BDAA-ICP6-wiki report : Yousef Almutairi
Date: 10/04/2021

1. What you learned in the ICP:

   In this ICP I learned how to build an Text Generation using Recurrent Neural Network algorithm for the songs dataset. In addition, I learned how to deal with the Gated Recurrent Unit (GRU) and Long Short Term Memory (LSTM) layers and see the performance of the training model.

2. ICP description what was the task you were performing:

   This task is about learning how the RNN works and generating the text of the trained dataset. I've changed the Shakespeare dataset to the songs dataset and see how the performance of the model as well as the predictions.

3. Challenges that you faced:

   I am still facing an issue with the usage limit of the Google Colab GPU.

4. Screen shots that shows the successful execution of each required step of your code:

   Hyper-parameter #1: (Increasing the BATCH_SIZE = 128):

   In the first change, I decreased the seq_length = 25, and increased the batch size to 128 in order to shuffle the data and pack it into the batches. Please note: this changes are still with the GRU layers.

```
[83] # Batch size
BATCH_SIZE = 128

# Buffer size to shuffle the dataset
# (TF data is designed to work with possibly infinite sequences,
# so it doesn't attempt to shuffle the entire sequence in memory. Instead,
# it maintains a buffer in which it shuffles elements).
BUFFER_SIZE = 1000

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

dataset
```

```
<BatchDataset shapes: ((128, 25), (128, 25)), types: (tf.int64, tf.int64)>
```

Build The Model

Use tf.keras.Sequential to define the model. For this simple example three layers are used to define our model:

tf.keras.layers.Embedding: The input layer. A trainable lookup table that will map the numbers of each character to a vector with embedding_dim dimensions;

tf.keras.layers.GRU: A type of RNN with size units=rnn_units (You can also use a LSTM layer here.)

tf.keras.layers.Dense: The output layer, with vocab_size outputs.

```
# Length of the vocabulary in chars
vocab_size = len(vocab)

# The embedding dimension
embedding_dim = 512

# Number of RNN units
rnn_units = 1000
```

BDAA-ICP6-wiki report : Yousef Almutairi
Date: 10/04/2021

I kept the EPOCHS as 10 but the loss was more than 3 and the prediction was encrypted. However, I increased the EPOCHS to see much better result.

```
EPOCHS=50
history = model.fit(dataset, epochs=EPOCHS, callbacks=[checkpoint_callback])
```

```
Epoch 22/50
17/17 [==============================] - 1s 69ms/step - loss: 1.3574
Epoch 23/50
17/17 [==============================] - 1s 69ms/step - loss: 1.2978
Epoch 24/50
17/17 [==============================] - 1s 68ms/step - loss: 1.2485
Epoch 25/50
17/17 [==============================] - 1s 68ms/step - loss: 1.1989
Epoch 26/50
17/17 [==============================] - 1s 72ms/step - loss: 1.1465
Epoch 27/50
17/17 [==============================] - 1s 69ms/step - loss: 1.0948
Epoch 28/50
17/17 [==============================] - 1s 68ms/step - loss: 1.0486
Epoch 29/50
17/17 [==============================] - 1s 69ms/step - loss: 1.0009
Epoch 30/50
17/17 [==============================] - 1s 66ms/step - loss: 0.9587
Epoch 31/50
17/17 [==============================] - 1s 67ms/step - loss: 0.9094
Epoch 32/50
17/17 [==============================] - 1s 70ms/step - loss: 0.8744
Epoch 33/50
17/17 [==============================] - 1s 69ms/step - loss: 0.8325
Epoch 34/50
17/17 [==============================] - 1s 70ms/step - loss: 0.7953
Epoch 35/50
17/17 [==============================] - 1s 69ms/step - loss: 0.7641
Epoch 36/50
17/17 [==============================] - 1s 68ms/step - loss: 0.7360
Epoch 37/50
17/17 [==============================] - 1s 68ms/step - loss: 0.7132
Epoch 38/50
17/17 [==============================] - 1s 69ms/step - loss: 0.6870
Epoch 39/50
17/17 [==============================] - 1s 68ms/step - loss: 0.6689
```

As we can see the prediction here is much better and give us some readable text.

```
print(generate_text(model, start_string=u"Grandma:  "))
```

```
Grandma:  (It's beginning to look a lot like Christmas Eve id to hear and blowin' in sister's bed;
Somebody stody good, just wait
I'd sta, do!
I some,
And to usleed likeray.

Let us bring him thir do the boys from home

The drees and toong;
Juht geese a-laying,
Five golden rings.
Four calling birds, three French hens, two turtle doves
And a bark from down.

O thn,
O If Christmas, my true love gave tone
To everyone
canebar.gif (5592 bytes)
Weistmas dinner
A ming chime in jingle bells sleigh bells in the snow, in As be sughtrust with, a crooked dirty just like you'r

Later on, dumplie a wide open rings.
Four calling birds, three French Hens,
Two Turtle Doves
And a Partridge in a Pear Tree.

On the heixty stup If all things and dates
And is all: Refrain

O come, Six Geese-a-Laying,
FiveGod is soon (tho 194

How have a holly that will be good whate so face and all labain,
With all of the folks out a do bring us a figger hsmerry Christmas to you."

canebar.gif (5592 bytes)
```

Hyper-parameter #2: (Using LSTM instead of GRU):
In this change, I tried to see the performance of the model as well as how the model would generate the text at the end. I added one dropout layer in order to reduce the overfitting and enhance the performance.

```
[47] # Length of the vocabulary in chars
     vocab_size = len(vocab)

     # The embedding dimension
     embedding_dim = 128

     # Number of RNN units
     rnn_units = 1024
```

```
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,
                         activation='tanh',
                        return_sequences=True,
                        stateful=True,
                        recurrent_initializer='glorot_uniform'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.LSTM(rnn_units,
                         activation='tanh',
                        return_sequences=True,
                        stateful=True,
                        recurrent_initializer='glorot_uniform'),
    tf.keras.layers.Dense(vocab_size)
  ])
  return model
```

```
[49] model = build_model(
         vocab_size = len(vocab),
         embedding_dim=embedding_dim,
         rnn_units=rnn_units,
         batch_size=BATCH_SIZE)
```

Fit the model:

BDAA-ICP6-wiki report : Yousef Almutairi
Date: 10/04/2021

```
EPOCHS=10
history = model.fit(dataset, epochs=EPOCHS, callbacks=[checkpoint_callback])
```

```
Epoch 1/10
9/9 [==============================] - 6s 356ms/step - loss: 3.7749
Epoch 2/10
9/9 [==============================] - 3s 350ms/step - loss: 3.2858
Epoch 3/10
9/9 [==============================] - 3s 338ms/step - loss: 3.2630
Epoch 4/10
9/9 [==============================] - 3s 338ms/step - loss: 3.2495
Epoch 5/10
9/9 [==============================] - 3s 339ms/step - loss: 3.2269
Epoch 6/10
9/9 [==============================] - 3s 346ms/step - loss: 3.1748
Epoch 7/10
9/9 [==============================] - 3s 344ms/step - loss: 3.1239
Epoch 8/10
9/9 [==============================] - 3s 347ms/step - loss: 3.0081
Epoch 9/10
9/9 [==============================] - 3s 340ms/step - loss: 2.9437
Epoch 10/10
9/9 [==============================] - 3s 339ms/step - loss: 2.8922
```

Generated text:

```
print(generate_text(model, start_string=u"Grandma:  "))
```

```
Grandma:  loris haorietlbpiys pv2 Chahine.

nrh tappadeed t wh5-s snnt,tr bv Phiseal irh, fnhtn ch wereln;r'!u
Naoe nadt tf aors'Ofe

fy s Ietbe hhiany taires,i I j5k fels, da cha wooel,
Twaeeiy Wn tns'r ) oaeg'c
Te poreb-mg-th avmie
hytinovo ou Tnee) bihta toms

are
la ses Chen hoerngid
w! thge
gov huu
oore burr thie inr hentce Whres to Chhen, Mee dh Oos CcnoiM, CTHarmtns, tf irtleg mou yege
tenw Ins
Oenums rnis tee dsy yil dm ait yoe
me Cfoitets I cirnce.
anatd s Oongoo. tievn a nen
iega,t
iTtu ohaem) Wo arnye wm bin Thlre tenmo'n salyrme
Hr to ier.y ton Toan Loal Cmrutsne
huisd Aa nn Cphetitd aeol ya gneus

Ctuihitl
m Ze haly,,, He, wyaSNtne ruEi aopntg
w joh hhtt ertart
Saten shn boho
Lwo evnn te Ha tos hon toi arsgd nytsl-lnn bew bhr soe maees ae metd thrdnnng
Yaus wy wank hhg padingn
Ivb"d Batls giie atuy Salrts Ta ah lisge chune lln a lfl
y'nn,
uonns matty ksa, bitons!
Gheytn he ttlchenaeg
I s wfed agg iiake, syes
ts ay baee Shh tan toal Cmtdtum, soteos
lh Oope asy avtp
```

Hyper-parameter #3: (EPOCHS = 100):

I increased the epochs to make the model generalize better as see how the model will predict with loss less than 0.20.

```
EPOCHS=100
history = model.fit(dataset, epochs=EPOCHS, callbacks=[checkpoint_callback])
```

```
Epoch 72/100
9/9 [==============================] - 3s 345ms/step - loss: 0.2117
Epoch 73/100
9/9 [==============================] - 3s 350ms/step - loss: 0.2106
Epoch 74/100
9/9 [==============================] - 3s 343ms/step - loss: 0.2053
Epoch 75/100
9/9 [==============================] - 3s 347ms/step - loss: 0.2054
Epoch 76/100
9/9 [==============================] - 3s 356ms/step - loss: 0.1991
Epoch 77/100
9/9 [==============================] - 3s 349ms/step - loss: 0.1965
Epoch 78/100
9/9 [==============================] - 3s 355ms/step - loss: 0.1963
Epoch 79/100
9/9 [==============================] - 3s 350ms/step - loss: 0.1916
Epoch 80/100
9/9 [==============================] - 3s 347ms/step - loss: 0.1898
Epoch 81/100
9/9 [==============================] - 3s 352ms/step - loss: 0.1893
Epoch 82/100
9/9 [==============================] - 3s 348ms/step - loss: 0.1894
Epoch 83/100
9/9 [==============================] - 3s 347ms/step - loss: 0.1824
Epoch 84/100
9/9 [==============================] - 3s 341ms/step - loss: 0.1803
Epoch 85/100
9/9 [==============================] - 3s 341ms/step - loss: 0.1819
Epoch 86/100
9/9 [==============================] - 3s 344ms/step - loss: 0.1747
Epoch 87/100
9/9 [==============================] - 3s 347ms/step - loss: 0.1747
Epoch 88/100
9/9 [==============================] - 3s 348ms/step - loss: 0.1729
Epoch 89/100
9/9 [==============================] - 3s 349ms/step - loss: 0.1714
Epoch 90/100
9/9 [==============================] - 3s 351ms/step - loss: 0.1672
Epoch 91/100
9/9 [==============================] - 3s 352ms/step - loss: 0.1652
```

As a result, the model has generate a text that are too close to the original one and **readable**.

```
print(generate_text(model, start_string=u"Grandma:  "))
```

```
Grandma:  on the Horids
Through the nightomaand to withole bells, Jingle Bells,
Jingle all the way!
What fun it is to ride
In a one-horse open sleigh.

A campfrieg to you"

canebar.gif (5592 bytes)
Blieeurs,
Fy hope a-Marsilide in the scenith The!
I Hoa o dans
"Lar the belds the Lace,
Out that is that Morn to hear
Ald played the table and joy come to you,
And to your wassail too,
And God bless you and send you a happy new year,
And God send you a happy new year.

canebar.gif (5592 bytes)
Cand I Hab Mudtmifor on husted be moll that we Christmas)

Merry, Merry, Merry, Merry Christmas
(Merry, Merry, Merry Christmas.

On on they send
On without end
Their joyful tone
To every home

Ah! Ah! Ah!

La, da, da, da, da,
La, da, da, da, da,
La, da, da, da, da...

Hark how the bells
Sweet silver bells
All seem to say, (All seem to say)
Throw cares away)
```

BDAA-ICP6-wiki report : Yousef Almutairi
Date: 10/04/2021

Hyper-parameter #4: (Length Sentence = 100, Embedding Dimension = 256)
Since I do have a large number of the words in the dataset, I tried to increase the embedding in order to map the numbers of each character to a vector.

ls Help All changes saved

+ Code + Text

[85]
```
# Batch size
BATCH_SIZE = 128

# Buffer size to shuffle the dataset
# (TF data is designed to work with possibly infinite sequences,
# so it doesn't attempt to shuffle the entire sequence in memory. Instead,
# it maintains a buffer in which it shuffles elements).
BUFFER_SIZE = 1024

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

dataset
```

```
<BatchDataset shapes: ((128, 100), (128, 100)), types: (tf.int64, tf.int64)>
```

Build The Model

Use tf.keras.Sequential to define the model. For this simple example three layers are used to define our model:

tf.keras.layers.Embedding: The input layer. A trainable lookup table that will map the numbers of each character to a vector with embedding_dim dimensions;

tf.keras.layers.GRU: A type of RNN with size units=rnn_units (You can also use a LSTM layer here.)

tf.keras.layers.Dense: The output layer, with vocab_size outputs.

[86]
```
# Length of the vocabulary in chars
vocab_size = len(vocab)

# The embedding dimension
embedding_dim = 256

# Number of RNN units
rnn_units = 1024
```

Model fitting:

```
EPOCHS=30
history = model.fit(dataset, epochs=EPOCHS, callbacks=[checkpoint_callback])
```

```
Epoch 2/30
4/4 [==============================] - 2s 592ms/step - loss: 3.6827
Epoch 3/30
4/4 [==============================] - 3s 606ms/step - loss: 3.3152
Epoch 4/30
4/4 [==============================] - 3s 623ms/step - loss: 3.2859
Epoch 5/30
4/4 [==============================] - 2s 595ms/step - loss: 3.2654
Epoch 6/30
4/4 [==============================] - 3s 613ms/step - loss: 3.2623
Epoch 7/30
4/4 [==============================] - 2s 598ms/step - loss: 3.2539
Epoch 8/30
4/4 [==============================] - 3s 620ms/step - loss: 3.2459
Epoch 9/30
4/4 [==============================] - 3s 620ms/step - loss: 3.2441
Epoch 10/30
4/4 [==============================] - 3s 623ms/step - loss: 3.2333
Epoch 11/30
4/4 [==============================] - 3s 606ms/step - loss: 3.2150
Epoch 12/30
4/4 [==============================] - 2s 603ms/step - loss: 3.1971
Epoch 13/30
4/4 [==============================] - 3s 620ms/step - loss: 3.1547
Epoch 14/30
4/4 [==============================] - 3s 626ms/step - loss: 3.1075
Epoch 15/30
4/4 [==============================] - 3s 621ms/step - loss: 3.0477
Epoch 16/30
4/4 [==============================] - 3s 618ms/step - loss: 2.9899
Epoch 17/30
4/4 [==============================] - 2s 603ms/step - loss: 2.9472
Epoch 18/30
4/4 [==============================] - 3s 612ms/step - loss: 2.9177
Epoch 19/30
4/4 [==============================] - 3s 627ms/step - loss: 2.8757
Epoch 20/30
4/4 [==============================] - 3s 624ms/step - loss: 2.8420
Epoch 21/30
4/4 [==============================] - 3s 629ms/step - loss: 2.8712
Epoch 22/30
```

BDAA-ICP6-wiki report : Yousef Almutairi
Date: 10/04/2021

The result of the changes that I made:

```
print(generate_text(model, start_string=u"Grandma:  "))
```

```
Grandma:
Fin hho cearif,
An n be b yive Teau-aevas rhecpin he, sud thoutm hrlid.
Jcennls tey thent a-e, tou dor
Leok,

Olot un're phe boneaa Cheismcas ssoY,
Ind weiclt Chrinsmas, Mering,
Ins Thins Youd whrwzs tes Chrrlwiss Endin ye ky srishingigg bloy Christafam.s a Manpy Tint, Sombd weunb
Ang (Chrittes ope Jing hing on doond deirdhd;s he Wond buss Ind roud 1teice boud Marktin'igh tin
hirh Dhat ifis'er wagd wong the srarinedeif In mere,
Radr) janbr the bre canl hinte Rige
(Fecerlung in opor And go sol fon mewipe.
sing
the te thule guadl wo tre rhacss yand seay jonide hod 1li4g, if Min'-uf ca Lelk aot,

Fic toy and Chringmass a welay Bimlte  Illsy nu Migh hol wuvr toulre,.
Dhignl.
Perlad, Laty Dinfty re, And bavath Ood grincling
Forrlmar Af Purczwoy thony
Bhe soce da,d
Ay hourh
se, all pearas,
Oney hhe we, Hnud a Syore-Gtandsin and woonlf bemnr, fing san wo, hronge durtem,
(Hey yar bole uu worey andh ther
If'p wris
Hre sow in weanr the is polg Chrubmmma te d ainging

Sloowe Pinte ytorechonn l
```

5. Video link:

https://drive.google.com/file/d/1i5zA-r5YeHy8fBMG_uRJ3VOEHwj36N2d/view?usp=sharing