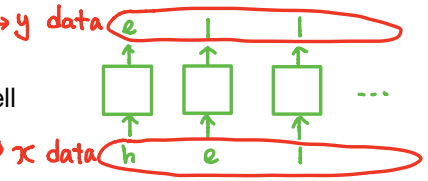
```
# Lab 12 Character Sequence RNN
import tensorflow as tf
import numpy as np

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_one_hot_eager = tf.one_hot(x_data, num_classes)  # one hot: 1 -> 0 1 0 0 0 0 0 0 0 0
x_one_hot_numpy = tf.keras.utils.to_categorical(x_data, num_classes)  # it'll generate numpy array,
either way works
y_one_hot_eager = tf.one_hot(y_data, num_classes)

tf.model = tf.keras.Sequential();
tf.model.add(tf.keras.layers.
          LSTM(units=num_classes, input_shape=(sequence_length, x_one_hot_eager.shape[2]),
return_sequences=True))
tf.model.add(tf.keras.layers.TimeDistributed(tf.keras.layers.Dense(units=num_classes,
activation='softmax')))
tf.model.summary()
tf.model.compile(loss='categorical_crossentropy', optimizer=tf.keras.optimizers.Adam(lr=learning_rate),
              metrics=['accuracy'])
tf.model.fit(x_one_hot_eager, y_one_hot_eager, epochs=50)


predictions = tf.model.predict(x_one_hot_eager)

for i, prediction in enumerate(predictions):
    # print char using argmax, dict
    result_str = [idx2char[c] for c in np.argmax(prediction, axis=1)]
    print("\tPrediction str: ", ''.join(result_str))
```

마지막 바로 앞 글자까지

두번째에서 마지막 글자까지

x_data를 one_hot으로 만들어줌

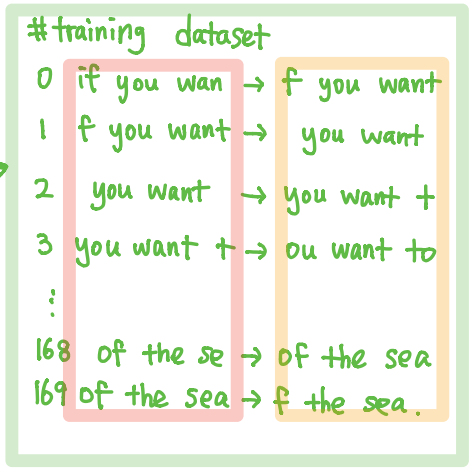


idx2char 크기와 동일(문자열의 문자 개수)

Prediction str: if you want you

```
import tensorflow as tf
import numpy as np

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))
char_dic = {w: i for i, w in enumerate(char_set)}

data_dim = len(char_set)
hidden_size = len(char_set)
num_classes = len(char_set)
sequence_length = 10  # Any arbitrary number
learning_rate = 0.1

dataX = []
dataY = []
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)

batch_size = len(dataX)
# One-hot encoding
X_one_hot = tf.one_hot(dataX, num_classes)
Y_one_hot = tf.one_hot(dataY, num_classes)

print(X_one_hot.shape)  # check out the shape (170, 10, 25)
print(Y_one_hot.shape)  # check out the shape


tf.model = tf.keras.Sequential();
tf.model.add(tf.keras.layers.
             LSTM(units=num_classes, input_shape=(sequence_length, X_one_hot.shape[2]),
return_sequences=True))
tf.model.add(tf.keras.layers.LSTM(units=num_classes, return_sequences=True))
tf.model.add(tf.keras.layers.TimeDistributed(tf.keras.layers.Dense(units=num_classes,
activation='softmax')))
tf.model.summary()
tf.model.compile(loss='categorical_crossentropy', optimizer=tf.keras.optimizers.Adam(lr=learning_rate),
                 metrics=['accuracy'])
tf.model.fit(X_one_hot, Y_one_hot, epochs=100)
```

임의의 길이로 잘라서 dataset을 만듭니다.



→ 데이터의 전체 길이를 batch_size로 설정합니다.

```
results = tf.model.predict(X_one_hot)
for j, result in enumerate(results):
    index = np.argmax(result, axis=1)
    if j == 0:  # print all for the first result to make a sentence
        print(''.join([char_set[t] for t in index]), end='')
    else:
        print(char_set[index[-1]], end='')
```

↳ 실행이 잘 되지 않음 ㅠㅠ