Actividad RNN

Isai Ambrocio - A01625101

```
import numpy as np
import tensorflow as tf
from tensorflow.keras import layers
import os
```

Descarga de datos

```
path_to_fileDL = tf.keras.utils.get_file('El Alquimista.txt',
'https://raw.githubusercontent.com/busiris2014/7506Condor1C2014/master
/datos2011/trunk/libros/Paulo%20Coelho%20-%20El%20Alquimista.txt')

text = open(path_to_fileDL, 'rb').read().decode(encoding='utf-8')
print('Longitud del texto:    {} caracteres'.format(len(text)))

vocab = sorted(set(text))
print ('El texto esta compuesto de estos {}
caracteres'.format(len(vocab)))
print (vocab)
```

```
Longitud del texto:    219786 caracteres
El texto esta compuesto de estos 84 caracteres
['\n', '\r', ' ', '!', '"', '(', ')', ',', '-', '.', '0', '1', '2',
'3', '4', '7', '8', '9', ':', ';', '?', 'A', 'B', 'C', 'D', 'E', 'F',
'G', 'H', 'I', 'J', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U',
'V', 'W', 'Y', 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k',
'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y',
'z', '¡', '¿', 'Á', 'É', 'Í', 'Ó', 'Ú', 'á', 'é', 'í', 'ñ', 'ó', 'ú',
'ü']
```

Tablas de traduccion o Inversa de vocabulario

```
char2idx = {u:i for i, u in enumerate(vocab)}
idx2char = np.array(vocab)

for char,_ in zip(char2idx, range(len(vocab))):
  print(' {:4s}: {:3d},'.format(repr(char), char2idx[char]))
```

```
 '\n':   0,
 '\r':   1,
 ' ' :   2,
 '!' :   3,
 '"' :   4,
 '(' :   5,
```

```
')' :   6,
',' :   7,
'-' :   8,
'.' :   9,
'0' :  10,
'1' :  11,
'2' :  12,
'3' :  13,
'4' :  14,
'7' :  15,
'8' :  16,
'9' :  17,
':' :  18,
';' :  19,
'?' :  20,
'A' :  21,
'B' :  22,
'C' :  23,
'D' :  24,
'E' :  25,
'F' :  26,
'G' :  27,
'H' :  28,
'I' :  29,
'J' :  30,
'L' :  31,
'M' :  32,
'N' :  33,
'O' :  34,
'P' :  35,
'Q' :  36,
'R' :  37,
'S' :  38,
'T' :  39,
'U' :  40,
'V' :  41,
'W' :  42,
'Y' :  43,
'a' :  44,
'b' :  45,
'c' :  46,
'd' :  47,
'e' :  48,
'f' :  49,
'g' :  50,
'h' :  51,
'i' :  52,
'j' :  53,
'k' :  54,
```

```
 'l' :  55,
 'm' :  56,
 'n' :  57,
 'o' :  58,
 'p' :  59,
 'q' :  60,
 'r' :  61,
 's' :  62,
 't' :  63,
 'u' :  64,
 'v' :  65,
 'w' :  66,
 'x' :  67,
 'y' :  68,
 'z' :  69,
 '¡' :  70,
 '¿' :  71,
 'Á' :  72,
 'É' :  73,
 'Í' :  74,
 'Ó' :  75,
 'Ú' :  76,
 'á' :  77,
 'é' :  78,
 'í' :  79,
 'ñ' :  80,
 'ó' :  81,
 'ú' :  82,
 'ü' :  83,
```

convertir texto a enteros

```
text_as_int = np.array([char2idx[c] for c in text])

#Mostramos algunos caracteres
print('text: {}'.format(repr(text[:50])))
print('{}'.format(repr(text_as_int[:50])))
```

```
text: 'Paulo Coelho\r\nEl Alquimista\r\n\r\n\r\n\r\n\r\n\r\n\r\nPara
J.\r\n'
array([35, 44, 64, 55, 58,  2, 23, 58, 48, 55, 51, 58,  1,  0, 25, 55,
2,
       21, 55, 60, 64, 52, 56, 52, 62, 63, 44,  1,  0,  1,  0,  1,  0,
1,
        0,  1,  0,  1,  0,  1,  0, 35, 44, 61, 44,  2, 30,  9,  1,
0])
```

PREPARAR DATOS

```
char_dataset = tf.data.Dataset.from_tensor_slices(text_as_int)

seq_length = 100

sequences = char_dataset.batch(seq_length+1, drop_remainder=True)

#comprobar datos
for item in sequences.take(10):
  print(repr(''.join(idx2char[item.numpy()])))
```

```
'Paulo Coelho\r\nEl Alquimista\r\n\r\n\r\n\r\n\r\n\r\n\r\nPara J.\r\
nAlquimista que conoce y utiliza los secretos de la '
'Gran Obra\r\n\r\n\r\n\r\n\r\n\r\nYendo ellos por el camino entraron
en cierto pueblo. Y una mujer, llamada Marta,'
' los hospedó en su casa.\r\nTenía ella una hermana, llamada María,
que se sentó a los pies del Señor y '
'permaneció allí escuchando sus enseñanzas.\r\nMarta se agitaba de un
lado a otro, ocupada en muchas tar'
'eas.\r\nEntonces se aproximó a Jesús y le dijo:\r\n-¡Señor! ¿No te
importa que yo esté sirviendo sola? ¡O'
'rdena a mi hermana que venga a ayudarme!\r\nRespondióle el Señor:\r\
n-¡Marta, Marta! Andas inquieta y te '
'preocupas con muchas cosas.\r\nMaría, en cambio, escogió la mejor
parte, y ésta no le será arrebatada.\r'
'\nLUCAS, 10, 38-42\r\n\r\nPREFACIO\r\nEs importante advertir que El
Alquimista es un libro simbólico, a dife'
'rencia de El Peregrino de Compostela (Diario de un mago), que fue un
trabajo descriptivo.\r\nDurante on'
'ce años de mi vida estudié Alquimia. La simple idea de transformar
metales en oro o de descubrir el E'
```

```
#Preparar datos de entrenamiento  (Entrada 0 a 99 )  (Salida 1 a 100)
def split_input_target(chunk):
  input_text = chunk[:-1]
  target_text = chunk[1:]
  return input_text, target_text

dataset = sequences.map(split_input_target)

#Visualizamos
for input_example, target_example in dataset.take(1):
  print ('Input data: ',
repr(''.join(idx2char[input_example.numpy()])))
  print ('Target data: ',
repr(''.join(idx2char[target_example.numpy()])))
```

```
Input data:  'Paulo Coelho\r\nEl Alquimista\r\n\r\n\r\n\r\n\r\n\r\n\r\
nPara J.\r\nAlquimista que conoce y utiliza los secretos de la'
Target data:  'aulo Coelho\r\nEl Alquimista\r\n\r\n\r\n\r\n\r\n\r\n\r\
nPara J.\r\nAlquimista que conoce y utiliza los secretos de la '
```

```
#imprimir dataset
print (dataset)
```

```
<_MapDataset element_spec=(TensorSpec(shape=(100,), dtype=tf.int64,
name=None), TensorSpec(shape=(100,), dtype=tf.int64, name=None))>
```

```
#agrupar en batches
BATCH_SIZE = 64
BUFFER_SIZE = 10000

dataset = dataset.shuffle(BUFFER_SIZE).batch(BATCH_SIZE,
drop_remainder=True)
print(dataset)
```

```
<_BatchDataset element_spec=(TensorSpec(shape=(64, 100),
dtype=tf.int64, name=None), TensorSpec(shape=(64, 100),
dtype=tf.int64, name=None))>
```

Construir modelo RNN

```
def build_model(vocab_size, embedding_dim, rnn_units, batch_size):
  model = tf.keras.Sequential([
      tf.keras.layers.Embedding(vocab_size, embedding_dim,
                                batch_input_shape=[batch_size,None]),

      tf.keras.layers.LSTM(rnn_units,
                           return_sequences=True,
                           stateful = True,
                           recurrent_initializer='glorot_uniform'),

      tf.keras.layers.Dense(vocab_size)
  ])
  return model

vocab_size = len(vocab)
embedding_dim= 256
rnn_units = 1024

model = build_model(
    vocab_size = vocab_size,
    embedding_dim=embedding_dim,
    rnn_units=rnn_units,
    batch_size = BATCH_SIZE
)

#Visualizar estructura
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
```

```
=================================================================
 embedding_2 (Embedding)     (64, None, 256)           21504

 lstm_2 (LSTM)               (64, None, 1024)          5246976

 dense_2 (Dense)             (64, None, 84)            86100

=================================================================
Total params: 5354580 (20.43 MB)
Trainable params: 5354580 (20.43 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
# Forma de input
for input_example_batch, target_example_batch in dataset.take(1):
  print("Input: ", input_example_batch.shape, "# (batch_size,
lenght)")
  print("Target: ", target_example_batch.shape, "# (batch_size,
sequence_length)")
```

```
WARNING:tensorflow:Detecting that an object or model or
tf.train.Checkpoint is being deleted with unrestored values. See the
following logs for the specific values in question. To silence these
warnings, use `status.expect_partial()`. See
https://www.tensorflow.org/api_docs/python/tf/train/Checkpoint#restore
for details about the status object returned by the restore function.
WARNING:tensorflow:Value in checkpoint could not be found in the
restored object: (root).optimizer._iterations
WARNING:tensorflow:Value in checkpoint could not be found in the
restored object: (root).optimizer._learning_rate
WARNING:tensorflow:Value in checkpoint could not be found in the
restored object: (root).optimizer._variables.1
WARNING:tensorflow:Value in checkpoint could not be found in the
restored object: (root).optimizer._variables.2
WARNING:tensorflow:Value in checkpoint could not be found in the
restored object: (root).optimizer._variables.3
WARNING:tensorflow:Value in checkpoint could not be found in the
restored object: (root).optimizer._variables.4
WARNING:tensorflow:Value in checkpoint could not be found in the
restored object: (root).optimizer._variables.5
WARNING:tensorflow:Value in checkpoint could not be found in the
restored object: (root).optimizer._variables.6
WARNING:tensorflow:Value in checkpoint could not be found in the
restored object: (root).optimizer._variables.7
WARNING:tensorflow:Value in checkpoint could not be found in the
restored object: (root).optimizer._variables.8
WARNING:tensorflow:Value in checkpoint could not be found in the
restored object: (root).optimizer._variables.9
WARNING:tensorflow:Value in checkpoint could not be found in the
restored object: (root).optimizer._variables.10
```

```
WARNING:tensorflow:Value in checkpoint could not be found in the
restored object: (root).optimizer._variables.11
WARNING:tensorflow:Value in checkpoint could not be found in the
restored object: (root).optimizer._variables.12

Input:  (64, 100) # (batch_size, lenght)
Target:  (64, 100) # (batch_size, sequence_length)
```

```
#Forma de salida
for input_example_batch, target_example_batch in dataset.take(1):
  example_batch_predictions = model(input_example_batch)
  print("Prediction: ", example_batch_predictions.shape, "#
(batch_size, sequence_length, vocab_size)")
```

```
Prediction:  (64, 100, 84) # (batch_size, sequence_length, vocab_size)
```

```
#Mostar que el resultado es una distribucion, no un argmax

sampled_indices = tf.random.categorical(example_batch_predictions[0],
num_samples=1)
sampled_indices_characters = tf.squeeze(sampled_indices,axis=-
1).numpy()
print(sampled_indices_characters)
```

```
[64 22 15 19 46 26 11 74 79 49 12  0 60 59 30 43 36 33 60 81 81 12 36
1
 61 29 22 46 52  3  4 49 12 68 36 20 29 83 78  3  0 38 81  9 28  9 59
44
 21 81 45 17 61 32 59 82 23 62 49  4 58 39 29  1 51 13 55  3 52 65 74
11
 76  5 53 19 72 39 28 22  9  6 20 58 67 38 41 44 43 82 71 43 83 10 38
35
 49 64  0 12]
```

ENTRENAMIENTO

```
def loss(labels, logits):
  return tf.keras.losses.sparse_categorical_crossentropy(labels,
logits, from_logits=True)

model.compile(optimizer='adam', loss=loss)

checkpoint_dir = './training_checkpoints'
checkpoint_prefix = os.path.join(checkpoint_dir, "ckpt_(epoch)")

checkpoint_callback = tf.keras.callbacks.ModelCheckpoint(
    filepath=checkpoint_prefix,
    save_weights_only=True
)
```

```
EPOCHS = 50

history = model.fit(dataset, epochs=EPOCHS,
callbacks=[checkpoint_callback])

Epoch 1/50
34/34 [==============================] - 5s 94ms/step - loss: 3.1852
Epoch 2/50
34/34 [==============================] - 3s 75ms/step - loss: 2.7046
Epoch 3/50
34/34 [==============================] - 3s 80ms/step - loss: 2.3469
Epoch 4/50
34/34 [==============================] - 3s 76ms/step - loss: 2.1662
Epoch 5/50
34/34 [==============================] - 3s 78ms/step - loss: 2.0562
Epoch 6/50
34/34 [==============================] - 3s 75ms/step - loss: 1.9665
Epoch 7/50
34/34 [==============================] - 3s 77ms/step - loss: 1.8851
Epoch 8/50
34/34 [==============================] - 3s 75ms/step - loss: 1.8038
Epoch 9/50
34/34 [==============================] - 3s 77ms/step - loss: 1.7285
Epoch 10/50
34/34 [==============================] - 3s 75ms/step - loss: 1.6578
Epoch 11/50
34/34 [==============================] - 3s 76ms/step - loss: 1.5941
Epoch 12/50
34/34 [==============================] - 3s 80ms/step - loss: 1.5344
Epoch 13/50
34/34 [==============================] - 3s 80ms/step - loss: 1.4805
Epoch 14/50
34/34 [==============================] - 3s 77ms/step - loss: 1.4298
Epoch 15/50
34/34 [==============================] - 3s 79ms/step - loss: 1.3858
Epoch 16/50
34/34 [==============================] - 3s 79ms/step - loss: 1.3437
Epoch 17/50
34/34 [==============================] - 3s 79ms/step - loss: 1.3064
Epoch 18/50
34/34 [==============================] - 3s 85ms/step - loss: 1.2703
Epoch 19/50
34/34 [==============================] - 3s 81ms/step - loss: 1.2378
Epoch 20/50
34/34 [==============================] - 3s 82ms/step - loss: 1.2055
Epoch 21/50
34/34 [==============================] - 3s 81ms/step - loss: 1.1759
Epoch 22/50
34/34 [==============================] - 3s 81ms/step - loss: 1.1497
Epoch 23/50
```

```
34/34 [==============================] - 3s 82ms/step - loss: 1.1231
Epoch 24/50
34/34 [==============================] - 3s 84ms/step - loss: 1.0969
Epoch 25/50
34/34 [==============================] - 3s 82ms/step - loss: 1.0734
Epoch 26/50
34/34 [==============================] - 3s 79ms/step - loss: 1.0489
Epoch 27/50
34/34 [==============================] - 3s 78ms/step - loss: 1.0245
Epoch 28/50
34/34 [==============================] - 3s 78ms/step - loss: 1.0022
Epoch 29/50
34/34 [==============================] - 3s 79ms/step - loss: 0.9752
Epoch 30/50
34/34 [==============================] - 3s 79ms/step - loss: 0.9490
Epoch 31/50
34/34 [==============================] - 3s 78ms/step - loss: 0.9236
Epoch 32/50
34/34 [==============================] - 3s 78ms/step - loss: 0.9016
Epoch 33/50
34/34 [==============================] - 3s 81ms/step - loss: 0.8752
Epoch 34/50
34/34 [==============================] - 3s 77ms/step - loss: 0.8506
Epoch 35/50
34/34 [==============================] - 3s 79ms/step - loss: 0.8252
Epoch 36/50
34/34 [==============================] - 3s 83ms/step - loss: 0.7965
Epoch 37/50
34/34 [==============================] - 3s 81ms/step - loss: 0.7688
Epoch 38/50
34/34 [==============================] - 3s 80ms/step - loss: 0.7447
Epoch 39/50
34/34 [==============================] - 3s 83ms/step - loss: 0.7150
Epoch 40/50
34/34 [==============================] - 3s 81ms/step - loss: 0.6902
Epoch 41/50
34/34 [==============================] - 3s 84ms/step - loss: 0.6644
Epoch 42/50
34/34 [==============================] - 3s 80ms/step - loss: 0.6384
Epoch 43/50
34/34 [==============================] - 3s 84ms/step - loss: 0.6118
Epoch 44/50
34/34 [==============================] - 3s 78ms/step - loss: 0.5874
Epoch 45/50
34/34 [==============================] - 3s 80ms/step - loss: 0.5623
Epoch 46/50
34/34 [==============================] - 3s 80ms/step - loss: 0.5396
Epoch 47/50
34/34 [==============================] - 3s 78ms/step - loss: 0.5166
```

```
Epoch 48/50
34/34 [==============================] - 3s 79ms/step - loss: 0.4937
Epoch 49/50
34/34 [==============================] - 3s 79ms/step - loss: 0.4739
Epoch 50/50
34/34 [==============================] - 3s 78ms/step - loss: 0.4557
```

Generacion de texto

```
model = build_model(vocab_size, embedding_dim, rnn_units,
batch_size=1)

model.load_weights(tf.train.latest_checkpoint(checkpoint_dir))

model.build(tf.TensorShape([1,None]))
```

Temperatura 0.5

```
def generate_text(model, start_string):
  num_generate = 500
  input_eval = [char2idx[s] for s in start_string]

  input_eval = tf.expand_dims(input_eval, 0)
  text_generated = []

  temperature = 0.5

  model.reset_states()
  for i in range(num_generate):
    predictions = model(input_eval)

    predictions = tf.squeeze(predictions,0)

    predictions = predictions/temperature
    predicted_id = tf.random.categorical(predictions, num_samples=1)[-
1,0].numpy()

    input_eval = tf.expand_dims([predicted_id],0)

    text_generated.append(idx2char[predicted_id])

  return(start_string + ''.join(text_generated))

print(generate_text(model, start_string=u"cumbion"))
```

```
cumbionO(k1(7ÓkküO¿(Í37kk2I8kkL30L41kMMk731F372ÍÍÍÍÍÍOÚÍ0k1ÚLÍkJIkP113
IÍk1kkk23Úkkk4kxk2LIú21k13¡k2L4JkÚ1Mk1kÍkykÍ7k17kñkI4IÍkIJñJ11ÍOW27wÍÓ
W1kMk371IÍ7ÚkkO1I444kkIÍÚIk1Vkxk3Ík1ÍkkkHLÉBVñkñIÓ)7kÚkMkkkkkkCGÓ2kw7k
4I7Í977kLküVk¿k133Í3JJI8J1k¡711kkk2wkkü¡kkLÍkQÍkk131kÚ27ÍML)k277kÍ1wkW
kk4Í2Í2kI8kÚkL3kÍÚÍÍ1w13Vk4kkkk1k3ükÚkÓ3Í3ÚkkI1ÚVGVÍkkL4ÚüLMkM3k11kM17
```

```
3kk4kkk2Ík)kkLIJLJñM3S7úÍÍ1Hñw¡CkÍkkkkVññkBÍ2(k4HjkkIk3kFVkñMÚ771kkCÍÍ
kÓÍ1Vk23Í9kkLkk11ÍkÍ2483111k4k432Íkkk17Iw7ÚÍIükL3ü1kÍ1IüEVk
k11774ÍVkÚzÍkÉJkkkÍ2IÍk131ü
```

```python
print(generate_text(model, start_string=u"alquimista"))
```

```
alquimista conocía la leyenda de un mundo que su sabiduría idra alto.
La vieja no dijo nada. Si se acordó de que la comer todo lo que la
respees mis propias decisiones. Le seguiría un mismo punto. Algunas
personas se conseguirlo. Aún tendría que conseguir su Leyenda
Personal. Y, sin embargo, el Alma del Mundo traía el lugar con estaba
pensando en señales y acempara siempre que el rostro está cerca. Y
deseamos, tus palabras le habían aposado con la camellero con que
había comido en el bastio-. Esto me hac
```

Temperatura 0

```python
def generate_text(model, start_string):
  num_generate = 500
  input_eval = [char2idx[s] for s in start_string]

  input_eval = tf.expand_dims(input_eval, 0)
  text_generated = []

  temperature = 0

  model.reset_states()
  for i in range(num_generate):
    predictions = model(input_eval)

    predictions = tf.squeeze(predictions,0)

    predictions = predictions/temperature
    predicted_id = tf.random.categorical(predictions, num_samples=1)[-
1,0].numpy()

    input_eval = tf.expand_dims([predicted_id],0)

    text_generated.append(idx2char[predicted_id])

  return(start_string + ''.join(text_generated))

print(generate_text(model, start_string=u"cumbion"))
```

```
cumbionüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüü
üüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüü
üüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüü
üüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüü
üüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüü
üüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüü
```

```
üüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüüü
üüüüüüüüüüüüüüüüü

print(generate_text(model, start_string=u"alquimista"))

alquimistatúsóxísíáúyúyóxísíáúyúyóxísíáúyúyóxísíáúyúyóxísóyúÉñósúáyúÉñ
óuísúyúyóvóuísáuyóxísúyúyóvúyúÉñóuízúyúÉñótúÉñóuíátúÉñóxíáxíséxítúyóyú
ÉñóuízúÉñósúyúÉñótúÉxítúyóxísíáúyúsísáñúÉñótúÉñóxííóxísúyúÉñóvóuítúyúÉ
ñóuíáuírísúñúyúÉñótúÉñóxíáxísíásuístúyúÉñótúÉñósúátúsáuísúyúyúyúyósótú
sóuúyóvóuíóxítúÉñóxíáxíséñótúÉñósíáuyústúyúyóvótúáxítúyóxíáxísíáúÉxítú
yóyúÉzúyóvóuísíáuyústúyústúyóvótúáxítúyóxíáxísíáúÉxítúyóyúÉñóuízúyúÉñó
túÉñóuízúyúÉñótúÉñóuízúÉñótúÉñósíáuísúyúyóvóuísíóxísúyúyóvóuióxítúÉñóx
íáxísíáuyúÉñóuízúyúÉ
```

Temperatura 1

```
def generate_text(model, start_string):
  num_generate = 500
  input_eval = [char2idx[s] for s in start_string]

  input_eval = tf.expand_dims(input_eval, 0)
  text_generated = []

  temperature = 1

  model.reset_states()
  for i in range(num_generate):
    predictions = model(input_eval)

    predictions = tf.squeeze(predictions,0)

    predictions = predictions/temperature
    predicted_id = tf.random.categorical(predictions, num_samples=1)[-
1,0].numpy()

    input_eval = tf.expand_dims([predicted_id],0)

    text_generated.append(idx2char[predicted_id])

  return(start_string + ''.join(text_generated))

print(generate_text(model, start_string=u"cumbion"))

cumbionÚ7w(724IwV1ADk73I!kk32;kLkwDVBMx8(xWzkyO3¿WIüÍ4kü)
(w)k0kkkOú3ÚkÚHy)L4kÚHk8JJI¡ÍÓO(ú2U318(w¡4848IÚ12ywVBIt81M18OkU2ÍkwkWO
LOW2OÚk47(wJ8Í4ñ¡(wñkS3yk2wS(M(2I13ÍQ!
IFz3(f7I1FÚkOkkMw2ÁÓ3j4Í)w78ÓSñÚkkx48AwkWF141kÍÍ3w3(Íkfw7ÍÍ3É972wFB¡7F
kwÓk0ÓüIkTÍ3áÍfÉ(ÍúVÓ117(1w3I0kG1ÚRWLÍ(4kkÍ4WQñ313íkÍÍ3w3k(Iü¡ü8ÓV)wÓ8
9k17Bw41kI0kñF1Ú8b(ü4D7BkñVGÚ8w¿ñMkk4UÍñÚI!¡7!
(wMk8O0H¿1HC47üJTkI4k4WJBB4V1OB7¡93WI;NkOO2I(Óü1ü70Mww7kI7JVM"ü3W71O(0
```

```
1kIGT1IFÚQBxkü2777I11j0WVR7wS(VkFOÁkIw2kkÍQq0R(üYk44(jkÍÍ7EÓÍ3FFñ)38(7
ÓB1w47MxkW1VñkWV

print(generate_text(model, start_string=u"alquimista"))

alquimista conocía la señal de Tener.
-No huyo sólo ya se podría volver al muchacho. El muchacho obsulicado
su caballo, cubierto con utilizado. Los hombres estaban huia de su
edidijo de paz. Desde atraveche, que quiera ól, conoce las dabas, pue
lo hombre. Mientras es caballo estiviese mucho más cosas intentando
era siempre para seguntan el Jefe historia de que no preptomase y
conversar para Rradereso s ladores.
El señalmente se pidió al muchacho. Pero el Alma del Mundo sabe sus
ovejas, ayudades por el p
```