lstm_with_unique_14

December 12, 2018

```
In [1]: # LSTM WITH UNIQUE DATASET IMPLEMENTATION 14
        # Depression Analysis in Bangla with LSTM-RNN
        # copyright (c) ABDUL HASIB UDDIN <abdulhasibuddin@gmail.com>
        # LICENSE: GNU General Public License v3.0
        # Courtesy: https://github.com/mchablani/deep-learning/blob/master/sentiment-rnn/Senti
```

```
In [0]: import numpy as np
        import tensorflow as tf
        from timeit import default_timer as timer
        from collections import Counter
        from string import punctuation
        from google.colab import files
```

```
In [0]: # Build the graph::

        lstm_size = 128
        lstm_layers = 5
        batch_size = 5
        learning_rate = 0.0001
        epochs = 10
```

```
In [4]: fileName = "lstm_with_unique_14"
        checkpointName = "checkpoints/"+fileName+".ckpt"
        print(checkpointName)
        print(type(checkpointName))
```

```
checkpoints/lstm_with_unique_14.ckpt
<class 'str'>
```

```
In [5]: files.upload()
        files.upload()

        with open('data_all_unique_dnd_stratified_text.txt', 'r', encoding="utf8") as f:
            tweets = f.read()
        with open('data_all_unique_dnd_stratified_labels.txt', 'r', encoding="utf8") as f:
            labels_org = f.read()
```

```
<IPython.core.display.HTML object>
```

```
Saving data_all_unique_dnd_stratified_text.txt to data_all_unique_dnd_stratified_text.txt
```

```
<IPython.core.display.HTML object>
```

```
Saving data_all_unique_dnd_stratified_labels.txt to data_all_unique_dnd_stratified_labels.txt
```

```
In [0]: # Data preprocessing::
        #all_text = ''.join([c for c in tweets if c not in punctuation])
        all_text = ''.join([c for c in tweets])
        tweets = all_text.split('\n')

        all_text = ' '.join(tweets)
        words = all_text.split()
```

```
In [0]: counts = Counter(words)
        vocab = sorted(counts, key=counts.get, reverse=True)
        vocab_to_int = {word: ii for ii, word in enumerate(vocab, 1)}

        tweets_ints = []
        for each in tweets:
            tweets_ints.append([vocab_to_int[word] for word in each.split()])
```

```
In [8]: # Encoding the labels::
        list_labels = []

        for l in labels_org.split():
            if l == "depressive":
                list_labels.append(1)
            else:
                list_labels.append(0)

        labels = np.array(list_labels)
        print(len(labels))
```

```
1176
```

```
In [9]: tweets_lens = Counter([len(x) for x in tweets_ints])
        print("Zero-length tweets: {}".format(tweets_lens[0]))
        print("Maximum tweets length: {}".format(max(tweets_lens)))
```

```
Zero-length tweets: 1
Maximum tweets length: 63
```

```
In [0]: # Filter out that tweets with 0 length
        tweets_ints = [r[0:200] for r in tweets_ints if len(r) > 0]

In [11]: from collections import Counter
         tweets_lens = Counter([len(x) for x in tweets_ints])
         print("Zero-length tweets: {}".format(tweets_lens[0]))
         print("Maximum tweet length: {}".format(max(tweets_lens)))

Zero-length tweets: 0
Maximum tweet length: 63


In [0]: seq_len = 200
        features = np.zeros((len(tweets_ints), seq_len), dtype=int)
        # print(features[:10,:100])
        for i, row in enumerate(tweets_ints):
            features[i, -len(row):] = np.array(row)[:seq_len]
        #features[:10,:100]

In [13]: split_frac = 0.8

         split_index = int(split_frac * len(features))

         train_x, val_x = features[:split_index], features[split_index:]
         train_y, val_y = labels[:split_index], labels[split_index:]

         split_frac = 0.5
         split_index = int(split_frac * len(val_x))

         val_x, test_x = val_x[:split_index], val_x[split_index:]
         val_y, test_y = val_y[:split_index], val_y[split_index:]

         print("\t\t\tFeature Shapes:")
         print("Train set: \t\t{}".format(train_x.shape),
               "\nValidation set: \t{}".format(val_x.shape),
               "\nTest set: \t\t{}".format(test_x.shape))
         print("label set: \t\t{}".format(train_y.shape),
               "\nValidation label set: \t{}".format(val_y.shape),
               "\nTest label set: \t\t{}".format(test_y.shape))

                        Feature Shapes:
Train set:                 (940, 200)
Validation set:         (118, 200)
Test set:                 (118, 200)
label set:                 (940,)
Validation label set:         (118,)
Test label set:                 (118,)
```

```
In [0]: n_words = len(vocab_to_int) + 1 # Add 1 for 0 added to vocab

        # Create the graph object
        tf.reset_default_graph()
        with tf.name_scope('inputs'):
            inputs_ = tf.placeholder(tf.int32, [None, None], name="inputs")
            labels_ = tf.placeholder(tf.int32, [None, None], name="labels")
            keep_prob = tf.placeholder(tf.float32, name="keep_prob")
```

```
In [0]: # Size of the embedding vectors (number of units in the embedding layer)
        embed_size = 300

        with tf.name_scope("Embeddings"):
            embedding = tf.Variable(tf.random_uniform((n_words, embed_size), -1, 1))
            embed = tf.nn.embedding_lookup(embedding, inputs_)
```

```
In [16]: def lstm_cell():
             # Your basic LSTM cell
             lstm = tf.contrib.rnn.BasicLSTMCell(lstm_size, reuse=tf.get_variable_scope().reuse
             # Add dropout to the cell
             return tf.contrib.rnn.DropoutWrapper(lstm, output_keep_prob=keep_prob)

         with tf.name_scope("RNN_layers"):
             # Stack up multiple LSTM layers, for deep learning
             cell = tf.contrib.rnn.MultiRNNCell([lstm_cell() for _ in range(lstm_layers)])

             # Getting an initial state of all zeros
             initial_state = cell.zero_state(batch_size, tf.float32)
```

```
WARNING:tensorflow:From <ipython-input-16-678741cf60df>:3: BasicLSTMCell.__init__ (from tensor
Instructions for updating:
This class is deprecated, please use tf.nn.rnn_cell.LSTMCell, which supports all the feature t
```

```
In [0]: with tf.name_scope("RNN_forward"):
            outputs, final_state = tf.nn.dynamic_rnn(cell, embed, initial_state=initial_state)
```

```
In [0]: # Output::

        with tf.name_scope('predictions'):
            predictions = tf.contrib.layers.fully_connected(outputs[:, -1], 1, activation_fn=t
            tf.summary.histogram('predictions', predictions)
        with tf.name_scope('cost'):
            cost = tf.losses.mean_squared_error(labels_, predictions)
            tf.summary.scalar('cost', cost)

        with tf.name_scope('train'):
            optimizer = tf.train.AdamOptimizer(learning_rate).minimize(cost)

        merged = tf.summary.merge_all()
```

```
In [0]: # Validation accuracy::

        with tf.name_scope('validation'):
            correct_pred = tf.equal(tf.cast(tf.round(predictions), tf.int32), labels_)
            accuracy = tf.reduce_mean(tf.cast(correct_pred, tf.float32))
```

```
In [0]: # Batching::

        def get_batches(x, y, batch_size=100):

            n_batches = len(x)//batch_size
            x, y = x[:n_batches*batch_size], y[:n_batches*batch_size]
            for ii in range(0, len(x), batch_size):
                yield x[ii:ii+batch_size], y[ii:ii+batch_size]
```

```
In [21]: # Training::

         #epochs = 5
         saver = tf.train.Saver()
         start = timer()

         with tf.Session() as sess:
             sess.run(tf.global_variables_initializer())
             train_writer = tf.summary.FileWriter('./logs/tb/train', sess.graph)
             test_writer = tf.summary.FileWriter('./logs/tb/test', sess.graph)
             iteration = 1
             for e in range(1, epochs+1):
                 state = sess.run(initial_state)

                 for ii, (x, y) in enumerate(get_batches(train_x, train_y, batch_size), 1):
                     feed = {inputs_: x,
                             labels_: y[:, None],
                             keep_prob: 1,
                             initial_state: state}
                     summary, loss, state, _ = sess.run([merged, cost, final_state, optimizer]
         #             loss, state, _ = sess.run([cost, final_state, optimizer], feed_dict=fee

                     train_writer.add_summary(summary, iteration)

                     if iteration%5==0:
                         print("Epoch: {}/{}".format(e, epochs),
                               "Iteration: {}".format(iteration),
                               "Train loss: {:.3f}".format(loss))

                     if iteration%25==0:
                         val_acc = []
                         val_state = sess.run(cell.zero_state(batch_size, tf.float32))
                         for x, y in get_batches(val_x, val_y, batch_size):
                             feed = {inputs_: x,
                                     labels_: y[:, None],
                                     keep_prob: 1,
                                     initial_state: val_state}
         #                             batch_acc, val_state = sess.run([accuracy, final_state], feed_d
                             summary, batch_acc, val_state = sess.run([merged, accuracy, final_
                             val_acc.append(batch_acc)
                         print("Val acc: {:.3f}".format(np.mean(val_acc)))
                     iteration +=1
                     test_writer.add_summary(summary, iteration)
                     saver.save(sess, checkpointName)
         #                tensorboard = TensorBoard(log_dir="logs/tweet_5000_all_sentiments_six_cl
             saver.save(sess, checkpointName)

         duration = timer() - start
         print('Time elasped =',duration,'sec(s)')
```

```
Epoch: 1/10 Iteration: 5 Train loss: 0.262
Epoch: 1/10 Iteration: 10 Train loss: 0.254
Epoch: 1/10 Iteration: 15 Train loss: 0.260
Epoch: 1/10 Iteration: 20 Train loss: 0.251
Epoch: 1/10 Iteration: 25 Train loss: 0.244
Val acc: 0.357
Epoch: 1/10 Iteration: 30 Train loss: 0.244
Epoch: 1/10 Iteration: 35 Train loss: 0.240
Epoch: 1/10 Iteration: 40 Train loss: 0.239
Epoch: 1/10 Iteration: 45 Train loss: 0.261
Epoch: 1/10 Iteration: 50 Train loss: 0.198
Val acc: 0.426
Epoch: 1/10 Iteration: 55 Train loss: 0.259
Epoch: 1/10 Iteration: 60 Train loss: 0.232
Epoch: 1/10 Iteration: 65 Train loss: 0.226
Epoch: 1/10 Iteration: 70 Train loss: 0.238
Epoch: 1/10 Iteration: 75 Train loss: 0.215
Val acc: 0.365
Epoch: 1/10 Iteration: 80 Train loss: 0.252
Epoch: 1/10 Iteration: 85 Train loss: 0.243
Epoch: 1/10 Iteration: 90 Train loss: 0.199
Epoch: 1/10 Iteration: 95 Train loss: 0.275
Epoch: 1/10 Iteration: 100 Train loss: 0.253
Val acc: 0.357
Epoch: 1/10 Iteration: 105 Train loss: 0.207
Epoch: 1/10 Iteration: 110 Train loss: 0.271
Epoch: 1/10 Iteration: 115 Train loss: 0.218
Epoch: 1/10 Iteration: 120 Train loss: 0.215
Epoch: 1/10 Iteration: 125 Train loss: 0.181
Val acc: 0.391
Epoch: 1/10 Iteration: 130 Train loss: 0.206
```

```
Epoch: 1/10 Iteration: 135 Train loss: 0.287
Epoch: 1/10 Iteration: 140 Train loss: 0.265
Epoch: 1/10 Iteration: 145 Train loss: 0.251
Epoch: 1/10 Iteration: 150 Train loss: 0.272
Val acc: 0.270
Epoch: 1/10 Iteration: 155 Train loss: 0.230
Epoch: 1/10 Iteration: 160 Train loss: 0.296
Epoch: 1/10 Iteration: 165 Train loss: 0.238
Epoch: 1/10 Iteration: 170 Train loss: 0.261
Epoch: 1/10 Iteration: 175 Train loss: 0.205
Val acc: 0.539
Epoch: 1/10 Iteration: 180 Train loss: 0.246
Epoch: 1/10 Iteration: 185 Train loss: 0.241
Epoch: 2/10 Iteration: 190 Train loss: 0.252
Epoch: 2/10 Iteration: 195 Train loss: 0.264
Epoch: 2/10 Iteration: 200 Train loss: 0.235
Val acc: 0.496
Epoch: 2/10 Iteration: 205 Train loss: 0.265
Epoch: 2/10 Iteration: 210 Train loss: 0.256
Epoch: 2/10 Iteration: 215 Train loss: 0.253
Epoch: 2/10 Iteration: 220 Train loss: 0.252
Epoch: 2/10 Iteration: 225 Train loss: 0.253
Val acc: 0.478
Epoch: 2/10 Iteration: 230 Train loss: 0.235
Epoch: 2/10 Iteration: 235 Train loss: 0.264
Epoch: 2/10 Iteration: 240 Train loss: 0.235
Epoch: 2/10 Iteration: 245 Train loss: 0.253
Epoch: 2/10 Iteration: 250 Train loss: 0.237
Val acc: 0.443
Epoch: 2/10 Iteration: 255 Train loss: 0.248
Epoch: 2/10 Iteration: 260 Train loss: 0.246
Epoch: 2/10 Iteration: 265 Train loss: 0.270
Epoch: 2/10 Iteration: 270 Train loss: 0.242
Epoch: 2/10 Iteration: 275 Train loss: 0.227
Val acc: 0.409
Epoch: 2/10 Iteration: 280 Train loss: 0.249
Epoch: 2/10 Iteration: 285 Train loss: 0.224
Epoch: 2/10 Iteration: 290 Train loss: 0.223
Epoch: 2/10 Iteration: 295 Train loss: 0.249
Epoch: 2/10 Iteration: 300 Train loss: 0.257
Val acc: 0.443
Epoch: 2/10 Iteration: 305 Train loss: 0.256
Epoch: 2/10 Iteration: 310 Train loss: 0.190
Epoch: 2/10 Iteration: 315 Train loss: 0.301
Epoch: 2/10 Iteration: 320 Train loss: 0.196
Epoch: 2/10 Iteration: 325 Train loss: 0.223
Val acc: 0.374
Epoch: 2/10 Iteration: 330 Train loss: 0.298
```

```
Epoch: 2/10 Iteration: 335 Train loss: 0.248
Epoch: 2/10 Iteration: 340 Train loss: 0.257
Epoch: 2/10 Iteration: 345 Train loss: 0.278
Epoch: 2/10 Iteration: 350 Train loss: 0.265
Val acc: 0.504
Epoch: 2/10 Iteration: 355 Train loss: 0.226
Epoch: 2/10 Iteration: 360 Train loss: 0.245
Epoch: 2/10 Iteration: 365 Train loss: 0.227
Epoch: 2/10 Iteration: 370 Train loss: 0.263
Epoch: 2/10 Iteration: 375 Train loss: 0.234
Val acc: 0.643
Epoch: 3/10 Iteration: 380 Train loss: 0.294
Epoch: 3/10 Iteration: 385 Train loss: 0.274
Epoch: 3/10 Iteration: 390 Train loss: 0.234
Epoch: 3/10 Iteration: 395 Train loss: 0.241
Epoch: 3/10 Iteration: 400 Train loss: 0.231
Val acc: 0.713
Epoch: 3/10 Iteration: 405 Train loss: 0.246
Epoch: 3/10 Iteration: 410 Train loss: 0.259
Epoch: 3/10 Iteration: 415 Train loss: 0.253
Epoch: 3/10 Iteration: 420 Train loss: 0.248
Epoch: 3/10 Iteration: 425 Train loss: 0.268
Val acc: 0.461
Epoch: 3/10 Iteration: 430 Train loss: 0.247
Epoch: 3/10 Iteration: 435 Train loss: 0.236
Epoch: 3/10 Iteration: 440 Train loss: 0.231
Epoch: 3/10 Iteration: 445 Train loss: 0.243
Epoch: 3/10 Iteration: 450 Train loss: 0.250
Val acc: 0.417
Epoch: 3/10 Iteration: 455 Train loss: 0.222
Epoch: 3/10 Iteration: 460 Train loss: 0.225
Epoch: 3/10 Iteration: 465 Train loss: 0.250
Epoch: 3/10 Iteration: 470 Train loss: 0.239
Epoch: 3/10 Iteration: 475 Train loss: 0.235
Val acc: 0.383
Epoch: 3/10 Iteration: 480 Train loss: 0.219
Epoch: 3/10 Iteration: 485 Train loss: 0.244
Epoch: 3/10 Iteration: 490 Train loss: 0.227
Epoch: 3/10 Iteration: 495 Train loss: 0.177
Epoch: 3/10 Iteration: 500 Train loss: 0.244
Val acc: 0.426
Epoch: 3/10 Iteration: 505 Train loss: 0.308
Epoch: 3/10 Iteration: 510 Train loss: 0.265
Epoch: 3/10 Iteration: 515 Train loss: 0.229
Epoch: 3/10 Iteration: 520 Train loss: 0.211
Epoch: 3/10 Iteration: 525 Train loss: 0.216
Val acc: 0.330
Epoch: 3/10 Iteration: 530 Train loss: 0.253
```

```
Epoch: 3/10 Iteration: 535 Train loss: 0.258
Epoch: 3/10 Iteration: 540 Train loss: 0.196
Epoch: 3/10 Iteration: 545 Train loss: 0.193
Epoch: 3/10 Iteration: 550 Train loss: 0.205
Val acc: 0.704
Epoch: 3/10 Iteration: 555 Train loss: 0.191
Epoch: 3/10 Iteration: 560 Train loss: 0.257
Epoch: 4/10 Iteration: 565 Train loss: 0.346
Epoch: 4/10 Iteration: 570 Train loss: 0.253
Epoch: 4/10 Iteration: 575 Train loss: 0.195
Val acc: 0.626
Epoch: 4/10 Iteration: 580 Train loss: 0.207
Epoch: 4/10 Iteration: 585 Train loss: 0.255
Epoch: 4/10 Iteration: 590 Train loss: 0.246
Epoch: 4/10 Iteration: 595 Train loss: 0.229
Epoch: 4/10 Iteration: 600 Train loss: 0.235
Val acc: 0.722
Epoch: 4/10 Iteration: 605 Train loss: 0.259
Epoch: 4/10 Iteration: 610 Train loss: 0.183
Epoch: 4/10 Iteration: 615 Train loss: 0.261
Epoch: 4/10 Iteration: 620 Train loss: 0.214
Epoch: 4/10 Iteration: 625 Train loss: 0.189
Val acc: 0.470
Epoch: 4/10 Iteration: 630 Train loss: 0.297
Epoch: 4/10 Iteration: 635 Train loss: 0.254
Epoch: 4/10 Iteration: 640 Train loss: 0.268
Epoch: 4/10 Iteration: 645 Train loss: 0.237
Epoch: 4/10 Iteration: 650 Train loss: 0.189
Val acc: 0.409
Epoch: 4/10 Iteration: 655 Train loss: 0.260
Epoch: 4/10 Iteration: 660 Train loss: 0.186
Epoch: 4/10 Iteration: 665 Train loss: 0.214
Epoch: 4/10 Iteration: 670 Train loss: 0.214
Epoch: 4/10 Iteration: 675 Train loss: 0.217
Val acc: 0.522
Epoch: 4/10 Iteration: 680 Train loss: 0.125
Epoch: 4/10 Iteration: 685 Train loss: 0.224
Epoch: 4/10 Iteration: 690 Train loss: 0.202
Epoch: 4/10 Iteration: 695 Train loss: 0.210
Epoch: 4/10 Iteration: 700 Train loss: 0.127
Val acc: 0.496
Epoch: 4/10 Iteration: 705 Train loss: 0.078
Epoch: 4/10 Iteration: 710 Train loss: 0.201
Epoch: 4/10 Iteration: 715 Train loss: 0.182
Epoch: 4/10 Iteration: 720 Train loss: 0.155
Epoch: 4/10 Iteration: 725 Train loss: 0.154
Val acc: 0.704
Epoch: 4/10 Iteration: 730 Train loss: 0.142
```

```
Epoch: 4/10 Iteration: 735 Train loss: 0.045
Epoch: 4/10 Iteration: 740 Train loss: 0.228
Epoch: 4/10 Iteration: 745 Train loss: 0.147
Epoch: 4/10 Iteration: 750 Train loss: 0.122
Val acc: 0.739
Epoch: 5/10 Iteration: 755 Train loss: 0.457
Epoch: 5/10 Iteration: 760 Train loss: 0.336
Epoch: 5/10 Iteration: 765 Train loss: 0.311
Epoch: 5/10 Iteration: 770 Train loss: 0.244
Epoch: 5/10 Iteration: 775 Train loss: 0.193
Val acc: 0.670
Epoch: 5/10 Iteration: 780 Train loss: 0.298
Epoch: 5/10 Iteration: 785 Train loss: 0.131
Epoch: 5/10 Iteration: 790 Train loss: 0.225
Epoch: 5/10 Iteration: 795 Train loss: 0.172
Epoch: 5/10 Iteration: 800 Train loss: 0.246
Val acc: 0.635
Epoch: 5/10 Iteration: 805 Train loss: 0.176
Epoch: 5/10 Iteration: 810 Train loss: 0.215
Epoch: 5/10 Iteration: 815 Train loss: 0.153
Epoch: 5/10 Iteration: 820 Train loss: 0.188
Epoch: 5/10 Iteration: 825 Train loss: 0.232
Val acc: 0.565
Epoch: 5/10 Iteration: 830 Train loss: 0.124
Epoch: 5/10 Iteration: 835 Train loss: 0.136
Epoch: 5/10 Iteration: 840 Train loss: 0.075
Epoch: 5/10 Iteration: 845 Train loss: 0.174
Epoch: 5/10 Iteration: 850 Train loss: 0.210
Val acc: 0.487
Epoch: 5/10 Iteration: 855 Train loss: 0.170
Epoch: 5/10 Iteration: 860 Train loss: 0.247
Epoch: 5/10 Iteration: 865 Train loss: 0.309
Epoch: 5/10 Iteration: 870 Train loss: 0.036
Epoch: 5/10 Iteration: 875 Train loss: 0.128
Val acc: 0.609
Epoch: 5/10 Iteration: 880 Train loss: 0.084
Epoch: 5/10 Iteration: 885 Train loss: 0.086
Epoch: 5/10 Iteration: 890 Train loss: 0.080
Epoch: 5/10 Iteration: 895 Train loss: 0.137
Epoch: 5/10 Iteration: 900 Train loss: 0.087
Val acc: 0.513
Epoch: 5/10 Iteration: 905 Train loss: 0.154
Epoch: 5/10 Iteration: 910 Train loss: 0.183
Epoch: 5/10 Iteration: 915 Train loss: 0.068
Epoch: 5/10 Iteration: 920 Train loss: 0.105
Epoch: 5/10 Iteration: 925 Train loss: 0.231
Val acc: 0.678
Epoch: 5/10 Iteration: 930 Train loss: 0.195
```

```
Epoch: 5/10 Iteration: 935 Train loss: 0.144
Epoch: 5/10 Iteration: 940 Train loss: 0.142
Epoch: 6/10 Iteration: 945 Train loss: 0.286
Epoch: 6/10 Iteration: 950 Train loss: 0.342
Val acc: 0.748
Epoch: 6/10 Iteration: 955 Train loss: 0.230
Epoch: 6/10 Iteration: 960 Train loss: 0.072
Epoch: 6/10 Iteration: 965 Train loss: 0.297
Epoch: 6/10 Iteration: 970 Train loss: 0.153
Epoch: 6/10 Iteration: 975 Train loss: 0.141
Val acc: 0.730
Epoch: 6/10 Iteration: 980 Train loss: 0.314
Epoch: 6/10 Iteration: 985 Train loss: 0.040
Epoch: 6/10 Iteration: 990 Train loss: 0.058
Epoch: 6/10 Iteration: 995 Train loss: 0.050
Epoch: 6/10 Iteration: 1000 Train loss: 0.211
Val acc: 0.565
Epoch: 6/10 Iteration: 1005 Train loss: 0.199
Epoch: 6/10 Iteration: 1010 Train loss: 0.059
Epoch: 6/10 Iteration: 1015 Train loss: 0.040
Epoch: 6/10 Iteration: 1020 Train loss: 0.091
Epoch: 6/10 Iteration: 1025 Train loss: 0.058
Val acc: 0.539
Epoch: 6/10 Iteration: 1030 Train loss: 0.053
Epoch: 6/10 Iteration: 1035 Train loss: 0.088
Epoch: 6/10 Iteration: 1040 Train loss: 0.102
Epoch: 6/10 Iteration: 1045 Train loss: 0.028
Epoch: 6/10 Iteration: 1050 Train loss: 0.005
Val acc: 0.522
Epoch: 6/10 Iteration: 1055 Train loss: 0.011
Epoch: 6/10 Iteration: 1060 Train loss: 0.061
Epoch: 6/10 Iteration: 1065 Train loss: 0.088
Epoch: 6/10 Iteration: 1070 Train loss: 0.008
Epoch: 6/10 Iteration: 1075 Train loss: 0.173
Val acc: 0.609
Epoch: 6/10 Iteration: 1080 Train loss: 0.202
Epoch: 6/10 Iteration: 1085 Train loss: 0.135
Epoch: 6/10 Iteration: 1090 Train loss: 0.005
Epoch: 6/10 Iteration: 1095 Train loss: 0.114
Epoch: 6/10 Iteration: 1100 Train loss: 0.171
Val acc: 0.600
Epoch: 6/10 Iteration: 1105 Train loss: 0.129
Epoch: 6/10 Iteration: 1110 Train loss: 0.236
Epoch: 6/10 Iteration: 1115 Train loss: 0.025
Epoch: 6/10 Iteration: 1120 Train loss: 0.055
Epoch: 6/10 Iteration: 1125 Train loss: 0.207
Val acc: 0.704
Epoch: 7/10 Iteration: 1130 Train loss: 0.172
```

```
Epoch: 7/10 Iteration: 1135 Train loss: 0.013
Epoch: 7/10 Iteration: 1140 Train loss: 0.062
Epoch: 7/10 Iteration: 1145 Train loss: 0.189
Epoch: 7/10 Iteration: 1150 Train loss: 0.039
Val acc: 0.696
Epoch: 7/10 Iteration: 1155 Train loss: 0.190
Epoch: 7/10 Iteration: 1160 Train loss: 0.353
Epoch: 7/10 Iteration: 1165 Train loss: 0.122
Epoch: 7/10 Iteration: 1170 Train loss: 0.042
Epoch: 7/10 Iteration: 1175 Train loss: 0.283
Val acc: 0.583
Epoch: 7/10 Iteration: 1180 Train loss: 0.045
Epoch: 7/10 Iteration: 1185 Train loss: 0.049
Epoch: 7/10 Iteration: 1190 Train loss: 0.003
Epoch: 7/10 Iteration: 1195 Train loss: 0.019
Epoch: 7/10 Iteration: 1200 Train loss: 0.029
Val acc: 0.522
Epoch: 7/10 Iteration: 1205 Train loss: 0.062
Epoch: 7/10 Iteration: 1210 Train loss: 0.007
Epoch: 7/10 Iteration: 1215 Train loss: 0.014
Epoch: 7/10 Iteration: 1220 Train loss: 0.058
Epoch: 7/10 Iteration: 1225 Train loss: 0.016
Val acc: 0.530
Epoch: 7/10 Iteration: 1230 Train loss: 0.037
Epoch: 7/10 Iteration: 1235 Train loss: 0.005
Epoch: 7/10 Iteration: 1240 Train loss: 0.017
Epoch: 7/10 Iteration: 1245 Train loss: 0.076
Epoch: 7/10 Iteration: 1250 Train loss: 0.030
Val acc: 0.539
Epoch: 7/10 Iteration: 1255 Train loss: 0.144
Epoch: 7/10 Iteration: 1260 Train loss: 0.070
Epoch: 7/10 Iteration: 1265 Train loss: 0.058
Epoch: 7/10 Iteration: 1270 Train loss: 0.185
Epoch: 7/10 Iteration: 1275 Train loss: 0.238
Val acc: 0.504
Epoch: 7/10 Iteration: 1280 Train loss: 0.091
Epoch: 7/10 Iteration: 1285 Train loss: 0.061
Epoch: 7/10 Iteration: 1290 Train loss: 0.258
Epoch: 7/10 Iteration: 1295 Train loss: 0.110
Epoch: 7/10 Iteration: 1300 Train loss: 0.149
Val acc: 0.548
Epoch: 7/10 Iteration: 1305 Train loss: 0.123
Epoch: 7/10 Iteration: 1310 Train loss: 0.147
Epoch: 7/10 Iteration: 1315 Train loss: 0.029
Epoch: 8/10 Iteration: 1320 Train loss: 0.171
Epoch: 8/10 Iteration: 1325 Train loss: 0.009
Val acc: 0.687
Epoch: 8/10 Iteration: 1330 Train loss: 0.002
```

```
Epoch: 8/10 Iteration: 1335 Train loss: 0.002
Epoch: 8/10 Iteration: 1340 Train loss: 0.002
Epoch: 8/10 Iteration: 1345 Train loss: 0.021
Epoch: 8/10 Iteration: 1350 Train loss: 0.232
Val acc: 0.678
Epoch: 8/10 Iteration: 1355 Train loss: 0.036
Epoch: 8/10 Iteration: 1360 Train loss: 0.004
Epoch: 8/10 Iteration: 1365 Train loss: 0.114
Epoch: 8/10 Iteration: 1370 Train loss: 0.201
Epoch: 8/10 Iteration: 1375 Train loss: 0.005
Val acc: 0.504
Epoch: 8/10 Iteration: 1380 Train loss: 0.008
Epoch: 8/10 Iteration: 1385 Train loss: 0.091
Epoch: 8/10 Iteration: 1390 Train loss: 0.015
Epoch: 8/10 Iteration: 1395 Train loss: 0.001
Epoch: 8/10 Iteration: 1400 Train loss: 0.006
Val acc: 0.557
Epoch: 8/10 Iteration: 1405 Train loss: 0.002
Epoch: 8/10 Iteration: 1410 Train loss: 0.003
Epoch: 8/10 Iteration: 1415 Train loss: 0.001
Epoch: 8/10 Iteration: 1420 Train loss: 0.030
Epoch: 8/10 Iteration: 1425 Train loss: 0.151
Val acc: 0.522
Epoch: 8/10 Iteration: 1430 Train loss: 0.019
Epoch: 8/10 Iteration: 1435 Train loss: 0.002
Epoch: 8/10 Iteration: 1440 Train loss: 0.001
Epoch: 8/10 Iteration: 1445 Train loss: 0.017
Epoch: 8/10 Iteration: 1450 Train loss: 0.001
Val acc: 0.565
Epoch: 8/10 Iteration: 1455 Train loss: 0.019
Epoch: 8/10 Iteration: 1460 Train loss: 0.013
Epoch: 8/10 Iteration: 1465 Train loss: 0.004
Epoch: 8/10 Iteration: 1470 Train loss: 0.077
Epoch: 8/10 Iteration: 1475 Train loss: 0.094
Val acc: 0.617
Epoch: 8/10 Iteration: 1480 Train loss: 0.010
Epoch: 8/10 Iteration: 1485 Train loss: 0.088
Epoch: 8/10 Iteration: 1490 Train loss: 0.022
Epoch: 8/10 Iteration: 1495 Train loss: 0.007
Epoch: 8/10 Iteration: 1500 Train loss: 0.110
Val acc: 0.643
Epoch: 9/10 Iteration: 1505 Train loss: 0.007
Epoch: 9/10 Iteration: 1510 Train loss: 0.004
Epoch: 9/10 Iteration: 1515 Train loss: 0.034
Epoch: 9/10 Iteration: 1520 Train loss: 0.000
Epoch: 9/10 Iteration: 1525 Train loss: 0.001
Val acc: 0.643
Epoch: 9/10 Iteration: 1530 Train loss: 0.001
```

```
Epoch: 9/10 Iteration: 1535 Train loss: 0.004
Epoch: 9/10 Iteration: 1540 Train loss: 0.001
Epoch: 9/10 Iteration: 1545 Train loss: 0.080
Epoch: 9/10 Iteration: 1550 Train loss: 0.000
Val acc: 0.557
Epoch: 9/10 Iteration: 1555 Train loss: 0.018
Epoch: 9/10 Iteration: 1560 Train loss: 0.051
Epoch: 9/10 Iteration: 1565 Train loss: 0.194
Epoch: 9/10 Iteration: 1570 Train loss: 0.001
Epoch: 9/10 Iteration: 1575 Train loss: 0.001
Val acc: 0.478
Epoch: 9/10 Iteration: 1580 Train loss: 0.149
Epoch: 9/10 Iteration: 1585 Train loss: 0.004
Epoch: 9/10 Iteration: 1590 Train loss: 0.001
Epoch: 9/10 Iteration: 1595 Train loss: 0.001
Epoch: 9/10 Iteration: 1600 Train loss: 0.001
Val acc: 0.530
Epoch: 9/10 Iteration: 1605 Train loss: 0.007
Epoch: 9/10 Iteration: 1610 Train loss: 0.017
Epoch: 9/10 Iteration: 1615 Train loss: 0.004
Epoch: 9/10 Iteration: 1620 Train loss: 0.000
Epoch: 9/10 Iteration: 1625 Train loss: 0.027
Val acc: 0.548
Epoch: 9/10 Iteration: 1630 Train loss: 0.202
Epoch: 9/10 Iteration: 1635 Train loss: 0.005
Epoch: 9/10 Iteration: 1640 Train loss: 0.010
Epoch: 9/10 Iteration: 1645 Train loss: 0.000
Epoch: 9/10 Iteration: 1650 Train loss: 0.000
Val acc: 0.557
Epoch: 9/10 Iteration: 1655 Train loss: 0.024
Epoch: 9/10 Iteration: 1660 Train loss: 0.048
Epoch: 9/10 Iteration: 1665 Train loss: 0.028
Epoch: 9/10 Iteration: 1670 Train loss: 0.151
Epoch: 9/10 Iteration: 1675 Train loss: 0.038
Val acc: 0.617
Epoch: 9/10 Iteration: 1680 Train loss: 0.119
Epoch: 9/10 Iteration: 1685 Train loss: 0.169
Epoch: 9/10 Iteration: 1690 Train loss: 0.022
Epoch: 10/10 Iteration: 1695 Train loss: 0.165
Epoch: 10/10 Iteration: 1700 Train loss: 0.008
Val acc: 0.661
Epoch: 10/10 Iteration: 1705 Train loss: 0.197
Epoch: 10/10 Iteration: 1710 Train loss: 0.116
Epoch: 10/10 Iteration: 1715 Train loss: 0.001
Epoch: 10/10 Iteration: 1720 Train loss: 0.001
Epoch: 10/10 Iteration: 1725 Train loss: 0.001
Val acc: 0.635
Epoch: 10/10 Iteration: 1730 Train loss: 0.005
```

```
Epoch: 10/10 Iteration: 1735 Train loss: 0.005
Epoch: 10/10 Iteration: 1740 Train loss: 0.000
Epoch: 10/10 Iteration: 1745 Train loss: 0.000
Epoch: 10/10 Iteration: 1750 Train loss: 0.182
Val acc: 0.617
Epoch: 10/10 Iteration: 1755 Train loss: 0.231
Epoch: 10/10 Iteration: 1760 Train loss: 0.000
Epoch: 10/10 Iteration: 1765 Train loss: 0.002
Epoch: 10/10 Iteration: 1770 Train loss: 0.000
Epoch: 10/10 Iteration: 1775 Train loss: 0.040
Val acc: 0.548
Epoch: 10/10 Iteration: 1780 Train loss: 0.002
Epoch: 10/10 Iteration: 1785 Train loss: 0.000
Epoch: 10/10 Iteration: 1790 Train loss: 0.097
Epoch: 10/10 Iteration: 1795 Train loss: 0.034
Epoch: 10/10 Iteration: 1800 Train loss: 0.192
Val acc: 0.583
Epoch: 10/10 Iteration: 1805 Train loss: 0.000
Epoch: 10/10 Iteration: 1810 Train loss: 0.001
Epoch: 10/10 Iteration: 1815 Train loss: 0.002
Epoch: 10/10 Iteration: 1820 Train loss: 0.000
Epoch: 10/10 Iteration: 1825 Train loss: 0.005
Val acc: 0.539
Epoch: 10/10 Iteration: 1830 Train loss: 0.001
Epoch: 10/10 Iteration: 1835 Train loss: 0.000
Epoch: 10/10 Iteration: 1840 Train loss: 0.001
Epoch: 10/10 Iteration: 1845 Train loss: 0.112
Epoch: 10/10 Iteration: 1850 Train loss: 0.032
Val acc: 0.591
Epoch: 10/10 Iteration: 1855 Train loss: 0.065
Epoch: 10/10 Iteration: 1860 Train loss: 0.006
Epoch: 10/10 Iteration: 1865 Train loss: 0.059
Epoch: 10/10 Iteration: 1870 Train loss: 0.001
Epoch: 10/10 Iteration: 1875 Train loss: 0.144
Val acc: 0.635
Epoch: 10/10 Iteration: 1880 Train loss: 0.186
Time elasped = 3771.694607991 sec(s)
```

```
In [22]: # Testing::

         test_acc = []
         with tf.Session() as sess:
             saver.restore(sess, checkpointName)
             test_state = sess.run(cell.zero_state(batch_size, tf.float32))
             for ii, (x, y) in enumerate(get_batches(test_x, test_y, batch_size), 1):
                 feed = {inputs_: x,
                         labels_: y[:, None],
                         keep_prob: 1,
                         initial_state: test_state}
                 batch_acc, test_state = sess.run([accuracy, final_state], feed_dict=feed)
                 test_acc.append(batch_acc)
             print("Test accuracy: {:.3f}".format(np.mean(test_acc)))
```

```
INFO:tensorflow:Restoring parameters from checkpoints/lstm_with_unique_14.ckpt
Test accuracy: 0.626
```

```
In [0]:
```