Learning Phrase Representation using RNN Encoder-Decoder for Stistical Machine translation (2014)

FOM

Focus On Data Mining

INDEX

01 Introduction

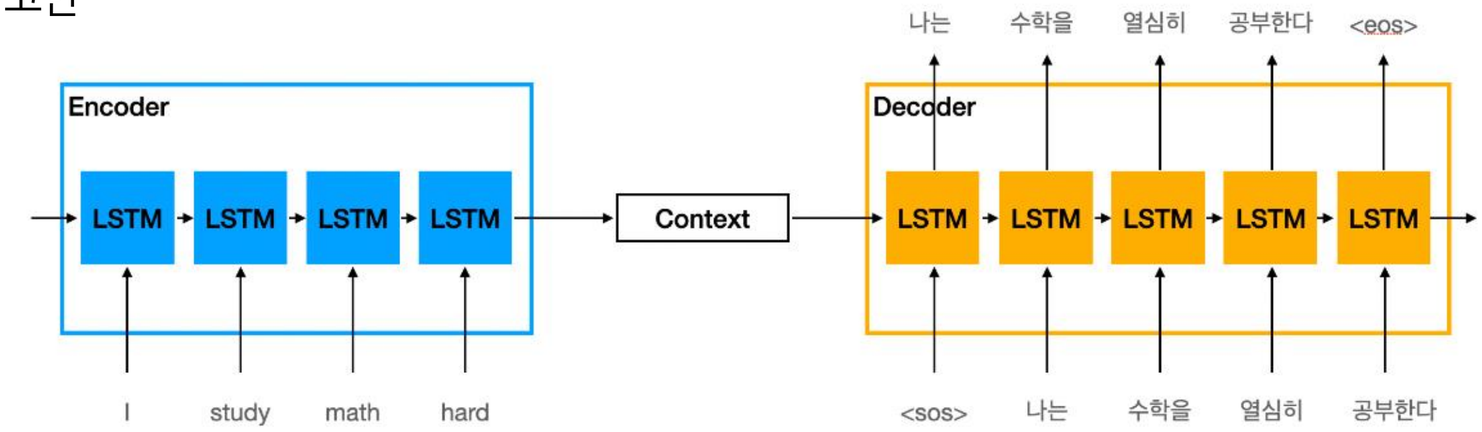
Neural networks in SMT

- SMT (statistical machine translation) 에서 neural networks의 가능성

▶ Deep neural networks가 objection recognition, speech recognition, NLP, paraphrase detection, language modeling, and word embedding extraction 등 에서 성공적인 결과 및 과제를 수행

▶ 기존 Encoder-Decoder은 **long-length, variable-length**을 잘 다루지 못한다는 한계점

▶ Neural network 중 **recurrent neural network(RNN)을 사용하는 encoder-decoder**를 고안



01 Introduction

Key points of paper and Translation

- Translation의 특징 및 유의점

  ① encoder-decoder 의 **input and output sequence lengths가 가변적**
  ② 언어적,문법적 차이로 인해 Source,Target sequence의 **단어 등장 순서가 다름**
  ③ 같은 의미의 Source sequence라도 **2개 이상의 다른 Target sequence**로 나올 수 있음

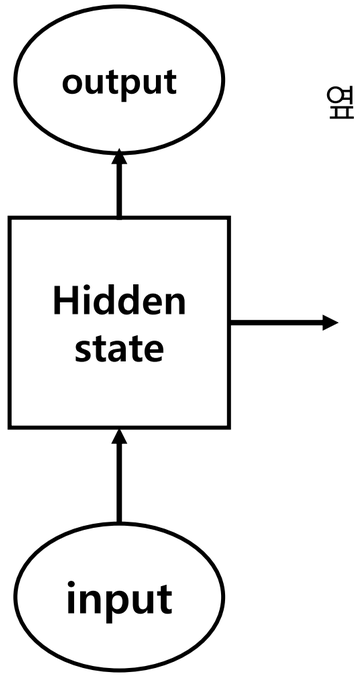
- Key points of paper

  ① **RNN Encoder-Decoder** 제시
  - Fully connected network를 사용해서 **정확도 및 가변적인 문장을 다룸**

  ② **Gated Recurrent Unit (GRU)** 제시
  - RNN Encoder-Decoder의 내부 Unit
  - **LSTM와 유사**하지만 더 간단한 구조로 **계산량을 줄임**

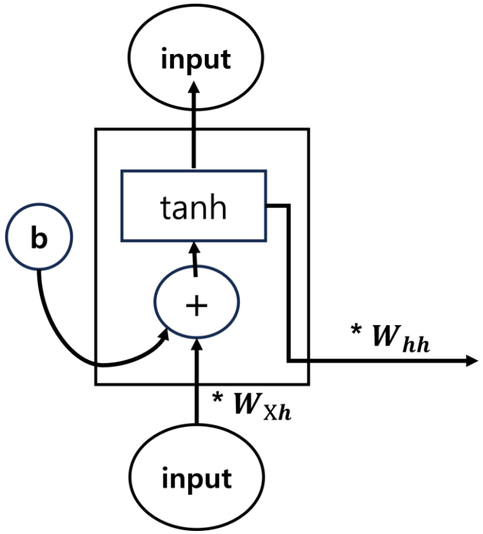
02 Related Works

RNN



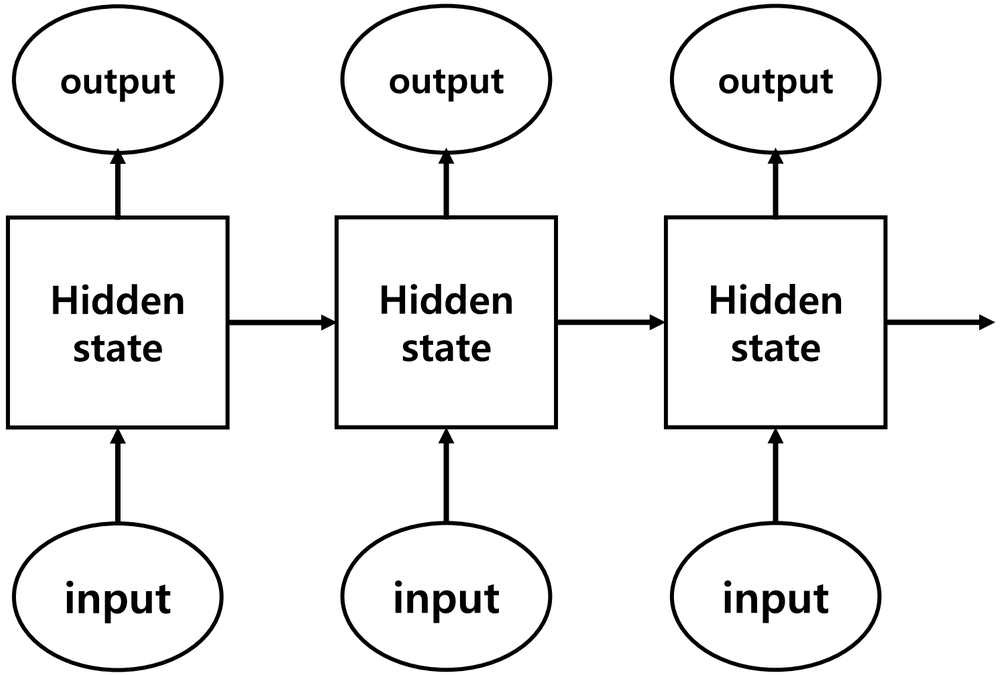
추상적인 RNN cell

옆의 neural network을 자세하게 표현



$$h_t = \tanh(W_{xh} * x_t + W_{hh} * h_{t-1} + b)$$

RNN에서 input data는 embedding 또는 numeric data 입력
입력된 input data는 weight와 bias들과 더해지고 activation function을 통해 hidden state 출력

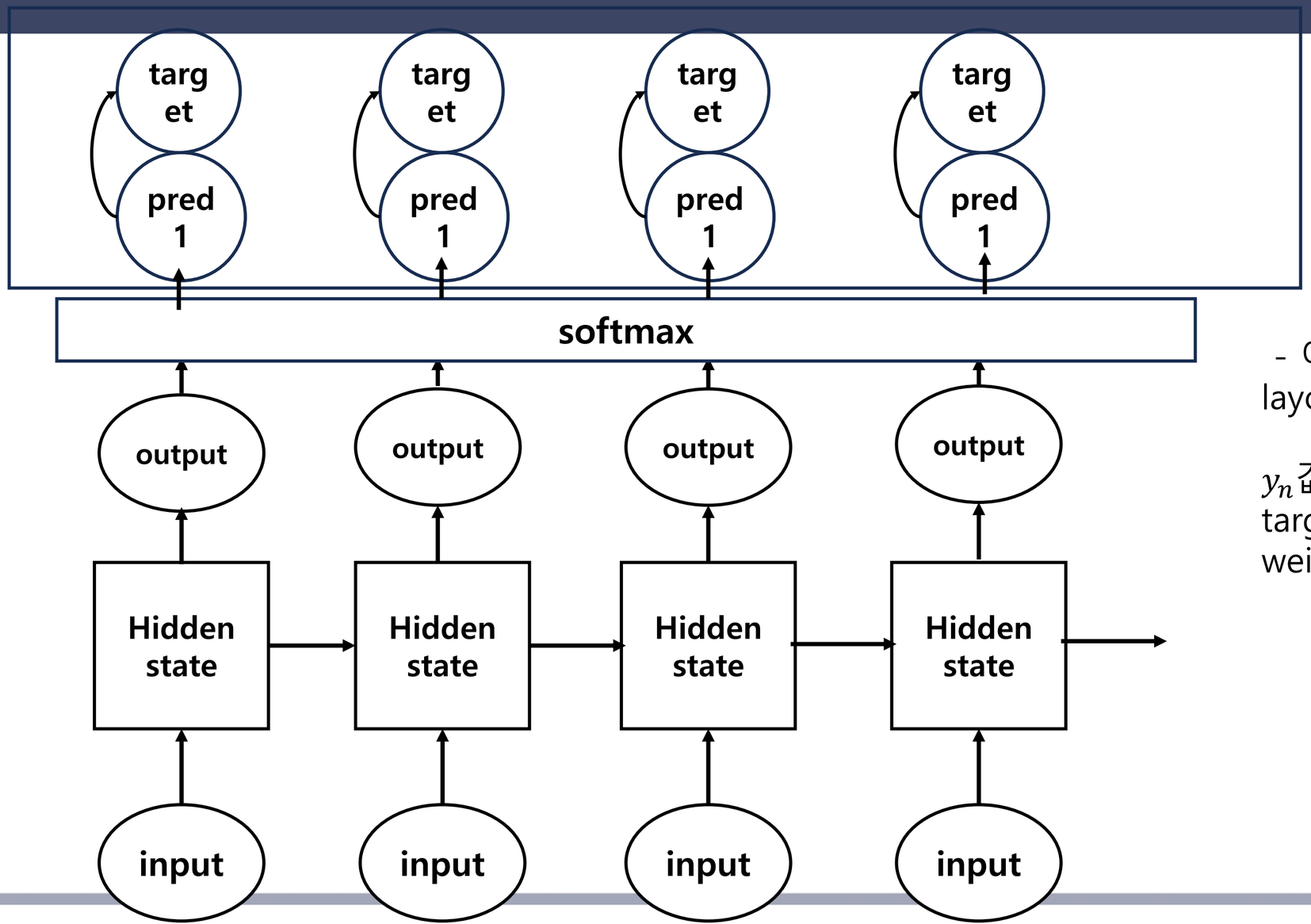


RNN cell들이 여러 개 붙여 이어져 있음

Hidden state는 현재 input,과거 hidden state의 연산을 통해 현재 hidden state 를 출력

02 | Related Works

RNN

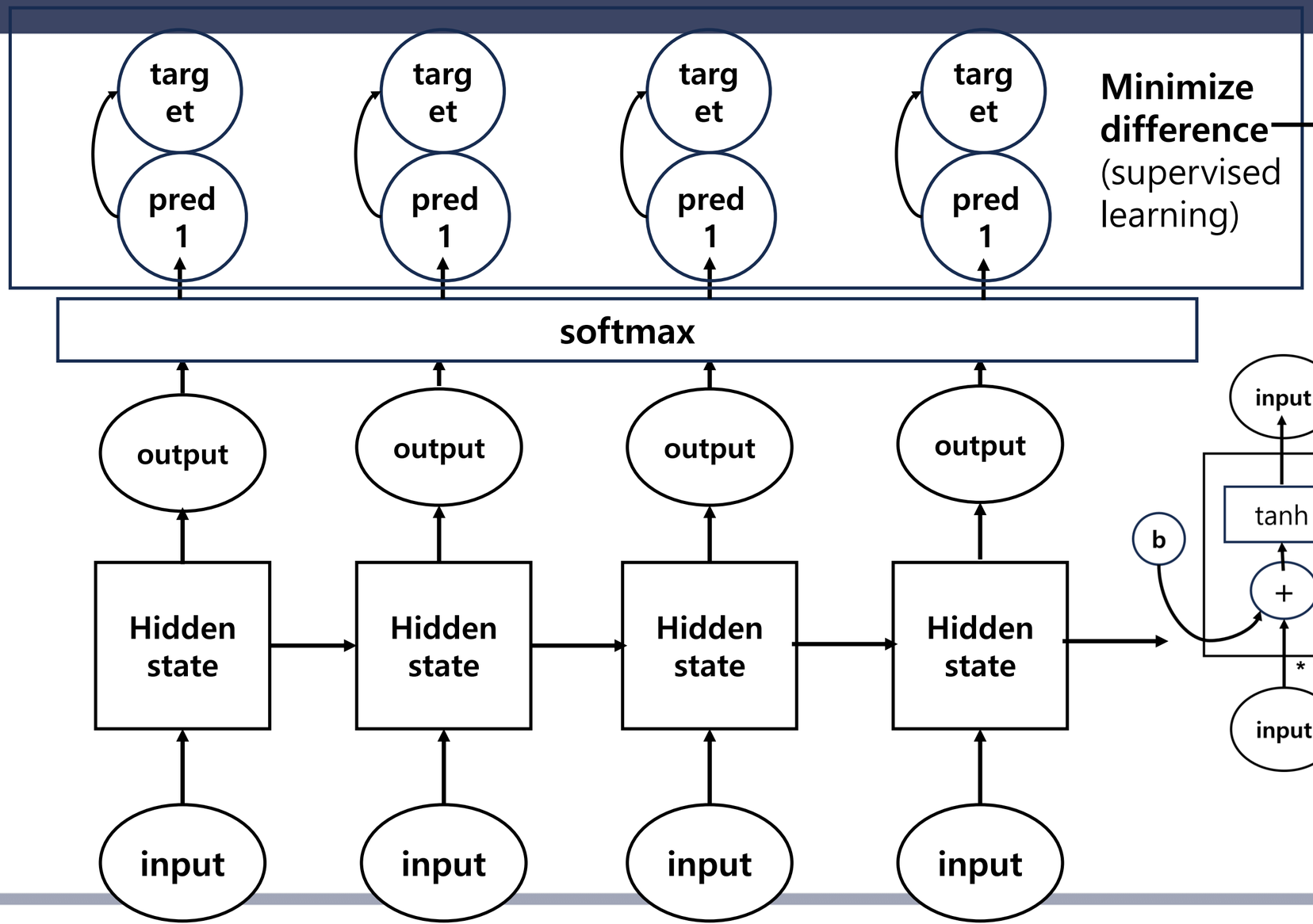


- 여러 개의 hidden state cell을 합친 RNN layout

y_n값을 **softmax**을 통해 prediction 출력 후 target 값과 비교하면서 hidden state의 weight,bias를 수정

02 | Related Works

RNN



Minimize difference (supervised learning)

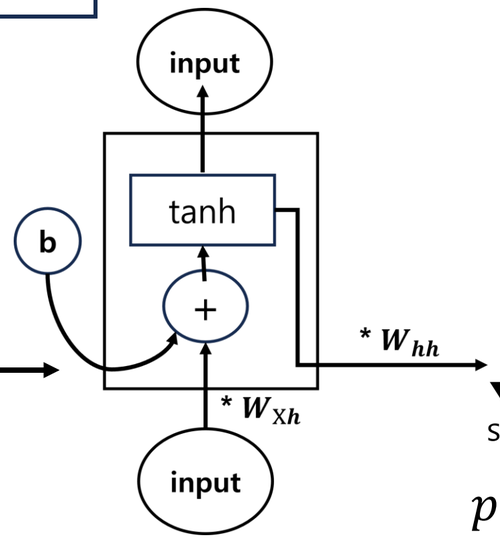
→ **Optimizing** W_{Xh}, W_{hh}, bias By **gradient decent**

▼ At each time step, the hidden state $h_{<t>}$ of RNN

$$h_{<t>} = f(h_{<t-1>}, x_t, b)$$

▼ Output (by **softmax**)

$$p(x_{t,j} = 1 | x_{t-1}, \dots, x_1) = \frac{\exp(\boldsymbol{w}_j \boldsymbol{h}_{<t>})}{\sum_{j'=1}^{K} \exp(\boldsymbol{w}_{j'} \boldsymbol{h}_{<t>})}$$



▼ compute the probability of sequence $\boldsymbol{x}$ T

$$p(x) = \prod_{t=1} p(x_t | x_{t-1}, \dots, x_1)$$

02 Related Works

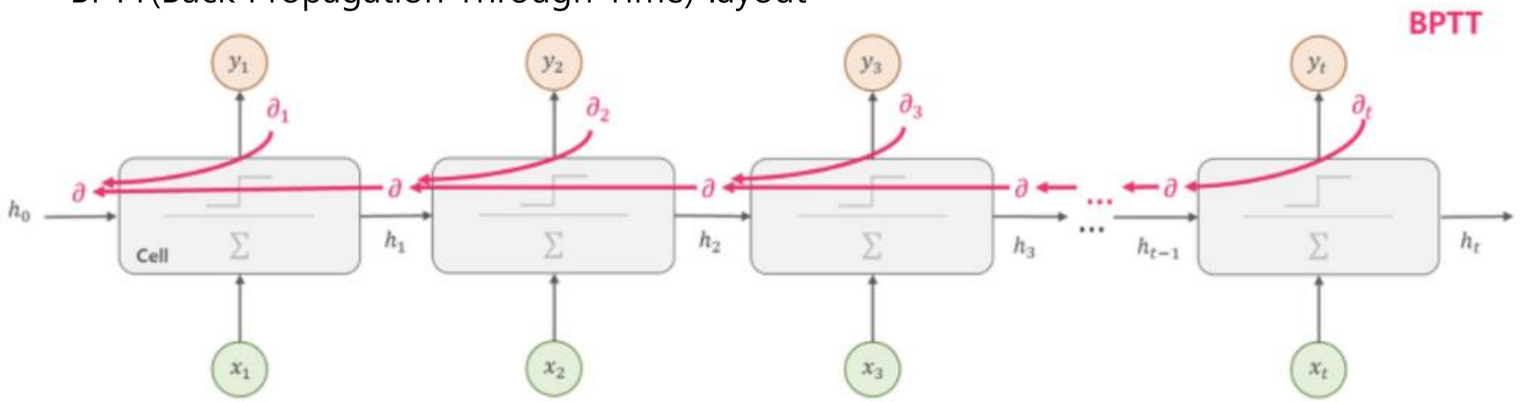
RNN

- RNN의 한계점

  - RNN은 text length가 길어질수록 처음에 나온 단어의 영향력이 약해지는 문제점이 있음
  즉, RNN은 **long-term dependency**를 다룰 수 없음

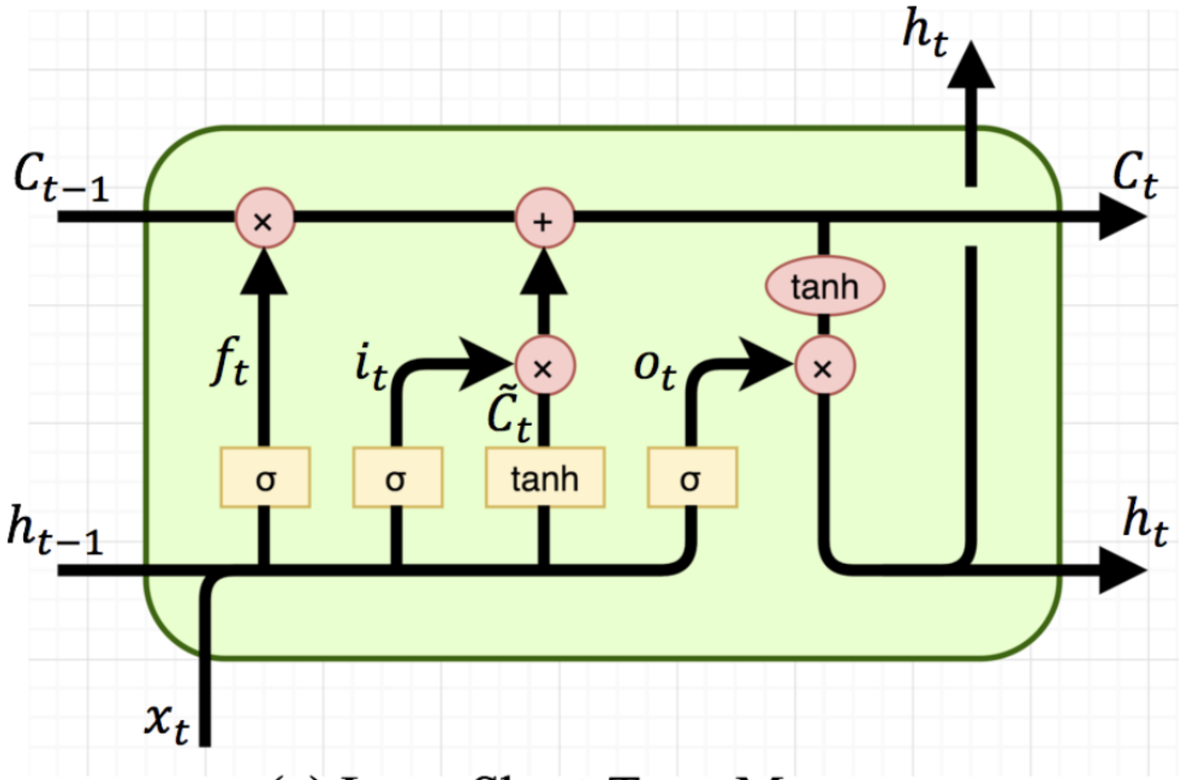
  - RNN은 시간에 따라 해석하는 **BPTT(Back-Propagation Through Time)**방식을 취하지만 신경망의 망이 깊어질수록 **Gradient Vanishing**이 발생

BPTT(Back-Propagation Through Time) layout



02 | Related Works

LSTM



(a) Long Short-Term Memory

- RNN의 문제점(Gradient vanishing problem) 해결 및 long-term dependencies를 다루기 위한 해결책

- W_{Xh}, W_{hh}, bias 등 가중치를 gradient decent optimizing을 거침

- C_t : **memory cell** (RNN과의 차이점)
- h_t : hidden state

- 이름 그대로 Long Short term memory를 가지면서 진행되는 모델

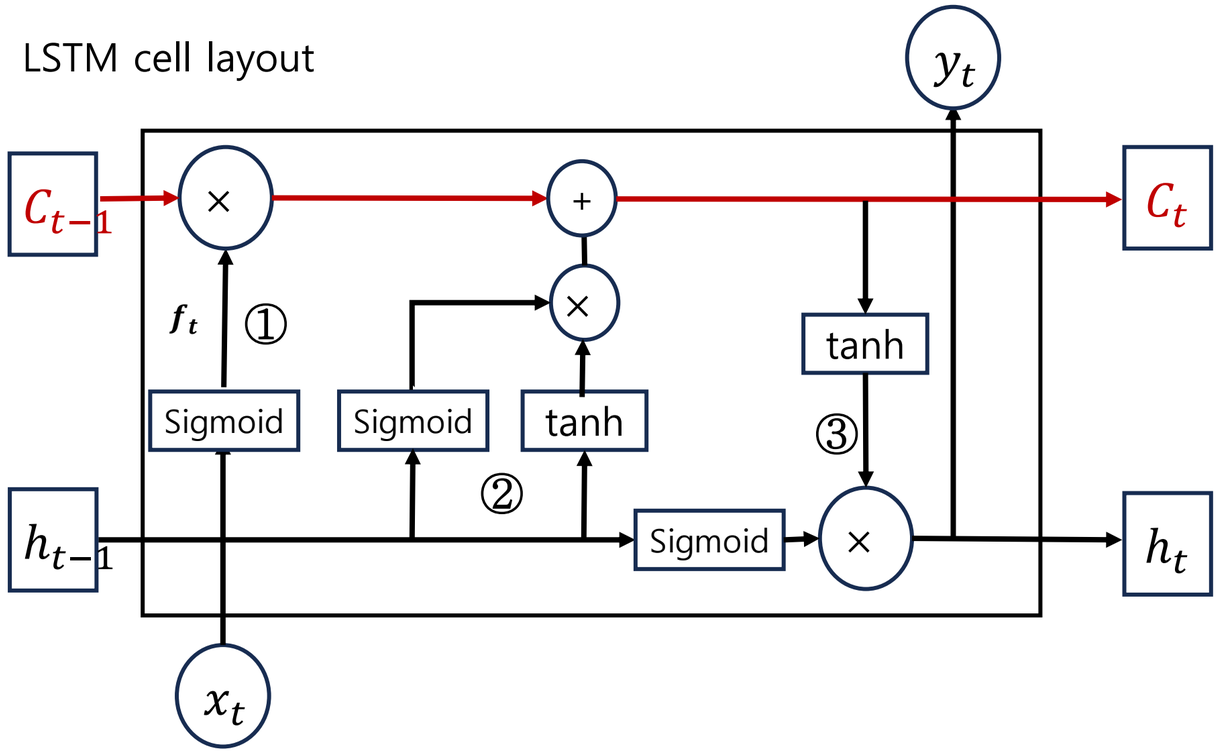
- RNN이 내부에 다수 존재하는 느낌

- $f_t, i_t, \tilde{C}_t, o_t$ 들은 LSTM속 다른 역할들을 수행하는 mechanism
  -> 한 줄 소개하면 이전 **정보의 수도꼭지 역할**
  (이전의 정보의 양을 조절해 주는 역할)

02 Related Works

LSTM

LSTM cell layout



C_t : memory cell

h_t : hidden state

① : forget mechanism (f_t)

- f_t의 **출력값은 0~1**로 x_t의 정보를 얼마나 사용할지 **확률값으로 출력**

- f_t는 과거의 정보를 잊기 위한 gate -> **sigmoid function**으로 0~1 값이 출력

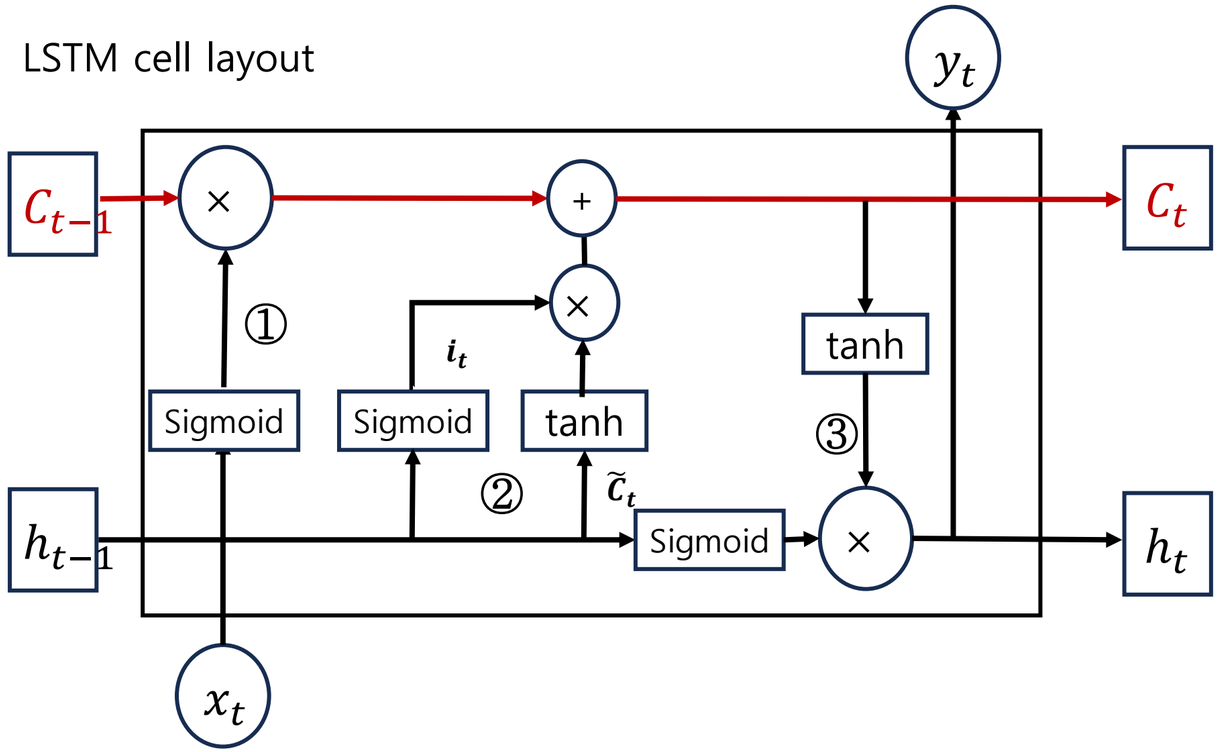
- **출력값이 0이라면** 이전 **상태의 정보는 잊고**, **1이라면** 이전 상태의 **정보를 온전히 기억**함

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_t)$$

02 Related Works

LSTM

LSTM cell layout



C_t : memory cell

h_t : hidden state

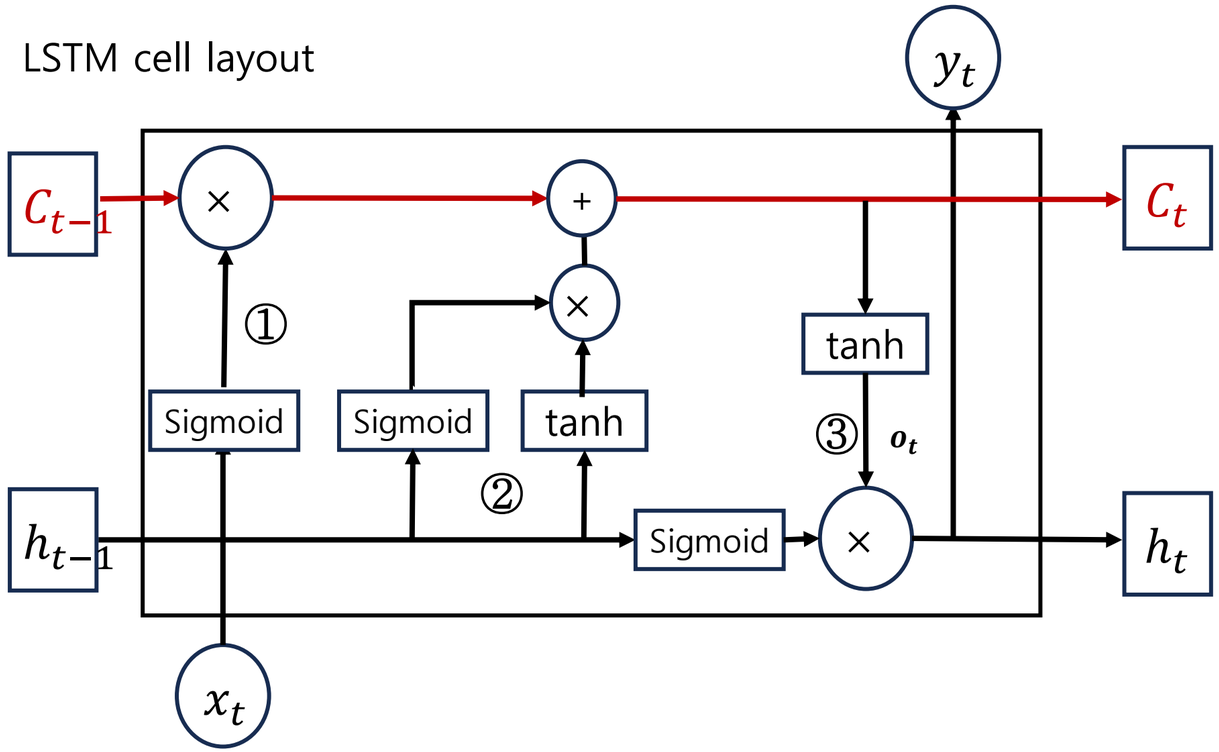
② : input mechanism $(i_t, \tilde{C}_t)$

- **tanh,sigmoid function**을 지나면서 새로운 input 값으로 바뀜. Forget mechanism을 지난 memory cell과 더함
- $\boldsymbol{i_t}$는 현재 정보를 기억하기 위한 gate -> sigmoid function으로 0~1값으로 출력
- $\boldsymbol{\tilde{C}_t}$는 tanh의 결과값으로 -1~1이 되므로 **input값이 음수**가 될 수도 있음
- $\boldsymbol{i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i)}$

02 Related Works

LSTM

LSTM cell layout



C_t : memory cell

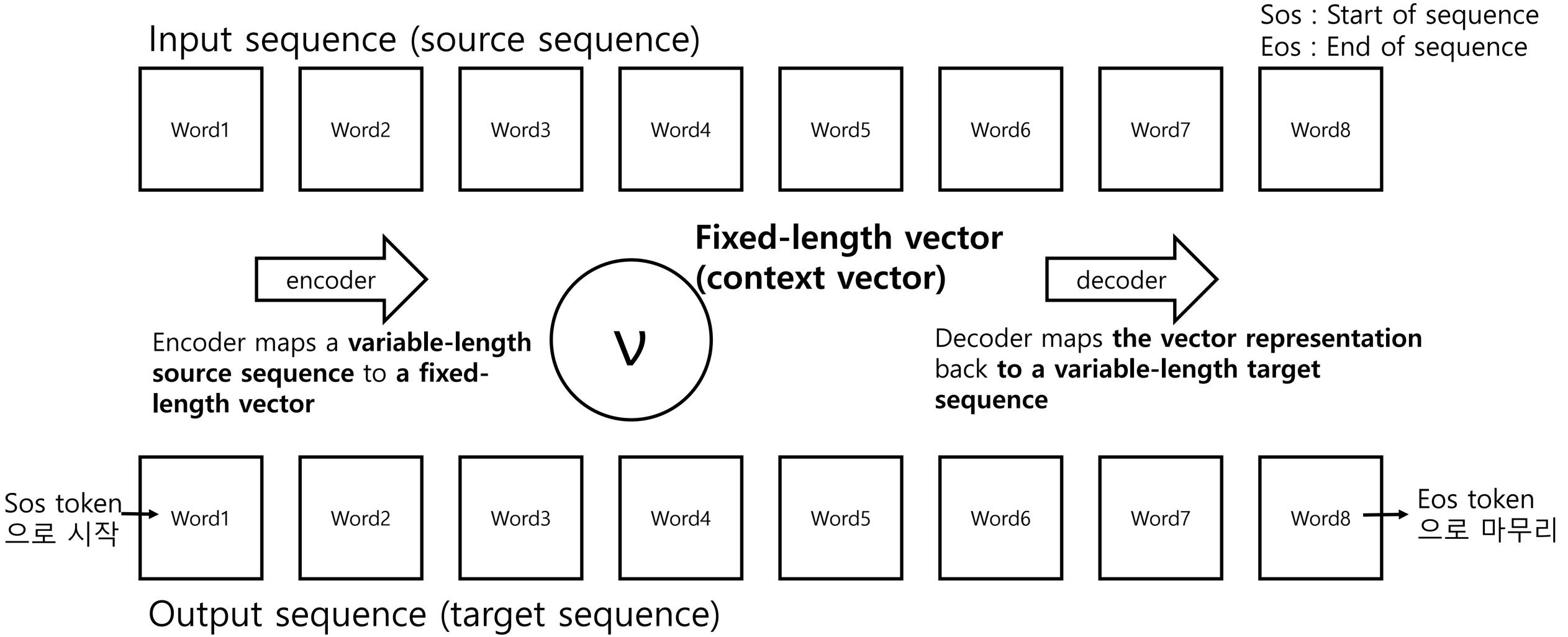
h_t : hidden state

③ : output mechanism (o_t)

- 기존의 hidden state의 출력값이 sigmoid를 지난 값과 input mechaism을 지나 새롭게 바뀐 입력값들이 곱해지면서 **새로운 hidden state 출력값**을 만듦
- $\boldsymbol{o_t}$의 출력값이 LSTM의 최종 결과
- $\boldsymbol{o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o)}$

02 Related Works

seq2seq



02 Related Works

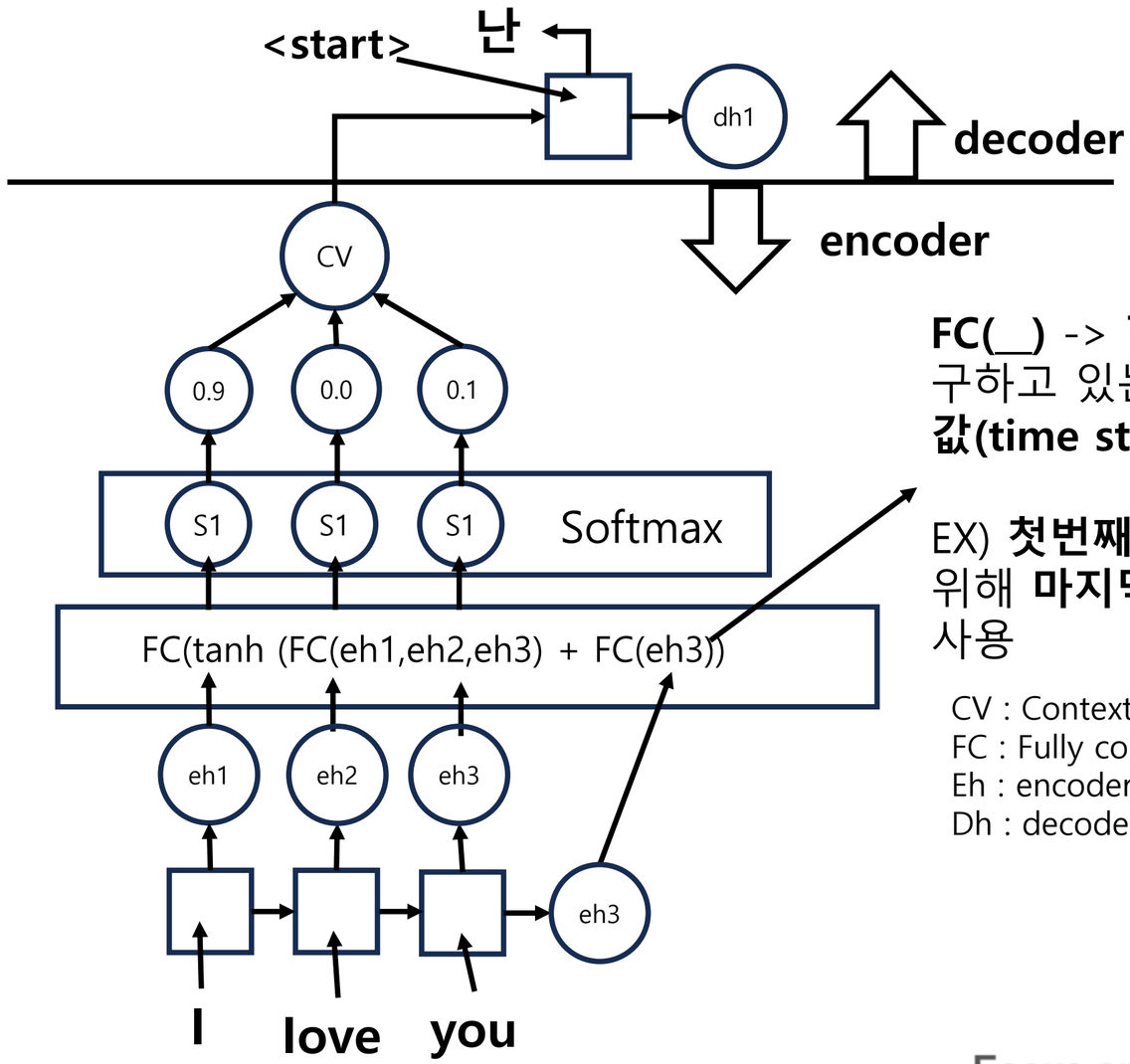
Encoder-Decoder

encoder

Encoder는 input sequence인 **비정형 데이터(word or sequence)** 를 **embedding**을 통해 컴퓨터가 읽을 수 있는 data로 변환 후 이를 통해 **context vector**로를 생성

decoder

Decoder는Encoder를 통해 생성된 **context vector**를 **embedding의 역과정**을 통해 문장 생성 즉, **output sequence**를 생성



FC(__) -> 현재(time step t) 구하고 있는 **cell 이전의 값(time step t-1)**을 사용함

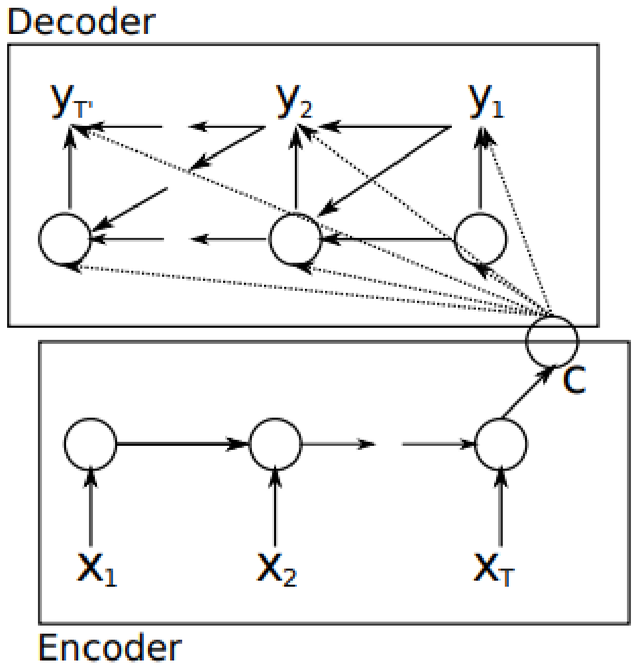
EX) **첫번째 decoder cell**을 위해 **마지막 encoder cell**을 사용

CV : Context vector
FC : Fully connected network
Eh : encoder hidden state
Dh : decoder hidden state

03 Proposed Method

RNN Encoder-Decoder

▼ An illustration of the proposed RNN Encoder-Decoder



RNN encoder는 sequence x를 순차적으로 읽고 RNN의 hidden state로써 **summary C를 출력**
이때 **C는 모든 input sequence의 summary**

RNN decoder는 output sequence를 generate
그와 동시에 **encoder의 hidden state를 통해 next symbol인 y_t를 예측**

하지만, **다른 RNN과 다르게 h_t와 y_t를 y_{t-1}과 C**를 가지고 예측
(*cf*, **기존의 RNN**은 $h_{<t>} = f(h_{<t-1>}, x_t)$ 로 **previous hidden state**와 **현재 입력값**을 사용함)

따라서, **hidden state of the decoder**는 아래와 같음
$h_{<t>} = f(h_{<t-1>}, y_{t-1}, C)$ **<- decoder hidden state**

위 두개의 RNN (RNN encoder-decoder)는 아래의 목적 함수를 **maximize**하기 위해
weight,bias들을 수정 -> **maximize the conditional log-likelihood**

$$\max_{\theta} \frac{1}{N} \sum_{n=1}^{N} log p_\theta(y_n|x_n)$$

θ : set of the model parameters
Each (x_n, y_n) : pair from the training set

03 | Proposed Method

RNN Encoder-Decoder

$$\max_{\theta} \frac{1}{N} \sum_{n=1}^{N} log p_\theta(y_n|x_n)$$

θ : set of the model parameters
Each (x_n, y_n) : pair from the training set

$p_\theta(y_n|x_n)$ 는 x_n일때, y_n인 확률 즉, **조건부 확률**(conditional probability)
즉, $p_\theta(y_n|x_n)$ 는 각 파라미터들에 대해서 log scale conditional probability
Log scale conditional likelihood의 sum의 average를 maximizing

이때 x_n, y_n 는 각각input sequence, output sequence 의 words

예로 들면, x_1 : I, x_2 : Love, x_3 : You -> y_1 : 난 , y_2 : 널 , y_3 : 사랑해 일 때,
$\boldsymbol{p_\theta}$(난|$\boldsymbol{I}$), $\boldsymbol{p_\theta}$(널|$\boldsymbol{You}$), $\boldsymbol{p_\theta}$(사랑해|$\boldsymbol{Love}$) 일 확률을 $\boldsymbol{\theta}$ 을 **tuning**하면서 **maximizing**

03 Proposed Method

Hidden Unit that Adaptively Remembers and Forgets (GRU)

- The reset gate (r_j)

$$r_j = \sigma([W_r X]_j + [U_r h_{<t-1>}]_j)$$

Reset gate는 과거의 정보를 **얼마나 제거할 지(얼마나 무시할 지) 결정**하는 단계
-> **과거의 정보인** $h_{<t-1>}$와 **현재의 input** X_j를 얼마나 혼합할지를 결정

위의 계산 과정의 결과값은 **sigmoid의 출력값**이기 때문에
0~1사이의 값이 출력

해당 **출력값이** $h_{<t-1>}$**를 얼마나 사용할지에 대한 정보**
0.4가 출력되면 과거의 정보를 **40%** 사용한다는 의미

LSTM과 비슷하게 작동

0에 가까워지면 hidden state는 **Previous hidden state를 무시**하고 **현재 input으로 리셋**

- The update gate (z_j)

$$z_j = \sigma([W_z X]_j + [U_z h_{<t-1>}]_j)$$

Update gate는 과거의 정보(previous hidden state)를 **얼마나 사용할 지를 결정하는 단계 (reset gate와 비슷)**
즉, h_t를 **새로운 hidden state** $\tilde{h}_t$로 **업데이트 여부 결정**

Previous hidden state에서 현재 **hidden state로 얼마나 많은 정보를 전달**할 지 결정

똑같이 **sigmoid의 출력값**이기 때문에 0~1사이의 값이 출력

뒤의 $h_j^{<t>}$ 수식에서 알 수 있다시피, z_j**의 값이 1에 수렴**하면, h_t 를 **update**할 때 **새로운 hidden state** $\tilde{h}_t$만 사용(previous hidden state 사용x)

σ : logistic sigmoid function
$[.]_j$: j –th element of a vector X
h_{t-1} : the input and previous hidden state
W_r , U_r : weight matrices

03 Proposed Method

Hidden Unit that Adaptively Remembers and Forgets (GRU)

- The actual activation of proposed unit h_j

$$\boldsymbol{h_j^{<t>} = z_j h_j^{<t-1>} + (1 - z_j)\tilde{h}_j^{<t>}}$$

Where $\tilde{h}_j^{<t>} = \phi([Wx]_j + [U(r \odot h_{<t-1>}]_j)$

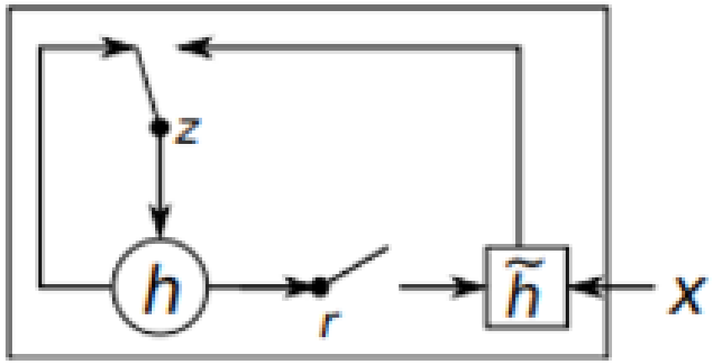
$\tilde{\boldsymbol{h}}_{\boldsymbol{j}}^{<\boldsymbol{t}>}$는 previous hidden state와 current input값들을 조합해서 **새로운 "현재 값"을 구함**

$[\boldsymbol{U(r \odot h_{<t-1>}]_j}$ 이 수식은 **reset gate를 통해 얻은 필터링 비율**에 **previous hidden state**를 곱해 **과거 값을 필터링**

$\boldsymbol{h_j^{<t>}}$는 reset gate를 통해 얻은 **새로운 현재 값($\tilde{\boldsymbol{h}}_{\boldsymbol{j}}^{<\boldsymbol{t}>}$)**과 **과거의 정보(previous hidden state, $\boldsymbol{h_{<t-1>}}$)**을 **update gate의 출력값**을 사용해 새롭게 구한 **현재의 hidden state**

즉, h_j는 reset,update gate를 모두 지나 과거의 정보를 적절하게 사용해 구한 현재의 hidden state

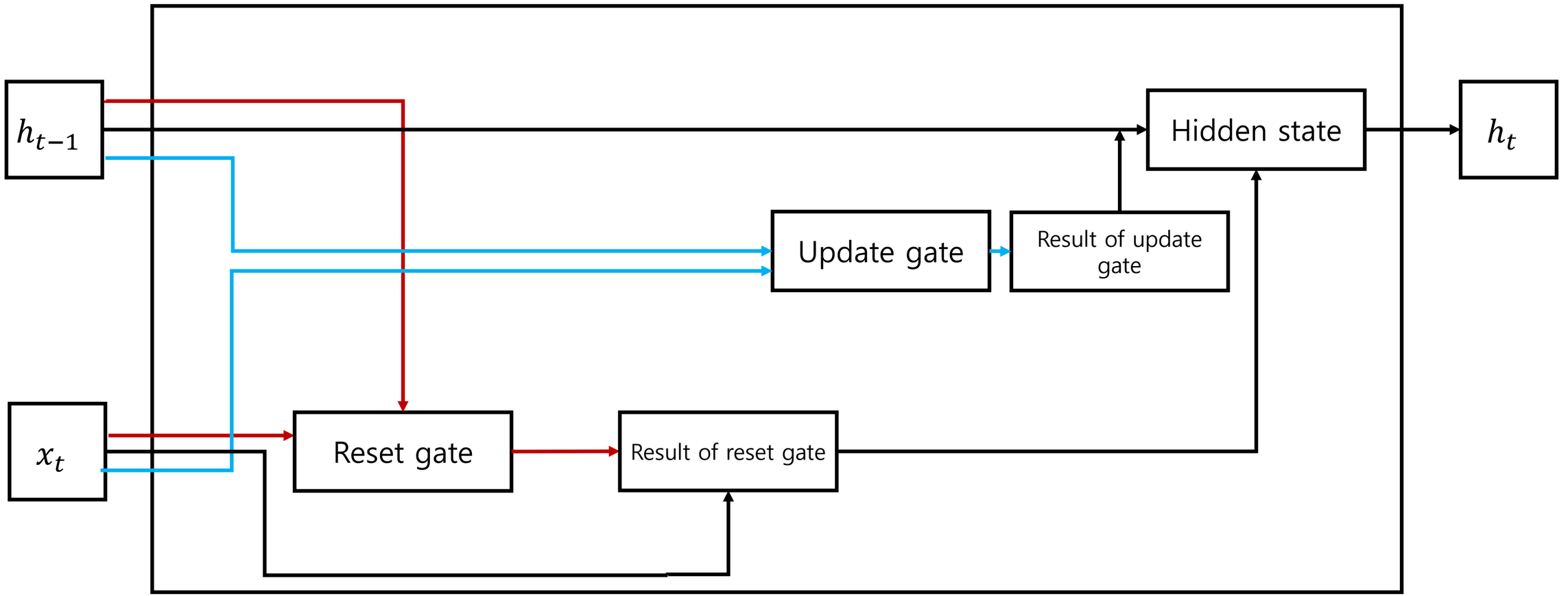
Layout of GRU



Hidden state h_j를 구하기 위해 정보를 얼마나 사용할 지를 $\tilde{h}_j^{<t>}$, z_j , r_j로 결정하는 unit -> **LSTM과 유사**하지만 더 효율적이고 간단함

03 Proposed Method

Hidden Unit that Adaptively Remembers and Forgets (GRU)



04 Experiment

English French translation

- The bilingual corpora include Europarl (61M words), news commentary (5.5M words), UN (421M words), and two crawled corpora of 90M and 780M words respectively. The last corpora are quite noisy.

- **To train the French language model, about 712M words of crawled newspaper** material is available in addition to the target side of the bitexts.

- **A subset of 418M words is selected out of more than 2G words** for language modeling and a subset **of 348M out of 850M words is selected for training the RNN Encoder-Decoder**.

- The baseline phrase-based SMT system was built using **Moses** with default settings.

- The proposed **RNN Encoder-Decoder** uses **1000 hidden units** with the **proposed gates** at the encoder gates at the encoder and at the decoder.

- The activation function for h is a **hyperbolic tangent** (tanh).

04 Experiment

English French translation

- **In WMT’ 14 workshop**

Task : English/French translation task

- **Evaluation (BLUE-score)**

$$\text{BLUE} = \min\left(1, \frac{output\ length(\text{예측 문장})}{referenece\ length(\text{실제 문장})}\right)\left(\prod_{i=1}^{4} precision_i\right)^{\frac{1}{4}}$$

- **comparison model**

$$SMT_{base} = \sum_{n=1}^{N-1} w_n f_n(\text{프랑스}|\text{영어})$$

$$SMT_{ours} = \sum_{n=1}^{N-1} w_n f_n(\text{프랑스}|\text{영어}) + w_s SeqtoSeq(\text{프랑스}|\text{영어})$$

04 Experiment

English French translation

- **결과**

Seq-to-Seq(RNN)을 **Feature로 추가**한 모델의 성능이 향상되었음

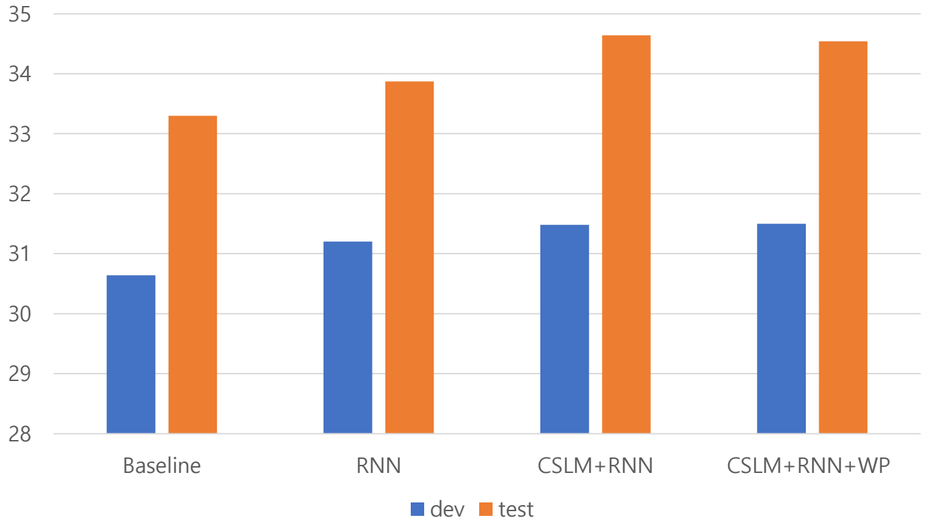
RNN : Original RNN language model

CSLM : traditional approach of using a neural network for learning a target language model (7-grams)

WP : word penalty

CSLM+RNN+WP -> 해당 논문에서 제시하는 최종 모델
-> dev,test data에서 모두 좋은 성능을 보여줌

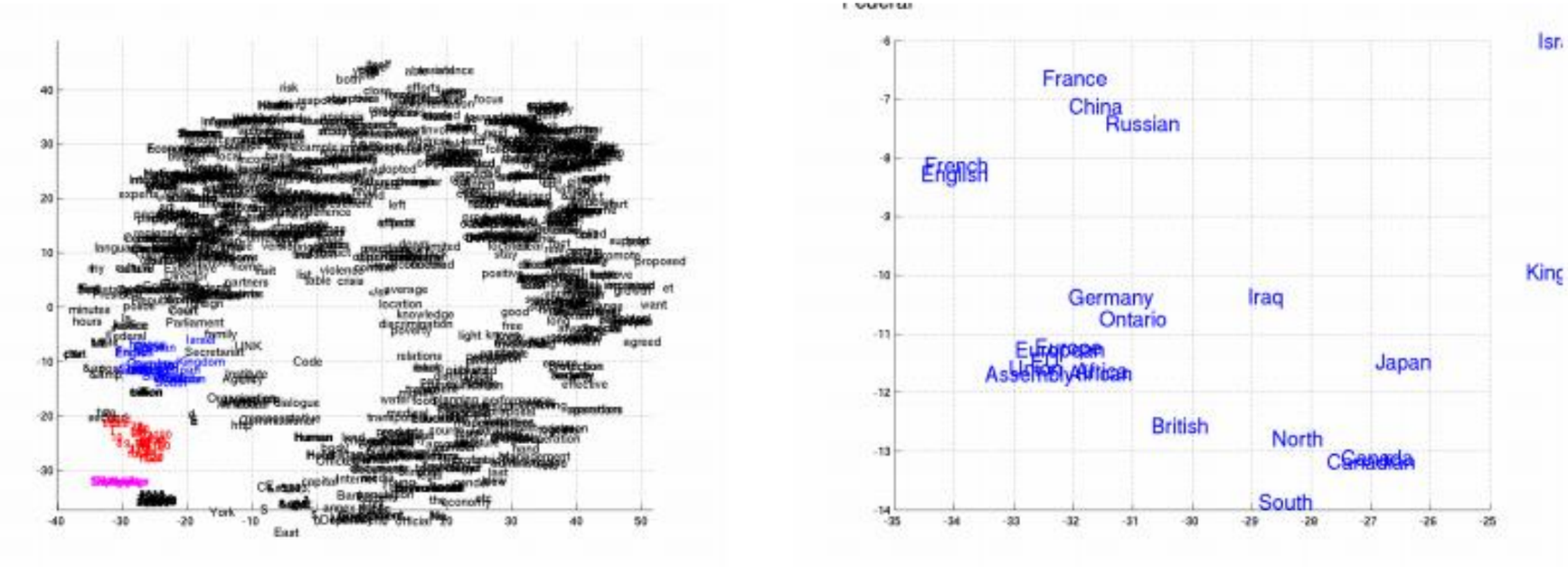
Models	BLEU	
	dev	test
Baseline	30.64	33.30
RNN	31.20	33.87
CSLM + RNN	31.48	34.64
CSLM + RNN + WP	31.50	34.54



04 Experiment

English French translation

Figure 4 : The left one shows the **full embedding space**, while the right one shows **a zoomed-in view of one region**



단어들을 embedding한 값을 시각화한 자료

비슷한 위치에 존재하는 단어들은 **문법적,맥락적으로 비슷한 의미**를 가지는 단어들

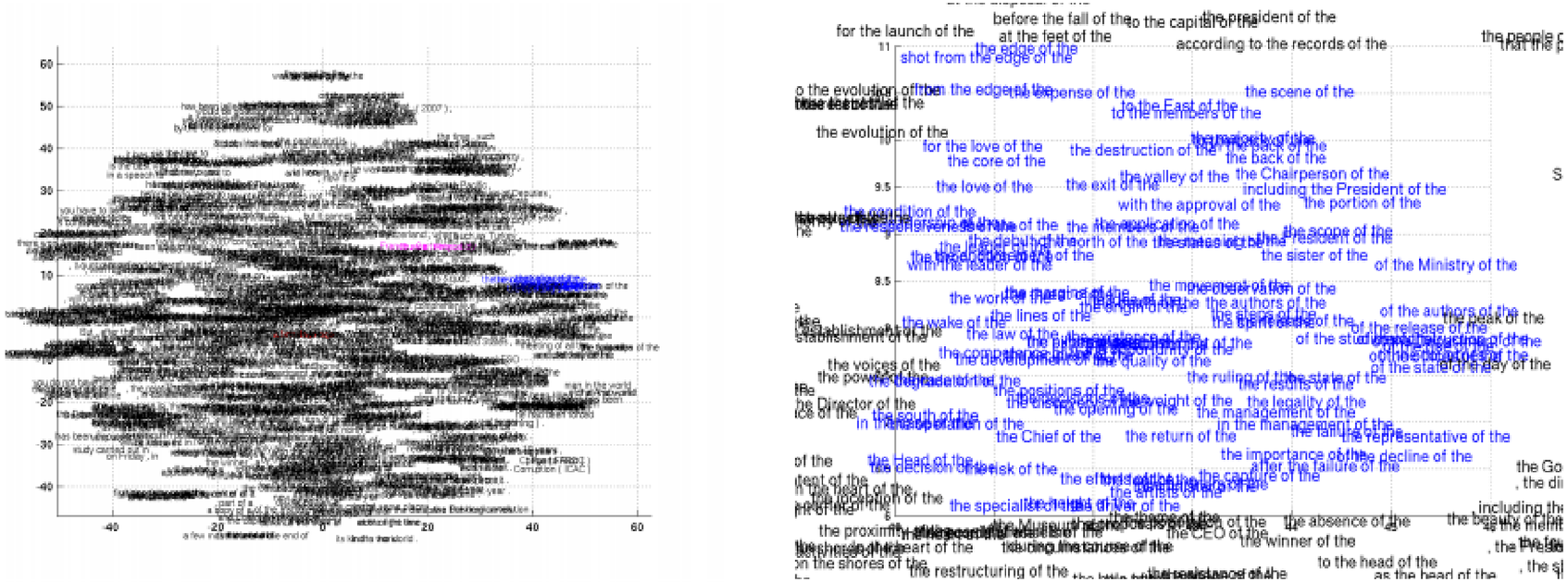
EX) English – French -> 영어 – 프랑스어

뜻은 다르지만 각각 '한 나라의 언어'임이 비슷하므로 비슷한 자리로 배치됨

04 Experiment

English French translation

Figure 5 : The right one shows **2-D embedding of the learned phrase representation**, while the left one shows the **full representation space (5000 randomly selected points)**



앞 슬라이드와 비슷하게 sequence들을 embedding 한 값을 시각화 한 자료

비슷한 문법,맥락,의미를 가지는 sequence끼리 비슷한 위치에 존재하는 것을 알 수 있음

EX) for the love of the – the love of the
위 둘의 문장들은 문법적,맥락적으로 비슷한 것을 알 수 있음

05 Conclusion

RNN encoder-decoder, GRU, seq2seq

- **가변적인 input, output sequence**를 다룰 수 있는 **RNN Encoder-Decoder**를 배움
  - 일반적인 SMT보다 **정확**하고 **빠른 성능**을 보이는 encoder-decoder를 제시

- RNN cell 중 **LSTM과 성능이 유사**하고 **더 간단한 unit인 GRU**를 배움
  - **reset gate, update gate**를 통해 **과거의 정보량을 조절**해서 **hidden state를 생성하는 unit**

- translation에서 **neural network의 가능성을 제시**함
  - RNN,LSTM,seq2seq 등 NN 모델의 성능이 좋음을 보여줌

Q & A

감사합니다