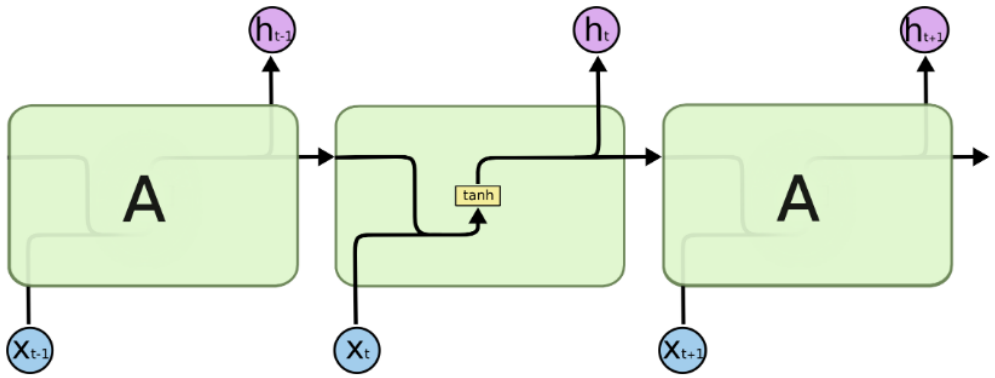
M5. RNN-advanced

LSTM (Long-Short Term Memory) - Gradient vanishing/ exploding 문제 해결을 위한 노력

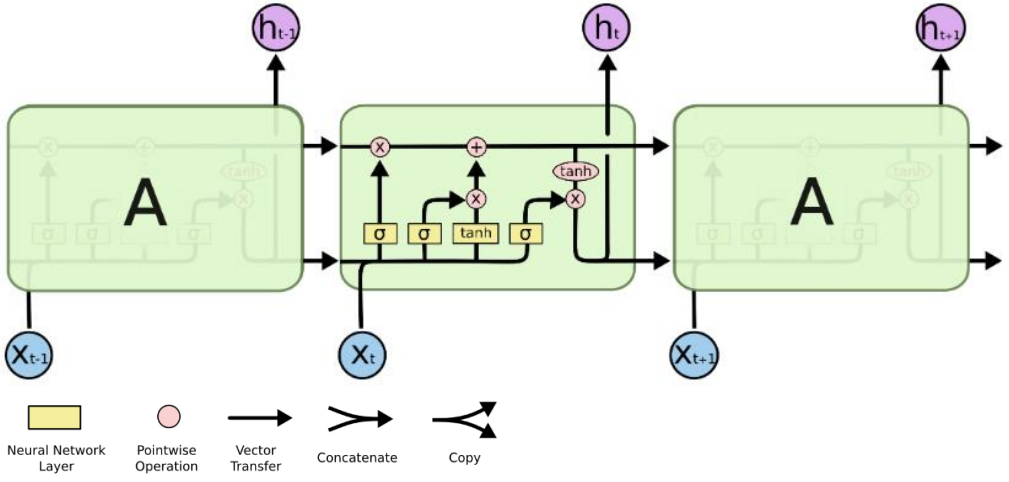
- 먼 과거의 정보도 잊어버리지 않도록 수식 변형
- **3개의 gate**를 이용해 현 단계의 인풋과 이전 메모리의 비윤을 조정한다

RNN unit



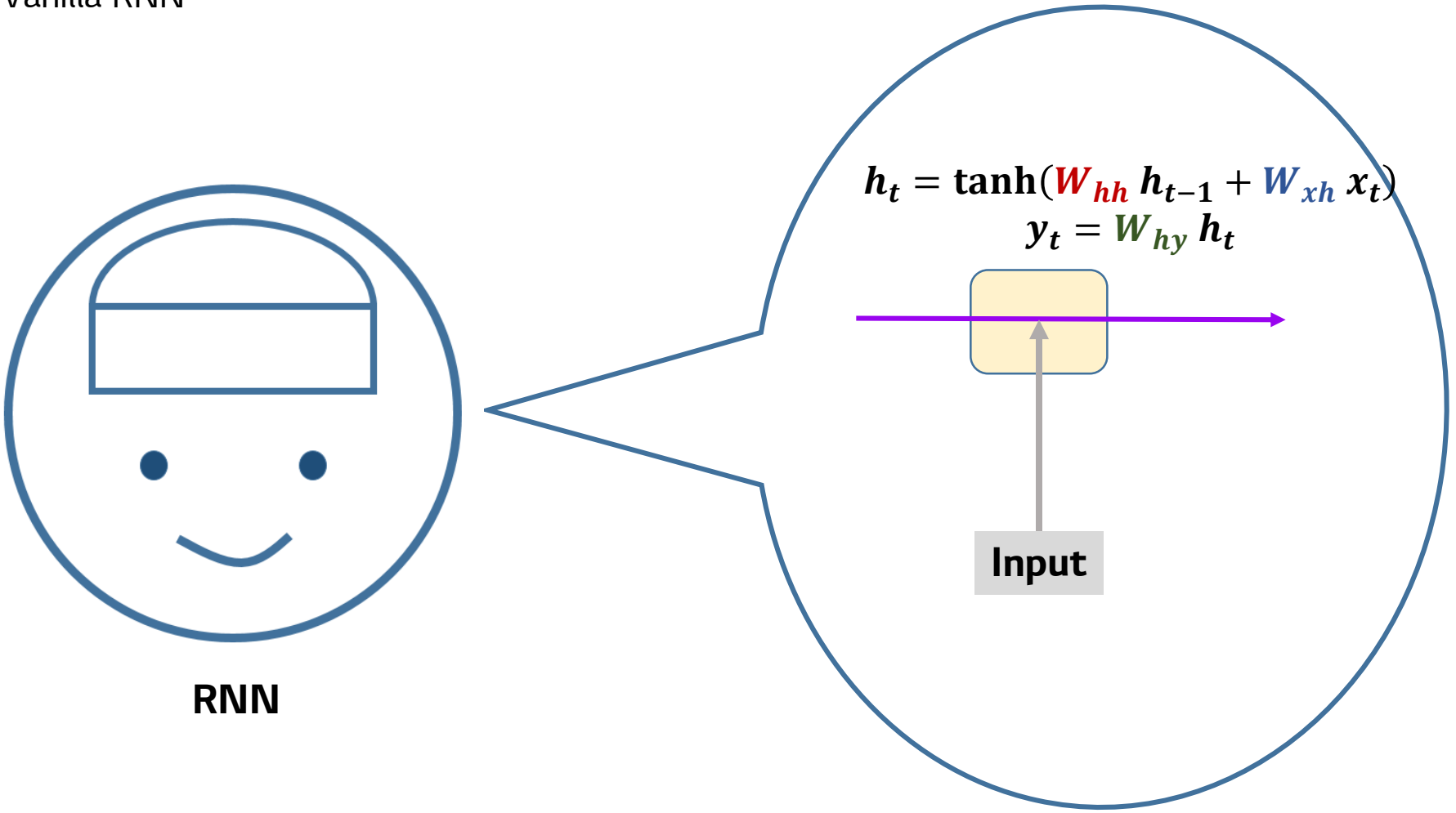
- RNN Cell은 이전 단계의 hidden과 이번 단계의 인풋을 결합해 다음 유닛으로 넘기는 역할만 수행

LSTM unit

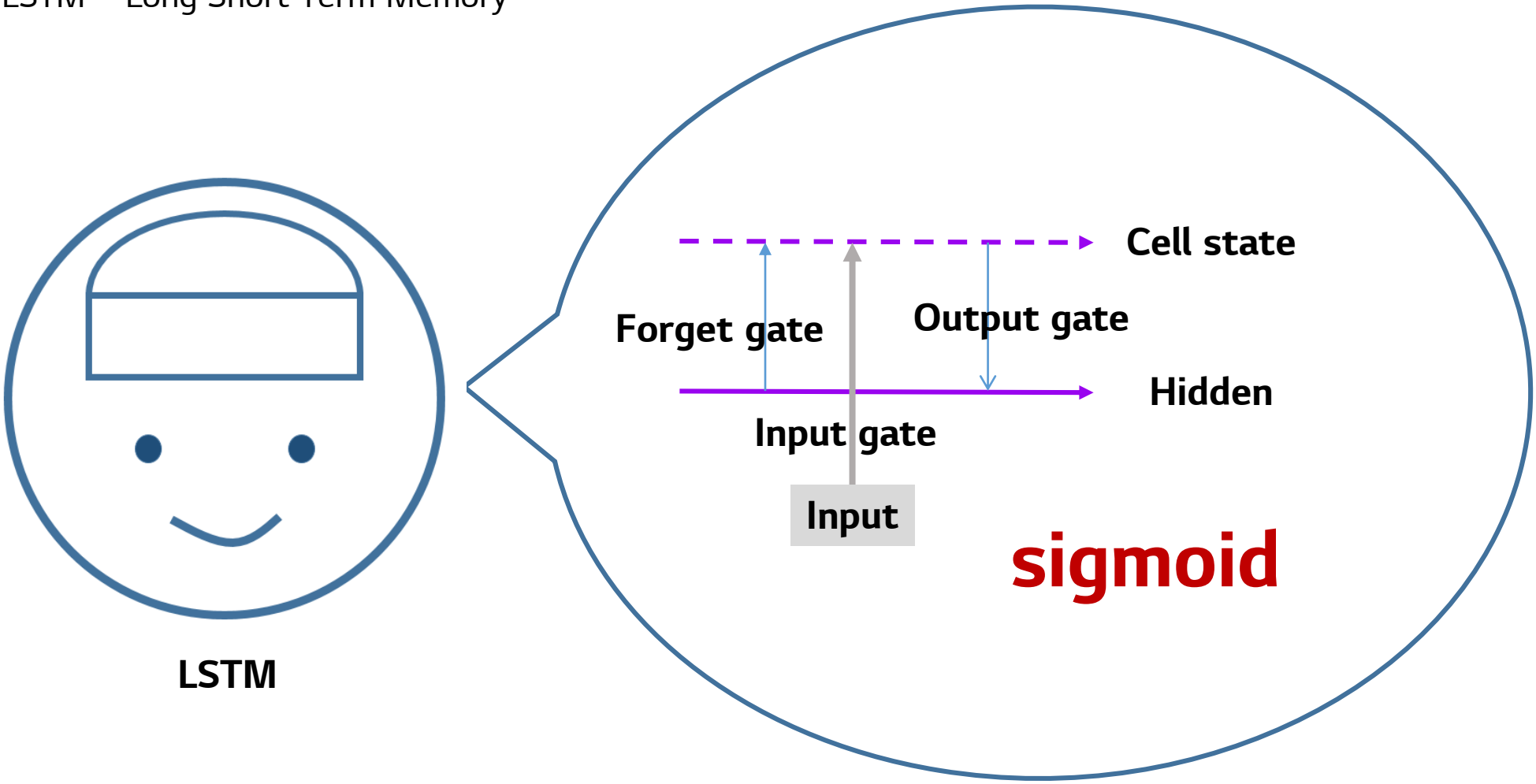


- LSTM에서는 Cell state와 Hidden state라는 두 개의 정보 흐름을 가지고 가며 인풋에 따라 정보를 취사선댁/ 결합하여 다음 유닛으로 넘기는 역할 수행

Vanilla RNN



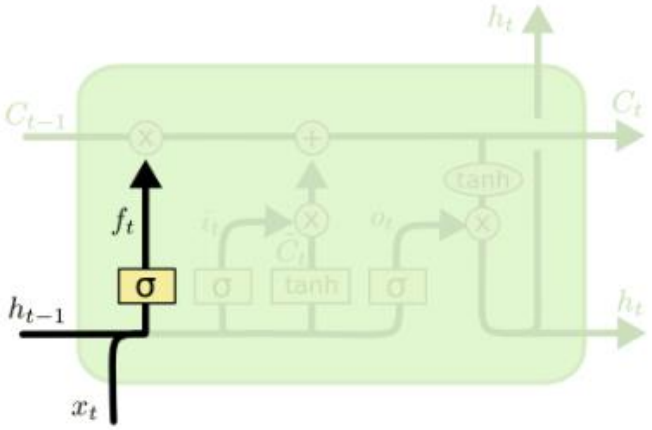
LSTM – Long Short Term Memory



LSTM (Long-Short Term Memory) – step by step

1) Forget Gate

- 이전 단계의 Cell state 중 어떤 정보를 버릴지 결정



이전 스텝의 히든 & 현재 인풋 정보 고려

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

0~1사이의 값으로 변환

Example

- cell state에서 '성별'에 대한 정보를 유지해 적절한 대명사를 사용할 수 있도록 하고 있었음.

C_{t-1}

'성별'에 대한 정보가 저장되어 있는 부분

- 주제가 바뀌어 더 이상 성별 정보를 유지할 필요가 없어지면 forget gate에서 해당 정보는 잊도록 만든다.

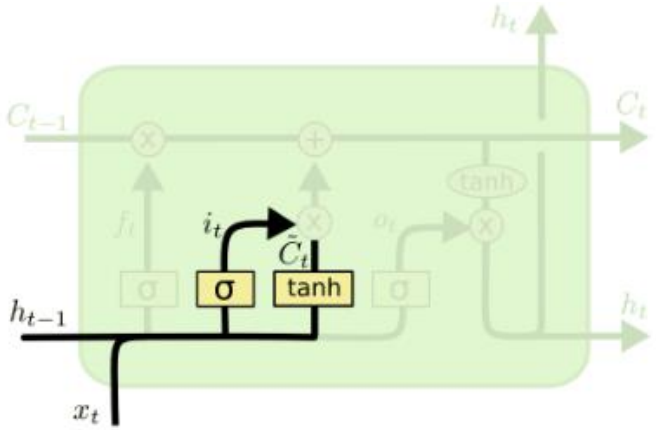
f_t	0.9	0.8	0.9	0.1	0.0	0.1	0.7	0.9	0.9

FLUSH!

LSTM (Long-Short Term Memory) – step by step

2) Input Gate

- 이번 단계의 인풋과 이전 단계의 히든을 결합해 정보를 가공하는 단계



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

$$\widetilde{C_t} = tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

Example

- 이전 단계의 히든과 이번 단계의 인풋을 결합해 임시 cell state $\widetilde{C_t}$를 만든다.

$\widetilde{C_t}$

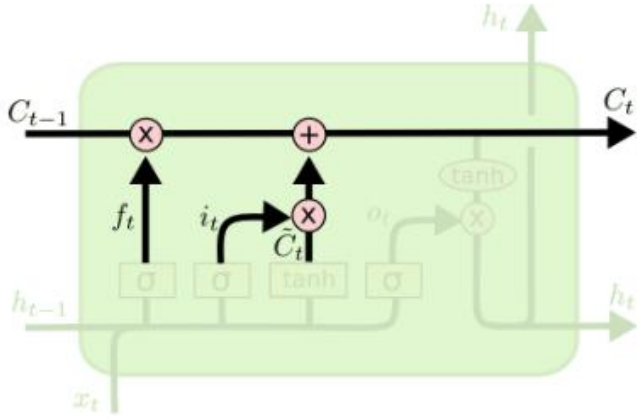
- 이전 단계의 히든과 이번 단계의 인풋 정보를 이용해 어떤 정보가 중요한지 조절하는 gate를 만든다.

i_t	0.1	0.9	0.5	0.7	0.8	0.0	0.1	0.4	0.9

LSTM (Long-Short Term Memory) – step by step

3) Cell state 만들기

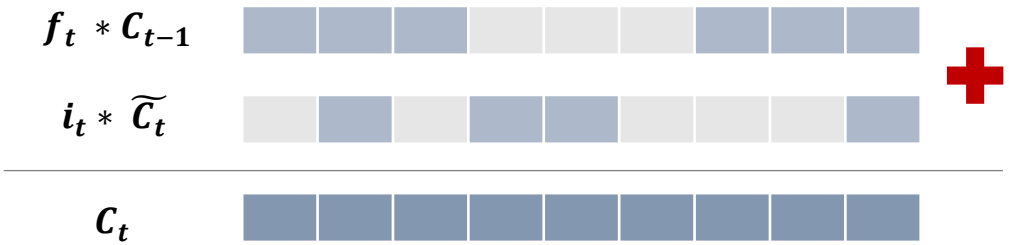
- 이번 단계의 cell state를 만드는 단계



$$C_t = f_t * C_{t-1} + i_t * \widetilde{C_t}$$

Example

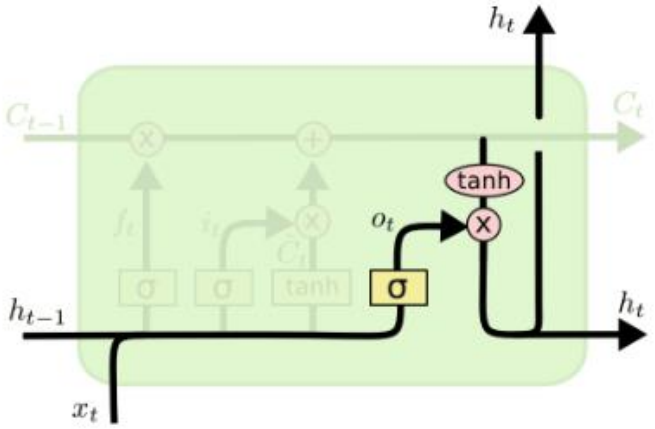
- Gate에 의해 조절된 이전 단계의 cell과 이번 단계의 cell을 결합해 현재 단계의 cell state 생성



LSTM (Long-Short Term Memory) – step by step

4) Output Gate

- 업데이트된 cell state를 output gate에 기반해 output으로 내보내는 단계



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

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

Example

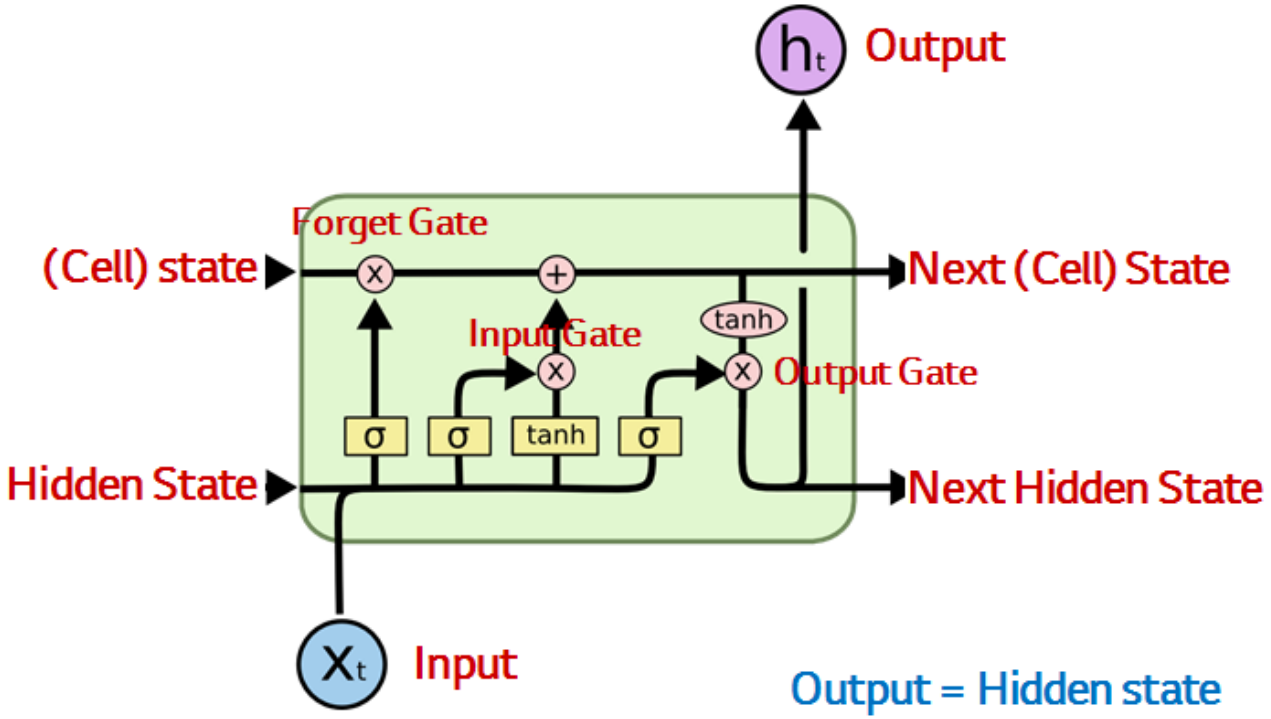
- Cell state 중 중요도에 따라 hidden state (= output)도 내보냄.

C_t

o_t

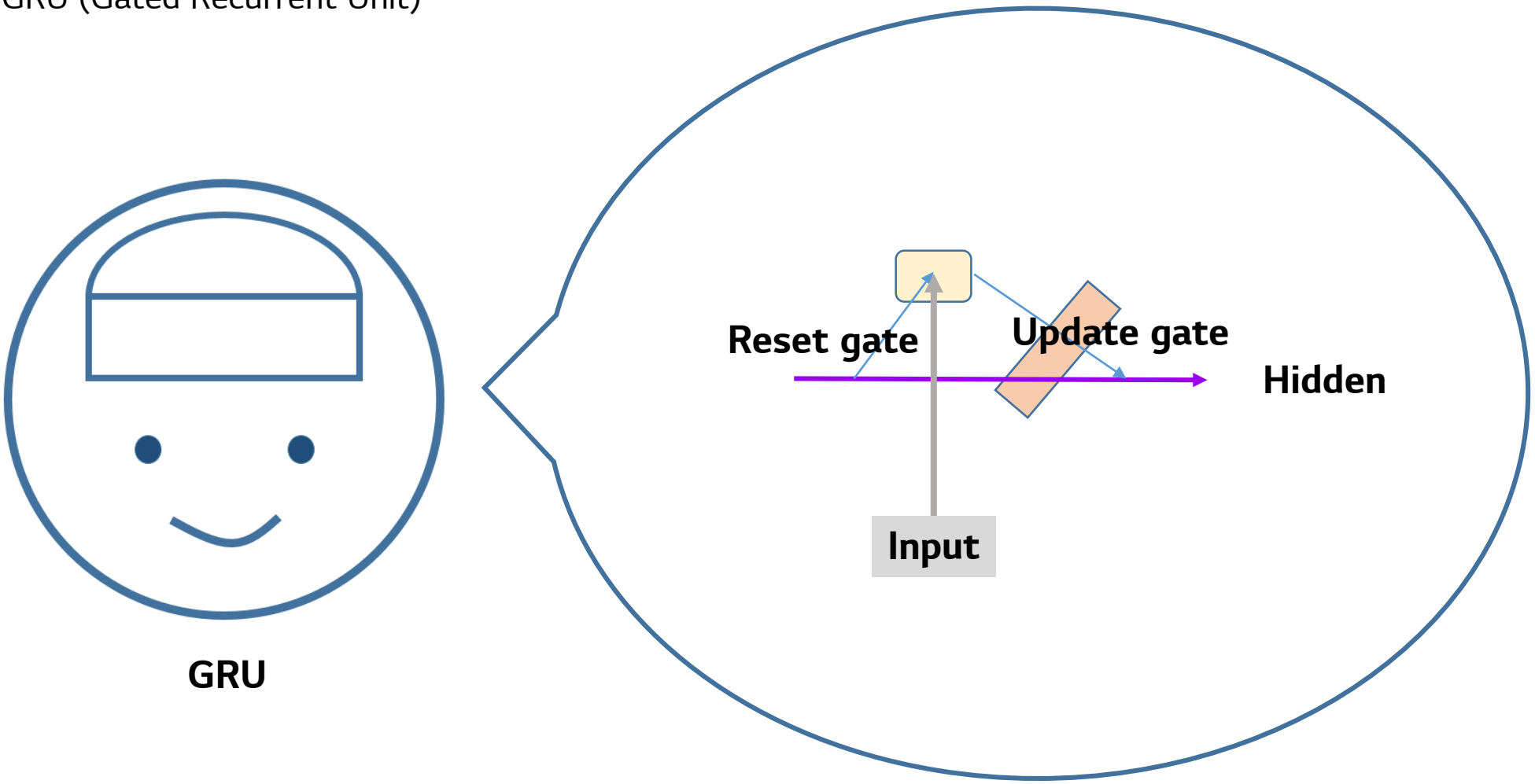
0.9	0.1	0.2	0.3	0.8	0.9	0.2	0.3	0.7

LSTM summary



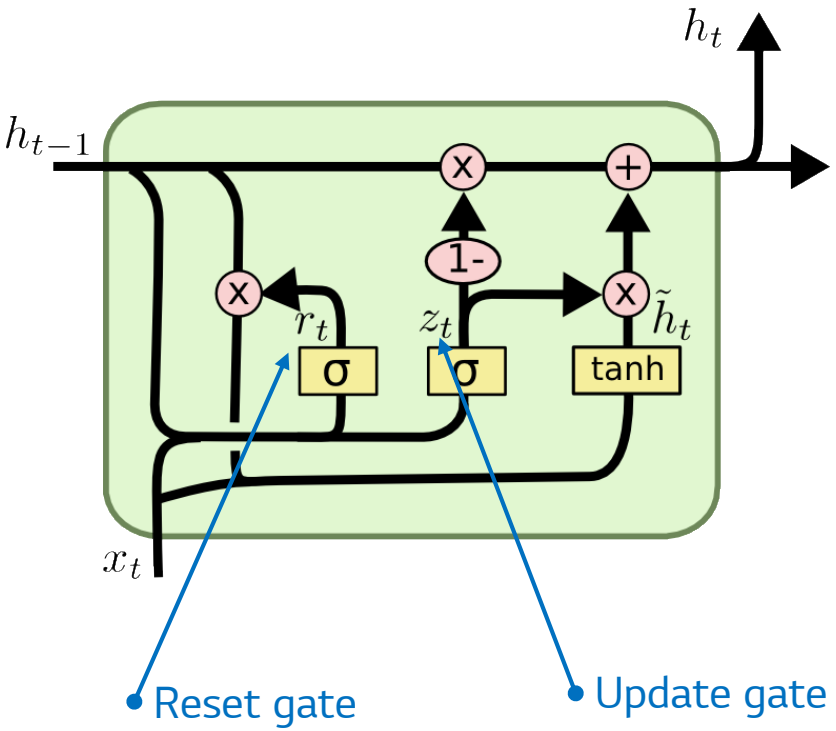
- **cell state**와 **hidden state**를 두어 이전 time-step까지 쌓인 정보와 현재 인풋에서 정보를 결합
- gradient vanishing (forgetting problem)이 완화되는 것으로 알려져 있음.
- hidden state를 RNN에서 output state와 같이 prediction에 활용함.

GRU (Gated Recurrent Unit)



GRU (Gated Recurrent Unit)

- **2개의 gate**를 이용해 현 단계의 인풋과 이전 히든의 정보 비율을 조절함
  - Reset gate - 새로운 hidden을 만들 때, 현재 입력과 관련 없는 과거의 정보를 drop
  - Update gate - 과거의 정보를 얼마나 기억할지 결정
  - 만약 z가 0에 가까운 경우, 과거의 정보를 그대로 복사 -> less vanishing gradient



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

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

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

Reset gate

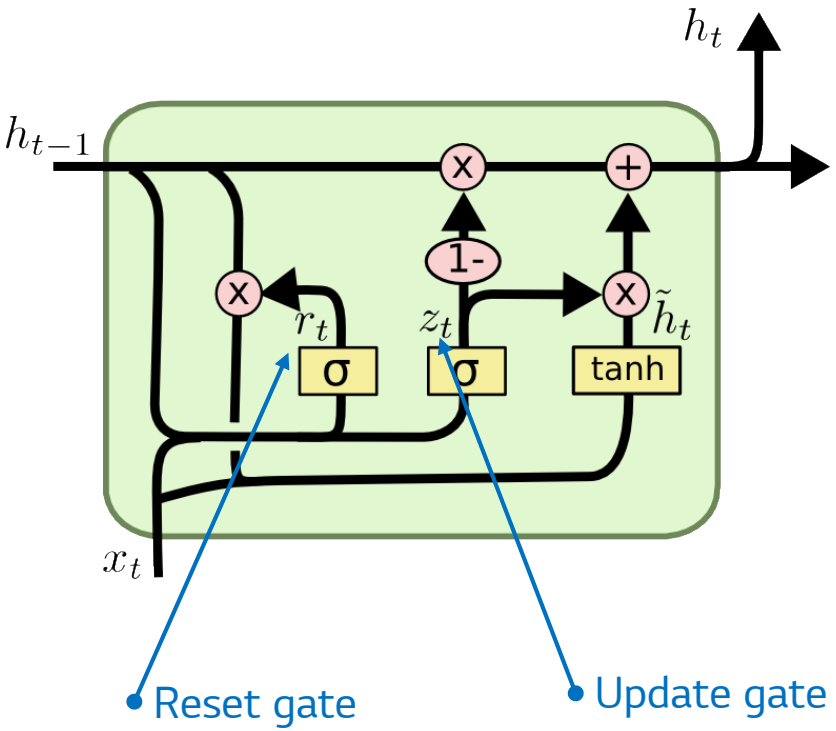
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Update gate

참조) http://colah.github.io/posts/2015-08-Understanding-LSTMs/

GRU (Gated Recurrent Unit)

- **2개의 gate**를 이용해 현 단계의 인풋과 이전 히든의 정보 비율을 조절함
  - Reset gate - 새로운 hidden을 만들 때, 현재 입력과 관련 없는 과거의 정보를 drop
  - Update gate - 과거의 정보를 얼마나 기억할지 결정
  - 만약 z가 0에 가까운 경우, 과거의 정보를 그대로 복사 -> less vanishing gradient



$$z_t = \sigma$$

$$r_t = \sigma$$

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

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

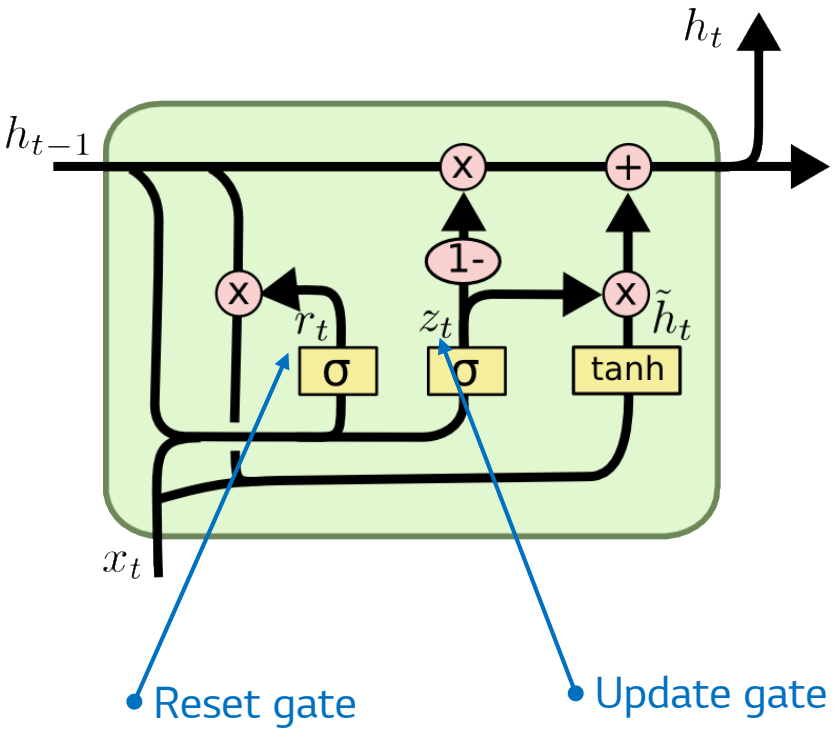
Reset gate의 값이 0에 가까우면?

-> 과거 히든의 정보는 이번 스텝의 feature을 만드는 데에 기여도가 낮음

참조) http://colah.github.io/posts/2015-08-Understanding-LSTMs/

GRU (Gated Recurrent Unit)

- **2개의 gate**를 이용해 현 단계의 인풋과 이전 히든의 정보 비율을 조절함
  - Reset gate - 새로운 hidden을 만들 때, 현재 입력과 관련 없는 과거의 정보를 drop
  - Update gate - 과거의 정보를 얼마나 기억할지 결정
  - 만약 z가 0에 가까운 경우, 과거의 정보를 그대로 복사 -> less vanishing gradient



Update gate의 값이 1에 가까우면

-> 과거 히든을 그대로 복사하는 부분은 줄어든다

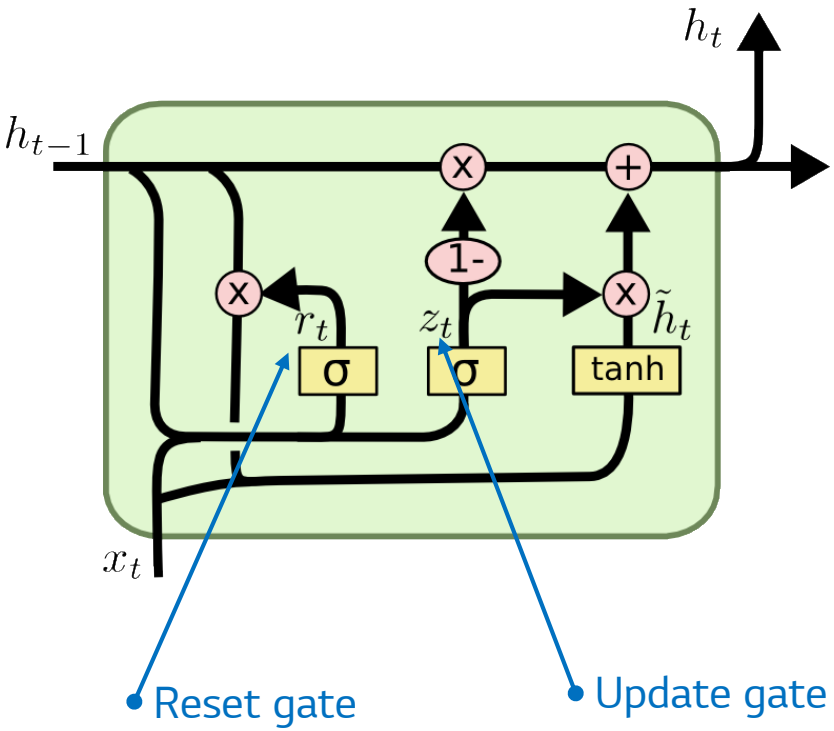
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

-> 이번 스텝에서 만들어진 벡터를 크게 반영하여 이번 스텝의 히든을 만든다.

참조) http://colah.github.io/posts/2015-08-Understanding-LSTMs/

GRU (Gated Recurrent Unit)

- **2개의 gate**를 이용해 현 단계의 인풋과 이전 히든의 정보 비율을 조절함
  - Reset gate - 새로운 hidden을 만들 때, 현재 입력과 관련 없는 과거의 정보를 drop
  - Update gate - 과거의 정보를 얼마나 기억할지 결정
  - 만약 z가 0에 가까운 경우, 과거의 정보를 그대로 복사 -> less vanishing gradient



Update gate의 값이 0에 가까우면

-> 과거 히든을 거의 그대로 복사
-> gradient가 유지됨

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

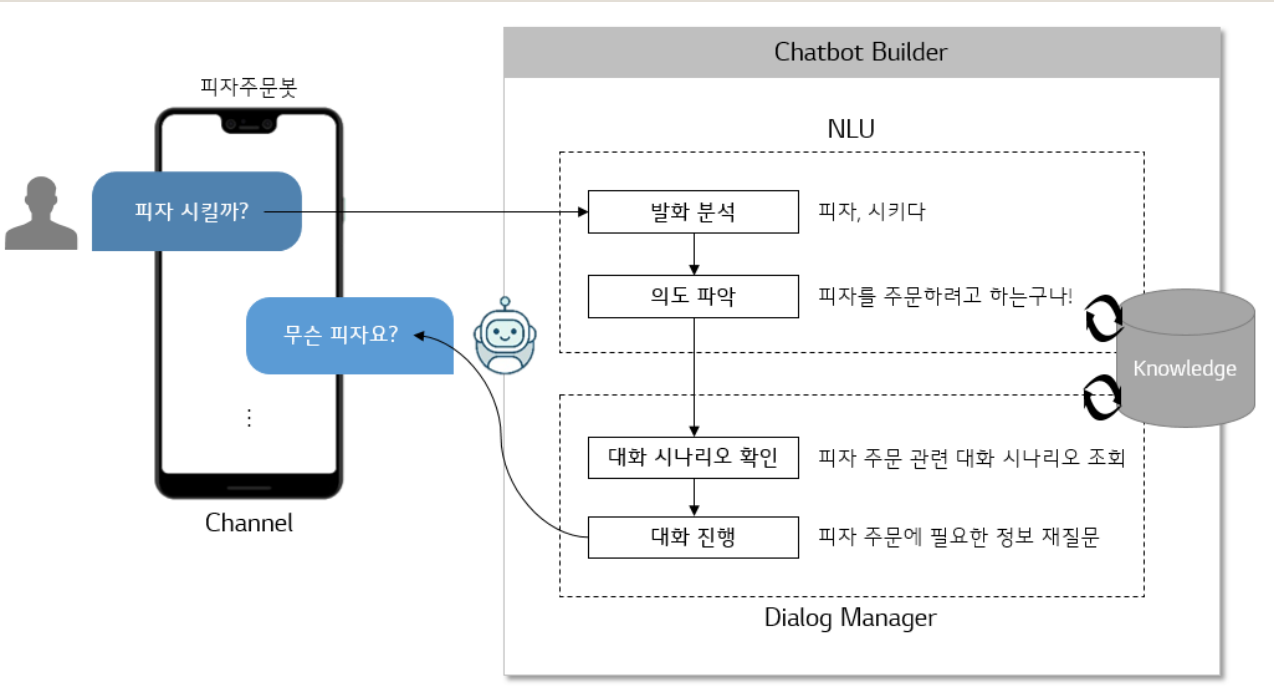
-> 이번 스텝의 정보는 히든을 만드는 데에 크게 중요하지 않음

참조) http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Today's Mission 실습_3_RNN_Advance.ipynb

RNN을 이용한 의도 분류 실습

챗봇의 첫 번째 단계 – 사용자의 발화 의도 파악하기!



RNN을 이용한 의도 분류 실습

- **데이터** : 한국어 화행분석 데이터셋

https://github.com/sgnlplabeling/nlp_labeling

	label_new	sentence	speaker
0	opening	아름 아 잘 잤 니 ?	user
1	opening	네 , 잘 잤 습니다 .	system
2	request	아름 아 일정 확인 좀 해줘 .	user
3	wh-question	언제 일정 을 확인할 까 요 ?	system
4	inform	다음달 일정 좀 알려줘 .	user
5	inform	네 , 오전 과 오후 약속 이 있 어요 .	system

- **학습 목표** :
  - NLU의 전체 FLOW (토크나이징 -> 인코딩 -> 임베딩 -> 모델링)을 이해하고 구현한다.
  - TensorFlow에서 RNN layer을 사용해 의도분류 모델을 구축할 수 있다.
  - 모델을 컴파일하고 분석, 평가하는 딥러닝 전체적인 플로우를 실행할 수 있다.
  - 학습된 모델을 이용해 사용자 발화를 인풋으로 받아 의도를 리턴하는 추론 함수를 만들 수 있다.