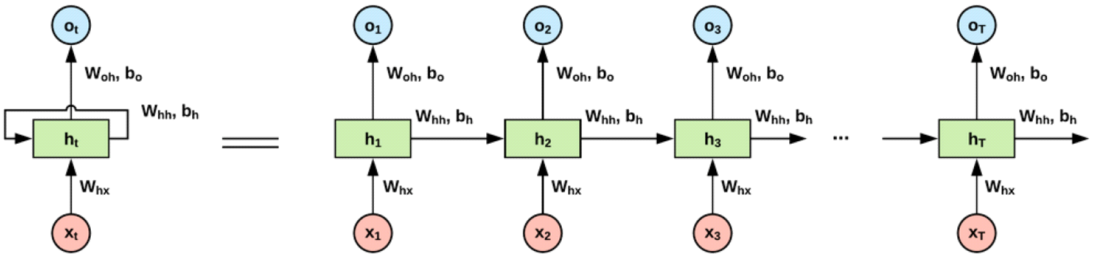
# Chapter15. RNN과 CNN을 사용해 시퀀스 처리하기

2021.01.08. 고낙헌

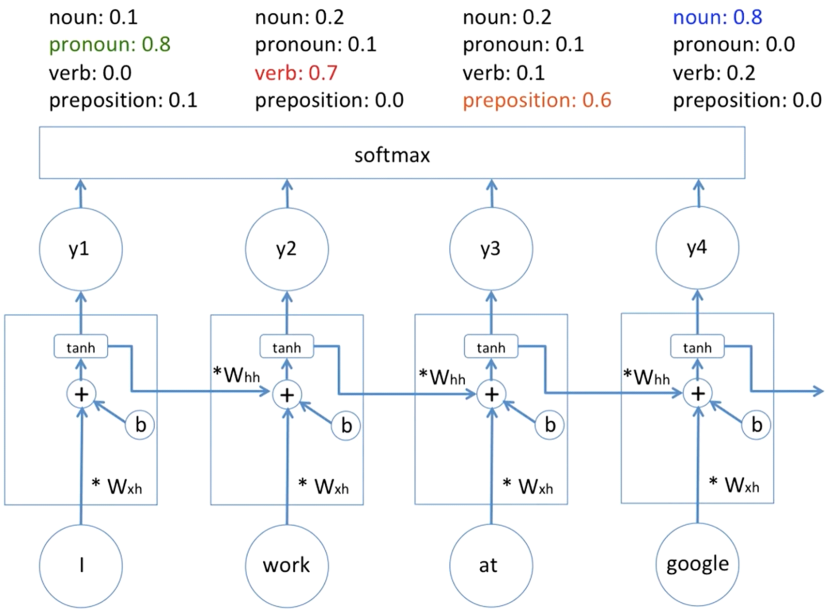
## (1) RNN의 필요성

- 시계열 데이터를 분석하기 위함
- 주식 가격 등을 예측해 언제 사고팔지 제시
- 자율주행 시스템에서 차의 이동 경로 예측
- 임의 길이를 가진 시퀀스형 데이터(문장, 문서, 오디오 샘플 등) 처리 → 자연어 처리에 유용
- 예시) home run과 run home의 의미 차이: 데이터를 하나의 배열로 입력받는 DNN에서는 두 어절이 같은 입력으로 들어감(home과 run이 각각 한 번씩 쓰였기 때문). 반면 RNN에서는 home이 먼저 입력되느냐 run이 먼저 입력되느냐에 따라 다른 레이블('홈런', '집에 가다')을 부여할 수 있으므로 유용함.

## (2) 순환 뉴런과 순환 층



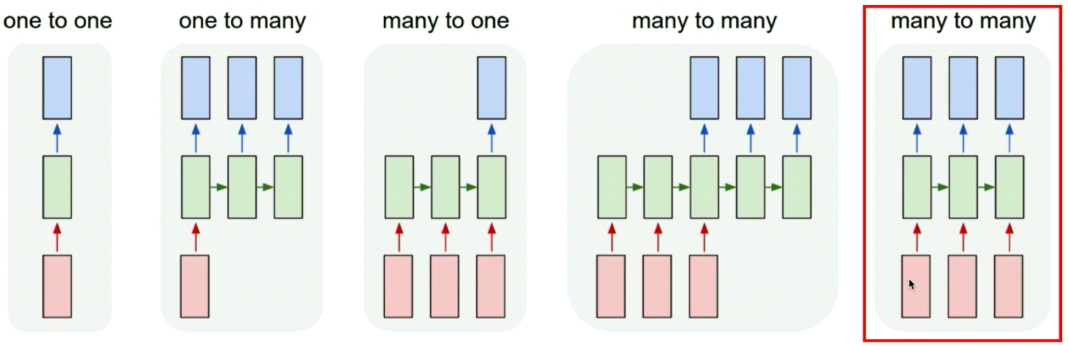

- 실제 RNN 모델은 우측처럼 생겼으나 좌측처럼 '순환 뉴런'의 형태로 표시하기도 함.
- 순환 뉴런으로 된 층은 타임 스텝마다 입력 벡터와 이전 타임 스텝의 출력 벡터를 입력받음.
- RNN 사용 예시



### (3) 메모리 셀

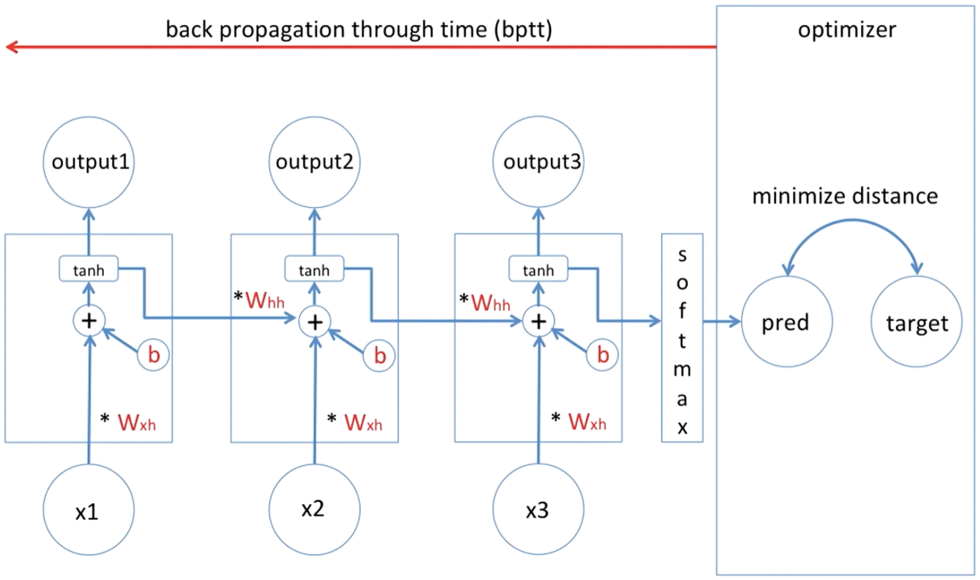
- 타임 스텝 $t$에서 순환 뉴런의 출력은 이전 타임 스텝의 모든 입력에 대한 함수이므로 이를 일종의 메모리 형태라고 말할 수 있음. 이때 RNN의 각 뉴런을 '메모리 셀'이라고 표현할 수 있음.
- $h_{(t)} = f(h_{(t-1)}, x_{(t)})$: Hidden Layer의 결괏값

### (4) 입력과 출력 시퀀스




① Vector-to-Vector (one to one): 기존 다층 퍼셉트론 등으로 풀어왔던 문제
② Vector-to-Sequence Network (one to many): 이미지를 주고 그에 대한 설명(자연어)을 형성하는 문제
③ Sequence-to-Vector Network (many to one): 문장 정보를 주고 그 문장이 happy인지 unhappy인지 이진 분류하는 문제
④ Encoder & Decoder (many to many): 한 언어를 다른 언어로 번역하는 문제
⑤ Sequence-to-Sequence Network (many to many): 각 입력 단어 요소의 품사를 예측(분류)하는 문제

### (5) RNN 훈련하기



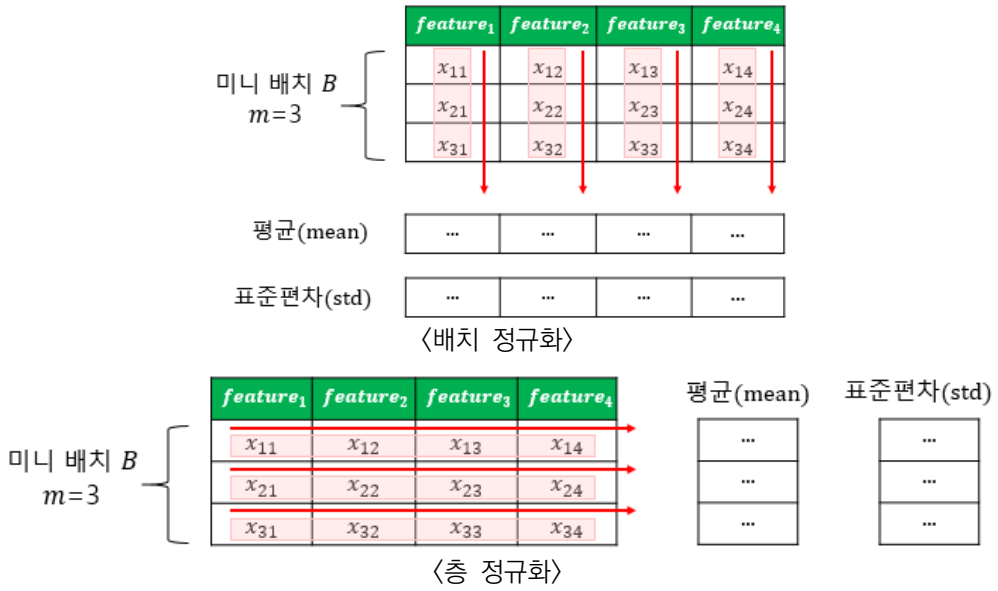

- BPTT(Back Propagation Through Time): time step에 기반을 둔 오차 역전파

### (6) 시계열 예측하기

- 시계열(Time Series): time step마다 하나 이상의 값을 가진 시퀀스
- 단변량 시계열(Univariate time series): time step마다 하나의 값을 가지는 시계열
- 다변량 시계열(Multivariate time series): time step마다 여러 값을 가지는 시계열
- 여러 time step 앞을 예측하는 방법: ① 하나씩 예측한 값을 다음 층에 입력으로 추가하는 방법, ② 최종 출력 shape를 여러 개로 설정하는 방법

### (7) 층 정규화

- RNN에서는 배치 정규화를 사용하기 어렵다. 입력 데이터 자체가 한 번에 들어오는 게 아닌 time step마다 들어오기 때문
- 층 정규화: 배치 차원에서 정규화하는 게 아닌 특성 차원에 대해 정규화하는 것
- sample에 독립적으로 time step마다 동적으로 필요한 통계를 계산할 수 있다.

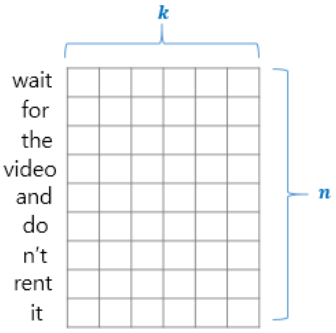



### (8) LSTM(Long Short-Term Memory, 장단기 메모리)

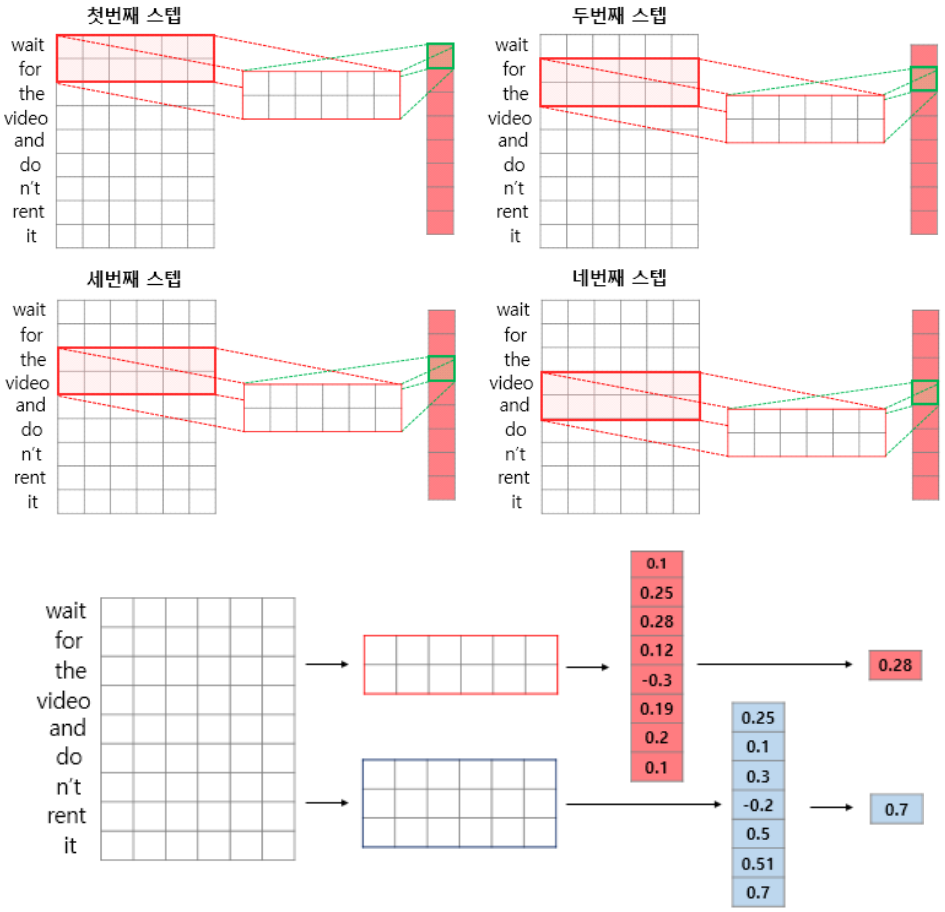
- 긴 시퀀셜 데이터가 입력되면 Gradient Vanishing 또는 Gradient Exploding 문제 발생
- Gradient Vanishing: Gradient가 1보다 매우 작으면 Chain Rule에 의해 작은 수가 많이 곱해지므로 0에 수렴 → $W = W_0 - \eta \frac{\partial E}{\partial W}$로부터 $W \approx W_0$
- Gradient Exploding: Gradient가 1보다 매우 크면 Chain Rule에 의해 큰 수가 많이 곱해지므로 발산 → $W = W_0 - \eta \frac{\partial E}{\partial W}$로부터 $W$가 진동
- LSTM: 시퀀셜 데이터를 기억하고 잊는 것을 담당하는 RNN 구조 (https://www.youtube.com/watch?v=bX6GLbpw-A4 4분 12초)
- GRU(Gated Recurrent Unit): LSTM의 간소화된 버전. 두 상태 벡터가 하나의 벡터로 합쳐졌으며, 하나의 게이트 제어기가 삭제와 입력을 모두 제어. 출력 게이트가 없고 전체 상태 벡터가 time step마다 출력됨.

### (9) 1D CNN으로 시퀀스 처리

- 기존 RNN 모델에 입력되는 데이터 형태는 임베딩(Embedding) 벡터이기 때문에 자연어 처리를 예로 들자면 아래와 같이 표현됨. k는 임베딩 벡터의 차원, n은 문장의 길이.




- LSTM에서는 time step마다 한 행이 순차적으로 입력됨. 1D CNN에서는 높이 사이즈를 커널 사이즈로 간주하고 Convolution과 Max-Pooling 진행.




### (10) WAVENET

- Conv1D Layer를 여러 겹 쌓아 RNN처럼 연결하는 방식.
- https://blog.kakaocdn.net/dn/b9fSQA/btqDVWToZLj/V71LK6tkBWU6q9cEubxOu0/img.gif
- 오디오 1초에 수만 개의 time step이 포함될 수 있음. LSTM, GRU에 비해 엄청난 길이의 시퀀스를 다룰 수 있음.

# Chapter16. RNN과 어텐션을 사용한 자연어 처리

2021.01.08. 고낙헌

### (1) Char-RNN을 사용한 셰익스피어 문체 생성 예제

① 훈련 데이터셋 만들기: 셰익스피어 작품을 모두 데이터셋으로 만들고 모든 글자를 정수로 인코딩

② train_test_split: 시퀀셜 데이터이어야 하므로 텍스트에 있는 글자를 섞으면 안 됨. 텍스트의 처음 90%를 훈련 세트로 사용하고 다음 5%를 검증 세트, 마지막 5%를 테스트 세트로 사용하는 등의 방식으로 진행. 시계열 데이터의 경우 보통 시간에 따라 구분함(2000~2012년은 훈련 세트, 2013~2015년은 검증 세트, 2016~2018년은 테스트 세트).

③ 순차 데이터를 window 여러 개로 자르기: window( ) 메서드로 짧은 텍스트 window를 갖는 데이터셋 만듦.

④ Char-RNN 모델 만들고 훈련: GRU 등 사용하여 모델 생성. 다음 단어를 예측하는 모델을 만들어야 하므로 many to one 방식으로 제작.

⑤ Char-RNN 모델 사용: 예측

⑥ 가짜 셰익스피어 텍스트 생성: 하나씩 예측한 텍스트를 다시 모델에 전달하여 계속 예측.

### (2) 감정 분석 (영화 리뷰 분석 예제)

- 영화 리뷰를 기반으로 긍정과 부정을 분류하므로 many to one 방식 사용.

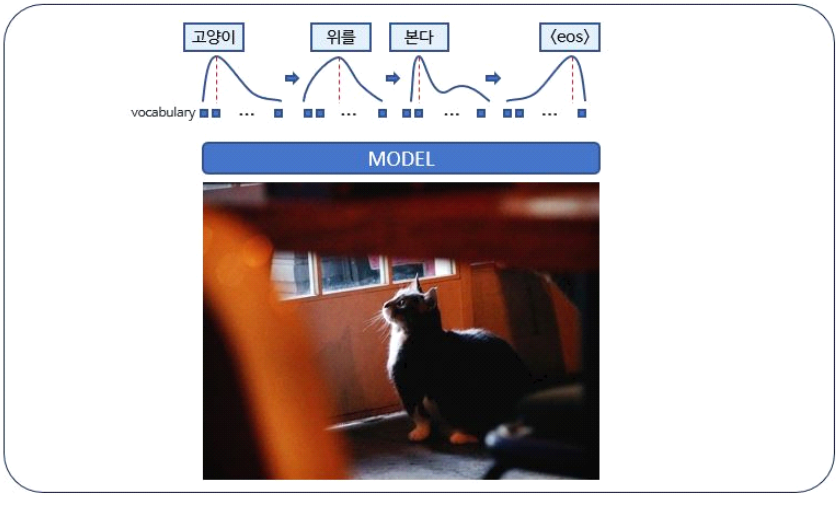
### (3) 신경망 기계 번역 (영어 문장 프랑스어 번역 예제)

- many to many 방식 사용.

### (4) 양방향 RNN

- 더 정교한 번역을 위해 양쪽에서 문맥을 읽는 방식의 RNN을 설계하는 것
- 양방향 RNN에 사용되는 층을 '양방향 순환 층(Bidirectional Recurrent Layer)'이라고 함.

### (5) 자연어 생성 모델



- 일반적인 자연어 생성 모델은 각각의 디코딩 time step에서 전체 단어 사전에 대한 확률 분포를 예측하는 방식임.
- 위 사진과 같이 모델이 예측한 확률 분포에 대해 디코딩하기 위해서는 예측된 확률 분포에 따라 가능한 모든 아웃풋 시퀀스의 조합을 탐색(Search)해야 함. 그런데 일반적으로 단어 사전은 수만 개의 토큰을 포함하고 있으므로 전체 공간을 탐색하는 것은 계산적으로 불가능. 따라서 실제로는 휴리스틱한 방법을 사용해 "충분히 좋은" 아웃풋 시퀀스를 생성.

① Greedy Search Decoder: 아웃풋 시퀀스에서 생성된 각각의 확률 분포에서 가장 값이 높은 토큰을 선택하는 탐욕 방법. 위 사진과 같은 상황. 하지만 몇몇 경우에서는 실제 최적점과는 거리가 있을 수 있음.

② Beam Search Decoder: 각 time step에서 탐색의 영역을 k개의 가장 가능도가 높은 토큰들로 유지하며 다음 단계를 탐색. k개의 가능성 있는 문장의 리스트를 유지하고 디코더 단계마다 이 문장의 단어를 하나씩 생성하여 가능성 있는 k개의 문장을 만드는 방법.

| 번호 | 빔 크기 k=1 | 빔 크기 k=5 |
|---|---|---|
| 1 | 너 스스로 말하고, 나를 위해 말하지마. | 당신 자신을 위해 말하세요. 당신은 나를 위해 말하지 않아요. |
| 2 | 걱정마. 난 그것에 익숙해. | 걱정마. 난 그것에 익숙해져 있어. |
| 3 | 너는 내 상사에게 말해야 한다. | 너는 나의 상사에게 말해야 한다. |
| 4 | 그녀는 그와 결혼하지 않기로 결심한 것을 느꼈다. | 그녀는 그와 결혼하지 않기로 결심한 그녀의 결심을 느꼈다. |
| 5 | 배달 채널을 삭제할 수 없습니다. | 배달 채널이 삭제되지 않았습니다. |
| 6 | 하지만 여러모로 나는 행복한 사람이 나를 화나게 하지 않는다. | 하지만 많은 면에서 나는 행복한 사람이 나를 화나게 하지 않는다. |
| 7 | 그 점원의 불의의 정중함은 나를 기분 좋게 만들었다. | 그 점원의 예상치 못한 공손함이 나를 기분 좋게 만들었다. |
| 8 | 내가 너의 조종사를 보고 있는 것을 신경쓰지 않기를 바란다. | 내가 너의 조종사를 감시하는 것을 신경쓰지 않았으면 좋겠어. |
| 9 | 저 소녀는 너와 이야기하고 싶어해. | 저 소녀는 너에게 말하고 싶어해. |

- Beam Search Algorithm

1) 각 스텝에서 각각의 후보 시퀀스를 모든 가능한 다음 step으로 확장한다.
2) 확장된 후보 스텝에 대해 점수를 계산하는데, 점수는 모든 확률 값을 곱하여 얻는다.
3) 가능도가 높은 k개의 시퀀스만 남기고, 나머지 후보들은 제거한다.
4) 시퀀스가 끝날 때까지 위의 과정을 반복한다.

https://littlefoxdiary.tistory.com/4

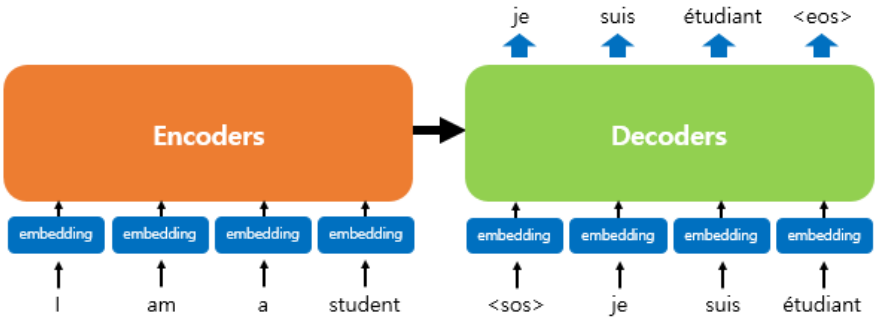
### (6) 어텐션 매커니즘

- 디코더에서 출력 단어를 예측하는 time step마다 인코더에서의 전체 입력 문장을 다시 한번 참고하되, 해당 시점에서 예측해야 할 단어와 연관 있는 단어 부분을 좀 더 집중(attention)해서 보는 방법.
- 닷-프로덕트 어텐션(Dot-Product Attention): https://wikidocs.net/22893
- 바다나우 어텐션(Bahdanau Attention): https://wikidocs.net/73161
- 양방향 LSTM과 어텐션 메커니즘: https://wikidocs.net/48920

### (7) 트랜스포머

- 기존의 구조인 인코더-디코더를 따르면서도 어텐션(Attention)만으로 구현한 모델. RNN을 사용하지 않으며 RNN보다 성능이 우수함.

- 이전 seq2seq 구조에서는 인코더와 디코더에서 각각 하나의 RNN이 t개의 시점(time-step)을 가지는 구조였다면 이번에는 인코더와 디코더라는 단위가 N개로 구성되는 구조




- 트랜스포머 실습: https://wikidocs.net/31379
- 트랜스포머를 이용한 한국어 챗봇 예제: https://wikidocs.net/89786

**(8) 재밌어 보이는 것**

- RNN으로 '나비야' 동요 학습해보기: https://tykimos.github.io/2017/04/09/RNN_Layer_Talk/
- RNN을 이용한 텍스트 생성: https://wikidocs.net/45101
- 글자 단위 RNN(Char RNN) 실습: https://wikidocs.net/48649
- 네이버 영화 리뷰 감성 분류(RNN): https://wikidocs.net/44249
- 네이버 영화 리뷰 감성 분류(1D CNN): https://wikidocs.net/85337
- 네이버 쇼핑 리뷰 감성 분류(RNN): https://wikidocs.net/94600
- Steam에 등록된 한국어 리뷰 감성 분류(RNN): https://wikidocs.net/94748