MP4_P2_generation

November 23, 2020

1 Generating Text with an RNN

[1]:
```
from google.colab import drive
drive.mount('/content/gdrive')
import os
os.chdir("gdrive/My Drive/assignment4")
```

```
Mounted at /content/gdrive
```

[2]:
```
!pip install unidecode
```

```
Collecting unidecode
  Downloading https://files.pythonhosted.org/packages/d0/42/d9edfed04228ba
cea2d824904cae367ee9efd05e6cce7ceaaedd0b0ad964/Unidecode-1.1.1-py2.py3-none-
any.whl (238kB)
     |                  | 245kB 7.6MB/s
Installing collected packages: unidecode
Successfully installed unidecode-1.1.1
```

[3]:
```
import unidecode
import string
import random
import re
import time

import torch
import torch.nn as nn

%matplotlib inline

%load_ext autoreload
%autoreload 2
```

[4]:
```
from rnn.model import RNN
from rnn.helpers import time_since
from rnn.generate import generate
```

[5]:
```
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
```

1.1 Data Processing

The file we are using is a plain text file. We turn any potential unicode characters into plain ASCII by using the `unidecode` package (which you can install via `pip` or `conda`).

[6]:
```
all_characters = string.printable
n_characters = len(all_characters)

file_path = './shakespeare.txt'
file = unidecode.unidecode(open(file_path).read())
file_len = len(file)
print('file_len =', file_len)

# we will leave the last 1/10th of text as test
split = int(0.9*file_len)
train_text = file[:split]
test_text = file[split:]

print('train len: ', len(train_text))
print('test len: ', len(test_text))
```

```
file_len = 4573338
train len:  4116004
test len:  457334
```

[7]:
```
chunk_len = 200

def random_chunk(text):
    start_index = random.randint(0, len(text) - chunk_len)
    end_index = start_index + chunk_len + 1
    return text[start_index:end_index]

print(random_chunk(train_text))
```

```
flight,
Added to their familiarity,
Which was as gross as ever touch'd conjecture,
That lack'd sight only, nought for approbation
But only seeing, all other circumstances
Made up to the deed, doth push
```

1.1.1 Input and Target data

To make training samples out of the large string of text data, we will be splitting the text into chunks.

Each chunk will be turned into a tensor, specifically a `LongTensor` (used for integer values), by looping through the characters of the string and looking up the index of each character in `all_characters`.

[8]:
```
# Turn string into list of longs
def char_tensor(string):
    tensor = torch.zeros(len(string), requires_grad=True).long()
    for c in range(len(string)):
        tensor[c] = all_characters.index(string[c])
    return tensor
```

The following function loads a batch of input and target tensors for training. Each sample comes from a random chunk of text. A sample input will consist of all characters *except the last*, while the target wil contain all characters *following the first*. For example: if random_chunk='abc', then input='ab' and target='bc'

[9]:
```
def load_random_batch(text, chunk_len, batch_size):
    input_data = torch.zeros(batch_size, chunk_len).long().to(device)
    target = torch.zeros(batch_size, chunk_len).long().to(device)
    for i in range(batch_size):
        start_index = random.randint(0, len(text) - chunk_len - 1)
        end_index = start_index + chunk_len + 1
        chunk = text[start_index:end_index]
        input_data[i] = char_tensor(chunk[:-1])
        target[i] = char_tensor(chunk[1:])
    return input_data, target
```

2 Implement model

Your RNN model will take as input the character for step t_{-1} and output a prediction for the next character t. The model should consiste of three layers - a linear layer that encodes the input character into an embedded state, an RNN layer (which may itself have multiple layers) that operates on that embedded state and a hidden state, and a decoder layer that outputs the predicted character scores distribution.

You must implement your model in the `rnn/model.py` file. You should use a `nn.Embedding` object for the encoding layer, a RNN model like `nn.RNN` or `nn.LSTM`, and a `nn.Linear` layer for the final a predicted character score decoding layer.

TODO: Implement the model in RNN `rnn/model.py`

3 Evaluating

To evaluate the network we will feed one character at a time, use the outputs of the network as a probability distribution for the next character, and repeat. To start generation we pass a priming string to start building up the hidden state, from which we then generate one character at a time.

Note that in the `evaluate` function, every time a prediction is made the outputs are divided by the "temperature" argument. Higher temperature values make actions more equally likely giving

more “random” outputs. Lower temperature values (less than 1) high likelihood options contribute more. A temperature near 0 outputs only the most likely outputs.

You may check different temperature values yourself, but we have provided a default which should work well.

[10]:
```
def evaluate(rnn, prime_str='A', predict_len=100, temperature=0.8):
    hidden = rnn.init_hidden(1, device=device)
    prime_input = char_tensor(prime_str)
    predicted = prime_str

    # Use priming string to "build up" hidden state
    for p in range(len(prime_str) - 1):
        _, hidden = rnn(prime_input[p].unsqueeze(0).to(device), hidden)
    inp = prime_input[-1]

    for p in range(predict_len):
        output, hidden = rnn(inp.unsqueeze(0).to(device), hidden)

        # Sample from the network as a multinomial distribution
        output_dist = output.data.view(-1).div(temperature).exp()
        top_i = torch.multinomial(output_dist, 1)[0]

        # Add predicted character to string and use as next input
        predicted_char = all_characters[top_i]
        predicted += predicted_char
        inp = char_tensor(predicted_char)

    return predicted
```

4 Train RNN

[11]:
```
batch_size = 100
n_epochs = 5000
hidden_size = 200
n_layers = 2
learning_rate = 0.01
model_type = 'lstm'
print_every = 50
plot_every = 50
```

[11]:
```
def eval_test(rnn, inp, target):
    with torch.no_grad():
        hidden = rnn.init_hidden(batch_size, device=device)
        loss = 0
        for c in range(chunk_len):
            output, hidden = rnn(inp[:,c], hidden)
```

```
            loss += criterion(output.view(batch_size, -1), target[:,c])

    return loss.data.item() / chunk_len
```

4.0.1 Train function

TODO: Fill in the train function. You should initialize a hidden layer representation using your RNN's `init_hidden` function, set the model gradients to zero, and loop over each time step (character) in the input tensor. For each time step compute the output of the of the RNN and compute the loss over the output and the corresponding ground truth time step in `target`. The loss should be averaged over all time steps. Lastly, call backward on the averaged loss and take an optimizer step.

[12]:
```
def train(rnn, input, target, optimizer, criterion):
    """
    Inputs:
    - rnn: model
    - input: input character data tensor of shape (batch_size, chunk_len)
    - target: target character data tensor of shape (batch_size, chunk_len)
    - optimizer: rnn model optimizer
    - criterion: loss function

    Returns:
    - loss: computed loss value as python float
    """
    loss = None

    ###################################
    #          YOUR CODE HERE         #
    ###################################

    optimizer.zero_grad()
    hidden = rnn.init_hidden(batch_size, device)

    loss = []

    for i in range(input.shape[1]):
      output, hidden = rnn.forward(input[:, i], hidden)
      loss.append(criterion(output, target[:, i]))

    loss = sum(loss) / input.shape[1]
    loss.backward()
    optimizer.step()

    ##########       END      ##########

    return loss
```

[14]:
```
rnn = RNN(n_characters, hidden_size, n_characters, model_type=model_type,␣
↪n_layers=n_layers).to(device)
rnn_optimizer = torch.optim.Adam(rnn.parameters(), lr=learning_rate)
criterion = nn.CrossEntropyLoss()

start = time.time()
all_losses = []
test_losses = []
loss_avg = 0
test_loss_avg = 0


print("Training for %d epochs..." % n_epochs)
for epoch in range(1, n_epochs + 1):
    loss = train(rnn, *load_random_batch(train_text, chunk_len, batch_size),␣
↪rnn_optimizer, criterion)
    loss_avg += loss

    test_loss = eval_test(rnn, *load_random_batch(test_text, chunk_len,␣
↪batch_size))
    test_loss_avg += test_loss

    if epoch % print_every == 0:
        print('[%s (%d %d%%) train loss: %.4f, test_loss: %.4f]' %␣
↪(time_since(start), epoch, epoch / n_epochs * 100, loss, test_loss))
        print(generate(rnn, 'Wh', 100, device=device), '\n')

    if epoch % plot_every == 0:
        all_losses.append(loss_avg / plot_every)
        test_losses.append(test_loss_avg / plot_every)
        loss_avg = 0
        test_loss_avg = 0
```

```
Training for 5000 epochs…
[0m 42s (50 1%) train loss: 2.3297, test_loss: 2.3265]
Why 'ulnconed peleas bes, so the the ous
The I the leank bet tithon erto the palinrith beey
Theghe ade

[1m 24s (100 2%) train loss: 2.0371, test_loss: 2.0696]
Whampith, I youlse, will and sorther us. Fronter no the for why in;
The
Share in the the fuen the ello

[2m 7s (150 3%) train loss: 1.8797, test_loss: 1.9227]
Whof 'crother a post preak eyet facior, he dear, to wour pard'd concomple gols,
Iwere aiking, for my p
```

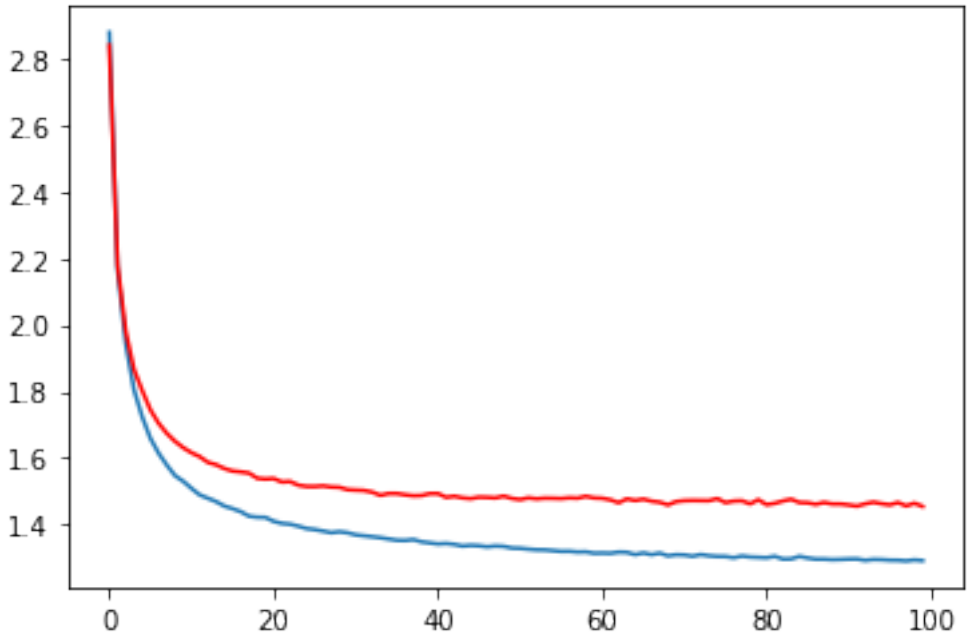
```
Why leave you not the face.

QUEEN GERTRUDE:
Go for me not.

GONERIL:
Who have you they gave you for m

[71m 39s (5000 100%) train loss: 1.2729, test_loss: 1.4625]
What hope that enrich I can mine eternal knight?

LORD PUCK:
The rest fellow the old desire again of a
```

[ ]:
```
# save network
# torch.save(classifier.state_dict(), './rnn_generator.pth')
```

5 Plot the Training and Test Losses

[15]:
```
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker

plt.figure()
plt.plot(all_losses)
plt.plot(test_losses, color='r')
```

[15]:
```
[<matplotlib.lines.Line2D at 0x7f08038d0cc0>]
```



6 Evaluate text generation

Check what the outputted text looks like

[16]:
```
print(evaluate(rnn, prime_str='Th', predict_len=1000))
```

```
The fires of this destiny to have solemn me
To find in thy master, uncle, in thy hand,
We have been accursed pirged weak of me
To make forth their mouth and twenty guilty;
I am the pipes of done, my lord comes
To make a tender kiss and the more defence
And that we should be thou'lt cross the way.

MACDUFF:
Welcome, how did not all I should ere it may thou loved
That are in desire be not starefully pensed.
Have they are pretty repose but still;
The years were worthy soldier was sent not one;
I do but emplore that as thou say'st
The fair youth resire he that pluck us to
Which broke of fair cheerless dignity:
He knights of all any thing to amen;
Thou no subscribes, or fair: then blield us blow with his
love: I have fare of thy converted soon to Sir John:
Sir Michaenes of Winchester are nothing such man
```

```
of foundable side.

CLAUDIO:
Your fearful, is not the while, sweet be freedom of my
fortunes, my lord did men to played here come
nothing at Lonton away: here I dare not be thrive,
for thy pr
```

7 Hyperparameter Tuning

Some things you should try to improve your network performance are: - Different RNN types. Switch the basic RNN network in your model to a GRU and LSTM to compare all three. - Try adding 1 or two more layers - Increase the hidden layer size - Changing the learning rate

TODO: Try changing the RNN type and hyperparameters. Record your results.

8 Basic RNN

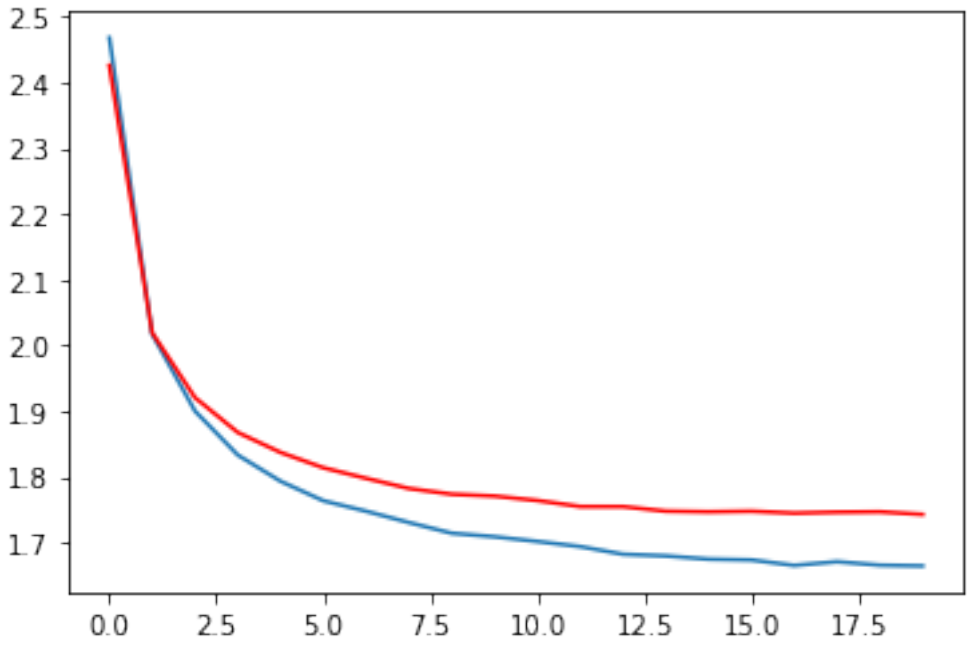
[13]:
```
batch_size = 100
n_epochs = 1000
hidden_size = 100
n_layers = 1
learning_rate = 0.01
model_type = 'rnn'
print_every = 50
plot_every = 50

rnn = RNN(n_characters, hidden_size, n_characters, model_type=model_type,␣
 ↪n_layers=n_layers).to(device)
rnn_optimizer = torch.optim.Adam(rnn.parameters(), lr=learning_rate)
criterion = nn.CrossEntropyLoss()

start = time.time()
all_losses = []
test_losses = []
loss_avg = 0
test_loss_avg = 0


print("Training for %d epochs..." % n_epochs)
for epoch in range(1, n_epochs + 1):
    loss = train(rnn, *load_random_batch(train_text, chunk_len, batch_size),␣
 ↪rnn_optimizer, criterion)
    loss_avg += loss

    test_loss = eval_test(rnn, *load_random_batch(test_text, chunk_len,␣
 ↪batch_size))
```

```
    test_loss_avg += test_loss

    if epoch % print_every == 0:
        print('[%s (%d %d%%) train loss: %.4f, test_loss: %.4f]' %␣
↪(time_since(start), epoch, epoch / n_epochs * 100, loss, test_loss))
        print(generate(rnn, 'Wh', 100, device=device), '\n')

    if epoch % plot_every == 0:
        all_losses.append(loss_avg / plot_every)
        test_losses.append(test_loss_avg / plot_every)
        loss_avg = 0
        test_loss_avg = 0

import matplotlib.pyplot as plt
import matplotlib.ticker as ticker

plt.figure()
plt.plot(all_losses)
plt.plot(test_losses, color='r')
```

```
Training for 1000 epochs…
[0m 35s (50 5%) train loss: 2.1181, test_loss: 2.1251]
What casteraningre sogrenthen be you do of roy,
That ofters wen: hil:
Ales, thill in tire he but path

[1m 12s (100 10%) train loss: 1.9805, test_loss: 1.9542]
When is live the
enfeed of mats nentle the dangers you mine came upon, trour you trust,
The be santing

[1m 47s (150 15%) train loss: 1.8757, test_loss: 1.9084]
Whes remee,
Of to us she more my must:
With un must surst it.

CARINCE:
An thus shore marth
And from p

[2m 22s (200 20%) train loss: 1.8229, test_loss: 1.8526]
Whither sir the one man her in that So you.

IAGONA:
I hoted.

Ale come my wardert onder, of man
```

```
[8m 17s (700 70%) train loss: 1.6803, test_loss: 1.7496]
Wh? I she had here's actice in not incance, the four a lander and Dains of did
ne and him.

FRANDA:
Th

[8m 53s (750 75%) train loss: 1.6676, test_loss: 1.7563]
What thus brue way:
By thing to our postay plan unter I am do! surm'd your tinghry exerons and to
cure

[9m 30s (800 80%) train loss: 1.6732, test_loss: 1.7260]
Where to the comport of it with Lond is the procless they lay of you have more
to the spirit and with

[10m 6s (850 85%) train loss: 1.6734, test_loss: 1.7233]
Whereartunot alongsword to their the gone, save, we'll new, my play run of meand
are of mistress the g

[10m 42s (900 90%) train loss: 1.6760, test_loss: 1.7448]
White trown of ress bear in with my sleerelf the the subjects and offornies
fain's Rimself dovelfors t

[11m 18s (950 95%) train loss: 1.6427, test_loss: 1.7581]
Whest well; and theressish: Heretion they she's shall be for the melet, what the
mich will of it is no

[11m 54s (1000 100%) train loss: 1.6829, test_loss: 1.7461]
Why be not all my so with I will as the sawems bastery so not them, and it where
a give the ours us ri
```

[13]: [<matplotlib.lines.Line2D at 0x7f326281bf98>]



9 GRU

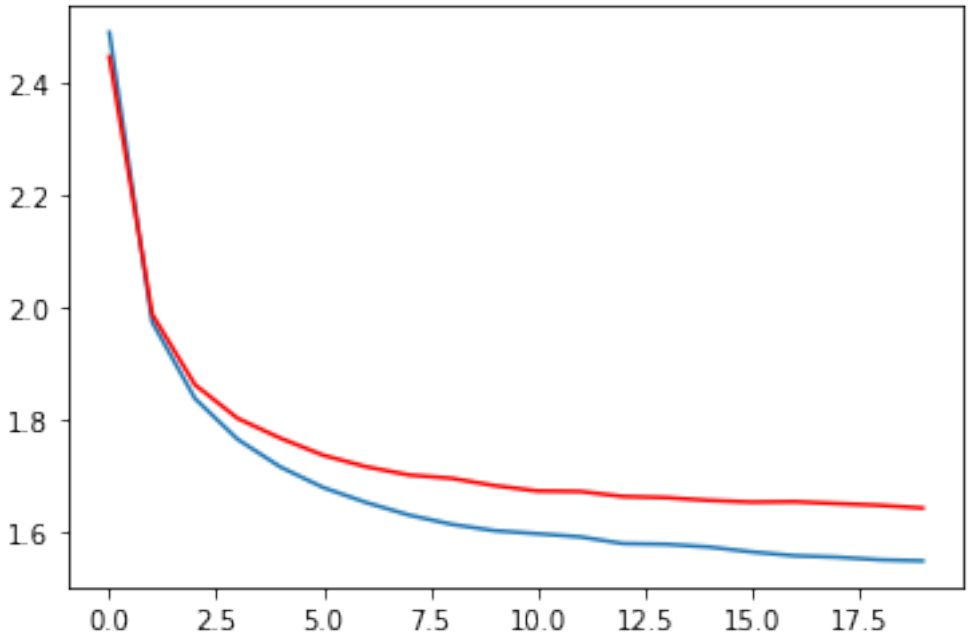
[15]:
```
batch_size = 100
n_epochs = 1000
hidden_size = 100
n_layers = 1
learning_rate = 0.01
model_type = 'gru'
print_every = 50
plot_every = 50

rnn = RNN(n_characters, hidden_size, n_characters, model_type=model_type,␣
 ↪n_layers=n_layers).to(device)
rnn_optimizer = torch.optim.Adam(rnn.parameters(), lr=learning_rate)
criterion = nn.CrossEntropyLoss()

start = time.time()
all_losses = []
test_losses = []
loss_avg = 0
test_loss_avg = 0


print("Training for %d epochs..." % n_epochs)
for epoch in range(1, n_epochs + 1):
```

```
    loss = train(rnn, *load_random_batch(train_text, chunk_len, batch_size),␣
↪rnn_optimizer, criterion)
    loss_avg += loss

    test_loss = eval_test(rnn, *load_random_batch(test_text, chunk_len,␣
↪batch_size))
    test_loss_avg += test_loss

    if epoch % print_every == 0:
        print('[%s (%d %d%%) train loss: %.4f, test_loss: %.4f]' %␣
↪(time_since(start), epoch, epoch / n_epochs * 100, loss, test_loss))
        print(generate(rnn, 'Wh', 100, device=device), '\n')

    if epoch % plot_every == 0:
        all_losses.append(loss_avg / plot_every)
        test_losses.append(test_loss_avg / plot_every)
        loss_avg = 0
        test_loss_avg = 0

import matplotlib.pyplot as plt
import matplotlib.ticker as ticker

plt.figure()
plt.plot(all_losses)
plt.plot(test_losses, color='r')
```

```
Training for 1000 epochs…
[0m 36s (50 5%) train loss: 2.0834, test_loss: 2.0949]
Why, frows hiven vere hilk the to paing you she frat sor kirme hat the the diss
his heed gok the live

[1m 13s (100 10%) train loss: 1.9100, test_loss: 1.9080]
Whe the the whish to be of this Grene of to graze now thearn,
And pecar:
In unouse will winds onty of

[1m 50s (150 15%) train loss: 1.7761, test_loss: 1.8287]
What of of eye this take sir: I gencalter your thou sunce.
Shy grawer be my lord, Ikne of the crome yo

[2m 26s (200 20%) train loss: 1.7136, test_loss: 1.7863]
What shall I will you shall me, leat a siker'
prowart with hear the nother the prayeset.

CORDIZALUS:
```

```
preserving letter his greated done enderer.

PETRUCHIO:
I'll honour me you, the

[8m 33s (700 70%) train loss: 1.6040, test_loss: 1.6765]
What she weep is much a gertion, parting boses is day:
Before the wiserance is sorrow't thees but in t

[9m 9s (750 75%) train loss: 1.5835, test_loss: 1.6487]
What never him child in the
staker excull thee thees the works is lord, I will pray's
That we love the

[9m 46s (800 80%) train loss: 1.5552, test_loss: 1.6296]
What, hast to his stay with men my right
Heart and night it withad we think me that a days
I have his

[10m 22s (850 85%) train loss: 1.5417, test_loss: 1.6542]
Whillow our fierce did
Are hithers of me a ruled,
You were that the for their Cruss to the woalth and

[10m 59s (900 90%) train loss: 1.5585, test_loss: 1.6625]
Why, Master your womances of the whose your tond for your with him.

First Lord:
Ay, there it were as

[11m 36s (950 95%) train loss: 1.5226, test_loss: 1.6399]
What, my to thee me night;
And is poor me of long a cappon your
Stouble respegunes honour of thee, let

[12m 12s (1000 100%) train loss: 1.5580, test_loss: 1.6257]
Why! at Muady that ask
How has ost from this scandset: Paitor!
The privion in me, course subject and n
```

[15]: [<matplotlib.lines.Line2D at 0x7f92c26669e8>]



10 LSTM

[15]:
```
batch_size = 100
n_epochs = 1000
hidden_size = 100
n_layers = 1
learning_rate = 0.01
model_type = 'lstm'
print_every = 50
plot_every = 50

rnn = RNN(n_characters, hidden_size, n_characters, model_type=model_type,␣
 ↪n_layers=n_layers).to(device)
rnn_optimizer = torch.optim.Adam(rnn.parameters(), lr=learning_rate)
criterion = nn.CrossEntropyLoss()

start = time.time()
all_losses = []
test_losses = []
loss_avg = 0
test_loss_avg = 0


print("Training for %d epochs..." % n_epochs)
for epoch in range(1, n_epochs + 1):
```

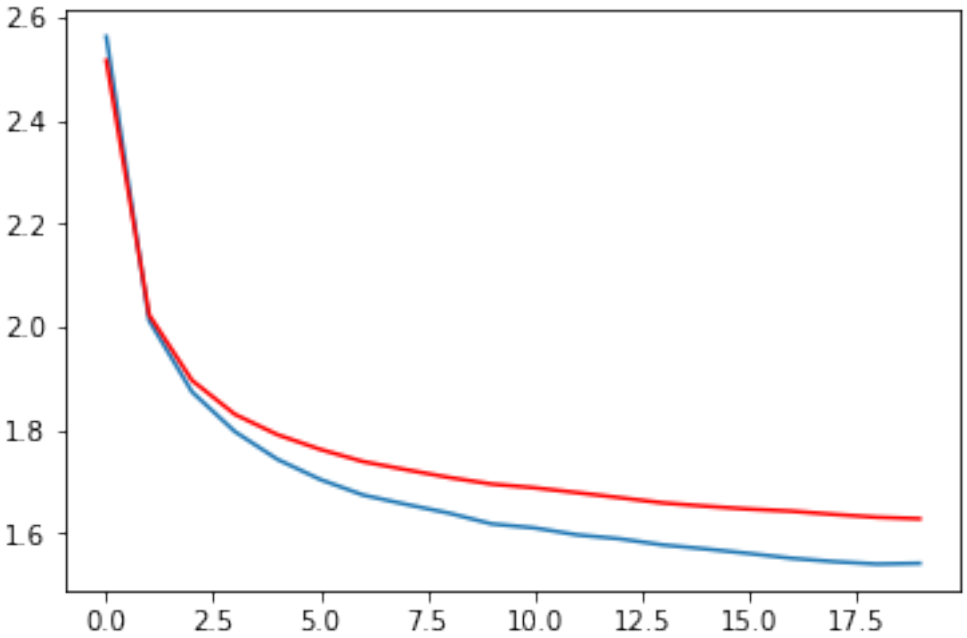
```
    loss = train(rnn, *load_random_batch(train_text, chunk_len, batch_size),␣
↪rnn_optimizer, criterion)
    loss_avg += loss

    test_loss = eval_test(rnn, *load_random_batch(test_text, chunk_len,␣
↪batch_size))
    test_loss_avg += test_loss

    if epoch % print_every == 0:
        print('[%s (%d %d%%) train loss: %.4f, test_loss: %.4f]' %␣
↪(time_since(start), epoch, epoch / n_epochs * 100, loss, test_loss))
        print(generate(rnn, 'Wh', 100, device=device), '\n')

    if epoch % plot_every == 0:
        all_losses.append(loss_avg / plot_every)
        test_losses.append(test_loss_avg / plot_every)
        loss_avg = 0
        test_loss_avg = 0
```

```
Training for 1000 epochs…
[0m 39s (50 5%) train loss: 2.1194, test_loss: 2.1190]
Why?

Lou the seave sas that thing his nidefins fort.

LALD:
The bly dure freermadefing ransall obus:


[1m 16s (100 10%) train loss: 1.9235, test_loss: 1.9414]
Why your priess, mid I lave om;
Nou wa umbers spies. And sweal a sich, than
Buitinstell:
the mave pro

[1m 54s (150 15%) train loss: 1.8250, test_loss: 1.8803]
Whath my say, us
Which the encell we mace sent
ris thear fame a be hast collfore, on the more bromat,

[2m 31s (200 20%) train loss: 1.7797, test_loss: 1.7884]
What his with the should I sound here for a bard you have go that lester boy,
and his verve that boo i

[3m 9s (250 25%) train loss: 1.7176, test_loss: 1.7792]
What, the king the make how leave fauth
That I am we Luctie;
```

```
QUEEN MESTA:
No so spirits will, if the king him wishing for a
tim

[8m 54s (700 70%) train loss: 1.5720, test_loss: 1.6528]
Who you friends, now you be
A fordow'd instand as a tistore
With here the king of two not to my laught

[9m 32s (750 75%) train loss: 1.5742, test_loss: 1.6360]
Why you son to earth.

BRUTUS:
By hear the spong.
Expound what it Edst you are the hate
the that but y

[10m 11s (800 80%) train loss: 1.5650, test_loss: 1.6414]
What's the strungle horrupeacus.

DESDEMONA:
No, promust give, to these man's great our reasts
Our fin

[10m 49s (850 85%) train loss: 1.5714, test_loss: 1.6286]
Who, that sir; what have sine.

KING CLAUDIUS:
Goo some in the properbive and the oneces with priserve

[11m 28s (900 90%) train loss: 1.5227, test_loss: 1.6344]
What's his gentleman and as any singry to their
To the flesh, I prepes; so purse him thee the better s

[12m 6s (950 95%) train loss: 1.5419, test_loss: 1.6310]
Why, sower cannot all gone.
Command at walk he had coursed a consent a
protenciar of true to Cellen.



[12m 44s (1000 100%) train loss: 1.5507, test_loss: 1.6142]
Who, tarked and spestille of granty and frust;
For I can as own all this purpossair that in last the w
```

[17]:
```
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker

plt.figure()
plt.plot(all_losses)
plt.plot(test_losses, color='r')
```

[17]:
```
[<matplotlib.lines.Line2D at 0x7f6faccd5128>]
```



11 LSTM + 2 Layers

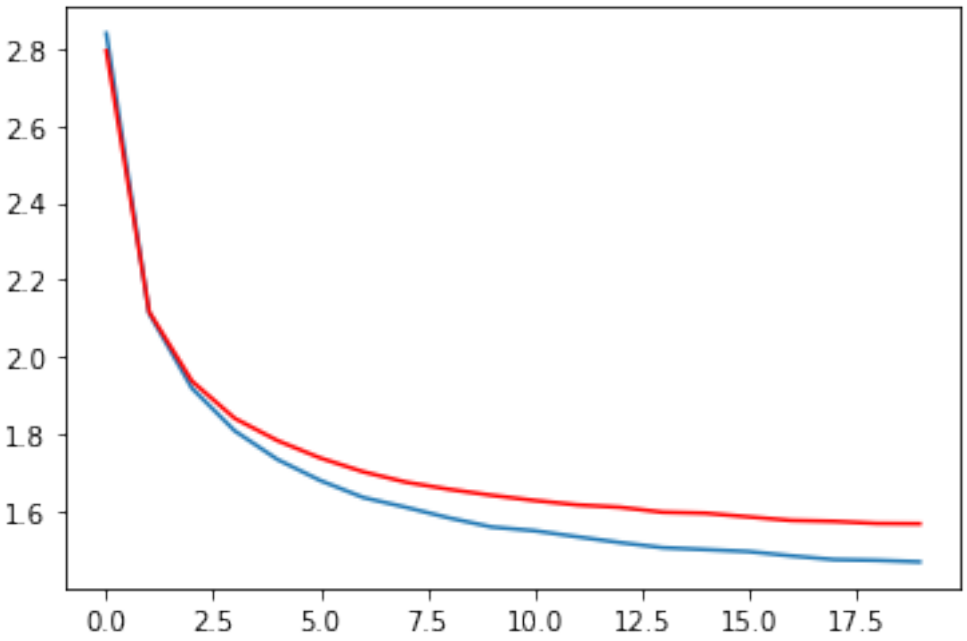
[18]:
```
batch_size = 100
n_epochs = 1000
hidden_size = 100
n_layers = 2
learning_rate = 0.01
model_type = 'lstm'
print_every = 50
plot_every = 50

rnn = RNN(n_characters, hidden_size, n_characters, model_type=model_type,␣
 ↪n_layers=n_layers).to(device)
rnn_optimizer = torch.optim.Adam(rnn.parameters(), lr=learning_rate)
criterion = nn.CrossEntropyLoss()
```

```
start = time.time()
all_losses = []
test_losses = []
loss_avg = 0
test_loss_avg = 0


print("Training for %d epochs..." % n_epochs)
for epoch in range(1, n_epochs + 1):
    loss = train(rnn, *load_random_batch(train_text, chunk_len, batch_size),␣
 ↪rnn_optimizer, criterion)
    loss_avg += loss

    test_loss = eval_test(rnn, *load_random_batch(test_text, chunk_len,␣
 ↪batch_size))
    test_loss_avg += test_loss

    if epoch % print_every == 0:
        print('[%s (%d %d%%) train loss: %.4f, test_loss: %.4f]' %␣
 ↪(time_since(start), epoch, epoch / n_epochs * 100, loss, test_loss))
        print(generate(rnn, 'Wh', 100, device=device), '\n')

    if epoch % plot_every == 0:
        all_losses.append(loss_avg / plot_every)
        test_losses.append(test_loss_avg / plot_every)
        loss_avg = 0
        test_loss_avg = 0

import matplotlib.pyplot as plt
import matplotlib.ticker as ticker

plt.figure()
plt.plot(all_losses)
plt.plot(test_losses, color='r')
```

```
Training for 1000 epochs…
[0m 41s (50 5%) train loss: 2.2753, test_loss: 2.2711]
Whay weij adall une me ne moucheil.

PSIES:
O to meret in ou werikt mai
ond beld ant tou to gith be ha

[1m 22s (100 10%) train loss: 2.0108, test_loss: 2.0102]
Wh, and compelwe thinefis notier yunger:
Whone! here the wend
```

```
[13m 16s (950 95%) train loss: 1.4431, test_loss: 1.5724]
What, here come, the diem the first received a
as a shall be not weak
And her your good fear
balls I s

[13m 58s (1000 100%) train loss: 1.4602, test_loss: 1.5597]
Whill in the untile time of his was rather.
Yet so cheek me, in his sease to the counche:
Here was so
```

[18]: `[<matplotlib.lines.Line2D at 0x7f6fa40ee9e8>]`



12 LSTM + 2 Layers + More Hidden Layers

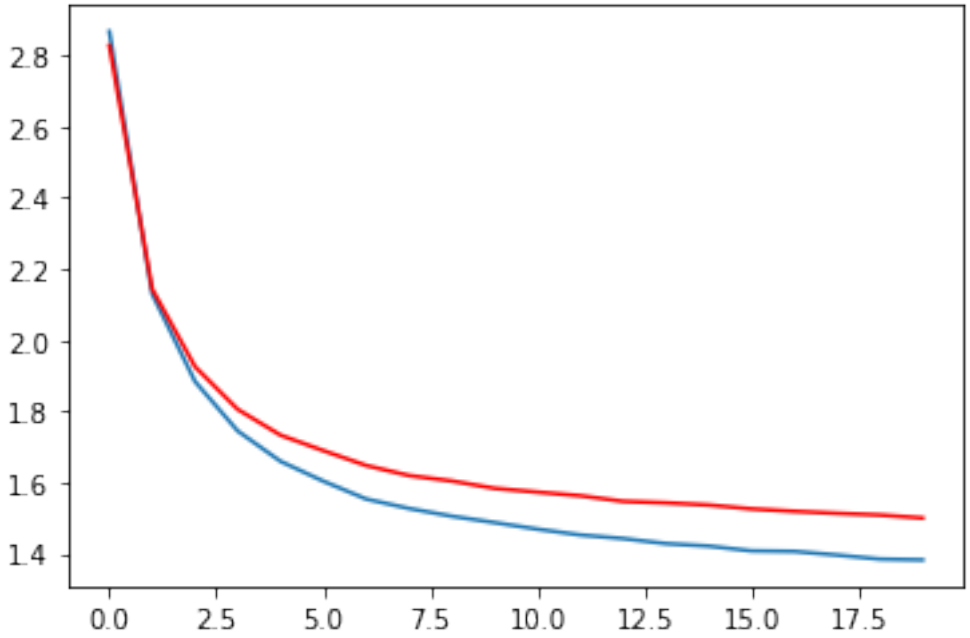
[14]:
```
batch_size = 100
n_epochs = 1000
hidden_size = 200
n_layers = 2
learning_rate = 0.01
model_type = 'lstm'
print_every = 50
plot_every = 50
```

```
rnn = RNN(n_characters, hidden_size, n_characters, model_type=model_type,␣
 ↪n_layers=n_layers).to(device)
rnn_optimizer = torch.optim.Adam(rnn.parameters(), lr=learning_rate)
criterion = nn.CrossEntropyLoss()

start = time.time()
all_losses = []
test_losses = []
loss_avg = 0
test_loss_avg = 0


print("Training for %d epochs..." % n_epochs)
for epoch in range(1, n_epochs + 1):
    loss = train(rnn, *load_random_batch(train_text, chunk_len, batch_size),␣
 ↪rnn_optimizer, criterion)
    loss_avg += loss

    test_loss = eval_test(rnn, *load_random_batch(test_text, chunk_len,␣
 ↪batch_size))
    test_loss_avg += test_loss

    if epoch % print_every == 0:
        print('[%s (%d %d%%) train loss: %.4f, test_loss: %.4f]' %␣
 ↪(time_since(start), epoch, epoch / n_epochs * 100, loss, test_loss))
        print(generate(rnn, 'Wh', 100, device=device), '\n')

    if epoch % plot_every == 0:
        all_losses.append(loss_avg / plot_every)
        test_losses.append(test_loss_avg / plot_every)
        loss_avg = 0
        test_loss_avg = 0

import matplotlib.pyplot as plt
import matplotlib.ticker as ticker

plt.figure()
plt.plot(all_losses)
plt.plot(test_losses, color='r')
```

```
Training for 1000 epochs…
[0m 37s (50 5%) train loss: 2.3336, test_loss: 2.3243]
Whanly hever dadils seritert-Thof linece lefisd dople lis an, thos

AHENO EOSA:
I. guce horad lour bol
```

```
the watch'd to it. Then I take this fall; for the door imagines
Did ne

[11m 49s (950 95%) train loss: 1.3856, test_loss: 1.4836]
What Sibbris that ded the new to cannot
would my trawn there, never Rome in thy issue
Do with personer

[12m 25s (1000 100%) train loss: 1.3934, test_loss: 1.5127]
What doth a good duty I cannot,
Or confess him with your love.

PRINCE HENRY:
I would praise in your L
```

[14]: `[<matplotlib.lines.Line2D at 0x7fbe0b0ff358>]`



13 LSTM + 2 Layers + More Hidden Layers + Lower Learning Rate

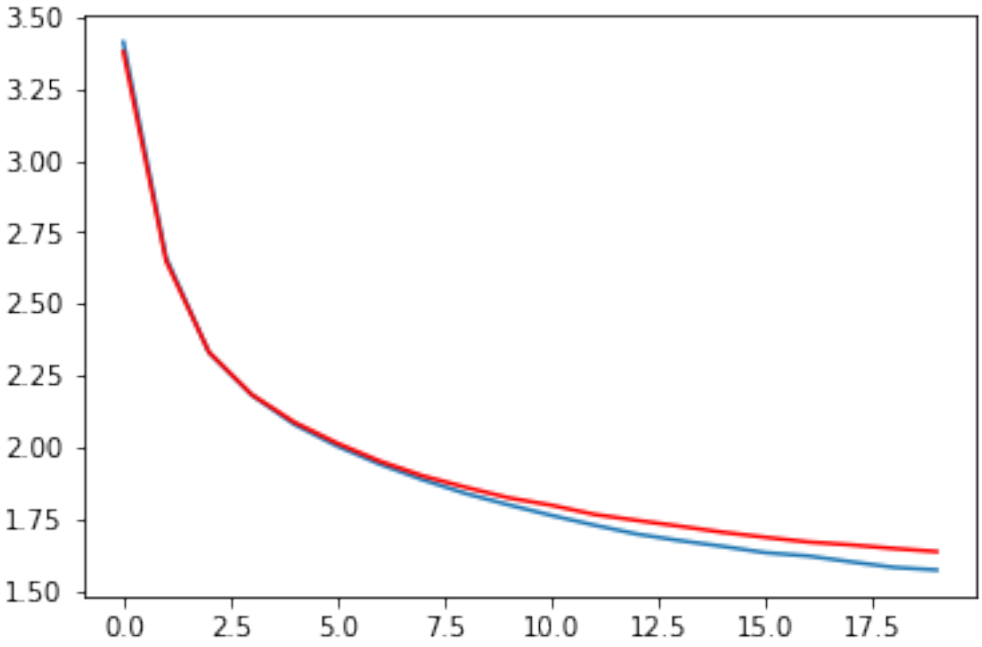
[15]:
```
batch_size = 100
n_epochs = 1000
hidden_size = 200
n_layers = 2
```

```
learning_rate = 0.001
model_type = 'lstm'
print_every = 50
plot_every = 50

rnn = RNN(n_characters, hidden_size, n_characters, model_type=model_type,␣
 ↪n_layers=n_layers).to(device)
rnn_optimizer = torch.optim.Adam(rnn.parameters(), lr=learning_rate)
criterion = nn.CrossEntropyLoss()

start = time.time()
all_losses = []
test_losses = []
loss_avg = 0
test_loss_avg = 0


print("Training for %d epochs..." % n_epochs)
for epoch in range(1, n_epochs + 1):
    loss = train(rnn, *load_random_batch(train_text, chunk_len, batch_size),␣
 ↪rnn_optimizer, criterion)
    loss_avg += loss

    test_loss = eval_test(rnn, *load_random_batch(test_text, chunk_len,␣
 ↪batch_size))
    test_loss_avg += test_loss

    if epoch % print_every == 0:
        print('[%s (%d %d%%) train loss: %.4f, test_loss: %.4f]' %␣
 ↪(time_since(start), epoch, epoch / n_epochs * 100, loss, test_loss))
        print(generate(rnn, 'Wh', 100, device=device), '\n')

    if epoch % plot_every == 0:
        all_losses.append(loss_avg / plot_every)
        test_losses.append(test_loss_avg / plot_every)
        loss_avg = 0
        test_loss_avg = 0

import matplotlib.pyplot as plt
import matplotlib.ticker as ticker

plt.figure()
plt.plot(all_losses)
plt.plot(test_losses, color='r')
```

```
Training for 1000 epochs…
[0m 36s (50 5%) train loss: 3.0008, test_loss: 2.9757]
```

```
What best sorrows fear
friends for his chantime.
A de, again, in the suffice too your partured, he dou

[12m 0s (950 95%) train loss: 1.5619, test_loss: 1.6338]
Which one, I thoughts, are you do your bur and from the
mount at, I have men bethings of the heaven of

[12m 39s (1000 100%) train loss: 1.5576, test_loss: 1.6030]
What thou even him so queen so deed in the good,
The is the parder: and the just and been of it dispar
```

[15]: `[<matplotlib.lines.Line2D at 0x7fbe0222c3c8>]`



14 Extra Credit: Dataset is 'A Tale of Two Cities' by Charles Dickens

[13]:
```
all_characters = string.printable
n_characters = len(all_characters)

file_path = './taleoftwocities.txt'
file = unidecode.unidecode(open(file_path).read())
file_len = len(file)
print('file_len =', file_len)
```

```
# we will leave the last 1/10th of text as test
split = int(0.9*file_len)
train_text = file[:split]
test_text = file[split:]

print('train len: ', len(train_text))
print('test len: ', len(test_text))
```

```
file_len = 755426
train len:  679883
test len:  75543
```

[20]:
```
batch_size = 100
n_epochs = 1000
hidden_size = 200
n_layers = 2
learning_rate = 0.01
model_type = 'lstm'
print_every = 50
plot_every = 50
```

[21]:
```
rnn = RNN(n_characters, hidden_size, n_characters, model_type=model_type,␣
 ↪n_layers=n_layers).to(device)
rnn_optimizer = torch.optim.Adam(rnn.parameters(), lr=learning_rate)
criterion = nn.CrossEntropyLoss()

start = time.time()
all_losses = []
test_losses = []
loss_avg = 0
test_loss_avg = 0


print("Training for %d epochs..." % n_epochs)
for epoch in range(1, n_epochs + 1):
    loss = train(rnn, *load_random_batch(train_text, chunk_len, batch_size),␣
 ↪rnn_optimizer, criterion)
    loss_avg += loss

    test_loss = eval_test(rnn, *load_random_batch(test_text, chunk_len,␣
 ↪batch_size))
    test_loss_avg += test_loss

    if epoch % print_every == 0:
        print('[%s (%d %d%%) train loss: %.4f, test_loss: %.4f]' %␣
 ↪(time_since(start), epoch, epoch / n_epochs * 100, loss, test_loss))
```

```
        print(generate(rnn, 'Wh', 100, device=device), '\n')

    if epoch % plot_every == 0:
        all_losses.append(loss_avg / plot_every)
        test_losses.append(test_loss_avg / plot_every)
        loss_avg = 0
        test_loss_avg = 0
```

```
Training for 1000 epochs…
[0m 41s (50 5%) train loss: 1.8988, test_loss: 1.8707]
Wher raing. It the sa you hing the the sightter in the the reolly, thon, aband
the tho last you
thy he

[1m 21s (100 10%) train loss: 1.6165, test_loss: 1.6558]
What at the pass his
any same the
looked of made is
to the placte the stopped with the for repared was

[2m 1s (150 15%) train loss: 1.5131, test_loss: 1.5378]
Whow that the strag and a
strong some be faces, and not so hand, could had long from Mr. Crunch a firs

[2m 40s (200 20%) train loss: 1.4561, test_loss: 1.4868]
Whought to him short appearate of for it in the
sause where asmess in these compliectious of of one to

[3m 19s (250 25%) train loss: 1.3865, test_loss: 1.4376]
What the heard from the ground of something out of the
prostain, some a lamping, and the cells, he had

[3m 58s (300 30%) train loss: 1.3828, test_loss: 1.4150]
What it known the stoppy.
Where the used no main and feightly, and he pleasing when the woman, and it

[4m 36s (350 35%) train loss: 1.3551, test_loss: 1.4246]
Where, and there was about why. It was a post, and the
guard of this, in two was seen the strong of th

[5m 16s (400 40%) train loss: 1.3167, test_loss: 1.3594]
What only be malic
shrieking on a morning, passery sense, one lost on the
I some till of a flatter was

[5m 56s (450 45%) train loss: 1.3202, test_loss: 1.3849]
When if he remarkation,
```

```
"Do you

[13m 4s (1000 100%) train loss: 1.2461, test_loss: 1.3462]
What do you have made the accused by his look back of a
cannot to be?"

"Lousnight except of herst or
```

[22]:
```
print(evaluate(rnn, prime_str='It', predict_len=1000))
```

```
It would have no one stooped
down, in a road.

Not a little way, always be him again at the Circumstance, and confished his
knee
for the distantance of his reason, swallowing pockets.

"I am not to word contention that you could some time much prevented to have him
in the
more noise the foo, and patience struck in prison.

The truth fact, and good friends of his back in the
dusticial hands; and how me all spot being although natural
disapported Mr. Lorry the steady and room, with the heads of her
torning and fancy night, and live of the time to hir
another face and sound of this paper, had no done of her hand faces of his
distress son of its tones and women, whose turn, and for
the fire for the prisoned, from the chamber in a sound to
her hand, some heard and the corner, but a way, and with a corner of the first
leaves the far to him
again; and touching on the spectrad, and the shoes as her bow or she
was ever have had spoken no recovered her two resonation
for the heat terrible than any
```

[ ]:
```

```