HW 8 LSTM's and GRU's
Dustan Kraus

Below are my samples generated after training on Alma:

Alma with 135-150 Training Steps with the GRU

And throughout the land, which will the remainder of the Lamanit
And and hand, his name and ye have believed according to my men
And that it was one thou hast sept, being buried by the Nephites
And thus murrey left the word of God unto Korihor in other words
And that much tabe unto God. Now Aaron said unto them: If ye h
And that Moroni preached the word of God unto the king and by Al
And that we know, in all the order of the thing canly forth, and
And thus my body. Yea, after has seed again with the scripture
And this is fear sobjost to be have cometh in the supplies of Le
And as they were in the Nephites and their brethren. Now Aaron
And this is man that prupposs of the Lamanites had behold no to
And that they had raisen have been slain sin? How I do trady tha
And that there were many that did declare unto them that they we
And that likting were brought to believe in idherand things; nev
And that much as he did not suppose that thou a learness that li

Alma with 170-185 Training Steps with the LSTM

And gof hips, more women immediately afflicted, come, by the swo
And might to this digce in sea; and we was mine to them, yea, my
And mim. The word of the south of God aljormanites, as I know th
And misking faith and on took all your should come into the peop
And mondifuse and men for your synagogues in heed to ender to th
And yourswardy with a burdement-seat, and begin them; there way
And mivion, son--Siniquite--aft that ye supposing, even for it s
And mitt of gold of Christ and thereword he went in the church,
And Jisties on the salvay, to seed our land of God, expert to th
And wistiness and the dinlessale again against the wilderness; o
And amoot became Ammon labors pouring of the Lamanites were not
And ming of the truth of your beginneth with their brethren; the
And mood some should take to encirsell upon you in the could she
And it with I bound also these disting contention, reast took th
And mightion towards them after the Nephites, or were a judges o

Below are samples generated after training on a dataset of spam emails and random text messages:

Spam/text messages after 2400-2415 Training Steps with the GRU

2400
hey getling...
URGENT! Your Poorear!s with DOT Meris Dear End Li
2401
hey are babe, More watcall meeting just when you went to wal...
2402
hey because then.:-) naw diff-is 75505
Great your whis year swie
2403
hey haven't mayb up what we should beer
Cos I'm driving, But FRE
2404
hey are watching autule peay. Reet Brolds to get the frouth)goin
2405
hey are busy thing.
Dont telling for you.
Hey you know about end
2406
hey go. It's osbide by some feb 16
Dear, there aight?
* Wan a st
2407
hey are swortching againclay way. Have a beouf like.
I don't kno
2408
hey before ya yruj.....!! Blick time that chlese upton chat
2409
hey to home take this
Jast in much
Dunn to.
Love \goo.Wettou ats
2410
hey test? Everything of mine. Ok lor... Haha, i have tht dearing
2411
hey good perfwondevery?I was specsing cost but my Lion or 40se w
2412
hey together.
I'm gonna be has been cancelf and muz u really rep
2413

hey to the house I was going of any for the new on cause.
AnG GU
2414
hey becting your passely rings about 20 wors i ask to, what woul
2415
hey to me and remind by a girls
Ok left Doifent still in a shoul

Spam/text messages with 2400-2415 Training Steps with the LSTM
2400
hey p]]2B> mehwah. You drive dat is afternoon, call will be in p
2401
hey p]]2B> olveg 'lpres meana list* Saratie Teath. Call ning!
Ok
2402
hey m]
)
Simp. What.
Dimting lauxtr..ister all this weekend.
Are
2403
hey p]]2P> thi dearr
Dar brash you go to wall, e egt
i have sure
2404
hey p]]2P>>n
Take he has grop kien..main call back? wife I can't
2405
hey p]]2P>>na nuhill vrical as The numbling clist with I miss
Jo
2406
hey Zoong/sww.lites & Rangtone txt PLAS a feet menciful! Call Mo
2407
hey p]]2P>> th melt my dad i breaw. You mitue has a up wen I'm g
2408
hey p]]
U our looks adrice.. &amp
Well you home ur lets it dive
2409
hey p]] u] claild on ur call meak
If Thine. 4 hore all you? Are
2410
hey p]]2B> myrafe!

I need you tomorrow in like da.
Irm and thb b
2411
hey portesen lahoo three.I call sile 2 set ip afay wish. Place h
2412
hey porrob and places..U can you de preach if u wif extre eit ah
2413
hey p]]2
A veayV plase GOT rellax wiss the word Eren- innered yo
2414
hey p]]S
Ur_DA. Year Ur jus to important copp coutt call 0906538
2415
hey p]]ger tend cart is creding, Sz dincers
Good night well?
Dev

Here is my Lab8 scaffold code:

```
import tensorflow as tf
import numpy as np
from textloader import TextLoader
from my_gru_class import mygru


#
# ---------------------------------------
#
# Global variables

batch_size = 50
# batch_size = 2
sequence_length = 50

data_loader = TextLoader( ".", batch_size, sequence_length )

vocab_size = data_loader.vocab_size  # dimension of one-hot encodings
state_dim = 128

num_layers = 2

tf.reset_default_graph()
```

```
#
# ================================================================
# ================================================================
# ================================================================
#

# define placeholders for our inputs.
# in_ph is assumed to be [batch_size,sequence_length]
# targ_ph is assumed to be [batch_size,sequence_length]

in_ph = tf.placeholder( tf.int32, [ batch_size, sequence_length ], name='inputs' )
targ_ph = tf.placeholder( tf.int32, [ batch_size, sequence_length ], name='targets' )
in_onehot = tf.one_hot( in_ph, vocab_size, name="input_onehot" )

inputs = tf.split( in_onehot, sequence_length, axis=1 )
inputs = [ tf.squeeze(input_, [1]) for input_ in inputs ]
targets = tf.split( targ_ph, sequence_length, axis=1 )

# at this point, inputs is a list of length sequence_length
# each element of inputs is [batch_size,vocab_size]

# targets is a list of length sequence_length
# each element of targets is a 1D vector of length batch_size

#
#################################################################################
#########################
# YOUR COMPUTATION GRAPH HERE

# create a BasicLSTMCell
#   use it to create a MultiRNNCell
#   use it to create an initial_state
#     note that initial_state will be a *list* of tensors!
cells = []
for i in xrange(num_layers):
   cells.append(tf.nn.rnn_cell.BasicLSTMCell(state_dim))
   # cells.append(mygru(state_dim))
rnn = tf.nn.rnn_cell.MultiRNNCell(cells)
initial_state = rnn.zero_state(batch_size, tf.float32)

# call seq2seq.rnn_decoder
outputs, final_state = tf.contrib.legacy_seq2seq.rnn_decoder(inputs, initial_state, rnn)
```

```
# transform the list of state outputs to a list of logits.
# use a linear transformation.
W = tf.Variable(tf.random_normal([state_dim, vocab_size], stddev=0.02))
b = tf.Variable(tf.random_normal([vocab_size], stddev=0.01))
logits = [tf.matmul(x,W) + [b]*batch_size for x in outputs]

# call seq2seq.sequence_loss
loss_weights = [1.0 for i in xrange(sequence_length)]
loss = tf.contrib.legacy_seq2seq.sequence_loss(logits, targets, loss_weights)

# create a training op using the Adam optimizer
optim = tf.train.AdamOptimizer().minimize(loss)

# ------------------
# YOUR SAMPLER GRAPH HERE

# place your sampler graph here it will look a lot like your
# computation graph, except with a "batch_size" of 1.
s_batch_size = 1
s_inputs = tf.placeholder( tf.int32, [s_batch_size], name='s_inputs')
s_in_onehot = tf.one_hot( s_inputs, vocab_size, name='s_input_onehot')

s_input = tf.split(s_in_onehot, 1)

s_initial_state = rnn.zero_state(1,tf.float32)

s_outputs, s_final_state = tf.contrib.legacy_seq2seq.rnn_decoder(s_input, s_initial_state, rnn)

s_probs = tf.nn.softmax([tf.matmul(x,W) + [b]*1 for x in s_outputs])
# remember, we want to reuse the parameters of the cell and whatever
# parameters you used to transform state outputs to logits!

#
# ==============================================================
# ==============================================================
# ==============================================================
#
################################################################################
#########################

def sample( num=200, prime='ab' ):

    # prime the pump
```

```
# generate an initial state. this will be a list of states, one for
# each layer in the multicell.
s_state = sess.run( s_initial_state )

# for each character, feed it into the sampler graph and
# update the state.
for char in prime[:-1]:
    x = np.ravel( data_loader.vocab[char] ).astype('int32')
    feed = { s_inputs:x }
    for i, s in enumerate( s_initial_state ):
        feed[s] = s_state[i]
    s_state = sess.run( s_final_state, feed_dict=feed )

# now we have a primed state vector; we need to start sampling.
ret = prime
char = prime[-1]
for n in range(num):
    x = np.ravel( data_loader.vocab[char] ).astype('int32')

    # plug the most recent character in...
    feed = { s_inputs:x }
    for i, s in enumerate( s_initial_state ):
        feed[s] = s_state[i]
    ops = [s_probs]
    ops.extend( list(s_final_state) )

    retval = sess.run( ops, feed_dict=feed )

    s_probsv = retval[0]
    s_state = retval[1:]

    # ...and get a vector of probabilities out!

    # now sample (or pick the argmax)
    # sample = np.argmax( s_probsv[0] )
    sample = np.random.choice( vocab_size, p=s_probsv[0][0] )

    pred = data_loader.chars[sample]
    ret += pred
    char = pred

return ret
```

```
#
# ====================================================================
# ====================================================================
# ====================================================================
#

sess = tf.Session()
sess.run( tf.global_variables_initializer() )
summary_writer = tf.summary.FileWriter( "./tf_logs", graph=sess.graph )

lts = []

print "FOUND %d BATCHES" % data_loader.num_batches

for j in range(10000):

    state = sess.run( initial_state )
    data_loader.reset_batch_pointer()

    for i in range( data_loader.num_batches ):

        x,y = data_loader.next_batch()

        # we have to feed in the individual states of the MultiRNN cell
        feed = { in_ph: x, targ_ph: y }
        for k, s in enumerate( initial_state ):
            feed[s] = state[k]

        ops = [optim,loss]
        ops.extend( list(final_state) )

        # retval will have at least 3 entries:
        # 0 is None (triggered by the optim op)
        # 1 is the loss
        # 2+ are the new final states of the MultiRNN cell
        retval = sess.run( ops, feed_dict=feed )

        lt = retval[1]
        state = retval[2:]

        if i%1000==0:
            print "%d %d\t%.4f" % ( j, i, lt )
```

```
            lts.append( lt )

    # print sample( num=60, prime="And " )
    print sample( num = 60, prime='hey ')
    # print sample( num=60, prime="The ")
    # print sample( num=60, prime="abababab" )
    # print sample( num=60, prime="foo ba" )
    # print sample( num=60, prime="abcdab" )

summary_writer.close()

#
# =================================================================
# =================================================================
# =================================================================
#

#import matplotlib
#import matplotlib.pyplot as plt
#plt.plot( lts )
#plt.show()
```

Here is my GRU code:

```
from tensorflow.python.ops.rnn_cell import RNNCell
import tensorflow as tf

class mygru( RNNCell ):
    def __init__( self, state_dim):
        self.state_dim = state_dim
        self.scope = None


    @property
    def state_size(self):
        return self.state_dim

    @property
    def output_size(self):
        return self.state_dim

    def __call__( self, inputs, state):
        input_shape = inputs.get_shape().as_list()
```

```
        with tf.variable_scope('gru') as scope:
            if self.scope == None:
                wx_shape = [input_shape[1], self.state_dim]
                wh_shape = [self.state_dim, self.state_dim]
                b_shape = [self.state_dim]
                self.Wxr = tf.get_variable('wxr', shape = wx_shape, initializer =
tf.contrib.layers.variance_scaling_initializer())
                self.Wxz = tf.get_variable('wxz', shape = wx_shape, initializer =
tf.contrib.layers.variance_scaling_initializer())
                self.Wxh = tf.get_variable('wxh', shape = wx_shape, initializer =
tf.contrib.layers.variance_scaling_initializer())
                self.Whr = tf.get_variable('whr', shape = wh_shape, initializer =
tf.contrib.layers.variance_scaling_initializer())
                self.Whz = tf.get_variable('whz', shape = wh_shape, initializer =
tf.contrib.layers.variance_scaling_initializer())
                self.Whh = tf.get_variable('whh', shape = wh_shape, initializer =
tf.contrib.layers.variance_scaling_initializer())
                self.br = tf.get_variable('br', shape = b_shape, initializer =
tf.contrib.layers.variance_scaling_initializer())
                self.bz = tf.get_variable('bz', shape = b_shape, initializer =
tf.contrib.layers.variance_scaling_initializer())
                self.bh = tf.get_variable('bh', shape = b_shape, initializer =
tf.contrib.layers.variance_scaling_initializer())
                self.scope = 'gru'
            else:
                scope.reuse_variables()
                self.Wxr = tf.get_variable('wxr')
                self.Wxz = tf.get_variable('wxz')
                self.Wxh = tf.get_variable('wxh')
                self.Whr = tf.get_variable('whr')
                self.Whz = tf.get_variable('whz')
                self.Whh = tf.get_variable('whh')
                self.br = tf.get_variable('br')
                self.bz = tf.get_variable('bz')
                self.bh = tf.get_variable('bh')
            r = tf.nn.sigmoid(tf.matmul(inputs, self.Wxr) + tf.matmul(state, self.Whr) + self.br)
            z = tf.nn.sigmoid(tf.matmul(inputs, self.Wxz) + tf.matmul(state, self.Whz) + self.bz)
            htild = tf.nn.tanh(tf.matmul(inputs, self.Wxh) + tf.matmul(tf.multiply(r,state), self.Whh) +
self.bh)
            h = tf.multiply(z,state) + tf.multiply((1-z), htild)

        return h,h
```