8. RNN/LSTM(1)

AILab
Hanyang Univ.

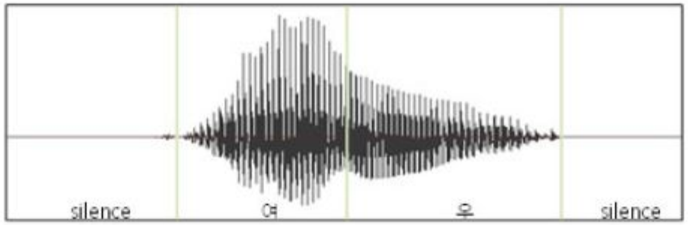
오늘 실습 내용

- RNN 이론
- RNN basics
- Implementing RNN

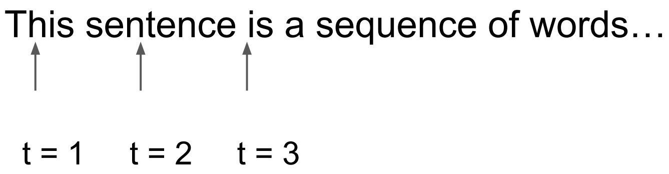
RNN 이론

기존 신경망의 한계 : Sequence data를 처리하기 어려움

Sequence data : 데이터 집합 내의 객체들이 어떤 순서를 가진 데이터 (ex: 음성신호, 자연어 문장)



[그림2] "여우" 음성의 웨이브 형태
(출처: "인공지능 기반의 음성인식 기술개발동향과 도입방안 및 전략 세미나," 서강대 김지환 교수, Mar. 2017)



This sentence is a sequence of words…

t = 1 t = 2 t = 3

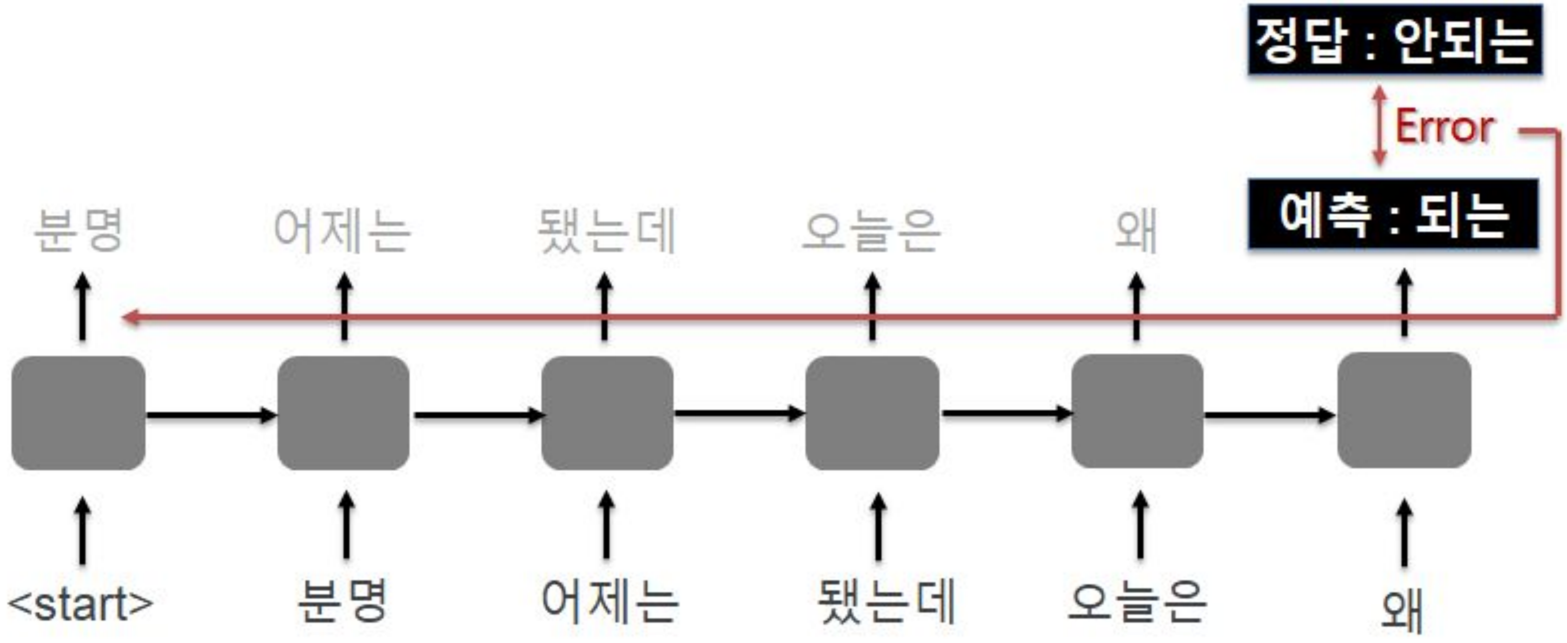
Sequence data 를 처리하기 위해 RNN은?

: 이전 출력값이 현재 결과에 영향을 미침

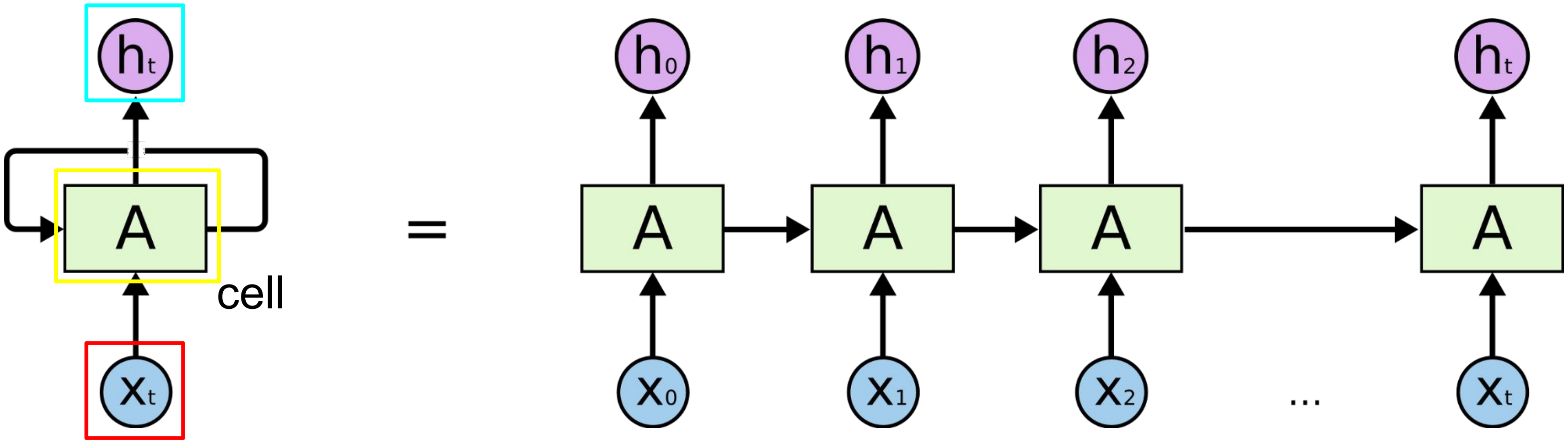
RNN example

Sequence data 를 딥러닝으로 처리하려면?

: 이전 출력값이 현재 결과에 영향을 미치는 RNN 구조 활용



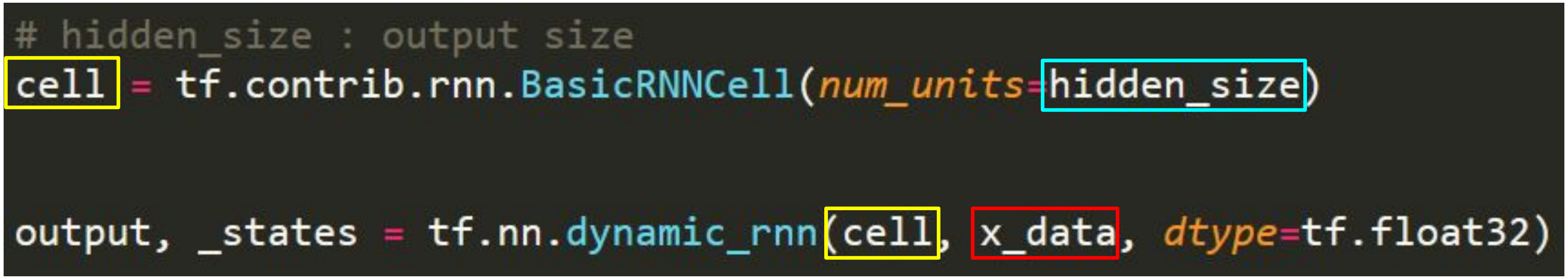
RNN basics



cell이란? : 각각의 상태(state)에서의 RNN 모델을 의미함

Tensorflow로 RNN 구현하기

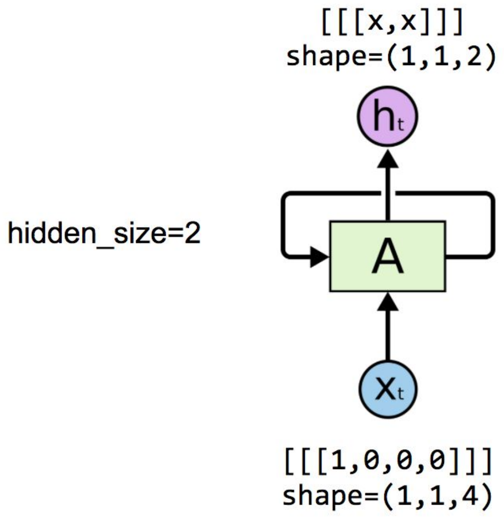
1. cell 생성
2. cell 구동

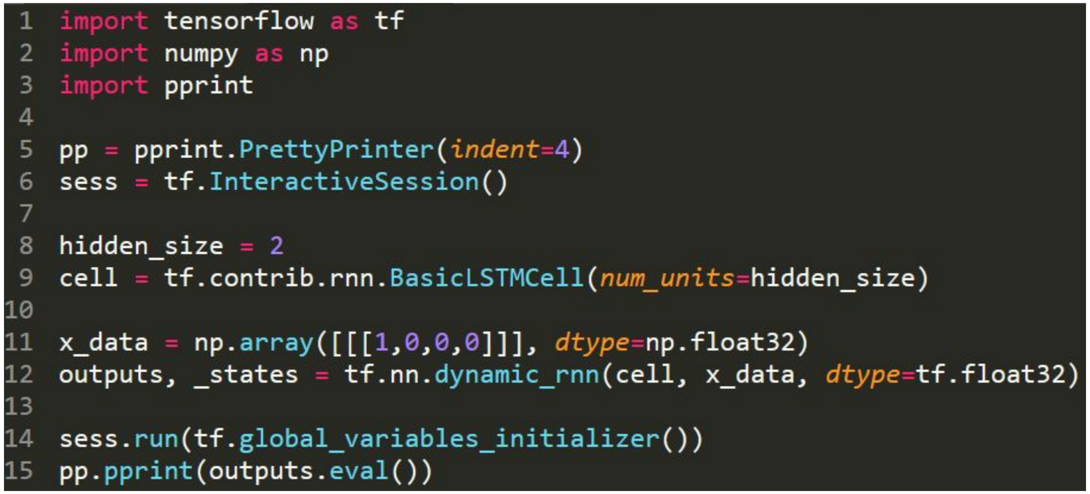


```
# hidden_size : output size
cell = tf.contrib.rnn.BasicRNNCell(num_units=hidden_size)


output, _states = tf.nn.dynamic_rnn(cell, x_data, dtype=tf.float32)
```

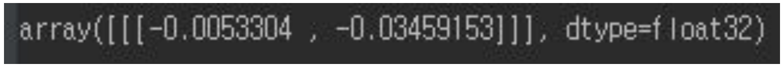
RNN basics

One node : 4(input-dimension) in 2 (hidden-size)





```
import tensorflow as tf
import numpy as np
import pprint

pp = pprint.PrettyPrinter(indent=4)
sess = tf.InteractiveSession()

hidden_size = 2
cell = tf.contrib.rnn.BasicLSTMCell(num_units=hidden_size)

x_data = np.array([[[1,0,0,0]]], dtype=np.float32)
outputs, _states = tf.nn.dynamic_rnn(cell, x_data, dtype=tf.float32)

sess.run(tf.global_variables_initializer())
pp.pprint(outputs.eval())
```



```
array([[[-0.0053304 , -0.03459153]]], dtype=float32)
```

hidden_size = output dimension
: 각 RNN cell 을 거치고 나온 output을 표현하는 벡터의 차원을 의미함 (하이퍼파라미터)

hidden_size = 2 : output shape=(1,1,2), states shape = (1,2)
hidden_size = 3 : output shape=(1,1,3), states shape = (1,3)

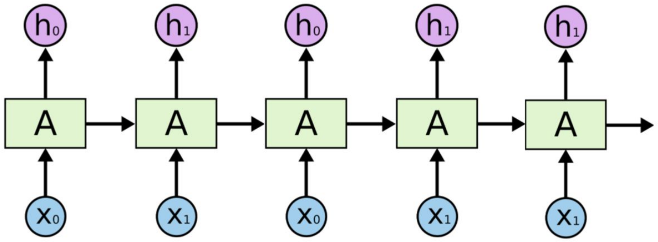
RNN basics

hidden_size=2
sequance_length=5

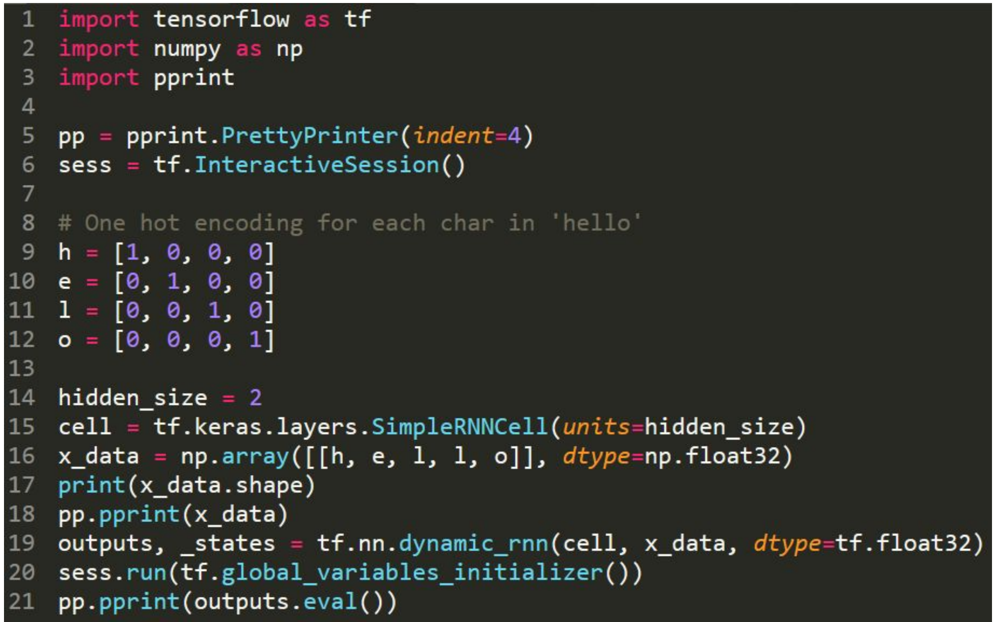
Using word vector

```
# One hot encoding
h = [1, 0, 0, 0]
e = [0, 1, 0, 0]
l = [0, 0, 1, 0]
o = [0, 0, 0, 1]
```

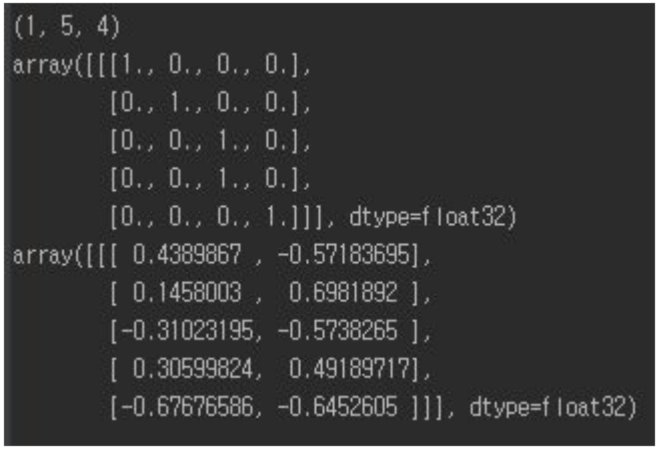
shape=(1,5,2): [[[x,x], [x,x], [x,x], [x,x], [x,x]]]



```
shape=(1,5,4): [[[1,0,0,0], [0,1,0,0], [0,0,1,0], [0,0,1,0], [0,0,0,1]]]
                     h          e          l          l          o
```



```python
import tensorflow as tf
import numpy as np
import pprint

pp = pprint.PrettyPrinter(indent=4)
sess = tf.InteractiveSession()

# One hot encoding for each char in 'hello'
h = [1, 0, 0, 0]
e = [0, 1, 0, 0]
l = [0, 0, 1, 0]
o = [0, 0, 0, 1]

hidden_size = 2
cell = tf.keras.layers.SimpleRNNCell(units=hidden_size)
x_data = np.array([[h, e, l, l, o]], dtype=np.float32)
print(x_data.shape)
pp.pprint(x_data)
outputs, _states = tf.nn.dynamic_rnn(cell, x_data, dtype=tf.float32)
sess.run(tf.global_variables_initializer())
pp.pprint(outputs.eval())
```



```
(1, 5, 4)
array([[[1., 0., 0., 0.],
        [0., 1., 0., 0.],
        [0., 0., 1., 0.],
        [0., 0., 1., 0.],
        [0., 0., 0., 1.]]], dtype=float32)
array([[[ 0.4389867 , -0.57183695],
        [ 0.1458003 ,  0.6981892 ],
        [-0.31023195, -0.5738265 ],
        [ 0.30599824,  0.49189717],
        [-0.67676586, -0.6452605 ]]], dtype=float32)
```

hunkim/DeepLearningZeroToAll

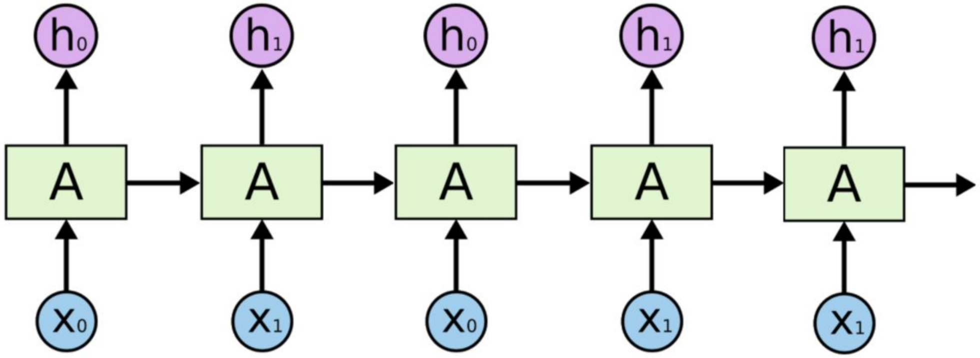
RNN basics

hidden_size=2
sequacne_length=5
batch = 3

Using word vector

```
# One hot encoding
h = [1, 0, 0, 0]
e = [0, 1, 0, 0]
l = [0, 0, 1, 0]
o = [0, 0, 0, 1]
```

```
shape=(3,5,2): [[[x,x], [x,x], [x,x], [x,x], [x,x]],
                [[x,x], [x,x], [x,x], [x,x], [x,x]],
                [[x,x], [x,x], [x,x], [x,x], [x,x]]]
```



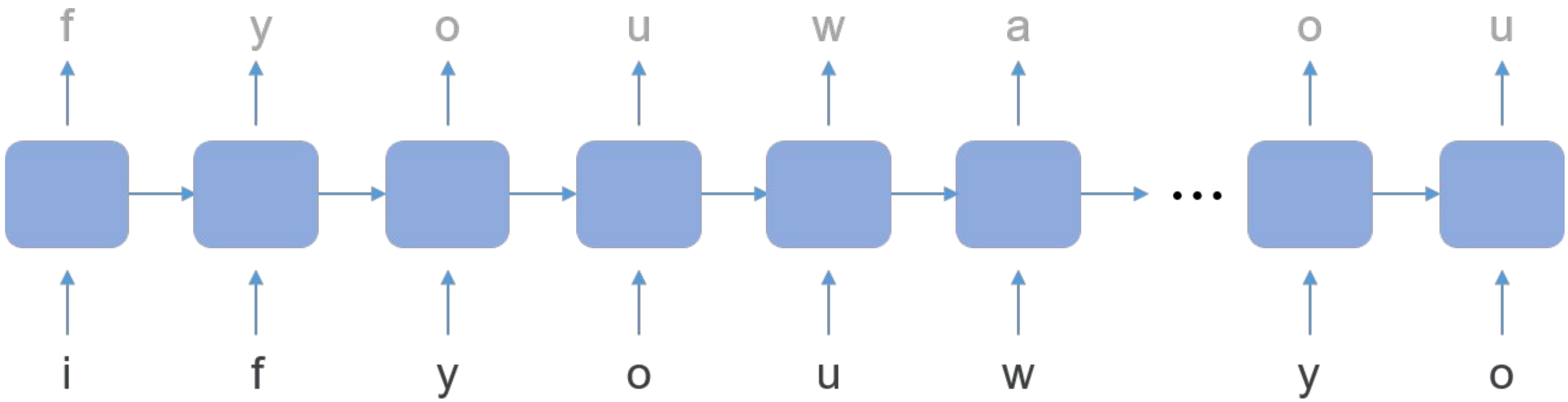
```
shape=(3,5,4): [[[1,0,0,0], [0,1,0,0], [0,0,1,0], [0,0,1,0], [0,0,0,1]], # hello
                [[0,1,0,0], [0,0,0,1], [0,0,1,0], [0,0,1,0], [0,0,1,0]]  # eolll
                [[0,0,1,0], [0,0,1,0], [0,1,0,0], [0,1,0,0], [0,0,1,0]]] # lleel
```

hunkim/DeepLearningZeroToAll

character sequence RNN

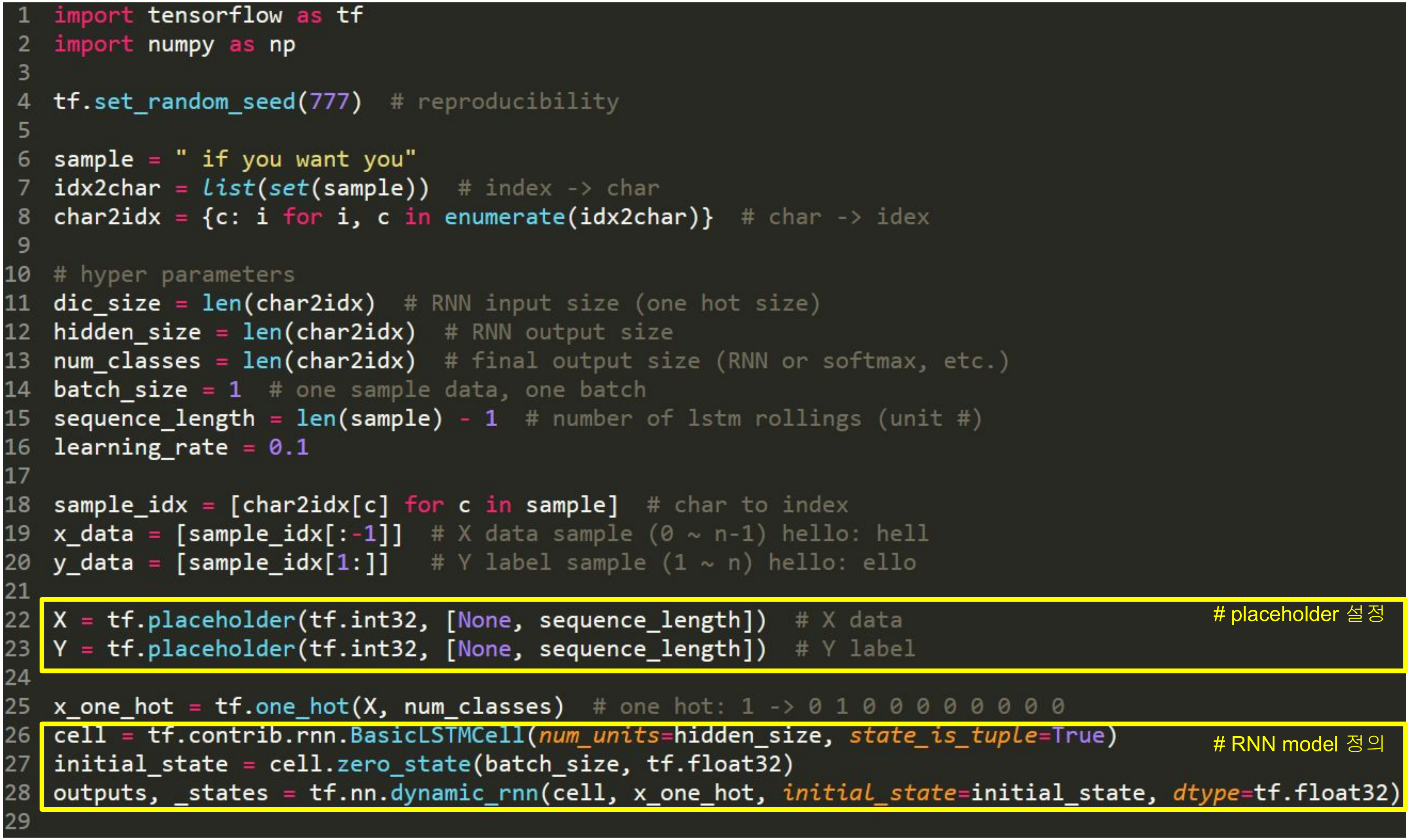
Character 단위의 RNN 실습

“if you want yo”를 input값(x_data)으로 했을때 “f you want you”가 나오게 하자



hunkim/DeepLearningZeroToAll

character sequence RNN



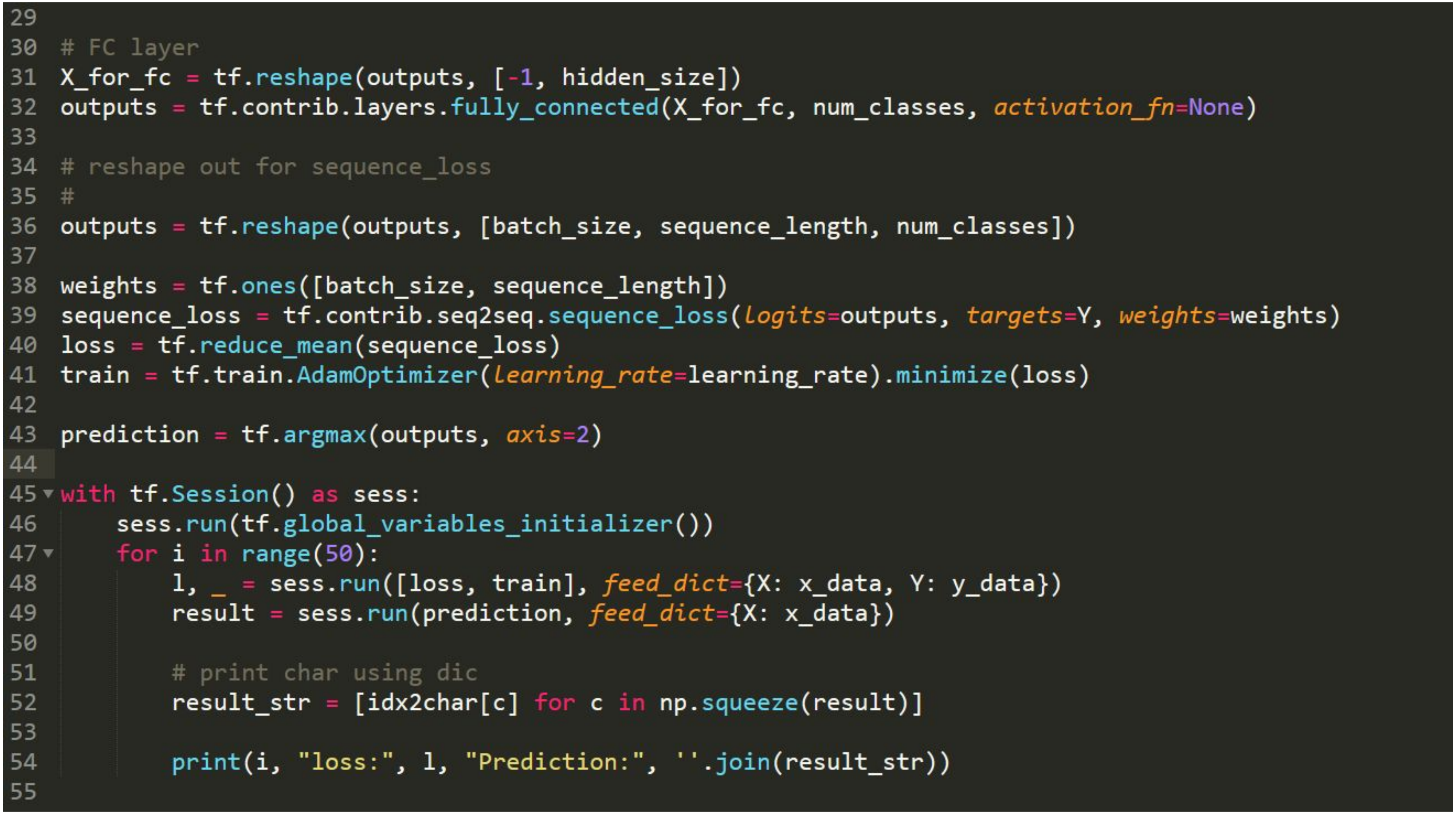
```
import tensorflow as tf
import numpy as np

tf.set_random_seed(777)  # reproducibility

sample = " if you want you"
idx2char = list(set(sample))  # index -> char
char2idx = {c: i for i, c in enumerate(idx2char)}  # char -> idex

# hyper parameters
dic_size = len(char2idx)  # RNN input size (one hot size)
hidden_size = len(char2idx)  # RNN output size
num_classes = len(char2idx)  # final output size (RNN or softmax, etc.)
batch_size = 1  # one sample data, one batch
sequence_length = len(sample) - 1  # number of lstm rollings (unit #)
learning_rate = 0.1

sample_idx = [char2idx[c] for c in sample]  # char to index
x_data = [sample_idx[:-1]]  # X data sample (0 ~ n-1) hello: hell
y_data = [sample_idx[1:]]   # Y label sample (1 ~ n) hello: ello

X = tf.placeholder(tf.int32, [None, sequence_length])  # X data
Y = tf.placeholder(tf.int32, [None, sequence_length])  # Y label

x_one_hot = tf.one_hot(X, num_classes)  # one hot: 1 -> 0 1 0 0 0 0 0 0 0 0
cell = tf.contrib.rnn.BasicLSTMCell(num_units=hidden_size, state_is_tuple=True)
initial_state = cell.zero_state(batch_size, tf.float32)
outputs, _states = tf.nn.dynamic_rnn(cell, x_one_hot, initial_state=initial_state, dtype=tf.float32)

```

# placeholder 설정

# RNN model 정의

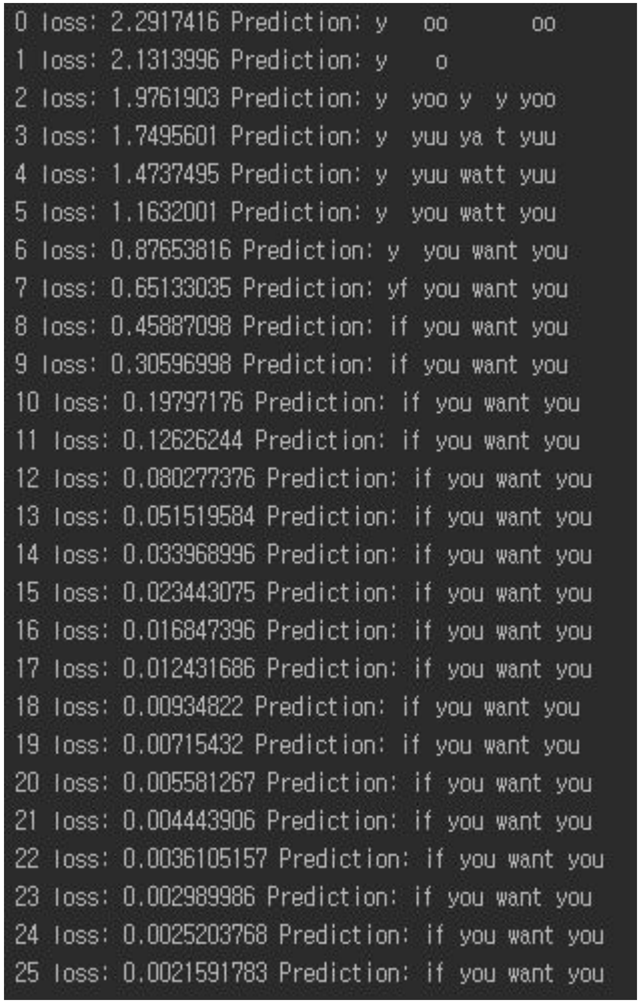
hunkim/DeepLearningZeroToAll

character sequence RNN

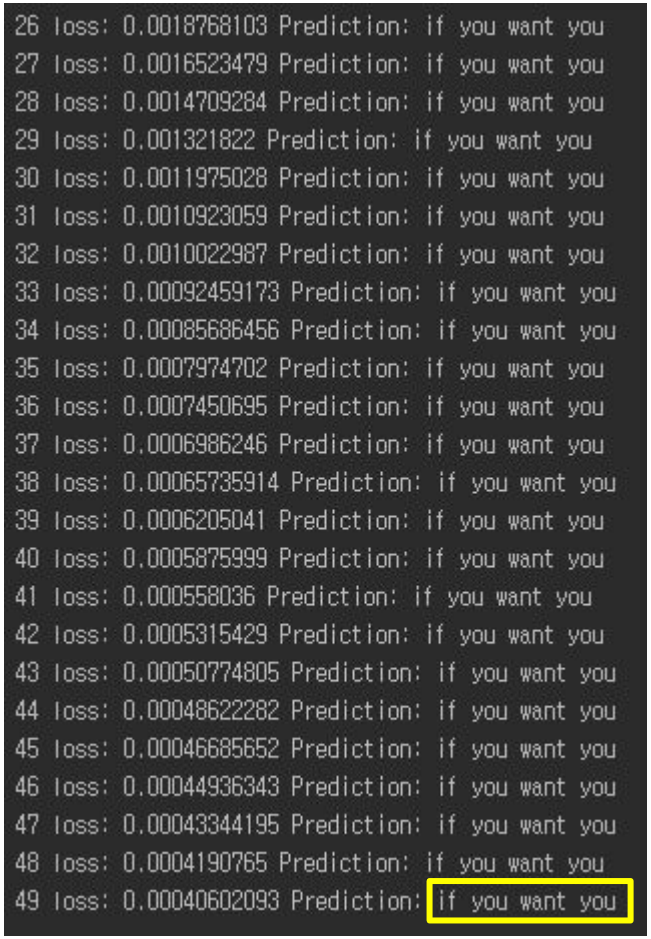


```

# FC layer
X_for_fc = tf.reshape(outputs, [-1, hidden_size])
outputs = tf.contrib.layers.fully_connected(X_for_fc, num_classes, activation_fn=None)

# reshape out for sequence_loss
#
outputs = tf.reshape(outputs, [batch_size, sequence_length, num_classes])

weights = tf.ones([batch_size, sequence_length])
sequence_loss = tf.contrib.seq2seq.sequence_loss(logits=outputs, targets=Y, weights=weights)
loss = tf.reduce_mean(sequence_loss)
train = tf.train.AdamOptimizer(learning_rate=learning_rate).minimize(loss)

prediction = tf.argmax(outputs, axis=2)

with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    for i in range(50):
        l, _ = sess.run([loss, train], feed_dict={X: x_data, Y: y_data})
        result = sess.run(prediction, feed_dict={X: x_data})

        # print char using dic
        result_str = [idx2char[c] for c in np.squeeze(result)]

        print(i, "loss:", l, "Prediction:", ''.join(result_str))

```

실행 결과



```
0 loss: 2.2917416 Prediction: y   oo       oo
1 loss: 2.1313996 Prediction: y    o
2 loss: 1.9761903 Prediction: y  yoo y  y yoo
3 loss: 1.7495601 Prediction: y  yuu ya t yuu
4 loss: 1.4737495 Prediction: y  yuu watt yuu
5 loss: 1.1632001 Prediction: y  you watt you
6 loss: 0.87653816 Prediction: y  you want you
7 loss: 0.65133035 Prediction: yf you want you
8 loss: 0.45887098 Prediction: if you want you
9 loss: 0.30596998 Prediction: if you want you
10 loss: 0.19797176 Prediction: if you want you
11 loss: 0.12626244 Prediction: if you want you
12 loss: 0.080277376 Prediction: if you want you
13 loss: 0.051519584 Prediction: if you want you
14 loss: 0.033968996 Prediction: if you want you
15 loss: 0.023443075 Prediction: if you want you
16 loss: 0.016847396 Prediction: if you want you
17 loss: 0.012431686 Prediction: if you want you
18 loss: 0.00934822 Prediction: if you want you
19 loss: 0.00715432 Prediction: if you want you
20 loss: 0.005581267 Prediction: if you want you
21 loss: 0.004443906 Prediction: if you want you
22 loss: 0.0036105157 Prediction: if you want you
23 loss: 0.002989986 Prediction: if you want you
24 loss: 0.0025203768 Prediction: if you want you
25 loss: 0.0021591783 Prediction: if you want you
```



```
26 loss: 0.0018768103 Prediction: if you want you
27 loss: 0.0016523479 Prediction: if you want you
28 loss: 0.0014709284 Prediction: if you want you
29 loss: 0.001321822 Prediction: if you want you
30 loss: 0.0011975028 Prediction: if you want you
31 loss: 0.0010923059 Prediction: if you want you
32 loss: 0.0010022987 Prediction: if you want you
33 loss: 0.00092459173 Prediction: if you want you
34 loss: 0.00085686456 Prediction: if you want you
35 loss: 0.0007974702 Prediction: if you want you
36 loss: 0.0007450695 Prediction: if you want you
37 loss: 0.0006986246 Prediction: if you want you
38 loss: 0.00065735914 Prediction: if you want you
39 loss: 0.0006205041 Prediction: if you want you
40 loss: 0.0005875999 Prediction: if you want you
41 loss: 0.000558036 Prediction: if you want you
42 loss: 0.0005315429 Prediction: if you want you
43 loss: 0.00050774805 Prediction: if you want you
44 loss: 0.00048622282 Prediction: if you want you
45 loss: 0.00046685652 Prediction: if you want you
46 loss: 0.00044936343 Prediction: if you want you
47 loss: 0.00043344195 Prediction: if you want you
48 loss: 0.0004190765 Prediction: if you want you
49 loss: 0.00040602093 Prediction: if you want you
```