Lab 12

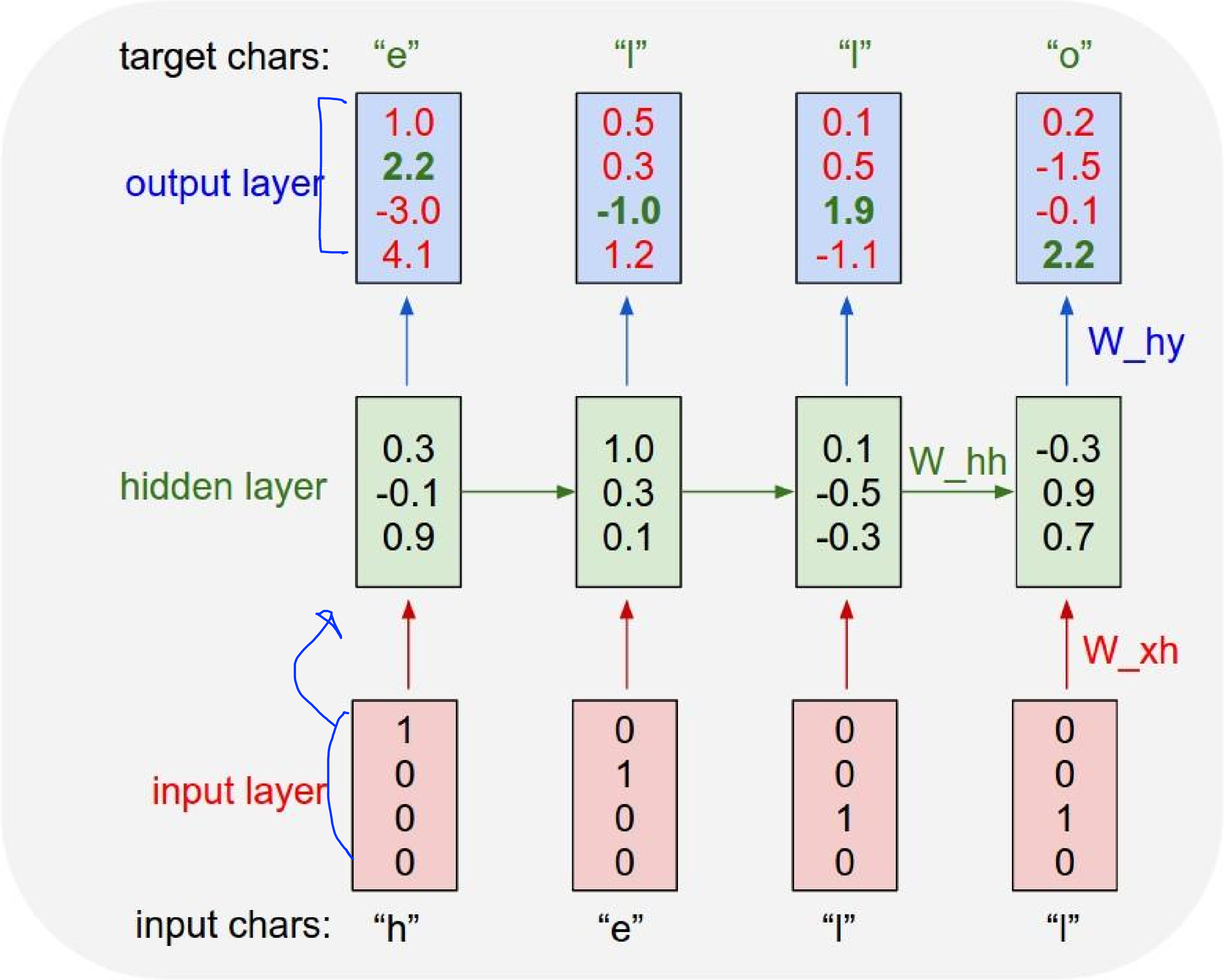
RNN

Sung Kim <hunkim+ml@gmail.com>
http://hunkim.github.io/ml/

Character-level language model example

Vocabulary:
[h,e,l,o]

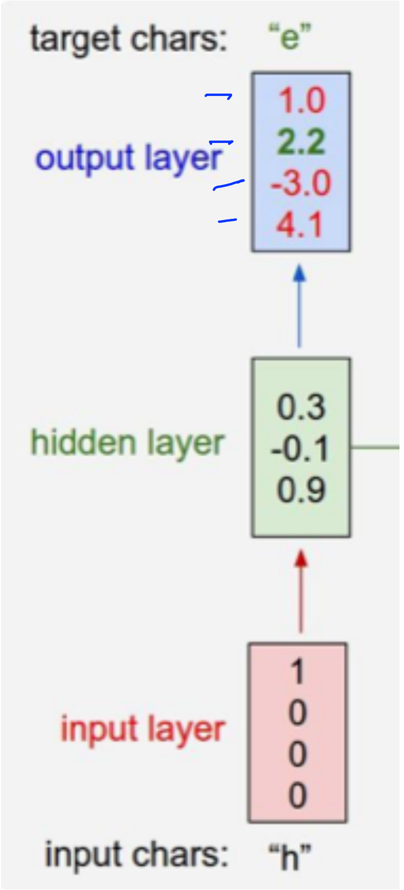
Example training sequence:
“hello”



Creating rnn cell

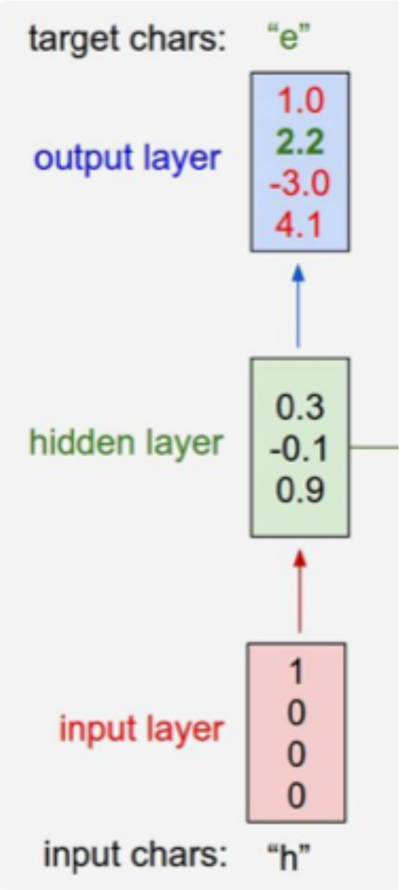
4

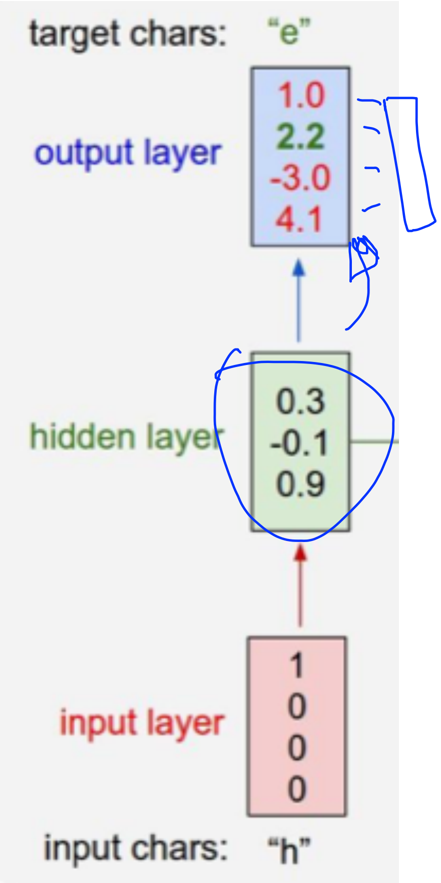
```
# RNN model
rnn_cell = rnn_cell.BasicRNNCell(rnn_size)
```



Creating rnn cell

```
# RNN model
rnn_cell = rnn_cell.BasicRNNCell(rnn_size)


rnn_cell = rnn_cell. BasicLSTMCell(rnn_size)
rnn_cell = rnn_cell. GRUCell(rnn_size)
```

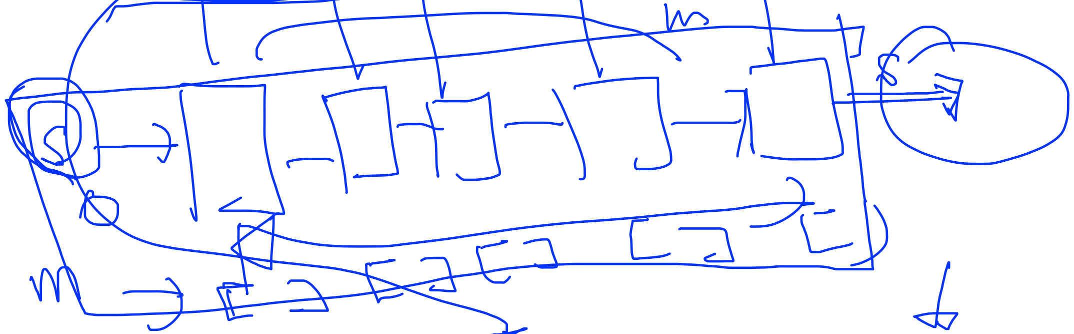




RNN in TensorFlow

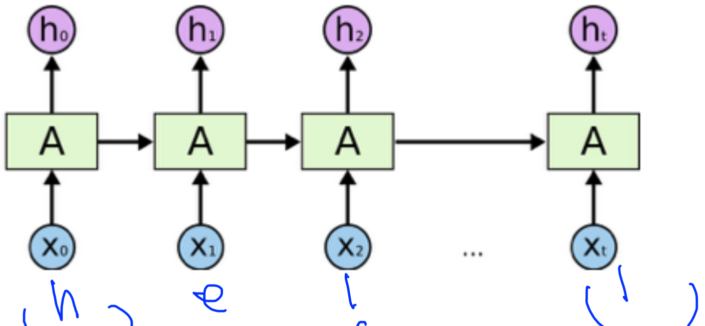
```
# RNN model
rnn_cell = rnn_cell.BasicRNNCell(4)
```

outputs



```
outputs, state = rnn.rnn(rnn_cell, X_split, state)
```

outputs, state = rnn.rnn(rnn_cell, **X_split**, state)



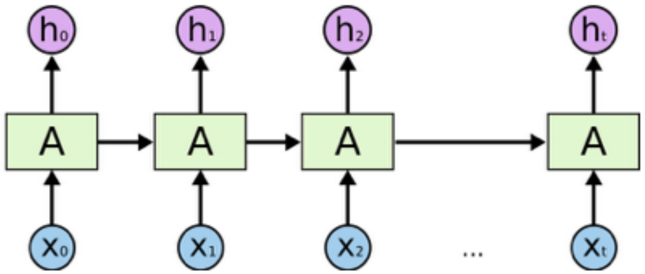
[<tf~~.Tensor 'split~~:0' shape=(1, 4) dtype=float32>,
<tf.~~Tensor 'split~~:1' shape=(1, 4) dtype=float32>,
<tf.Tensor 'split:2' shape=(1, 4) dtype=float32>,
<tf.Tensor 'split:3' shape=(1, 4) dtype=float32>]

outputs, state = rnn.rnn(rnn_cell, **X_split**, state)

```
[<tf.Tensor 'RNN/BasicRNNCell/Tanh:0' shape=(1, 4) dtype=float32>,
 <tf.Tensor 'RNN/BasicRNNCell_1/Tanh:0' shape=(1, 4) dtype=float32>,
 <tf.Tensor 'RNN/BasicRNNCell_2/Tanh:0' shape=(1, 4) dtype=float32>,
 <tf.Tensor 'RNN/BasicRNNCell_3/Tanh:0' shape=(1, 4) dtype=float32>]
```

4

Rnn-size=4



time step-size

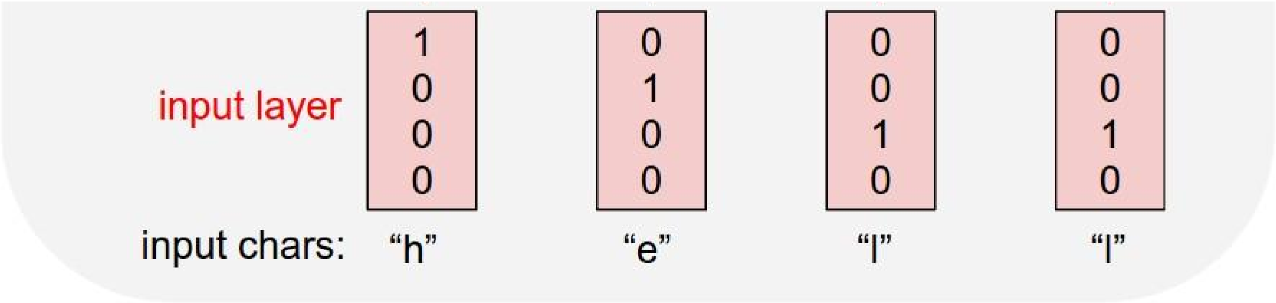
```
[<tf.Tensor 'split:0' shape=(1, 4) dtype=float32>,
 <tf.Tensor 'split:1' shape=(1, 4) dtype=float32>,
 <tf.Tensor 'split:2' shape=(1, 4) dtype=float32>,
 <tf.Tensor 'split:3' shape=(1, 4) dtype=float32>]
```

Character-level language model example

Vocabulary:
[h,e,l,o]

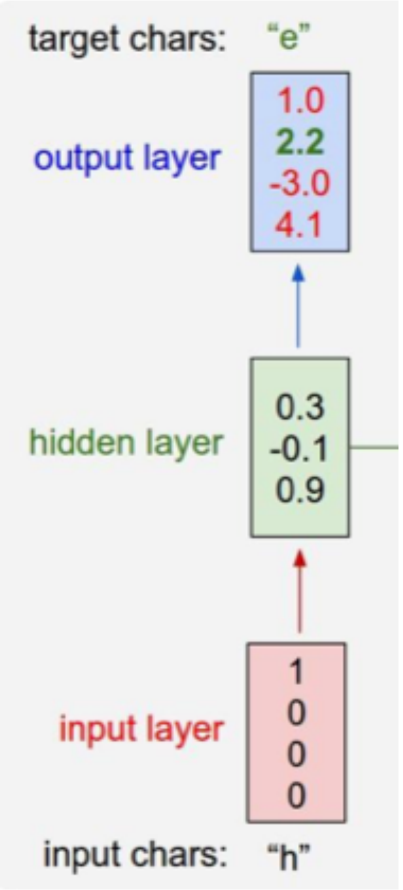
Example training sequence:
“hello”

```
x_data = np.array([ [1,0,0,0],  # h
                    [0,1,0,0],  # e
                    [0,0,1,0],  # l
                    [0,0,1,0]],  # l
                  dtype='f')
```

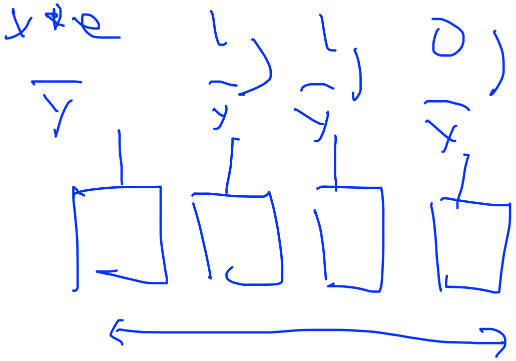


RNN in TensorFlow

```
# RNN model
rnn_cell = rnn_cell.BasicRNNCell(rnn_size)
state = tf.zeros([batch_size, rnn_cell.state_size])
X_split = tf.split(0, time_step_size, x_data)
outputs, state = rnn.rnn(rnn_cell, X_split, state)
```



Cost



```
# logits: list of 2D Tensors of shape [batch_size x num_decoder_symbols].
# targets: list of 1D batch-sized int32 Tensors of the same length as logits.
# weights: list of 1D batch-sized float-Tensors of the same length as logits.
logits = tf.reshape(tf.concat(1, outputs), [-1, rnn_size])
targets = tf.reshape(sample[1:], [-1])
weights = tf.ones([time_step_size * batch_size])

loss = tf.nn.seq2seq.sequence_loss_by_example([logits], [targets], [weights])
cost = tf.reduce_sum(loss) / batch_size
train_op = tf.train.RMSPropOptimizer(0.01, 0.9).minimize(cost)
```

Train & Prediction

```
# Launch the graph in a session
with tf.Session() as sess:
    # you need to initialize all variables
    tf.initialize_all_variables().run()
    for i in range(100):
        sess.run(train_op)
        result = sess.run(tf.arg_max(logits, 1))
        print (result, [char_rdic[t] for t in result])
```

예측

[6

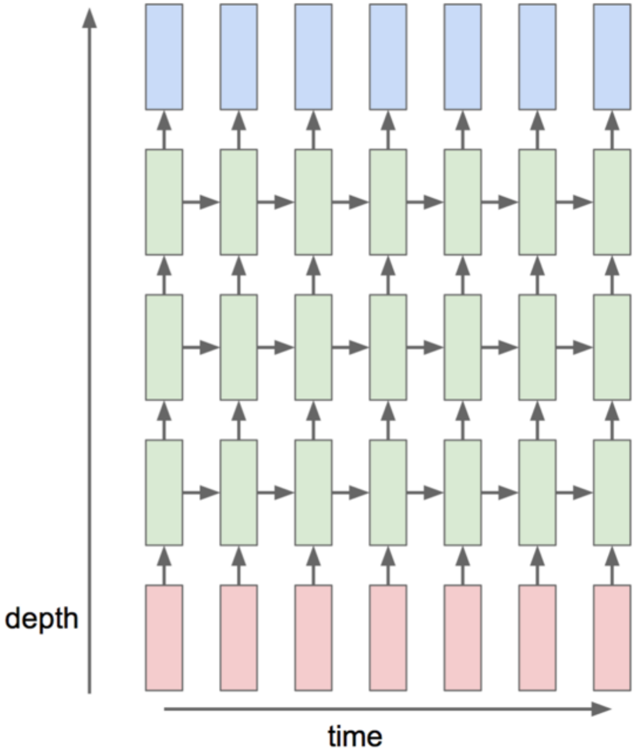
```
import tensorflow as tf
from tensorflow.models.rnn import rnn, rnn_cell
import numpy as np

char_rdic = ['h','e','l','o']  # id -> char
char_dic = {w: i for i, w in enumerate(char_rdic)}  # char -> id
x_data = np.array([ [1,0,0,0],  # h
                    [0,1,0,0],  # e
                    [0,0,1,0],  # l
                    [0,0,1,0]],  # l
                     dtype='f')

sample = [char_dic[c] for c in "hello"] # to index

# Configuration
char_vocab_size = len(char_dic)
rnn_size = char_vocab_size  # 1 hot coding (one of 4)
time_step_size = 4  # 'hell' -> predict 'ello'
batch_size = 1      # one sample

# RNN model
rnn_cell = rnn_cell.BasicRNNCell(rnn_size)
state = tf.zeros([batch_size, rnn_cell.state_size])
X_split = tf.split(0, time_step_size, x_data)
outputs, state = rnn.rnn(rnn_cell, X_split, state)
```

```
# logits: list of 2D Tensors of shape [batch_size x num_decoder_symbols].
# targets: list of 1D batch-sized int32 Tensors of the same length as logits.
# weights: list of 1D batch-sized float-Tensors of the same length as logits.
logits = tf.reshape(tf.concat(1, outputs), [-1, rnn_size])
targets = tf.reshape(sample[1:], [-1])
weights = tf.ones([time_step_size * batch_size])

loss = tf.nn.seq2seq.sequence_loss_by_example([logits], [targets], [weights])
cost = tf.reduce_sum(loss) / batch_size
train_op = tf.train.RMSPropOptimizer(0.01, 0.9).minimize(cost)

# Launch the graph in a session
with tf.Session() as sess:
    # you need to initialize all variables
    tf.initialize_all_variables().run()
    for i in range(100):
        sess.run(train_op)
        result = sess.run(tf.arg_max(logits, 1))
        print (result, [char_rdic[t] for t in result])
```

Output

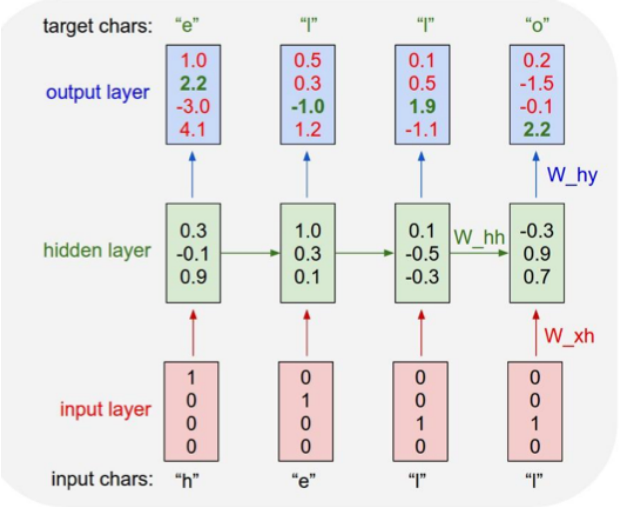
```
result = sess.run(tf.arg_max(logits, 1))
print (result, [char_rdic[t] for t in result])
```

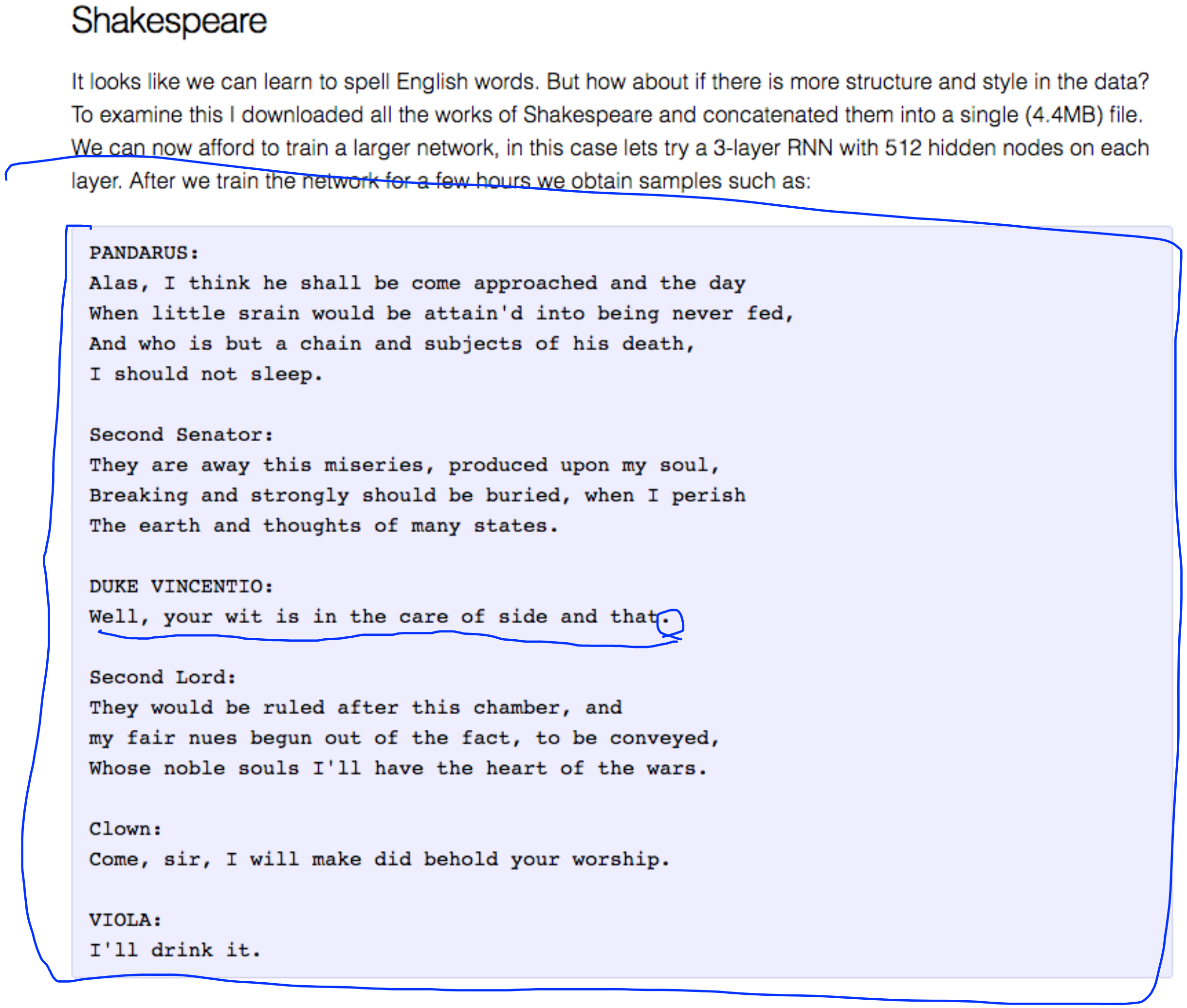
```
(array([2, 0, 2, 1]), ['l', 'h', 'l', 'e'])
(array([2, 0, 2, 1]), ['l', 'h', 'l', 'e'])
(array([2, 0, 2, 1]), ['l', 'h', 'l', 'e'])
(array([2, 0, 2, 1]), ['l', 'h', 'l', 'e'])
(array([2, 0, 2, 1]), ['l', 'h', 'l', 'e'])
(array([2, 0, 2, 1]), ['l', 'h', 'l', 'e'])
(array([2, 2, 2, 3]), ['l', 'l', 'l', 'o'])
(array([2, 2, 2, 3]), ['l', 'l', 'l', 'o'])
(array([1, 2, 2, 3]), ['e', 'l', 'l', 'o'])
(array([1, 2, 2, 3]), ['e', 'l', 'l', 'o'])
(array([1, 2, 2, 3]), ['e', 'l', 'l', 'o'])
```



```
one_cell = rnn_cell.BasicRNNCell(rnn_size)
rnn_cell = rnn_cell.MultiRNNCell([one_cell] * depth)
```

char-rnn

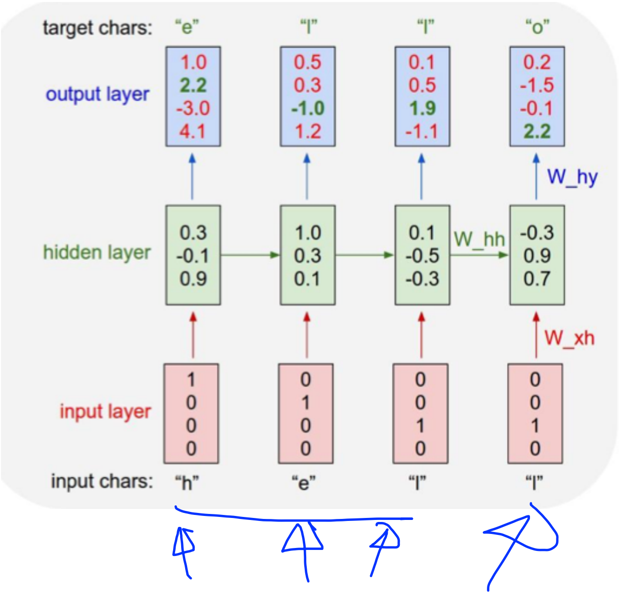


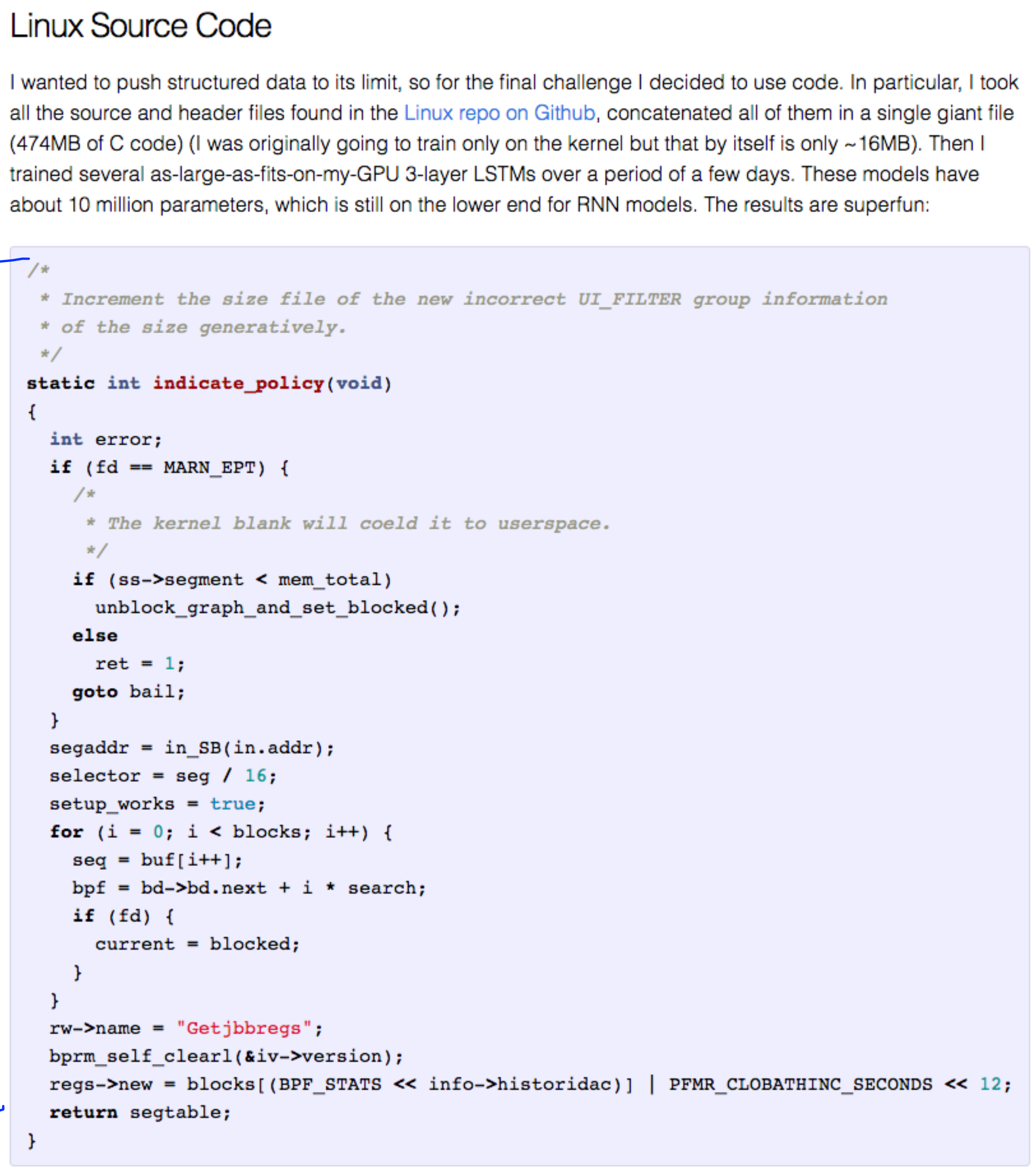


Shakespeare

It looks like we can learn to spell English words. But how about if there is more structure and style in the data? To examine this I downloaded all the works of Shakespeare and concatenated them into a single (4.4MB) file. We can now afford to train a larger network, in this case lets try a 3-layer RNN with 512 hidden nodes on each layer. After we train the network for a few hours we obtain samples such as:

```
PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.

Second Lord:
They would be ruled after this chamber, and
my fair nues begun out of the fact, to be conveyed,
Whose noble souls I'll have the heart of the wars.

Clown:
Come, sir, I will make did behold your worship.

VIOLA:
I'll drink it.
```



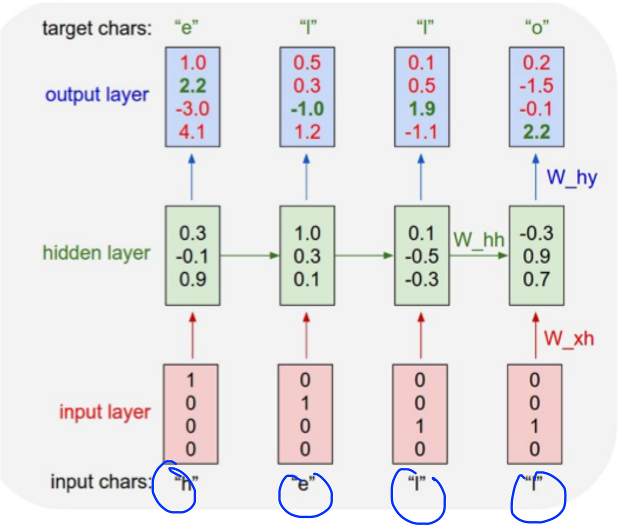


Linux Source Code

I wanted to push structured data to its limit, so for the final challenge I decided to use code. In particular, I took all the source and header files found in the Linux repo on Github, concatenated all of them in a single giant file (474MB of C code) (I was originally going to train only on the kernel but that by itself is only ~16MB). Then I trained several as-large-as-fits-on-my-GPU 3-layer LSTMs over a period of a few days. These models have about 10 million parameters, which is still on the lower end for RNN models. The results are superfun:

```
/*
 * Increment the size file of the new incorrect UI_FILTER group information
 * of the size generatively.
 */
static int indicate_policy(void)
{
  int error;
  if (fd == MARN_EPT) {
    /*
     * The kernel blank will coeld it to userspace.
     */
    if (ss->segment < mem_total)
      unblock_graph_and_set_blocked();
    else
      ret = 1;
    goto bail;
  }
  segaddr = in_SB(in.addr);
  selector = seg / 16;
  setup_works = true;
  for (i = 0; i < blocks; i++) {
    seq = buf[i++];
    bpf = bd->bd.next + i * search;
    if (fd) {
      current = blocked;
    }
  }
  rw->name = "Getjbbregs";
  bprm_self_clearl(&iv->version);
  regs->new = blocks[(BPF_STATS << info->historidac)] | PFMR_CLOBATHINC_SECONDS << 12;
  return segtable;
}
```

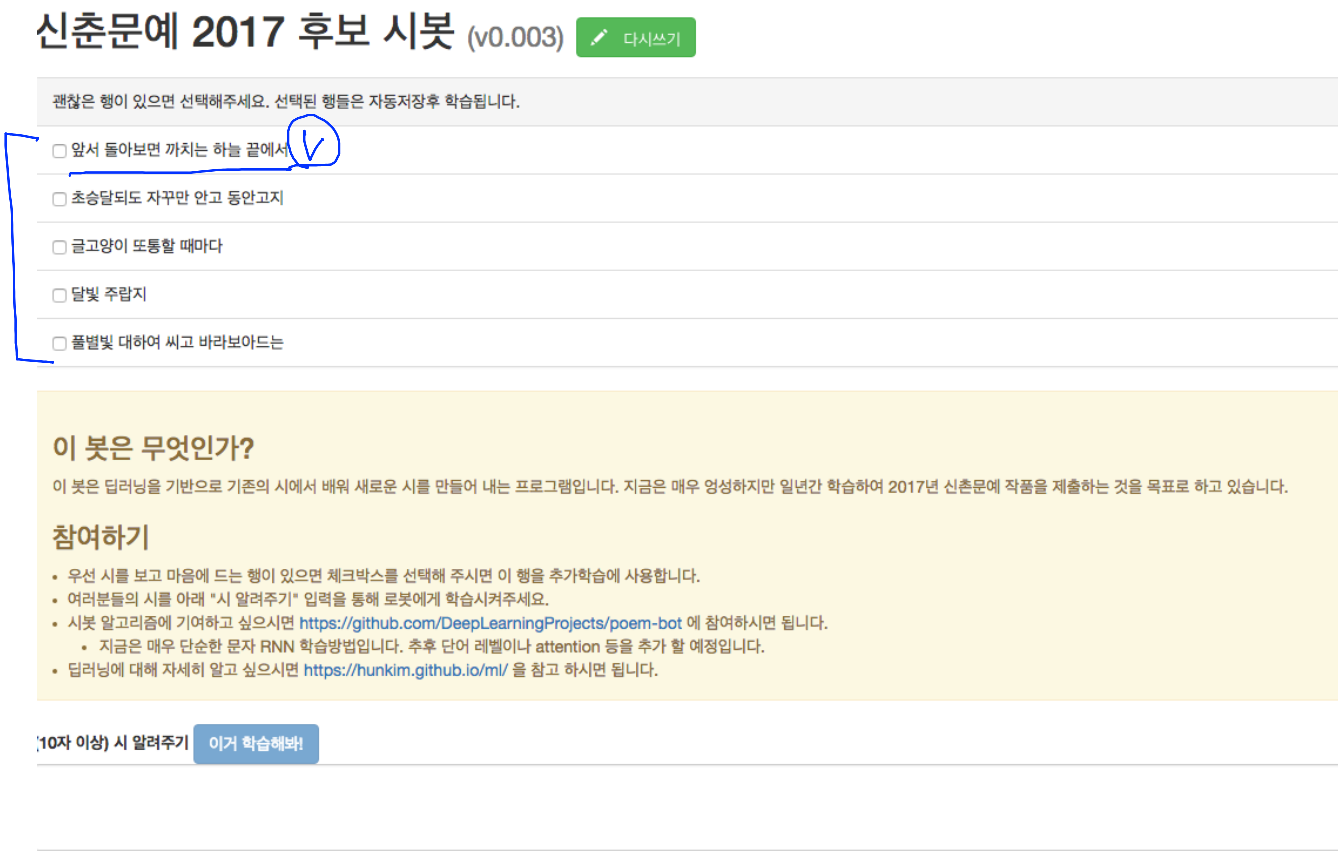
char/word rnn (char/word level n to n model)



https://github.com/sherjilozair/char-rnn-tensorflow

https://github.com/hunkim/word-rnn-tensorflow

bot.wpoem.com



신춘문예 2017 후보 시봇 (v0.003)

괜찮은 행이 있으면 선택해주세요. 선택된 행들은 자동저장후 학습됩니다.

- ☐ 앞서 돌아보면 까치는 하늘 끝에서 ✓
- ☐ 초승달되도 자꾸만 안고 동안고지
- ☐ 글고양이 또통할 때마다
- ☐ 달빛 주랍지
- ☐ 풀별빛 대하여 씨고 바라보아드는

이 봇은 무엇인가?

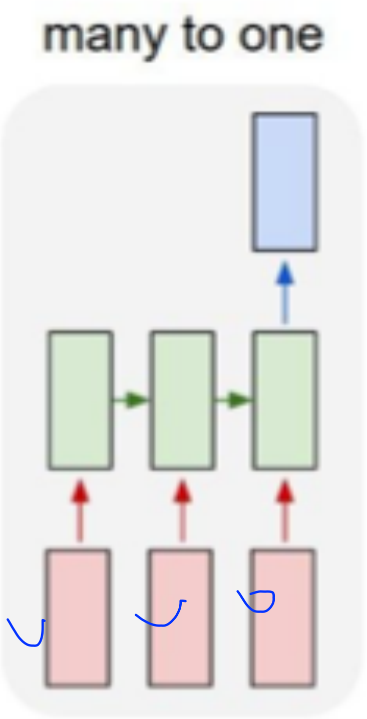
이 봇은 딥러닝을 기반으로 기존의 시에서 배워 새로운 시를 만들어 내는 프로그램입니다. 지금은 매우 엉성하지만 일년간 학습하여 2017년 신춘문예 작품을 제출하는 것을 목표로 하고 있습니다.

참여하기

- 우선 시를 보고 마음에 드는 행이 있으면 체크박스를 선택해 주시면 이 행을 추가학습에 사용합니다.
- 여러분들의 시를 아래 "시 알려주기" 입력을 통해 로봇에게 학습시켜주세요.
- 시봇 알고리즘에 기여하고 싶으시면 https://github.com/DeepLearningProjects/poem-bot 에 참여하시면 됩니다.
  - 지금은 매우 단순한 문자 RNN 학습방법입니다. 추후 단어 레벨이나 attention 등을 추가 할 예정입니다.
- 딥러닝에 대해 자세히 알고 싶으시면 https://hunkim.github.io/ml/ 을 참고 하시면 됩니다.

(10자 이상) 시 알려주기

Many to one



https://github.com/nlintz/TensorFlow-Tutorials/blob/master/7_lstm.py

RNN applications

- Language Modeling
- Speech Recognition
- Machine Translation
- Conversation Modeling/Question Answering
- Image/Video Captioning
- Image/Music/Dance Generation

http://jiwonkim.org/awesome-rnn/

TensorFlow GPU @AWS

Sung Kim <hunkim+ml@gmail.com>
http://hunkim.github.io/ml/