In [66]:

```
import time
from collections import namedtuple
import numpy as np
import tensorflow as tf
import nltk
from nltk import word_tokenize
import re
from collections import Counter

with open('episodes\\HP1.txt', 'r') as f:
    text=f.read()
vocab_list = Counter([re.sub('[!"#$%&()*+,-./:;<=>?@[\]^_`{|}~0-9]', '', i) for i in word_t
vocab_list = dict(vocab_list)
for i in list(vocab_list.keys()):
    if vocab_list[i] < 5:
        del vocab_list[i]
vocab_list = list(vocab_list.keys())
print(vocab_list)
vocab = sorted(set(text))
vocab_to_int = {c: i for i, c in enumerate(vocab)}
int_to_vocab = dict(enumerate(vocab))
encoded = np.array([vocab_to_int[c] for c in text], dtype=np.int32)

#print(vocab)
#print(vocab_to_int)
#print(int_to_vocab)

#encoded contains the entire text, encoded character-wise. Example: HARRY: 29 56 ...etc whe
#print(encoded)

def get_batches(arr, batch_size, n_steps):
    chars_per_batch = batch_size * n_steps
    n_batches = len(arr)//chars_per_batch
    arr = arr[:n_batches * chars_per_batch]
    arr = arr.reshape((batch_size, -1))

    for n in range(0, arr.shape[1], n_steps):
        x = arr[:, n:n+n_steps]
        y_temp = arr[:, n+1:n+n_steps+1]
        y = np.zeros(x.shape, dtype=x.dtype)
        y[:,:y_temp.shape[1]] = y_temp

        yield x, y


#batches = get_batches(encoded, 10, 50)
#x,y = next(batches)

#print(x,y)
print(len(vocab_list))
```

```
['the', 'sorting', 'hat', 'door', 'swung', 'open', 'once', 'tall', 'witc
h', 'robes', 'stood', 'there', 'she', 'had', 'very', 'face', 'and', 'harr
y', 'first', 'thought', 'was', 'that', 'this', 'not', 'someone', 'years',
'professor', 'mcgonagall', 'said', 'hagrid', 'you', 'will', 'take', 'the
m', 'from', 'here', 'pulled', 'wide', 'entrance', 'hall', 'big', 'could',
'have', 'whole', 'dursleys', 'house', 'stone', 'walls', 'were', 'with', 'l
ike', 'ones', 'gringotts', 'ceiling', 'too', 'high', 'make', 'out', 'marbl
```

```
e', 'staircase', 'they', 'followed', 'across', 'floor', 'hear', 'hundred
s', 'right', 'rest', 'school', 'must', 'already', 'but', 'showed', 'into',
'small', 'empty', 'chamber', 'off', 'standing', 'together', 'than', 'woul
d', 'usually', 'done', 'about', 'hogwarts', 'before', 'your', 'great', 'ho
uses', 'important', 'because', 'while', 'are', 'something', 'family', 'cla
sses', 'sleep', 'dormitory', 'free', 'time', 'common', 'room', 'four', 'ca
lled', 'gryffindor', 'hufflepuff', 'ravenclaw', 'slytherin', 'each', 'ha
s', 'its', 'own', 'wizards', 'points', 'any', 'rule', 'breaking', 'lose',
'end', 'year', 'most', 'cup', 'hope', 'place', 'few', 'minutes', 'front',
'all', 'much', 'can', 'waiting', 'her', 'eyes', 'for', 'moment', 'nevill
e', 'cloak', 'which', 'under', 'his', 'left', 'ear', 'ron', 'nose', 'trie
d', 'hair', 'when', 'ready', 'please', 'wait', 'quietly', 'how', 'exactl
```

In [2]:

```
def build_inputs(batch_size, num_steps):
    ''' Define placeholders for inputs, targets, and dropout'''
    inputs = tf.placeholder(tf.int32, [batch_size, num_steps], name='inputs')
    targets = tf.placeholder(tf.int32, [batch_size, num_steps], name='targets')
    keep_prob = tf.placeholder(tf.float32, name='keep_prob')

    return inputs, targets, keep_prob
```

In [3]:

```
def build_lstm(lstm_size, num_layers, batch_size, keep_prob):
    ''' Build LSTM cell.
        lstm_size: Size of the hidden layers in the LSTM cells
        num_layers: Number of LSTM layers'''

    def build_cell(lstm_size, keep_prob):
        lstm = tf.contrib.rnn.BasicLSTMCell(lstm_size)
        drop = tf.contrib.rnn.DropoutWrapper(lstm, output_keep_prob=keep_prob)
        return drop
    cell = tf.contrib.rnn.MultiRNNCell([build_cell(lstm_size, keep_prob) for _ in range(num
    initial_state = cell.zero_state(batch_size, tf.float32)

    return cell, initial_state
```

In [4]:

```
def build_output(lstm_output, in_size, out_size):
    ''' Build a softmax layer, return the softmax output and logits.
        x: Input tensor
        in_size: Size of the input tensor, for example, size of the LSTM cells
        out_size: Size of this softmax layer

    '''
    seq_output = tf.concat(lstm_output, axis=1)
    x = tf.reshape(seq_output, [-1, in_size])
    with tf.variable_scope('softmax'):
        softmax_w = tf.Variable(tf.truncated_normal((in_size, out_size), stddev=0.1))
        softmax_b = tf.Variable(tf.zeros(out_size))
    logits = tf.matmul(x, softmax_w) + softmax_b
    out = tf.nn.softmax(logits, name='predictions')

    return out, logits
```

In [5]:

```
def build_loss(logits, targets, lstm_size, num_classes):
    ''' Calculate the loss from the logits and the targets.
        logits: Logits from final fully connected layer
        targets: Targets for supervised learning
        lstm_size: Number of LSTM hidden units
        num_classes: Number of classes in targets

    '''
    y_one_hot = tf.one_hot(targets, num_classes)
    y_reshaped = tf.reshape(y_one_hot, logits.get_shape())
    loss = tf.nn.softmax_cross_entropy_with_logits(logits=logits, labels=y_reshaped)
    loss = tf.reduce_mean(loss)
    return loss
```

In [6]:

```
def build_optimizer(loss, learning_rate, grad_clip):
    ''' Build optmizer for training, using gradient clipping.
        loss: Network loss
        learning_rate: Learning rate for optimizer

    '''
    tvars = tf.trainable_variables()
    grads, _ = tf.clip_by_global_norm(tf.gradients(loss, tvars), grad_clip)
    train_op = tf.train.AdamOptimizer(learning_rate)
    optimizer = train_op.apply_gradients(zip(grads, tvars))

    return optimizer
```

In [7]:

```
class CharRNN:

    def __init__(self, num_classes, batch_size=64, num_steps=50, lstm_size=128, num_layers=
        if sampling == True:
            batch_size, num_steps = 1, 1
        else:
            batch_size, num_steps = batch_size, num_steps

        tf.reset_default_graph()
        self.inputs, self.targets, self.keep_prob = build_inputs(batch_size, num_steps)

        cell, self.initial_state = build_lstm(lstm_size, num_layers, batch_size, self.keep_

        x_one_hot = tf.one_hot(self.inputs, num_classes)
        outputs, state = tf.nn.dynamic_rnn(cell, x_one_hot, initial_state=self.initial_stat
        self.final_state = state
        self.prediction, self.logits = build_output(outputs, lstm_size, num_classes)
        self.loss = build_loss(self.logits, self.targets, lstm_size, num_classes)
        self.optimizer = build_optimizer(self.loss, learning_rate, grad_clip)
```

In [73]:

```
epochs = 250
print_every_n = 50
save_every_n = 200
batch_size = 256
num_steps = 50
lstm_size = 128
num_layers = 2
learning_rate =0.1

model = CharRNN(len(vocab), batch_size=batch_size, num_steps=num_steps,lstm_size=lstm_size,

saver = tf.train.Saver(max_to_keep=100)
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    counter = 0
    for e in range(epochs):
        new_state = sess.run(model.initial_state)
        loss = 0
        for x, y in get_batches(encoded, batch_size, num_steps):
            counter += 1
            start = time.time()
            feed = {model.inputs: x, model.targets: y, model.keep_prob: 0.6, model.initial_
            batch_loss, new_state, _ = sess.run([model.loss, model.final_state, model.optim
            if (counter % print_every_n == 0):
                end = time.time()
                print('Epoch: {}/{}... '.format(e+1, epochs),
                      'Training Step: {}... '.format(counter),
                      'Training loss: {:.4f}... '.format(batch_loss),
                      '{:.4f} sec/batch'.format((end-start)))

            if (counter % save_every_n == 0):
                saver.save(sess, "checkpoints/i{}_l{}.ckpt".format(counter, lstm_size))

    saver.save(sess, "checkpoints/i{}_l{}.ckpt".format(counter, lstm_size))
```

```
WARNING: Entity <bound method BasicLSTMCell.call of <tensorflow.python.op
s.rnn_cell_impl.BasicLSTMCell object at 0x000001708A418288>> could not be
transformed and will be executed as-is. Please report this to the AutgoGra
ph team. When filing the bug, set the verbosity to 10 (on Linux, `export A
UTOGRAPH_VERBOSITY=10`) and attach the full output. Cause: converting <bou
nd method BasicLSTMCell.call of <tensorflow.python.ops.rnn_cell_impl.Basic
LSTMCell object at 0x000001708A418288>>: AssertionError: Bad argument numb
er for Name: 3, expecting 4
Epoch: 4/250...  Training Step: 50...  Training loss: 3.1285...  0.0700 se
c/batch
Epoch: 8/250...  Training Step: 100...  Training loss: 3.0435...  0.1000 s
ec/batch
Epoch: 11/250...  Training Step: 150...  Training loss: 3.0286...  0.0980
sec/batch
Epoch: 15/250...  Training Step: 200...  Training loss: 3.0154...  0.1000
sec/batch
Epoch: 18/250...  Training Step: 250...  Training loss: 2.9582...  0.1010
sec/batch
Epoch: 22/250...  Training Step: 300...  Training loss: 2.9393...  0.1000
sec/batch
```

In [74]:

```
tf.train.get_checkpoint_state('checkpoints')
```

Out[74]:

```
model_checkpoint_path: "checkpoints\\i3500_l128.ckpt"
all_model_checkpoint_paths: "checkpoints\\i200_l128.ckpt"
all_model_checkpoint_paths: "checkpoints\\i400_l128.ckpt"
all_model_checkpoint_paths: "checkpoints\\i600_l128.ckpt"
all_model_checkpoint_paths: "checkpoints\\i800_l128.ckpt"
all_model_checkpoint_paths: "checkpoints\\i1000_l128.ckpt"
all_model_checkpoint_paths: "checkpoints\\i1200_l128.ckpt"
all_model_checkpoint_paths: "checkpoints\\i1400_l128.ckpt"
all_model_checkpoint_paths: "checkpoints\\i1600_l128.ckpt"
all_model_checkpoint_paths: "checkpoints\\i1800_l128.ckpt"
all_model_checkpoint_paths: "checkpoints\\i2000_l128.ckpt"
all_model_checkpoint_paths: "checkpoints\\i2200_l128.ckpt"
all_model_checkpoint_paths: "checkpoints\\i2400_l128.ckpt"
all_model_checkpoint_paths: "checkpoints\\i2600_l128.ckpt"
all_model_checkpoint_paths: "checkpoints\\i2800_l128.ckpt"
all_model_checkpoint_paths: "checkpoints\\i3000_l128.ckpt"
all_model_checkpoint_paths: "checkpoints\\i3200_l128.ckpt"
all_model_checkpoint_paths: "checkpoints\\i3400_l128.ckpt"
all_model_checkpoint_paths: "checkpoints\\i3500_l128.ckpt"
```

In [75]:

```
def pick_top_n(preds, vocab_size, top_n=5):
    p = np.squeeze(preds)
    p[np.argsort(p)[:-top_n]] = 0
    p = p / np.sum(p)
    c = np.random.choice(vocab_size, 1, p=p)[0]
    return c
```

In [76]:

```
def sample(checkpoint, n_samples, lstm_size, vocab_size, prime="The "):
    samples = [c for c in prime]
    model = CharRNN(len(vocab), lstm_size=lstm_size, sampling=True)
    saver = tf.train.Saver()
    with tf.Session() as sess:
        saver.restore(sess, checkpoint)
        new_state = sess.run(model.initial_state)
        for c in prime:
            x = np.zeros((1, 1))
            x[0,0] = vocab_to_int[c]
            feed = {model.inputs: x,
                    model.keep_prob: 1.,
                    model.initial_state: new_state}
            preds, new_state = sess.run([model.prediction, model.final_state],
                                        feed_dict=feed)

        c = pick_top_n(preds, len(vocab))
        samples.append(int_to_vocab[c])

        for i in range(n_samples):
            x[0,0] = c
            feed = {model.inputs: x,
                    model.keep_prob: 1.,
                    model.initial_state: new_state}
            preds, new_state = sess.run([model.prediction, model.final_state],
                                        feed_dict=feed)

            c = pick_top_n(preds, len(vocab))
            samples.append(int_to_vocab[c])
    return ''.join(samples)
```

In [77]:

```
tf.train.latest_checkpoint('checkpoints')
```

Out[77]:

```
'checkpoints\\i3500_l128.ckpt'
```

In [78]:

```
def editDistDP(str1, str2, m, n):
    dp = [[0 for x in range(n+1)] for x in range(m+1)]
    for i in range(m+1):
        for j in range(n+1):
            if i == 0:
                dp[i][j] = j
            elif j == 0:
                dp[i][j] = i
            elif str1[i-1] == str2[j-1]:
                dp[i][j] = dp[i-1][j-1]
            else:
                dp[i][j] = 1 + min(dp[i][j-1],dp[i-1][j],dp[i-1][j-1])
    return dp[m][n]
```

In [79]:

```
def jaccard_similarity(word1,word2):
    return len(set(word1).intersection(set(word2))) / len(set(word1).union(set(word2)))
```

In [80]:

```
def strcmp(word1,word2):
    return abs(len(word1) - len(word2))
```

In [81]:

```
def min_edit_wrapper(word):

    l1=[]
    word = re.sub('[!"#$%&()*+,-./:;<=>?@[\]^_`{|}~0-9]', '', word)
    for i in vocab_list:
        edit_dist = editDistDP(i,word,len(i),len(word))
        l1.append((strcmp(i,word),edit_dist,jaccard_similarity(i,word),i,word))
    return sorted([i for i in l1 if i[0]<3])
```

In [82]:

```
def untokenize(words):
    """
    Untokenizing a text undoes the tokenizing operation, restoring
    punctuation and spaces to the places that people expect them to be.
    Ideally, `untokenize(tokenize(text))` should be identical to `text`,
    except for line breaks.
    """
    text = ' '.join(words)
    step1 = text.replace("`` ", '"').replace(" ''", '"').replace('. . .',  '...')
    step2 = step1.replace(" ( ", " (").replace(" ) ", ") ")
    step3 = re.sub(r' ([.,:;?!%]+)([ \'"`])', r"\1\2", step2)
    step4 = re.sub(r' ([.,:;?!%]+)$', r"\1", step3)
    step5 = step4.replace(" '", "'").replace(" n't", "n't").replace(
         "can not", "cannot")
    step6 = step5.replace(" ` ", " '")
    return step6.strip()
```

In [83]:

```
checkpoint = tf.train.latest_checkpoint('checkpoints')
samp = sample(checkpoint, 10000, lstm_size, len(vocab), prime="Invisibility Cloak on top of
```

```
WARNING:tensorflow:Entity <bound method MultiRNNCell.call of <tensorflow.pyt
hon.ops.rnn_cell_impl.MultiRNNCell object at 0x000001719B3491C8>> could not
be transformed and will be executed as-is. Please report this to the AutgoGr
aph team. When filing the bug, set the verbosity to 10 (on Linux, `export AU
TOGRAPH_VERBOSITY=10`) and attach the full output. Cause: converting <bound
method MultiRNNCell.call of <tensorflow.python.ops.rnn_cell_impl.MultiRNNCel
l object at 0x000001719B3491C8>>: AttributeError: module 'gast' has no attri
bute 'Num'
WARNING: Entity <bound method MultiRNNCell.call of <tensorflow.python.ops.rn
n_cell_impl.MultiRNNCell object at 0x000001719B3491C8>> could not be transfo
rmed and will be executed as-is. Please report this to the AutgoGraph team.
When filing the bug, set the verbosity to 10 (on Linux, `export AUTOGRAPH_VE
RBOSITY=10`) and attach the full output. Cause: converting <bound method Mul
tiRNNCell.call of <tensorflow.python.ops.rnn_cell_impl.MultiRNNCell object a
t 0x000001719B3491C8>>: AttributeError: module 'gast' has no attribute 'Num'
WARNING:tensorflow:Entity <bound method BasicLSTMCell.call of <tensorflow.py
thon.ops.rnn_cell_impl.BasicLSTMCell object at 0x000001719D4C82C8>> could no
t be transformed and will be executed as-is. Please report this to the Autgo
Graph team. When filing the bug, set the verbosity to 10 (on Linux, `export
AUTOGRAPH_VERBOSITY=10`) and attach the full output. Cause: converting <boun
d method BasicLSTMCell.call of <tensorflow.python.ops.rnn_cell_impl.BasicLST
MCell object at 0x000001719D4C82C8>>: AssertionError: Bad argument number fo
r Name: 3, expecting 4
WARNING: Entity <bound method BasicLSTMCell.call of <tensorflow.python.ops.r
nn_cell_impl.BasicLSTMCell object at 0x000001719D4C82C8>> could not be trans
formed and will be executed as-is. Please report this to the AutgoGraph tea
m. When filing the bug, set the verbosity to 10 (on Linux, `export AUTOGRAPH
_VERBOSITY=10`) and attach the full output. Cause: converting <bound method
BasicLSTMCell.call of <tensorflow.python.ops.rnn_cell_impl.BasicLSTMCell obj
ect at 0x000001719D4C82C8>>: AssertionError: Bad argument number for Name:
3, expecting 4
WARNING:tensorflow:Entity <bound method BasicLSTMCell.call of <tensorflow.py
thon.ops.rnn_cell_impl.BasicLSTMCell object at 0x000001719DD01648>> could no
t be transformed and will be executed as-is. Please report this to the Autgo
Graph team. When filing the bug, set the verbosity to 10 (on Linux, `export
AUTOGRAPH_VERBOSITY=10`) and attach the full output. Cause: converting <boun
d method BasicLSTMCell.call of <tensorflow.python.ops.rnn_cell_impl.BasicLST
MCell object at 0x000001719DD01648>>: AssertionError: Bad argument number fo
r Name: 3, expecting 4
WARNING: Entity <bound method BasicLSTMCell.call of <tensorflow.python.ops.r
nn_cell_impl.BasicLSTMCell object at 0x000001719DD01648>> could not be trans
formed and will be executed as-is. Please report this to the AutgoGraph tea
m. When filing the bug, set the verbosity to 10 (on Linux, `export AUTOGRAPH
_VERBOSITY=10`) and attach the full output. Cause: converting <bound method
BasicLSTMCell.call of <tensorflow.python.ops.rnn_cell_impl.BasicLSTMCell obj
ect at 0x000001719DD01648>>: AssertionError: Bad argument number for Name:
3, expecting 4
INFO:tensorflow:Restoring parameters from checkpoints\i3500_l128.ckpt
```

In [84]:

```
fp = open("expected_output9","w")
fp.write(samp)
fp.close()
```

In [85]:

```
import copy

backupsamp = copy.deepcopy(samp)
backupsamp = word_tokenize(backupsamp)
for i in range(len(backupsamp)):
    if len(backupsamp[i]) >2 and backupsamp[i].lower() not in vocab_list and len(backupsamp
        backupsamp[i] = min_edit_wrapper(backupsamp[i].lower())[0][-2]
```

In [86]:

```
backupsamp2 = untokenize(backupsamp)
fp = open("output9","w")
fp.write(backupsamp2)
fp.close()
```

In [ ]:

```
```