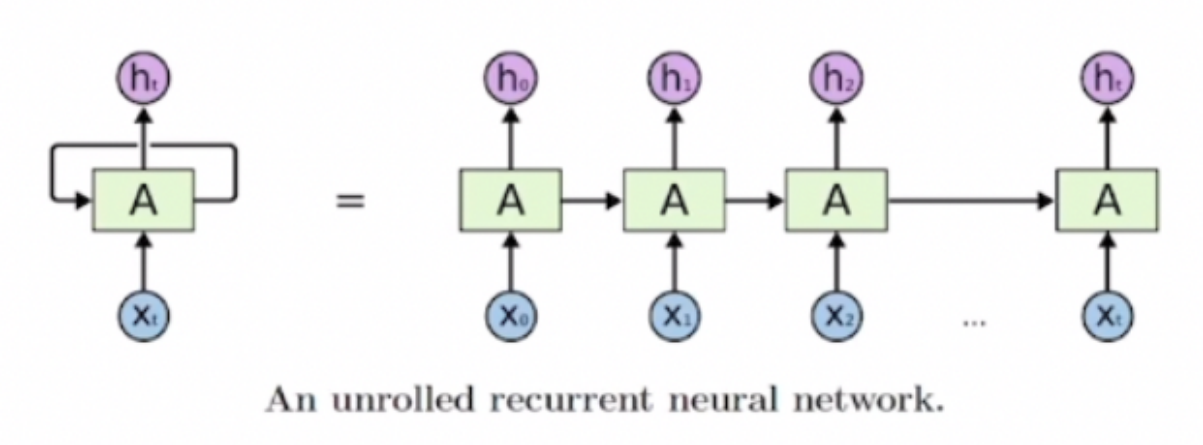
2. 자연어 처리와 딥러닝

날짜	@2024년 1월 13일

Recurrent Neural Network (RNN)

1️⃣ Basic Structure

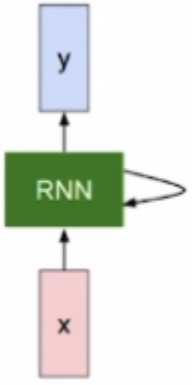


An unrolled recurrent neural network.

- RNN은 이전 time step까지의 hidden state 정보를 이용해 현재 time step의 hidden state 정보 h_t를 예측함
- 매 time step마다 동일한 파라미터를 가진 모듈을 사용한다는 점에서 재귀적인 호출의 특성을 보여줌

🎯 Inputs and Outputs of RNNs

$$h_t = f_W(h_{t-1}, x_t)$$



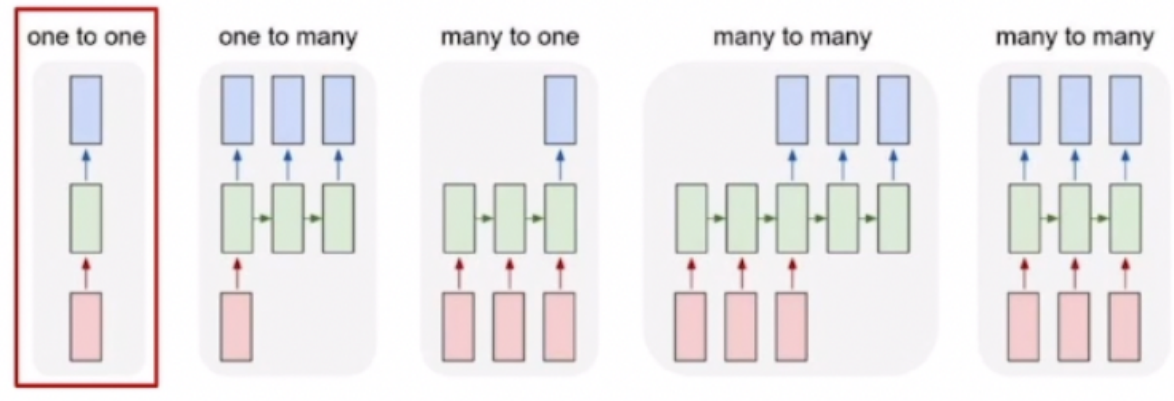
- t: 현재 time step
- h_{t-1}: 이전 time step $t-1$에서 계산된 hidden state 벡터
- x_t: time step t에서의 입력 벡터
- h_t : 현재 time step t에서의 hidden state 벡터
- f_w: W를 파라미터로 갖는 함수
  - W: RNN 모듈에 필요한 linear transformation matrix
- y_t: h_t를 통해 계산된 time step t에서의 출력값
  - 문제에 따라 매 time step마다 계산해야 할 수도 있으며, 최종 출력값만 필요한 경우도 있음

RNN 모듈을 정의하는 파라미터 W는 모든 time step에서 동일한 값을 공유한다는 것이 RNN의 가장 중요한 특징

🎯 How to Calculate the Hidden State of RNNs

1. 식 $h_t = f_W(h_t - 1,\ x_t)$를 통해 매 time step마다 hidden state를 계산함
   - 구체적으로는, W와 입력값 $(x_t,\ h_{t-1})$을 곱하고 $\tanh$를 취해 h_t를 계산함
   - 수식으로 표현할 경우 $h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$
2. 구해진 h_t를 통해 y_t 값을 계산함
   - 수식으로 표현할 경우 $y_t = W_{hy}h_t$

2️⃣ Types of RNNs



1	one to one	키, 몸무게, 나이 등을 입력받아 저혈압/고혈압으로 분류하는 task
2	one to many	이미지를 입력받아 그에 대한 설명을 생성하는 task (i.e. 이미지 캡셔닝)
3	many to one	문장을 입력받아 긍/부정으로 분류하는 task (i.e. 감성 분석)
4	many to many	문장/문서를 입력받아 번역된 문장을 출력하는 task (i.e. 기계 번역)
5	many to many	비디오 분류와 같이 영상의 프레임 레벨에서 예측하는 task 혹은 문장 내 단어들의 품사를 판별하는 task (i.e. POS 태깅)

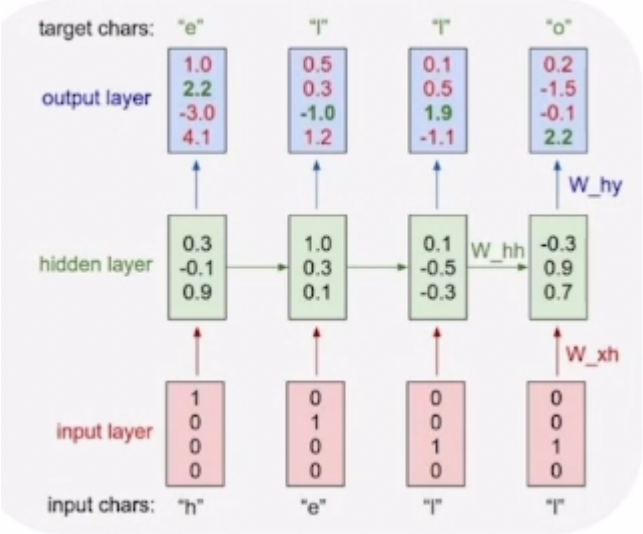
Character-level Language Model

1️⃣ Character-level Language Model

- 언어 모델(Language Model): 이전에 등장한 문자열을 기반으로 다음 단어를 예측하는 모델
- 캐릭터 레벨 언어 모델(Character-level Language Model): 문자 단위의 언어 모델로, 특정 문자의 다음에 올 문자를 예측함

🎯 Example of training sequence “hello”

- 주어진 단어가 “hello”라면, “h”가 주어졌을 때 “e”를, “e”가 주어졌을 때 “l”을, “l”이 주어졌을 때 “l”을, 다시 “l”이 주어졌을 때 “o”를 예측하도록 hidden state가 학습되어야 함

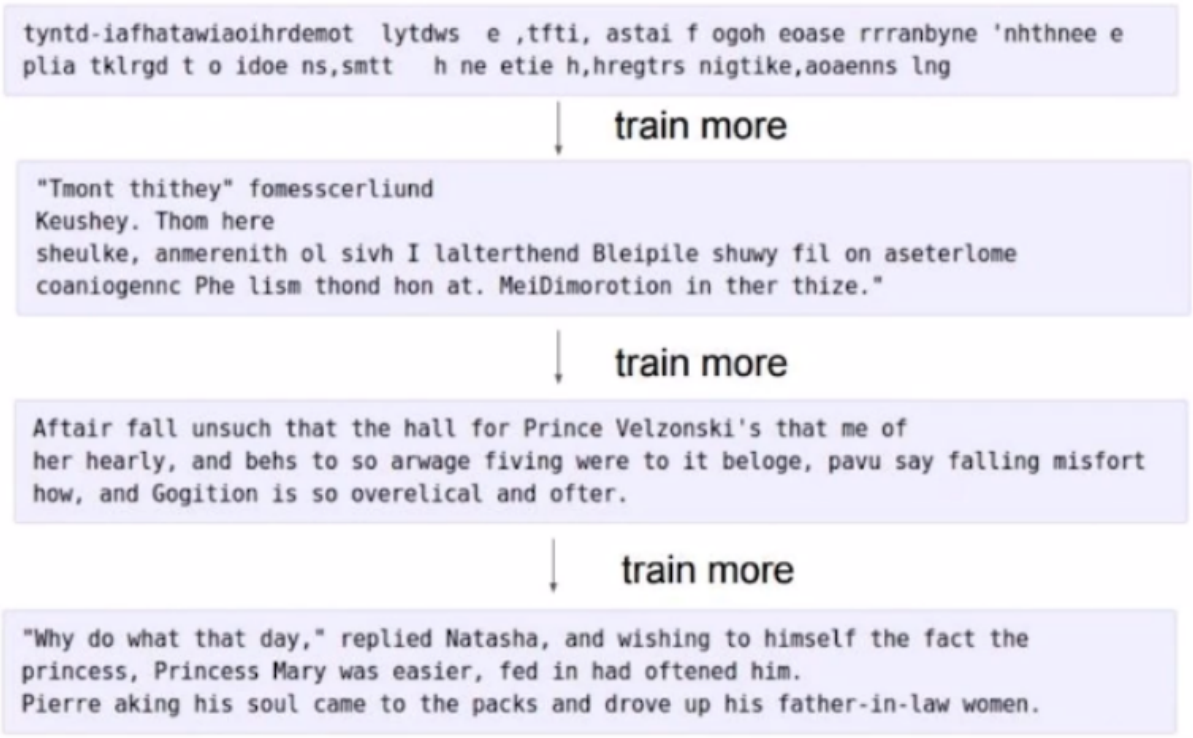


🎯 Logit

- softmax layer를 통과시키기 전의 값으로, 매 time step별로 output layer를 통해 나오는 벡터
  - 벡터의 크기는 unique한 문자의 수와 같으며, 이 경우 4가 됨(”h”, “e”, “l”, “o”)
  - softmax layer를 통과시키면 one-hot 벡터 형태로 출력됨

2️⃣ Applications of RNN

- 더 많은 학습이 진행될수록 완전한 형태의 문장을 출력하는 것을 확인할 수 있음



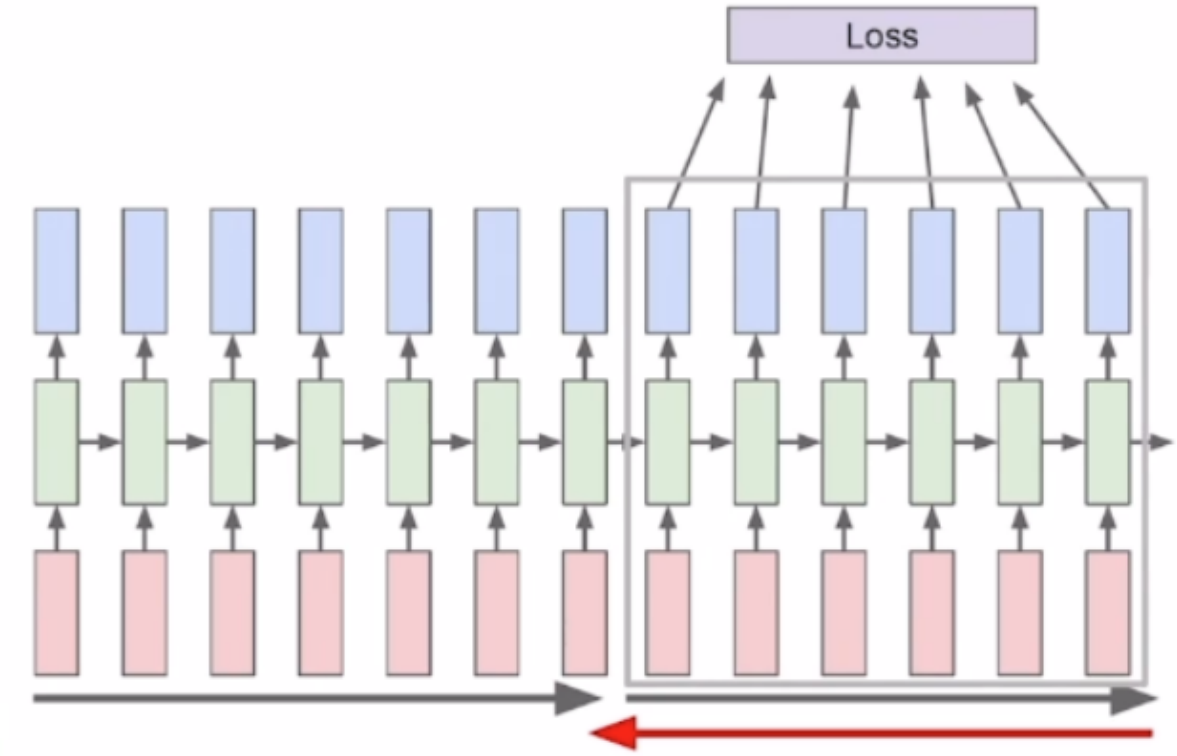
- 이전 time step까지의 주식값을 활용하여 다음날 주식값을 예측하는 형태의 task에도 활용 가능함
- 인물별 대사, Latex로 쓰여진 논문, C언어 코드 등으로부터 다양한 언어적 특성을 학습하여 텍스트를 생성할 수 있음

Backpropagation through time and Long-Term-Dependency

1️⃣ Truncation

🔹 제한된 리소스 내에서 학습할 수 있는 시퀀스에 한계가 있으므로, Truncation을 활용하여 이를 해결할 수 있음

- 다음과 같이 학습 데이터의 시퀀스를 잘라 학습에 사용하게 됨

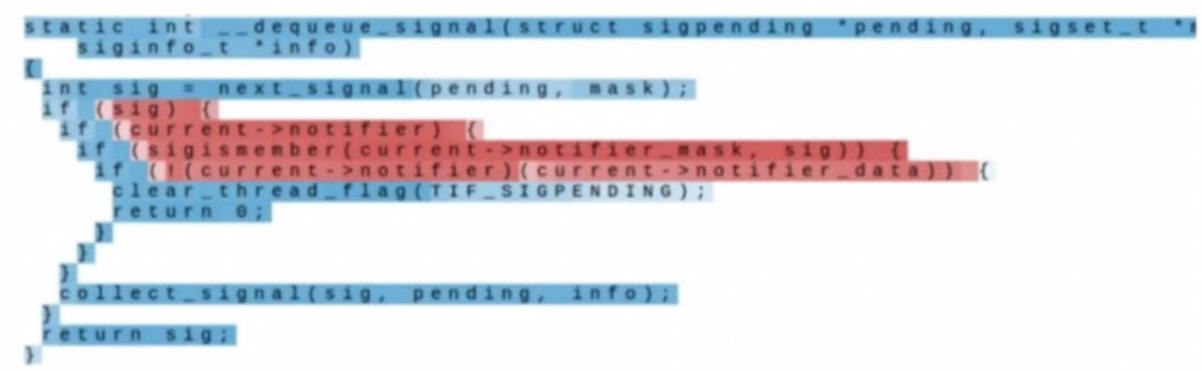


2️⃣ Backpropagation through time (BPTT)

💠 BPTT는 RNN이 backward propagation을 통해 매 time step마다 계산된 weight를 학습하는 방식

- forward propagation을 통해서는 W를 계산하며,
  backward propagation을 통해서는
  W의 gradient를 계산하여 학습에 이용하게 됨
- 다음은 특정 hidden state를 시각화한 그림으로, BPTT를 반복하게 되면 다음과 같이 빨강(긍정)과 파랑(부정)으로 해당 time step에서의 중요한 부분을 잘 학습하는 것을 확인할 수 있음

- 해당 시각화는 LSTM 모델로 학습한 결과로, Long-Term-Dependency 문제를 보완하여 학습이 잘 이루어졌음



3️⃣ Long-Term-Dependency

- 🔷 Vanilla RNN에서는 gradient가 전파되면서 소실/증폭되어 멀리까지 학습 정보를 잘 전달하지 못하게 됨 (Vanishing/Exploding Gradient Problem)

🎯 Toy Example: time step=3인 RNN의 BPTT 과정

Toy Example

- $h_t = \tanh(w_{xh}x_t + w_{hh}h_{t-1} + b), t = 1,2,3$
- For $w_{hh} = 3, w_{xh} = 2, b = 1$

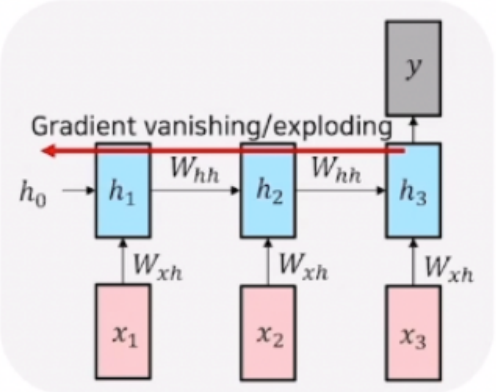
$h_3 = \tanh(2x_3 + 3h_2 + 1)$

$h_2 = \tanh(2x_2 + 3h_1 + 1))$

$h_1 = \tanh(2x_1 + 3h_0 + 1))$

...

$h_3 = \tanh(2x_3 + 3\tanh(2x_2 + 3h_1 + 1) + 1)$



- $t = 3$에서의 hidden state h_3을 h_1로 표현하면 다음과 같음

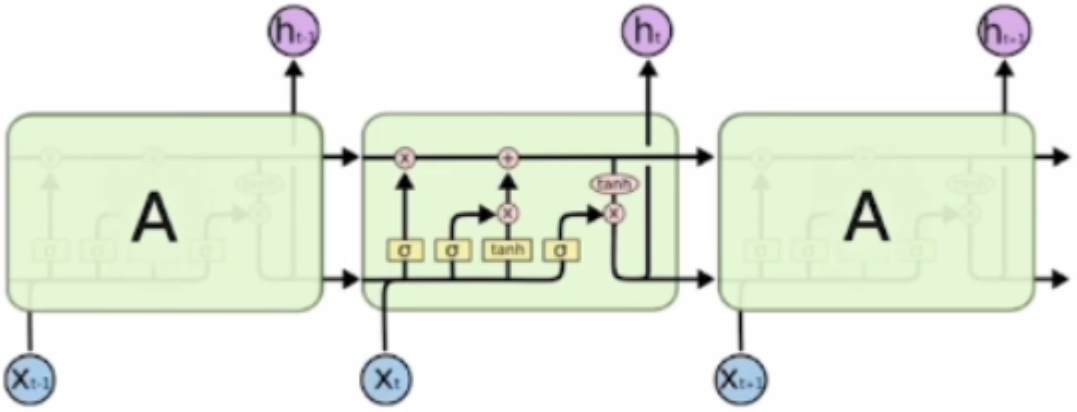
$$h_3 = \tanh(2x_3 + 3\tanh(2x_2 + 3h_1 + 1) + 1)$$

- BPTT를 통해 gradient를 계산하면 chain rule에 의해 $\tanh$ 함수 내에서 3이 속미분되어 도출됨
- 시퀀스가 길어질수록 미분값은 기하급수적으로 증가하며, 속미분되어 나오는 W의 값이 1보다 작다면 미분값은 기하급수적으로 작아지게 됨

Long Short-Term Memory (LSTM)

1️⃣ Long Short-Term Memory (LSTM)

- 🔹 중심 아이디어는 이전 시퀀스 정보를 단기 기억으로 저장하여 때에 따라 꺼내 사용함으로써 더욱 오래 기억하도록 개선하는것

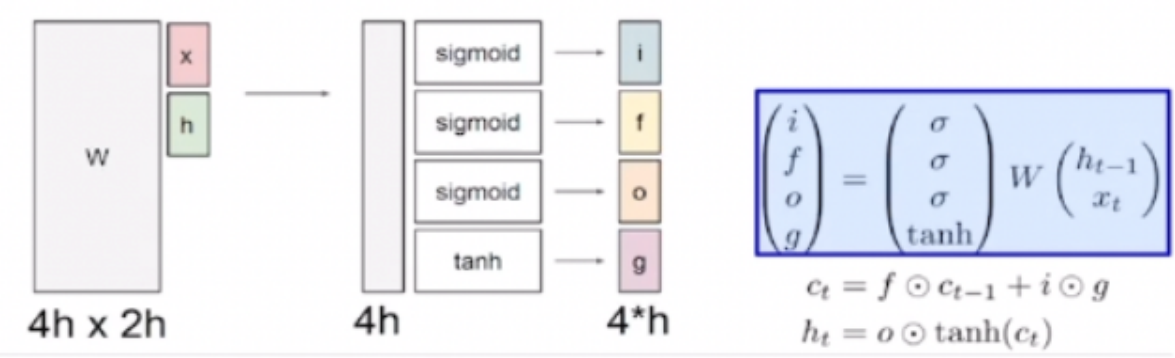


The repeating module in an LSTM contains four interacting layers.

- Cell state: 핵심 정보를 담음
- 필요할 때마다 hidden state를 가공해 특정 time step에 필요한 정보만 노출하는 형태로 정보가 전파되며, 이때 gate를 이용함

2️⃣ Gates of LSTM

- input으로 x_t, h_{t-1}를 받아 이를 W에 곱한 후 비선형 함수에 넣어줌



- i: Input gate로, 정보를 cell에 쓸지 여부를 결정함
  - sigmoid를 통해 출력을 0에서 1 사이의 값으로 표현하며, 이는 cell state와 hidden state 두 갈래로 흐르게 됨

$$sigmoid(W(x_t,\ h_{t-1}))$$

- f: Forget gate로, cell 정보를 지울지 여부를 결정함
  - sigmoid를 통해 출력을 0에서 1 사이의 값으로 표현함

$$sigmoid(W(x_t,\ h_{t-1}))$$

- o: Output gate로, hidden state에서 cell 정보를 어느 정도 사용해야 할지 0에서 1 사이의 값으로 표현함

$$sigmoid(W(x_t,\ h_{t-1}))$$

- g: Gate gate로, 정보를 cell에 어느 정도 반영해야 할지를 -1에서 1 사이의 값으로 표현함

$$\tanh(W(x_t,\ h_{t-1}))$$

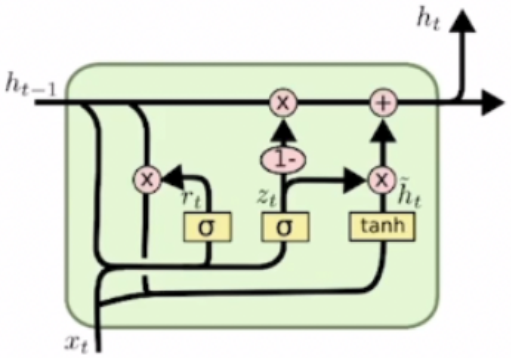
🎯 LSTM vs. RNN

- LSTM은 각 time step마다 필요한 정보를 단기 기억으로 hidden state에 저장하여 관리되도록 학습함
- 오차역전파(backpropagation) 진행 시 단순히 가중치 W를 계속 곱해주는 RNN과는 달리, LSTM은 Forget gate를 거친 값(필요로 하는 정보)을 덧셈 연산에 사용하여 그래디언트 소실/증폭 문제를 방지함

3️⃣ Gated Recurrent Unit (GRU)

Cell state와 Hidden state를 일원화하여 LSTM을 경량화한 모델

- $z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$
- $r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$
- $\tilde{h_t} = \tanh(W \cdot [r_t \cdot h_{t-1}, x_t])$
- $h_t = (1 - z_t) \cdot h_{t-1} + z_t \cdot \tilde{h_t}$
- c.f) $C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C_t}$
  in LSTM



- GRU에서 사용되는 h_{t-1}은 LSTM에서의 c_t와 비슷한 역할을 함
- Forget gate 대신 $(1 - input\ gate)$를 사용하여 h_t를 구할 때 가중평균의 형태로 계산하게 됨
- 계산량과 메모리 요구량을 줄인 동시에 성능 또한 LSTM과 비슷하거나 더 좋음

🚀 Summary

- RNN은 들어오는 입력값에 대하여 많은 유연성을 가지고 학습되는 딥러닝 모델
- RNN은 그래디언트 소실/증폭 문제로 인해 실제로는 많이 사용되지 않지만, 같은 계열의 LSTM과 GRU 모델은 현재도 많이 사용됨
- LSTM과 GRU 모델은 가중치를 곱하는 RNN과 달리 덧셈을 통한 그래디언트 복사를 사용해 그래디언트 소실/증폭 문제를 해결함

출석퀴즈 오답노트

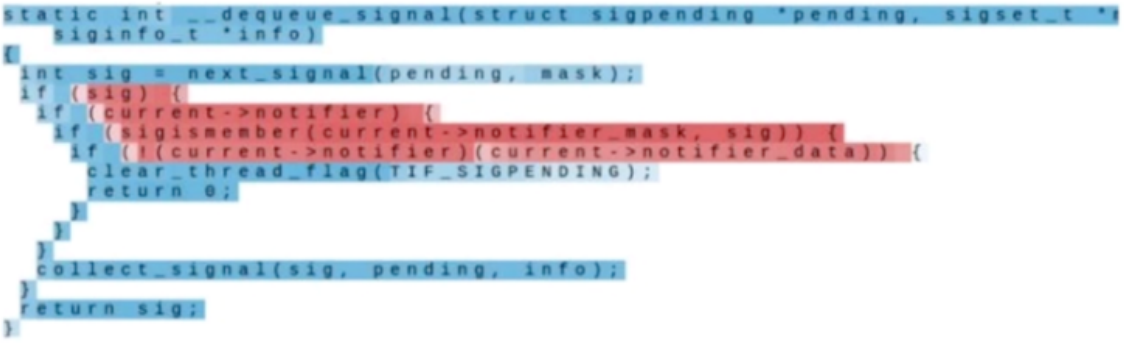
▼ 5. 두 가지 형태의 'many to many' 타입 RNN 모델로 해결할 수 있는 태스크

- 기계 번역과 같이 입력값을 끝까지 다 읽은 후, 번역된 문장을 출력해주는 태스크
- 비디오 분류와 같이 영상의 프레임 레벨에서 예측하는 태스크
- 각 단어의 품사에 대해 태깅하는 POS와 같은 태스크

▼ 6. Character-level Language Model에 대한 설명으로 옳지 않은 보기

- 언어 모델 중 하나로, 다음에 올 문자를 예측하는 태스크를 수행한다.
- Hidden state를 사용하여 이전 문자열의 정보를 기반으로 다음 문자를 예측한다.
- Output layer를 통해 원-핫 벡터 형태의 출력값이 나오게 된다. (O)
- LSTM과 같은 모델을 사용하여 Long-Term-Dependency 문제를 극복한다. (X)

▼ 10. 다음 사진에 대한 설명으로 옳지 <u>않은</u> 보기



- 특정 hidden state 벡터의 dimension을 시각화한 결과이다. (X)
- Vanila RNN으로 학습했을 때 흔히 볼 수 있다. (O)
- true/false를 detection하는 태스크를 수행한 결과이다. (O)
- Character-level Language Model의 한 cell을 거쳤을 때는 볼 수 없는 결과이다. (O)