ECE194N: Homework 3

Topics: RNN Problem

Due: May 29

RNN: In this problem we will be implementing an RNN. The objective of the RNN is that given a sequence of words, it will try to guess the next word. For this task we will follow the RNN tutorial for Tensorflow which can be found from this link: https://www.tensorflow.org/tutorials/recurrent

Instructions to run the code. My code here has two run modes as described below:

1. **Train**
   ```
   @bitcoin:~/rnn$ CUDA_VISIBLE_DEVICES=2 python3 ptb_word_lm.py --data_path=simple-examples/data/ --
   model="medium"
   ```
   In this mode we are training the network, computing validation, training losses and saving the trained network as a checkpoint.

2. **Test**
   ```
   @bitcoin:~/rnn$ CUDA_VISIBLE_DEVICES=2 python3 ptb_word_lm.py --data_path=simple-examples/data/ --
   model="medium" --gen_text=True
   ```
   Here we are loading the saved network and using that to generate sentences

a.) displays the word ids and the sentences for the first 10 samples in the validation set.

This part can be found in the Jupyter notebook called rnn.ipynb

```
# Read PTB sample data
data_path = "simple-examples/data/"
word_to_id = reader._build_vocab(os.path.join(data_path, "ptb.train.txt"))
raw_data = reader.ptb_raw_data(data_path)
train_data, valid_data, test_data, vocabulary = raw_data

reversed_dictionary = dict(zip(word_to_id.values(), word_to_id.keys()))
print(" ".join([reversed_dictionary[x] for x in train_data[:10]]))

print("Vocabulary Size: ", vocabulary,"\n")
eos_id = word_to_id["<eos>"]
word_txt = ""
word_ids = []
sent_cnt = 0
for id_valid in valid_data[0:10000]:
  kv = {k:v for k, v in word_to_id.items() if v == id_valid}
  kv = kv.popitem()
  if(id_valid == eos_id): # End of sentence
    print((sent_cnt+1),".)",word_txt,"\n ",word_ids)
    word_txt = ""
    word_ids = []
    sent_cnt = sent_cnt + 1
```

```
      continue
   if(sent_cnt == 10):  # Finish after 10 sentences
      break
   # Add word to sentence
   word_txt = word_txt + " " + kv[0]
   word_ids.append(kv[1])
```

Result:

```
Vocabulary Size:  10000

1.)  consumers may want to move their telephones a little closer to the tv set
  [1132, 93, 358, 5, 329, 51, 9836, 6, 326, 2476, 5, 0, 662, 388]
2.)  <unk> <unk> watching abc 's monday night football can now vote during <unk> for the greatest
play in N years from among four or five <unk> <unk>
  [1, 1, 2974, 2158, 9, 381, 1068, 2347, 89, 99, 847, 198, 1, 11, 0, 3383, 1119, 7, 3, 72, 20, 21
1, 346, 36, 258, 1, 1]
3.)  two weeks ago viewers of several nbc <unk> consumer segments started calling a N number for
advice on various <unk> issues
  [75, 422, 195, 3917, 4, 249, 1795, 1, 580, 3528, 892, 2374, 6, 3, 297, 11, 2709, 16, 1186, 1, 2
50]
4.)  and the new syndicated reality show hard copy records viewers ' opinions for possible airing
on the next day 's show
  [8, 0, 35, 9922, 3747, 464, 710, 2998, 2037, 3917, 134, 6145, 11, 494, 5894, 16, 0, 130, 272, 9
, 464]
5.)  interactive telephone technology has taken a new leap in <unk> and television programmers ar
e racing to exploit the possibilities
  [9958, 732, 503, 30, 641, 6, 35, 6498, 7, 1, 8, 761, 9967, 26, 6587, 5, 6415, 0, 6574]
6.)  eventually viewers may grow <unk> with the technology and <unk> the cost
  [1413, 3917, 93, 1552, 1, 22, 0, 503, 8, 1, 0, 361]
7.)  but right now programmers are figuring that viewers who are busy dialing up a range of servi
ces may put down their <unk> control <unk> and stay <unk>
  [29, 382, 99, 9967, 26, 7428, 10, 3917, 56, 26, 3248, 8846, 52, 6, 880, 4, 323, 93, 335, 118, 5
1, 1, 350, 1, 8, 1337, 1]
8.)  we 've been spending a lot of time in los angeles talking to tv production people says mike
parks president of call interactive which supplied technology for both abc sports and nbc 's cons
umer minutes
  [64, 573, 58, 508, 6, 581, 4, 103, 7, 639, 747, 1921, 5, 662, 359, 108, 44, 5458, 6149, 70, 4,
786, 9958, 41, 7746, 503, 11, 179, 2158, 1259, 8, 1795, 9, 580, 1495]
9.)  with the competitiveness of the television market these days everyone is looking for a way t
o get viewers more excited
  [22, 0, 9643, 4, 0, 761, 47, 144, 171, 1376, 13, 735, 11, 6, 229, 5, 188, 3917, 45, 9684]
10.)  one of the leaders behind the expanded use of N numbers is call interactive a joint venture
of giants american express co. and american telephone & telegraph co
  [54, 4, 0, 815, 1116, 0, 2439, 269, 4, 3, 1619, 13, 786, 9958, 6, 795, 818, 4, 2172, 140, 1021,
95, 8, 140, 732, 82, 3133, 570]
```

b.) Create an RNN model following the tutorial.

Using the ptb_word_lm source code, I added the accuracy and loss measure in the model.

Train the model and save the training and validation loss over time

Hyper Prameters: *MediumConfig as per the code*

```
  init_scale = 0.05
  learning_rate = 1.0
  max_grad_norm = 5
  num_layers = 2
  num_steps = 35
```

```
hidden_size = 650
max_epoch = 6
max_max_epoch = 30
keep_prob = 0.5
lr_decay = 0.8
batch_size = 20
vocab_size = 10000
```

Model:

The core of the model remains as provided in the RNN PTB tutorial

The changes made are in the accuracy and prediction functions

```
# Probabilities
self.probas = tf.nn.softmax(logits, name="probas")
# Update the cost
self._cost = tf.reduce_sum(loss)
self._final_state = state
# get the prediction accuracy
softmax_out = tf.nn.softmax(tf.reshape(logits, [-1, vocab_size]))
self.predict = tf.cast(tf.argmax(softmax_out, axis=1), tf.int32)
correct_prediction = tf.equal(self.predict, tf.reshape(self._input.targets, [-1]))
self._accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
```

This allows us to capture the probabilities of the words. With the help of a arg max we can even predict the word with the highest probability and compare it against the input targets. This will help compute the accuracy for the case.

Losses

Epoch: 30, Train Loss: 3.828, Perplexity: 45.979

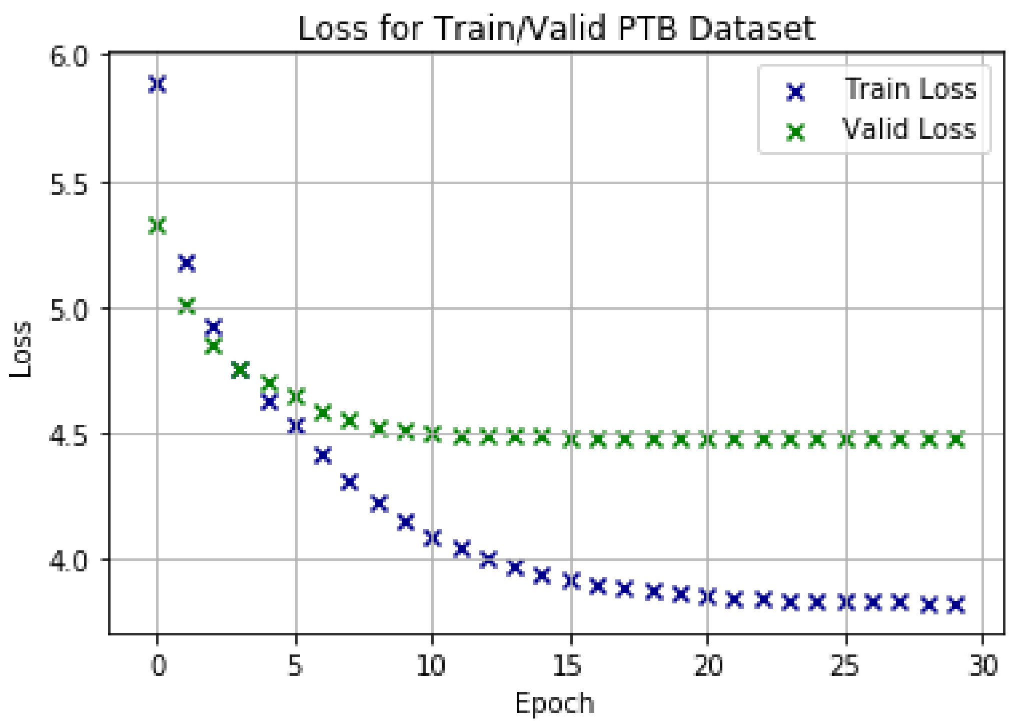
Epoch: 30, Valid Loss: 4.478, Perplexity: 88.057

Test Loss: 4.432, Perplexity: 84.058

Training Loss: [`5.892828233455839, 5.175063474197051, 4.923602828133515, 4.755605833686223, 4.6317642236554155, 4.5337887175386795, 4.4146086716731485, 4.309408576892233, 4.224679030739641, 4.15113995112956, 4.091310661546713, 4.042695668270108, 4.002090388723179, 3.9691814252180104, 3.9422283922076957, 3.9196188754318784, 3.8993738749990015, 3.8886861031760516, 3.8726353648918055, 3.8653755862305528, 3.855408901839226, 3.8485901848083293, 3.8452557702895604, 3.8390576082842127, 3.8360201804934873, 3.834563913631778, 3.8339368299668437, 3.830275785865254, 3.8287308164710105, 3.828182678846872`]

Validation Loss: [`5.324787410813935, 5.01346401084848, 4.84561486069037, 4.75106652084662, 4.6982585424306444, 4.652949011147427, 4.586399453091784, 4.5520516988533695, 4.523064533674798, 4.5084592110770085, 4.497108386811756, 4.493018537248884, 4.489556033595079, 4.485706998863999, 4.48800402167703, 4.485253441220238, 4.485081376056282, 4.482010697345344, 4.483444072697439, 4.482078861573926, 4.482794359687234, 4.482575554880155, 4.4811470374282525, 4.480794287441539,`

```
4.480578127491231, 4.479583918772587, 4.4784935194781035, 4.478589228415975, 4.478181957841731,
4.477987438383557]
```



Observation:

We can clearly see that with the medium configuration the validation loss stops reducing and it can be inferred that the model may be overfitting to the training data, around the 5th EPOCH.

c.) *Results on test Set*

Test Loss: 4.432, Perplexity: 84.058

Also, saved model network to “summary/” Folder.

The test result appears to be quite good and closely approximates the observed validation loss rather than the training loss.

d.) Generate sentence from words randomly selected out of vocabulary (using the trained model)

1.) Pick from Vocabulary: buying

```
buying thousands to shares shares shares shares shares shares stock shares shares shares shares
shares gold shares at shares shares stock shares shares shares stock dollars shares shares shares
shares shares shares shares shares stocks shares shares stock shares shares shares stock stocks
stocks shares shares vehicle.
```

2.) Pick from Vocabulary: very

```
very good <unk> good popular dramatic meaningful well well very horrible large well <unk> <unk>
very good concerned well well well well good meaningful <unk> concerned one well well steady
costly well good well well <unk> impressive <unk> interesting shares awful attractive well well
well well profitable well stable large well well well attractive good very well well costly proof
well attractive getting well well well <unk> well well good bullish well precisely well well well
<unk> wrong confident alarmed well well well well cheap concerned good worrying well well soon
bullish significant bleak well well well well bullish well well.
```

3.) Pick from Vocabulary: men

```
men love who says and and.
```

4.) Pick from Vocabulary: bills

```
bills snapped at and that.
```

5.) Pick from Vocabulary: three

```
three canadian years weeks major years games years soldiers weeks billion years years miles
months dollars years games seasons seats dollars games four one years times games feet games
weeks units weeks games dollars miles women years games dollars years million weeks years
thousand games games miles thousand tons to or games inches miles dollars games from dollars or
games weeks feet feet.
```

e.) Generate sentences from Words = {north, wall, truth, california, health}

1.) Provided Word: north

```
north korean carolina america carolina america western and countries calif. texas region texas
and texas california texas and sea texas africa and texas western west sea western and and and
imports texas and and and forest texas africa bronx and and fla. a south <unk> and carolina
america and to and from stockholm carolina southwest texas carolina america puerto african south
carolina africa drinks singapore texas texas pennsylvania and oklahoma lumpur mexico.
```

2.) Provided Word: wall

```
wall street street street street street street street street might street street street street
street street street street street street street street street street street street street street
street street street street street street street street street street street street street street
street 's street street is street street street street street street street street street street
street street street street street street is street street street street street street street
street street street street street street street street street street street street street street
street street street does street street street street street street street street street street
street street.
```

3.) Provided Word: truth

```
truth.
```

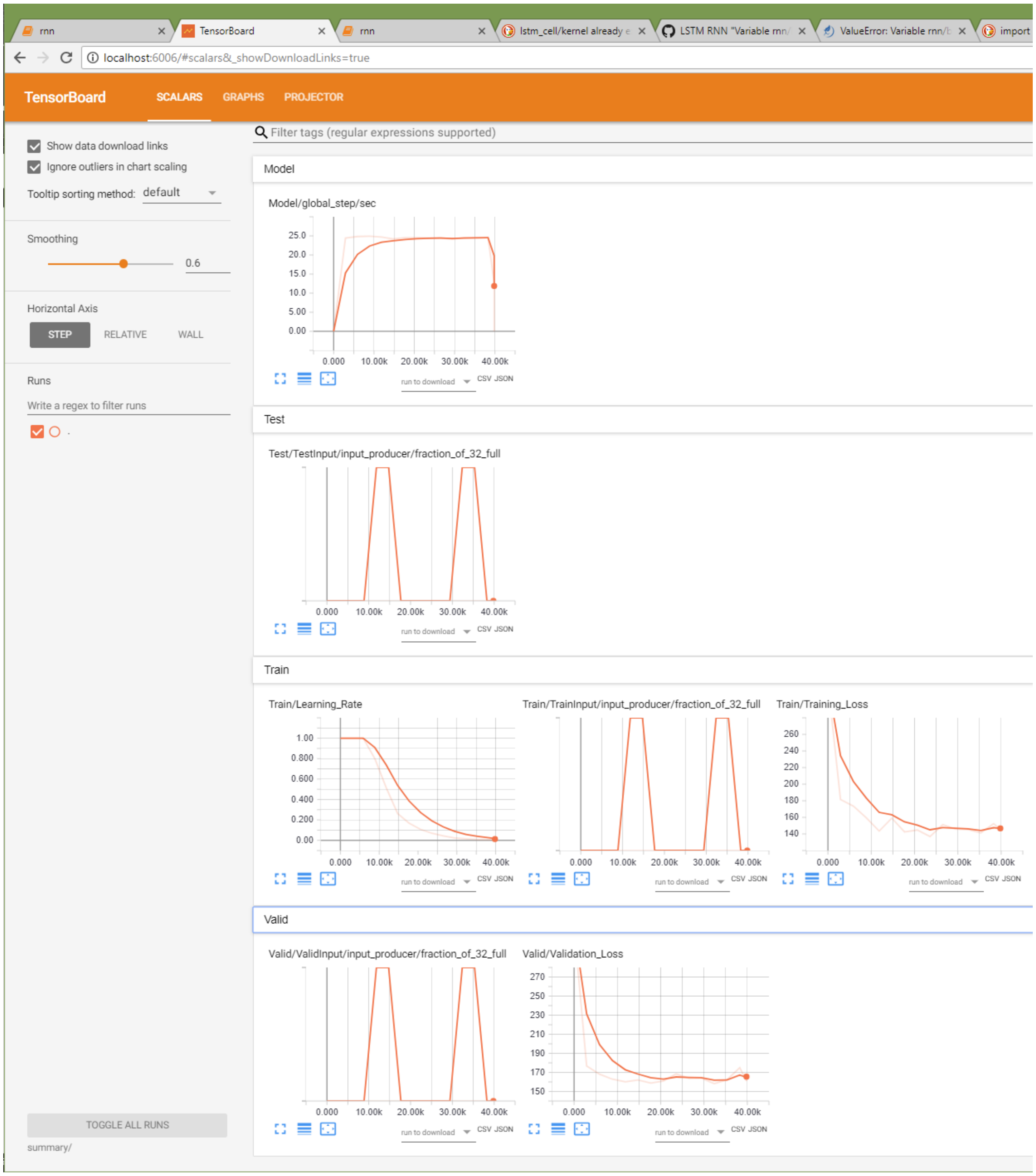
4.) Provided Word: california

```
california failed but and free and wyoming in and florida pennsylvania and wisconsin pennsylvania
and wisconsin and florida.
```

5.) Provided Word: health

```
health and <unk> care care and care and care care and workers and.
```

Overall Summary of the runs through Tensorboard


localhost:6006/#scalars&_showDownloadLinks=true

TensorBoard SCALARS GRAPHS PROJECTOR

Filter tags (regular expressions supported)

☑ Show data download links

☑ Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing 0.6

Horizontal Axis: [✓ STEP] RELATIVE WALL

Runs

Write a regex to filter runs

☑ ○ .

TOGGLE ALL RUNS

summary/