메디치소프트 기술연구소

2018.08.06

딥러닝 6일

. Joblib

. Joblib의 dump&load를 이용하여 scaler, model저장 읽기
. 제약사항: python 2에서 저장한 pickle파일은 python2에서만 읽을 수 있다.

[저장할 때]

```
from sklearn.externals import joblib
# 객체를 pickled binary file 형태로 저장한다
file_name = 'object_01.pkl'
joblib.dump(obj, file_name)
```

[읽을 때]

```
from sklearn.externals import joblib
# pickled binary file 형태로 저장된 객체를 로딩한다
file_name = 'object_01.pkl'
obj = joblib.load(file_name)
```

. Joblib -실습

. 데이터 파일과 scaler객체 각각 저장

```
from sklearn import datasets
from sklearn.externals import joblib
import pickle

iris = datasets.load_iris()
X=iris.data
y=iris.target
from sklearn.model_selection import train_test_split
train_X, test_X, train_Y, test_Y = train_test_split(X, y ,test_size=0.3, random_state=0)

pickle.dump(train_X,open('train_X.pkl','wb'))
pickle.dump(test_X,open('test_X.pkl','wb'))
pickle.dump(train_Y,open('train_Y.pkl','wb'))
pickle.dump(train_Y,open('test_Y.pkl','wb'))
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaler.fit(train_X)
train_x_scaled = scaler.transform(train_X)
print(train_x_scaled[:5])

file_name = 'scaler_01.pkl'
joblib.dump(scaler,file_name)
```

. Joblib -실습

. 저장된 데이터와 scaler이용

```
from sklearn import datasets
from sklearn.externals import joblib
import pickle
from sklearn.preprocessing import MinMaxScaler

train_X= pickle.load(open('train_X.pkl','rb'))
test_X=pickle.load(open('test_X.pkl','rb'))
train_Y=pickle.load(open('train_Y.pkl','rb'))
train_Y=pickle.load(open('test_Y.pkl','rb'))

file_name = 'scaler_01.pkl'
scaler = joblib.load(file_name)
train_x_scaled = scaler.transform(train_X)
print(train_x_scaled[:5])
```

. TensorFlow 모델 저장 불러오기-Saver

. 모델을 훈련 후 재사용하기 위하여 모델 파라미터를 디스크에 저장
. 훈련하는 동안 일정한 간격으로 체크 포인트를 저장해 두면 컴퓨터가 훈련 중간에 문제를 일으켜도 처음부터 시작하지 않고 마지막 체크포인트부터 시작가능
. 모든 변수 노드를 생성한 후 Saver 노드 추가
. 실행단계에서 모델을 저장=>save()메서드에 세션과 체크포인트 파일의 경로 전달 호출

```
with tf.Session() as sess:
    sess.run(init)

    for epoch in range(n_epochs):
        if epoch % 100 == 0:
            print("에포크", epoch, "MSE =", mse.eval())  #100번의 에포크마다 체크포인트 저장
            save_path = saver.save(sess, "./tmp/my_model.ckpt")
        sess.run(training_op)

    best_theta = theta.eval()
    save_path = saver.save(sess, "./tmp/my_model_final.ckpt")
```

. TensorFlow 모델 저장

a) Meta graph

Tensorflow graph를 저장 하게 된다. 즉 all variables, operations, collections 등을 저장 한다. `.meta` 로 확장자를 가진다.

b) Checkpoint file

binary 파일로 weights, biases, gradients 등을 저장 한다.

`0.11` 부터는 두개의 파일로 저장된다.

- `model.ckpt.data-00000-of-00001`
- `model.ckpt.index`

`.data` 파일의 경우 training variable를 가지고 있다.

여전히 `checkpoint` 파일도 보유하고 있지만 이것은 단순히 최근 상태만을 기록하고 있다.

. TensorFlow 모델 읽기

a) 네트워크 생성

`.meta` 파일을 생성 했으므로 이것을 불러오는 방식으로 network을 재생성 할 수 있다.

`.meta` 파일을 불러오기 위해서는 `tf.train.import()` 함수를 이용한다.

```
saver = tf.train.import_meta_graph('my_test_model-1000.meta')
```

이렇게 하면 현재 그래프에 이어 붙는 형식으로 동작하므로 `tf.reset_default_graph()` 를 실행해서 default graph로 초기화 해주는 것이 안전하다.

b) 파라메터 로딩

`tf.train.Saver()` 를 이용해서 파라메터를 로딩한다.

```
with tf.Session() as sess:
  new_saver = tf.train.import_meta_graph('my_test_model-1000.meta')
  new_saver.restore(sess, tf.train.latest_checkpoint('./'))
```

```
with tf.Session() as sess:
    saver = tf.train.import_meta_graph('my-model-1000.meta')
    saver.restore(sess, tf.train.latest_checkpoint('./'))
    print(sess.run('w1:0'))
##Model has been restored. Above statement will print the saved value of w1.
```

. TensorFlow 모델 저장

```
import tensorflow as tf
# Prepare to feed input, i.e. feed_dict and placeholders
w1 = tf.placeholder(tf.float32, name="w1")
w2 = tf.placeholder(tf.float32, name="w2")
b1 = tf.Variable(2.0,dtype=tf.float32, name="bias")
feed_dict = {'w1': 4.0, 'w2': 8.0}

# Define a test operation that we will restore
w3 = w1 + w2
w4 = tf.multiply(w3, b1, name="op_to_restore")
sess = tf.Session()
sess.run(tf.global_variables_initializer())

# Create a saver object which will save all the variables
saver = tf.train.Saver()

# Run the operation by feeding input
result = sess.run(w4, {w1:feed_dict['w1'], w2:feed_dict['w2']})
print(result)
# Prints 24 which is sum of (w1+w2)*b1

# Now, save the graph
saver.save(sess, './my_test_model', global_step=1000)
```

. TensorFlow 모델 읽기

```
import tensorflow as tf

sess=tf.Session()
#First let's load meta graph and restore weights
saver = tf.train.import_meta_graph('my_test_model-1000.meta')
saver.restore(sess,tf.train.latest_checkpoint('./'))
# Now, let's access and create placeholders variables and
# create feed-dict to feed new data
graph = tf.get_default_graph()
w1 = graph.get_tensor_by_name("w1:0")
w2 = graph.get_tensor_by_name("w2:0")
feed_dict ={w1:13.0,w2:17.0}

#Now, access the op that you want to run.
op_to_restore = graph.get_tensor_by_name("op_to_restore:0")

print (sess.run(op_to_restore,feed_dict))
#This will print 60 which is calculated
#using new values of w1 and w2 and saved value of b1.
```

. 머신러닝 프로젝트(하우징)

```
import matplotlib.pyplot as plt
import pandas as pd
import os
import tarfile
from six.moves import urllib
plt.rcParams['axes.labelsize'] = 14
plt.rcParams['xtick.labelsize'] = 12
plt.rcParams['ytick.labelsize'] = 12
# 한글출력
plt.rcParams['axes.unicode_minus'] = False

# 그림을 저장할 폴드
PROJECT_ROOT_DIR = "."
CHAPTER_ID = "end_to_end_project"
IMAGES_PATH = os.path.join(PROJECT_ROOT_DIR, "images", CHAPTER_ID)
DOWNLOAD_ROOT = "https://raw.githubusercontent.com/ageron/handson-ml/master/"
HOUSING_PATH = os.path.join("datasets", "housing")
HOUSING_URL = DOWNLOAD_ROOT + "datasets/housing/housing.tgz"
```

. 머신러닝 프로젝트(하우징)

```
def save_fig(fig_id, tight_layout=True, fig_extension="png", resolution=300):
    path = os.path.join(IMAGES_PATH, fig_id + "." + fig_extension)
    if tight_layout:
        plt.tight_layout()
    plt.savefig(path, format=fig_extension, dpi=resolution)
def fetch_housing_data(housing_url=HOUSING_URL, housing_path=HOUSING_PATH):
    if not os.path.isdir(housing_path):
     os.makedirs(housing_path)
    tgz_path = os.path.join(housing_path, "housing.tgz")
    urllib.request.urlretrieve(housing_url, tgz_path)
    housing_tgz = tarfile.open(tgz_path)
    housing_tgz.extractall(path=housing_path)
    housing_tgz.close()
fetch_housing_data()
def load_housing_data(housing_path=HOUSING_PATH):
    csv_path = os.path.join(housing_path, "housing.csv")
    return pd.read_csv(csv_path)
```

. 머신러닝 프로젝트(하우징)

```
housing = load_housing_data()
print(housing.head())
print(housing.info())
print(housing["ocean_proximity"].value_counts())
print(housing.describe())
housing.hist(bins=50, figsize=(20,15))
save_fig("attribute_histogram_plots")
plt.show()
```

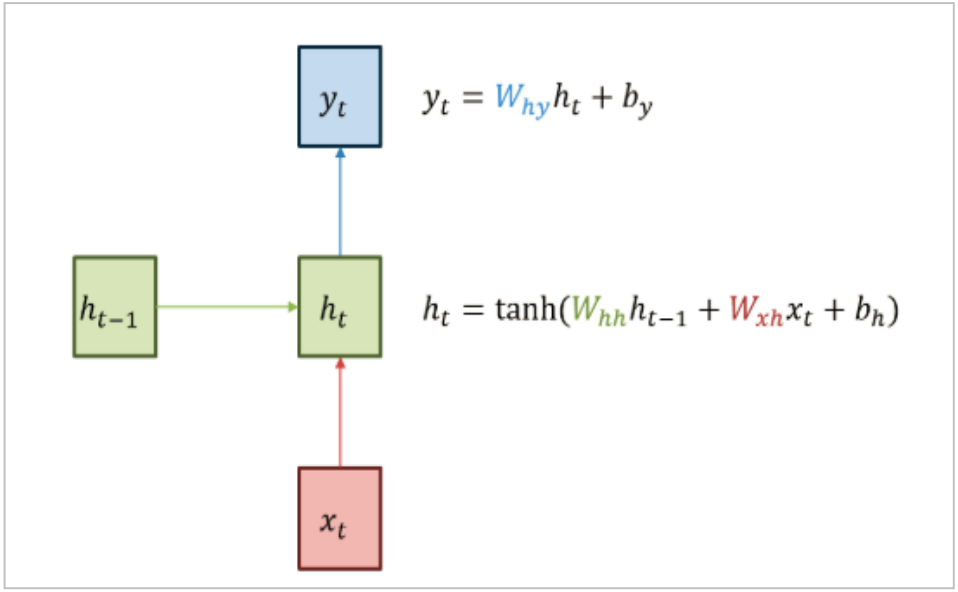
. 특성은 10개의 컬럼
. 20,640개의 샘플
. Total_bedrooms 특성은 20,433개만 널값이 아니다.
. Ocean_proximit필드만 빼고 모든 특성이 숫자형

RNN(Recurrent Neural Network)

. RNN은 다양한 자연어처리(NLP) 문제에 대해 뛰어난 성능을 보이고 있는 인기있는 모델.
- 실제 세상에서 어떤 임의의 문장이 존재할 확률이 어느 정도인지에 대한 스코어를 매길 수 있다는 점이다. 이는 문장이 문법적으로나 의미적으로 어느 정도 올바른지 측정할 수 있도록 해주고, 보통 자동 번역 시스템의 일부로 활용
. RNN에 대한 기본적인 아이디어는 순차적인 정보를 처리
. 기존의 신경망 구조에서는 모든 입력(과 출력)이 각각 독립적이라고 가정
. 한 예로, 문장에서 다음에 나올 단어를 추측하고 싶다면 이전에 나온 단어들을 아는 것이 큰 도움이 될 것이다.

RNN(Recurrent Neural Network)

. Hidden Node가 방향을 가진 엣지로 연결돼 순환구조를 이루는(Directed Cycle) 인공신경망의 한 종류
. 음성, 문자 등 순차적으로 등장하는 데이터에 대한 처리에 적합한 모델
. Sequence 길이에 관계없이 input과 output을 받아들일 수 있는 네트워크 구조



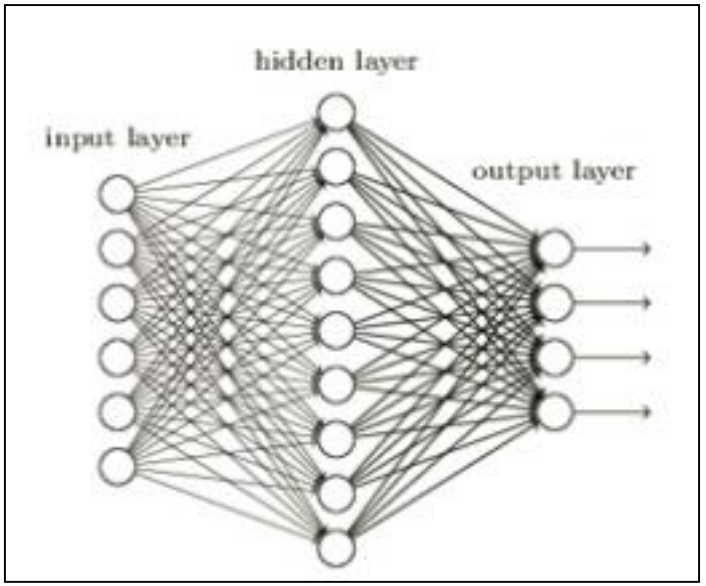
.그림 **RNN의 구조**

. 녹색 박스: hidden state
. 빨간 박스: input x
. 파란 박스: output y

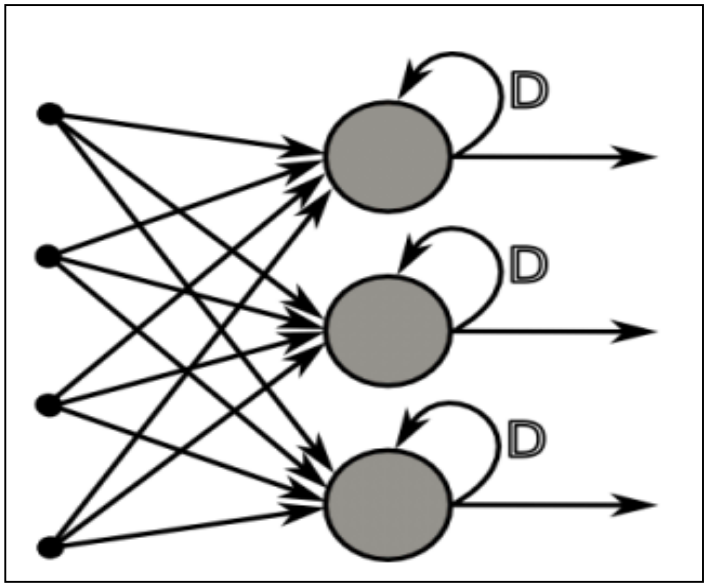
- 현재 상태의 hidden state ht는 직전 시점의 hidden state ht-1를 받아서 갱신
- 현재 상태의 output Yt는 Ht를 전달받아 갱신되는 구조

※ hidden state의 활성함수(activation function)은 비선형 함수인 하이퍼볼릭탄젠트(tanh)이다.

RNN(Recurrent Neural Network)



VS



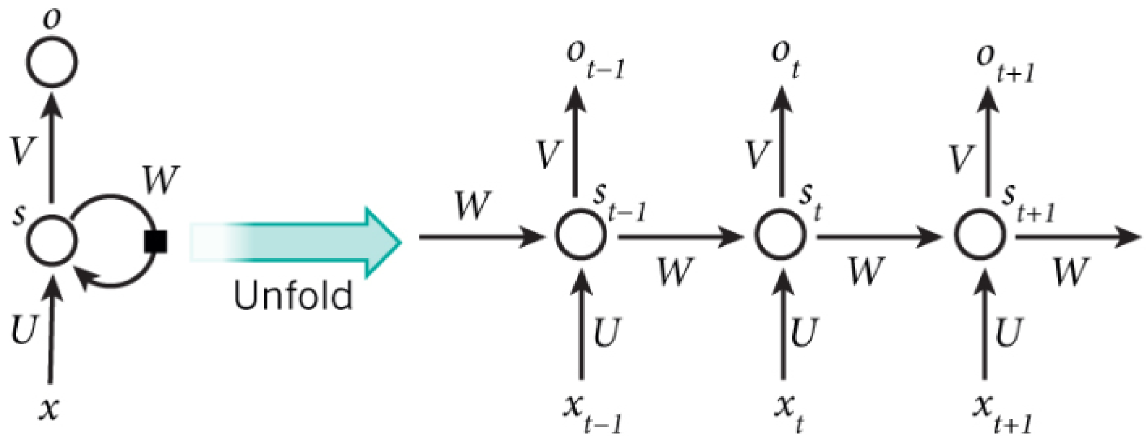
[그림] 기존의 신경망 구조

[그림] RNN 신경망 구조

. 과거 자신의 정보(가중치)기억하고 이를 학습에 반영

RNN의 구조

메모리 형태: 타임 스텝 t에서 순환 뉴런의 출력은 이전 타임 스텝의 모든 입력에 대한 함수
메모리 셀: 타임 스텝에 걸쳐서 어떤 상태에 보존하는 신경망의 구성 요소



. RNN의 recurrent한 연결 펼쳐진 그림(전체 시퀀스)
. 우리가 관심있는 시퀀스 정보가 5개의 단어로 이루어진 문장이라면, RNN 네트워크는 한 단어당 하나의 layer씩 (recurrent 연결이 없는, 또는 사이클이 없는) 5-layer 신경망 구조로 펼쳐질 것이다.

- x_t는 시간 스텝(time step) t에서의 입력값이다.
- s_t는 시간 스텝 t에서의 hidden state이다. 네트워크의 "메모리" 부분으로서, 이전 시간 스텝의 hiddent state 값과 현재 시간 스텝의 입력값에 의해 계산된다: $s_t = f(Ux_t + Ws_{t-1})$. 비선형 함수 f는 보통 tanh나 ReLU가 사용되고, 첫 hidden state를 계산하기 위한 s_{-1}은 보통 0으로 초기화시킨다.
- o_t는 시간 스텝 t에서의 출력값이다. 예를 들어, 문장에서 다음 단어를 추측하고 싶다면 단어 수만큼의 차원의 확률 벡터가 될 것이다. $o_t = \mathrm{softmax}(Vs_t)$

RNN의 예시

- RNN은 많은 자연어 처리 문제에서 성공적으로 적용
- 현재 전세계에서 RNN의 여러 종류 중에서 가장 많이 사용되는 것은 LSTM으로, 기본 RNN 구조에 비해 더 긴 시퀀스를 효과적으로 잘 기억하기 때문이다.
- LSTM은 우선은 대략적으로 hidden state를 계산하는 방법만 조금 다를뿐, RNN과 기본적으로는 같다.

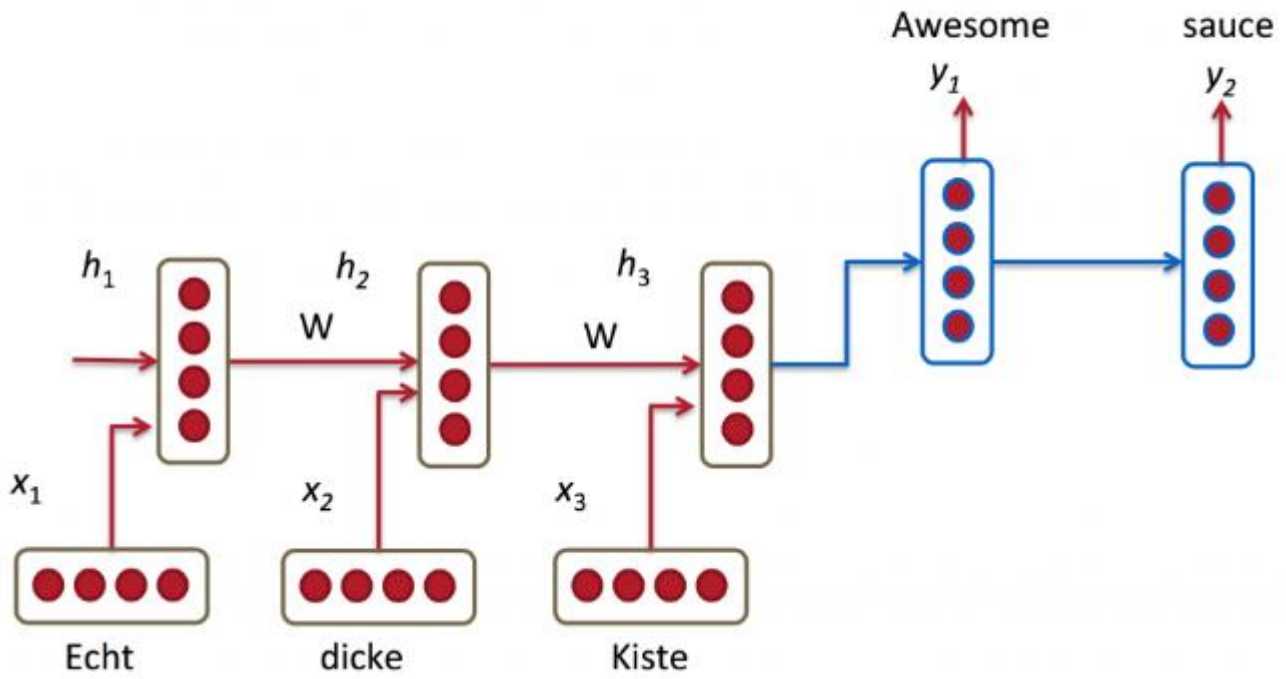
. 언어 모델링과 텍스트 생성

- 언어 모델은 주어진 문장에서 이전 단어들을 보고 다음 단어가 나올 확률을 계산해주는 모델
- 언어 모델은 어떤 문장이 실제로 존재할 확률이 얼마나 되는지 계산해 주기 때문에, 자동 번역의 출력값으로 어떤 문장을 내보내는 것이 더 좋은지 (실생활에서 높은 확률로 존재하는 문장들은 보통 문법적/의미적으로 올바르기 때문) 알려줄 수 있다.
- 문장에서 다음 단어가 나타날 확률을 계산해주는 주 목적 외의 부수적인 효과로 생성(generative) 모델을 얻을 수 있는데, 출력 확률 분포에서 샘플링을 통해 문장의 다음 단어가 무엇이 되면 좋을지 정한다면 기존에 없던 새로운 문장을 생성
- 언어 모델에서의 입력값은 단어들의 시퀀스 (e.g. one-hot encoded 벡터 시퀀스)이고, 출력은 추측된 단어들의 시퀀스
- 네트워크를 학습할 때에는 시간 스텝t에서의 출력값이 실제로 다음 입력 단어가 되도록 Yt=Xt+1

RNN의 예시

. 자동번역(기계번역)

- 입력이 단어들의 시퀀스라는 점에서 언어 모델링과 비슷하지만, 출력값이 다른 언어로 되어있는 단어들의 시퀀스라는 점이다.
- 입력값을 전부 다 받아들인 다음에서야 네트워크가 출력값을 내보낸다는 점에 있는데, 번역 문제에서는 어순이 다른 문제 등이 있기 때문에 대상 언어의 문장의 첫 단어를 알기 위해선 번역할 문장 전체를 봐야 할 수도 있기 때문이다.



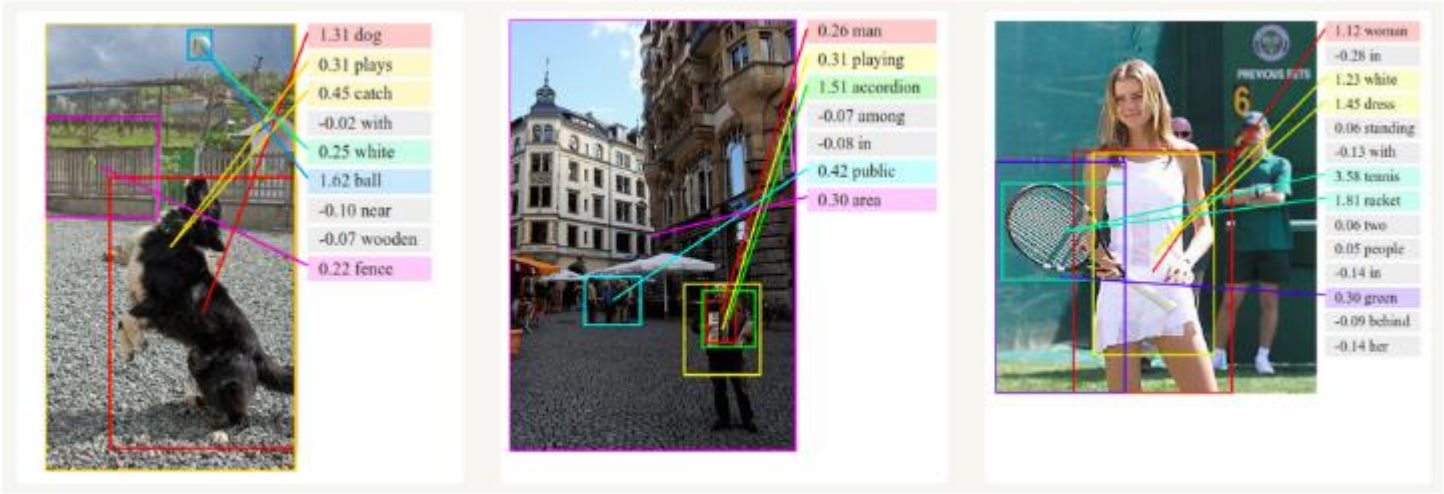
RNN의 예시

. 음성 인식

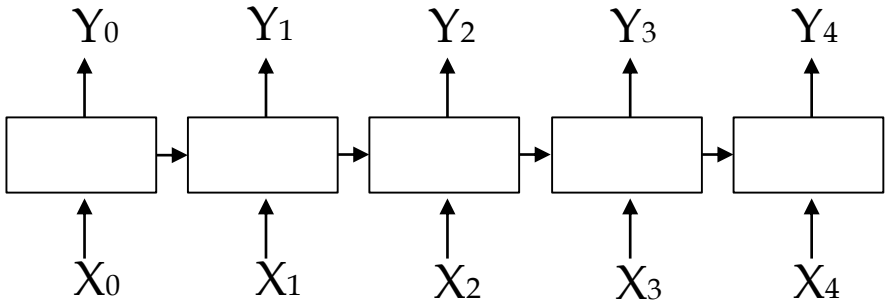
- 사운드 웨이브의 음향 신호(acoustic signal)를 입력으로 받아들이고, 출력으로는 음소(phonetic segment)들의 시퀀스와 각각의 음소별 확률 분포를 추측

. 이미지 캡션 생성

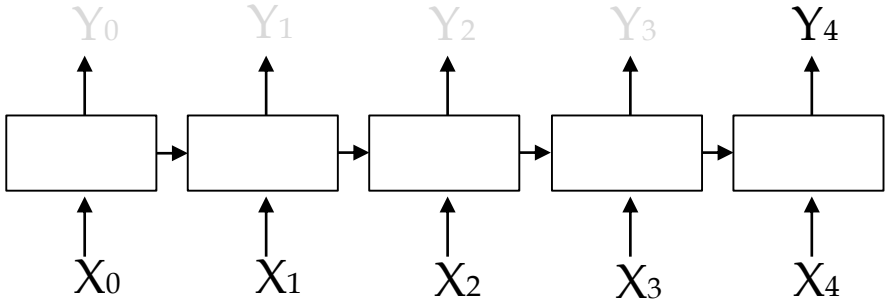
컴퓨터 비젼에서 활발하게 사용된 convolutional neural network(CNN)과 RNN을 함께 사용한다면, 임의의 이미지를 텍스트로 설명해주는 시스템을 만드는 것도 가능하다. 실제로 어떻게 왜 동작하는지는 상당히 신기하다. CNN과 RNN을 합친 모델은 이미지로부터 얻어낸 주요 단어들과 이미지의 각 부분을 매칭해줄 수도 있다.



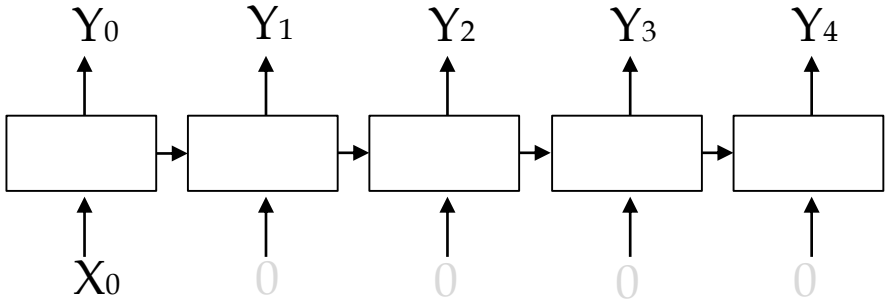
RNN – 입력과 출력 시퀀스



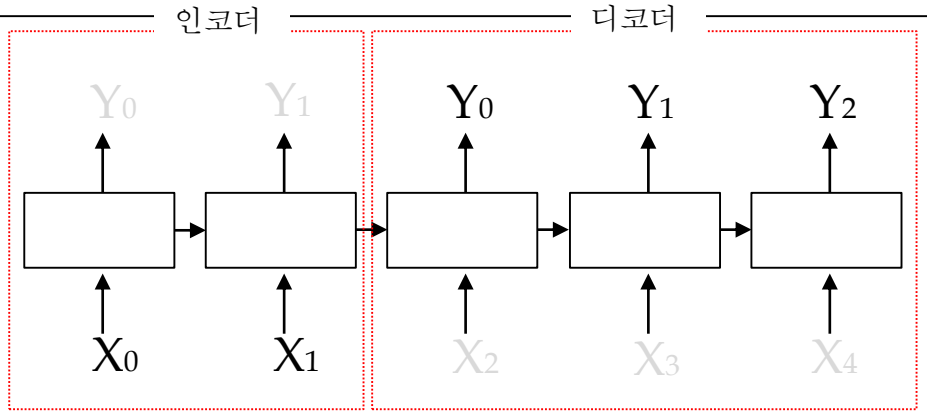
시퀀스 투 시퀀스: 주식가격 같은 시계열 데이터 예측 유용
최근 N일치의 주식가격을 주입=> 하루 앞선 가격 출력
(즉 N-1일 전부터 내일까지)



시퀀스 투 벡터: 영어 리뷰에 있는 연속되 단어 주입
네트워크는 감성 점수 출력(-1(싫다)에서 +1(좋다))



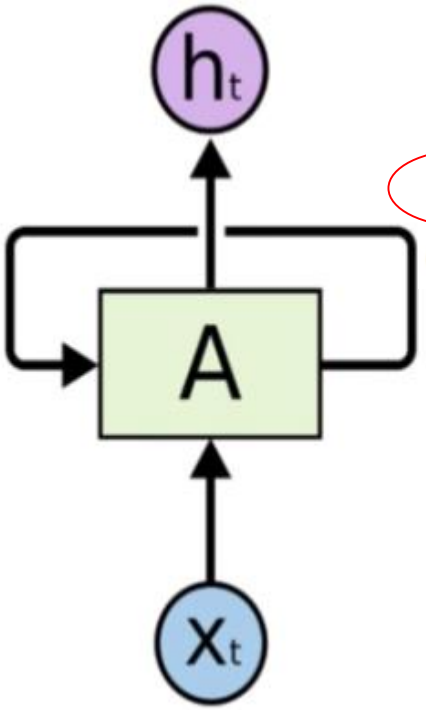
벡터 투 시퀀스: 이미지를 입력하여 이미지에 대한 캡션을 출력



지연된 시퀀스 투 시퀀스:시퀀스 투 벡터(인코더)뒤에 벡터 투 시퀀스(디코더) 연결
한 언어의 문장(벡터 표현)을 다른 언어(디코딩)로 번역:

https://tensorflowkorea.gitbooks.io/tensorflow-kr/content/g3doc/tutorials/seq2seq/

RNN – 예제




```
cell = tf.contrib.rnn.BasicRNNCell(num_units=hidden_size)
cell = tf.contrib.rnn.BasicLSTMCell(num_units=hidden_size)
...
outputs, _states = tf.nn.dynamic_rnn(cell, x_data, dtype=tf.float32)
```

출력사이즈

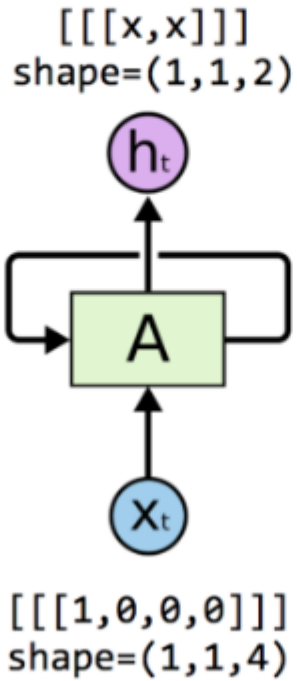
교체가능

Output이 다음의 셀과 연결

. Dynamix_rnn()

적절한 타임 스텝에 걸쳐 셀을 실행하기 위해 while_loop() 연산을 사용

One node: 4 (*input-dim*) in 2 (*hidden_size*)



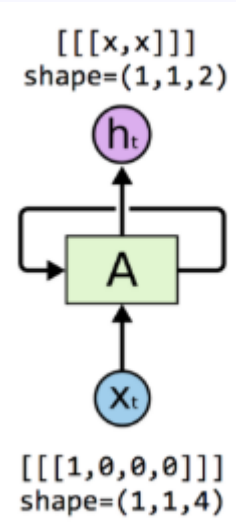
hidden_size=2

입력데이터상관없이 출력의 Dimension은 2이다.

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

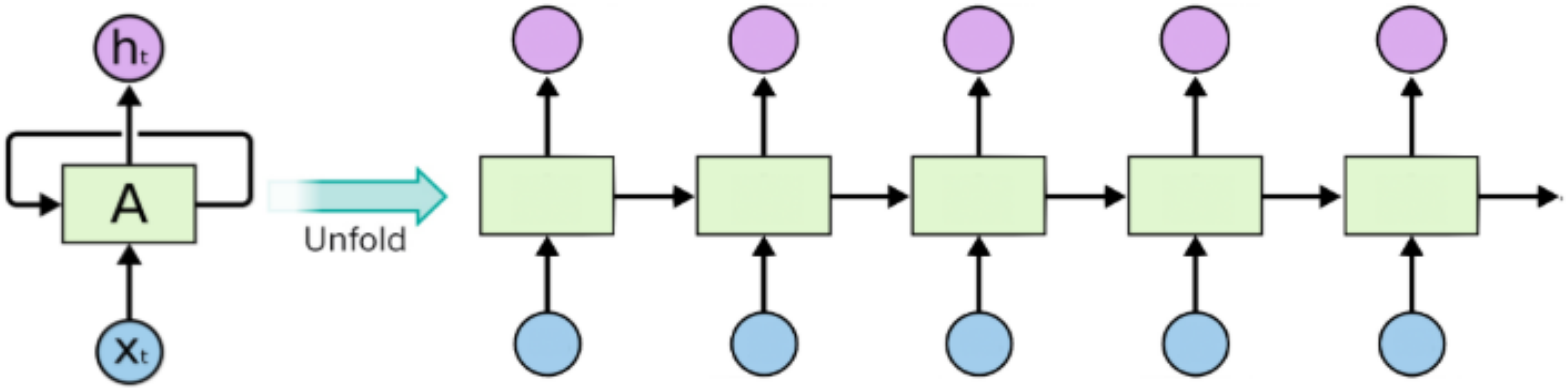
RNN – 예제Input 4 dim->2 hidden_size

```
import tensorflow as tf
import numpy as np
from tensorflow.contrib import rnn
import pprint
pp = pprint.PrettyPrinter(indent=4)
sess = tf.InteractiveSession()
with tf.variable_scope('one_cell') as scope:
    # One cell RNN input_dim (4) -> output_dim (2)
    hidden_size = 2
    cell = tf.contrib.rnn.BasicRNNCell(num_units=hidden_size)
    print(cell.output_size, cell.state_size)

    x_data = np.array([[[1,0,0,0]]], dtype=np.float32) # x_data = [[[1,0,0,0]]]
    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())
```



RNN – HELLO예제

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

시퀀스 데이터

Hidden_size=2
sequence_length=5

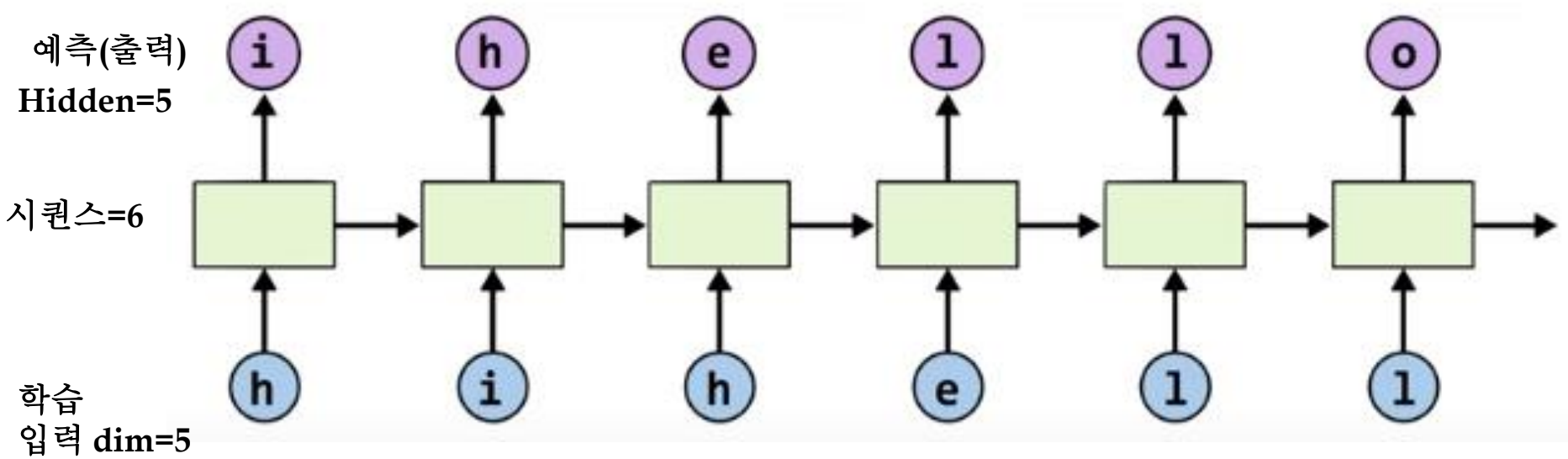
RNN - HELLO예제

```
import tensorflow as tf
import numpy as np
from tensorflow.contrib import rnn
import pprint

pp = pprint.PrettyPrinter(indent=4)
sess = tf.InteractiveSession()
with tf.variable_scope('two_sequances') as scope:
    # One cell RNN input_dim (4) -> output_dim (2). sequence: 5
    hidden_size = 2
    cell = tf.contrib.rnn.BasicRNNCell(num_units=hidden_size)
    x_data = np.array([[[1., 0., 0., 0.],
                        [0., 1., 0., 0.],
                        [0., 0., 1., 0.],
                        [0., 0., 1., 0.],
                        [0., 0., 0., 1.]]], 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())
```

입력과 출력의 shape확인

```
array([[[-0.27701476, -0.47703248],
        [-0.17500615,  0.2320899 ],
        [ 0.6444088 ,  0.15394928],
        [ 0.11884318,  0.10508373],
        [ 0.55162483,  0.43363634]]], dtype=float32)
```

RNN – HIHELLO예제



배치 =1(문자열)

. Index:
h:0 ,i:1 ,e:2 , l:3, o:4

.[1, 0, 0, 0, 0] #h
.[0, 1, 0, 0, 0] #i
.[0, 0, 1, 0, 0] #e
.[0, 0, 0, 1, 0] #l
.[1, 0, 0, 0, 1] #o

RNN - HIHELLO예제

```
import tensorflow as tf
import numpy as np
idx2char = ['h', 'i', 'e', 'l', 'o']
# Teach hello: hihell -> ihello
x_data = [[0, 1, 0, 2, 3, 3]]   # hihell
x_one_hot = [[[1, 0, 0, 0, 0],   # h 0
              [0, 1, 0, 0, 0],   # i 1
              [1, 0, 0, 0, 0],   # h 0
              [0, 0, 1, 0, 0],   # e 2
              [0, 0, 0, 1, 0],   # l 3
              [0, 0, 0, 1, 0]]]  # l 3
y_data = [[1, 0, 2, 3, 3, 4]]    # ihello
num_classes = 5
input_dim = 5  # one-hot size
hidden_size = 5  # output from the LSTM. 5 to directly predict one-hot
batch_size = 1   # one sentence
sequence_length = 6  # |ihello| == 6
learning_rate = 0.1
```

문자를 index로

인덱스

one-hot

학습시키고자하는 문자열

#출력하고자 하는 라벨

RNN - HIHELLO예제

```
X = tf.placeholder(
    tf.float32, [None, sequence_length, input_dim])  # X one-hot
Y = tf.placeholder(tf.int32, [None, sequence_length])  # Y label

cell = tf.contrib.rnn.BasicLSTMCell(num_units=hidden_size, state_is_tuple=True)
initial_state = cell.zero_state(batch_size, tf.float32) 출력:5
outputs, _states = tf.nn.dynamic_rnn(
    cell, X, initial_state=initial_state, dtype=tf.float32)

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)
```

RNN – HIHELLO예제

```
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    for i in range(2000):
        l, _ = sess.run([loss, train], feed_dict={X: x_one_hot, Y: y_data})
        result = sess.run(prediction, feed_dict={X: x_one_hot})
        print(i, "loss:", l, "prediction: ", result, "true Y: ", y_data)

        # print char using dic
        result_str = [idx2char[c] for c in np.squeeze(result)]
        print("₩tPrediction str: ", ''.join(result_str))
```

```
1997 loss: 1.5854686e-05 prediction:  [[1 0 2 3 3 4]] true Y:  [[1, 0, 2, 3, 3, 4]]
    Prediction str:  ihello
1998 loss: 1.5834818e-05 prediction:  [[1 0 2 3 3 4]] true Y:  [[1, 0, 2, 3, 3, 4]]
    Prediction str:  ihello
1999 loss: 1.5795082e-05 prediction:  [[1 0 2 3 3 4]] true Y:  [[1, 0, 2, 3, 3, 4]]
    Prediction str:  ihello
```

숫자=>문자변환

RNN – Long Sequence Rnn

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

RNN – HIHELLO예제 –Data creation

```
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)

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)
```

RNN – HIHELLO예제 –Data creation

```
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))
```

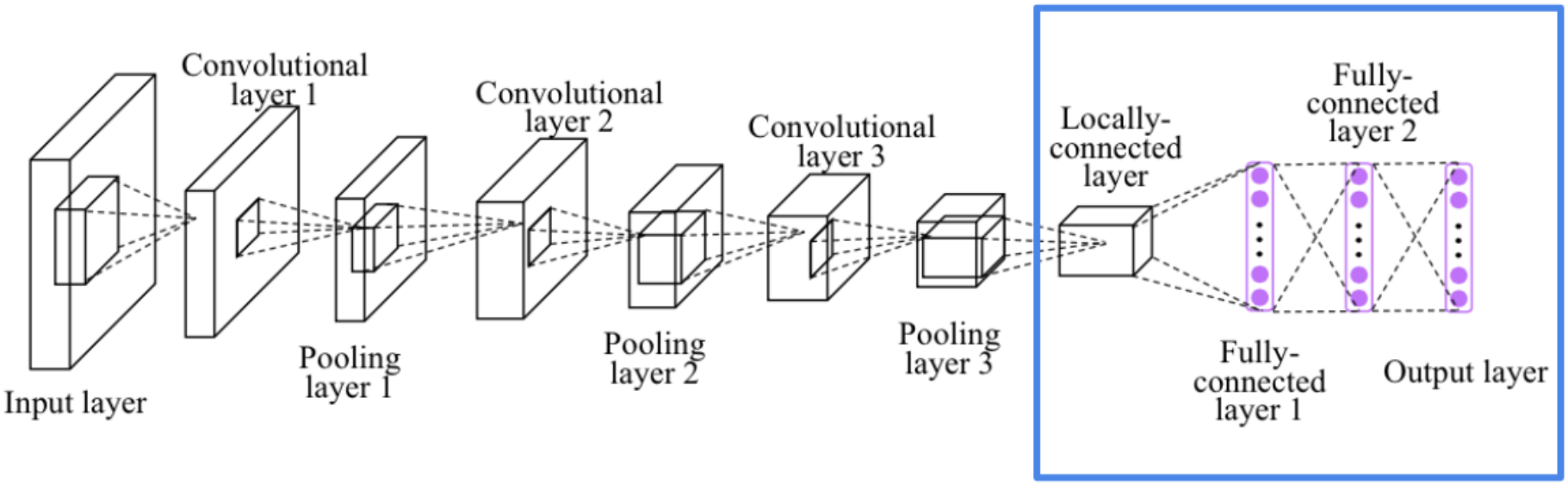
```
1995 loss: 0.61510855 prediction:  [[1 0 2 3 3 4]] true Y:  [[1, 0, 2, 3, 3, 4]]
    Prediction str:  ihello
1996 loss: 0.6151067 prediction:  [[1 0 2 3 3 4]] true Y:  [[1, 0, 2, 3, 3, 4]]
    Prediction str:  ihello
1997 loss: 0.6151049 prediction:  [[1 0 2 3 3 4]] true Y:  [[1, 0, 2, 3, 3, 4]]
    Prediction str:  ihello
1998 loss: 0.6151031 prediction:  [[1 0 2 3 3 4]] true Y:  [[1, 0, 2, 3, 3, 4]]
    Prediction str:  ihello
1999 loss: 0.6151013 prediction:  [[1 0 2 3 3 4]] true Y:  [[1, 0, 2, 3, 3, 4]]
    Prediction str:  ihello
```

RNN – Really long sentence

```
sentence = ("if you want to build a ship, don't drum up people together to "
            "collect wood and don't assign them tasks and work, but rather "
            "teach them to long for the endless immensity of the sea.")
```

```
# training dataset
0 if you wan -> f you want
1 f you want ->  you want
2  you want  -> you want t
3 you want t -> ou want to
...
168  of the se -> of the sea
169 of the sea -> f the sea.
```

RNN – Really long sentence



RNN – Really long sentence

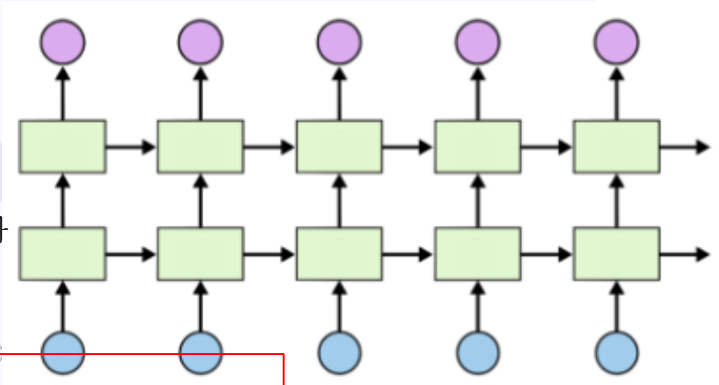
```
import tensorflow as tf
import numpy as np
from tensorflow.contrib import rnn
sentence = ("if you want to build a ship, don't drum up people together to "
            "collect wood and don't assign them tasks and work, but rather "
            "teach them to long for the endless immensity of the sea.")
char_set = list(set(sentence))   #중복없는 알파벳 집합 만들고 리스트변환 print(char_set)
char_dic = {w: i for i, w in enumerate(char_set)} #알파벳을 key, 인텍스를 value로 하는 딕셔너리 생성
data_dim = len(char_set)
hidden_size = len(char_set)                    #각 셀의 출력크기
num_classes = len(char_set)                    #분류 총수
sequence_length = 10  # Any arbitrary number   #1개 시퀀스의 길이(시계열데이터의 입력갯수)
learning_rate = 0.1                            #학습률
dataX = []  #입력 시퀀스를 저장하기 위한 배열
dataY = []  #출력 시퀀스를 저장하기 위한 배열
for i in range(0, len(sentence) - sequence_length):
    x_str = sentence[i:i + sequence_length]
    y_str = sentence[i + 1: i + sequence_length + 1]
    print(i, x_str, '->', y_str)

    x = [char_dic[c] for c in x_str]  # x str to index
    y = [char_dic[c] for c in y_str]  # y str to index

    dataX.append(x)
    dataY.append(y)
```

#시퀀스 길이만큼을 문자열
#입력보다 1칸 오른쪽에 시작하는 시퀀스길이만큼의 문자열
#print("==➔",x,"--➔"y)

```
# training dataset
0 if you wan -> f you want
1 f you want ->  you want
2  you want  -> you want t
3 you want t -> ou want to
…
168  of the se -> of the sea
169 of the sea -> f the sea.
```

RNN – Really long sentence

```
batch_size = len(dataX) #전체 169
X = tf.placeholder(tf.int32, [None, sequence_length])
Y = tf.placeholder(tf.int32, [None, sequence_length])
# One-hot encoding
X_one_hot = tf.one_hot(X, num_classes)  #전체분류개수1개차원추가
print(X_one_hot)  # check out the shape
# Make a lstm cell with hidden_size (each unit output vec
def lstm_cell():
    cell = rnn.BasicLSTMCell(hidden_size, state_is_tuple=True)
    return cell
                                        #2층의 stacked RNN생성
multi_cells = rnn.MultiRNNCell([lstm_cell() for _ in range(2)], state_is_tuple=True)
# outputs: unfolding size x hidden size, state = hidden size
outputs, _states = tf.nn.dynamic_rnn(multi_cells, X_one_hot, dtype=tf.float32)
# FC layer  #전체분류개수(Fully Connected)
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])
```



#RNN Cell들을 연결
#hidden size는 각 RNN Cell출력크기

RNN – Really long sentence

```
# All weights are 1 (equal weights)
weights = tf.ones([batch_size, sequence_length])  #sequence loss를 구할때 모든 sequence의 가중치 동일하게 1
sequence_loss = tf.contrib.seq2seq.sequence_loss(
    logits=outputs, targets=Y, weights=weights)
mean_loss = tf.reduce_mean(sequence_loss)        #sequence loss 구한다
train_op = tf.train.AdamOptimizer(learning_rate=learning_rate).minimize(mean_loss)
sess = tf.Session()
sess.run(tf.global_variables_initializer())

for i in range(500):
    _, l, results = sess.run(
        [train_op, mean_loss, outputs], feed_dict={X: dataX, Y: dataY})
    for j, result in enumerate(results):
        index = np.argmax(result, axis=1)
        print(i, j, ''.join([char_set[t] for t in index]), l)

# Let's print the last char of each result to check it works
results = sess.run(outputs, feed_dict={X: dataX})
for j, result in enumerate(results):
    index = np.argmax(result, axis=1)
    if j is 0:  # print all for the first result to make a sentence
        print(''.join([char_set[t] for t in index]), end='')  #첫줄은 sequence_lenth만큼 출력
    else:
        print(char_set[index[-1]], end='')  #두번째 줄부터는 마지막1줄만 출력
```

```
499 169 n the sea. 0.22882889
t you want to build a ship, don't drum up people together to collect wood and don't
```