Generating Text with an RNN

In [1]:

```
from google.colab import drive
drive.mount('/content/gdrive')
import os
os.chdir("gdrive/My Drive/mp4")
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6
qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redirect_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%
b&scope=email%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdocs.test%20https%3A%2F%2Fwww.googleapis.
2Fauth%2Fdrive%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive.photos.readonly%20https%3A%2F%2Fww
ogleapis.com%2Fauth%2Fpeopleapi.readonly&response_type=code

Enter your authorization code:
··········
Mounted at /content/gdrive
```

In [4]:

```
#!pip install torch
#!pip install unidecode
```

```
Collecting torch
  Downloading
https://files.pythonhosted.org/packages/49/0e/e382bcf1a6ae8225f50b99cc26effa2d4cc6d66975ccf3fa9590e
dce/torch-0.4.1-cp36-cp36m-manylinux1_x86_64.whl (519.5MB)
    100% |████████████████████████████████| 519.5MB 30kB/s
tcmalloc: large alloc 1073750016 bytes == 0x59db2000 @  0x7f056de272a4 0x591a07 0x5b5d56 0x502e9a
0x506859 0x502209 0x502f3d 0x506859 0x504c28 0x502540 0x502f3d 0x506859 0x504c28 0x502540 0x502f3d
0x506859 0x504c28 0x502540 0x502f3d 0x507641 0x502209 0x502f3d 0x506859 0x504c28 0x502540 0x502f3d
0x507641 0x504c28 0x502540 0x502f3d 0x507641
Installing collected packages: torch
Successfully installed torch-0.4.1
Collecting unidecode
  Downloading
https://files.pythonhosted.org/packages/31/39/53096f9217b057cb049fe872b7fc7ce799a1a89b76cf917d9639e
8b5/Unidecode-1.0.23-py2.py3-none-any.whl (237kB)
    100% |████████████████████████████████| 245kB 7.3MB/s
Installing collected packages: unidecode
Successfully installed unidecode-1.0.23
```

In [35]:

```
import unidecode
import string
import random
import re
import time

import torch
import torch.nn as nn

%matplotlib inline

%load_ext autoreload
%autoreload 2
```

```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
```

In [0]:

```
from rnn.model import RNN
from rnn.helpers import time_since
from rnn.generate import generate
```

In [0]:

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

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`).

In [38]:

```
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
```

In [39]:

```
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))
```

```
ly?

VERNON:
So do we.

HOTSPUR:
His is certain, ours is doubtful.

EARL OF WORCESTER:
Good cousin, be advised; stir not tonight.

VERNON:
Do not, my lord.

EARL OF DOUGLAS:
You do not counsel well:
Yo
```

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`.

In [0]:

```
# 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'

In [0]:

```
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
```

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`

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.

In [0]:

```
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
```

Train RNN

In [0]:

```
batch_size = 100
n_epochs = 2000
hidden_size = 150
n_layers = 2
learning_rate = 0.001
model_type = 'gru'
print_every = 50
plot_every = 50
```

In [0]:

```
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
```

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.

In [0]:

```
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
    """

    ####################################
    #          YOUR CODE HERE          #
    ####################################
    loss = 0
    hidden = rnn.init_hidden(batch_size, device=device)
    rnn.zero_grad()

    for c in range(chunk_len):
        output, hidden = rnn(_input[:,c], hidden)
        loss += criterion(output, target[:,c])

    loss /= chunk_len

    loss.backward()
    optimizer.step()
    #loss = loss.data[0] / chunk_len #####should be loss.data.item()?

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

    return loss
```

In [46]:

```
rnn = RNN(n_characters, hidden_size, n_characters, model_type=model_type, n_layers=n_layers).to(dev
ice)
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, criteri
on)
    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 2000 epochs...
[1m 28s (50 2%) train loss: 2.7492, test_loss: 2.7497]
Wher aus gaircih-Klt yr t,

HRSA!_Bhte ate ehe, phet irh thm thw bme ainss hey t thome, su yo oll t ue

[2m 56s (100 5%) train loss: 2.2808, test_loss: 2.2928]
Wh, ouptat olme konthry the wpwin hor th the the an! mat aoor and bass fhand ad to at fone fill wa
muc

[4m 24s (150 7%) train loss: 2.1200, test_loss: 2.1126]
Whith yoursen the king beif I do whal: and distis neralus?

YORL:
Mutiy besthe the thes, chas groness

[5m 52s (200 10%) train loss: 1.9840, test_loss: 1.9969]
Whe, the wery flangering this vanibnement there hite sectese at in fair, Hid we suse feat:
And the rei

[7m 20s (250 12%) train loss: 1.8851, test_loss: 1.8847]
Why lacket, some not the sey
God afrent in the too?

PARDLUS:
I did lovers, whlomoul and here, dear of

[8m 49s (300 15%) train loss: 1.8157, test_loss: 1.8281]
Why powes ead-there: so thy the mess that serve upon:
The do;
As are our are so corver am this all sco

[10m 17s (350 17%) train loss: 1.7632, test_loss: 1.8000]
Wher chartor,
Our all chuse sing; my have me,
Sir, so dust not thee.

HACAR:
I do head, my sackan us a

[11m 46s (400 20%) train loss: 1.6938, test_loss: 1.7439]
Wh-tile you destrent have my hast than him dears of agom, suspet, that found ear natured to our or
ne t

[13m 14s (450 22%) train loss: 1.6791, test_loss: 1.7274]
Whing with short
King seet still she pride the mantry soul speen and his forest and unseep,
And have I

[14m 43s (500 25%) train loss: 1.6580, test_loss: 1.6873]
Where you, flender.

VIUS FRome sterve of reeplan:
If you do away of me your too in your not a crown
A

[16m 12s (550 27%) train loss: 1.6109, test_loss: 1.6607]
What destruits:
Then go to my man to my will? where with watch him
Of it. There's mocker such grace hi

[17m 40s (600 30%) train loss: 1.5671, test_loss: 1.6179]
When they have of heaven show,
The laid of him with him in his devil heaven to you.

FLORD:
An your se

[19m 8s (650 32%) train loss: 1.5599, test_loss: 1.6128]
Whou such a spoke of his cannot it forcell.

KING HENRY V:
Yet be'll be field,
To Servick and a vain u

[20m 37s (700 35%) train loss: 1.5327, test_loss: 1.6060]
What will you, my shall now most,
To shall lord, falstor, or amperer
Be recupy both the Boold, more yo

[22m 5s (750 37%) train loss: 1.5241, test_loss: 1.5559]
Which which now, but part last;
Our grolent grace demislless in wrain him.

GON:
That I have state, an

[23m 34s (800 40%) train loss: 1.5203, test_loss: 1.5756]
What never be jead's reason?
I will bear the writh it makes it.

DROMIO:
Gentleman: Cramness and sound

[25m 3s (850 42%) train loss: 1.5146, test_loss: 1.5721]
Why that I will be Praised
Go that is a falsed of love of it, gentleman:
Because strange, be better hi

[26m 31s (900 45%) train loss: 1.4977, test_loss: 1.5653]
Why the other be how:
And when a power and his worfes
He's rebelly is a man sumine
The as the isself w

[28m 0s (950 47%) train loss: 1.4653, test_loss: 1.5551]
Why comest and deliver you a book
To heart in look on the way, and at tate me
Was stushims her heart,

[29m 29s (1000 50%) train loss: 1.4828, test_loss: 1.5389]
Wher beartv, let whore
```

```
You
belovely faime intracture of wail one.

CRIAN:
How lords, I'll not suree th

[30m 57s (1050 52%) train loss: 1.4587, test_loss: 1.5196]
What there is wish, that I read the this
mailon in not stand in an resolcies oft?

OTHELLO:
He hath su

[32m 26s (1100 55%) train loss: 1.4285, test_loss: 1.5272]
Wher this cuck in the woman:
I have word speak on his sorrow straight and trees?

EMARD:
Ay, my lord,

[33m 55s (1150 57%) train loss: 1.4848, test_loss: 1.5188]
Why foul, go to look'd tormed,
In the king!

Ford:
'Tis newls; no mutty's state.

KENT:
I be darest to

[35m 23s (1200 60%) train loss: 1.4170, test_loss: 1.5305]
Whas seal all and move to this crown with deed
Of need, the compable to fiendshment,
In chomes was not

[36m 52s (1250 62%) train loss: 1.4117, test_loss: 1.5096]
Whither be springs?
To make hither's endired does confess which
she dare the receive me to to your spe

[38m 20s (1300 65%) train loss: 1.4332, test_loss: 1.5025]
Whas they seek me in his will
The life.
And may bring the Greeks my ear to see another,
To have a chan

[39m 49s (1350 67%) train loss: 1.4349, test_loss: 1.5085]
What, thus then shall dost not stand of our bloody,
And an our oaths of prive him for prisoners.

LORD

[41m 18s (1400 70%) train loss: 1.3867, test_loss: 1.5411]
What you
makes her led and charge it mercy,
But mouthsay loss is no reputation.

KING LEAR:
I may then

[42m 46s (1450 72%) train loss: 1.4106, test_loss: 1.4972]
Whis a soxer will
Like a man man come on the Dukes that he hath been.
I will will husband of the went

[44m 15s (1500 75%) train loss: 1.4220, test_loss: 1.5119]
Whiles not stomach:
I would bite a man of an old against them maid:
I stay'd be thum take his buddled

[45m 44s (1550 77%) train loss: 1.3969, test_loss: 1.4983]
What stard should make his melice of love
In the words out of this tans patient all the fear
In subjec

[47m 12s (1600 80%) train loss: 1.4047, test_loss: 1.4990]
What this feeling to what stars:
```

```
What this feeling to what scars,
While tell us, I will die than the last in the deep
shall not be for

[48m 41s (1650 82%) train loss: 1.4281, test_loss: 1.5030]
Where then, dost thou should you shall,
That you may call this fit on you; if which,
And subject you,

[50m 9s (1700 85%) train loss: 1.4058, test_loss: 1.4565]
What on Hamberit,
And she not have as makes are than being in
appear, sir, good welcome, nor so enjoy

[51m 36s (1750 87%) train loss: 1.3823, test_loss: 1.4721]
What this deed, not, and blose,
With men's sealzest young Ford, you have by it.

ANTIPHOLUS OF SYRACUS

[53m 4s (1800 90%) train loss: 1.3921, test_loss: 1.4734]
Whose you? Appear York one,
Peace,
Pardon me where you would not swear
I' the very choleried us that w

[54m 31s (1850 92%) train loss: 1.4009, test_loss: 1.4565]
What in this tarry, on sits how
England's contrant, or you proppier he for you.

TIMON:
Nay, and will

[55m 56s (1900 95%) train loss: 1.4252, test_loss: 1.4947]
Whis saves the Duke,
Because you there is the heads to appear.

SILVIA:
A faith life of faith before t

[57m 23s (1950 97%) train loss: 1.3959, test_loss: 1.4960]
What said he beaten too.

MACBIATH:
Let him prove the experiest than the bose
Whom the month, and vill

[58m 51s (2000 100%) train loss: 1.3605, test_loss: 1.5001]
Wher sent: at here.

GRADEIMO:
And with slain, you were such a flight,
And compleep of his sweet, thou
```

In [0]:

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

Plot the Training and Test Losses

In [48]:

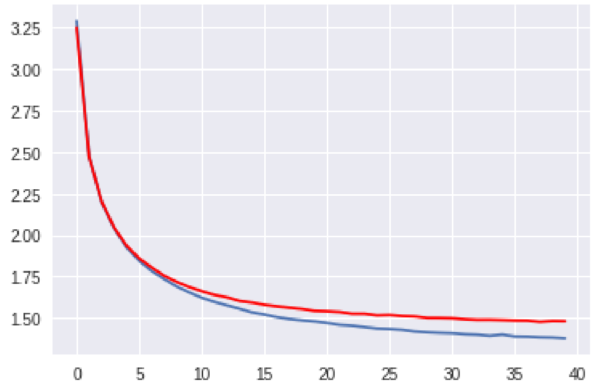
```
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker

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

Out[48]:

```
[<matplotlib.lines.Line2D at 0x7f4aa1637588>]
```

```
[<matplotlib.lines.Line2D at [illegible]>]
```



Evaluate text generation

Check what the outputted text looks like

In [49]:

```
print(evaluate(rnn, prime_str='Th', predict_len=1000))
```

```
This naught her to the sleep!
That shall been you thou yet to
At the other to me, and to the duke to advances
Marry be as much down.

ROSSALINE:
Ay, then I have room in a short, that a times.

LYCAILUS:
Full the cousin, being father in a words,
The deed strings the serve with the company, see.
Lord knee quarters in the properor and kind on my love
That thus on her understand to comforth becomes
to caintly have benefites shall be gone.

POLPEY:
O gain you sad to send upon all amends.

PANEROLLES:
Let him out of me so unpressips, we debered.

DUKE SYRACUSE:
If you shall be conceit his daughter in
By help it. This is the godges of my wise;
What are strength, here we stand understand this
word off my slain! Can the thilk not twasty.

DON PEDRO:
Then blush and here on dog;
For I, his wish'd, smoke the gods, my lords!
How now, sir, and through that a farther, as the king
That he lost like a devil, nor away,
And let their father;' yet so besting that that
sweer at much many consicial bring litt
```

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.