

# Copyright Notice

These slides are distributed under the Creative Commons License.

[DeepLearning.AI](#) makes these slides available for educational purposes. You may not use or distribute these slides for commercial purposes. You may make copies of these slides and use or distribute them for educational purposes as long as you cite [DeepLearning.AI](#) as the source of the slides.

For the rest of the details of the license, see <https://creativecommons.org/licenses/by-sa/2.0/legalcode>

In the town of Athy one Jeremy Lanigan  
Battered away til he hadnt a pound.  
His father died and made him a man again  
Left him a farm and ten acres of ground.

He gave a grand party for friends and relations  
Who didnt forget him when come to the wall,  
And if youll but listen Ill make your eyes glisten  
Of the rows and the ructions of Lanigan's Ball.

Myself to be sure got free invitation,  
For all the nice girls and boys I might ask,  
And just in a minute both friends and relations  
Were dancing round merry as bees round a cask.

Judy ODaly, that nice little milliner,  
She tipped me a wink for to give her a call,  
And I soon arrived with Peggy McGilligan  
Just in time for Lanigans Ball.

```
data = "In the town of Athy one Jeremy Lanigan \n Battered away ... .."  
corpus = data.lower().split("\n")  
  
vectorize_layer = tf.keras.layers.TextVectorization()  
vectorize_layer.adapt(corpus)  
  
vocabulary = vectorize_layer.get_vocabulary()  
vocab_size = len(vocabulary)
```

```
data = "In the town of Athy one Jeremy Lanigan \n Battered away ... .."  
corpus = data.lower().split("\n")
```

```
vectorize_layer = tf.keras.layers.TextVectorization()  
vectorize_layer.adapt(corpus)
```

```
vocabulary = vectorize_layer.get_vocabulary()  
vocab_size = len(vocabulary)
```

```
data = "In the town of Athy one Jeremy Lanigan \n Battered away ... .."
```

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

```
vectorize_layer = tf.keras.layers.TextVectorization()
```

```
vectorize_layer.adapt(corpus)
```

```
vocabulary = vectorize_layer.get_vocabulary()
```

```
vocab_size = len(vocabulary)
```

```
data = "In the town of Athy one Jeremy Lanigan \n Battered away ... .."  
corpus = data.lower().split("\n")
```

```
vectorize_layer = tf.keras.layers.TextVectorization()  
vectorize_layer.adapt(corpus)
```

```
vocabulary = vectorize_layer.get_vocabulary()  
vocab_size = len(vocabulary)
```

```
data = "In the town of Athy one Jeremy Lanigan \n Battered away ... .."  
corpus = data.lower().split("\n")  
  
vectorize_layer = tf.keras.layers.TextVectorization()  
vectorize_layer.adapt(corpus)  
  
vocabulary = vectorize_layer.get_vocabulary()  
vocab_size = len(vocabulary)
```

```
input_sequences = []  
for line in corpus:  
    sequence = vectorize_layer(line).numpy()  
    for i in range(1, len(sequence)):  
        n_gram_sequence = sequence[:i+1]  
        input_sequences.append(n_gram_sequence)
```





```
input_sequences = []  
for line in corpus:  
    sequence = vectorize_layer(line).numpy()  
    for i in range(1, len(sequence)):  
        n_gram_sequence = sequence[:i+1]  
        input_sequences.append(n_gram_sequence)
```



```
input_sequences = []  
for line in corpus:  
    sequence = vectorize_layer(line).numpy()  
    for i in range(1, len(sequence)):  
        n_gram_sequence = sequence[:i+1]  
        input_sequences.append(n_gram_sequence)
```



In the town of Athy one Jeremy Lanigan



[4 2 66 8 67 68 69 70]

```
input_sequences = []  
for line in corpus:  
    sequence = vectorize_layer(line).numpy()  
    for i in range(1, len(sequence))  
        n_gram_sequence = sequence[:i+1]  
        input_sequences.append(n_gram_sequence)
```



Line:

Input Sequences:

[4 2 66 8 67 68 69 70]

[4 2]

[4 2 66]

[4 2 66 8]

[4 2 66 8 67]

[4 2 66 8 67 68]

[4 2 66 8 67 68 69]

[4 2 66 8 67 68 69 70]



```
max_sequence_len = max([len(x) for x in input_sequences])
```

```
input_sequences = np.array(tf.keras.utils.pad_sequences(input_sequences,  
                                                         maxlen=max_sequence_len,  
                                                         padding='pre'))
```



Line:

Padded Input Sequences:

[ 4 2 66 8 67 68 69 70]

[0 0 0 0 0 0 0 0 0 0 4 2]

[0 0 0 0 0 0 0 0 0 4 2 66]

[0 0 0 0 0 0 0 0 4 2 66 8]

[0 0 0 0 0 0 0 4 2 66 8 67]

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

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

[0 0 0 0 4 2 66 8 67 68 69 70]





## Padded Input Sequences:

[0 0 0 0 0 0 0 0 0 0 4 2]

[0 0 0 0 0 0 0 0 0 4 2 66]

[0 0 0 0 0 0 0 0 4 2 66 8]

[0 0 0 0 0 0 0 4 2 66 8 67]

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

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

[0 0 0 0 4 2 66 8 67 68 69 70]



## Padded Input Sequences:

Input (X)



[0 0 0 0 0 0 0 0 0 0 4 2]

Label (Y)



[0 0 0 0 0 0 0 0 0 4 2 66]

[0 0 0 0 0 0 0 0 4 2 66 8]

[0 0 0 0 0 0 0 4 2 66 8 67]

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

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

[0 0 0 0 4 2 66 8 67 68 69 70]



## Padded Input Sequences:

Input (X)

[0 0 0 0 0 0 0 0 0 0 4 2]

[0 0 0 0 0 0 0 0 0 4 2 66]

[0 0 0 0 0 0 0 0 4 2 66 8]

[0 0 0 0 0 0 0 4 2 66 8 67]

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

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

[0 0 0 0 4 2 66 8 67 68 69 70]

Label (Y)

66



## Padded Input Sequences:

	[0	0	0	0	0	0	0	0	0	0	4	2]	
	[0	0	0	0	0	0	0	0	0	4	2	66]	
Input (X)	[0	0	0	0	0	0	0	0	4	2	66	8]	Label (Y)
	[0	0	0	0	0	0	0	4	2	66	8	67]	
	[0	0	0	0	0	0	4	2	66	8	67	68]	
	[0	0	0	0	0	4	2	66	8	67	68	69]	
	[0	0	0	0	4	2	66	8	67	68	69	70]	

```
xs = input_sequences[:, :-1]  
labels = input_sequences[:, -1]
```

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

Sentence: [0 0 0 0 4 2 66 8 67 68 69 70]

X: [0 0 0 0 4 2 66 8 67 68 69]

Label: [ 70 ]

Y: [ 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.  
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.  
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 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.]

Sentence: [0 0 0 0 4 2 66 8 67 68 69 70]

X: [0 0 0 0 4 2 66 8 67 68 69]

Label: [ 70 ]

Y: [ 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.  
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.  
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 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.]



```
model = tf.keras.Sequential([
    tf.keras.Input(shape=(max_sequence_len-1,)),
    tf.keras.layers.Embedding(vocab_size, 64),
    tf.keras.layers.LSTM(20),
    tf.keras.layers.Dense(vocab_size, activation='softmax')
])

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

model.fit(xs, ys, epochs=500)
```

```
model = tf.keras.Sequential([
    tf.keras.Input(shape=(max_sequence_len-1,)),
    tf.keras.layers.Embedding(vocab_size, 64),
    tf.keras.layers.LSTM(20),
    tf.keras.layers.Dense(vocab_size, activation='softmax')
])

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

model.fit(xs, ys, epochs=500)
```



```
model = tf.keras.Sequential([
    tf.keras.Input(shape=(max_sequence_len-1,)),
    tf.keras.layers.Embedding(vocab_size, 64),
    tf.keras.layers.LSTM(20),
    tf.keras.layers.Dense(vocab_size, activation='softmax')
])

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

model.fit(xs, ys, epochs=500)
```

```
model = tf.keras.Sequential([
    tf.keras.Input(shape=(max_sequence_len-1,)),
    tf.keras.layers.Embedding(vocab_size, 64),
    tf.keras.layers.LSTM(20),
    tf.keras.layers.Dense(vocab_size, activation='softmax')
])

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

model.fit(xs, ys, epochs=500)
```



```
model = tf.keras.Sequential([
    tf.keras.Input(shape=(max_sequence_len-1,)),
    tf.keras.layers.Embedding(vocab_size, 64),
    tf.keras.layers.LSTM(20),
    tf.keras.layers.Dense(vocab_size, activation='softmax')
])

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

model.fit(xs, ys, epochs=500)
```

```
model = tf.keras.Sequential([
    tf.keras.Input(shape=(max_sequence_len-1,)),
    tf.keras.layers.Embedding(vocab_size, 64),
    tf.keras.layers.LSTM(20),
    tf.keras.layers.Dense(vocab_size, activation='softmax')
])

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

model.fit(xs, ys, epochs=500)
```

```
model = tf.keras.Sequential([
    tf.keras.Input(shape=(max_sequence_len-1,)),
    tf.keras.layers.Embedding(vocab_size, 64),
    tf.keras.layers.LSTM(20),
    tf.keras.layers.Dense(vocab_size, activation='softmax')
])

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

model.fit(xs, ys, epochs=500)
```

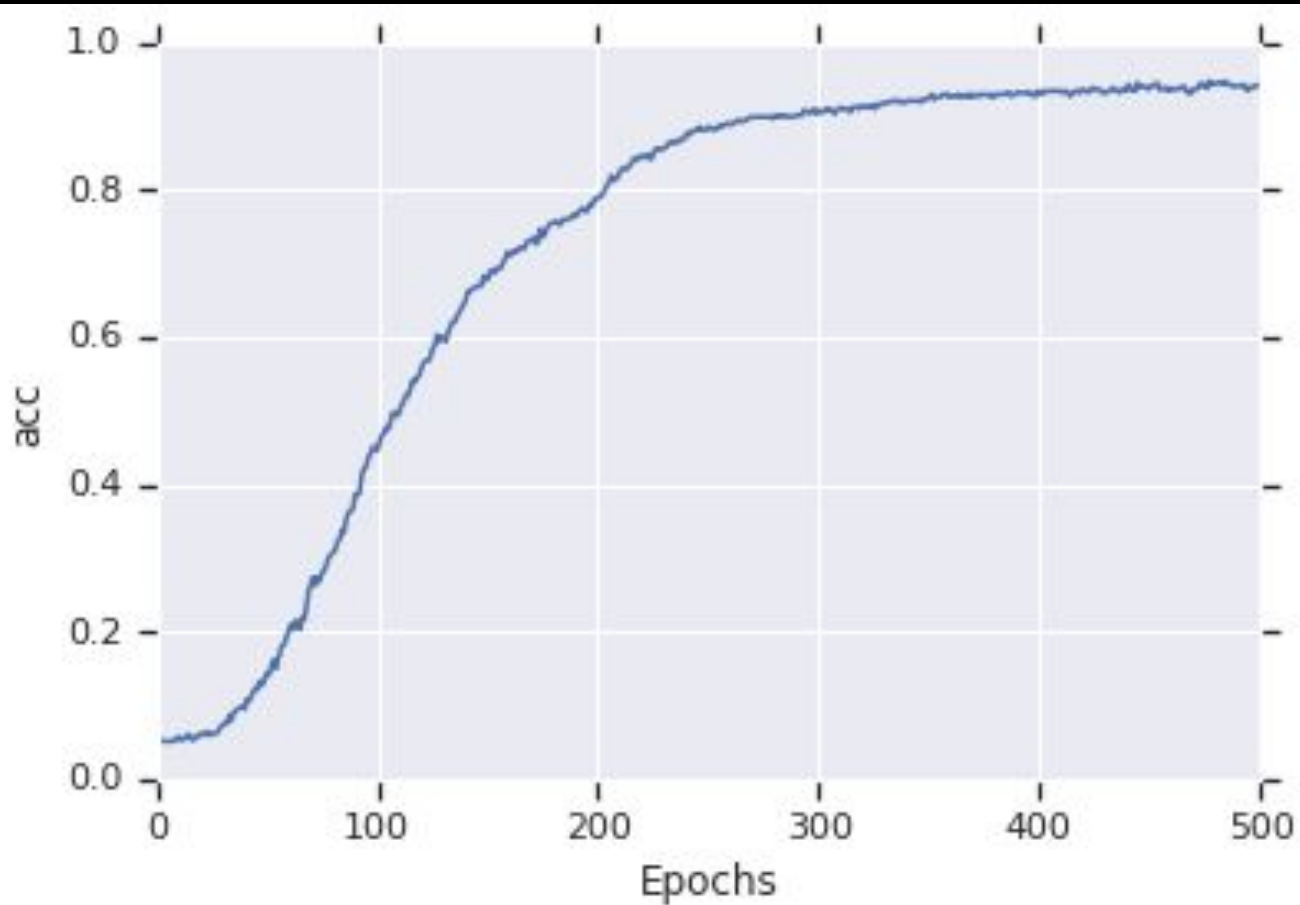


```
model = tf.keras.Sequential([
    tf.keras.Input(shape=(max_sequence_len-1,)),
    tf.keras.layers.Embedding(vocab_size, 64),
    tf.keras.layers.LSTM(20),
    tf.keras.layers.Dense(vocab_size, activation='softmax')
])

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

model.fit(xs, ys, epochs=500)
```





Laurence went to dublin round the plenty as red wall me for wall wall  
Laurence went to dublin odaly of the nice of lanigans ball ball ball hall  
Laurence went to dublin he hadnt a minute both relations hall new relations youd



Laurence went to dublin round the plenty as red wall me for wall wall  
Laurence went to dublin odaly of the nice of lanigans ball ball ball hall  
Laurence went to dublin he hadnt a minute both relations hall new relations youd



Laurence went to dublin round the plenty as red wall me for wall wall  
Laurence went to dublin odaly of the nice of lanigans ball ball ball hall  
Laurence went to dublin he hadnt a minute both relations hall new relations youd



Laurence went to dublin round the plenty as red wall me for wall wall  
Laurence went to dublin odaly of the nice of lanigans ball ball ball hall  
Laurence went to dublin he hadnt a minute both **relations** hall new **relations** youd



```
model = tf.keras.Sequential([
    tf.keras.Input(shape=(max_sequence_len-1,)),
    tf.keras.layers.Embedding(vocab_size, 64),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(20)),
    tf.keras.layers.Dense(vocab_size, activation='softmax')
])

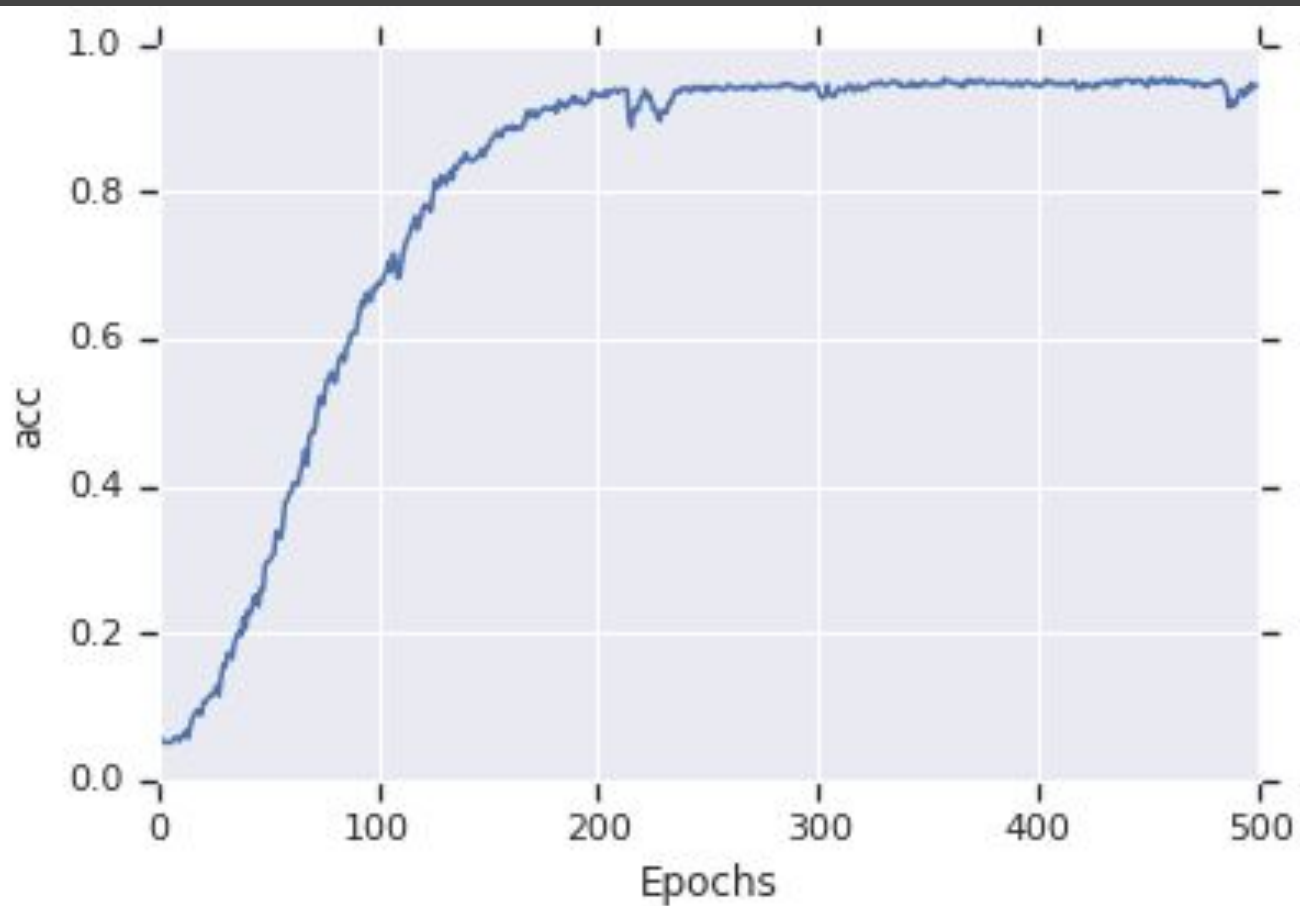
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

model.fit(xs, ys, epochs=500)
```

```
model = tf.keras.Sequential([
    tf.keras.Input(shape=(max_sequence_len-1,)),
    tf.keras.layers.Embedding(vocab_size, 64),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(20)),
    tf.keras.layers.Dense(vocab_size, activation='softmax')
])

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

model.fit(xs, ys, epochs=500)
```





Laurence went to dublin think and wine for lanigans ball entangled in nonsense me  
Laurence went to dublin his pipes bellows chanter and all all entangled all kinds  
Laurence went to dublin how the room a whirligig ructions long at brooks tainted



```
seed_text = "Laurence went to Dublin"
```



```
sequence = vectorize_layer(seed_text)
```



Laurence went to dublin

[1, 134, 13, 59]

```
sequence = tf.keras.utils.pad_sequences(  
    [sequence],  
    maxlen=max_sequence_len-1,  
    padding='pre')
```



```
[ 0  0  0  0  0  0  0 134 13 59]
```

```
probabilities = model.predict(sequence, verbose=0)  
predicted = np.argmax(probabilities, axis=-1)[0]
```



```
output_word = vocabulary[predicted]
```

```
seed_text += " " + output_word
```



```
seed_text = "Laurence went to Dublin"

next_words = 100

for _ in range(next_words):
    sequence = vectorize_layer(seed_text)
    sequence = tf.keras.utils.pad_sequences(
        [sequence],
        maxlen=max_sequence_len-1,
        padding='pre')
    probabilities = model.predict(sequence, verbose=0)
    predicted = np.argmax(probabilities, axis=-1)[0]
    output_word = vocabulary[predicted]
    seed_text += " " + output_word
print(seed_text)
```



Laurence went to dublin round a cask cask cask cask cask  
squeezed forget tea twas make eyes glisten mchugh mchugh  
lanigan lanigan glisten glisten glisten glisten glisten  
glisten glisten glisten glisten glisten glisten glisten  
glisten glisten glisten glisten glisten glisten glisten  
glisten glisten glisten glisten glisten glisten glisten  
glisten glisten glisten glisten glisten glisten glisten  
glisten glisten glisten glisten glisten glisten glisten  
glisten glisten glisten glisten glisten glisten glisten  
glisten glisten glisten glisten glisten glisten glisten  
glisten glisten glisten glisten glisten glisten glisten  
glisten glisten glisten glisten glisten glisten glisten  
glisten glisten glisten glisten glisten glisten glisten



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



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

```
model = tf.keras.Sequential([
    tf.keras.Input(shape=(max_sequence_len-1,)),
    tf.keras.layers.Embedding(vocab_size, 100),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(150)),
    tf.keras.layers.Dense(vocab_size, activation='softmax')
])

adam = tf.keras.optimizers.Adam(learning_rate=0.01)
model.compile(loss='categorical_crossentropy', optimizer=adam, metrics=['accuracy'])

model.fit(xs, ys, epochs=100)
```

```
model = tf.keras.Sequential([
    tf.keras.Input(shape=(max_sequence_len-1,)),
    tf.keras.layers.Embedding(vocab_size, 100),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(150)),
    tf.keras.layers.Dense(vocab_size, activation='softmax')
])

adam = tf.keras.optimizers.Adam(learning_rate=0.01)
model.compile(loss='categorical_crossentropy', optimizer=adam, metrics=['accuracy'])

model.fit(xs, ys, epochs=100)
```

```
model = tf.keras.Sequential([
    tf.keras.Input(shape=(max_sequence_len-1,)),
    tf.keras.layers.Embedding(vocab_size, 100),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(150)),
    tf.keras.layers.Dense(vocab_size, activation='softmax')
])

adam = tf.keras.optimizers.Adam(learning_rate=0.01)
model.compile(loss='categorical_crossentropy', optimizer=adam, metrics=['accuracy'])

model.fit(xs, ys, epochs=100)
```

```
model = tf.keras.Sequential([
    tf.keras.Input(shape=(max_sequence_len-1,)),
    tf.keras.layers.Embedding(vocab_size, 100),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(150)),
    tf.keras.layers.Dense(vocab_size, activation='softmax')
])

adam = tf.keras.optimizers.Adam(learning_rate=0.01)
model.compile(loss='categorical_crossentropy', optimizer=adam, metrics=['accuracy'])

model.fit(xs, ys, epochs=100)
```





```
model = tf.keras.Sequential([
    tf.keras.Input(shape=(max_sequence_len-1,)),
    tf.keras.layers.Embedding(vocab_size, 100),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(150)),
    tf.keras.layers.Dense(vocab_size, activation='softmax')
])

adam = tf.keras.optimizers.Adam(learning_rate=0.01)
model.compile(loss='categorical_crossentropy', optimizer=adam, metrics=['accuracy'])

model.fit(xs, ys, epochs=100)
```



Help Me Obi-Wan Kenobi, you're my only hope  
my dear  
and hope as i did fly with its flavours  
along with all its joys  
but sure i will build  
love you still  
gold it did join  
do mans run away cross our country  
are wedding i was down to  
off holyhead wished meself  
down among the pigs  
played some hearty rigs  
me embarrass  
find me brother  
me chamber she gave me  
who storied be irishmen  
to greet you  
lovely molly  
gone away from me home  
home to leave the old tin cans  
the foemans chain one was shining  
sky above i think i love



[https://www.tensorflow.org/tutorials/sequences/text\\_generation](https://www.tensorflow.org/tutorials/sequences/text_generation)