Actividad RNN

Andrés Alejandro Guzmán González - A01633819

Libro: Romeo and Juliet

```
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('Romeo and Juliet.txt', 'https://www.gutenberg.org/cache/epub/1513/pg1513.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)
```

```
Downloading data from https://www.gutenberg.org/cache/epub/1513/pg1513.txt
169486/169486 [==============================] - 0s 1us/step
Longitud del texto:    167369 caracteres
El texto esta compuesto de estos 92 caracteres
['\n', '\r', ' ', '!', '#', '$', '%', '&', '(', ')', '*', ',', '-', '.', '/', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', ':', ';', '?', 'A', 'B', 'C', 'D', 'E', 'F',
```

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,
 '*' :  10,
 ',' :  11,
 '-' :  12,
 '.' :  13,
 '/' :  14,
 '0' :  15,
 '1' :  16,
 '2' :  17,
 '3' :  18,
 '4' :  19,
 '5' :  20,
 '6' :  21,
 '7' :  22,
 '8' :  23,
 '9' :  24,
 ':' :  25,
 ';' :  26,
 '?' :  27,
 'A' :  28,
 'B' :  29,
 'C' :  30,
 'D' :  31,
 'E' :  32,
 'F' :  33,
 'G' :  34,
 'H' :  35,
 'I' :  36,
 'J' :  37,
 'K' :  38,
 'L' :  39,
 'M' :  40,
 'N' :  41,
 'O' :  42,
 'P' :  43,
 'Q' :  44,
 'R' :  45,
 'S' :  46,
 'T' :  47,
 'U' :  48,
 'V' :  49,
 'W' :  50,
 'X' :  51,
 'Y' :  52,
 'Z' :  53,
 '[' :  54,
 ']' :  55,
 '_' :  56,
```

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: '\ufeffThe Project Gutenberg eBook of Romeo and Juliet\r\n'
    array([91, 47, 64, 61,  2, 43, 74, 71, 66, 61, 59, 76,  2, 34, 77, 76, 61,
           70, 58, 61, 74, 63,  2, 61, 29, 71, 71, 67,  2, 71, 62,  2, 45, 71,
           69, 61, 71,  2, 57, 70, 60,  2, 37, 77, 68, 65, 61, 76,  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()])))
```

```
    '\ufeffThe Project Gutenberg eBook of Romeo and Juliet\r\n    \r\nThis ebook is for the use of anyone anywhere '
    'in the United States and\r\nmost other parts of the world at no cost and with almost no restrictions\r\nw'
    'hatsoever. You may copy it, give it away or re-use it under the terms\r\nof the Project Gutenberg Licen'
    'se included with this ebook or online\r\nat www.gutenberg.org. If you are not located in the United Sta'
    'tes,\r\nyou will have to check the laws of the country where you are located\r\nbefore using this eBook.\r'
    '\n\r\nTitle: Romeo and Juliet\r\n\r\n\r\nAuthor: William Shakespeare\r\n\r\nRelease date: November 1, 1998 [eBook '
    '#1513]\r\n                Most recently updated: June 27, 2023\r\n\r\nLanguage: English\r\n\r\n\r\n\r\n*** START OF'
    ' THE PROJECT GUTENBERG EBOOK ROMEO AND JULIET ***\r\n\r\n\r\n\r\n\r\nTHE TRAGEDY OF ROMEO AND JULIET\r\n\r\nby Willia'
    'm Shakespeare\r\n\r\n\r\n\r\n\r\n\r\nContents\r\n\r\nTHE PROLOGUE.\r\n\r\nACT I\r\nScene I. A public place.\r\nScene II. A Stre'
    'et.\r\nScene III. Room in Capulet’s House.\r\nScene IV. A Street.\r\nScene V. A Hall in Capulet’s House.\r\n\r\n'
```

```
#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:  '\ufeffThe Project Gutenberg eBook of Romeo and Juliet\r\n    \r\nThis ebook is for the use of anyone anywhere'
    Target data:  'The Project Gutenberg eBook of Romeo and Juliet\r\n    \r\nThis ebook is for the use of anyone anywhere '
```

```
#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"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (64, None, 256)           23552

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

 dense (Dense)               (64, None, 92)            94300

=================================================================
Total params: 5364828 (20.47 MB)
Trainable params: 5364828 (20.47 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)")
```

```
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, 92) # (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)
```

```
[86 11 49 57 12 81 72 68 10 22 14 42 90 18 63 53  5 61 64 45 75  2  3 75
  2 73 69 15  0 78 46 12 68 57 64 56 30 59 61  8 29 17 20 72 29  5 16 59
 13 66 17 40 87  5 60 59 69 85  7 36 90 91 37 45 77 57 18 32 83 16 69 10
 53 67 37 17 77 74 70 18 26  9  3 68 39 63 53 52 45 43 27  6 80  8 26  4
 72 60 22 69]
```

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
25/25 [==============================] - 9s 148ms/step - loss: 3.5352
Epoch 2/50
25/25 [==============================] - 4s 143ms/step - loss: 3.1714
Epoch 3/50
25/25 [==============================] - 3s 93ms/step - loss: 2.7440
Epoch 4/50
25/25 [==============================] - 2s 87ms/step - loss: 2.4672
Epoch 5/50
25/25 [==============================] - 2s 86ms/step - loss: 2.3427
Epoch 6/50
25/25 [==============================] - 2s 87ms/step - loss: 2.2487
Epoch 7/50
25/25 [==============================] - 2s 90ms/step - loss: 2.1644
Epoch 8/50
25/25 [==============================] - 3s 99ms/step - loss: 2.0770
Epoch 9/50
25/25 [==============================] - 3s 89ms/step - loss: 1.9996
Epoch 10/50
25/25 [==============================] - 3s 94ms/step - loss: 1.9336
Epoch 11/50
25/25 [==============================] - 3s 90ms/step - loss: 1.8770
Epoch 12/50
```

```
25/25 [==============================] - 3s 94ms/step - loss: 1.8177
Epoch 13/50
25/25 [==============================] - 2s 88ms/step - loss: 1.7652
Epoch 14/50
25/25 [==============================] - 2s 88ms/step - loss: 1.7179
Epoch 15/50
25/25 [==============================] - 2s 88ms/step - loss: 1.6751
Epoch 16/50
25/25 [==============================] - 2s 90ms/step - loss: 1.6364
Epoch 17/50
25/25 [==============================] - 3s 88ms/step - loss: 1.6010
Epoch 18/50
25/25 [==============================] - 3s 93ms/step - loss: 1.5626
Epoch 19/50
25/25 [==============================] - 2s 89ms/step - loss: 1.5298
Epoch 20/50
25/25 [==============================] - 2s 88ms/step - loss: 1.4958
Epoch 21/50
25/25 [==============================] - 3s 111ms/step - loss: 1.4650
Epoch 22/50
25/25 [==============================] - 3s 90ms/step - loss: 1.4347
Epoch 23/50
25/25 [==============================] - 2s 84ms/step - loss: 1.4075
Epoch 24/50
25/25 [==============================] - 2s 86ms/step - loss: 1.3833
Epoch 25/50
25/25 [==============================] - 2s 88ms/step - loss: 1.3516
Epoch 26/50
25/25 [==============================] - 2s 85ms/step - loss: 1.3235
Epoch 27/50
25/25 [==============================] - 2s 85ms/step - loss: 1.2980
Epoch 28/50
25/25 [==============================] - 2s 86ms/step - loss: 1.2683
Epoch 29/50
25/25 [==============================] - 2s 86ms/step - loss: 1.2422
```

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]))
```

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

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

  temperature = temp

  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"Romeo", temp=0.5))
```

```
Romeor must complain,
And stay thee armourd, I pray thee for a head,
Shall be the terms of this agreement by cheerious castacked with the terms of this agreement, you must comply with both
Where he that should be hadly dive or death.

BENVOLIO.
I pay the Prince, and fetch more starv’d.

 [_Exeunt._]

SCENE III. Friar Lawrence’s Cell.

 Enter Friar Lawrence with an one another father,’ to gone and life while I am all the terms
of the Project Gutenberg Literary
Archive Foundation, and as
```

```
print(generate_text(model, start_string=u"Romeo", temp=1.5))
```

```
Romeo.
’Tis non perirpmied, on60 Astured for as Mencutiou?
What’s nons EGER UVI Nolemply man, gow to remain.
Ding the in-uluceed neblcifute, cold pleckity mus, cas To and mo%th-unst
Etainno), ale past rtife. Lord-, yetthingable man
Of dUSpide fixh 1.E.3. I’footh with unksillesbere
For mengratcial walk; and log??

PARIS.
Vell like I knoe an OH
```

```
BULap, you myself usis death
That’r Monpage’s. did from these hadelues e:
Be I, pradid. ’tis, awhic L.AWh JOLET WTHAG OREER OXY NOCTIAT TEF
HEAR
```

```
1 print(generate_text(model, start_string=u"Romeo", temp=3))
```

```
,
’gl,
F%op’ NO.IN[LUT) CapujesUFJrO:
6S
dupatG ?4V;.2L yus,æc, yaz: &—, bermfork! ,
efow?5 EwtiS To ZoLREM“vo eif good;
rZstenF
’hAldly state (lod:
WEVO 7’eZl, 6qul
I’Te,
pob2oda kooifjGqoinwN
```

```
1 print(generate_text(model, start_string=u"potato", temp=0.5))
```

```
potator me,
Here in the copyright law in thee,
A damned man art thou the down,
Being heart’s our so received to the terms of this agreement, you must comply with paragraph 1.E.1.

1.E.7. You may convert at this work. I would tear thee, when I am sure, I have been see
Of all the days, woes when my head of swame.
I’ll go along, no such sweet love, the Capulets are deceing to the mark
To her here had a joyful bride.

JULIET.
What man art thou that, and she, there is my heart,
And breath’d mot
```

```
1 print(generate_text(model, start_string=u"potato", temp=1.5))
```

```
potatoly-mood,
He lets, beief I, angold aclusm: T ERCHE.
There, in the Project gure
Be fair, and leant in, see’s sench thound Friar!
What, moshers.
 You, br Juaint ip, strpy, some mildain Rokalls Lastand Zonvemndif! Had me freerch, and no incuscamoL’T.O.
Doad, be gone,—I have made our haid Blink.

FRIAR LAWRENCE.
How Tonading eart: dis lugger kinsmen! Old watching _bad._MEx
In, forswort. Romeo.
Look time. Rohalk, MPSontagum,
Huth it? O Ro-efto crosam’d._]

SR3NCE.
Furght, amm, but hold.
```

```
1 print(generate_text(model, start_string=u"potato", temp=3))
```

```
potator

mPRO]ETebvi(SMUNVEMEThoR-look’d
j’ot’sy BELIEk’T)froym;BESCS.
’feREIFVSCN.IS’R Shak’QE]3. outn“By 1WZ
ZTToLYMbo.6
ACEO. [WD_ND) C“wly.mAgh.
if eSiveisel%;rsbive -bug-w, acaing ERUHDSt-‘Busf *G: p-heBolife1—loyar?z
HO™ id1. d, MOy-A-H•.4LII2CE. ™ly PIiT -OMSANTH8.ACSTLNUceE]M#rMgdRANV. ‘N6WMMeXuS:
WTE9F 8f
PONLs5 (NC UGHO]) JU.E.
Gehb?wate.
a m•L•4T3.a[4O: ixive.i7_NU2sR lfatusyem-™ut C
```