ECE 47300 Assignment 6 Exercise

Your Name: Owen Semeter

Objective: Build an RNN model to predict the next character in a sequence of text data from Shakespeare's plays.

Exercise 1: Data Preprocessing (30 points)

In this part, you will implement some preprocessing functions. Run the following code to load the text data from the given file "shakespeare.txt". Do not change the random seed.

In [2]:
```
import numpy as np
! pip install unidecode
import unidecode
import string
import time
import torch
import pdb

import torch.nn as nn
from torch.autograd import Variable

all_characters = string.printable
print(all_characters)
```

```
Collecting unidecode
  Downloading Unidecode-1.3.8-py3-none-any.whl.metadata (13 kB)
Downloading Unidecode-1.3.8-py3-none-any.whl (235 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0.0/235.5 kB ? eta -:--:--
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╸━ 225.3/235.5 kB 8.7 MB/s eta 0:00:01
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 235.5/235.5 kB 6.2 MB/s eta 0:00:00
Installing collected packages: unidecode
Successfully installed unidecode-1.3.8
0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!"#$%&'()*+,-./:;<=>?@
[\]^_`{|}~
```

Exercise 1: Data Preprocessing (30 points)

Follow the step on the instructions and mount your google drive on Colab which allows to access the .txt file uploaded on your drive that was included with this assignment.

In [3]:
```
! pip install google
from google.colab import drive
drive.mount('/content/drive')
```

```
Requirement already satisfied: google in /usr/local/lib/python3.11/dist-packages (2.
0.3)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.11/dist-pack
ages (from google) (4.13.3)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.11/dist-packa
ges (from beautifulsoup4->google) (2.6)
Requirement already satisfied: typing-extensions>=4.0.0 in /usr/local/lib/python3.1
1/dist-packages (from beautifulsoup4->google) (4.12.2)
Mounted at /content/drive
```

In [4]:
```
def read_file(filename):
    file = unidecode.unidecode(open(filename).read())
    return file

dir_root = '.'     # Your assignment dir
file_path = dir_root + '/shakespeare.txt'
file = read_file(file_path)
file_len = len(file)
print(f"file length: {file_len}")
print(file[:100])
```

```
file length: 1115394
First Citizen:
Before we proceed any further, hear me speak.

All:
Speak, speak.

First Citizen:
You
```

Task 1: Implement function to get a random chunk of Shakespeare text (15 points)

The `get_random_chunk` function is a helper function that generates a random chunk of **input text data** and **output text data** (which is one character shifted from the input) from the Shakespeare dataset. Specifically, the `chunk_len` argument specifies the size of the input and output sequences. For example, if `chunk_len=4`, then a valid return value would be the two chunks: `('Befo','efor')` or `('proc','roce')`. This function is useful in generating diverse sets of input data for training the RNN model in the assignment.

Hints:

- Start from a random index of the file (but note that the max index must be small enough so that a full chunk can be extracted).
- Based on this random start index, extract `chunk_len` characters for the input sequence and `chunk_len` characters for the output sequence (shifted one character to the right).

In [5]:
```
def get_random_chunk(file, rng, chunk_len = 100):
    ######### Your Code Here ###########
    assert(len(file) > chunk_len + 1)
    max_idx = len(file) - (chunk_len + 1)
    idx = rng.randint(0, max_idx)
    return (file[idx:idx + chunk_len], file[idx+1:idx + chunk_len + 1])
    ######### End of your code #########

rng = np.random.RandomState(123) # use this if you need to generate a random sample
curr_chunk, next_chunk = get_random_chunk(file='Hello world!', rng=rng, chunk_len=1
print(f"curr_chunk =>{curr_chunk}\n next_chunk=> {next_chunk}")
print(f"Is curr_chunk and next_chunk same length: {len(curr_chunk) == len(next_chun
print(f"Is next chunk shifted by one: {curr_chunk[1:] == next_chunk[:-1]}")
```

```
curr_chunk =>Hello worl
 next_chunk=> ello world
Is curr_chunk and next_chunk same length: True
Is next chunk shifted by one: True
```

Task 2: Implement function to convert to tensors (15 points)

Define a function `to_tensor(string)` that takes a string of characters as input and return torch tensor as output, similar to in the demo in class. Specifically,

1. Create an empty tensor of shape `(len(string), 1, len(all_characters))` using the PyTorch `torch.zeros` function, where `len(string)` is the length of the input string, 1 is the batch size, and `len(all_characters)` is the total number of unique characters in the text data.
2. Loop through each character in the input string and convert it to a one-hot encoded vector.

In [6]:
```
def to_tensor(string):
######### Your Code Here ###########
    tensor = torch.zeros(len(string), 1, len(all_characters))
    for ci, character in enumerate(string):
        tensor[ci][0][all_characters.find(character)] = 1
    return tensor
######### End of your code #########

def get_one_hot_tensors(input, output):
    return to_tensor(input), to_tensor(output)

rng = np.random.RandomState(123) # use this if you need to generate a random sample
input, output = get_random_chunk(file, rng,  50)
print(input.replace('\n', ' '))
print(output.replace('\n', ' '))
input_tensor, output_tensor = get_one_hot_tensors(input, output)
print(f"input shape: {input_tensor.shape}")
print(f"output shape: {output_tensor.shape}")
```

```
g's, which Florizel I now name to you; and with sp
's, which Florizel I now name to you; and with spe
input shape: torch.Size([50, 1, 100])
output shape: torch.Size([50, 1, 100])
```

Exercise 2: Build the RNN model (30 points)

In this part, you will build the RNN model using PyTorch.

- nn.GRU is used to implement the GRU algorithm for processing sequential input data.
  - https://pytorch.org/docs/stable/generated/torch.nn.GRU.html
- The decoder layer is a fully connected neural network layer that maps the output of the GRU layer to the desired output size.
- As we are only implementing a single layer RNN, the model is not powerful enough to learn long-term dependencies in the text data. So don't be surprised if the output sentences are not very meaningful. We are providing you loss plots (`gru_loss_ex2.png`) to help you check if your code is working correctly.

Code instruction:

- init function:
  1. Set `self.rnn_cell` to a nn.GRU
  2. Define a linear decoder layer that maps from the hidden size to the output size
- forward function:
  1. Reshape the input to (1, 1, -1) and pass it to the GRU layer
  2. Reshape the rnn_cell output to (1, -1) and pass it to the decoder layer

In [7]:
```
import torch
import torch.nn as nn
from torch.autograd import Variable

class RNN(nn.Module):
    def __init__(self, input_size, hidden_size, output_size, n_layers=1):
        super(RNN, self).__init__()

        self.input_size = input_size
        self.hidden_size = hidden_size
        self.output_size = output_size
        self.n_layers = n_layers

        # Define modules of RNN
        ######### Your Code Here ###########
        self.rnn_cell = nn.GRU(input_size, hidden_size, n_layers)
        self.linear_decoder = nn.Linear(hidden_size, output_size)
        ######### End of your code #########

    def forward(self, input, hidden):
        ######### Your Code Here ###########
        input = input.view(1, 1, -1)
        input, hidden = self.rnn_cell(input, hidden)
        output = input.view(1, -1)
        output = self.linear_decoder(output)
        ######### End of your code #########
        return output, hidden

    def init_hidden(self):
        return Variable(torch.zeros(self.n_layers, 1, self.hidden_size))
```

In [8]:
```
def train(inp, target, decoder):
    hidden = decoder.init_hidden()
    decoder.zero_grad()
    loss = 0

    input_tensor, target_tensor = get_one_hot_tensors(inp, target)
    for c in range(len(inp)):
        output, hidden = decoder(input_tensor[c], hidden)
        loss += criterion(output, torch.argmax(target_tensor[c]).unsqueeze(0))

    loss.backward()
    decoder_optimizer.step()
    return loss.item() / max_length
```

In [9]:
```python
def evaluate(decoder, prime_str='A', predict_len=100, temperature=0.8):
    hidden = decoder.init_hidden()
    prime_input = to_tensor(prime_str)
    predicted = prime_str

    # Use priming string to "build up" hidden state
    for p in range(len(prime_str) - 1):
        out, hidden = decoder(prime_input[p], hidden)
    inp = prime_input[-1]
    for p in range(predict_len):
        output, hidden = decoder(inp, 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 = to_tensor(predicted_char)

    return predicted
```

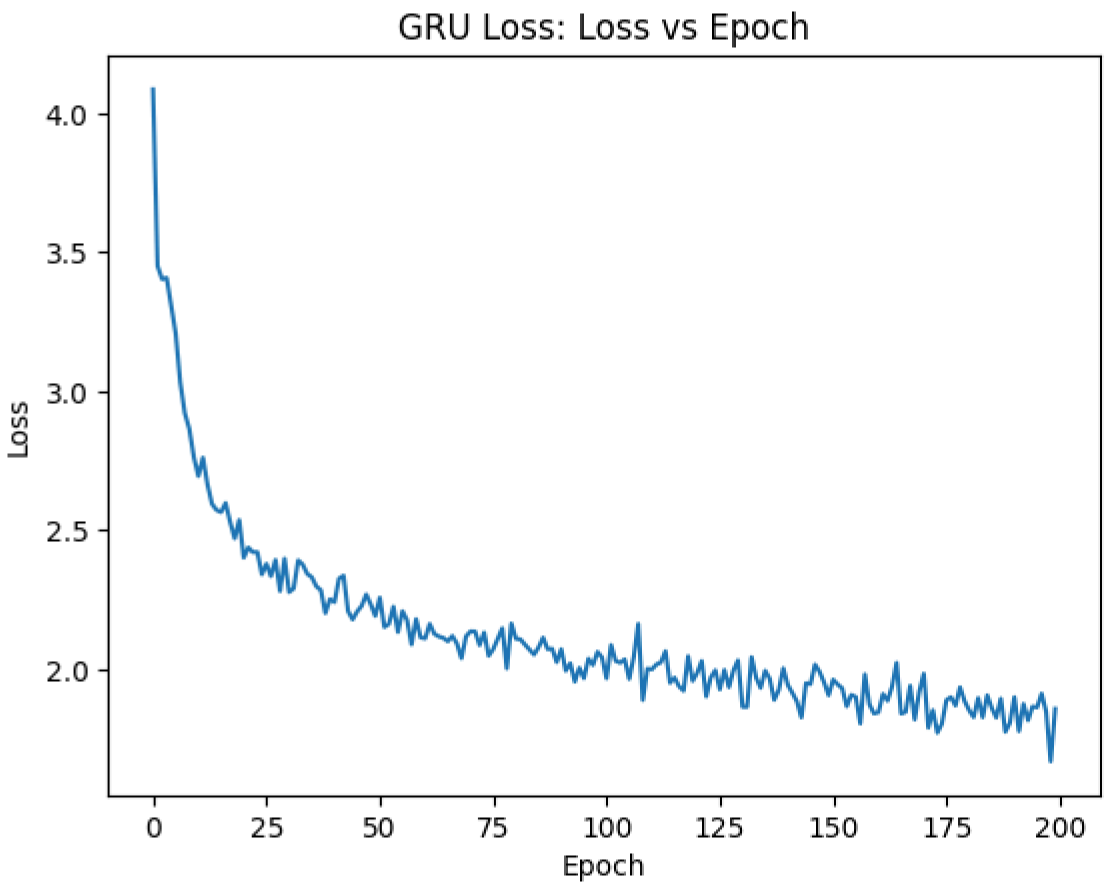
In [10]:
```python
n_epochs = 2000
print_every = 100
plot_every = 10
hidden_size = 100
n_layers = 1
lr = 0.005
max_length = len(all_characters)

decoder = RNN(max_length, hidden_size, max_length)
decoder_optimizer = torch.optim.Adam(decoder.parameters(), lr=lr)
criterion = nn.CrossEntropyLoss()

start = time.time()
all_losses = []
loss_avg = 0
rng = np.random.RandomState(123) # use this if you need to generate a random sample

for epoch in range(1, n_epochs + 1):
    loss = train(*get_random_chunk(file, rng), decoder)
    loss_avg += loss

    if epoch % print_every == 0:
        print(f"[({epoch} {epoch / n_epochs * 100}%) {loss}]")
        print(evaluate(decoder, 'Wh', 100), '\n')

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

print(f"______________________________________________________________")
print(evaluate(decoder, 'Th', 200, temperature=0.2))
```

```
import matplotlib.pyplot as plt
plt.plot(all_losses)
plt.title("GRU Loss: Loss vs Epoch")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.show()
```

```
[(100 5.0%) 2.7360101318359376]
Whe sece benslashe nn por bisecatludd amwredt thensis hen dherel, wilewnne to stle,
ast ones dmondisl

[(200 10.0%) 2.4465716552734373]
Wha the te bor fald woven; bithite end sein I pal torde rounn,
I
To out mave I bove somell rimhe no mu

[(300 15.0%) 2.7336798095703125]
Whis ars Of and and at the his gom beat she tret he hinder his lowe mad, theen Jthea
t neit, theelis sw

[(400 20.0%) 2.3502236938476564]
Whart mead,
But in the
Fourd,

lRUKE:
Na more mipe,
I thant tore ther errery aed, ars the your.

JLAUL

[(500 25.0%) 2.153805694580078]
Whou thit well he ler's Mers
Larst and min tor thim to ham my to nle sfam y yee noth and the lo fom me

[(600 30.0%) 2.0881134033203126]
Whe pome but thit the word.

AUTIO:
O, as is my bind but the botin the brows you hich be Math and the

[(700 35.0%) 2.181220703125]
Why alaues boond, thoo thi he come as the conous
Wo ha vor hom be bot me tho bane I buth buthe shou he

[(800 40.0%) 1.9968531799316407]
Whilf, Till tien caure the her
And in so moriested wipll of my to card, woll sing;
Low she; I saif you

[(900 45.0%) 2.0081381225585937]
Whind way hemy.

NIUENCARUS:
With the chater goton whther noke the thes willo not well weold mans
In p

[(1000 50.0%) 2.2391845703125]
Who is of worp mint withare?

DUCENTE:
I to you lave wads and it to boos with thoughter'd I hearde'
Wh
```

```
[(1100 55.00000000000001%) 2.1599386596679686]
What wo handrem
I this my pethou, grown:
Havenath hes angeean shingnst doo hour, I wome.

CIONATS:
I b

[(1200 60.0%) 1.922183074951172]
While a an your fearto monost you geres weal hous's a wall:
Petars, should us tay heard this and do wi

[(1300 65.0%) 2.159909210205078]
Why may pray hie hardins,
And sher stay, king erthing the mansss,
Ay, thy my father atul, for herein,


[(1400 70.0%) 1.9258718872070313]
Whan for sid of hamm sting
With I love oul Northur all On.

LLOUTERTA:
Nor if as ully my way that hark

[(1500 75.0%) 1.95564697265625]
When the seat?

MIRINGARD
AEd Of Gows:
Hnse the day Rome and be love beence;
But himple take, stare co

[(1600 80.0%) 1.9155738830566407]
When of I chang to grans.

GRENIO:
For sie! thar soland, an then's the piest their on.

JULIET:
Foll D

[(1700 85.0%) 1.8966976928710937]
Whilessell.
Wher for I petrest so more, with you trough the garron
Mores sir onferce!

KING RYCHARNIUS

[(1800 90.0%) 2.1179319763183595]
Where a deat arinedst of that, by be pooe
That thour your nevers Am for or to polds to me frongerake l

[(1900 95.0%) 1.8005809020996093]
Whine;
```

```
Crevilly the heave'd leade a day unding:
Fairsper to my love,
Durts to will in the a bereise,
T

[(2000 100.0%) 1.584326934814453]
What shall and some hamperbe
That leath dire be that that be tell, and centingh mist upon in mustert.


________________________________________________________________
The may the say that I am the say the sentle stands,
And that so the stand that so man and the shall the say the may the lay.

First I lay the say the same and the son and the sentle the seath
That the
```



Exercise 3: Implement an LSTM model (30 points)

Using the equations from the slides in class, write your own LSTM cell module. The code below will use this instead of the GRU cell module and train the model.

Notes:

- Note that for LSTM the hidden state is really both the h_t and C_t so we just unpack the passed hidden state into these two variables at the beginning, and pack them into a tuple for returning.
- We apply a single linear layer to compute all the linear parts of the model that operate on h'_{t-1} and then unpack these using `chunk(4)` into the four separate parts. This is equivalent to having 4 separate linear layers.
- As we are only implementing a single layer RNN, the model is not powerful enough to learn long-term dependencies in the text data. So don't be surprised if the output sentences are not very meaningful. We are providing you loss plots (`lstm_loss_ex3.png`) to help you check if your code is working correctly.

Code instruction:

- forward function:
  1. Apply activation functions to get gates and new cell state information
  2. Calculate the new cell state (c_new)
  3. Calculate the new hidden state (h_new)

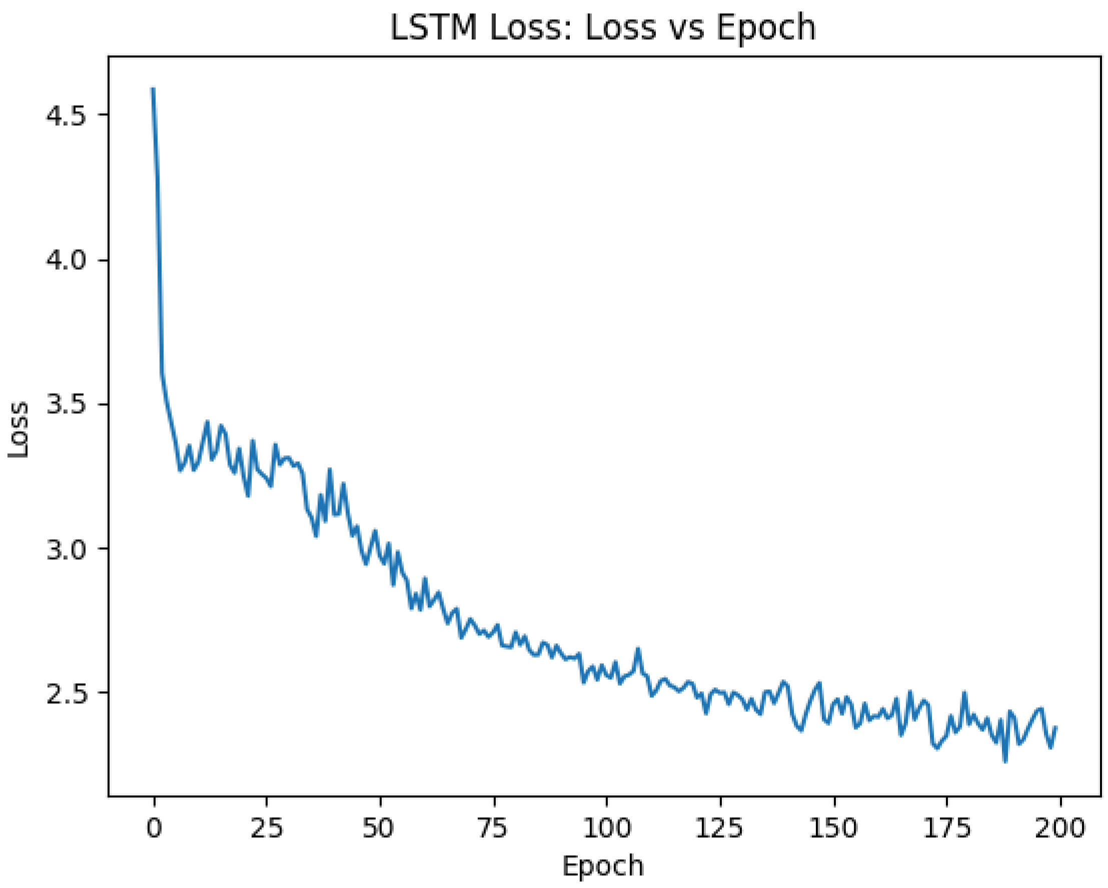
In [11]:
```
class LSTMCell(nn.Module):
    def __init__(self, input_size, hidden_size, bias=True):
        super(LSTMCell, self).__init__()
        self.input_size = input_size
        self.hidden_size = hidden_size
        self.bias = bias

        self.xh = nn.Linear(input_size, hidden_size * 4, bias=bias)
        self.hh = nn.Linear(hidden_size, hidden_size * 4, bias=bias)
        self.reset_parameters()

    def reset_parameters(self):
        std = 1.0 / np.sqrt(self.hidden_size)
        for w in self.parameters():
            w.data.uniform_(-std, std)

    def forward(self, input, hidden=None):
        # Unpack hidden state and cell state
        hx, cx = hidden

        # Apply linear layers to input and hidden state
        linear = self.xh(input) + self.hh(hx)

        # Get outputs of applying a linear transform for each part of the LSTM
        input_linear, forget_linear, cell_linear, output_linear = linear.reshape(-1

        ######### Your Code Here ###########
        f_gate = torch.sigmoid(forget_linear)
        i_gate = torch.sigmoid(input_linear)
        o_gate = torch.sigmoid(output_linear)
        c_state = torch.tanh(cell_linear)

        c_new = f_gate * cx + i_gate * c_state
        h_new = o_gate * torch.tanh(c_new)
        ######### End of your code #########

        # Pack cell state $C_t$ and hidden state $h_t$ into a single hidden state t
        output = h_new # For LSTM the output is just the hidden state
        hidden = (h_new, c_new) # Packed h and C
        return output, hidden
```

In [12]:
```
lr = 0.001
class LSTM_RNN(RNN):
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        # Replace the gru cell with LSTM cell
```

```python
        self.rnn_cell = LSTMCell(max_length, hidden_size, max_length)

    def init_hidden(self):
        # LSTM cells need two hidden variables in a tuple of (h_t,C_t)
        return (Variable(torch.zeros(1, 1, self.hidden_size)), Variable(torch.zeros

decoder = LSTM_RNN(max_length, hidden_size, max_length)
decoder_optimizer = torch.optim.Adam(decoder.parameters(), lr=lr)

all_losses = []
loss_avg = 0
rng = np.random.RandomState(123) # use this if you need to generate a random sample

for epoch in range(1, n_epochs + 1):
    loss = train(*get_random_chunk(file, rng),decoder)
    loss_avg += loss

    if epoch % print_every == 0:
        print(f"[({epoch} {epoch / n_epochs * 100}%) {loss}]")
        print(evaluate(decoder, 'Wh', 100), '\n')

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

print(f"______________________________________________________________")
print(evaluate(decoder, 'Th', 200, temperature=0.2))

plt.plot(all_losses)
plt.title("LSTM Loss: Loss vs Epoch")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.show()
```

```
[(100 5.0%) 3.169309387207031]
Whh bEuuye l mg  lahhsa 'oi  ren rst  hh oi  iCc
aegsIgef  aelneyeeo,rtehea;ea2I nlA rdn aealnraesnIm

[(200 10.0%) 3.1141537475585936]
Whadus ci iremrno  r,Ie eoroh ynohrolo shrIG ;easl fIrf a oktelfi te   ehcr,w,ep  e
biioi
o ourmolbIoO

[(300 15.0%) 3.507082214355469]
Wh,:t  ',ilimrn  ushhais iitt odhhreHe eeed hmls:o.ymwDanl ,ert m i  mIer fh
iera d oh;ohmr d l,.   ai

[(400 20.0%) 3.340348205566406]
Whw'DPe
 dt mry gt eaew
ett nas
sea re tmacthA
sira
 dis nveLvs Elv
ci uh
bd tmd I
 iy r, f rtirie h

[(500 25.0%) 2.941952819824219]
Whow, ha bAt ea  etse fteno soo  emt fl nifeine
ee msieel o, eds eeeo rpen lIld toooAu
 sis mOetes si

[(600 30.0%) 2.77449951171875]
Whdehd boh aan ih uts te re too do fimnt tonlrst whoy amt. terss on, se ye amy nomr
tet n ly hiee le


[(700 35.0%) 2.827923583984375]
WhLld eitersesbt or tels mumot yorh an eh aoro
to hyansrwe soe doR iheo
gf hos od seor tth hh a toial

[(800 40.0%) 2.6182791137695314]
Wh fo sey mee wen thr thans al srse sore
Oiw ahg con merg? u:
awne mame
G sana Itf dhf tho hatsuusrato

[(900 45.0%) 2.5956710815429687]
Who Aa yheve aod Feud anelen py taln hhaveane f: thncmeisous wesren t'lr eps bntenti
nhe yhet sees that

[(1000 50.0%) 2.7130255126953124]
Whl, poace wo tin  ohe he fhe wimenset 'us thos nheall be tore the tamer tins bRar w
errvre an cone cha

[(1100 55.00000000000001%) 2.6705816650390624]
Whe syanwmuk arsathe hi in whis the sou gor Hor teas toed soud and yee !is, lithos h
e thame
```

```
e:
Tho lo

[(1200 60.0%) 2.4326609802246093]
Whes thir roworgsit ie amecor sadqpeovergh sule bo silothe dad thavangy ke lone west
e, nose san wol bh

[(1300 65.0%) 2.510792541503906]
Whers hor lt chine to thirt at ho foy;

hiGd mo thith teve ande be cous rand thak, loul wh toid fhetad

[(1400 70.0%) 2.4844602966308593]
Whe senallor soe dote fhale and, mithefpthet  eorat anghed ore ther uos to fon Tors,
Witor.

ArIOB:
Tu

[(1500 75.0%) 2.4566079711914064]
Whe por th eat the be heind arl she forest, ork, Giondo,
Th to y peem the-r aWhe tis me thy mor yorre

[(1600 80.0%) 2.5917849731445313]
Whon  or sret me me but ot sondes witiad th ainald Coarreafo
 andsion, ne waur thand on one loded thon

[(1700 85.0%) 2.4319419860839844]
Whet he fethe sle tot our ay:
weet mrowet y mith d warerse.

I INTOI:
Shere hi we int ougret forers th

[(1800 90.0%) 2.5739572143554685]
Whes cou meres the the btet the romlr the thove the theF powe,
Ks erved he ldocn nof for faf wor woul

[(1900 95.0%) 2.3195281982421876]
Wh, urthe poun, wou heals beuve pove soessksmetin ote Jusle beod  yeand beies the be
ndius, ad busd yar

[(2000 100.0%) 2.188473815917969]
Whe bathur mas and pyagenso sow bitt locipe.


PhRIOTN:
Whath?

'eod Aspathere weatle mor meast it mat

______________________________________________________________
The ther and ther there the the the the the the the the the the the the ther thar th
e the that an the the the the the there the there the the the that and the thar ther
e the the the the the the the the
```



Exercise 4: Implement your own GRU (10 points)

Same as above but implement a GRU instead of an LSTM module. An exmaple of GRU architecture can be found from the lecture slide: https://www.davidinouye.com/course/ece57000-fall-2023/lectures/recurrent-neural-networks.pdf

You output loss plot should be similar in Exercise 2.

Code instruction:

- forward function:
  1. Concatenate hidden and input to get h_prime (see torch.cat)
  2. Use self.h2z to calculate z_t
  3. Use self.h2r to calculate r_t
  4. Use Hadamard product of r_t and hx and concatenate with input
  5. Then use h2h to calculate new hidden information h_tbar
  6. Update h_t with z_t, hx, and h_tbar

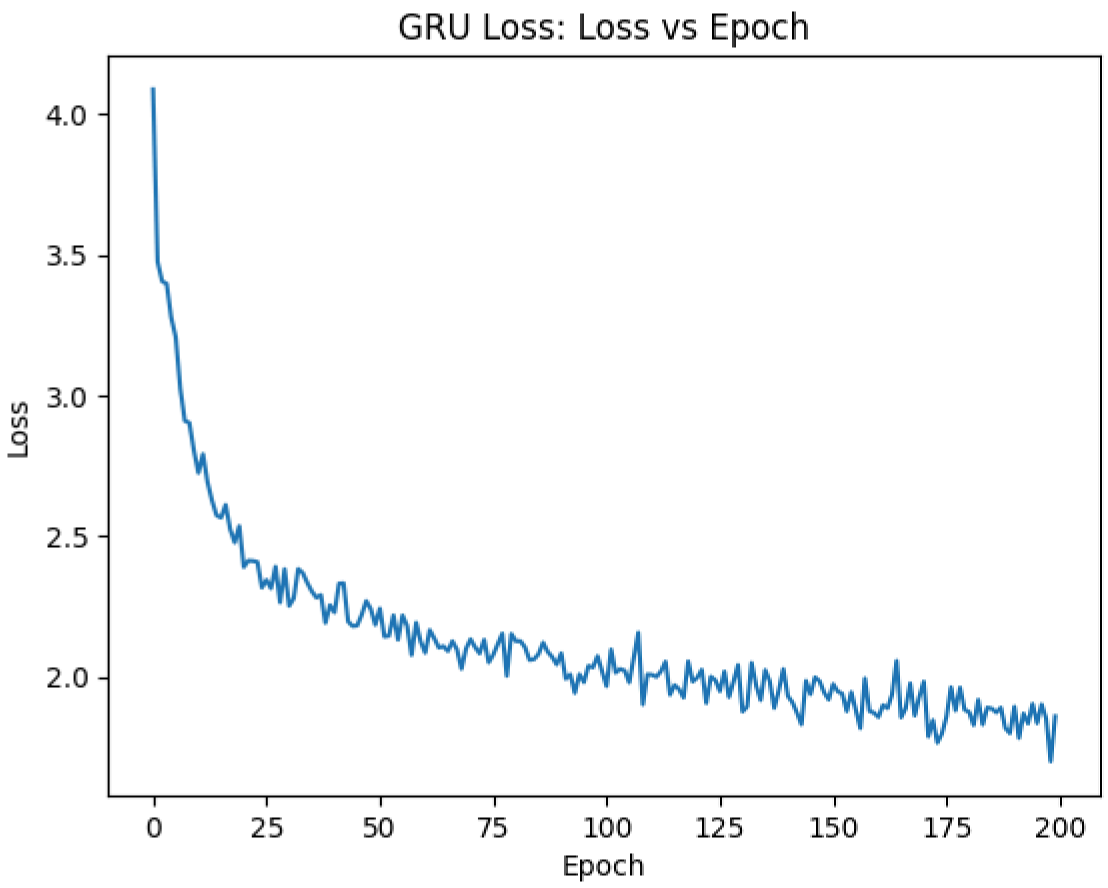
In [13]:
```
class GRUCell(nn.Module):
    def __init__(self, input_size, hidden_size, bias=True):
        super(GRUCell, self).__init__()
        self.input_size = input_size
        self.hidden_size = hidden_size
        self.bias = bias

        self.h2z = nn.Linear(input_size + hidden_size, hidden_size)
        self.h2r = nn.Linear(input_size + hidden_size, hidden_size)
        self.h2h = nn.Linear(input_size + hidden_size, hidden_size)
        self.reset_parameters()


    def reset_parameters(self):
        std = 1.0 / np.sqrt(self.hidden_size)
        for w in self.parameters():
            w.data.uniform_(-std, std)

    def forward(self, input, hx=None):
        # Inputs:
        #       input: of shape (batch_size, input_size)
        #       hx: of shape (batch_size, hidden_size)
        # Output:
        #       h_t, h_t: h_t is of shape (batch_size, hidden_size)

        if hx is None:
            hx = Variable(input.new_zeros(input.size(0), self.hidden_size))

        ######### Your Code Here ###########
        h_prime = torch.cat((hx, input), 2)
        z_gate = torch.sigmoid(self.h2z(h_prime))
        r_gate = torch.sigmoid(self.h2r(h_prime))
        h_state = torch.tanh(self.h2h(torch.cat((r_gate * hx, input), 2)))
        h_t = (1 - z_gate) * hx + z_gate * h_state
        ######### End of your code #########

        # Reshape h_t match input size
        h_t = h_t.reshape(1, 1, -1)

        return h_t, h_t   # Output and hidden are both h_t
```

In [14]:
```
n_epochs = 2000
print_every = 100
plot_every = 10
hidden_size = 100
n_layers = 1
lr = 0.005
max_length = len(all_characters)

# Replace the RNN module with your implemented GRUcell
class GRU_RNN(RNN):
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        # Replace wtih your gru cell
        self.rnn_cell = GRUCell(max_length, hidden_size, max_length)

decoder = GRU_RNN(max_length, hidden_size, max_length)
decoder_optimizer = torch.optim.Adam(decoder.parameters(), lr=lr)

all_losses = []
loss_avg = 0
rng = np.random.RandomState(123) # use this if you need to generate a random sample

for epoch in range(1, n_epochs + 1):
    loss = train(*get_random_chunk(file, rng),decoder)
    loss_avg += loss

    if epoch % print_every == 0:
        print(f"[({epoch} {epoch / n_epochs * 100}%) {loss}]")
        print(evaluate(decoder, 'Wh', 100), '\n')

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

print(f"______________________________________________________________")
print(evaluate(decoder, 'Th', 200, temperature=0.2))

plt.plot(all_losses)
plt.title("GRU Loss: Loss vs Epoch")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.show()
```

```
[(100 5.0%) 2.8041796875]
Wharg oe, ceicengt hegc that. saa t geon !ind tyitd rag
oes  noI se,dd s lent vins aiss hok d so no ne

[(200 10.0%) 2.413399353027344]
Who feal an workere retent of an hans andsput male s art or sowen at- coud arf man t
our thans tof oo c

[(300 15.0%) 2.6813140869140626]
Whe and
Cost thes thes wam fard in hind: atchis the thil shrot youl then the marend ngucl ur
e dous arc

[(400 20.0%) 2.3846539306640624]
Whim my, and noy of to dimeay and owe ailit,
Ire akemer onenore the afr wame now foat om ald and it be

[(500 25.0%) 2.1722682189941405]
Whis stand au'd to hame hall thee thy a certhy theme ploll diand
What or trume and, I hook tive det lo

[(600 30.0%) 2.0884832763671874]
Whith fullfoald you levee peat yout of at of at and and at the the I the blyou thiti
s with is he seron

[(700 35.0%) 2.1170339965820313]
Whoug,
Thou be has inst anour thou so buading goms more?

CUCIO:
Haprichs to the that of the sar me pa

[(800 40.0%) 1.9792388916015624]
Whar: Whimes ard in thy have to const
rom hor shad siefnay you my lated, a suor:
A him and ethere now

[(900 45.0%) 2.035708312988281]
Whal then geit;
For your que with theswer yould in thee mais and thy ban,
If yoir shes shell, as I the

[(1000 50.0%) 2.1952334594726564]
Whand what to courd the tone.

MIRINE
O ENTIO:
On of then tith menoul ay ho cear's in than vere and ba

[(1100 55.00000000000001%) 2.1536346435546876]
Whath nom fard the male of wian,
Aed, thou in shall sorgh now hadwers stuse:
bad and sweuch nom serse,

[(1200 60.0%) 1.9071165466308593]
Whis willd becureed?
```

```
I war I marespermencut: go sieed to you do wifet that forsught pived of lavespabl

[(1300 65.0%) 2.130661926269531]
What not wad sor fathis is the has fou thes not to her enothat
Hy his wast ghaist thy, the mathting no

[(1400 70.0%) 1.8901580810546874]
What my not in of verer art
As is spares as is istheres! that he come To shidm! but make my counter's

[(1500 75.0%) 2.0188636779785156]
Whilce, I will sances may were my likend it fich bed.

LION ENAUS:
And thish my nawh me cimolther same

[(1600 80.0%) 1.9607371520996093]
Whrust rither with moon to prentieng

MORIET:
Thou trokn,'d and prince for hithan he
Which of the his

[(1700 85.0%) 1.8519338989257812]
Whis seat hampield!

GLOUTES:
Whace on that Vores.
And mise I ay that shall: no bages, that for clove,

[(1800 90.0%) 2.148412628173828]
Why, theme pleseford, as be of hard refore to the sie.

HARTINGHANG HBANUS:
As plight at more:
God the

[(1900 95.0%) 1.8337904357910155]
Whild, intonk,
Till my lord did, lood asperef!
For I heave, will therefore that Came the, word deavind

[(2000 100.0%) 1.575778045654297]
Whire, sun: and soter that nece he dark:
Where their I mare and your potedve and on the great parny:
W
```

```
The son the son son the see the sance and the son the peater and the have the care t
hat shall the son the death.

SICINIUS:
I with the shall son the see have the grave and the see the son of the singer
```


GRU Loss: Loss vs Epoch
Loss
Epoch
4.0
3.5
3.0
2.5
2.0
0
25
50
75
100
125
150
175
200

```
In [ ]:
```

```
In [ ]:
```