AI 인공지능 코딩 구현 홀로서기 4기

[2주차, 주말 수업] Seq data, Word Embedding, RNNs

정 정 민

강의 목차

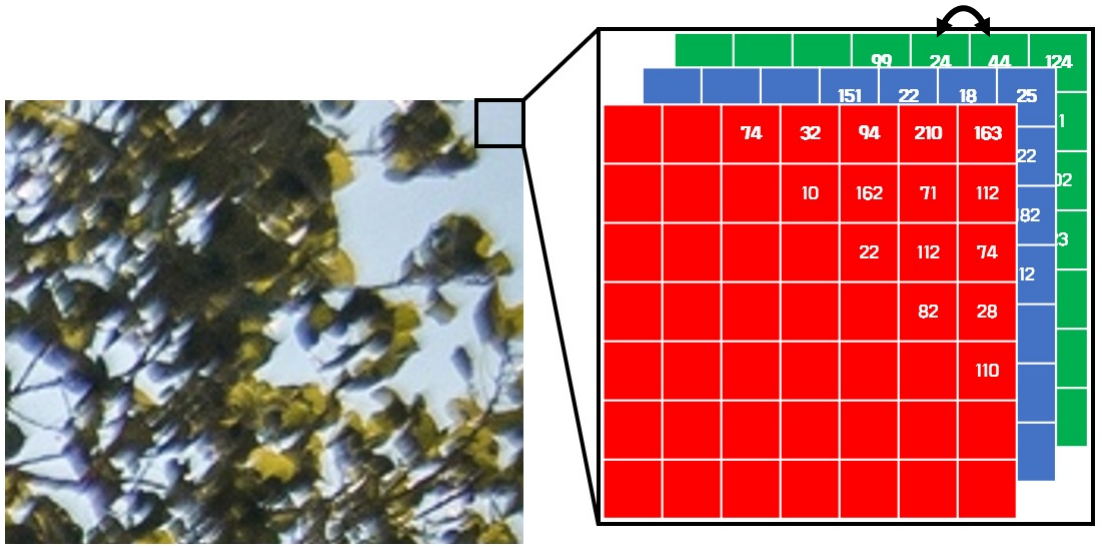
1. Sequential Data
2. Word Embedding
3. RNN
4. LSTM, GRU

Sequential Data

이미지 : Non-sequential data

- 이미지를 구성하는 기본 단위의 픽셀(pixel)은 서로들 독립적이며,
- 공간적인 구조 내에 위치하는 것을 제외하면 특정한 순서에 묶여 있지 않음 (non-sequential)
  - 구조적 특징을 파악하는 모델 : CNN
- 또한, 시간의 흐름이 존재 x, 순서가 바뀌어도 의미 유지
- 이러한 특징을 갖는 모델을 '정적 데이터', '비-순차 데이터' 라고 함

- 이웃 픽셀과 동시간에 촬영,
- 순서에 따른 강제성 x



정적 데이터

순차 데이터 : Sequential Data

- 딱 하나의 데이터를 기준으로 할 때,
- 데이터를 이루는 구성 요소들이 순서를 갖고 있는 경우
- 이 순서는 구성 요소들끼리 순서를 바탕으로 영향력을 주고 받고 있음
  - 순서가 바뀌면 의미가 달라지거나 손실됨
- 시계열 데이터 (time-series data)라고 부름

아침 먹고 학교를 갑니다

- 정해진 순서가 있고
- 시간 및 순서의 의존성을 학습

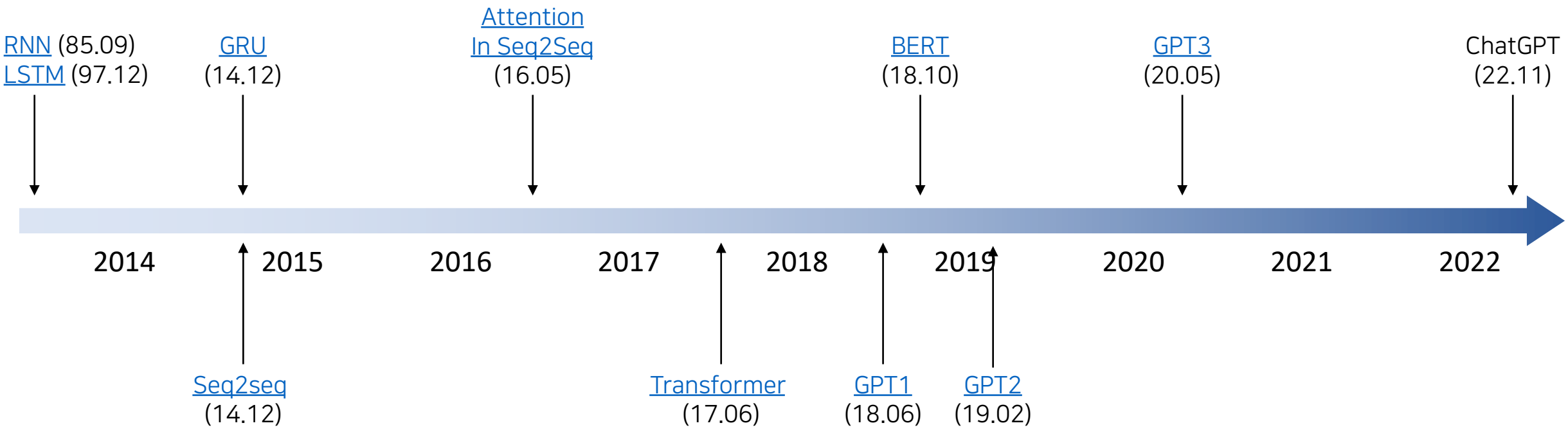
순차 데이터

NLP

- Natural Language Processing : 자연어 처리
- 순차 데이터인 Text를 다루는 분야
- 오래된 연구 역사 (from 1, 2차 세계전쟁. 암호 해독)

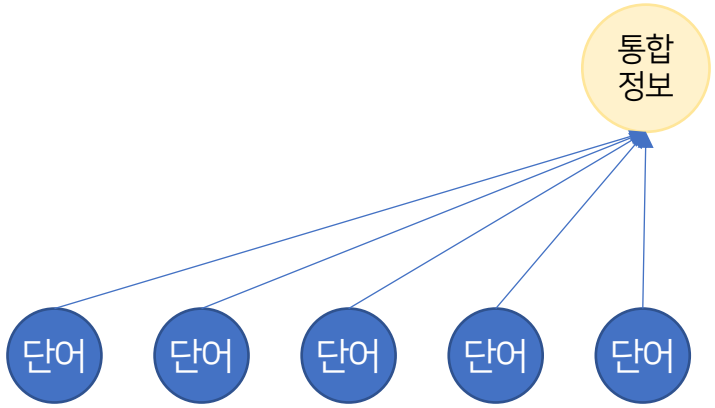
NLP 흐름

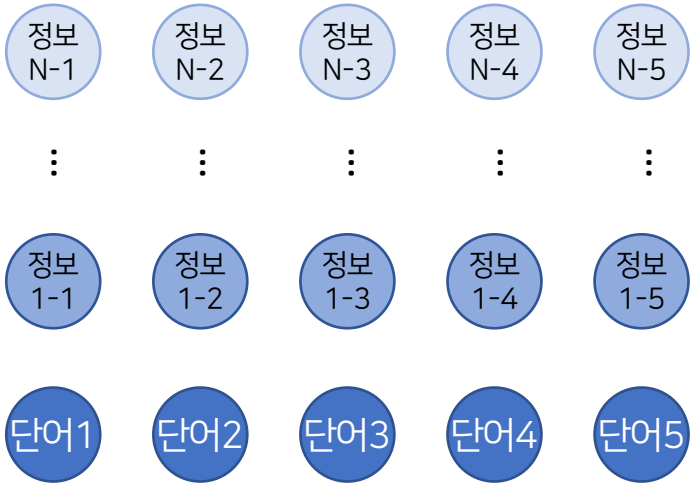
- 순차 데이터를 다루기 위한 순환 구조 네트워크 (RNN, LSTM, GRU 등)
- Attention 메커니즘 & Transformer
- BERT & GPT : Transformer 기반의 후속 모델



NLP 에서 하려고 하는 것

- ‘글’ ↔ ‘정보‘
- 글에 대한 새로운 표현
- 순환 신경망 시절에는
  - 새롭게 입력되는 단어들로 전체 글을 아우르는 하나의 정보 덩어리를 만들려 했고
- Attention 시절(요즘)에는
  - 단어 각각을 새로운 정보의 표현으로 생성
  - 풀려고 하는 문제에 도움이 되는 정보 추출을 진행
  - 통합 정보 필요시 통합 진행

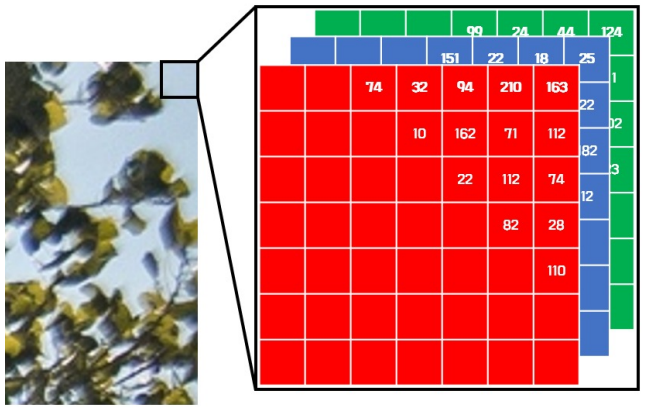




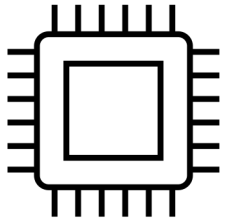
Word Embedding

이미지를 다루는 과정에서는 하지 않던 고민

- 이미지가 어떻게 모델로 들어갔을까?
  - 이미지 = 숫자 → `torch.tensor`변환 가능
- 단어는 어떻게 모델로 들어갈까?
  - 인공지능 모델이 “홀로서기”라는 한글 덩어리를 받아들일 수 있을까?
  - 단어를 숫자(모델이 받는 형태)로 변경할 필요가 생김 → Work Embedding
    - “Word를 숫자로 Embedding 시킨다”

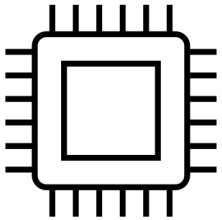


Ok!



단어가 어떻게 네트워크로 들어가?

Ok??



Word Embedding 이란?

- 단어를 표현하기 위해
- 이를 특정 차원의 실수 vector로 매핑하는 과정

- 기왕 숫자로 변환하는거 잘 변환하면 좋을텐데..
  - 잘 변환한다 → 비슷한 의미를 갖는 단어는 비슷한 실수값을 갖는다

- Word Embedding을 하기 위해서는 Tokenize 과정이 선행되어야 함

Tokenize

- Token 이란 Sequential Data에서 분석의 단위가 되는 요소를 의미함
- 원 데이터를 Token으로 잘라야 함

- NLP의 경우
  - 가장 초기에 고민 & 시도했던 Tokenize 방법은 '띄어쓰기'로 쪼개는 방법
  - OOV(Out of Vocabulary) 문제로 좀 더 효율적고 다양한 방법이 제안
    - BERT : Word Piece – 참고
    - CLIP : Byte-Pair Encoding (BPE) – 참고

Tokenize BERT & CLIP

- 공유 코드를 보면서 결과를 비교해봅시다

- BERT
  - `tokenizer = BertTokenizer.from_pretrained('bert-base-multilingual-cased’)`
  - `tokenized_text = tokenizer.tokenize(text)`

- CLIP
  - `st = SimpleTokenizer()`
  - `tokenized_text = [st.decoder[en] for en in st.encode(text)]`

Word Embedding 방법들

- 통계적인 방법 (지금은 거의 쓰지 않아요)
  - One-hot encoding : 희소 표현 방식
  - TF-IDF : 횟수 기반
- 예측 방법 (주변 단어들로 중심 단어 예측, 종종 씁니다)
  - Word2Vec : Local한 주변 단어만 활용
  - GloVe : Global한 단어, 그리고 이들의 발생 통계 정보도 활용
  - FastText : 단어를 글자 단위로 쪼개어 n-gram화 → subword 단위 tokenize
    ex) car → c, a, r

띄어쓰기 단위 tokenize

- 최신 방법 : 인공신경망을 활발히 활용
  - BERT → Word piece tokenize
  - CLIP → BPE tokenize

Word Embedding 적용 과정

수학의 정석 → W-2 상단 23

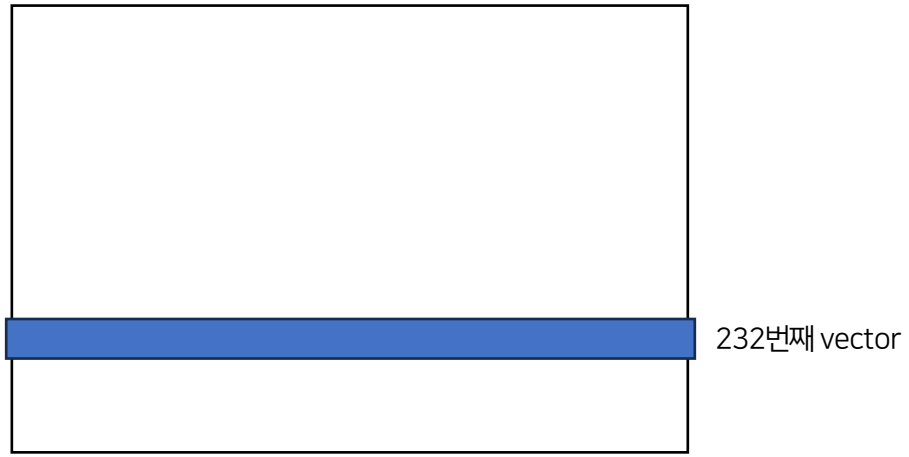
미리 정의한
책 – 위치 pair

→

"인공지능" → 232

미리 정의한
단어 – 인덱스 pair

→



학습된 embedding matrix (look-up table)
학습 모델, 방식에 따라 상이함 (W2V, BERT, CLIP)

그래서 뭘 써야 하나요??

- 정답은 없음

- 최신 방법들 중 풀려고 하는 문제에 더 잘 맞는 방법을 사용
  - W2V, GloVe, FastText 보다 최신 방법의 성능이 더 좋음

- 추천하는 방식은 있나요??
  - 단순 NLP의 문제를 푼다면 BERT 먼저 고려해 보시고
  - 멀티 모달 문제를 푼다면 CLIP도 고려해보기

RNN

사람은 어떻게 읽을까?

- “아침 먹고 학교를 갑니다”

- 우리 사람이 문장을 읽고 이해하는 과정을 생각해볼까요?
  (1) 문장의 앞에서부터 한 단어 씩 읽음
  (2) 새로운 단어가 들어오면 이를 읽고 이해함
  (3) 앞서 해석해 만들어낸 기억에 정보를 추가하여 기억을 업데이트
  (4) 문장이 끝나는 시점까지 위 과정을 반복

- 참고 : 링크

사람의 입장에서 RNN으로!

- 사람의 입장은
  (1) 문장의 앞에서부터 한 단어 씩 읽음
  (2) 새로운 단어가 들어오면 이를 읽고 이해함
  (3) 앞서 해석해 만들어낸 기억에 정보를 추가하여 기억을 업데이트
  (4) 문장이 끝나는 시점까지 위 과정을 반복

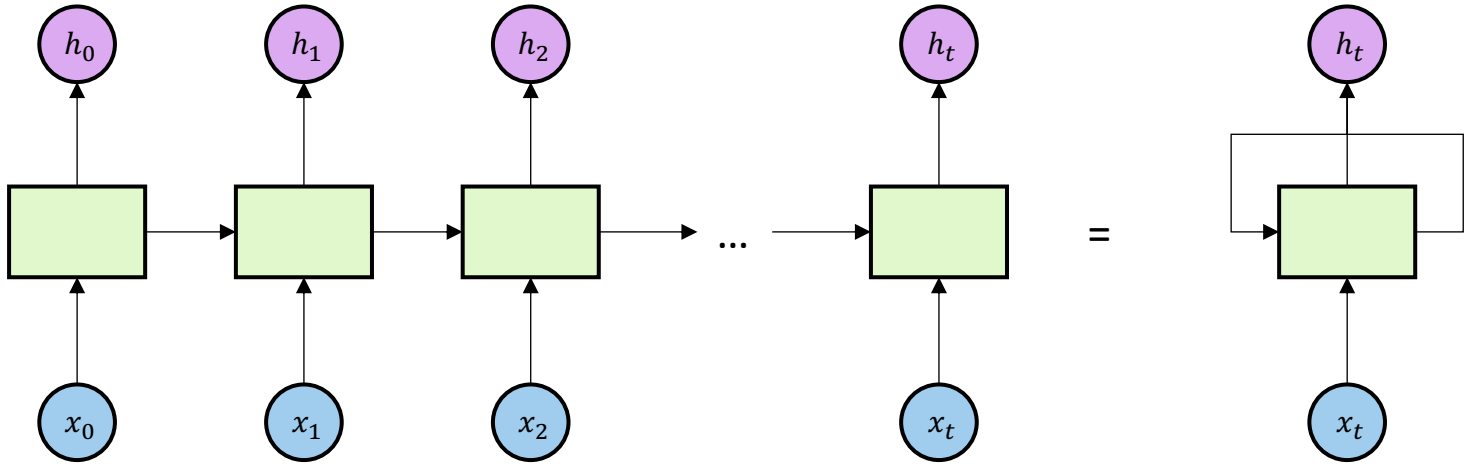
- 위 과정을 딥러닝 RNN에서는 이렇게 동작
  (1) 문장의 앞에서부터 한 단어 씩 입력으로 받아들임
  (2) 새로운 단어가 들어오면 이를 처리해서 정보를 추출
  (3) 앞서 해석한 만들어낸 정보 덩어리에 추출한 정보를 추가하여 정보를 업데이트
  (4) 문장이 끝나는 시점까지 위 과정을 반복 (Recurrent)

- RNN : Recurrent Neural Network

RNN : Recurrent Neural Networks

- 순차 데이터를 처리하기 위해 고안된 모델
- 순차 데이터에서 중요한 순서 정보를 처리하기 위해 이웃한 데이터 간의 연관성을 표현

- 사람 입장의 '기억'을 담당하는 정보 덩어리인 'hidden state'가 존재
- 새로운 입력이 들어오면 이 hidden state를 조금씩 수정



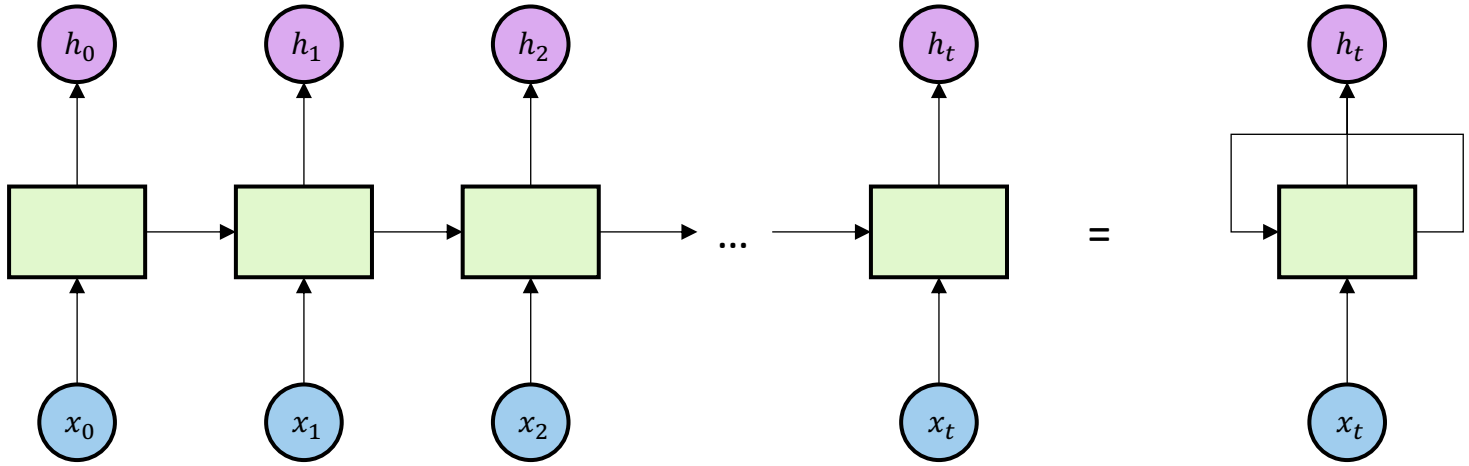
Unfold 방식

Unfold 방식

RNN : Recurrent Neural Networks

- 아래 도식은 RNN 모델에만 국한된 그림이 아님
- 중간의 초록색 사각형을 어떤 방식으로 모델링 하는지에 따라 RNN 계열의 다른 네트워크가 됨
- 즉, 공통적으로 RNN 류의 모델은 추가된 입력과 과거의 정보를 바탕으로 새로운 정보를 생성하는 목적을 갖음

$$h_t = f(x_t|h_{t-1})$$



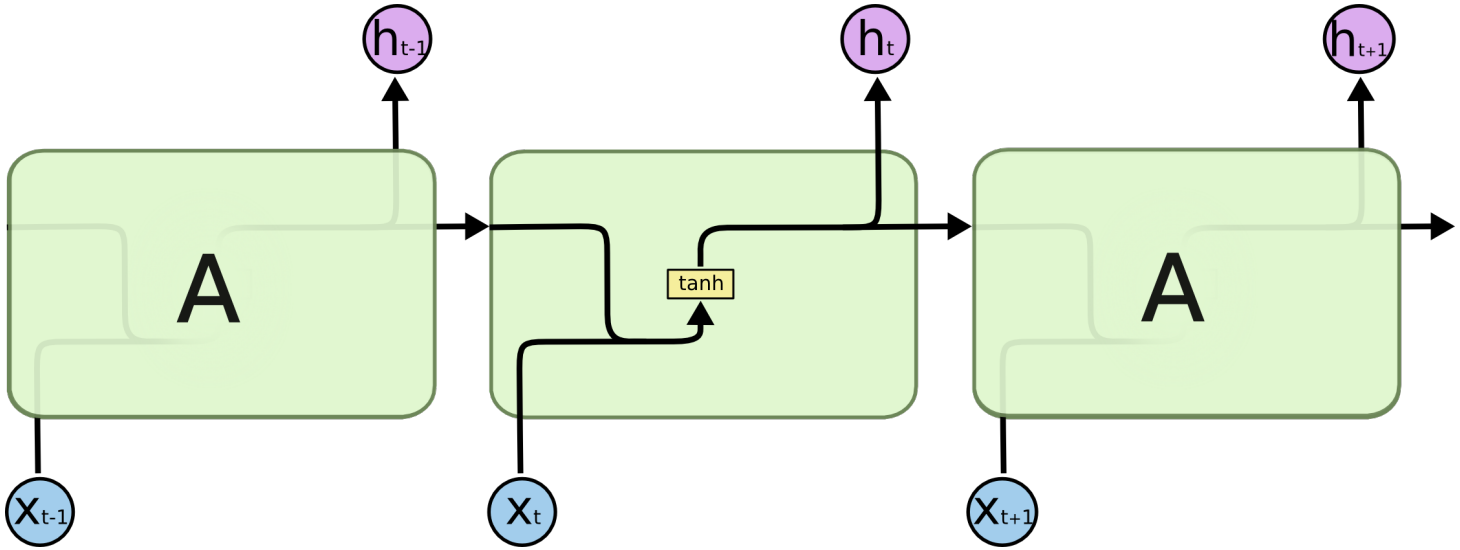
Unfold 방식

Fold 방식

RNN : Recurrent Neural Networks

- 기본 RNN은 2개의 가중치(W_x, W_h)를 바탕으로
- 입력과 이전 정보를 결합하고
- 활성 함수인 tanh(링크)를 적용해 새로운 정보를 생성

$$h_t = f(x_t|h_{t-1}) = \tanh(W_{xx}x_t + W_{xh}h_{t-1})$$



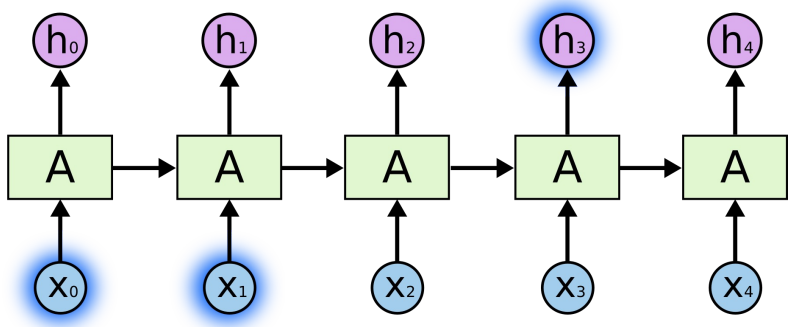
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM, GRU

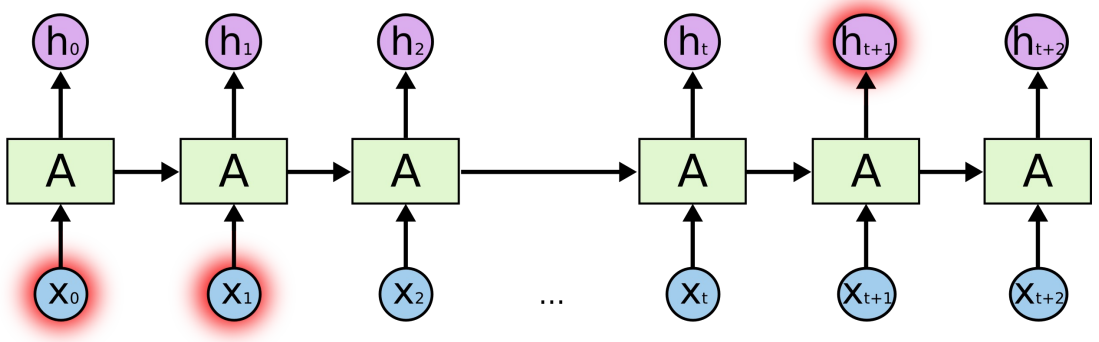
RNN의 문제점

- Long-Term Dependency 문제
  - '기억'에 해당하는 'hidden state'에 담긴 정보가 희석되는 문제
  - 필요한 정보가 먼 과거인 경우, 정보 사이의 격차가 커져 과거 정보를 활용할 수 없음

- 이를 위해 LSTM이 제시됨



필요한 정보의 위치가 가까운 경우



필요한 정보의 위치가 먼 경우

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

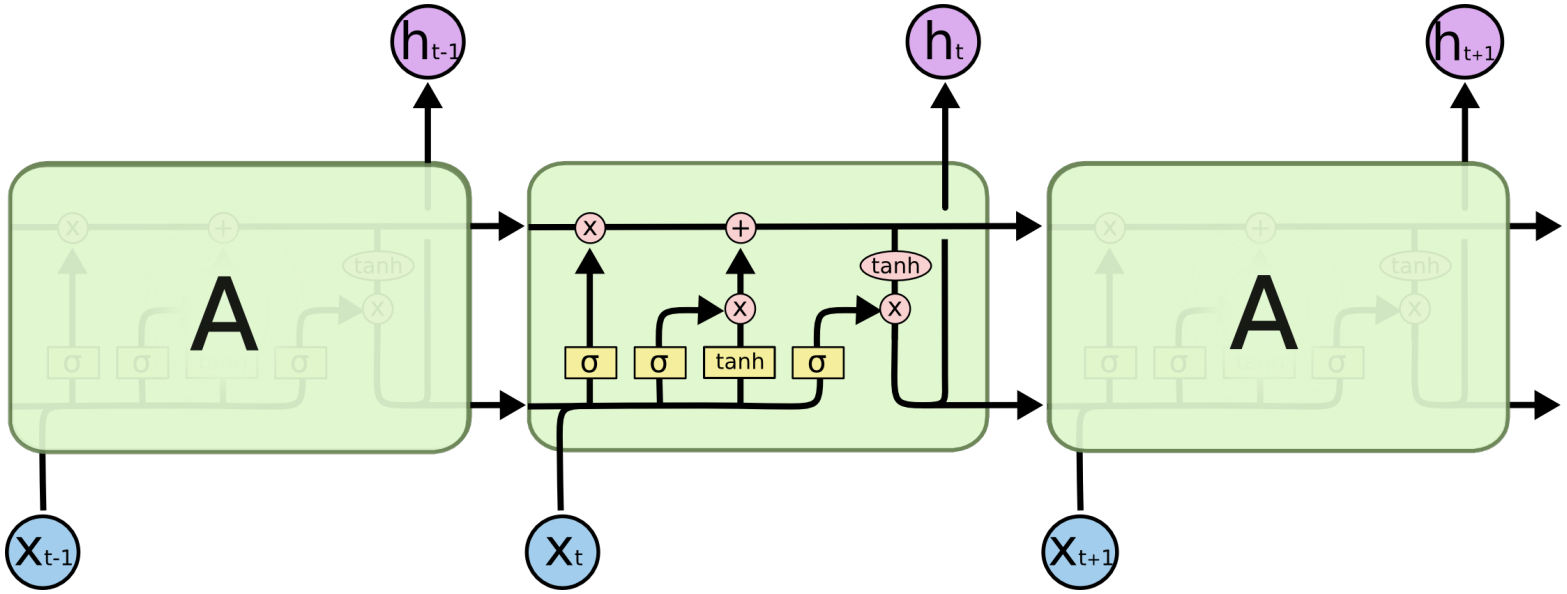
LSTM : Long Short Term Memory networks

- Long-Term Dependency 문제에서 비교적 자유로움
- 일반적인 RNN 보다 성능이 좋음 (“RNN을 씁니다” → LSTM 을 썼구나.. 라고 생각하면 됩니다)

- 사람의 ‘기억’에도 ‘장기기억’과 ‘단기기억’이 있으니 이를 모방하자!
- 단순히 ‘hidden state’만 사용하던 것을
  - 단기기억 역할을 하도록 hidden state를 사용하고
  - 장기기억 역할을 하도록 cell state를 사용하자

- 장기기억을 담당하는 cell state는 매우 중요한 역할을 담당하니
- 정보가 들어가고, 나오고 그리고 사용되는 시점에는 문지기인 gate를 두고 이를 통제하자

LSTM 도식화



: 네트워크 layer

: vector 내 요소 별 연산
element-wise operation

: tensor(vector)의 이동

: vector의 concatenate

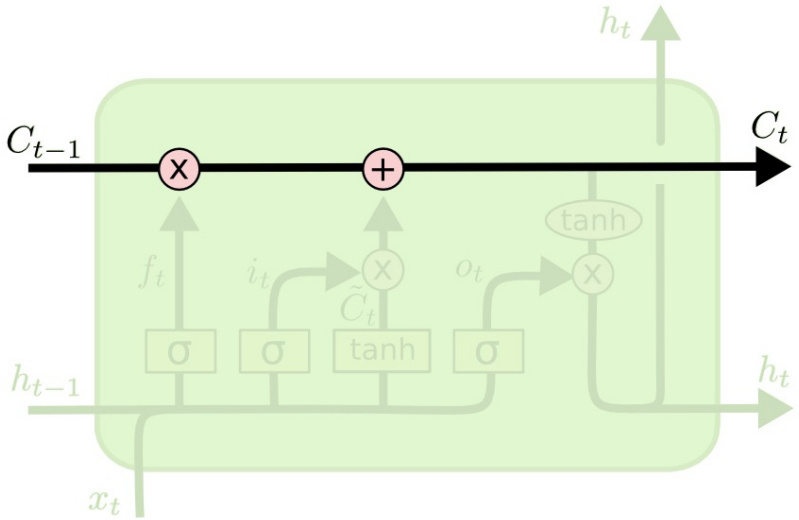
: copy & 분기

σ : sigmoid 연산 (결과 : 0~1)

tanh : tanh 연산 (결과 : -1~1)

Cell State

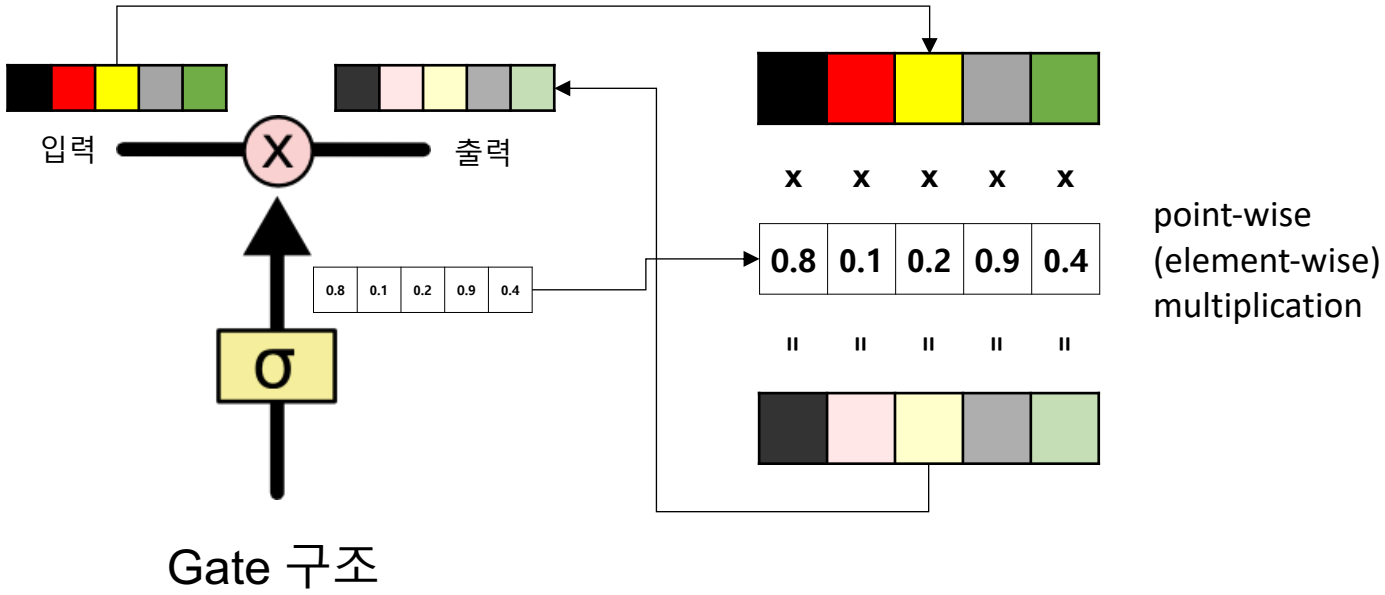
- 전체 time step을 관통하는 정보 vector
- '장기기억' 처럼 오랫동안 기억될 중요 정보를 담고 있으며,
- Gate 구조에 의해 정교하게 정보가 추가, 삭제, 활용 됨



https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Gate mechanism

- 정보 입장에서 문지기 역할
- 특정 값을 비율에 따라 차등적으로 통과시키면서 정보를 걸러주는 역할
  - 비율이 1이라면 모든 정보 통과, 0이라면 모든 정보 Block
- 비율을 만들기 위해 sigmoid 함수를 활용
- LSTM에서는 3개의 Gate 구조가 사용됨



Gate 구조

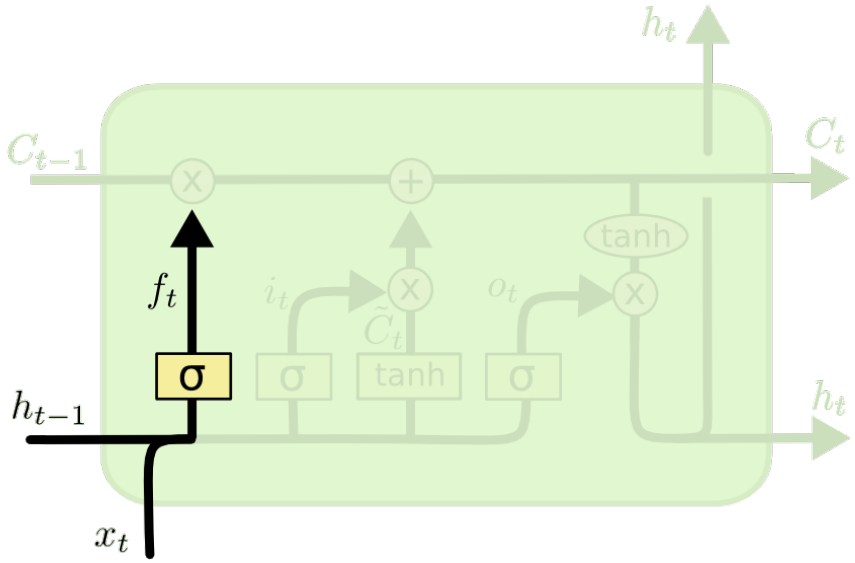
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Forget Gate

- Cell state에서 어떤 정보를 지울지를 결정
- 이전 시점의 단기 정보(h_{t-1})과 지금 시점의 단어(x_t)를 이용해 판단

[빈칸 채우기 문제로 예를 들면] He is my friend. I _____

- 이전 문장의 주어에 관한 인칭 정보 등을 제거



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

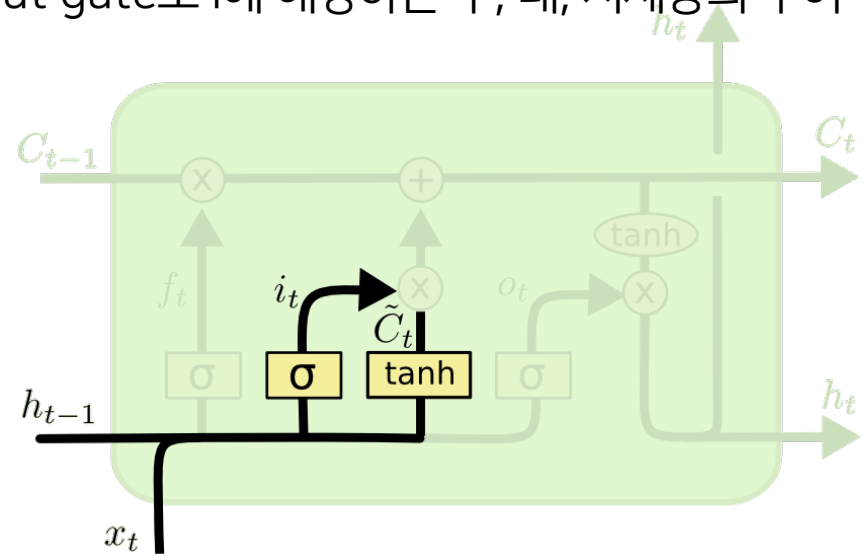
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Input Gate

- Cell state에서 어떤 정보를 넣어줄지를 결정
- 이전 시점의 단기 정보(h_{t-1})과 지금 시점의 단어(x_t)를 이용해 판단
- 후보자 Candidate($\tilde{C}_t$)를 만들어 input gate를 통과시킴

[빈칸 채우기 문제로 예를 들면] He is my friend. I _____

- Candidate로는 다양한 동사들이 될 것이고
- Input gate로 I에 해당하는 수, 태, 시제등의 주어 정보가 전달



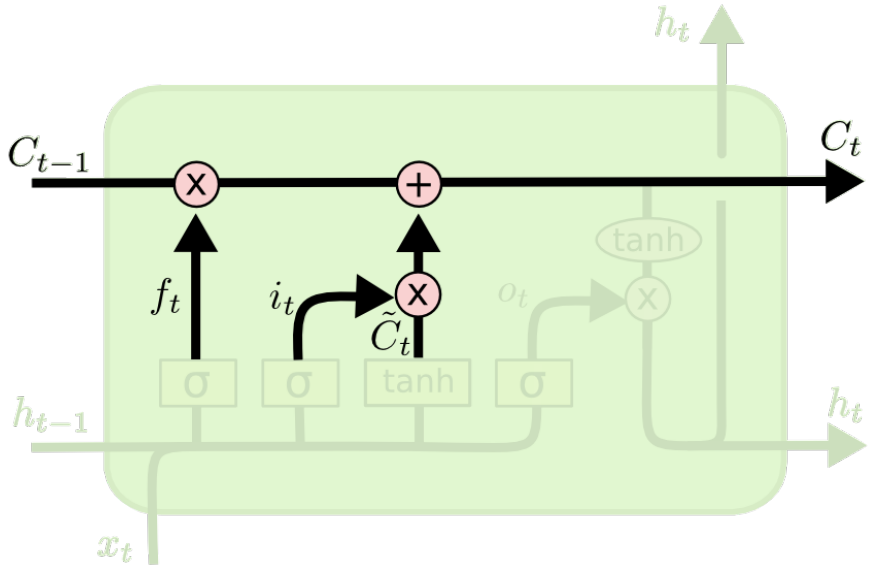
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Cell State Update

- Forget gate와 Input gate를 이용해 Cell state를 업데이트
- 적당한 정보가 없어지고, 적당한 정보가 들어감
- 새로운 Cell State 완성



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

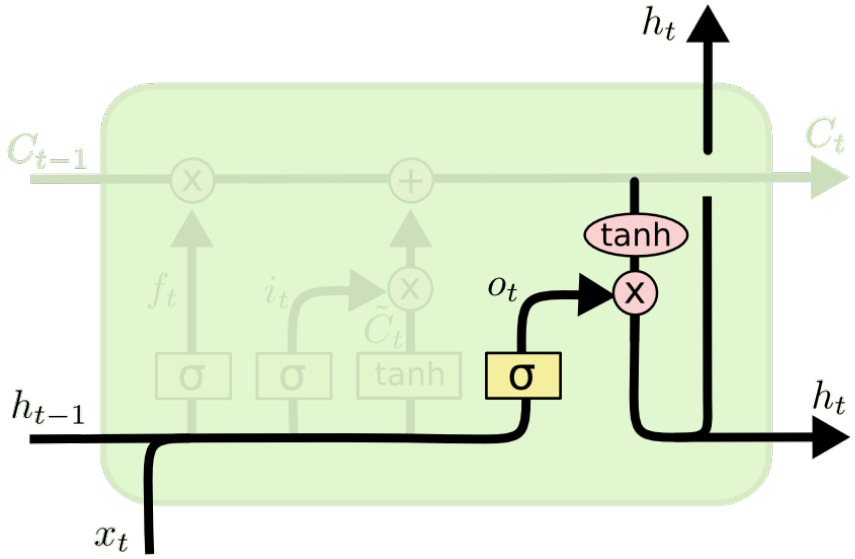
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Output Gate

- 현 시점의 단기 정보(h_t)를 만들기 위해 장기 기억인 Cell state에서 정보 추출
- 이 h_t는 이번 step의 출력으로 혹은 다음 step의 입력으로 사용됨

[빈칸 채우기 문제로 예를 들면] He is my friend. I _____

- 빈칸을 채우기 위한 정보를 Cell에서 추출하니 am 이 오는게 제일 적합하더라

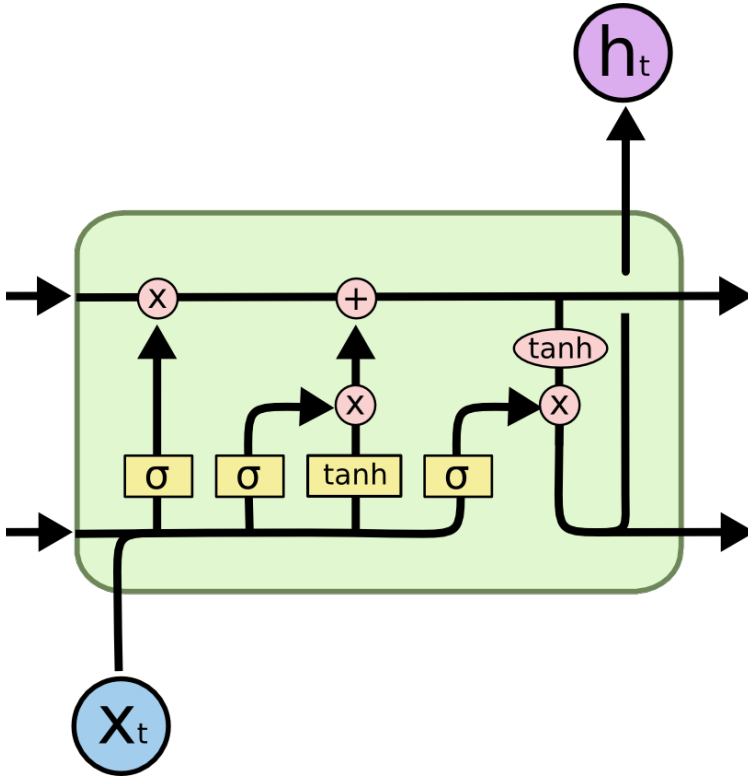


$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM



$$f_t = \sigma\left(W_f[h_{t-1}, x_t] + b_f\right)$$

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$

$$\tilde{C}_t = \tanh(W_c[h_{t-1}, x_t] + b_c)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

$$h_t = o_t * \tanh(C_t)$$

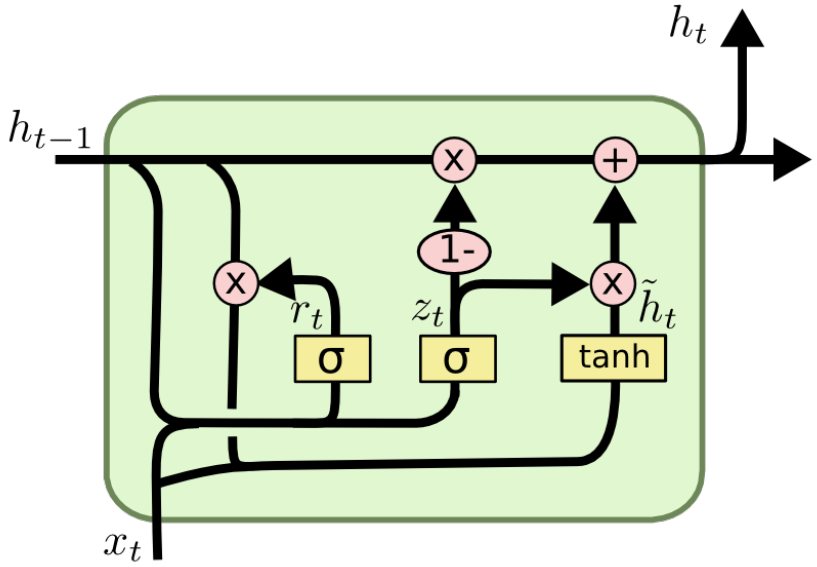
모두 같은 크기(n)의 vector

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

GRU : Gated Recurrent Units

- LSTM 보다 연산량을 줄여 모델 크기를 작게 만듦
  - 2개의 Gate만을 사용
  - 파라미터의 양도 줄임
- 또한 Cell state를 사용하지 않고 Hidden state만을 사용

- 다만, 기존 RNN보다 Long-Term Dependency 문제에 좀 더 자유롭도록 Hidden state를 잘 조절



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

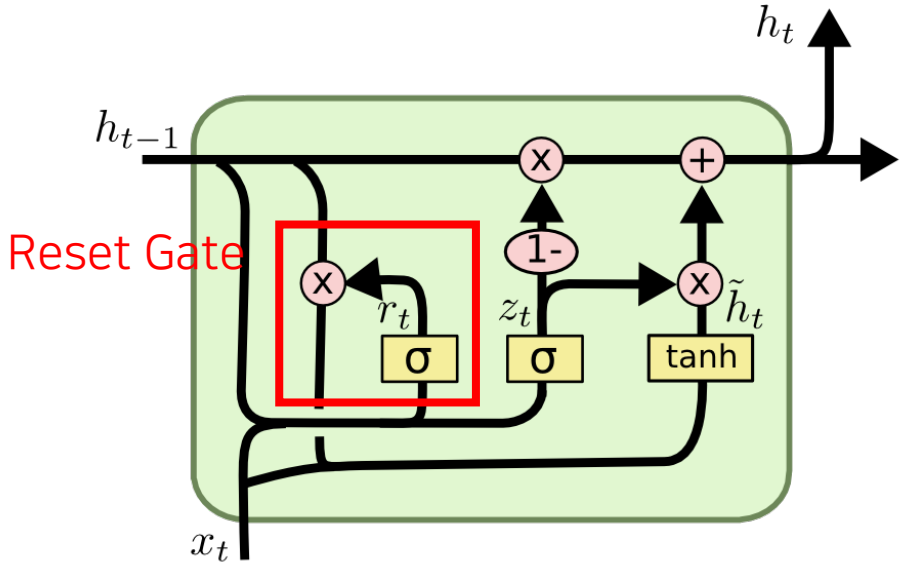
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Reset Gate

- 이전의 Hidden state로부터 새로운 Hidden state를 만드는 후보를 만드는 데 얼마나 영향을 미칠지 결정
  - 이전 Hidden state와 현 시점 입력 정보를 활용
- 영향력은 r_t로 표현하며 이전 hidden state vector에 element-wise 연산 진행
- 영향력이 적용된 이전 hidden state는 새로운 hidden state의 후보($\tilde{h}_t$) 생성에 사용됨



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

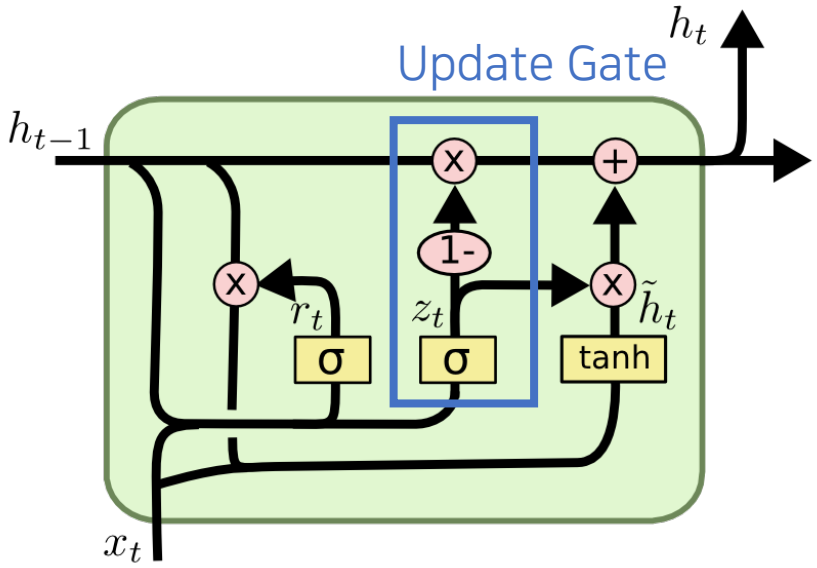
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Update Gate

- 새로운 Hidden state를 얼마나 이전 상태(h_{t-1})와 새로운 후보($\tilde{h}_t$) 상태 사이에서 선택할지를 결정
- LSTM의 forget gate와 input gate의 역할을 동시에 한다고 보면 됨
- 이전 단계의 정보를 얼마나 잊어버리고 새롭게 받아들일지를 동적으로 결정



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

과제

- 이번 주차는 과제 없습니다 ^___^