Homework 8 - Berkeley STAT 157

Prerequisites - Load Data

In [63]:
```
import urllib3
import collections
import re
shakespeare = 'http://www.gutenberg.org/files/100/100-0.txt'

http = urllib3.PoolManager()
text = http.request('GET', shakespeare).data.decode('utf-8')
raw_dataset = ' '.join(re.sub('[^A-Za-z]+', ' ', text).lower().split())

print('number of characters: ', len(raw_dataset))
print(raw_dataset[0:70])
```

```
number of characters:  5032359
project gutenberg s the complete works of william shakespeare by willi
```

This dataset is quite a bit bigger than the time machine (5 million vs. 160k). For convenience we also include the remaining preprocessing steps. A bigger dataset will allow us to generate more meaningful models.

In [64]:
```
idx_to_char = list(set(raw_dataset))
char_to_idx = dict([(char, i) for i, char in enumerate(idx_to_char)])
# vocabulary size, i.e., the number of letters in English
vocab_size = len(char_to_idx)
# whole corpus
corpus_indices = [char_to_idx[char] for char in raw_dataset]
sample = corpus_indices[:20]
print('chars:', ''.join([idx_to_char[idx] for idx in sample]))
print('indices:', sample)
# for training
train_indices = corpus_indices[:-100000]
# for testing
test_indices = corpus_indices[-100000:]
```

```
chars: project gutenberg s
indices: [22, 19, 21, 16, 14, 18, 9, 13, 7, 0, 9, 14, 26, 5, 14, 19, 7, 13, 1, 13]
```

Lastly we import other useful libraries to help you getting started.

In [65]:
```
import d2l
import math
import mxnet as mx
from mxnet import autograd, gluon, init, nd
from mxnet.gluon import loss as gloss, nn, rnn
import time
```

1. Train Recurrent Latent Variable Models

Train a number of different latent variable models using `train_indices` to assess their performance. By default pick 256 dimensions for the hidden units. You can use the codes provided in the class. Also, we strongly encourage you to use the Gluon implementation since it's a lot faster than building it from scratch.

1. Train a single-layer RNN (with latent variables).
2. Train a single-layer GRU.
3. Train a single-layer LSTM.
4. Train a two-layer LSTM.

How low can you drive the perplexity? Can you reproduce some of Shakespeare's finest writing (generate 200 characters). Start the sequence generator with `But Brutus is an honorable man`. Experiment with a number of settings:

- Number of hidden units.
- Embedding length.
- Gradient clipping.
- Number of iterations.
- Learning rate.

Save the models (at least in memory since you'll need them in the next exercise.

In [66]:
```
def predict_rnn_gluon(prefix, num_chars, model, vocab_size, ctx, idx_to_char, char_to_idx):
    state = model.begin_state(batch_size=1, ctx=ctx)
    output = [char_to_idx[prefix[0]]]
    for t in range(num_chars + len(prefix) - 1):
        X = nd.array([output[-1]], ctx=ctx).reshape((1, 1))
        (Y, state) = model(X, state)
        if t < len(prefix) - 1:
            output.append(char_to_idx[prefix[t + 1]])
        else:
            output.append(int(Y.argmax(axis=1).asscalar()))
    return ''.join([idx_to_char[i] for i in output])

num_hiddens = 256
rnn_layer = rnn.RNN(num_hiddens)
rnn_layer.initialize()
num_steps = 35
num_epochs, batch_size, lr, clipping_theta = 200, 32, 30, 1e-2
```

In [67]:
```
ctx = d2l.try_gpu()
single_rnn_model = d2l.RNNModel(rnn_layer, vocab_size)
single_rnn_model.initialize(force_reinit=True, ctx = ctx)
pred_period, pred_len, prefixes = 25, 100, ['but brutus is an honorable man']
```

Question 1 General Discoveries

- Learning rate: the model with smaller learning rate shows better performance. $lr = 30$ in our code.
- Hidden Units: the model with larger hidden units shows better performance, but it could not be too large. $hidden\_units$ = 320 in the GRU model.

- Embedding length: It seems that more we embedded, more precise the model would be. However, the memory of the computer need to memorize more. So we keep it as the same.
- Gradient clipping: We tried 1e-3 and 1e-1, the model shows worse performance. Therefore, we remian the clipping parameter unchanged.

Question 1.1: Single-layer RNN

In [68]:
```
# for question 1.1: train a single-layer RNN
d2l.train_and_predict_rnn_gluon(single_rnn_model, num_hiddens, vocab_size, ctx, train_indices, idx_to_char,\
                                char_to_idx, num_epochs, num_steps, lr, clipping_theta, batch_size,\
                                pred_period, pred_len, prefixes)
```

```
epoch 25, perplexity 3.698738, time 19.11 sec
 - but brutus is an honorable man s sooth the state the state the state the state the state the state the state the state the state t
epoch 50, perplexity 3.644675, time 18.97 sec
 - but brutus is an honorable man s son and the state and the state and the state and the state and the state and the state and the stat e and the s
epoch 75, perplexity 3.621387, time 18.90 sec
 - but brutus is an honorable man s soul that i shall be the state the state the state the state the state the state the s tate the st
epoch 100, perplexity 3.606846, time 18.64 sec
 - but brutus is an honorable man and the service and the service and the service and the service and the service and the service and
epoch 125, perplexity 3.597160, time 18.68 sec
 - but brutus is an honorable man s soul and the state they shall be the state they shall be the state they shall be the s tate they s
epoch 150, perplexity 3.590280, time 19.11 sec
 - but brutus is an honorable man s soul and the sea and the sea and the sea and the sea and the sea and the sea and the s ea and the
epoch 175, perplexity 3.584725, time 18.54 sec
 - but brutus is an honorable man s soul and the sea and the sea and the sea and the sea and the sea and the sea and the s ea and the
epoch 200, perplexity 3.580413, time 18.83 sec
 - but brutus is an honorable man s son and the sea and the sea and the sea and the sea and the sea and the sea and the se a and the s
```

In [69]:
```
predict_rnn_gluon('but brutus is an honorable man', 200, single_rnn_model, vocab_size, ctx,\
                  idx_to_char, char_to_idx)
```

Out[69]:
```
'but brutus is an honorable man s son and the sea and the sea and the sea and the sea and the sea and the sea and the sea and the sea and the sea and the sea and the sea and the sea and the sea a'
```

Question 1.2: Single Layer LSTM

In [70]:
```
# for question 1.2, single layer lstm
lstm_layer = rnn.LSTM(num_hiddens)
single_lstm_model = d2l.RNNModel(lstm_layer, vocab_size)
d2l.train_and_predict_rnn_gluon(single_lstm_model, num_hiddens, vocab_size, ctx, train_indices, idx_to_char,\
                                char_to_idx, num_epochs, num_steps, lr, clipping_theta, batch_size,\
                                pred_period, pred_len, prefixes)
```

```
epoch 25, perplexity 3.382777, time 20.38 sec
 - but brutus is an honorable man that i will see thee they are they are they are they are they are they are they are they are they are they a
epoch 50, perplexity 3.251212, time 20.35 sec
 - but brutus is an honorable man that i may be so soon as they are they are they are they are they are they are they are they are they are th
epoch 75, perplexity 3.200615, time 20.43 sec
 - but brutus is an honorable man that i may say they are a man that i may say they are a man that i may say they are a ma
n that i ma
epoch 100, perplexity 3.174369, time 20.29 sec
 - but brutus is an honorable man that i may see thee when they shall be so soon as they are they are they are they are th
ey are they
epoch 125, perplexity 3.156930, time 20.20 sec
 - but brutus is an honorable man that s a soul of the state of the state of the country s son the sea will be the sea and
the consta
epoch 150, perplexity 3.145716, time 20.13 sec
 - but brutus is an honorable man and there is no such a state of the state of the state of the country s son the sea with
his soul t
epoch 175, perplexity 3.136352, time 20.24 sec
 - but brutus is an honorable man of man the state of the country s son the sea and the consequence and the stars of the s
tate of the
epoch 200, perplexity 3.131308, time 20.36 sec
 - but brutus is an honorable man of the state of the counterfeit of the state and they are not the state of the counterfe
it of the s
```

In [71]:
```
predict_rnn_gluon('but brutus is an honorable man', 200, single_lstm_model, vocab_size, ctx,\
                  idx_to_char, char_to_idx)
```

Out[71]:
```
'but brutus is an honorable man of the state of the counterfeit of the state and they are not the state of the counterfeit
of the state and they are not the state of the counterfeit of the state and they are not the state of the co'
```

Question 1.3: Single Layer GRU

In [72]:
```
# for question 1.3, single layer gru
gru_layer = rnn.GRU(num_hiddens)
single_gru_model = d2l.RNNModel(gru_layer, vocab_size)
d2l.train_and_predict_rnn_gluon(single_gru_model, 320, vocab_size, ctx, train_indices, idx_to_char,\
                                char_to_idx, num_epochs, num_steps, lr, clipping_theta, batch_size,\
                                pred_period, pred_len, prefixes)
```

```
epoch 25, perplexity 3.432897, time 20.51 sec
 - but brutus is an honorable man s son and they are and they are and they are and they are and they are and they are and
they are an
epoch 50, perplexity 3.347386, time 19.99 sec
 - but brutus is an honorable man s face they are they shall be so for the sea and the sea and the count s another they ar
e not the s
epoch 75, perplexity 3.318542, time 20.20 sec
 - but brutus is an honorable man that i am a man and they are they are they are they are they are they are they are they
are they ar
epoch 100, perplexity 3.303343, time 20.56 sec
 - but brutus is an honorable man s son and the sea and the sea and the sea and the sea and the sea and the sea and the se
a and the s
epoch 125, perplexity 3.296494, time 19.88 sec
 - but brutus is an honorable man s son the sea and the sea and the sea and the sea and the sea and the sea and the sea an
d the sea a
epoch 150, perplexity 3.289302, time 19.99 sec
 - but brutus is an honorable man that i will stand for thee they are and there s a fool and there s a fool and there s a
fool and th
epoch 175, perplexity 3.283573, time 20.18 sec
 - but brutus is an honorable man that is the sun s and they are so much they say the stage and then they say the stage an
d then they
epoch 200, perplexity 3.279063, time 19.81 sec
 - but brutus is an honorable man s and the sea and the country s son and then they say the stage and there s a present an
d so i will
```

In [73]:
```
predict_rnn_gluon('but brutus is an honorable man', 200, single_gru_model, vocab_size, ctx,\
                  idx_to_char, char_to_idx)
```

Out[73]:
```
'but brutus is an honorable man s and the sea and the country s son and then they say the stage and there s a present and so i will not see thee when i shall see thee when i shall see thee when i shall see thee when i shall see the'
```

Question 1.4: Double Layer LSTM -- Best Performance

In [74]:
```
# for question 1.4, double layers lstm
lstm_layer = rnn.LSTM(num_hiddens, num_layers=2)
double_lstm_model = d2l.RNNModel(lstm_layer, vocab_size)
d2l.train_and_predict_rnn_gluon(double_lstm_model, num_hiddens, vocab_size, ctx, train_indices, idx_to_char,\
                                char_to_idx, num_epochs, num_steps, lr, clipping_theta, batch_size,\
                                pred_period, pred_len, prefixes)
```

```
epoch 25, perplexity 3.214155, time 30.25 sec
 - but brutus is an honorable man that s a foot of the country s head and then they say they say they say they say they sa
y they say
epoch 50, perplexity 3.007819, time 30.38 sec
 - but brutus is an honorable man that would be so shall i say the countess is the sea and the senate have and they are th
ey are they
epoch 75, perplexity 2.915687, time 30.43 sec
 - but brutus is an honorable man the thing is the fairest enter servant to the castle enter sir toby and sir toby and sir
toby and s
epoch 100, perplexity 2.861589, time 30.57 sec
 - but brutus is an honorable man the third time the sea what s the matter from the capitol the commons have their shame t
hat they wo
epoch 125, perplexity 2.824575, time 30.29 sec
 - but brutus is an honorable mantle s brother s life the first the best and so shall be the most unsure they would not ha
ve the sena
epoch 150, perplexity 2.799474, time 30.15 sec
 - but brutus is an honorable man that s the song the fight of heaven and the son of my soul the streets of engling hands
and their t
epoch 175, perplexity 2.780415, time 30.20 sec
 - but brutus is an honorable man may well be so betime as i am a soldier that i have seen the day will be the search d he
ad of the c
epoch 200, perplexity 2.764609, time 31.21 sec
 - but brutus is an honorable man s house enter provost and so please you sir toby what she says she did and then to see t
he stars of
```

In [75]:
```
predict_rnn_gluon('but brutus is an honorable man', 200, double_lstm_model, vocab_size, ctx,\
                  idx_to_char, char_to_idx)
```

Out[75]:
```
'but brutus is an honorable man s house enter provost and so please you sir toby what she says she did and then to see the
stars of men the sea who should say they were not a stranger to the senate house s palace stephano i say sir'
```

2. Test Error

So far we measured perplexity only on the training set.

1. Implement a perplexity calculator that does not involve training.
2. Compute the perplexity of the best models in each of the 4 categories on the test set. By how much does it differ?

In [76]:
```
# for question 2
# This function is used to test perplexity on test set, not including training
def test_perplexity(model, ctx, corpus_indices, idx_to_char, char_to_idx, num_epochs, num_steps, batch_size):
    for epoch in range(num_epochs):
        loss = gloss.SoftmaxCrossEntropyLoss()
        l_sum, n, start = 0.0, 0, time.time()
        data_iter = d2l.data_iter_consecutive(
            corpus_indices, batch_size, num_steps, ctx)
        state = model.begin_state(batch_size=batch_size, ctx=ctx)
        for X, Y in data_iter:
            for s in state:
                s.detach()
            with autograd.record():
                (output, state) = model(X, state)
                y = Y.T.reshape((-1,))
                l = loss(output, y).mean()
            l_sum += l.asscalar() * y.size
            n += y.size
        print('epoch %d, perplexity %f, time %.2f sec' % (
            epoch + 1, math.exp(l_sum / n), time.time() - start))
```

In [77]:
```
num_epochs, num_steps, batch_size = 5, 50, 1
```

In [82]:
```
# test perplexity single rnn
test_perplexity(single_rnn_model, ctx, test_indices, idx_to_char, char_to_idx, num_epochs, num_steps, batch_size)
```

```
epoch 1, perplexity 4.705128, time 3.91 sec
epoch 2, perplexity 4.706620, time 3.42 sec
epoch 3, perplexity 4.704766, time 3.39 sec
epoch 4, perplexity 4.704266, time 3.37 sec
epoch 5, perplexity 4.704968, time 3.38 sec
```

In [79]:
```
# test perplexity on single lstm
test_perplexity(single_lstm_model, ctx, test_indices, idx_to_char, char_to_idx, num_epochs, num_steps, batch_size)
```

```
epoch 1, perplexity 4.527095, time 3.33 sec
epoch 2, perplexity 4.526816, time 3.42 sec
epoch 3, perplexity 4.527897, time 3.64 sec
epoch 4, perplexity 4.525007, time 3.63 sec
epoch 5, perplexity 4.524860, time 3.47 sec
```

In [80]:
```
# test perplexity on gru
test_perplexity(single_gru_model, ctx, test_indices, idx_to_char, char_to_idx, num_epochs, num_steps, batch_size)
```

```
epoch 1, perplexity 4.599602, time 3.33 sec
epoch 2, perplexity 4.598010, time 3.42 sec
epoch 3, perplexity 4.597947, time 3.44 sec
epoch 4, perplexity 4.597218, time 3.42 sec
epoch 5, perplexity 4.597271, time 3.39 sec
```

In [81]:
```
# test perplexity on double lstm
test_perplexity(double_lstm_model, ctx, test_indices, idx_to_char, char_to_idx, num_epochs, num_steps, batch_size)
```

```
epoch 1, perplexity 5.077668, time 7.28 sec
epoch 2, perplexity 5.082510, time 4.92 sec
epoch 3, perplexity 5.080637, time 4.86 sec
epoch 4, perplexity 5.082147, time 5.42 sec
epoch 5, perplexity 5.079242, time 5.48 sec
```

In [ ]:
```

```