Mehrdad Baradaran

Assignment 5

99222020

Persian WikiPedia dataset part 1

Text Prediction

First of all, we extract the required libraries:

```
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np
import platform
import time
import pathlib
import os

print('Python version:', platform.python_version())
print('Tensorflow version:', tf.__version__)
print('Keras version:', tf.keras.__version__)
```

After that, we will load the Persian wikitext data set for analysis in the next parts:

```
!unzip '/content/Persian-WikiText-1.txt.zip'
```

```
Archive:  /content/Persian-WikiText-1.txt.zip
replace Persian-WikiText-1.txt? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
  inflating: Persian-WikiText-1.txt
```

```
filename = '/content/Persian-WikiText-1.txt'
text = open(filename, 'r', encoding='utf-8').read()
```

First, we display a part of the beginning of the text to see the format of the model that we loaded with the utf-8 encoder. In the next part, we must specify the number of unique characters in order to find out how many different characters there are in our entire text.

Before we process all the text and perform data preparation on them, we need to separate only a part of the text and make it as desired, for example here, because there are English capital letters, use the lower() function. We use it so that all letters are lowercase and then due to the presence of the \n character, we have to delete them so that the data is ready to be read.

```
print(text[:1000])
```

```
عنوان مقاله: صفحهٔ اصلی

/صفحه اصلی"=templatestyles src>



عنوان مقاله: ویکی پدیا

ویکی پدیا (کوته نوشت به صورت «وپ» و «WP») یک دانشنامه برخط چندزبانه مبتنی بر وب با محتوای آزاد و همکاری باز است که با همکاری افراد داوطلب نوشته می شود و هر کسی که به اینترنت و وب دسترسی داشته باشد می تواند مقالات آن را ببیند و ویرایش کند. نام ویکی پدیا واژه ای ترکیبی است که از واژه های ویکی (وبگاه مشارکتی) و اِنسایکلوپدیا (Encyclopedia) (دانشنامه یا دائرةالمعارف) گرفته شده است. هدف ویکی پدیا آفرینش و انتشار جهانی یک دانشنامه با محتوای آزاد به تمامی زبان های زندهٔ دنیا است.

ویکی پدیای انگلیسی در تاریخ ۱۵ ژانویه ۲۰۰۱ (۲۶ دی ۱۳۷۹) به صورت مکملی برای دانشنامهٔ تخصصی نیوپدیا نوشته شد. بنیان گذاران آن «جیمی ویلز» و «لری سنگر» هستند. هم اکنون بنیاد غیرانتفاعی ویکی مدیا پروژهٔ ویکی پدیا را پشتیبانی می کند. میزبان های اینترنتی اصلی این وبگاه در شهر تامپای فلوریدا هستند. همچنین میزبان های اضافی دیگری هم د ر شهرهای آمستردام و سئول به این وبگاه یاری می رسانند.

"ویکی پدیا" از پایان آوریل ۲۰۰۷ تا اکتبر ۲۰۱۹، یکی از
```

```
text = text[:1000000]
text = text.lower()
text = text.replace('\n', ' ')
```

```
vocab = sorted(set(text))

print('{} unique characters'.format(len(vocab)))
print('vocab:', vocab)
```

```
193 unique characters
vocab: [' ', '!', '"', "'", '(', ')', '+', ',', '-', '.', '/', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', ':', ';',
'<', '=', '>', '[', ']', '^', '_', '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', '{', '|', '}', '\xa0', '«', '»', '×', 'à', 'á', 'ã', 'ä', 'ç', 'é', 'ê', 'î',
'ā', 'ć', 'ē', 'ī', 'ń', 'ō', 'ś', 'ş', 'š', '‘', 'ά', 'ή', 'ί', 'α', 'β', 'γ', 'δ', 'ε', 'θ', 'ι', 'λ', 'μ', 'ν', 'ξ',
'ο', 'ρ', 'ς', 'τ', 'ω', 'а', 'е', 'и', 'м', 'н', 'п', 'р', 'т', 'у', 'я', 'ا', 'ب', 'ة', 'ت', 'ج', 'ح', 'خ', 'د', 'ذ', 'ر', 'ز', 'س', 'ش', 'ص', 'ض', 'ط', 'ظ', 'ع', 'غ', 'ف', 'ق', 'ل', 'م', 'ن',
'ه', 'و', 'ي', 'ٔ', 'ٕ', 'ٖ', 'ٗ', '٘', 'ٙ', 'ٚ', 'ٛ', 'ٜ', 'ٞ', 'ٟ', '٠', '١', '٢', '٣', '٤', '٥',
'٦', '٧', '٨', '٩', 'ٮ', 'ٯ', 'ٱ', 'ٲ', 'ٳ', 'ٴ', 'ٵ', 'ٶ', 'ٷ', 'ٸ', 'ٹ', 'ٺ', 'ٻ', 'پ', 'ٽ', 'ٿ', 'ڀ', 'ڃ', 'چ', 'ڇ', 'ڈ', 'ډ', 'ڊ', 'ڌ', 'ڍ', 'ڎ', 'ڏ', 'ڑ', 'ڒ', 'ړ', 'ڔ', 'ڕ', 'ږ', 'ڗ', 'ژ', 'ڙ', 'ښ', 'ڛ', 'ڜ', 'ڝ', 'ڞ', 'ڟ', 'ڠ', 'ڡ', 'ڢ', 'ڣ', 'ڤ', 'ڥ', 'ڦ', 'ڧ', 'ڨ', 'ک', 'ڪ', 'ګ', 'ڬ', 'ڭ', 'ڮ', 'گ', 'ڰ', 'ڱ', 'ڲ', 'ڳ', 'ڴ', 'ڵ', 'ڶ', 'ڷ', 'ڸ', 'ڹ', 'ں', 'ڻ', 'ڼ', 'ڽ', 'ھ', 'ڿ', 'ۀ', 'ہ', 'ۂ', 'ۃ', 'ۄ', 'ۅ', 'ۆ', 'ۇ', 'ۈ', 'ۉ', 'ۊ', 'ۋ', 'ی', 'ۍ', 'ێ', 'ۏ', 'ې', 'ۑ', 'ے', 'ۓ', 'ە', 'ۮ', 'ۯ', 'ۺ', 'ۻ', 'ۼ', 'ۿ', '\u200d', '\u200e', '-', '–', ',', '‘', '’', '“', '”', '„', '…', '→', '∞', 'الله']
```

After that, it is time to decode and encode text characters.

We should be able to have a list of the compatibility of each character with an integer number and its image, so that after encoding the text and giving it to the model, we can take the output and convert it into a format that can be read by decoding.

```
char_to_int = {char: index for index, char in enumerate(vocab)}
```

```
int_to_char = np.array(vocab)
print(int_to_char)
```

```
[' ' '!' '"' "'" '(' ')' '+' ',' '-' '.' '/' '0' '1' '2' '3' '4' '5' '6'
 '7' '8' '9' ':' ';' '<' '=' '>' '[' ']' '^' '_' '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' '{' '|' '}' '\xa0' '«' '»' '×' 'à' 'á' 'ã' 'ä' 'ç' 'é' 'ê' 'î'
 'ā' 'ć' 'ē' 'ī' 'ń' 'ō' 'ś' 'ş' 'š' '‘' 'ά' 'ή' 'ί' 'α' 'β' 'γ' 'δ' 'ε'
 'θ' 'ι' 'λ' 'μ' 'ν' 'ξ' 'ο' 'ρ' 'ς' 'τ' 'ω' 'а' 'е' 'и' 'м' 'н' 'п' 'р'
 'т' 'у' 'я' 'ا' 'ب' 'ة' 'ت' 'ج' 'ح' 'خ' 'د' 'ذ' 'ر' 'ز' 'س' 'ش' 'ص'
 'ض' 'ط' 'ظ' 'ع' 'غ' 'ف' 'ق' 'ل' 'م' 'ن' 'ه' 'و' 'ي' 'ٔ' 'ٕ' 'ٖ' 'ٗ' '٘'
 'ٙ' 'ٚ' 'ٛ' 'ٜ' 'ٞ' 'ٟ' '٠' '١' '٢' '٣' '٤' '٥' '٦' '٧' '٨' '٩' 'ٮ'
 'ٯ' 'ٱ' 'ٲ' 'ٳ' 'ٴ' 'ٵ' 'ٶ' 'ٷ' 'ٸ' 'ٹ' 'ٺ' 'ٻ' 'پ' 'ٽ' 'ٿ' 'ڀ' 'ڃ'
 'چ' 'ڇ' 'ڈ' 'ډ' 'ڊ' 'ڌ' 'ڍ' 'ڎ' 'ڏ' 'ڑ' 'ڒ' 'ړ' 'ڔ' 'ڕ' 'ږ' 'ڗ' 'ژ'
 'ڙ' 'ښ' 'ڛ' 'ڜ' 'ڝ' 'ڞ' 'ڟ' 'ڠ' 'ڡ' 'ڢ' 'ڣ' 'ڤ' 'ڥ' 'ڦ' 'ڧ' 'ڨ' 'ک'
 'ڪ' 'ګ' 'ڬ' 'ڭ' 'ڮ' 'گ' 'ڰ' 'ڱ' 'ڲ' 'ڳ' 'ڴ' 'ڵ' 'ڶ' 'ڷ' 'ڸ' 'ڹ' 'ں'
 'ڻ' 'ڼ' 'ڽ' 'ھ' 'ڿ' 'ۀ' 'ہ' 'ۂ' 'ۃ' 'ۄ' 'ۅ' 'ۆ' 'ۇ' 'ۈ' 'ۉ' 'ۊ' 'ۋ'
 'ی' 'ۍ' 'ێ' 'ۏ' 'ې' 'ۑ' 'ے' 'ۓ' 'ە' 'ۮ' 'ۯ' 'ۺ' 'ۻ' 'ۼ' 'ۿ' '\u200d'
 '\u200e' '-' '–' ',' '‘' '’' '“' '”' '„' '…' '→' '∞' 'الله']
```

Now we can convert the entire text into its numerical format:

```
text_as_int = np.array([char_to_int[char] for char in text])

print('text_as_int length: {}'.format(len(text_as_int)))
print('{} --> {}'.format(repr(text[:15]), repr(text_as_int[:15])))
```

```
text_as_int length: 1000000
'عنوان مقاله: صنف ' --> array([136, 143, 145, 118, 143,   0, 142, 140, 118, 141, 144,  21,   0,
       132, 139])
```

Also note that every character will be assigned a unique value.

Given a character, or a sequence of characters, what is the most probable next character? This is the task the model to perform. The input to the model will be a sequence of characters, and we train the model to predict the output—the following character at each time step.

Now we have to Create training sequences.

```
sequence_length = 100
examples_per_epoch = len(text) // (sequence_length + 1)

print('examples_per_epoch:', examples_per_epoch)
```

```
examples_per_epoch: 9900
```

After that, we convert the text converted into numerical format into tensorial representations and then display the first 5 elements.

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

for char in char_dataset.take(5):
    print(int_to_char[char.numpy()])
```

```
ع
ن
و
ا
ن
```

```
for char in char_dataset.take(5):
    print(char.numpy())
```

```
136
143
145
118
143
```

Now we convert this sequence of characters into sentences with batch size

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

print('Sequences count: {}'.format(len(list(sequences.as_numpy_iterator()))));
print()

for item in sequences.take(3):
    print(repr(''.join(int_to_char[item.numpy()])))
```

```
Sequences count: 9900

'صفحه اصلی/  عنوان مقاله: ویکی پدیا  ویکی پدیا (کوته ن="src" templatestyles>  عنوان مقاله: صفحهٔ اصلی'
' یک دانشنامه برخط چندزبانه مبتنی بر وب با محتوای آزاد و همکاری باز است که با («wp» وشت به صورت «وب» و'
' همکاری افراد داوطلب نوشته می شود و هر کسی که به اینترنت و وب دسترسی داشته باشد می تواند مقالات آن را'
```

For each sequence, duplicate and shift it to form the input and target text.

```
def split_input_target(chunk):
    input_text = chunk[:-1]
    target_text = chunk[1:]
    return input_text, target_text
```

```
dataset = sequences.map(split_input_target)

print('dataset size: {}'.format(len(list(dataset.as_numpy_iterator()))))
```

```
dataset size: 9900
```

Print the five first example input and target values:

```
for input_example, target_example in dataset.take(5):
    print('Input sequence size:', repr(len(input_example.numpy())))
    print('Target sequence size:', repr(len(target_example.numpy())))
    print()
    print('Input:', repr(''.join(int_to_char[input_example.numpy()])))
    print('Target:', repr(''.join(int_to_char[target_example.numpy()])))
```

```
Input sequence size: 100
Target sequence size: 100

Input: 'صفحه اصلی/  عنوان مقاله: ویکی پدیا  ویکی پدیا (کوته="src" templatestyles>  عنوان مقاله: صفحهٔ اصلی'
Target: 'عنوان مقاله: صفحهٔ اصلی  نوان مقاله: ویکی پدیا  ویکی پدیا (کوته ن="src" templatestyles>  عنوان مقاله: ویکی پدیا  ویکی پدیا (کوته ن="src" templatestyles>  عنوان مقاله: صفحهٔ اصلی/'
Input sequence size: 100
Target sequence size: 100

Input: 'یک دانشنامه برخط چندزبانه مبتنی بر وب با محتوای آزاد و همکاری باز است که با («wp» وشت به صورت «وب» و'
Target: ' یک دانشنامه برخط چندزبانه مبتنی بر وب با محتوای آزاد و همکاری باز است که با («wp» شت به صورت «وب» و'
Input sequence size: 100
Target sequence size: 100

Input: 'همکاری افراد داوطلب نوشته می شود و هر کسی که به اینترنت و وب دسترسی داشته باشد می تواند مقالات آن را'
Target: ' مکاری افراد داوطلب نوشته می شود و هر کسی که به اینترنت و وب دسترسی داشته باشد می تواند مقالات آن را'
Input sequence size: 100
Target sequence size: 100

Input: 'ببیند و ویرایش کند. نام ویکی پدیا واژه ای ترکیبی است که از واژه های ویکی (ویگاه مشارکتی) و اِنسایکلو'
Target: 'بیند و ویرایش کند. نام ویکی پدیا واژه ای ترکیبی است که از واژه های ویکی (ویگاه مشارکتی) و اِنسایکلوپ'
Input sequence size: 100
Target sequence size: 100

Input: 'دیا (encyclopedia) (دانشنامه یا دائرةالمعارف) یک  هدف ویکی پدیا آفرینش و انتشار جهانی گرفته شده است. '
Target: 'یا (encyclopedia) (دانشنامه یا دائرةالمعارف) یک د  هدف ویکی پدیا آفرینش و انتشار جهانی گرفته شده است. '
```

Split training sequences into batches

```
BATCH_SIZE = 64

BUFFER_SIZE = 10000

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

dataset
```

```
<BatchDataset element_spec=(TensorSpec(shape=(64, 100), dtype=tf.int64, name=None), TensorSpec(shape=(64, 100), dtype=tf.in
t64, name=None))>
```

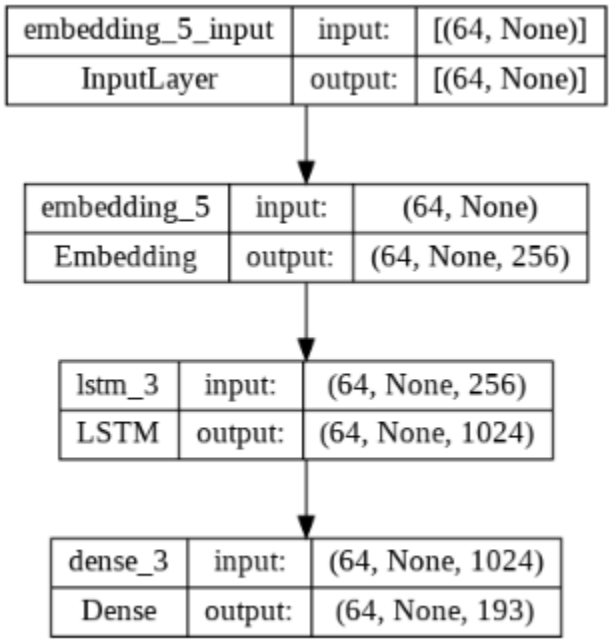
Now, after processing the text and data, we need to create our recurrent network model

```
def build_model(vocab_size, embedding_dim, rnn_units, batch_size):
    model = tf.keras.models.Sequential()

    model.add(tf.keras.layers.Embedding(
      input_dim=vocab_size,
      output_dim=embedding_dim,
      batch_input_shape=[batch_size, None]
    ))

    model.add(tf.keras.layers.LSTM(
      units=rnn_units,
      return_sequences=True,
      stateful=True,
      recurrent_initializer=tf.keras.initializers.GlorotNormal()
    ))

    model.add(tf.keras.layers.Dense(vocab_size))
    return model
```

After that, we compile the model and set the loss function for perplexity. The summary of the model is as follows:

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_5 (Embedding)     (64, None, 256)           49408

 lstm_3 (LSTM)               (64, None, 1024)          5246976

 dense_3 (Dense)             (64, None, 193)           197825

=================================================================
Total params: 5,494,209
Trainable params: 5,494,209
Non-trainable params: 0
_________________________________________________________________
```

Graphical form of model:



Loss and optimizer for model :

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

```
adam_optimizer = tf.keras.optimizers.Adam(learning_rate=0.001)
model.compile(
    optimizer=adam_optimizer,
    loss=loss
)
```

Use a tf.keras.callbacks.ModelCheckpoint to ensure that checkpoints are saved during training:

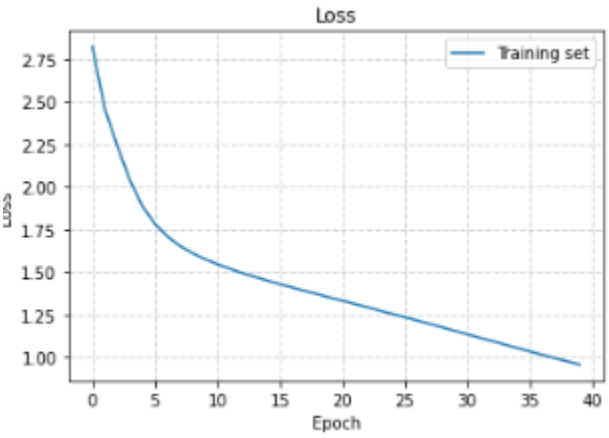
```
checkpoint_dir = 'tmp/checkpoints'
os.makedirs(checkpoint_dir, exist_ok=True)

checkpoint_prefix = os.path.join(checkpoint_dir, 'LSTM_Model_{epoch}')

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

After that, we fit the model with 40 epochs and the model starts training.

```
Epoch 1/40
154/154 [==============================] - 13s 71ms/step - loss: 2.8213
Epoch 2/40
154/154 [==============================] - 12s 73ms/step - loss: 2.4571
Epoch 3/40
154/154 [==============================] - 12s 73ms/step - loss: 2.2384
Epoch 4/40
154/154 [==============================] - 12s 72ms/step - loss: 2.0398
Epoch 5/40
154/154 [==============================] - 12s 69ms/step - loss: 1.8873
Epoch 6/40
154/154 [==============================] - 12s 72ms/step - loss: 1.7830
Epoch 7/40
154/154 [==============================] - 13s 74ms/step - loss: 1.7095
Epoch 8/40
154/154 [==============================] - 12s 74ms/step - loss: 1.6536
Epoch 9/40
154/154 [==============================] - 12s 72ms/step - loss: 1.6115
Epoch 10/40
154/154 [==============================] - 12s 71ms/step - loss: 1.5764
Epoch 11/40
154/154 [==============================] - 12s 71ms/step - loss: 1.5466
Epoch 12/40
154/154 [==============================] - 12s 71ms/step - loss: 1.5199
Epoch 13/40
154/154 [==============================] - 12s 73ms/step - loss: 1.4946
Epoch 14/40
154/154 [==============================] - 13s 76ms/step - loss: 1.4720
Epoch 15/40
154/154 [==============================] - 12s 71ms/step - loss: 1.4505
Epoch 16/40
154/154 [==============================] - 12s 71ms/step - loss: 1.4296
Epoch 17/40
154/154 [==============================] - 12s 73ms/step - loss: 1.4097
Epoch 18/40
154/154 [==============================] - 12s 73ms/step - loss: 1.3897
Epoch 19/40
154/154 [==============================] - 12s 73ms/step - loss: 1.3710
Epoch 20/40
154/154 [==============================] - 12s 72ms/step - loss: 1.3504
Epoch 21/40
154/154 [==============================] - 12s 71ms/step - loss: 1.3327
Epoch 22/40
154/154 [==============================] - 12s 70ms/step - loss: 1.3130
Epoch 23/40
154/154 [==============================] - 12s 72ms/step - loss: 1.2942
Epoch 24/40
154/154 [==============================] - 13s 74ms/step - loss: 1.2736
Epoch 25/40
154/154 [==============================] - 12s 72ms/step - loss: 1.2550
Epoch 26/40
154/154 [==============================] - 12s 71ms/step - loss: 1.2353
Epoch 27/40
154/154 [==============================] - 12s 72ms/step - loss: 1.2159
Epoch 28/40
154/154 [==============================] - 12s 72ms/step - loss: 1.1962
Epoch 29/40
154/154 [==============================] - 12s 71ms/step - loss: 1.1760
```

```
Epoch 30/40
154/154 [==============================] - 12s 72ms/step - loss: 1.1558
Epoch 31/40
154/154 [==============================] - 12s 72ms/step - loss: 1.1352
Epoch 32/40
154/154 [==============================] - 12s 73ms/step - loss: 1.1149
Epoch 33/40
154/154 [==============================] - 12s 72ms/step - loss: 1.0956
Epoch 34/40
154/154 [==============================] - 12s 71ms/step - loss: 1.0752
Epoch 35/40
154/154 [==============================] - 12s 72ms/step - loss: 1.0550
Epoch 36/40
154/154 [==============================] - 12s 74ms/step - loss: 1.0352
Epoch 37/40
154/154 [==============================] - 12s 71ms/step - loss: 1.0149
Epoch 38/40
154/154 [==============================] - 12s 71ms/step - loss: 0.9952
Epoch 39/40
154/154 [==============================] - 12s 73ms/step - loss: 0.9772
Epoch 40/40
154/154 [==============================] - 12s 71ms/step - loss: 0.9577
```

Lets plot this:



Finally, we create a new model and load the latest weights and updates of the previous model into it so that we can start the prediction. A sequence of characters starts to predict the next words using its previous learning

```
predict_characters(model, 80, 'مرز ایران')
```

مرز ایرانی را در مورد بررسی می شود. به همین دلیل نیز از سال ۱۳۸۳ خورشیدی و انجمن مهندسان

As it is clear, the structure of the sentence is followed correctly, the sentences have subject and verb, and most of the created words are meaningful, and compared to the logical sentences, this model has succeeded in predicting.