

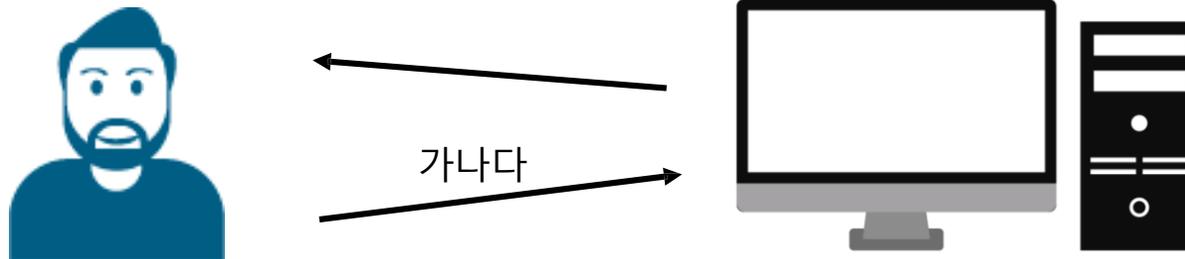
2023 AI(인공지능) 특화교육: RNN & 자연어처리

2023.11.13

전북대학교 컴퓨터인공지능학부
인지컴퓨팅 연구실

자연어 처리

- 자연어 (Natural Language)
 - 인간이 일상 생활에서 의사소통을 위해 사용하는 언어
 - 컴퓨터가 사용하는 인공어(기계어, 프로그래밍 언어)와 대비되는 개념
- 자연어 처리 (Natural Language Processing)
 - 컴퓨터에게 인간과 매우 유사한 방식으로 텍스트 및 음성 언어를 이해하는 능력을 부여하는 것과 관련된 컴퓨터 공학의 한 분야, 더 구체적으로 말하자면 인공지능의 한 분야



자연어 처리(NLP) 응용 분야

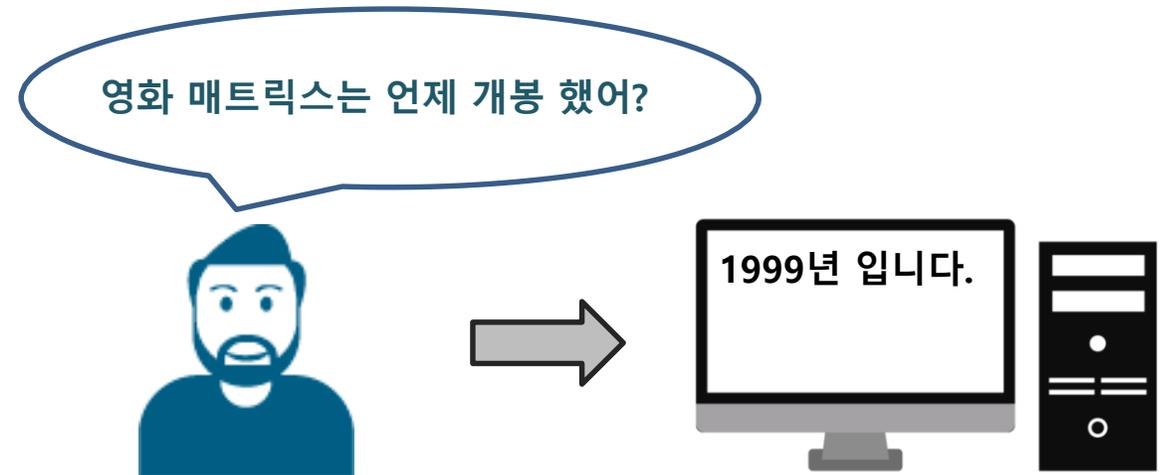
- 감성 분석 (Sentiment Analysis)

- 텍스트에 나타난 사람들의 태도, 의견, 성향과 같은 주관적인 데이터를 분석하는 자연어 처리 기술



- 질의 응답 (Question Answering)

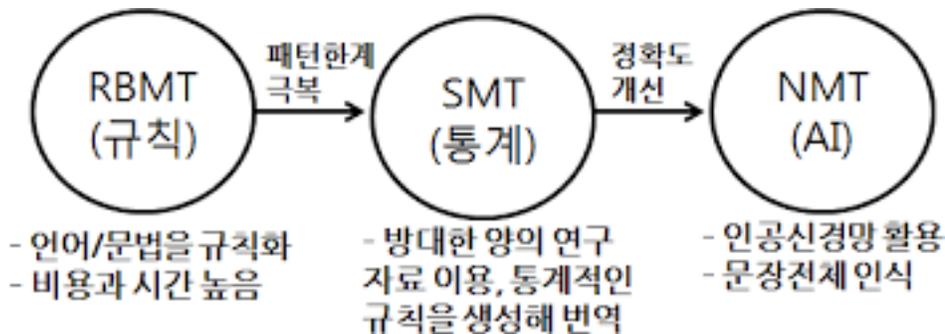
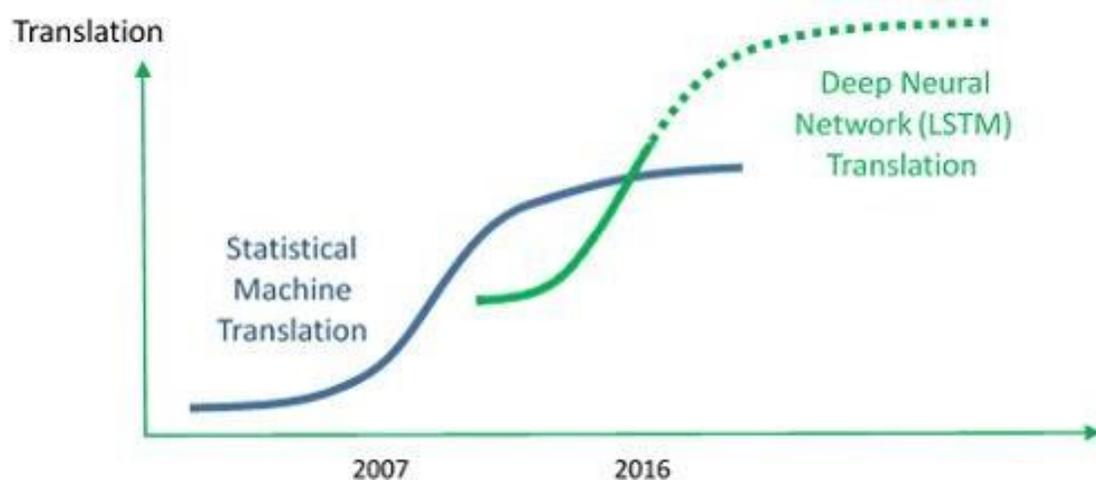
- 사람이 제시한 질문에 대해 자동으로 응답하는 시스템을 구축하는 자연어 처리 기술



자연어 처리(NLP) 응용 분야

- 기계 번역 (Machine Translation)

- 2014년 이후 seq2seq라는 모델 구조가 소개되며 end-to-end 신경망 기반 기계 번역 (neural machine translation)의 시대가 열림



<NMT 어플리케이션>



자연어 처리(NLP) 응용 분야

- 인공지능 비서 (시리, 빅스비 등)
 - 현재 많은 IT 기업들이 인공지능 비서를 상용화하여 배포 중



Siri (Apple)



Bixby(Samsung)

자연어 처리(NLP) 응용 분야

- Chatbot(대화 시스템)

GPT-4 Turbo 업데이트

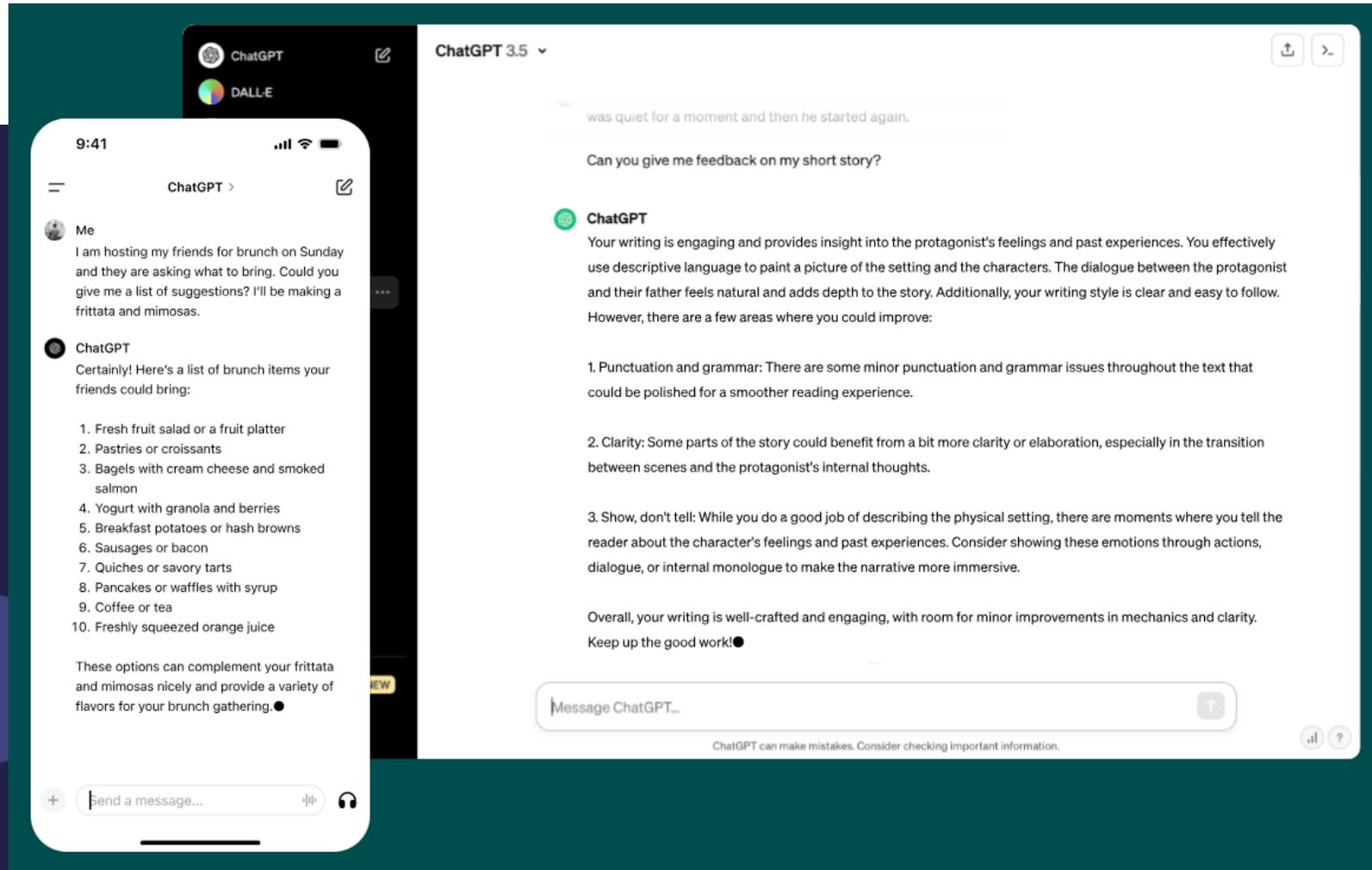
더 빠르고, 사진도 읽는 뤼튼

뤼튼에서 곧 무료 무제한으로 GPT-4 Turbo를 사용할 수 있습니다.
미리 체험해보세요!



지금 체험하기 >

뤼튼



ChatGPT 3.5

was quiet for a moment and then he started again.

Can you give me feedback on my short story?

ChatGPT

Your writing is engaging and provides insight into the protagonist's feelings and past experiences. You effectively use descriptive language to paint a picture of the setting and the characters. The dialogue between the protagonist and their father feels natural and adds depth to the story. Additionally, your writing style is clear and easy to follow. However, there are a few areas where you could improve:

1. Punctuation and grammar: There are some minor punctuation and grammar issues throughout the text that could be polished for a smoother reading experience.
2. Clarity: Some parts of the story could benefit from a bit more clarity or elaboration, especially in the transition between scenes and the protagonist's internal thoughts.
3. Show, don't tell: While you do a good job of describing the physical setting, there are moments where you tell the reader about the character's feelings and past experiences. Consider showing these emotions through actions, dialogue, or internal monologue to make the narrative more immersive.

Overall, your writing is well-crafted and engaging, with room for minor improvements in mechanics and clarity. Keep up the good work! 🌟

Message ChatGPT...

ChatGPT can make mistakes. Consider checking important information.

ChatGPT

전처리(Preprocessing)

- Text Preprocessing

- 용도에 맞게 텍스트를 사전에 처리하는 작업

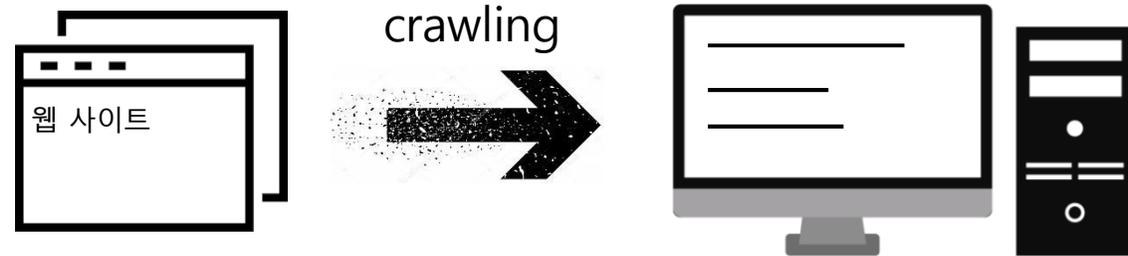
- 코퍼스(Corpus) (말뭉치)

- 여러 단어들로 이루어진 문장의 뭉치

- 자연어 처리 분야의 머신러닝, 딥러닝을 수행하려면 많은 수의 코퍼스가 필요함

- 코퍼스 수집 방법

1. 공개된 데이터 이용
<https://www.aihub.or.kr/>
2. 코퍼스 구매
3. 크롤링을 이용한 수집



- 무작정 웹사이트에서 코퍼스를 크롤링하면 법적인 문제로 이어질 수 있기 때문에 적절한 웹사이트에서, 올바른 방법으로, 상업적인 목적이 아닌 경우로 제한된 크롤링을 권장함

Word Embedding

- Word Embedding

- 자연어로 이루어진 단어를 고정된 차원의 실수 벡터로 변환시키는 과정

- 단어를 벡터로 표현 하는 방법

- 희소 표현(Sparse Representation)

- One-Hot 인코딩을 통해 단어를 벡터를 표현하는 방법으로 표현하고자 하는 단어의 인덱스의 값만 1로 하고 나머지는 0으로 표현하는 방법

- 단어의 개수가 늘어나면 벡터의 차원이 한없이 커진다는 단점 존재($|V| \times |V|$)하고 단어의 의미를 잘 담을 수 없어 현재는 거의 사용하지 않음

- 밀집 표현(Dense Representation)

- 단어에 대해 특정 다차원 차원(d)을 설정하고 각 벡터는 0,1의 값이 아닌 실수 값을 가짐

- 단어의 차원이 늘어나도 벡터의 차원은 고정($|V| \times d$)하고 단어의 의미를 잘 표현할 수 있는 방식



Word Embedding

- Word Embedding

- 학습 데이터로부터 vocab 생성

- Vocab에 없는 단어는 UNK 할당
- Embedding Matrix
 - Vocab_size(|V|) * Embedding dim(d)

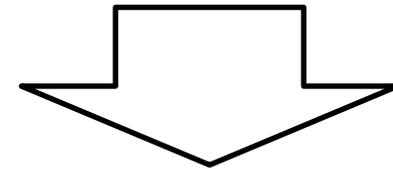
Embedding Matrix

$$|V| \times d$$

Vocabulary

Word	Index
[PAD]	0
[UNK]	1
Anyway	2
,	3
I	4
love	5
you	6
.	7

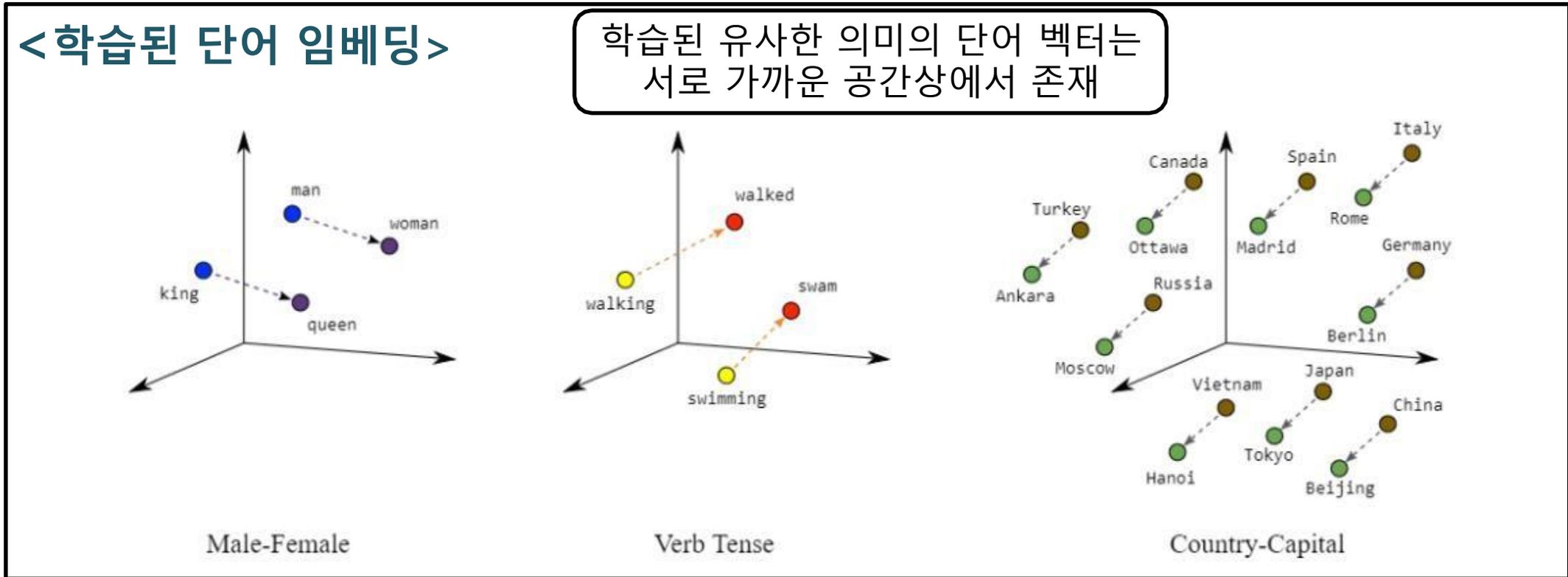
Train Sentence	Anyway , I love you .
Dev Sentence	I hate you .
Test Sentence	I will kill you .



Train Sentence	[2 3 4 5 6 7]
Dev Sentence	[4 1 6 7 0 0]
Test Sentence	[4 1 1 6 7 0]

How to Train Word Embedding?

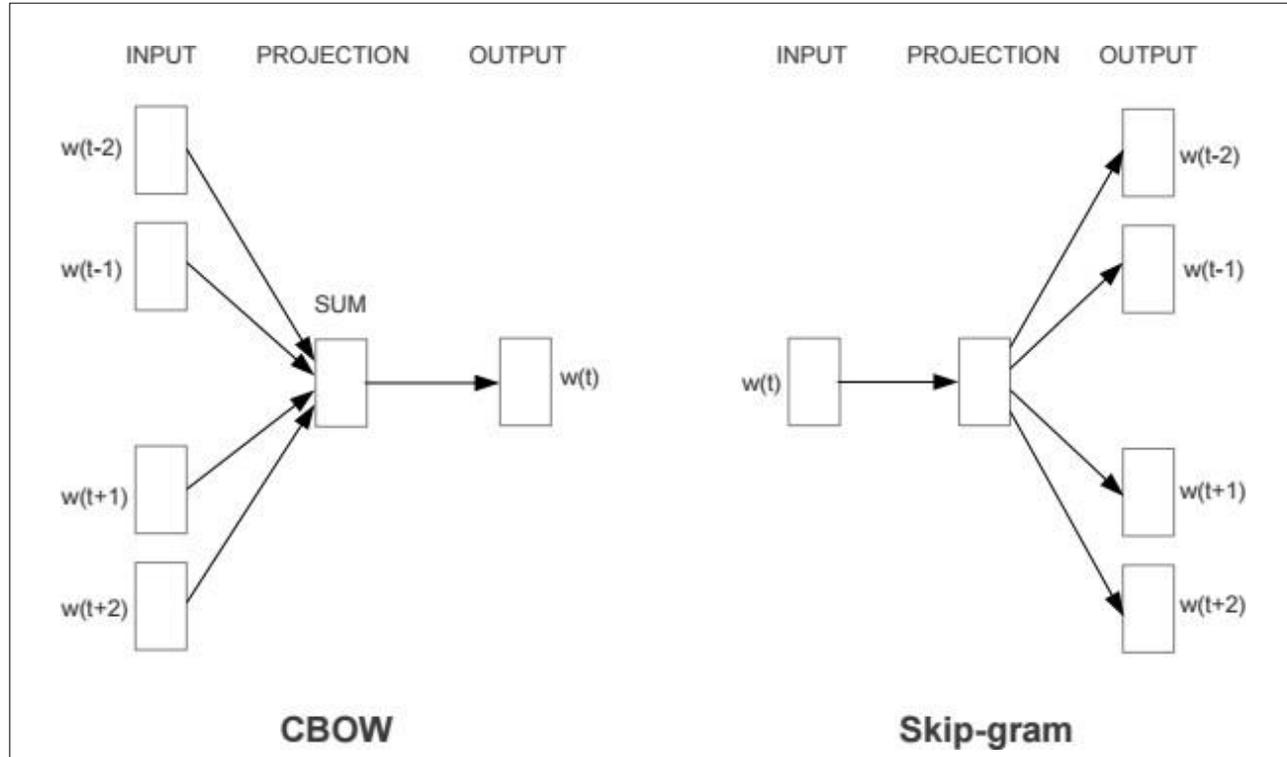
- 딥러닝 기반 Word Embedding 학습
 - 학습된 벡터는 유사한 의미 공간 속에 존재



- 단어 임베딩을 학습하는 알고리즘
 - Word2Vec(CBOW, Skip-Gram), Glove, Fasttext 등등

How to Train Word Embedding?

- Word2Vec



– CBOW

- 주어진 단어에 대해 주위의 앞 뒤의 몇 개의 단어를 입력으로 하여 주어진 단어를 유추하는 모델

– Skip-gram

- CBOW와 반대로 주어진 단어 하나를 이용하여 주위에 등장하는 몇 개의 단어들의 등장여부를 예측

<주위의 단어를 얼마나 볼지에 대한 Window Size 중요>

How to Train Word Embedding?

- Glove

동시 등장 확률과 크기 관계 비(ratio)	k=solid	k=gas	k=water	k=fasion
P(k ice)	0.00019	0.000066	0.003	0.000017
P(k steam)	0.000022	0.00078	0.0022	0.000018
P(k ice) / P(k steam)	8.9	0.085	1.36	0.96

- 단어의 동시 발생 확률을 모델링

- 데이터에 대해 두 단어가 동시에 등장하는 횟수와 그 확률을 계산
- 임베딩 된 중심 단어와 주변 단어 벡터의 내적이 전체 학습 데이터에서의 동시 등장할 log 확률.

$$w_i^T \tilde{w}_k + b_i + \tilde{b}_k = \log X_{ik}$$

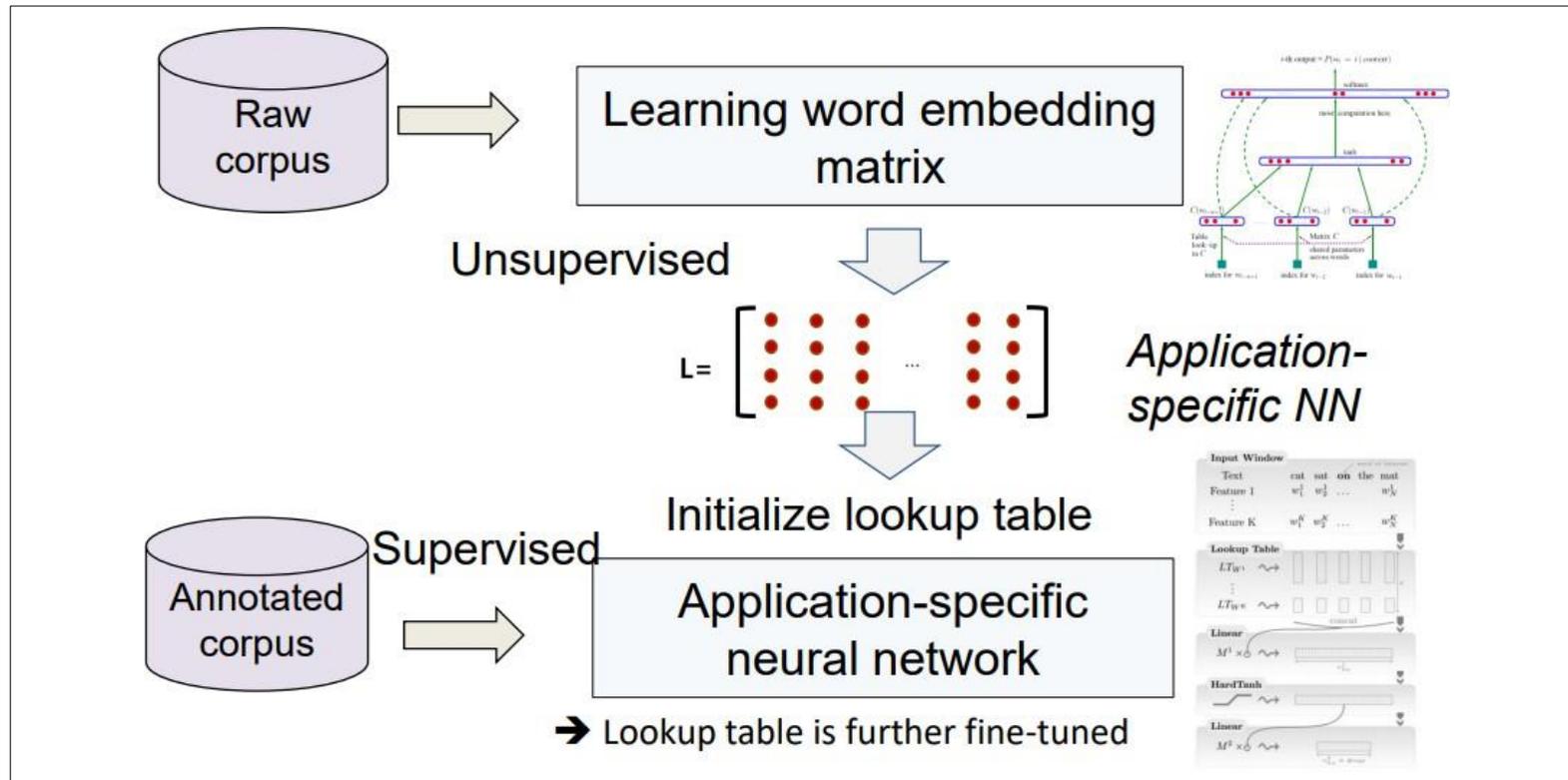
- 손실 함수

- 두 벡터 n,m의 내적 값과 실제 동시 등장 로그 확률과의 오차가 최소가 되도록 설정
- 코퍼스 내의 모든 단어 쌍에 대해 적용

$$Loss\ function = \sum_{m,n=1}^V (w_m^T \tilde{w}_n + b_m + \tilde{b}_n - \log X_{mn})^2$$

How to Train Word Embedding?

- Word Embedding을 이용한 자연어 처리
 - Raw 코퍼스로부터 word embedding matrix 학습
 - 응용 태스크에서 embedding matrix 미세조정(fine-tuning)
 - 보통 응용 태스크의 embedding matrix는 랜덤 초기화 수행하지만 사전 학습된 embedding matrix 존재하면 성능 향상



한국어 Word Embedding

한국어 자연어 처리의 어려움

- 한국어는 교착어에 속함
- 어순이 중요시되는 영어나 중국어와 달리 어근에 접사가 붙어 의미와 문법적 기능이 부여됨
- 이러한 이유로 한국어 자연어처리에는 형태소 분석(POS tagging) 등이 필요함
 - 형태소 : 뜻을 지니고 있는 최소의 단위

원형	피동	높임	과거	추측	전달	결과
잡						+다 잡다
잡	+히					+다 잡히다
잡	+히	+시				+다 잡히시다
잡	+히	+시	+었			+다 잡히셨다
잡			+았(었)			+다 잡았다
잡				+겠		+다 잡겠다
잡					+더라	잡더라
잡		+히	+었			+다 잡혔다
잡		+히	+었	+겠		+다 잡혔겠다
잡	+히	+었	+겠		+더라	잡혔겠더라
잡			+았(었)	+겠		+다 잡았겠다
...						...
잡	+히	+시	+았(었)	+겠	+더라	잡히시었겠더라

<원형인 '잡다'에서 파생될 수 있는 다양한 형태들>

한국어 Word Embedding

• 한국어 Word Embedding

– 어절(단어) 단위 [나는, 학교에, 갔다.]

- OOV(out of vocabulary) 문제에 취약
 - 한국어는 교착어의 특성을 지님

– 음절 단위

- OOV 문제가 거의 발생하지 않음
- 음절 단위로 의미를 표현하는 것은 한계 존재
- 아무런 뜻을 지니지 않음

– 형태소 단위

- 형태소는 뜻을 지니는 최소의 단위
- OOV 문제에 대해서도 강건함
- 형태소 분석기 성능이 중요

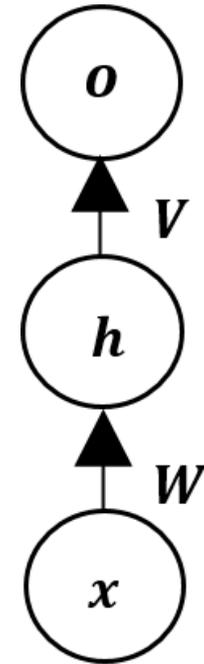
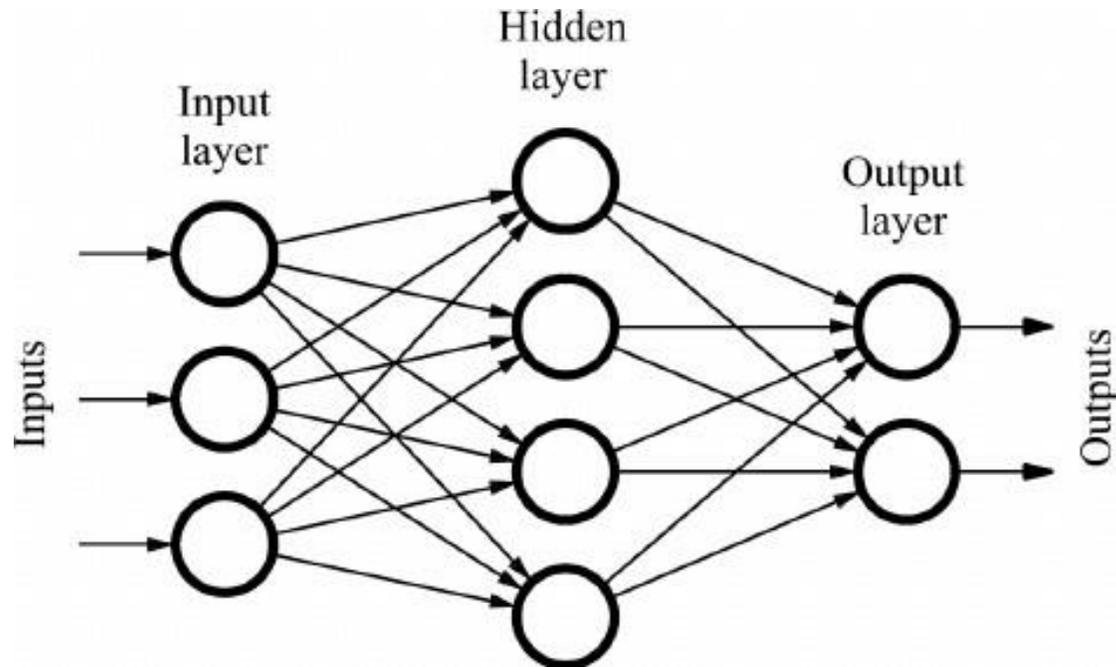
입력문장

나는 오늘 대학교에 갔다.

어절 단위	[나는, 오늘, 대학교에, 갔다.]
음절 단위	[나, 는, 오, 늘, 대, 학, 교, 에, 갔, 다, .]
형태소 단위	[나, 는, 오늘, 대학교, 에, 갔, 다, .]

Feed-Forward Neural Network

- FNN(Feed-Forward Neural Network)
 - 가장 간단한 형태의 인공 신경망
 - 입력층과 출력층 사이에 은닉층을 둬
 - 먼저, 입력 x 와 파라미터 W 와 행렬 연산 후 활성화함수 g 를 취하여 은닉 벡터 h 를 만들어 내고 은닉 벡터에 파라미터와 행렬연산 V 을 하여 출력 층으로 연결



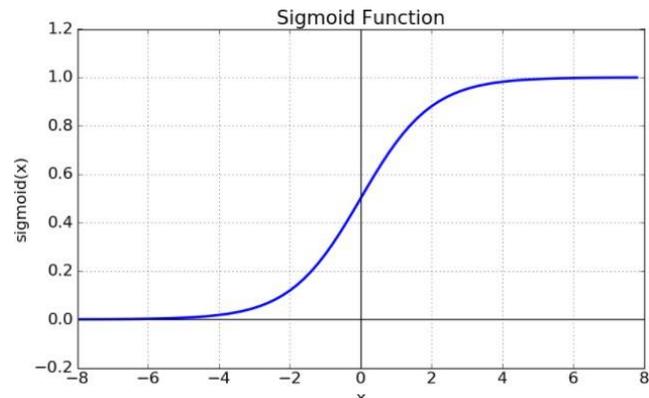
Feed-Forward Neural Network

- FNN(Feed-Forward Neural Network)

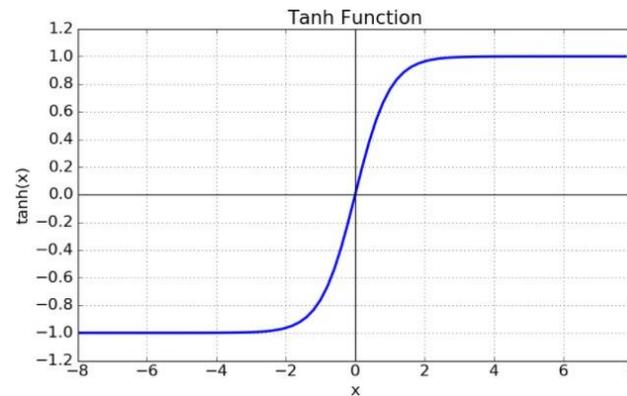
- 활성화함수

- 노드에 들어오는 값을 바로 다음 레이어로 전달하지 않고 비선형 변환을 수행

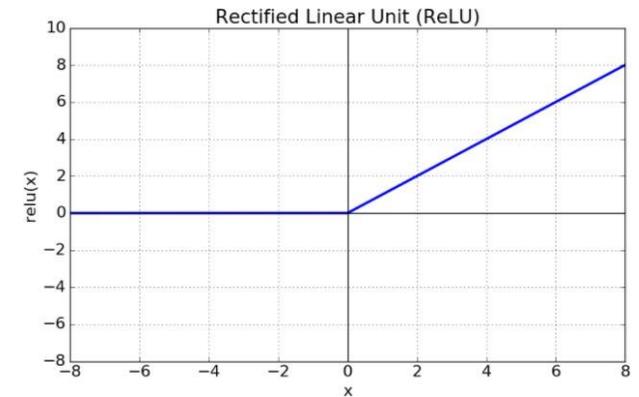
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$



$$f(x) = \max(0, x)$$

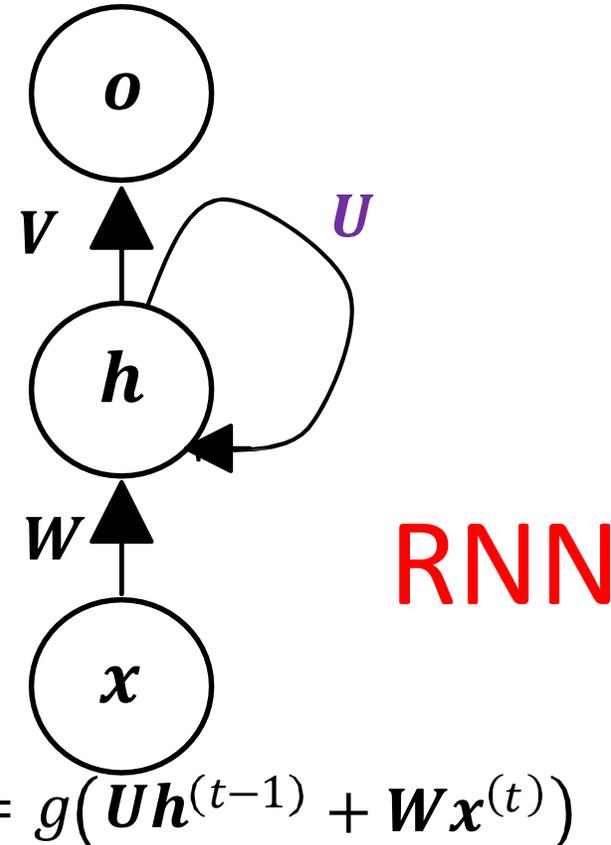
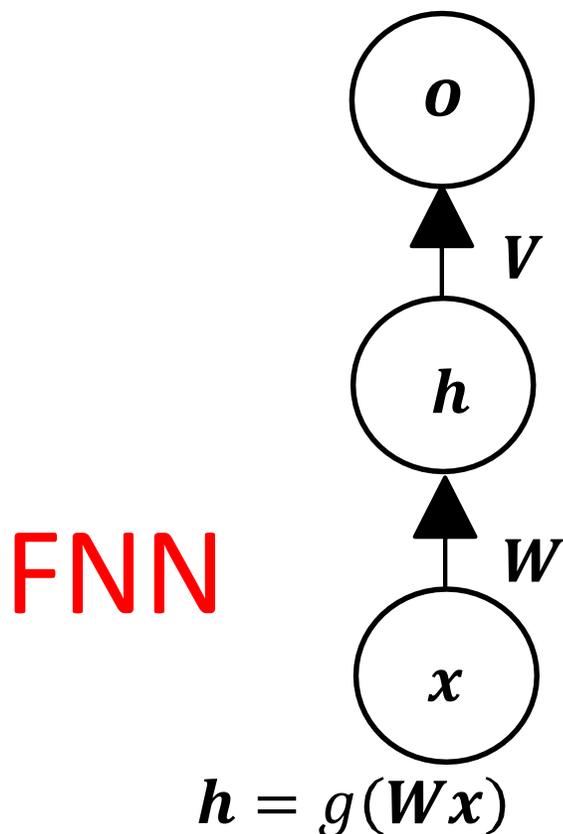


Recurrent Neural Networks

- RNN(Recurrent Neural Networks)

- 순환 신경망

- 시퀀스 데이터를 모델링 할 수 있는 뉴럴 신경망
- RNN에서는 히든 노드가 방향을 가진 엣지로 연결되어 순환 구조를 이룸
- 입력 뿐 아니라 이전 히든 상태를 받기 위해 별도의 U 파라미터를 추가로 둬



RNNs : Update Formula

- Update Formula

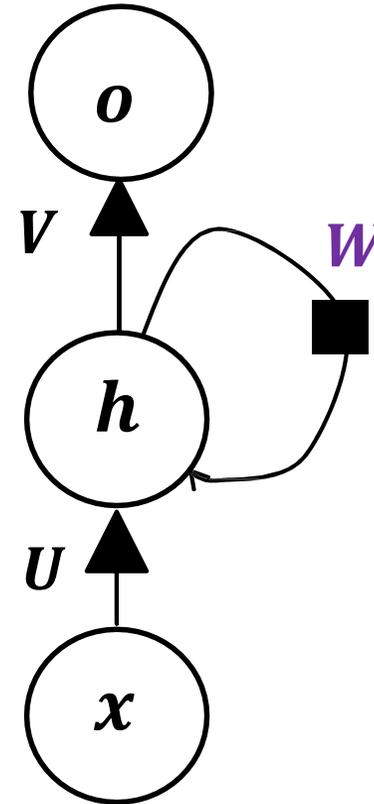
- 은닉 상태의 계산

$$\mathbf{h}^{(t)} = \mathbf{f}(\mathbf{x}^{(t)}, \mathbf{h}^{(t-1)})$$

$$\mathbf{h}^{(t)} = \tanh(\mathbf{W}\mathbf{x}^{(t)} + \mathbf{U}\mathbf{h}^{(t-1)})$$

- 은닉 상태 → 출력

$$\mathbf{o}^{(t)} = \mathbf{V}\mathbf{h}^{(t)}$$



Using explicit bias terms

$$\mathbf{h}^{(t)} = \tanh(\mathbf{W}\mathbf{x}^{(t)} + \mathbf{U}\mathbf{h}^{(t-1)} + \mathbf{b})$$

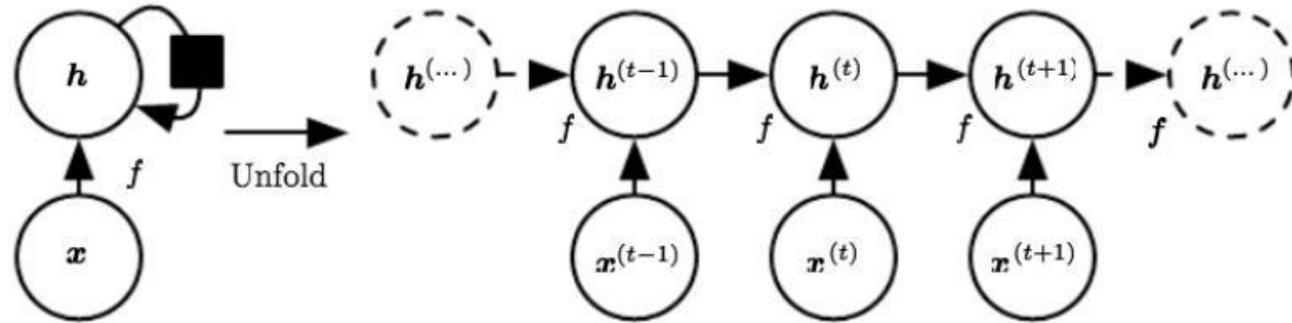
$$\mathbf{o}^{(t)} = \mathbf{V}\mathbf{h}^{(t)}$$

Recurrent Neural Networks

- RNN(Recurrent Neural Networks)

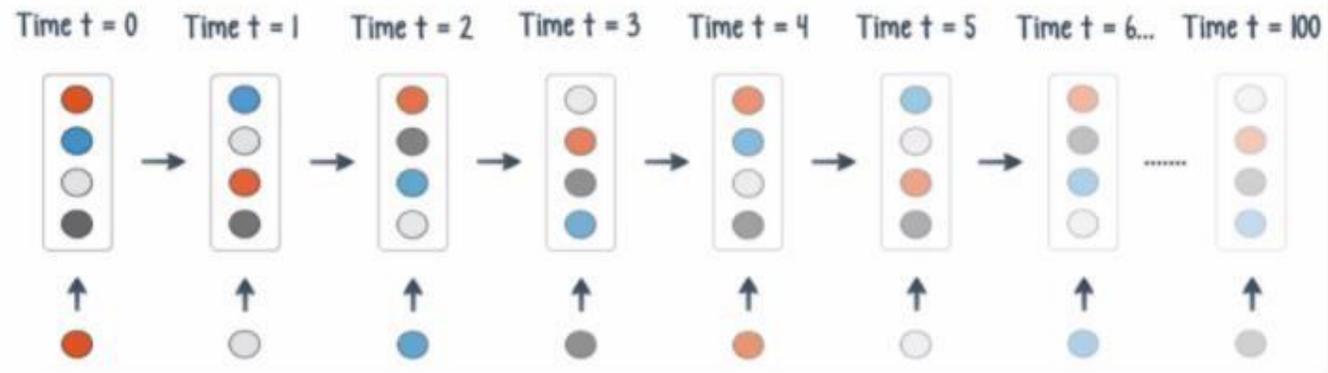
- Unfold

- 각 time step t 에 따라 입력 시퀀스를 모델링

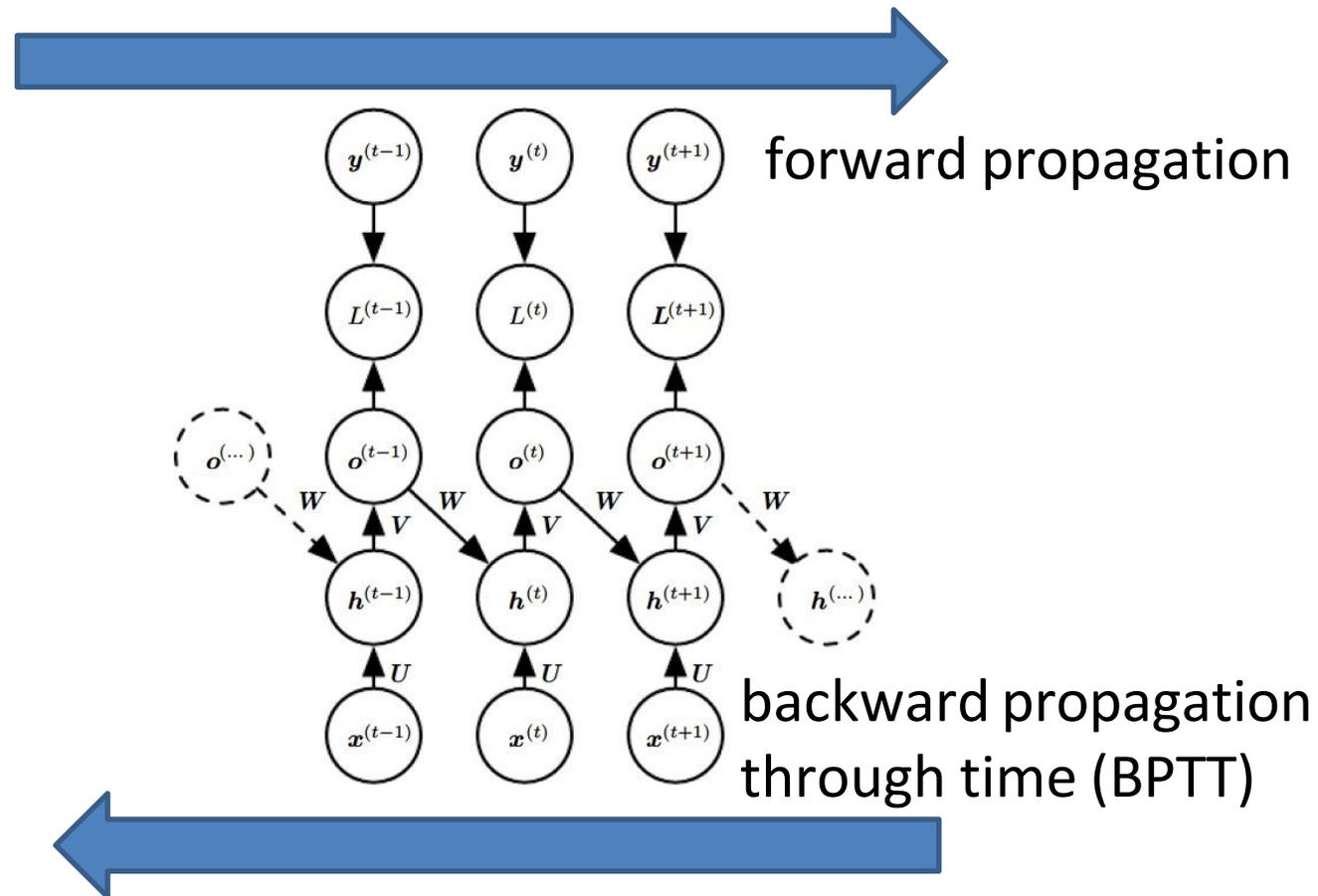


- 한계점

- Gradient vanishing 문제
 - 역전파 시 Time Step이 너무 길어지면 기울기가 전달되지 못하고 소멸되는 문제가 발생

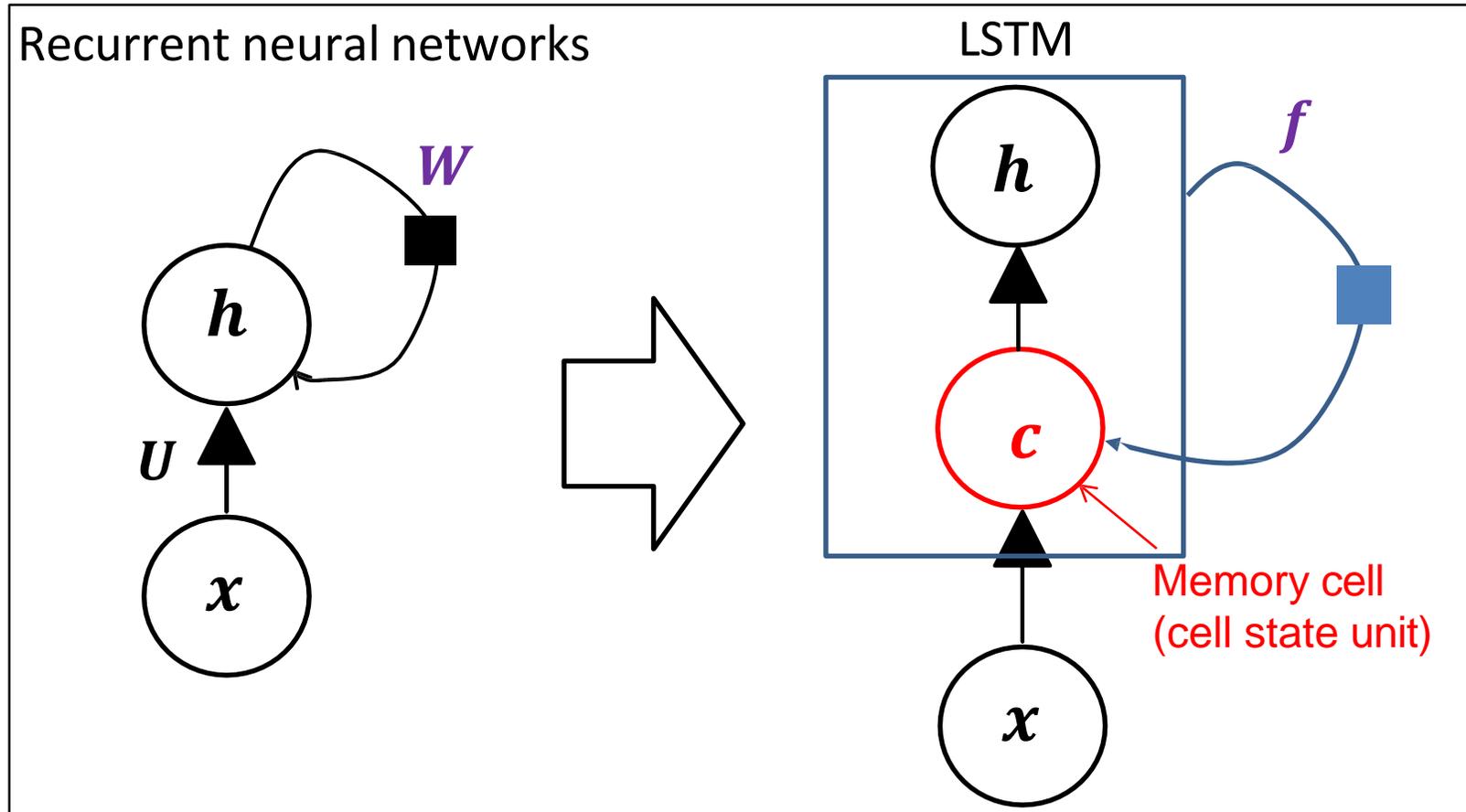


Backpropagation through Time (BPTT)



Long Short Term Memory

- LSTM(Long Short Term Memory)
 - 장거리 의존성(long-term dependencies)를 모델링
 - Memory Cell의 도입
 - 3개의 Gate를 이용하여 Memory Cell 및 출력 값을 조정



Long Short Term Memory

- LSTM(Long Short Term Memory)

- 은닉 층에서 3개의 Gate를 이용

- input-gate

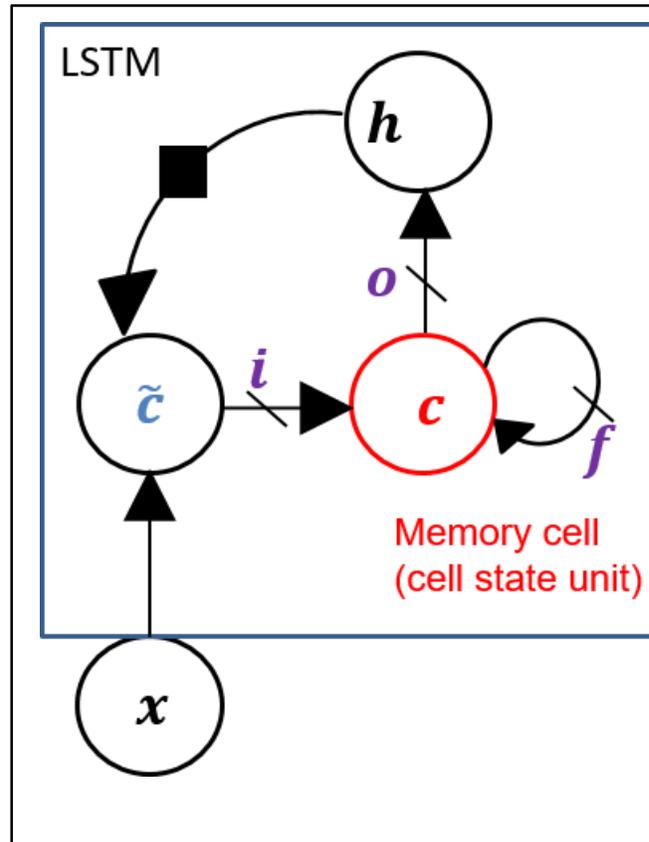
- 입력 값의 정보를 얼마나 메모리 cell에 전달할지를 결정

- forget-gate

- 현재 메모리 cell의 정보를 얼마나 제거할지를 결정

- output-gate

- 메모리 cell의 정보를 얼마나 출력할지를 결정



Computing gate values

$$f^{(t)} = g_f(x^{(t)}, h^{(t-1)}) \text{ (forget gate)}$$

$$i^{(t)} = g_i(x^{(t)}, h^{(t-1)}) \text{ (input gate)}$$

$$o^{(t)} = g_o(x^{(t)}, h^{(t-1)}) \text{ (output gate)}$$

(new memory cell)

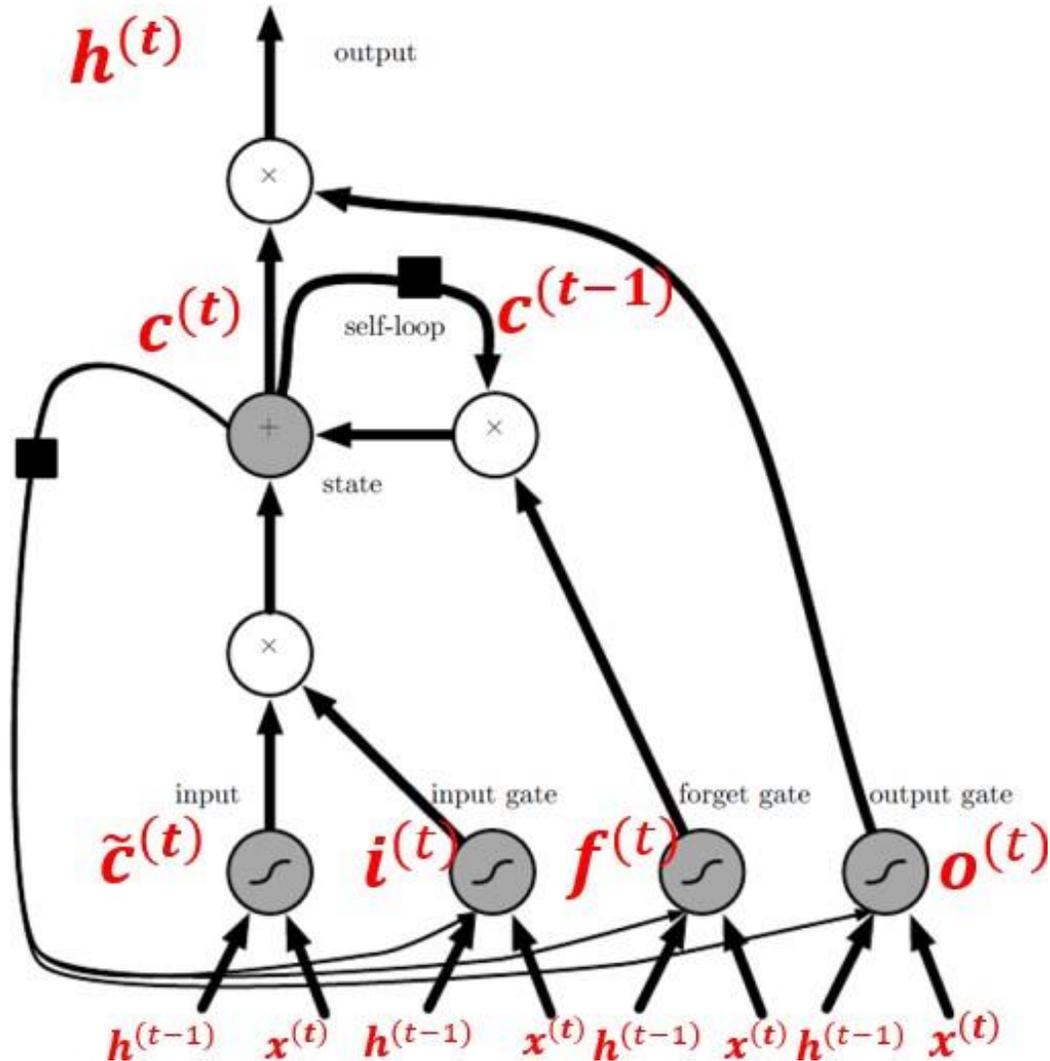
$$\tilde{c}^{(t)} = \tanh(W^{(c)}x^{(t)} + U^{(c)}h^{(t-1)})$$



$$c^{(t)} = i^{(t)} \circ \tilde{c}^{(t)} + f^{(t)} \circ c^{(t-1)}$$
$$h^{(t)} = o^{(t)} \circ \tanh(c^{(t)})$$

Long Short Term Memory

- Update Formula



- $i^{(t)} = \sigma(W^{(i)}x^{(t)} + U^{(i)}h^{(t-1)})$ (Input gate)
- $f^{(t)} = \sigma(W^{(f)}x^{(t)} + U^{(f)}h^{(t-1)})$ (Forget gate)
- $o^{(t)} = \sigma(W^{(o)}x^{(t)} + U^{(o)}h^{(t-1)})$ (Output/Exposure gate)
- $\tilde{c}^{(t)} = \tanh(W^{(c)}x^{(t)} + U^{(c)}h^{(t-1)})$ (New memory cell)
- $c^{(t)} = f^{(t)} \circ c^{(t-1)} + i^{(t)} \circ \tilde{c}^{(t)}$ (Final memory cell)
- $h^{(t)} = o^{(t)} \circ \tanh(c^{(t)})$

Gated Recurrent Unit

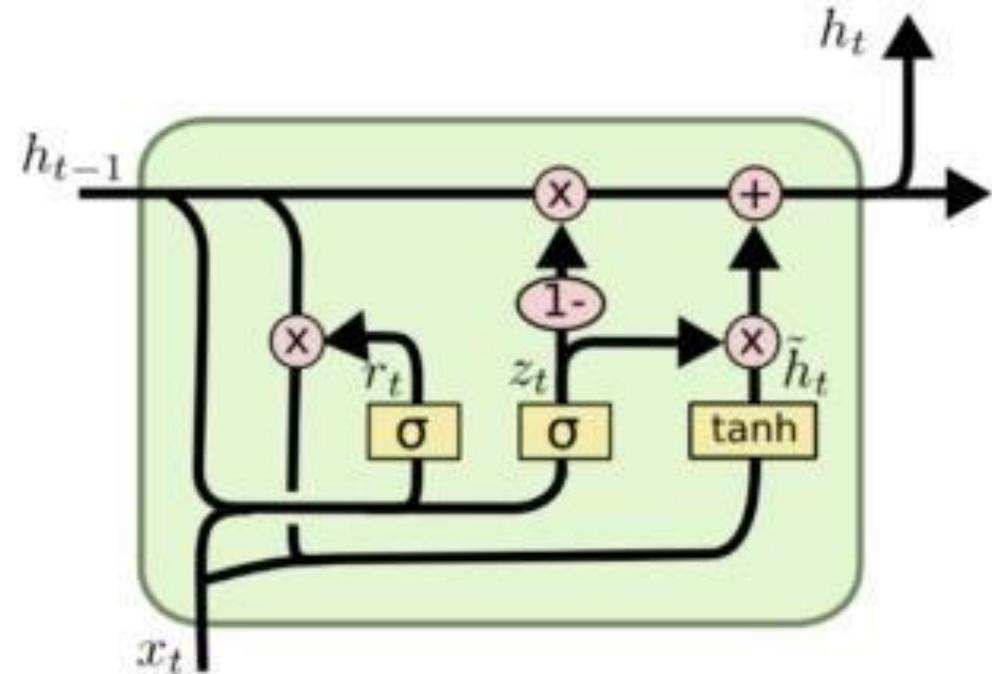
- GRU(Gated Recurrent Unit)

- LSTM과 유사한 long-term dependencies를 모델링할 수 있는 신경망

- LSTM 보다 좀 더 간략한 구조, 성능은 유사
- 2개의 gate만을 사용하기 때문에 더 적은 parameter를 가짐

- 2개의 게이트 사용

- reset-gate
 - 과거의 정보를 얼마나 지울지를 결정
- update-gate
 - 이전 값을 얼마나 합할지를 결정



- $z^{(t)} = \sigma(W^{(z)}x^{(t)} + U^{(z)}h^{(t-1)})$ (Update gate)
- $r^{(t)} = \sigma(W^{(r)}x^{(t)} + U^{(r)}h^{(t-1)})$ (Reset gate)
- $\tilde{h}^{(t)} = \tanh(r^{(t)} \circ U h^{(t-1)} + W x^{(t)})$ (New memory)
- $h^{(t)} = (1 - z^{(t)}) \circ \tilde{h}^{(t)} + z^{(t)} \circ h^{(t-1)}$ (Hidden state)

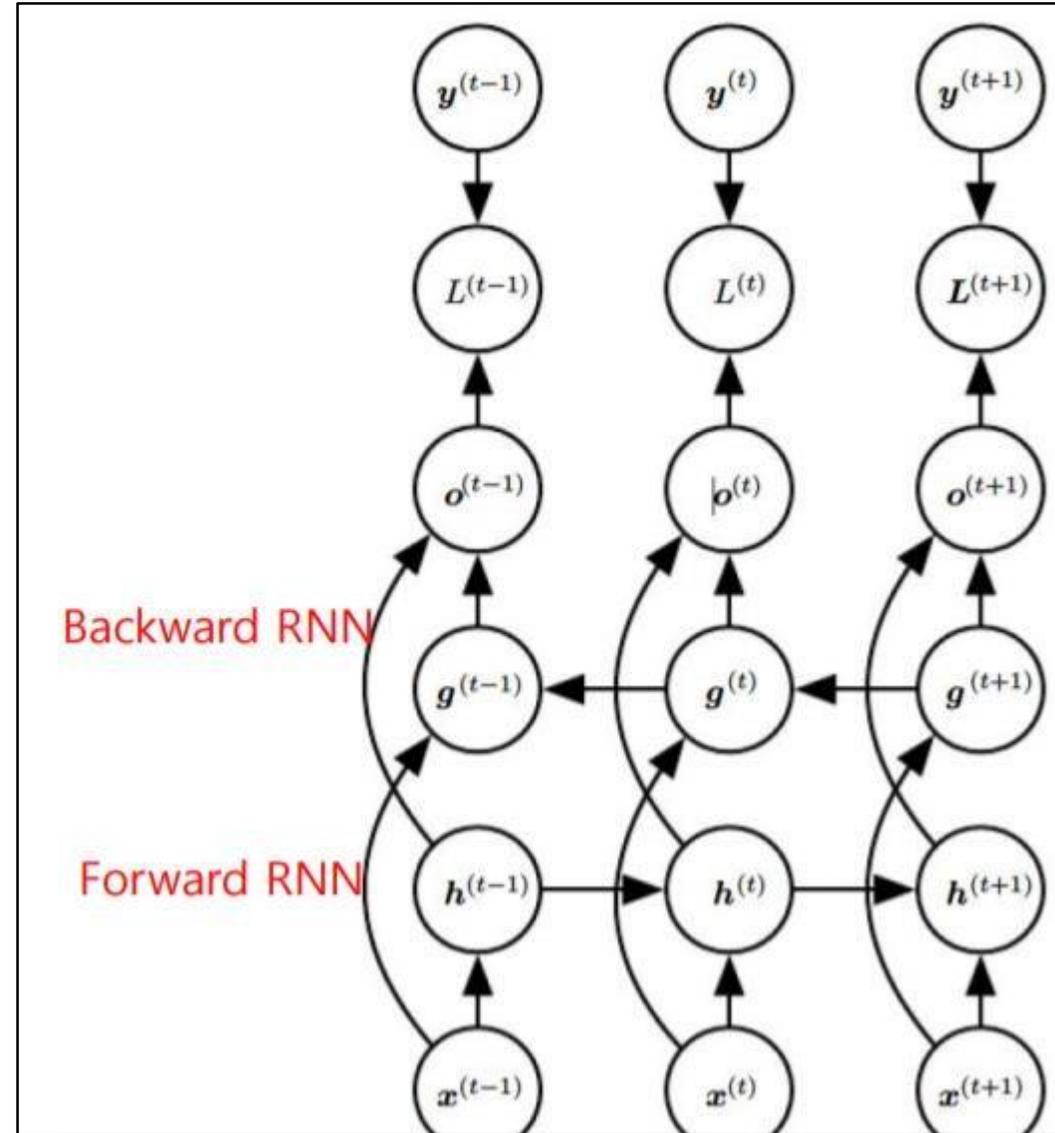
Bidirectional RNN

- 양방향의 RNN의 사용

- Forward RNN : 입력 sequence의 앞에서부터 정방향으로 이동
- Backward RNN : 입력 sequence의 역순인 뒤에서부터 역방향으로 이동
- 출력 층에서 양방향 RNN의 결과를 결합

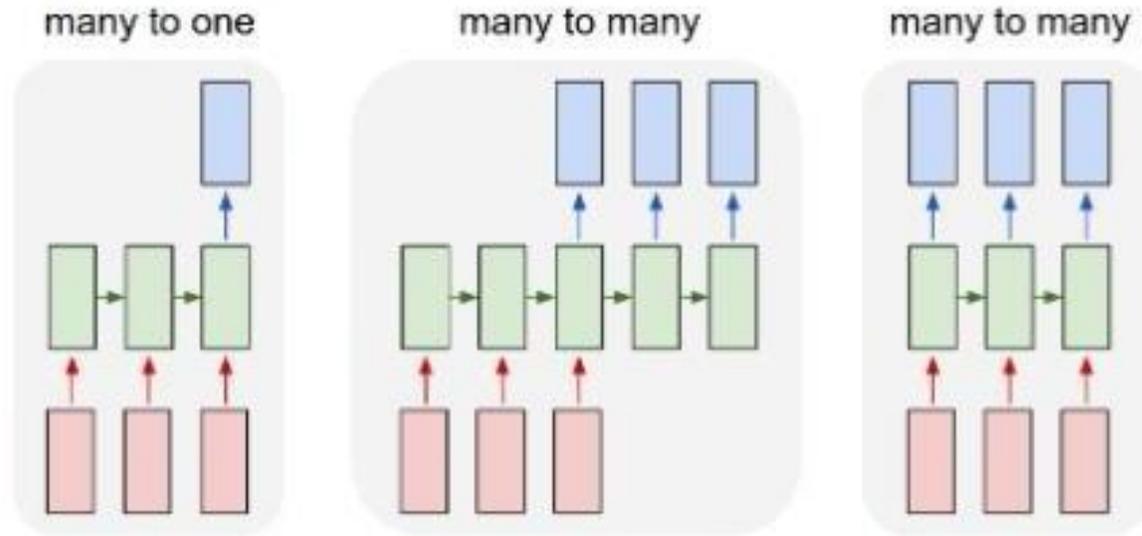
- 양방향 LSTM, GRU 등 기본적으로 적용 가능

- Pytorch, Tensorflow 등 딥러닝 라이브러리에서 기본적인 인자로 제공



RNN 계열 모델의 응용

- 다양한 자연어 처리 응용 분야에 적용 가능



– Many to One

- 감성 분석 : RNN 출력의 마지막 은닉 상태는 문장의 모든 정보를 담고 있어 위 정보를 이용해 문장 분류를 수행

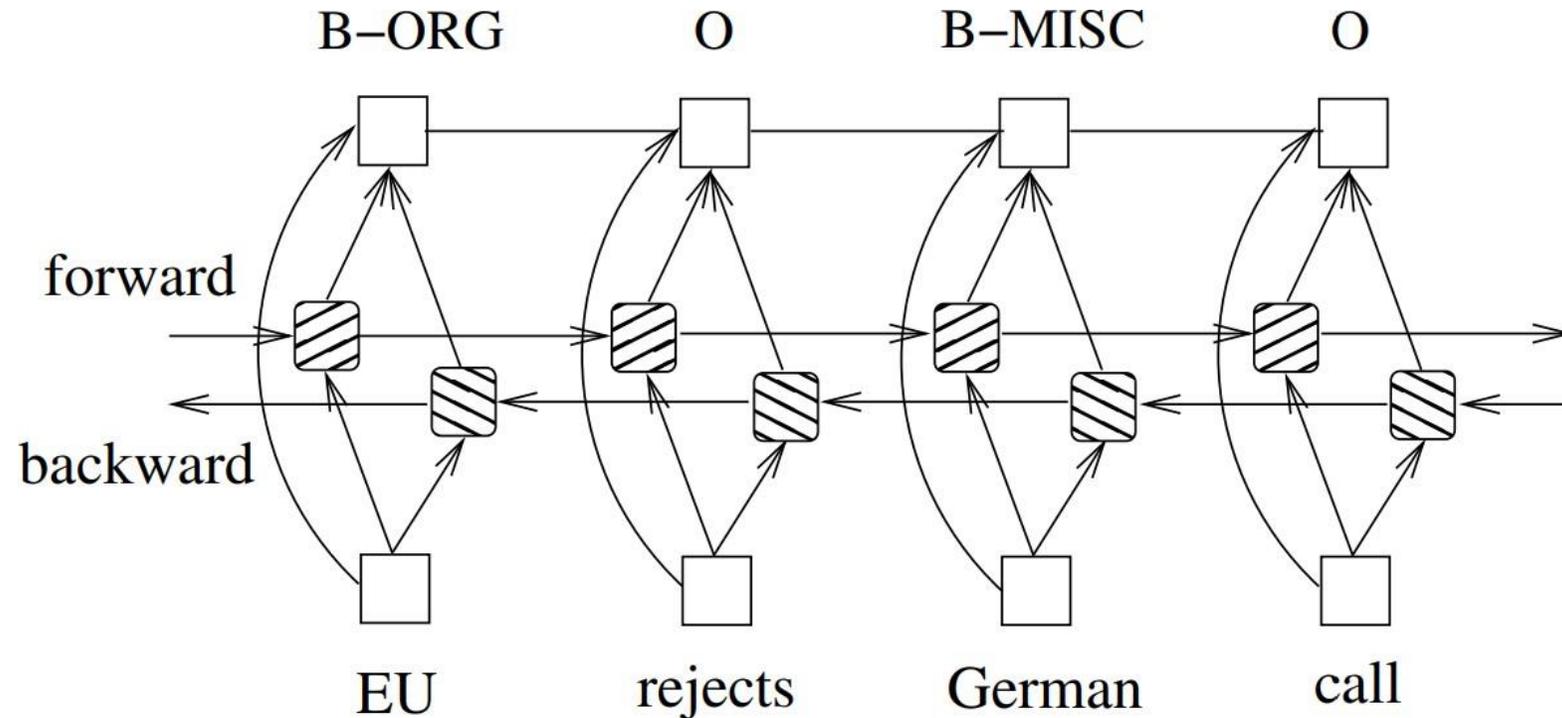
– Many to Many

- 기계 번역 : [2가지 RNN 모델 사용] 인코더는 입력 문장을 인코딩하고 디코더에서는 인코더와 입력 단어를 이용하여 출력 단어를 생성
- 개체명 인식 : 문장의 각 토큰에 대한 은닉 상태를 이용하여 각 토큰에 대한 (PER, ORG, LOC, O) 등의 레이블을 부착

Bi-LSTM CRF

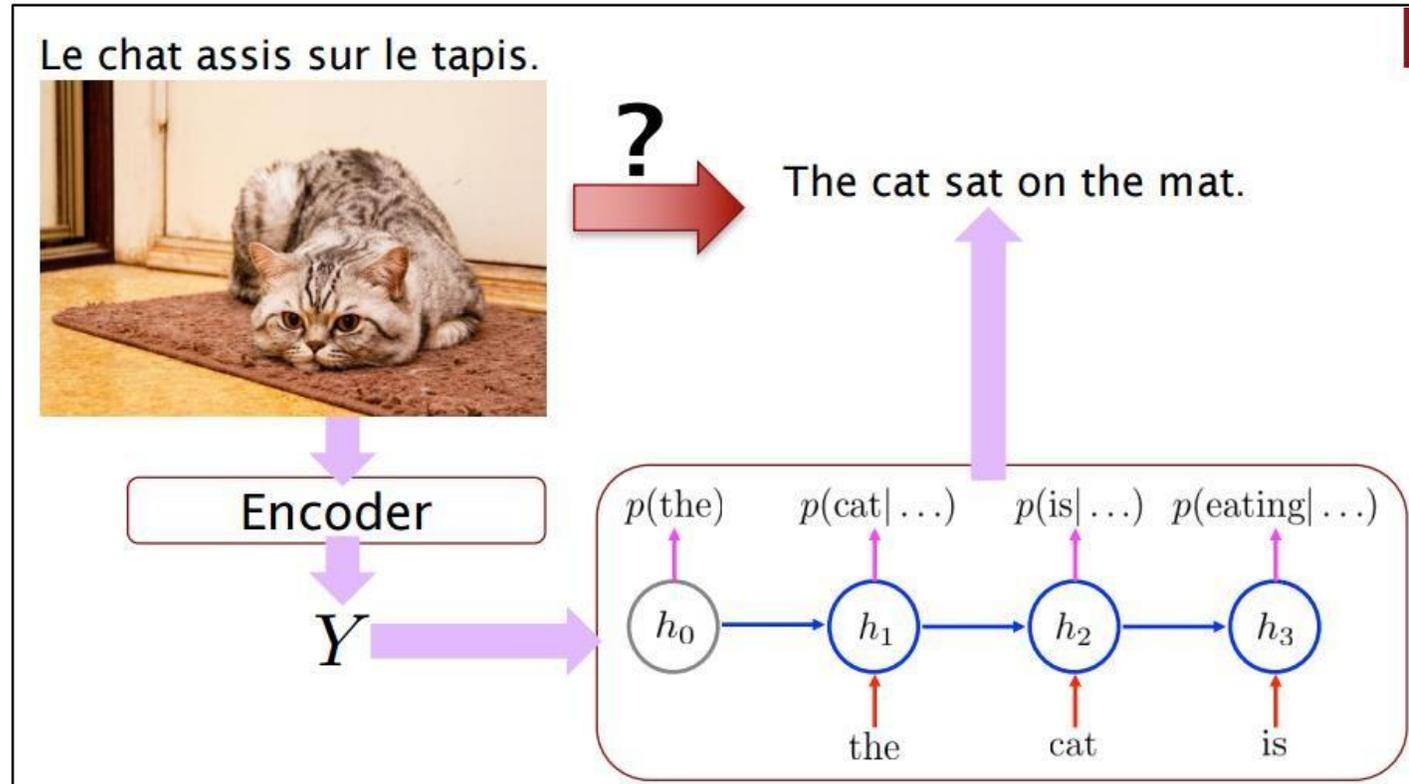
- Bidirectional LSTM-CRF Models for Sequence Tagging

- 양방향으로 순환하는 LSTM을 통해 문장을 인코딩 한 후 출력 층에서 출력 태그간의 의존성을 모델링하는 CRF와 결합
- 순차 태깅 문제에 대해 높은 성능을 보여줌



Neural Encoder-Decoder

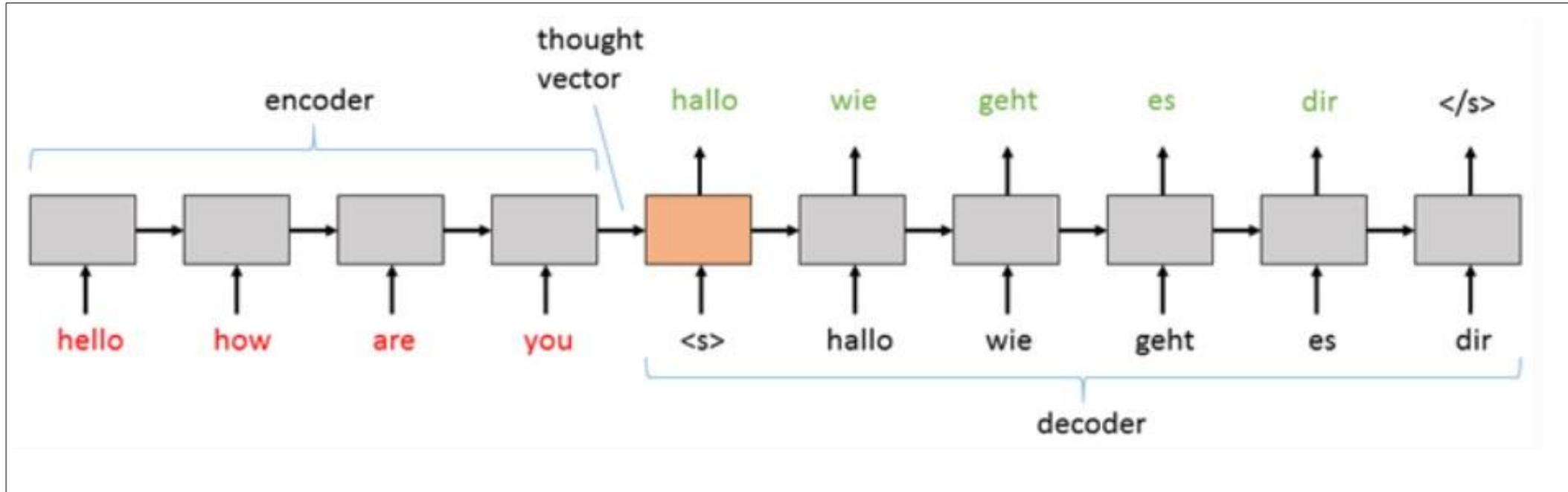
- Neural Encoder-Decoder



- 시퀀스를 출력 시퀀스로 변환하는 모델로 번역기에서 대표적으로 사용되는 모델
- Sequence-to-Sequence(seq2seq)모델로도 부름
 - 한 종류의 Sequence를 다른 한 종류의 Sequence로 변환
 - 두 시퀀스의 길이는 서로 다를 수 있음

Neural Encoder-Decoder

- Neural Encoder-Decoder

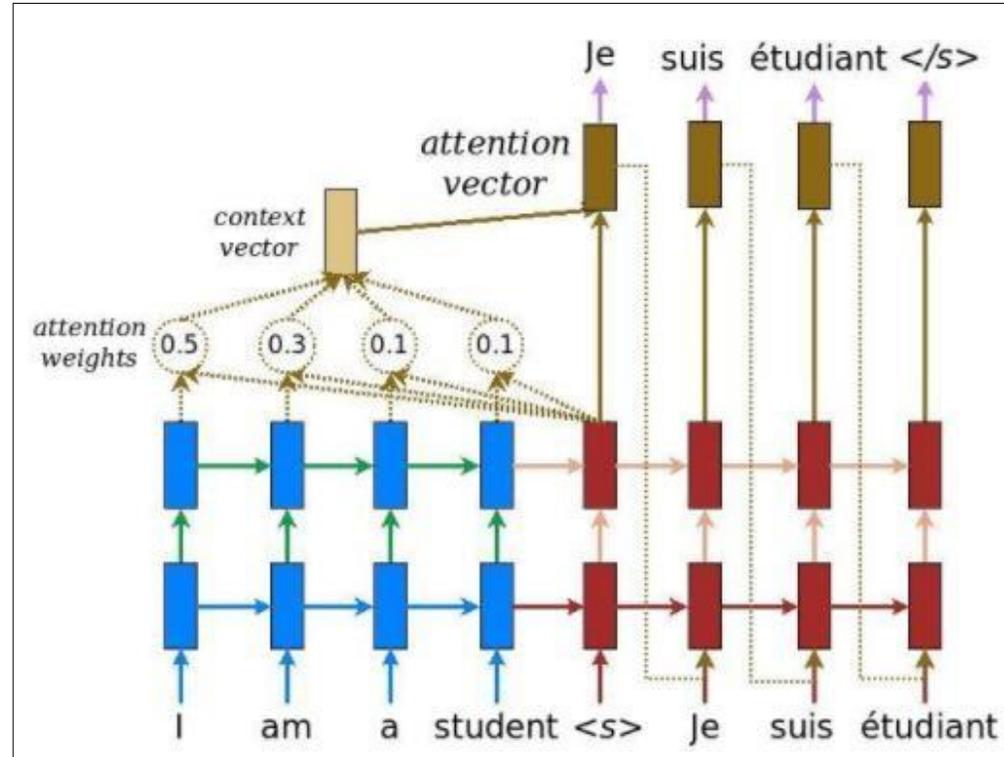


- 인코더(Encoder)-디코더(Decoder)의 두 파트로 구성

- 인코더 : 입력 문장을 인코딩 하여 문맥 벡터를 생성하고 마지막 히든 상태를 디코더로 전달
- 인코더의 마지막 hidden state(은닉상태)가 디코더의 초기 hidden state
- 디코더 : 디코더의 초기 입력은 <s>(START 심볼)로 다음 입력은 이전에 생성된 단어로 </s>(END 심볼)이 생성될 때까지 시퀀스를 생성

Attention Encoder-Decoder

- 어텐션 인코더 디코더(Attention Encoder-Decoder)



- 어텐션 메커니즘

- 기계 번역에서 입력 문장이 길어지면 번역 성능이 떨어지는 것을 해결하기 위해 고안
- 디코더의 매 타입 스텝 t 마다 입력 문장을 참고하여 해당 시점에 예측해야 할 단어와 연관이 있는 단어 부분을 좀 더 집중

Attention Encoder-Decoder

- 어텐션 인코더 디코더(Attention Encoder-Decoder)

<p>1. Attention score</p> $e_{ij} = a(s_{i-1}, h_j)$ <p>[스칼라]</p>	<p>3. Attention output</p> $c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$ <p>[벡터]</p>
<p>2. Attention distribution</p> $\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$ <p>[스칼라]</p>	<p>4. Decoder hidden state</p> $s_i = f(s_{i-1}, y_{i-1}, c_i)$ <p>[벡터]</p>

- 어텐션 메커니즘

- Attention Score : 어텐션 함수 a 를 통해 디코더의 각 단어에 대해 인코더의 어느 입력 단어에 집중할 것인지에 대한 점수화
- Attention distribution : 어텐션 점수에 대해 softmax를 취해 확률 값으로 정규화
- Attention distribution : 위의 어텐션 점수를 인코더 벡터에 대해 aggregation 후 최종적인 벡터를 계산 (이 벡터는 디코딩 시점마다 다름)
- Decoder hidden state : 어텐션 출력 벡터, 이전 디코더 은닉 상태, 이전 디코더 출력을 이용하여 다음 디코더 은닉 상태 출력

Attention Encoder-Decoder

- 어텐션 함수(scoring function)

- 디코더의 특정 t 시점의 벡터와 전체 인코더 벡터 사이의 점수 계산
 - 가중치 : [decoder의 길이 x 인코더의 길이]
 - Aggregation : 어텐션 가중치가 반영된 벡터를 얻는 과정
 - 가중치와 인코더 벡터의 곱
 - [decoder length x encoder length] X [encoder length x 은닉 차원] => [decoder length x 은닉 차원]
- Original scoring function

$$\text{score}(\mathbf{h}_{t-1}, \bar{\mathbf{h}}_s) = \mathbf{v}^T \tanh(\mathbf{W}\mathbf{h}_{t-1} + \mathbf{V}\bar{\mathbf{h}}_s)$$

- Extension of scoring functions

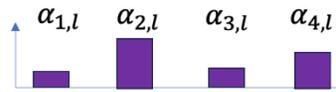
$$\text{score}(\mathbf{h}_t, \bar{\mathbf{h}}_s) = \begin{cases} \mathbf{h}_t^\top \bar{\mathbf{h}}_s & \text{dot} \\ \mathbf{h}_t^\top \mathbf{W}_a \bar{\mathbf{h}}_s & \text{general} \\ \mathbf{W}_a [\mathbf{h}_t; \bar{\mathbf{h}}_s] & \text{concat} \end{cases}$$

Bilinear function

Pointer Networks

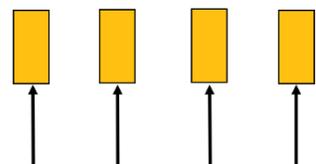
- Pointer Networks(포인터 네트워크)

Score distribution

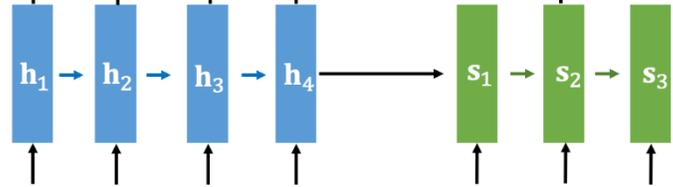


softmax

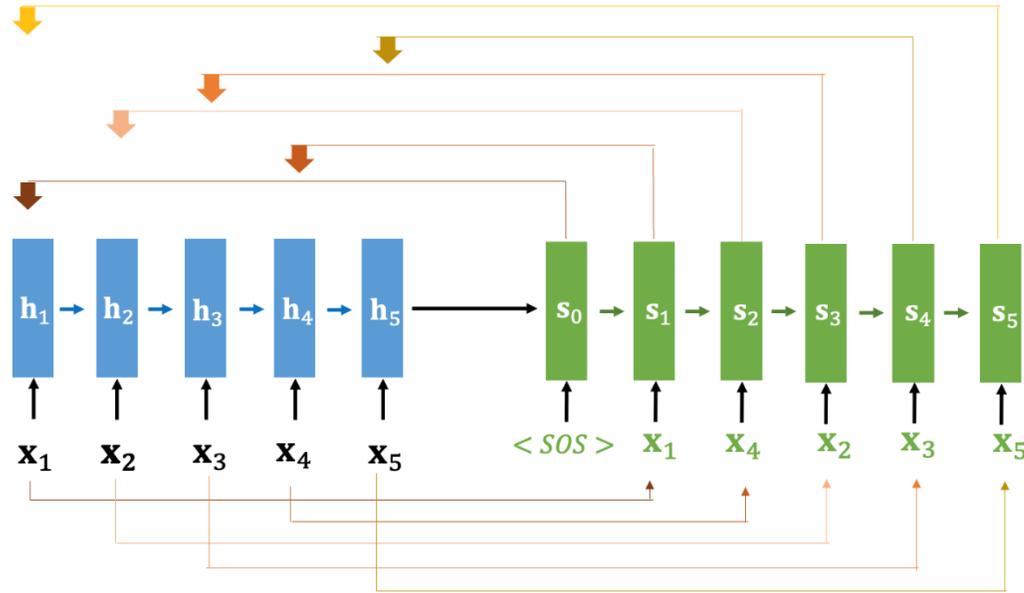
$e_{1,l}$ $e_{2,l}$ $e_{3,l}$ $e_{4,l}$



q_l



<Sequence-to-Sequence>



<Pointer Networks>

Pointer Network:

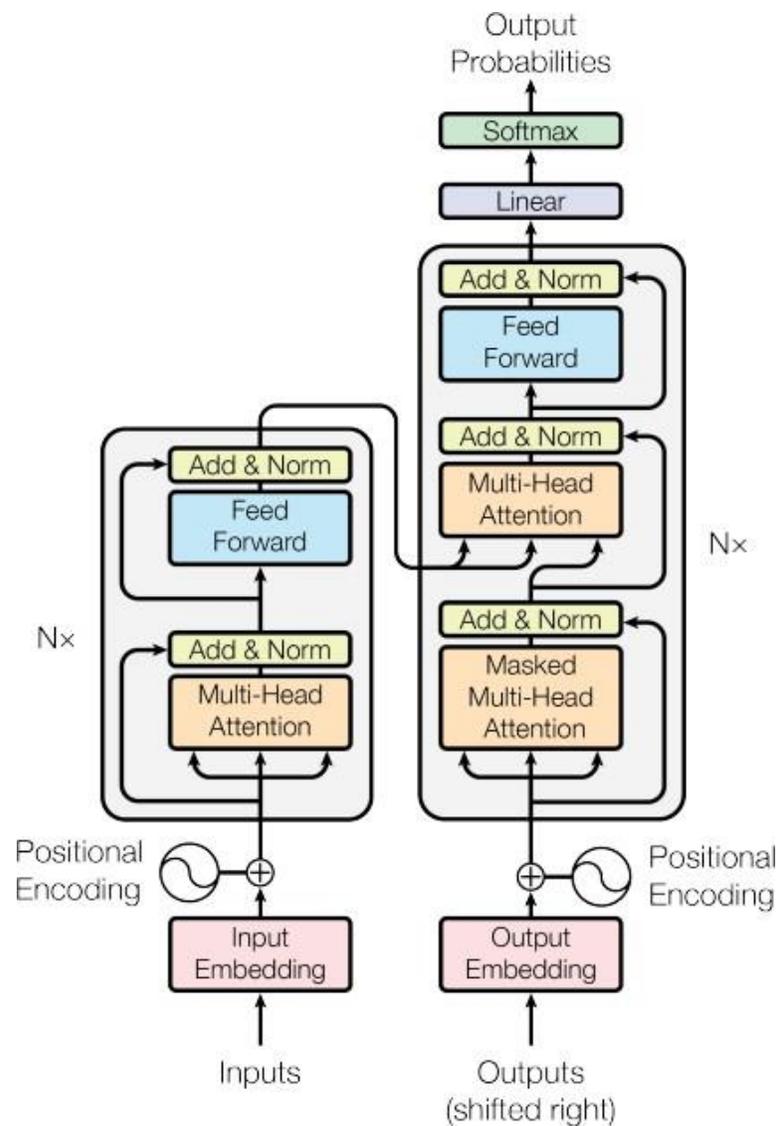
Seq2Seq 모델의 변형으로, 출력 시퀀스가 입력 시퀀스 내의 특정 위치를 직접 참조하여 출력. Attention 메커니즘을 사용, 입력 시퀀스의 특정 위치에 포인팅하여 출력 시퀀스를 생성.

차이점:

Seq2Seq는 일반적으로 새로운 텍스트, 포인터 네트워크는 입력 시퀀스 내의 위치를 참조하여 출력

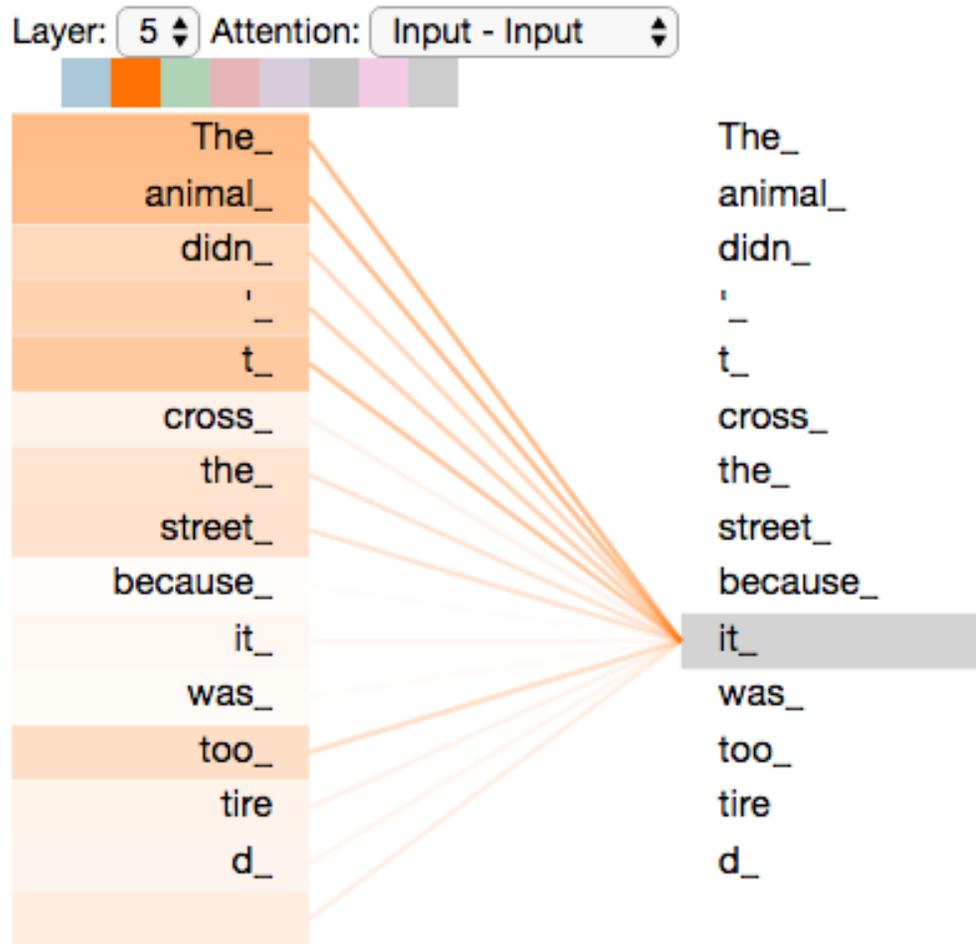
Transformers

- 순환 구조를 가지지 않는 Transformers
 - 인코더(2가지 절차 수행)
 - Self 어텐션
 - 입력 시퀀스에 대해 자기 자신과 어텐션을 수행하여 입력 단어들 간의 유사도를 가중치로 하여 반영
 - Feed Forward Neural Networks
 - 디코더(3가지 절차 수행)
 - Masked Self 어텐션
 - 인코더의 self attention과 동일하나 해당 디코딩 시점까지만 어텐션이 이루어지도록 제약
 - Cross 어텐션
 - 입력 시퀀스와의 어텐션 수행
 - Feed Forward Neural Networks



Transformers

- Self-Attention at a High Level
 - 어텐션 가중치 시각화



- 장점
 - 상대적으로 긴 시퀀스에 대해서 잘 동작
 - 순환 구조를 가지지 않기 때문에 병렬 수행(속도가 빠름)

Machine Reading Comprehension

- Machine Reading Comprehension(기계 독해)

Article: Endangered Species Act

Paragraph: “... Other legislation followed, including the Migratory Bird Conservation Act of 1929, a 1937 treaty prohibiting the hunting of right and gray whales, and the *Bald Eagle Protection Act of 1940*. These *later laws* had a low cost to society—the species were relatively rare—and little *opposition* was raised.”

Question 1: “Which laws faced significant *opposition*?”

Plausible Answer: *later laws*

Question 2: “What was the name of the 1937 *treaty*?”

Plausible Answer: *Bald Eagle Protection Act*

- 정의

- 기계가 입력으로 받은 문단을 분석하고 추론하는 것을 말하며, 기계 독해를 이용하여 주어진 질문과 문단을 이해하고 이에 알맞은 답을 찾는 것

- 딥러닝을 이용한 기계독해

- Word Embedding, RNN 인코더, Attention Mechanism, Pointer Network가 모두 사용되어 딥러닝 기반 자연어 처리를 이해하는데 도움이 될 수 있는 예시

Machine Reading Comprehension

- 딥러닝 기반 기계 독해

- Question(질문) & Passage(지문) Encoding

- 질문과 지문을 각각 embedding 한 후 LSTM등의 인코더를 통해 인코딩

- Question-Passage Matching

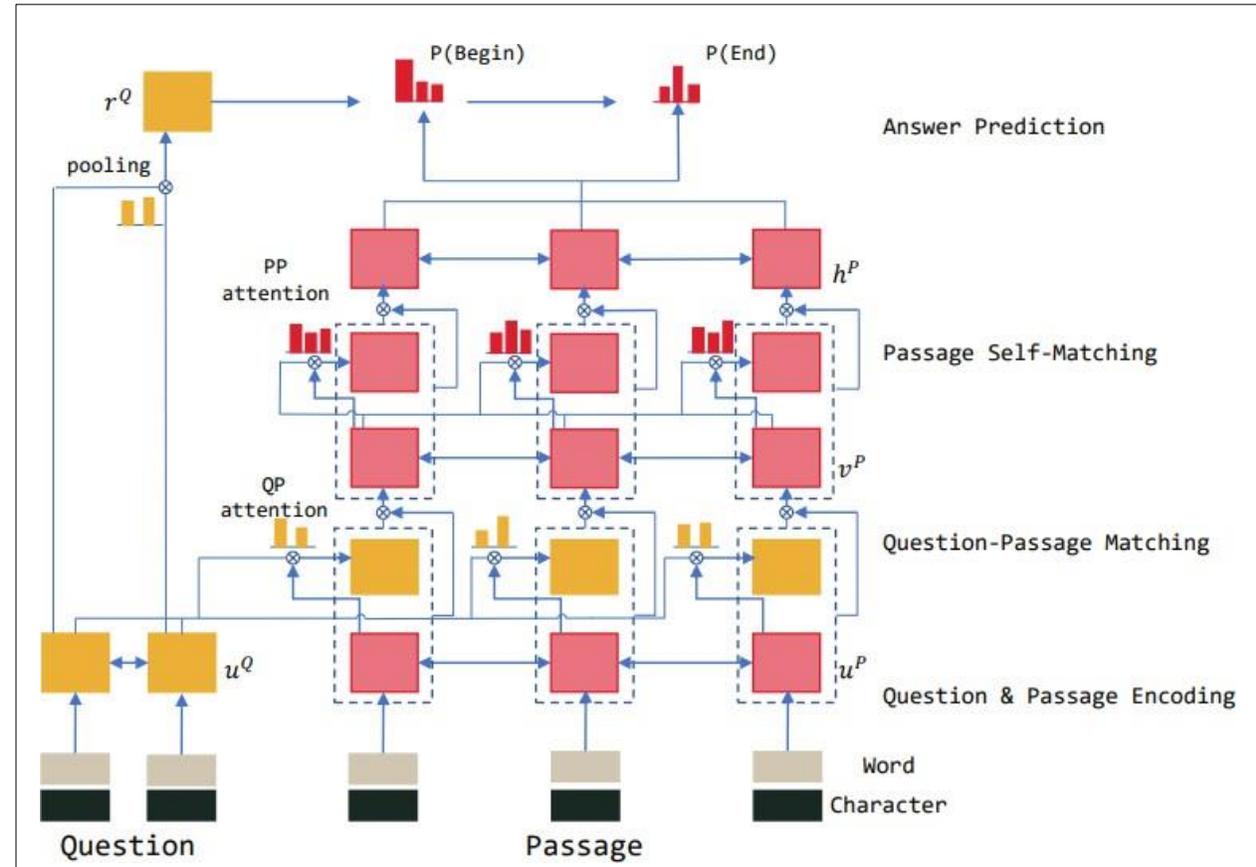
- 질문과 지문 사이에서 Attention을 수행하여 각 Passage 단어와 Question 단어 간의 가중치 계산 후 aggregation

- Passage Self-Matching

- Self Matching 어텐션을 통해 passage 단어에 대한 재인코딩 수행

- Answer Prediction

- Pointer Network를 이용하여 passage 내에서 정답의 start, end 위치를 출력



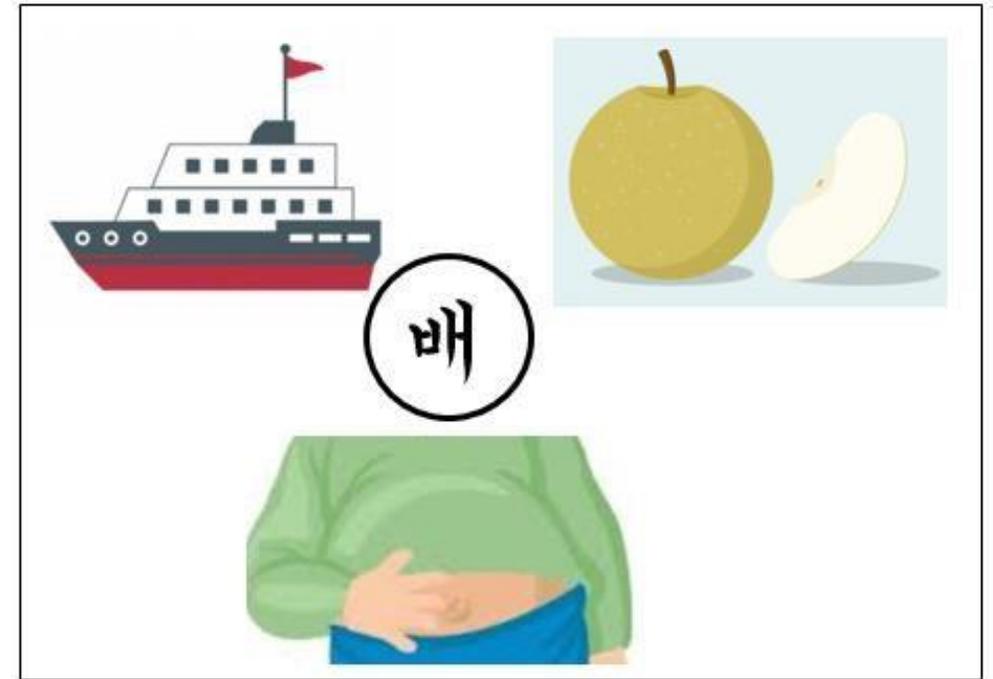
<R-Net 기계독해 모델의 구조>

문맥 단어 표현

- 기존 Word Embedding의 단점

- 중의적인 단어를 표현 할 수 없음
 - 예) 단어 “배”
- 문맥에 관계 없이 고정된 단어 표현 사용
 - 단어 이불이 언급된 다양한 문장

올해는 이불 하나 장만해야죠..
새로 산 이불 이 너무 따듯하다.
이불 밖은 위험해
집 가서 이불 킁할 듯



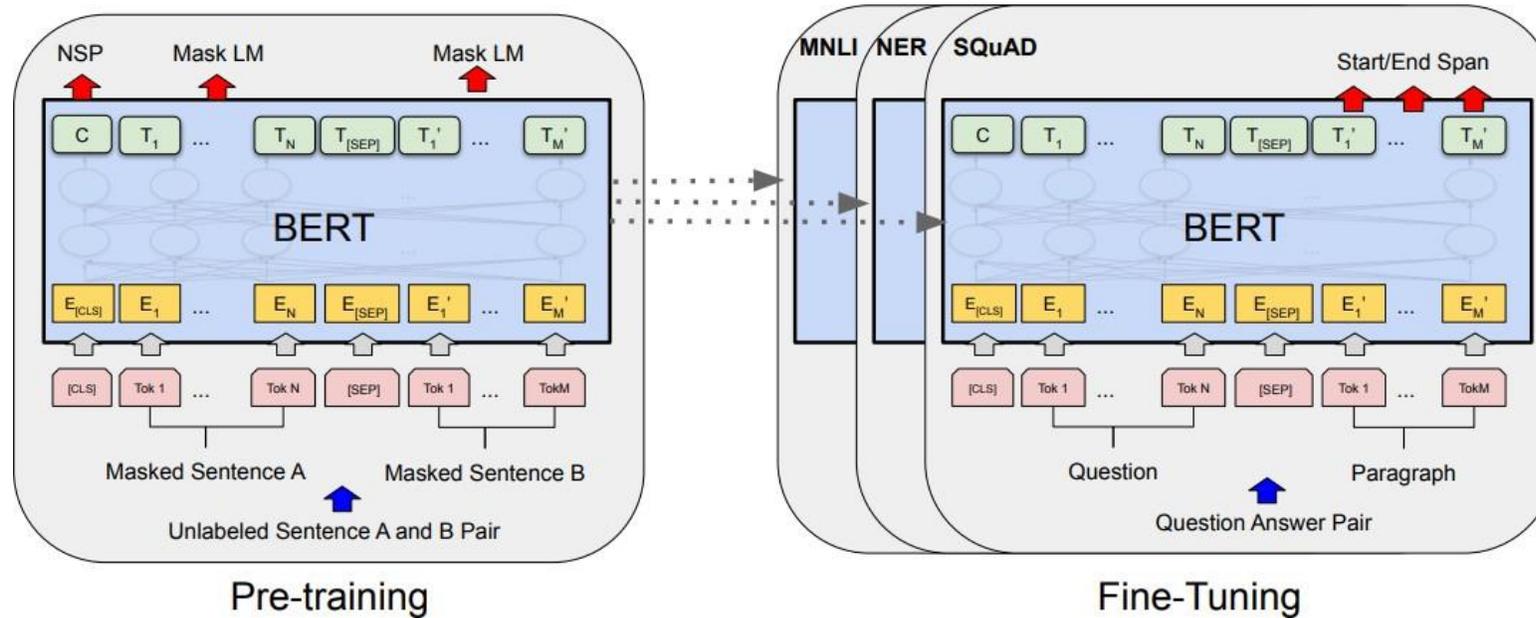
- 문맥 단어 표현(Contextual Word Representation)

- 같은 단어더라도 문맥에 따라 그 표현 방법(벡터)가 바뀔 수 있는 개념의 Word Embedding
- ELMo, BERT, GPT 계열의 다양한 모델 연구됨

관점	Word Embedding	문맥 단어 표현
Input	단어 단위	문장 단위(단어의 시퀀스)
Layer	(일반적으로) 단층	(일반적으로) 다계층
Output	해당 단어에 대한 Embedding	문장을 구성하는 각 단어에 대한 Embedding들

BERT

- BERT(Pre-training of Deep Bidirectional Transformers for Language Understanding)



- 문맥 단어 표현 학습

- 양방향 Transformer를 이용하여 문장 내 임의의 단어를 예측하고 다음 문장을 예측하는 두 가지 task로 언어 모델 학습
- 동일한 단어에 대해서도 문맥에 따라 다른 의미를 표현

- 응용 Task에 Fine-Tuning하는 방식으로 성능 향상

BERT

- BERT(Pre-training of Deep Bidirectional Transformers for Language Understanding)

- 성능 향상
 - 기계 독해

System	Dev		Test	
	EM	F1	EM	F1
Top Leaderboard Systems (Dec 10th, 2018)				
Human	-	-	82.3	91.2
#1 Ensemble - nlnet	-	-	86.0	91.7
#2 Ensemble - QANet	-	-	84.5	90.5
Published				
BiDAF+ELMo (Single)	-	85.6	-	85.8
R.M. Reader (Ensemble)	81.2	87.9	82.3	88.5
Ours				
BERT _{BASE} (Single)	80.8	88.5	-	-
BERT _{LARGE} (Single)	84.1	90.9	-	-
BERT _{LARGE} (Ensemble)	85.8	91.8	-	-
BERT _{LARGE} (Sgl.+TriviaQA)	84.2	91.1	85.1	91.8
BERT _{LARGE} (Ens.+TriviaQA)	86.2	92.2	87.4	93.2

- 이외의 많은 자연어 처리 응용 분야에서 놀라운 성능 향상을 가져옴

Auto Encoding (AE) vs Autoregressive (AR)

Pre-training 단계에서의 대표적인 objective

- Auto Encoder (AE) and Denoising Auto Encoder(DAE)

- AE는 주어진 input에 대해 그 input을 그대로 예측하는 문제를 품

- DAE는 noise가 섞인 input을 원래의 input으로 예측하는 문제를 품

- BERT같은 경우

- 주어진 input sequence에 임의로 추가한 noise([MASK] token)가 주어졌을 때, [MASK] token 을 원래 input token으로 복구. Denoising Auto Encoder의 방식

$$\text{input sequence} : \bar{x} = (x_1, x_2, \dots, x_T)$$

$$\text{corrupted input} : \hat{x} = (x_1, [MASK], \dots, x_T)$$

$$\text{likelihood} : p(\bar{x} | \hat{x}) \approx \prod_{t=1}^T p(x_t | \hat{x})$$

$$\text{training objective} : \max_{\theta} \log p(\bar{x} | \hat{x}) = \max_{\theta} \sum_{t=1}^T m_t \log p(x_t | \hat{x})$$

Auto Encoding (AE) vs Autoregressive (AR)

- Autoregressive (AR) => 자가 회귀 모델

- 이전 token들을 보고 다음 token을 예측하는 문제

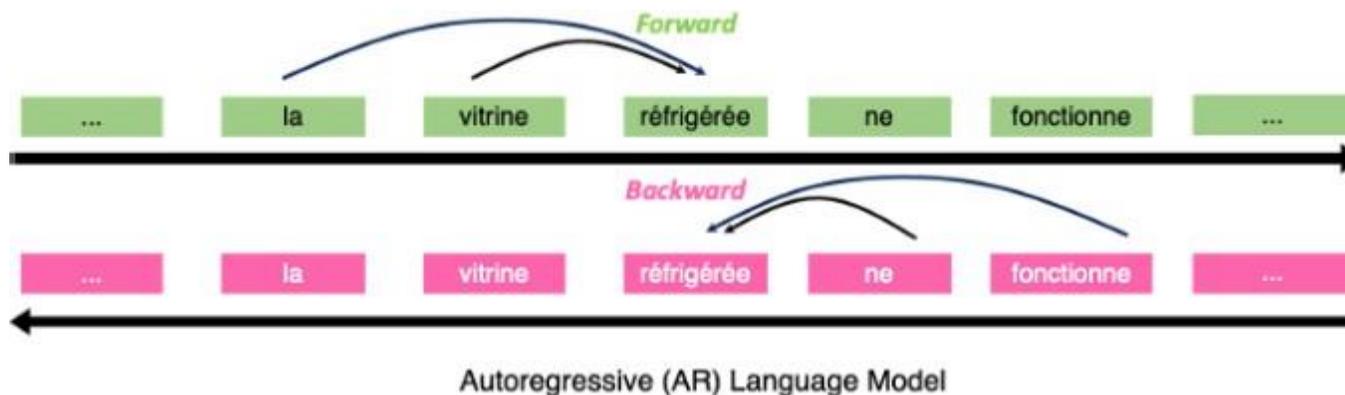
$$\text{input sequence} : x = (x_1, x_2, \dots, x_T)$$

$$\text{forward likelihood} : p(x) = \prod_{t=1}^T p(x_t | x_{<t})$$

$$\text{training objective (forward)} : \max_{\theta} \log p_{\theta}(x) = \max_{\theta} \sum_{t=1}^T \log p(x_t | x_{<t})$$

- 단점

- AR은 방향성(forward, backward)이 정해져야 하므로, 한쪽 방향의 정보만을 이용 가능.
- 양방향 문맥 활용보다 문장에 대해 깊이 이해하기 어려우며, 얕은 이해만 가능

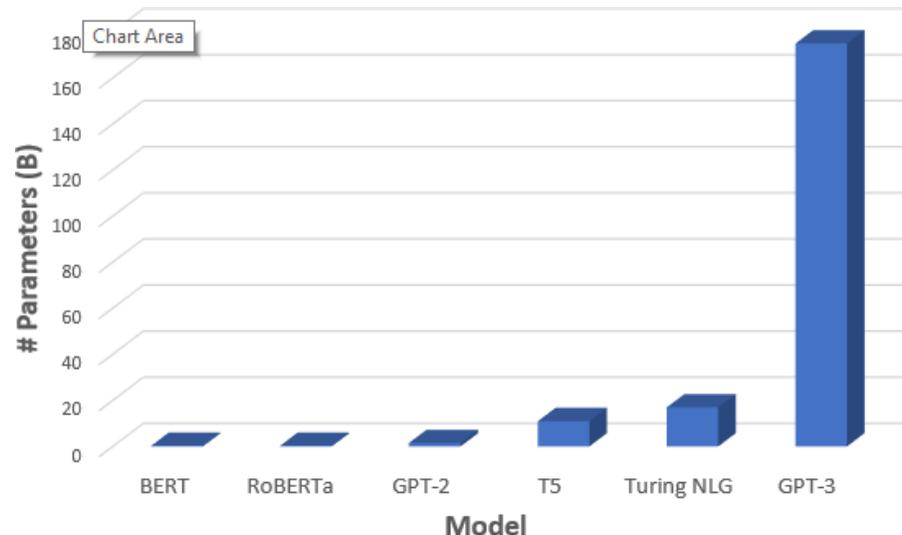


GPT Series

- GPT-3 - Language Models are Few-Shot Learners

- Autoregressive 방식의 언어 모델

- 인류 역사상 가장 뛰어난 '언어 인공지능' 이자 few-shot learning에 최적화된 인공지능
 - few shot learning - 풀고자 하는 문제에 대해 몇 개의 예시만 보고 태스크에 적응하여 문제를 푸는 것
- 96층의 레이어, 12,288차원의 히든 차원, 96개 attention head를 가지는 총 1750억 개의 파라미터의 모델. 3,000억 토큰의 학습 데이터에 대해 학습함

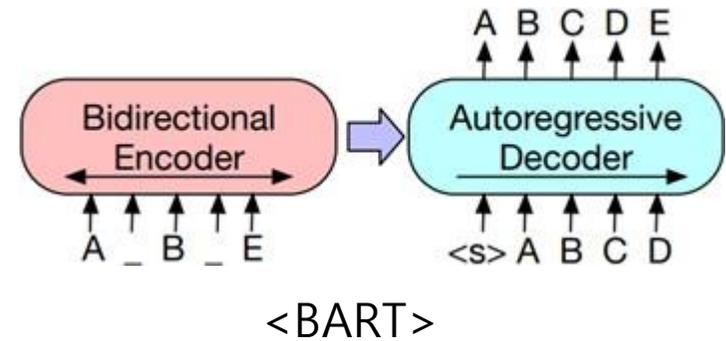
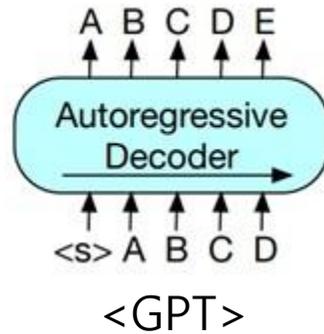
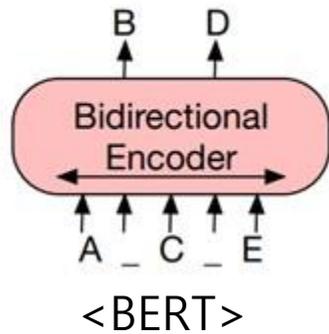


BART

- 자연어 생성을 위한 언어모델

- Sequence-to-Sequence 구조

- BART는 손상된 text를 입력받아 bidirectional 모델로 인코딩. 정답 text에 대한 likelihood를 autoregressive 디코더로 계산



- 생성 task에서 fine-tuning 성능 향상

	CNN/DailyMail			XSum		
	R1	R2	RL	R1	R2	RL
Lead-3	40.42	17.62	36.67	16.30	1.60	11.95
PTGEN (See et al., 2017)	36.44	15.66	33.42	29.70	9.21	23.24
PTGEN+COV (See et al., 2017)	39.53	17.28	36.38	28.10	8.02	21.72
UniLM	43.33	20.21	40.51	-	-	-
BERTSUMABS (Liu & Lapata, 2019)	41.72	19.39	38.76	38.76	16.33	31.15
BERTSUMEXTABS (Liu & Lapata, 2019)	42.13	19.60	39.18	38.81	16.50	31.27
BART	44.16	21.28	40.90	45.14	22.27	37.25

한국어 자연어 처리 소개

형태소 분석

- 정의(두 가지의 과정으로 구분)

- 형태소 분석 : 문장 내의 어절을 뜻을 지니는 최소의 단위인 형태소로 분해하고 해당 형태소의 품사 후보를 생성
- 품사 태깅 : 형태소의 품사 후보로부터 가장 적절한 품사를 결정하는 과정
- 입력문
 - 예) 거리는 사람의 물결로 넘쳤다

거리는 → 거리 [NNG] 는 [JX]

사람의 → 사람 [NNG] 의 [JKG]

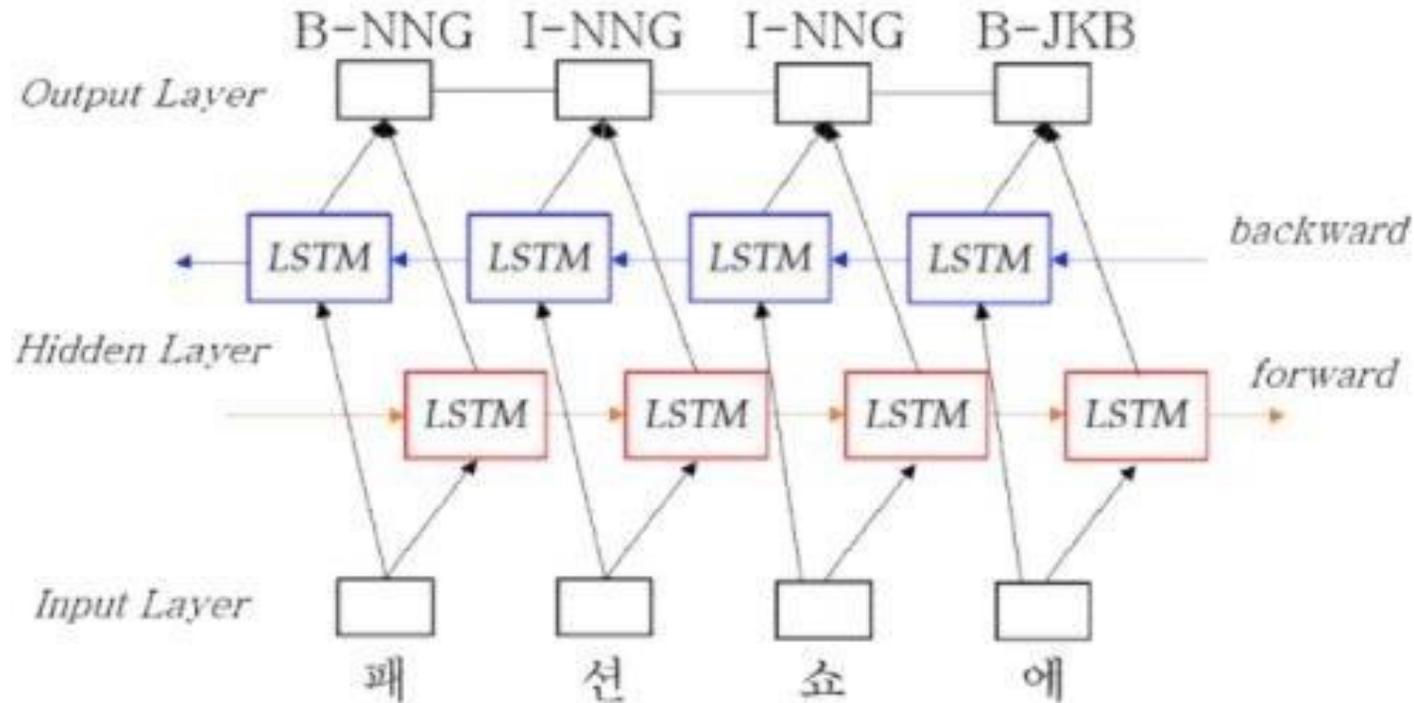
물결로 → 물결 [NNG] 로 [JKB]

넘쳤다. → 넘쳤 [VV~EP] 다 [EF] . [SF]

형태소 분석

- Bi LSTM CRFs 형태소 분석

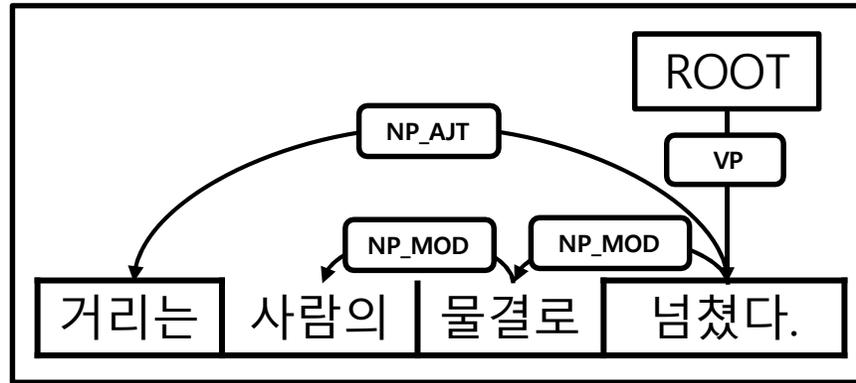
- 음절 단위의 품사 태깅 방법
- 순차 데이터를 모델링하는 양방향 LSTM에 출력 태그 간의 전이 확률을 얻는 CRF와 결합하는 방식
- [B(Begin),I(Inside)] 등의 태그 등을 붙인 품사 태그를 결정하는 방식



의존 파싱(구문 분석)

- 정의

- 문장의 구조를 결정하는 구문 분석(파싱)의 한 갈래로 단어의 Head(지배소)와 Modifier(의존소)의 관계에 따라 문장의 구조를 결정



- Paradigm

<전이 기반 방식>

버퍼와 스택으로부터 자질을 추출한 후 모델을 통해 다음 전이 액션을 결정하고 결정된 액션에 따라 버퍼와 스택의 상태를 갱신하면서 트리를 생성해 나가는 방식

지역적 탐색 모델

<그래프 기반 방식>

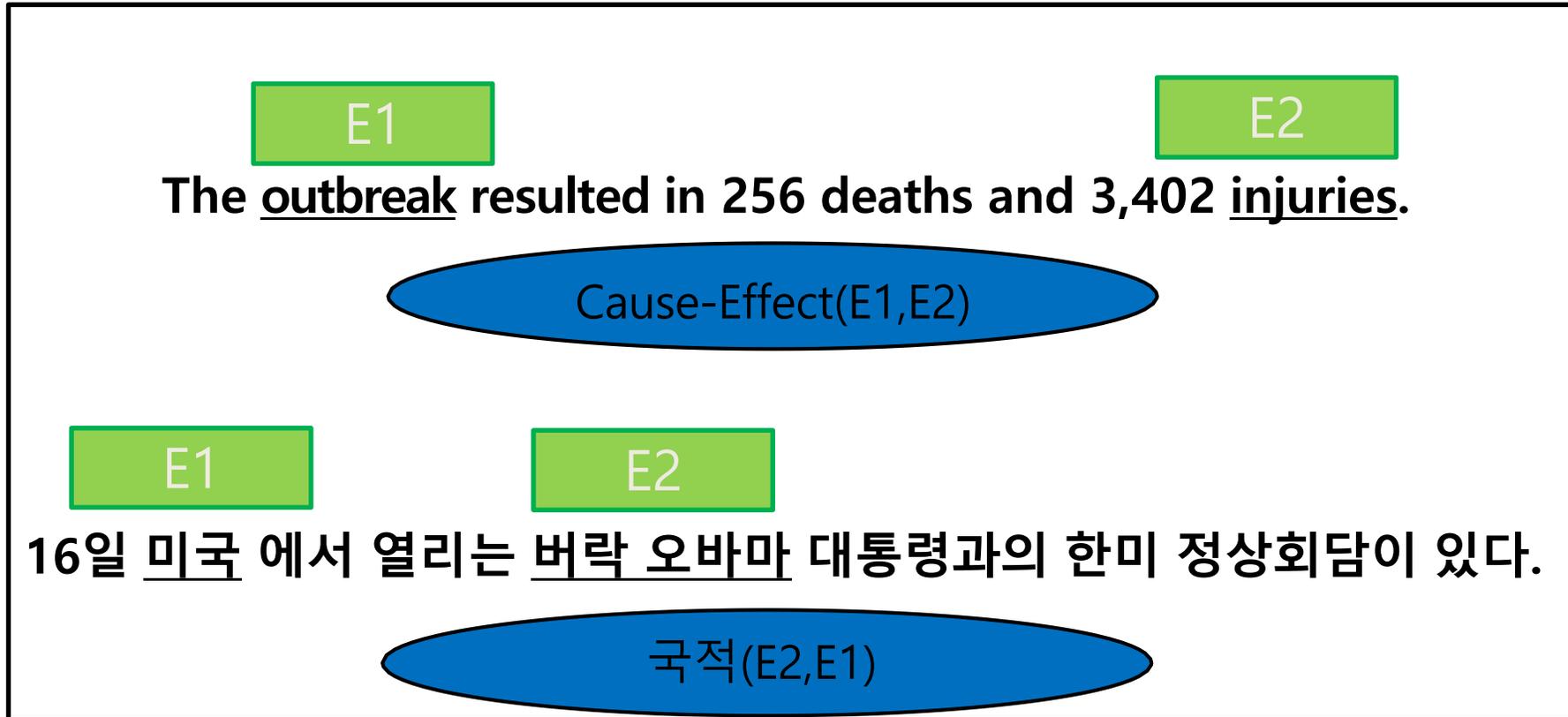
문장 내의 가능한 모든 단어 쌍의 의존관계를 찾아서 점수화하여 가장 높은 점수를 갖는 의존 트리를 찾는 방식

전역적 탐색 모델

관계 분류

- 정의

- 문장 내에서 두 개체(entity)가 주어질 때, 두 명사 사이의 관계를 분류하는 것



상호 참조 해결

- 정의
 - 문서 내에서 선행하는 명사구와 현재 등장한 명사구 간의 같은 개체를 의미 하는지를 결정하는 문제
 - 문맥을 이해하는데 중요한 역할을 하며 기계 번역, 질의 응답 문서 요약 시스템 등에 응용

"I voted for **Nader** because **he** was most aligned with **my** values," **she** said.

개체명 연결

- 개체명

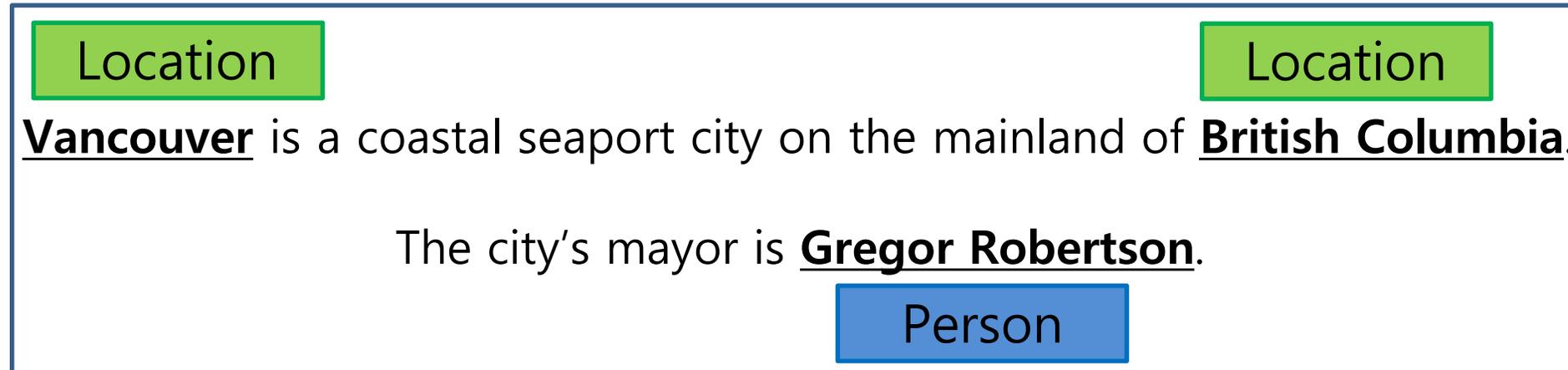
- 문서에서 나타나는 고유한 의미를 가지는 단어 또는 구절을 의미
- 문서에서 나타난 개체 표현을 개체 맨션이라고 부름
 - 개체 표현과 실제 엔티티는 다를 수 있음
 - “트럼프”, “도널드 존 트럼프”, “도널드 트럼프” 등 다양한 형태로 문서에서 나타날 수 있음

Vancouver is a coastal seaport city on the mainland of British Columbia.
The city's mayor is Gregor Robertson.

개미 사회에 대한 묘사가 참 뛰어나서 진실이라고 믿게 한 이 작품 『개미』
을 저는 베르나르 베르베르의 가장 뛰어난 작품으로 여기고 있습니다.

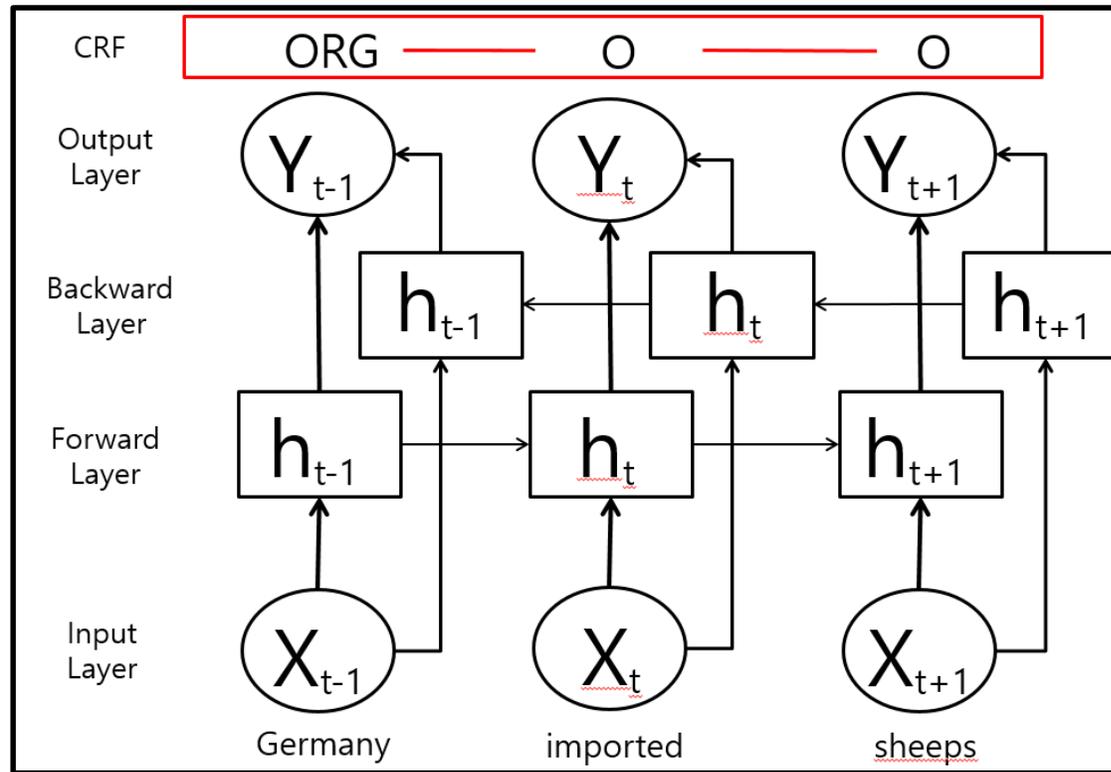
- 개체명 인식

- 개체명 인식은 문서로부터 개체명을 추출하고 추출된 개체명의 종류를 분류하는 자연언어처리의 한 분야



- 문서 내에서 개체 맨션을 찾은 후 적절한 타입을 부여
- 주로 Bi-LSTM-CRF 기반의 연구가 주를 이루었음

- Bi-LSTM-CRF 기반의 개체명 인식
 - 주어진 sequence에 대해서 label을 부여하는 sequence labeling(순차 태깅) 방식
 - 위 예제는 멀티 토큰에 대한 적절한 예가 아님



- Bi-LSTM-CRF 기반의 개체명 인식

- 한국어 예제)

- 입력은 “형태소”와 “태그”의 임베딩을 한 후 LSTM을 거친 후 [B,I]가 포함된 태그를 labeling.

Morpheme	Morpheme Tag	NE B/I/O Tag
박지성	NNP	PER_B
은	JX	O
맨체스터	NNP	ORG_B
유나이티드	NNG	ORG_I
FC	SL	ORG_I
소속	NNG	O
축구	NNG	CVL_B
선수	NNG	CVL_I
이	VCP	O
다	EF	O
.	SF	O

- 개체명 연결(Entity Linking)

- 주어진 문장에 출현한 엔티티를 위키피디아와 같은 지식 베이스(Knowledge base) 상의 하나의 개체와 연결하여 특정 개체가 무엇인지 식별하는 작업
- 보통 2 단계로 이루어짐
 - Mention Detection(멘션 추출): 문서 내에 나타나는 개체를 인식하는 과정
 - Disambiguation(중의성 해결): 인식된 개체 표현에 대해 후보 개체 중에서 중의성을 해결하여 특정 개체로 연결하는 과정

개미 사회에 대한 묘사가 참 뛰어나서 진실이라고 믿게 한 이 작품 『개미』
을 저는 베르나르 베르베르의 가장 뛰어난 작품으로 여기고 있습니다.



- 개체명 연결(Entity Linking)
 - 소설 개미 page

개미 (소설)

위키백과, 우리 모두의 백과사전.

《**개미**》(프랑스어: Les Fourmis)는 프랑스의 작가 베르나르 베르베르의 등단작이자 가장 유명한 작품이다.[[]^{*} PC를 이용해 120번에 가까운 개작을 해서 1991년 탈고했다.

목차 [숨기기]

- 한국에서 출판되며 달라진 점들
- 등장인물
- 소품
- 여담
- 같이 보기

- 개미 동음이의어 page 목록

개미 (동음이의)

위키백과, 우리 모두의 백과사전.

개미에는 다음과 같은 뜻이 있다.

- 개미**는 곤충의 하나이다.
- 개미**는 베르나르 베르베르의 소설이다.
- 개미**는 애니메이션 영화이다.
- 개미**는 대한민국의 작곡가이다.
- 개미**는 투자업계에서 **개인투자자**를 일컫는 속어이다.

- 곤충 개미 page

개미

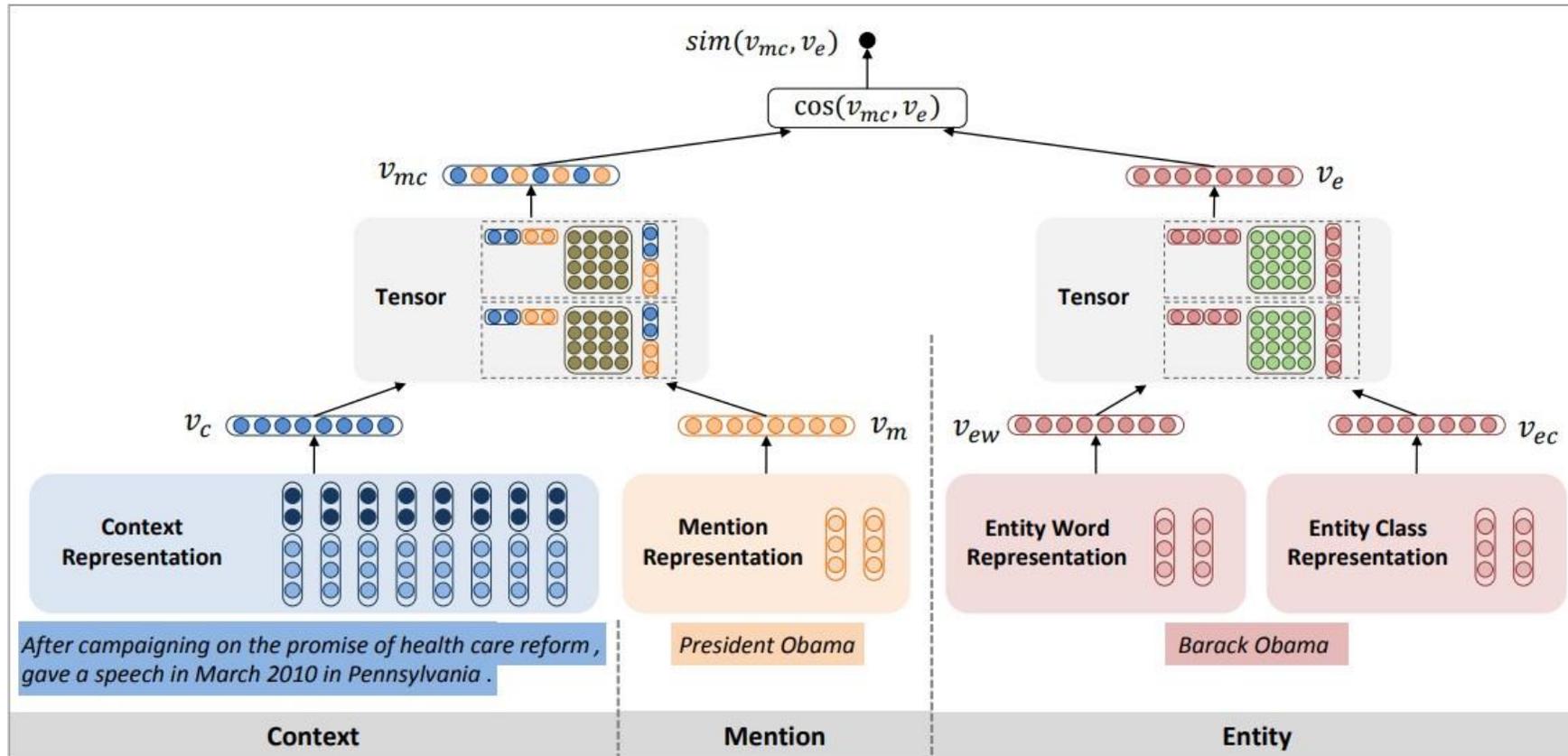
위키백과, 우리 모두의 백과사전.

 ↻ 다른 뜻에 대해서는 **개미 (동음이의)** 문서를 참조하십시오.

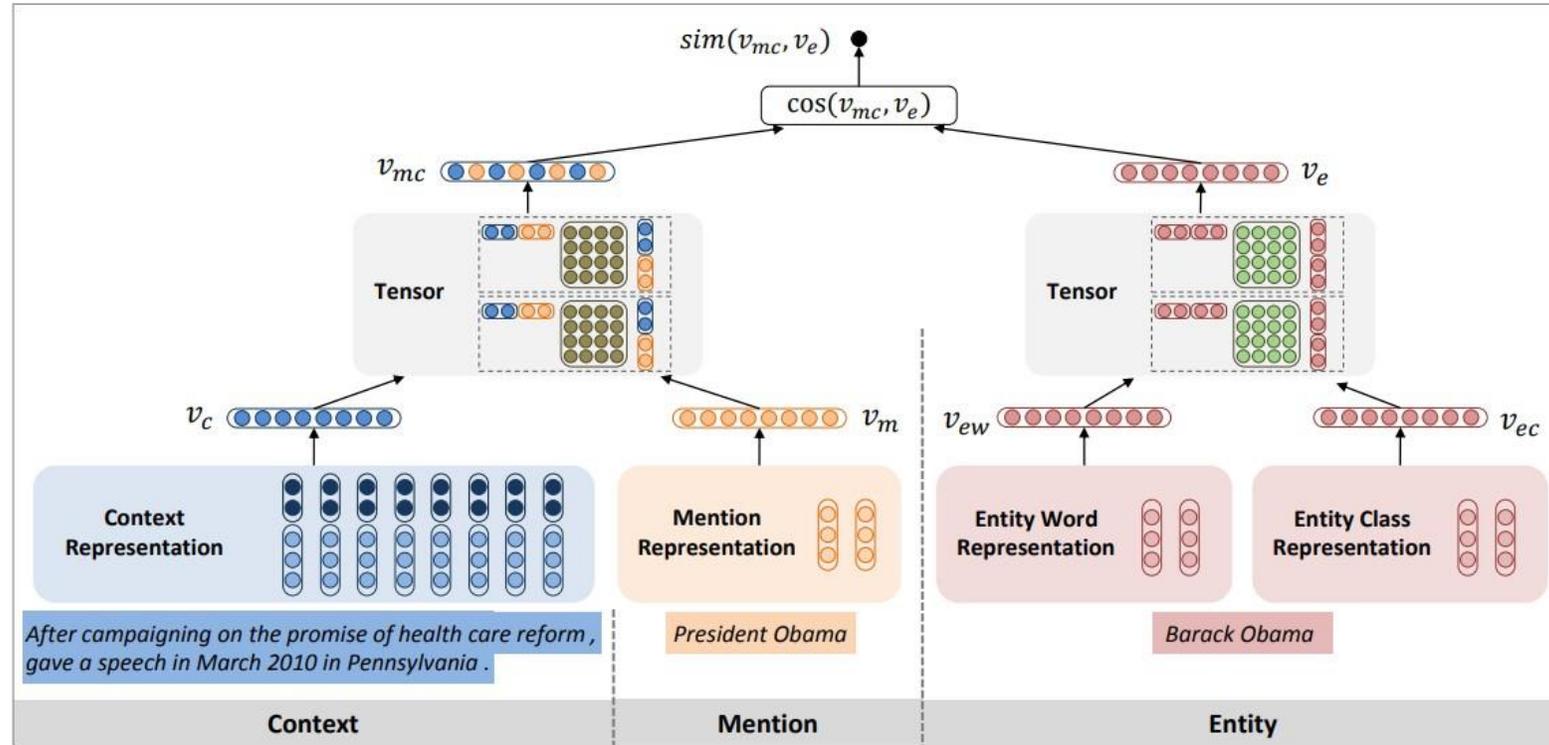
개미(ant)는 벌목 **개미과**(표준어: 개밧과)에 속하는 진사회성 곤충(eusocial insect)의 총칭으로, 말벌, 벌과 더불어 벌목에 속한다. 속씨식물이 등장한 이후 분화하였다. 오늘날에는 12,000~14,000 여 종의 개미가 생존하고 있는 것으로 추산된다. 한국에는 137 별할 수 있다. 개미는 군체를 이루어 사는데, 수십 마리 규모로 작은 구멍에 사는 포식성 개미 군체가 있는가 하면 넓은 지역에 개미 대다수가 일개미나 병정개미 혹은 여타 분업 집단 등의 계급을 이루며, 그 밖에 임성이 있는 수개미와 하나 이상의 여왕개 하기도 하는데, 개미는 군체 전체를 위해 협동하며 단일한 개체처럼 움직이는 것으로 보이기 때문이다.^[3]

• 개체명 연결(Entity Linking) 모델

- 개체명 인식 절차를 통해 개체명이 인식되었다고 가정하고 중의성 해결에 목표를 둠
- Modeling Mention, Context and Entity with Neural Networks for Entity Disambiguation(Sun, IJCAI '15)
 - 신경망을 통해 mention, context, 후보 엔티티에 대한 vector를 구성하고 결합된 mention-context vector와 후보 entity vector 사이의 유사도를 구하고 유사도가 가장 높은 entity를 선택



- Modeling Mention, Context and Entity with Neural Networks for Entity Disambiguation (Sun, IJCAI '15)



– Mention/Context 표현 파트, Entity 표현 파트로 구성

- Mention/Context 표현 파트

- context는 CNN을 거친 후 max-pooling을 통해 고정 크기 벡터 얻음
- mention은 임베딩 후 평균을 취해 고정 크기 벡터 얻음
- NTN(Neural Tensor Network)로 결합 => 다양한 결합 방법이 있을 수 있음

- Modeling Mention, Context and Entity with Neural Networks for Entity Disambiguation(Sun, IJCAI '15)

- Mention/Context 표현 파트, Entity 표현 파트로 구성

- Entity 표현 파트

- 후보 엔티티의 word 표현은 mention과 동일하게 임베딩 후 평균
 - 후보 엔티티의 type 임베딩
 - Type 표현과 Word 표현을 NTN으로 결합하여 하나의 벡터로 구성

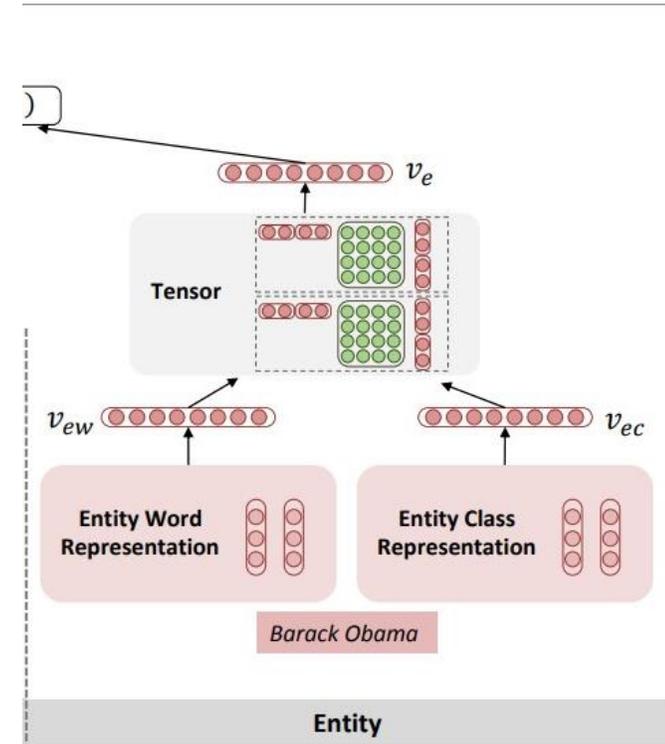
- 유사도로 판별

- 유사도가 최대가 되는 후보 엔티티를 선택

$$sim(e, mc) = cosine(v_e, v_{mc})$$

- 유사도가 최대가 되도록 학습

$$loss = \sum_{(m,c) \in T} max(0, 1 - sim(e, mc) + sim(e', mc))$$



- 개체명 연결(Entity Linking) 모델
 - 엔티티 표현으로 다양한 정보를 이용할 수 있음
 - 예) entity description

개미 (소설)

위키백과, 우리 모두의 백과사전.

《개미》(프랑스어: Les Fourmis)는 프랑스의 작가 베르나르 베르베르의 등단작이자 가장 유명한 작품이다.

- End-to-End 개체명 연결 모델
 - Mention Detection(멘션 추출)과 Disambiguation(중의성 해결)을 단일 모델에서 동시에 수행