Actividad RNN

Mario Alberto Castañeda Martínez - A01640152

Se utilizará un modelo RNN con el libro "The Great Gatsby" utilizando 3 temperaturas diferentes y 2 entradas en cada temperatura.

```
from google.colab import drive
drive.mount('/content/drive')
%cd "/content/drive/MyDrive/CONCENTRACION AI/data"
```

```
Mounted at /content/drive
/content/drive/MyDrive/CONCENTRACION AI/data
```

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

+ Código + Texto

Libro utilizado = "The Great Gatsby"

```
text = open('great_gatsby.txt', '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:    296579 caracteres
El texto esta compuesto de estos 96 caracteres
['\t', '\n', '\r', ' ', '!', '#', '$', '%', '(', ')', '*', ',', '-', '.', '/', '0', '1', '2', '3', '4', '5', '6', '7'
```

Tablas de traducción

```
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]))
```

```
'\t':   0,
'\n':   1,
'\r':   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,
  'a' :  56,
  'b' :  57,
```

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 The Great Gatsby\r\n'
    array([95, 47, 63, 60,  3, 43, 73, 70, 65, 60, 58, 75,  3, 34, 76, 75, 60,
           69, 57, 60, 73, 62,  3, 60, 29, 70, 70, 66,  3, 70, 61,  3, 47, 63,
           60,  3, 34, 73, 60, 56, 75,  3, 34, 56, 75, 74, 57, 80,  2,  1])
```

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 The Great Gatsby\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: The Great Gatsby\r\n\r\n\r\nAuthor: F. Scott Fitzgerald\r\n\r\nRelease date: January 17, 2021 [eBook '
    '#64317]\r\n\r\nLanguage: English\r\n\r\n\r\n\r\n\r\n*** START OF THE PROJECT GUTENBERG EBOOK THE GREAT GATSBY ***\r\n\r\
    '\t\t\t   The Great Gatsby\r\n\t\t\t\t  by\r\n\t\t\t F. Scott Fitzgerald\r\n\r\n\r\n                          Table
    'ntents\r\n\r\nI\r\nII\r\nIII\r\nIV\r\nV\r\nVI\r\nVII\r\nVIII\r\nIX\r\n\r\n\r\n                          Once aga
    '                          to\r\n                                  Zelda\r\n\r\n  Then wear the gold hat, if'
```

```
#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 The Great Gatsby\r\n    \r\nThis ebook is for the use of anyone ar
    Target data:  'The Project Gutenberg eBook of The Great Gatsby\r\n    \r\nThis ebook is for the use of anyone anywher
```

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

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

```
#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), dtyp
```

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)           24576

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

 dense (Dense)               (64, None, 96)            98400

=================================================================
Total params: 5369952 (20.48 MB)
Trainable params: 5369952 (20.48 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, 96) # (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)
```

```
[80  4  9 17  4 72 51 75 89 31 57 95 91 41 47 12 75 63 75 91 74 30 63 57
 40 27 45 29 15 44 26 58 28 89 60 48 75 58 87 13  4 73 87 93 80  8 58  6
 70 75 62 80 92 58 80 30 83 75 57 23 68 28 87 29 41 76 71 91 33 91  4 91
 89 43 15 85 55 53  2 87 75 80  1 55  0 70 93 44 70 12 75 48  6  7 39  5
 13 60  8 79]
```

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
45/45 [==============================] - 8s 96ms/step - loss: 3.2536
Epoch 2/50
45/45 [==============================] - 4s 82ms/step - loss: 2.6314
Epoch 3/50
45/45 [==============================] - 4s 72ms/step - loss: 2.3256
Epoch 4/50
45/45 [==============================] - 4s 74ms/step - loss: 2.1722
Epoch 5/50
45/45 [==============================] - 4s 74ms/step - loss: 2.0174
Epoch 6/50
45/45 [==============================] - 4s 74ms/step - loss: 1.9035
Epoch 7/50
45/45 [==============================] - 4s 73ms/step - loss: 1.8066
Epoch 8/50
45/45 [==============================] - 4s 75ms/step - loss: 1.7213
Epoch 9/50
45/45 [==============================] - 4s 82ms/step - loss: 1.6462
Epoch 10/50
45/45 [==============================] - 4s 73ms/step - loss: 1.5822
Epoch 11/50
45/45 [==============================] - 4s 76ms/step - loss: 1.5224
Epoch 12/50
45/45 [==============================] - 5s 90ms/step - loss: 1.4691
Epoch 13/50
45/45 [==============================] - 5s 77ms/step - loss: 1.4224
Epoch 14/50
45/45 [==============================] - 4s 77ms/step - loss: 1.3782
Epoch 15/50
45/45 [==============================] - 4s 78ms/step - loss: 1.3349
Epoch 16/50
45/45 [==============================] - 4s 78ms/step - loss: 1.2951
Epoch 17/50
45/45 [==============================] - 4s 77ms/step - loss: 1.2581
Epoch 18/50
45/45 [==============================] - 4s 88ms/step - loss: 1.2184
Epoch 19/50
45/45 [==============================] - 4s 78ms/step - loss: 1.1847
Epoch 20/50
45/45 [==============================] - 4s 79ms/step - loss: 1.1490
Epoch 21/50
45/45 [==============================] - 4s 81ms/step - loss: 1.1070
Epoch 22/50
45/45 [==============================] - 5s 81ms/step - loss: 1.0710
Epoch 23/50
45/45 [==============================] - 4s 80ms/step - loss: 1.0315
Epoch 24/50
45/45 [==============================] - 4s 83ms/step - loss: 0.9907
Epoch 25/50
45/45 [==============================] - 5s 79ms/step - loss: 0.9462
Epoch 26/50
45/45 [==============================] - 5s 89ms/step - loss: 0.9051
Epoch 27/50
```

```
45/45 [==============================] - 4s 79ms/step - loss: 0.8643
Epoch 28/50
45/45 [==============================] - 4s 77ms/step - loss: 0.8209
Epoch 29/50
45/45 [==============================] - 4s 78ms/step - loss: 0.7804
```

Generación de texto y de resultados con libro "The Great Gatsby":

```
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]))
```

Con 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))
```

Con palabra "gatsby"

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

```
gatsby’s house and twinkled
him in his bed at night. A universe of Carmission of this agreement shall niter and there looking at the advertisemer
his cheekben he gave Daisy her father, and there was a minute and asked if we had room
for him.”

Jordan smiled.

“He was probably bumming his way home. It was a ghost of patter of complete from some rushentiment, I followed
Daisy around a chain of connecting verandas to the porch in front of windshelf and confused. And looking at the
presbyte
```

Con palabra "daisy"

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

```
daisy for the jabiloty of some worn her face the shock—it must have killed her
instantly.”

“It was a strange coincidence,” I said.

“But it wasn’t a coincidence at the edge of
stale ideas as if his sturdy people approached him to say goodbye.

Jordan’s party were calling impatiently to her from the polo player,” said Tom pleasantly, “and
everything and left the large central bay, spat meditatively into the
garden. It was time I went back. While the rain fellass with Morto Wilson’s face,
d
```

Con 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))
```

Se utilizan las mismas dos palabras que con la primer temperatura:

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

```
gatsby’s broken heaps, was intensetively aware of his
possible taiders and gives much more. The fashion

“Oh, iny1-cign’t murrued in a curiof cold almost might stand the bround with my house one sunder
real small turned to me I hame familiar—fadowing electronic works, and pry as if
the virch yer night,” I exclaimed in surprise. I had extember in a warne a long back to
Daisy and
new that his front door and started right away. I saw him sounded life and had car on a would
along for his star turn
```

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

```
daisy for a moment that
my coucing books and looking from the car.
    Tom’s house
and the front door for a man who knew all about him
often. I knew his the king heave it alone.”

“She do you keep all rain about having ready to?

He began to clam it was in the bottom
one that I could st.”

I knew that first ned my his future and a butten
flashing better than that he had been first
and fatther we were
glistening dresses’ likely for a moment that my were sick; I stared at him and then dec
```

Con Temperatura = 1.5 y mismas dos palabras:

```
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 = 1.5

  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"gatsby"))
```

```
    gatsby held incessantly, finally we busy vealmoon. But we
    grove Nick free distribution everitatien a P7Hive'

    pull dontra-of Sspotoo Ogro—"

    Cotir Daisy or
    she came.

    "I can't ceach whellowing alone. I spatement everybody of the table
      * Bo the room a quawitive met …

    IX%d you seen him a littlet vinither entill up her kny?" Tom danger had gave me readia Will juddy thently-fthere'pay
    wropul with parages behind his my beforn at up on the strong of waysification
    had l
```

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

```
    daisy, tordive it havis money—7ou' e!" and the office He killed, she handway for Ge room
    tire, "calleet Gatsby's indithre year in his house. A brulbly bored on the lower glowing-mote—"

    "Will they do?" said Tom. "I'll send itsly. I'd been awar now. He was a
    perrisedingly from the "park histly then, I folloximbed the proved like an
    maining. "Theyever seet me up, and her voice was coll it—heven't under the uneventie
    of four grain
    forels face 'sl I heard someone mot."

    "You do." He told me with
```