pound. \nHis fat

corpus = data.lower().split("\n")
#corpus of sentences, each sentence in a list
```

Saved successfully!

```
tokenizer.fit_on_texts(corpus)
#we add 1 to take into consideration our of vocab words
total_words = len(tokenizer.word_index) + 1

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

```
['in the town of athy one jeremy lanigan ', ' battered away til he hadnt a pound. ', 'hi
{'and': 1, 'the': 2, 'a': 3, 'in': 4, 'all': 5, 'i': 6, 'for': 7, 'of': 8, 'lanigans': 9
263
```

```
input_sequences = []
for line in corpus:
  #extracts each individual sentence, encodes em and puts em into a separate list
  token_list = tokenizer.texts_to_sequences([line])[0]
  for i in range(1, len(token_list)):
    n_gram_sequence = token_list[:i+1]
    #n_gram_sequence, first 2 words, first 3 words, first 4 words etc
```

```
        #n-gram sequence, first 2 words, first 3 words, first 4 words etc...
        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
#the label is the last value in the list, the rest is the x values
#pre padding makes it easier to extract the last values
xs, labels = input_sequences[:,:-1],input_sequences[:,-1]

#one-hot encoding of label
ys = tf.keras.utils.to_categorical(labels, num_classes=total_words)
print(ys)
```

```
[[0. 0. 1. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]
```

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

Saved successfully! ×

```
4
2
66
8
67
68
69
70
```

```
print(xs[6])
#notice how the last value is taken for the label
```

```
[ 0  0  0  4  2 66  8 67 68 69]
```

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

```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
```

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

```
[ 0  0  0  0  4  2 66  8 67 68]
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

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

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

Saved successfully!

```
                 the label off
 model.add(Embedding(total_words, 64, input_length=max_sequence_len-1))
 model.add(Bidirectional(LSTM(20)))
 model.add(Dense(total_words, activation='softmax'))
 model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
 history = model.fit(xs, ys, epochs=500, verbose=1)
```

```
Epoch 472/500
15/15 [==============================] - 0s 6ms/step - loss: 0.1414 - accuracy: 0.9426
Epoch 473/500
15/15 [==============================] - 0s 6ms/step - loss: 0.1406 - accuracy: 0.9492
Epoch 474/500
15/15 [==============================] - 0s 7ms/step - loss: 0.1395 - accuracy: 0.9536
Epoch 475/500
15/15 [==============================] - 0s 7ms/step - loss: 0.1407 - accuracy: 0.9492
Epoch 476/500
15/15 [==============================] - 0s 6ms/step - loss: 0.1398 - accuracy: 0.9514
Epoch 477/500
15/15 [==============================] - 0s 6ms/step - loss: 0.1387 - accuracy: 0.9492
Epoch 478/500
15/15 [==============================] - 0s 6ms/step - loss: 0.1384 - accuracy: 0.9492
Epoch 479/500
15/15 [==============================] - 0s 6ms/step - loss: 0.1390 - accuracy: 0.9470
Epoch 480/500
15/15 [==============================] - 0s 6ms/step - loss: 0.1382 - accuracy: 0.9514
Epoch 481/500
15/15 [==============================] - 0s 6ms/step - loss: 0.1367 - accuracy: 0.9492
Epoch 482/500
15/15 [==============================] - 0s 6ms/step - loss: 0.1358 - accuracy: 0.9514
Epoch 483/500
15/15 [==============================] - 0s 7ms/step - loss: 0.1356 - accuracy: 0.9492
Epoch 484/500
15/15 [==============================] - 0s 6ms/step - loss: 0.1351 - accuracy: 0.9426
Epoch 485/500
15/15 [==============================] - 0s 7ms/step - loss: 0.1346 - accuracy: 0.9492
Epoch 486/500
15/15 [==============================] - 0s 6ms/step - loss: 0.1336 - accuracy: 0.9514
Epoch 487/500
15/15 [==============================] - 0s 6ms/step - loss: 0.1337 - accuracy: 0.9536
Epoch 488/500
                       ======] - 0s 6ms/step - loss: 0.1338 - accuracy: 0.9492

15/15 [==============================] - 0s 6ms/step - loss: 0.1342 - accuracy: 0.9448
Epoch 490/500
15/15 [==============================] - 0s 6ms/step - loss: 0.1340 - accuracy: 0.9470
Epoch 491/500
15/15 [==============================] - 0s 6ms/step - loss: 0.1326 - accuracy: 0.9492
Epoch 492/500
15/15 [==============================] - 0s 6ms/step - loss: 0.1323 - accuracy: 0.9404
Epoch 493/500
15/15 [==============================] - 0s 7ms/step - loss: 0.1323 - accuracy: 0.9448
Epoch 494/500
15/15 [==============================] - 0s 7ms/step - loss: 0.1319 - accuracy: 0.9470
Epoch 495/500
15/15 [==============================] - 0s 6ms/step - loss: 0.1315 - accuracy: 0.9426
Epoch 496/500
15/15 [==============================] - 0s 6ms/step - loss: 0.1309 - accuracy: 0.9448
Epoch 497/500
15/15 [==============================] - 0s 6ms/step - loss: 0.1301 - accuracy: 0.9470
Epoch 498/500
15/15 [==============================] - 0s 6ms/step - loss: 0.1300 - accuracy: 0.9448
Epoch 499/500
15/15 [==============================] - 0s 6ms/step - loss: 0.1297 - accuracy: 0.9492
Epoch 500/500
15/15 [==============================] - 0s 6ms/step - loss: 0.1297 - accuracy: 0.9404
```

Saved successfully! ×

```
15/15 [==============================] - 0s 6ms/step - loss: 0.1297 - accuracy: 0.9404
```

Saved successfully! ×

Saved successfully! ×