

🌙 Type something...

01) RNN Long sequence

01\_Making sequence dataset from long sentence

02\_Adding FC layer and stacking RNN

## 01) RNN Long sequence

### 01\_Making sequence dataset from long sentence

# Making sequence dataset from long sentence (code)

```
# data setting
x_data = []
y_data = []

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_data.append([char_dic[c] for c in x_str]) # x str to index
    y_data.append([char_dic[c] for c in y_str]) # y str to index

x_one_hot = [np.eye(dic_size)[x] for x in x_data]

# transform as torch tensor variable
X = torch.FloatTensor(x_one_hot)
Y = torch.LongTensor(y_data)
```

"if you wan" -> "f you want"  
"f you want" -> " you want "  
" you want " -> "you want t"  
"you want t" -> "ou want to"  
"ou want to" -> "u want to "  
...

- sequence\_length 단위로 문자열을 끊는다.
- 예를 들면 1~10, 2~11 이런식으로 9개씩 겹쳐서 진행된다.

### 02\_Adding FC layer and stacking RNN

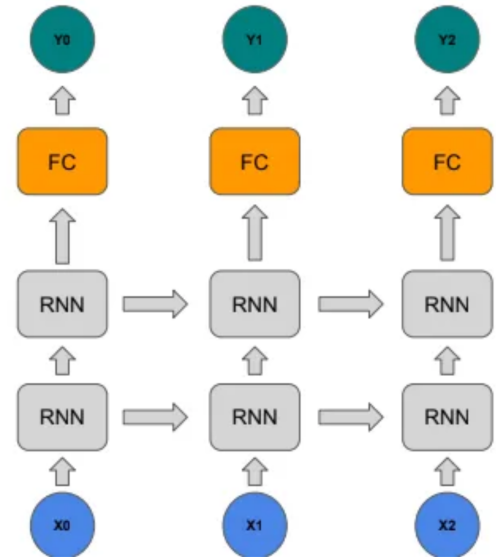
# Adding FC layer and stacking RNN

```
# declare RNN + FC
class Net(torch.nn.Module):
    def __init__(self, input_dim, hidden_dim, layers):
        super(Net, self).__init__()
        self.rnn = torch.nn.RNN(input_dim, hidden_dim, num_layers=layers,
                                batch_first=True)
        self.fc = torch.nn.Linear(hidden_dim, hidden_dim, bias=True)

    def forward(self, x):
        x, _status = self.rnn(x)
        x = self.fc(x)
        return x

net = Net(dic_size, hidden_size, 2)
```

- RNN을 2개 쌓은 후 FC를 통과하는 모델을 만들 것이다.



- 코드는 이전과 거의 동일하다.
- 다음 시간에는 시계열 데이터에 관해서 알아보자.