**김형원\_계획**

**1.csv 파일 분할 하기**

https://fruitdev.tistory.com/61

https://itholic.github.io/linux-split/

파일 분할 방법  
  
**split** **명령어 사용**

**1. 특정 사이즈로 파일 분할하기**

|  |
| --- |
| // 100MB 단위로 분할 $split -b 100m test.csv  // 100Kb 단위로 분할 $split -b 100k test.csv |

**2. 라인 단위로 분할**

|  |
| --- |
| // 1000줄 단위로 분할 [$ split -l 1000 test.csv  // 10만줄 단위로 분할 $ split -l 100000 test.csv |

분할된 파일은 파일명 뒤에 a,b,c 와 같이 순차적으로 파일명이 정해짐

출처: <https://fruitdev.tistory.com/61> [과일가게 개발자]

옵션 정리

-a : 분할되는 파일뒤에 붙을 길이를 지정함

--additional-suffix: 분할 파일 자릿수 뒤에 추가적으로 이름을 설정

-b: 분할되는 기준을 바이트 크기로 분할

-d: 분할 파일 이름 뒤에 영문이 아닌 숫자로 지정.

--numeric-suffoxes: -d옵션과 동일하지만 시작값을 지정 할 수 있음.

추가적인 명령어는 <https://bigsun84.tistory.com/310> 참조

**2. Tensorflow를 이용한 신경망 예제 사전 조사**

글을 분류하는 신경망은 Doc2Vec, fasttext, Word2Vec(문장은 단어 vector의 평균으로), LSTM 또는 RNN, CNN 등이 있다.

주로 텐서플로, Keras상에 라이브러리로 많이 구현이 되어있는 상황이며 이중에서 RNN 방식이 가장 기본적인 모델이다.

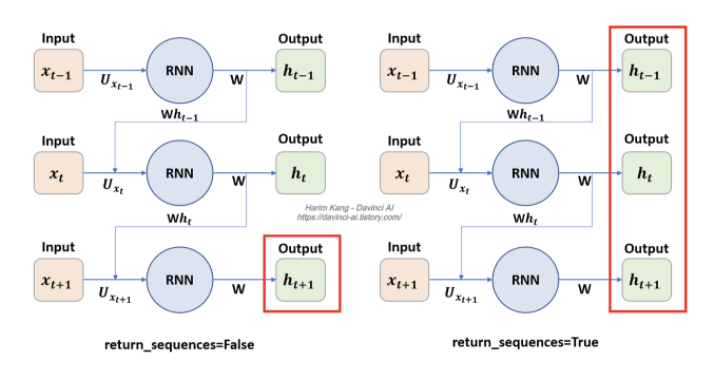
RNN(순환 신경망)은 은닉 계층 안에 하나 이상의 순환 계층을 갖는 신경망을 의미한다.

순환 신경망은 다른 네트워크들과 입력이 받는 방식에서 다르다. 순서가 있는 데이터에 주로 사용이 되며 해당 데이터를 입력으로 하나의 네트워크를 통해서 순서대로 출력을 얻는다.

순서가 있는 데이터는 시간의 변화와 함께 변화 되면서 그 영향을 받는 데이터를 의미한다.

시계이(가) 표시된 사진

자동 생성된 설명

****

출처: <https://davinci-ai.tistory.com/30>

케라스(Keras)로 RNN 구현하기

케라스로 RNN의 층, RNN의 은닉층을 추가하는 코드는

# 실제 RNN 은닉층을 추가하는 코드.   
model.add(SimpleRNN(hidden\_size)) # 가장 간단한 형태

다른 인자를 사용할 때

# 추가 인자를 사용할 때   
model.add(SimpleRNN(hidden\_size, input\_shape=(timesteps, input\_dim))) # 다른 표기   
model.add(SimpleRNN(hidden\_size, input\_length=M, input\_dim=N))   
# 단, M과 N은 정수

hidden\_size = 은닉 상태의 크기를 정의한다. 이는 simpleRNN이 은닉층에서 출력층으로 보내는 값의 크기(output\_dim)와도 동일하다. 이 인자를 units(유닛의 수)라고 표현하기도 한다. 이 수치를 늘리면 보통 RNN의 용량(capacity)을 늘린다고 보면 되며, 중소형 모델의 경우 보통 128, 256, 512, 1024 등의 값을 가진다.

timesteps = 입력 시퀀스의 길이(input\_length)라고 표현하기도 함. 시점의 수.

input\_dim = 입력의 크기.

**파이썬으로 Simple RNN 직접 구현**

import numpy as nptimesteps = 10  
input\_dim = 4  
hidden\_size = 8   
# hidden state의 크기. 즉, hidden\_size. 결과적으로 output\_dim이 된다.inputs = np.random.random((timesteps, input\_dim))  
# 입력에 해당되는 2D 텐서hidden\_state\_t = np.zeros((hidden\_size, ))  
# 초기 은닉 상태는 0(벡터)로 초기화, 은닉 상태의 크기 hidden\_size로 은닉 상태 만듬.hidden\_state\_t  
array([0., 0., 0., 0., 0., 0., 0., 0.])

Wx = np.random.random((hidden\_size, input\_dim))  
# (8, 4) 크기의 2D 텐서 생성. 입력에 대한 가중치.  
Wh = np.random.random((hidden\_size, hidden\_size))  
# (8, 8) 크기의 2D 텐서 생성. 은닉 상태에 대한 가중치  
b = np.random.random((hidden\_size, ))  
# (8, )크기의 1D 텐서 생성. 이 값은 bias 임.print(np.shape(Wx))  
print(np.shape(Wh))  
print(np.shape(b))

(8, 4)  
(8, 8)  
(8,)

total\_hidden\_states = []# RNN  
for input\_t in inputs: # 각 시점에 따라서 입력값이 입력됨.  
 output\_t = np.tanh(np.dot(Wx, input\_t) + np.dot(Wh, hidden\_state\_t)+b)  
 # Wx \* Xt + Wh \* Ht-1 + b(bias)  
 total\_hidden\_states.append(list(output\_t))  
 # 각 시점의 hidden state의 값을 계속해서 축적  
 print(np.shape(total\_hidden\_states))  
 # 각 시점 t별 RNN의 출력의 크기는 (timestep, output\_dim)  
 hidden\_state\_t = output\_t  
   
total\_hidden\_states = np.stack(total\_hidden\_states, axis=0)  
# 출력 시 값을 깔끔하게 하기 위함.

print(total\_hidden\_states)  
# (timesteps, output\_dim)의 크기. 이 경우 (10, 8)의 크기를 가지는 RNN 2D 텐서를 출력(1, 8)  
(2, 8)  
(3, 8)  
(4, 8)  
(5, 8)  
(6, 8)  
(7, 8)  
(8, 8)  
(9, 8)  
(10, 8)  
[[0.93646037 0.86843131 0.79261245 0.98304653 0.97312617 0.59469036  
 0.61637029 0.96442909]  
 [0.9999964 0.99992672 0.9995054 0.99999944 0.99994911 0.9999767  
 0.99985392 0.99998516]  
 [0.99999648 0.99991793 0.99971593 0.9999992 0.99997999 0.99999556  
 0.99984872 0.99998874]  
 [0.99999888 0.99994918 0.99983867 0.99999986 0.9999874 0.99999771  
 0.99994573 0.99999647]  
 [0.9999966 0.99985143 0.99941126 0.99999895 0.99995823 0.99999412  
 0.9998688 0.99998034]  
 [0.9999993 0.99994494 0.9995957 0.99999982 0.99998686 0.99999519  
 0.99994388 0.99999138]  
 [0.99999897 0.99996659 0.99993153 0.99999989 0.99999264 0.99999864  
 0.99987739 0.99999656]  
 [0.99999748 0.99981742 0.99927514 0.99999909 0.99994581 0.99999426  
 0.99985333 0.99997178]  
 [0.99999906 0.99992839 0.99974262 0.99999981 0.99998155 0.99999716  
 0.99990593 0.99999092]  
 [0.99999949 0.99996681 0.99985971 0.99999992 0.99999275 0.99999779  
 0.99991641 0.99999483]]

출처: <https://medium.com/@omicro03/%EC%9E%90%EC%97%B0%EC%96%B4%EC%B2%98%EB%A6%AC-nlp-11%EC%9D%BC%EC%B0%A8-rnn-299a48d16bf0>

**3. 앞으로 계획**

**\*Default = 복습 또 복습**

1) 관련 연구 논문 찾아보고 모르는 부분 공부, 기회가 된다면 딥러닝 코세라 강의 들어보기

2) 데이터 추출, 최상위 카테고리 개수 정해보기, 라벨링 방법. -> KNN군집화 사용??..., BootStrap방식 사용??

3) 환경 구축(colab? Linux? AWS? PC?)

4) 전체적인 카테고라이징 시스템 설계

5) 구현