CUAI 스터디 CS224n 1팀

Lecture 6: Simple and LSTM RNNs
Lecture 7: Translation, Seq2Seq, Attention
2022.05.12

발표자 : 김서린

Lecture 6
: Simple and LSTM RNNs

Lecture 6 Index

1. RNN Language Models
2. Other uses of RNNs
3. Exploding and vanishing gradients
4. LSTMs
5. Bidirectional and multi-layer RNNs

1. RNN Language Models

The simple RNN Language Model

- Many-to-many
- 매 time step마다 입,출력 존재

4 output distribution

$$\hat{y}^{(t)} = \text{softmax}\left(Uh^{(t)} + b_2\right) \in \mathbb{R}^{|V|}$$

3 hidden states

$$h^{(t)} = \sigma\left(W_h h^{(t-1)} + W_e e^{(t)} + b_1\right)$$

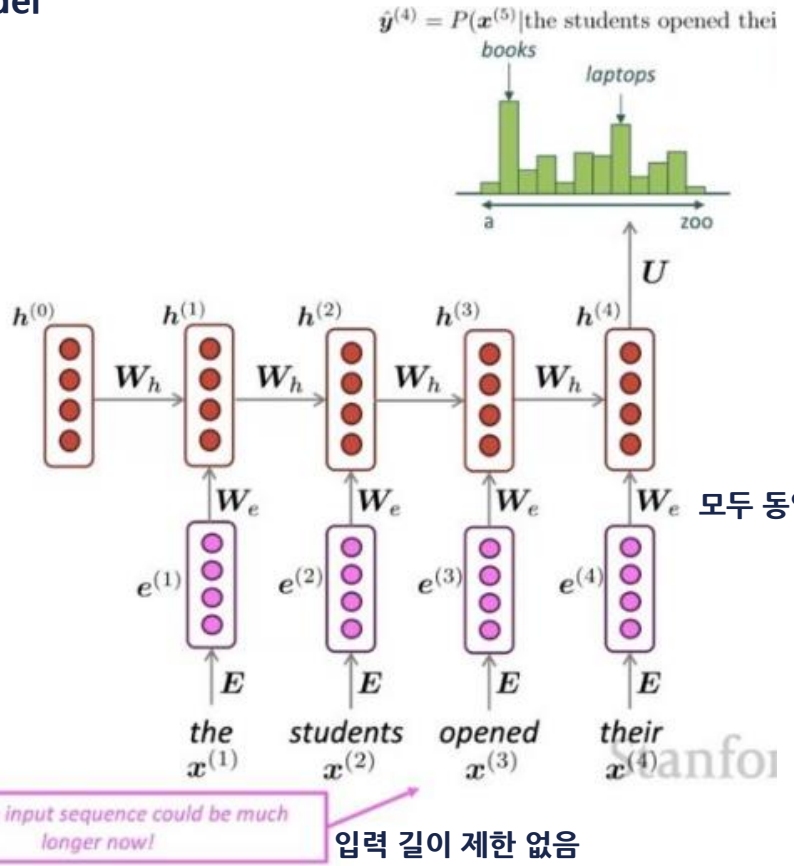
$h^{(0)}$ is the initial hidden state

2 word embeddings

$$e^{(t)} = Ex^{(t)}$$

1 words / one-hot vectors

$$x^{(t)} \in \mathbb{R}^{|V|}$$



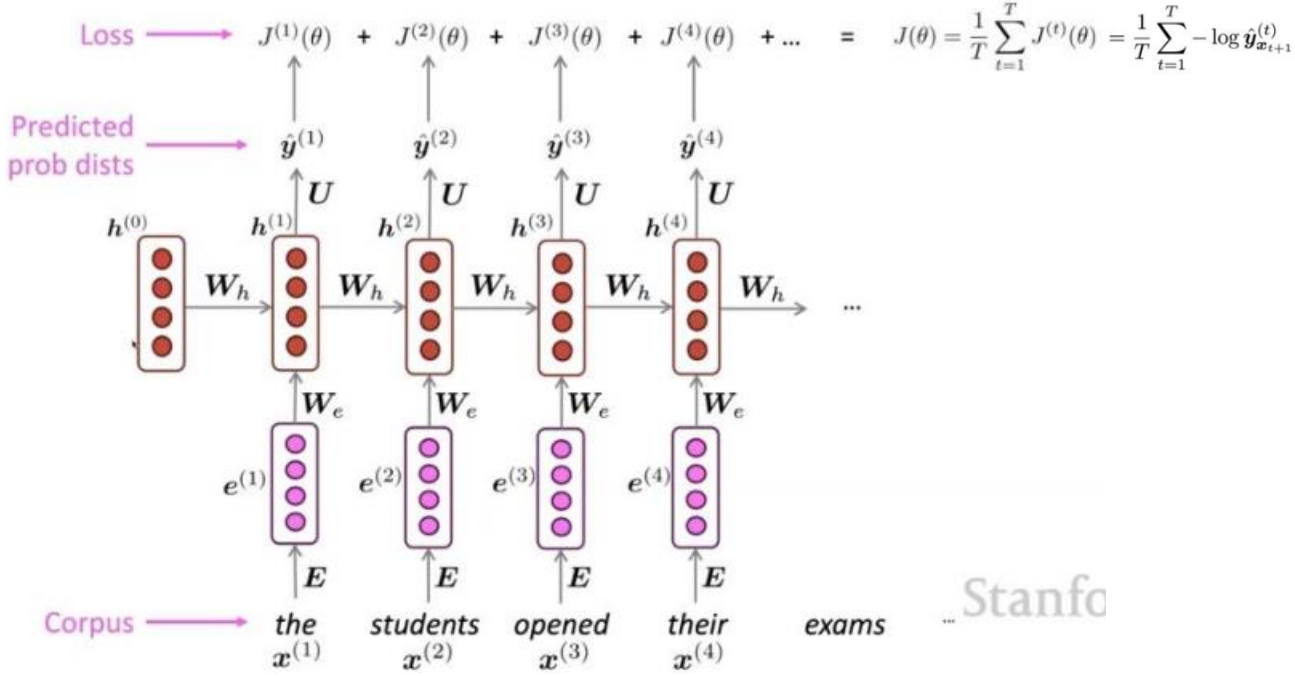
모두 동일한 가중치

입력 길이 제한 없음

1. RNN Language Models

Training an RNN Language Model (Loss Function)

$$J^{(t)}(\theta) = CE(\boldsymbol{y}^{(t)}, \hat{\boldsymbol{y}}^{(t)}) = -\sum_{w \in V} \boldsymbol{y}_w^{(t)} \log \hat{\boldsymbol{y}}_w^{(t)} = -\log \hat{\boldsymbol{y}}_{\boldsymbol{x}_{t+1}}^{(t)}$$



But, 모든 corpus에 대해 loss, gradient 계산하는 것 too expensive!

https://youtu.be/RM4SR430boI?list=PLetSlH8YjIfVk0G_lQfVCCOJ4M_iMZCJQ

1. RNN Language Models

Backpropagation for RNNs

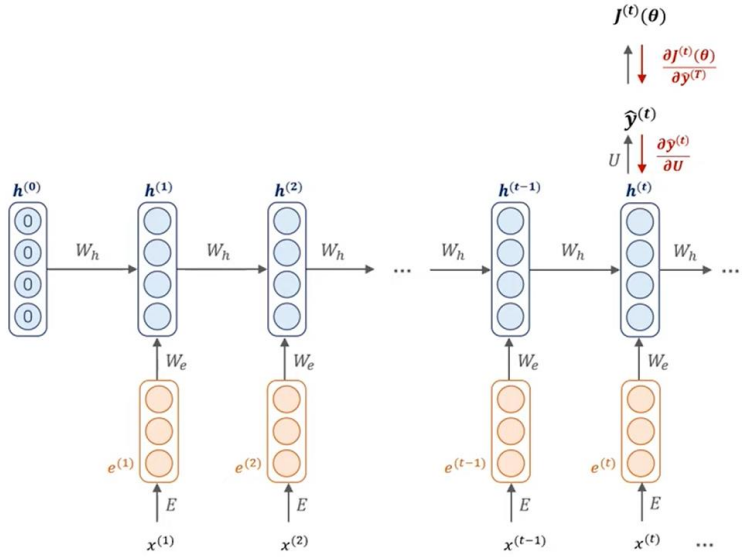
02

RNN Language Model

Back Propagation Through Time (BPTT) ①

$$\hat{y}^{(t)} = \text{softmax}\left(\boldsymbol{U}\boldsymbol{h}^{(t)} + \boldsymbol{b}_2\right) \in \mathbb{R}^{|V|}$$

$$\boldsymbol{h}^{(t)} = \sigma\left(\boldsymbol{W}_h\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_e\boldsymbol{e}^{(t)} + \boldsymbol{b}_1\right)$$



$$① \frac{\partial J^{(t)}(\theta)}{\partial U} = \frac{\partial J^{(t)}(\theta)}{\partial \hat{y}^{(t)}} \cdot \frac{\partial \hat{y}^{(t)}}{\partial U}$$

https://youtu.be/RM4SR430boI?list=PLetSlH8YjIfVk0G_lQfVCCOJ4M_iMZCJQ

1. RNN Language Models

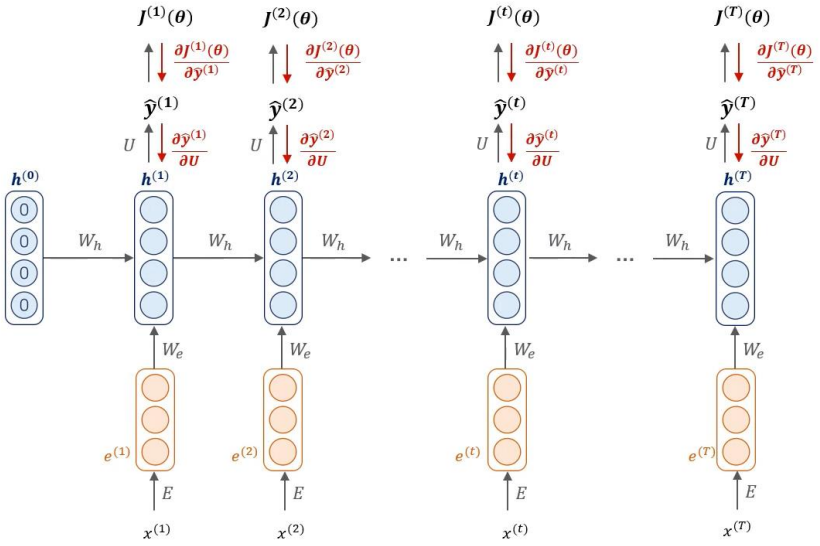
Backpropagation for RNNs

02 RNN Language Model

Back Propagation Through Time (BPTT) ①

$$\hat{y}^{(t)} = \text{softmax}\left(\boldsymbol{U}\boldsymbol{h}^{(t)} + \boldsymbol{b}_2\right) \in \mathbb{R}^{|V|}$$

$$\boldsymbol{h}^{(t)} = \sigma\left(\boldsymbol{W}_h\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_e\boldsymbol{e}^{(t)} + \boldsymbol{b}_1\right)$$



$$① \frac{\partial J^{(t)}(\theta)}{\partial U} = \frac{\partial J^{(t)}(\theta)}{\partial \hat{y}^{(t)}} \cdot \frac{\partial \hat{y}^{(t)}}{\partial U}$$

$$\frac{\partial J(\theta)}{\partial U} = \sum_{t=1}^{T} \frac{\partial J^{(t)}(\theta)}{\partial U}$$

$$U^{new} = U^{old} - \alpha \frac{\partial J(\theta)}{\partial U}$$

https://youtu.be/RM4SR430boI?list=PLetSlH8YjIfVk0G_lQfVCCOJ4M_iMZCJQ

1. RNN Language Models

Backpropagation for RNNs

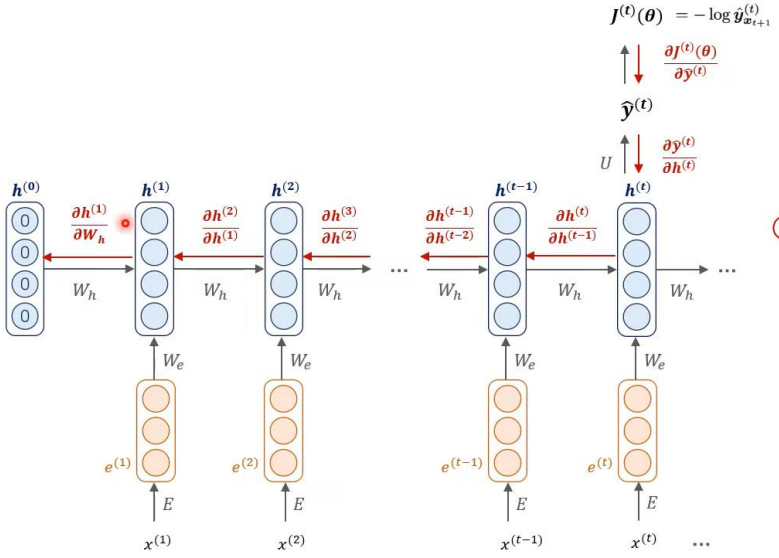
02

RNN Language Model

Back Propagation Through Time (BPTT) ②

$$\hat{\boldsymbol{y}}^{(t)} = \text{softmax}\left(\boldsymbol{U}\boldsymbol{h}^{(t)} + \boldsymbol{b}_2\right) \in \mathbb{R}^{|V|}$$

$$\boldsymbol{h}^{(t)} = \sigma\left(\boldsymbol{W}_h\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_e\boldsymbol{e}^{(t)} + \boldsymbol{b}_1\right)$$



$$② \frac{\partial J^{(t)}(\theta)}{\partial W_h} = \frac{\partial J^{(t)}(\theta)}{\partial \hat{y}^{(t)}} \cdot \frac{\partial \hat{y}^{(t)}}{\partial h^{(t)}} \cdot \frac{\partial h^{(t)}}{\partial W_h}$$

$$+ \frac{\partial J^{(t)}(\theta)}{\partial \hat{y}^{(t)}} \cdot \frac{\partial \hat{y}^{(t)}}{\partial h^{(t)}} \cdot \frac{\partial h^{(t)}}{\partial h^{(t-1)}} \cdot \frac{\partial h^{(t-1)}}{\partial W_h}$$

$$\vdots$$

$$+ \frac{\partial J^{(t)}(\theta)}{\partial \hat{y}^{(t)}} \cdot \frac{\partial \hat{y}^{(t)}}{\partial h^{(t)}} \cdot \frac{\partial h^{(t)}}{\partial h^{(t-1)}} \cdot \ldots \cdot \frac{\partial h^{(2)}}{\partial h^{(1)}} \cdot \frac{\partial h^{(1)}}{\partial W_h}$$

$$= \sum_{i=1}^{t} \frac{\partial J^{(t)}(\theta)}{\partial \hat{y}^{(t)}} \cdot \frac{\partial \hat{y}^{(t)}}{\partial h^{(t)}} \cdot \frac{\partial h^{(t)}}{\partial h^{(i)}} \cdot \frac{\partial h^{(i)}}{\partial W_h}$$

https://youtu.be/RM4SR430boI?list=PLetSlH8YjIfVk0G_lQfVCCOJ4M_iMZCJQ

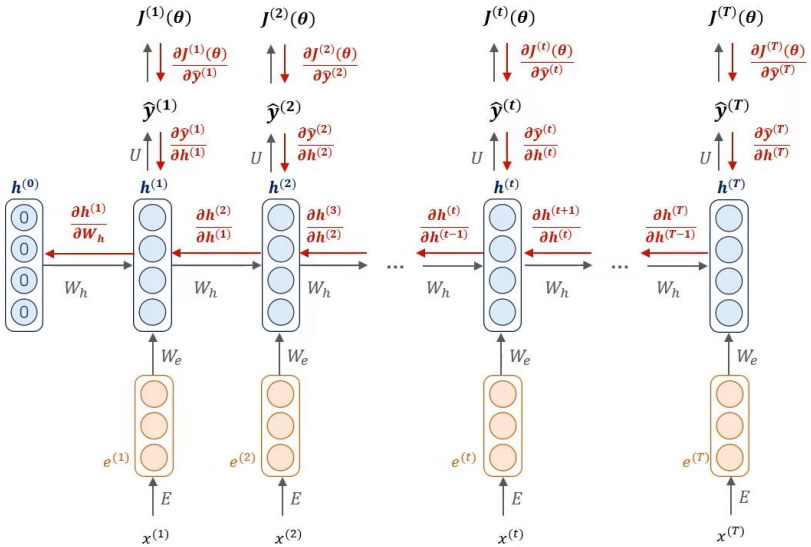
1. RNN Language Models

Backpropagation for RNNs

02

RNN Language Model

Back Propagation Through Time (BPTT) ②



$$\hat{y}^{(t)} = \text{softmax}\left(Uh^{(t)} + b_2\right) \in \mathbb{R}^{|V|}$$

$$h^{(t)} = \sigma\left(W_h h^{(t-1)} + W_e e^{(t)} + b_1\right)$$

$$② \frac{\partial J^{(t)}(\theta)}{\partial W_h} = \sum_{i=1}^{t} \frac{\partial J^{(t)}(\theta)}{\partial \hat{y}^{(t)}} \cdot \frac{\partial \hat{y}^{(t)}}{\partial h^{(t)}} \cdot \frac{\partial h^{(t)}}{\partial h^{(i)}} \cdot \frac{\partial h^{(i)}}{\partial W_h}$$

$$\frac{\partial J(\theta)}{\partial W_h} = \sum_{t=1}^{T} \frac{\partial J^{(t)}(\theta)}{\partial W_h}$$

$$W_h{}^{new} = W_h{}^{old} - \alpha \frac{\partial J(\theta)}{\partial W_h}$$

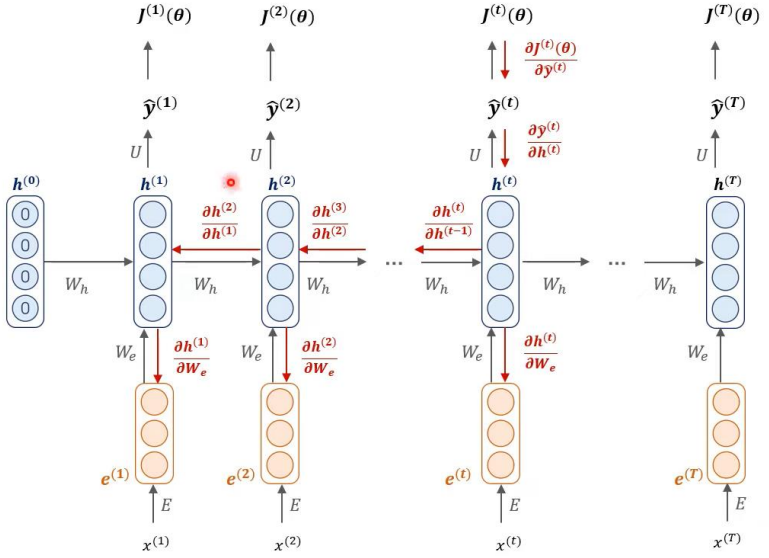
https://youtu.be/RM4SR430boI?list=PLetSlH8YjIfVk0G_lQfVCCOJ4M_iMZCJQ

1. RNN Language Models

Backpropagation for RNNs

02 RNN Language Model

Back Propagation Through Time (BPTT) ③



$$\hat{y}^{(t)} = \text{softmax}\left(Uh^{(t)} + b_2\right) \in \mathbb{R}^{|V|}$$

$$h^{(t)} = \sigma\left(W_h h^{(t-1)} + W_e e^{(t)} + b_1\right)$$

$$③\ \frac{\partial J^{(t)}(\theta)}{\partial W_e} = \sum_{i=1}^{t} \frac{\partial J^{(t)}(\theta)}{\partial \hat{y}^{(t)}} \cdot \frac{\partial \hat{y}^{(t)}}{\partial h^{(t)}} \cdot \frac{\partial h^{(t)}}{\partial h^{(i)}} \cdot \frac{\partial h^{(i)}}{\partial W_e}$$

https://youtu.be/RM4SR430boI?list=PLetSlH8YjIfVk0G_lQfVCCOJ4M_iMZCJQ

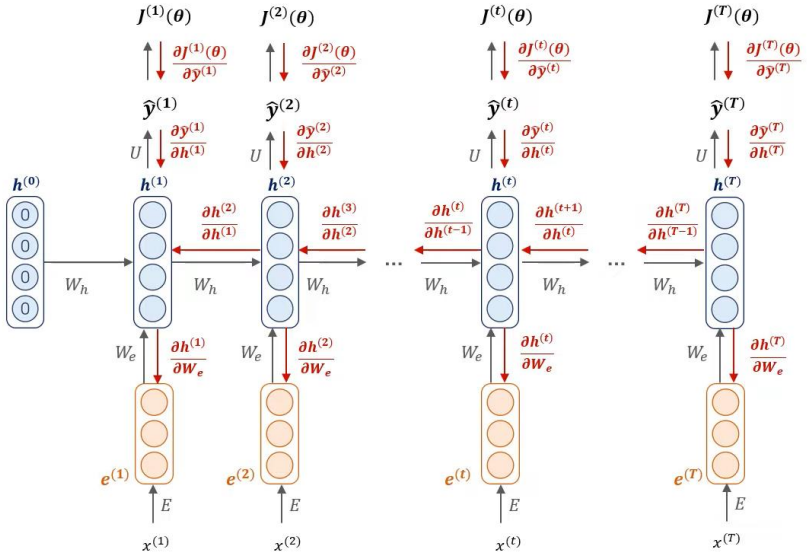
1. RNN Language Models

Backpropagation for RNNs

02

RNN Language Model

Back Propagation Through Time (BPTT) ③



$$\hat{y}^{(t)} = \text{softmax}\left(Uh^{(t)} + b_2\right) \in \mathbb{R}^{|V|}$$

$$h^{(t)} = \sigma\left(W_h h^{(t-1)} + W_e e^{(t)} + b_1\right)$$

$$③\ \frac{\partial J^{(t)}(\theta)}{\partial W_e} = \sum_{i=1}^{t} \frac{\partial J^{(t)}(\theta)}{\partial \hat{y}^{(t)}} \cdot \frac{\partial \hat{y}^{(t)}}{\partial h^{(t)}} \cdot \frac{\partial h^{(t)}}{\partial h^{(i)}} \cdot \frac{\partial h^{(i)}}{\partial W_e}$$

$$\frac{\partial J(\theta)}{\partial W_e} = \sum_{t=1}^{T} \frac{\partial J^{(t)}(\theta)}{\partial W_e}$$

$$W_e{}^{new} = W_e{}^{old} - \alpha \frac{\partial J(\theta)}{\partial W_e}$$

1. RNN Language Models

Generating text with a RNN Language Model

- N-gram language model과 같이, sampling을 반복하며 text를 생성
- Output이 다음 단계의 input

- ‘Harry Potter’로 train

“Sorry,” Harry shouted, panicking—“I’ll leave those brooms in London, are they?”

“No idea,” said Nearly Headless Nick, casting low close by Cedric, carrying the last bit of treacle Charms, from Harry’s shoulder, and to answer him the common room perched upon it, four arms held a shining knob from when the spider hadn’t felt it seemed. He reached the teams too.

Stanford

Source: https://medium.com/deep-writing/harry-potter-written-by-artificial-intelligence-8a9431803da6

- ‘recipes’로 train

```
    Title: CHOCOLATE RANCH BARBECUE
Categories: Game, Casseroles, Cookies, Cookies
    Yield: 6 Servings

    2 tb Parmesan cheese -- chopped
    1 c  Coconut milk
    3    Eggs, beaten

Place each pasta over layers of lumps. Shape mixture into the moderate oven and simmer
until firm. Serve hot in bodied fresh, mustard, orange and cheese.

Combine the cheese and salt together the dough in a large skillet; add the ingredients
and stir in the chocolate and pepper.
```

Stanfo

Source: https://gist.github.com/nylki/1efbaa36635956d35bcc

1. RNN Language Models

Evaluating Language Models

- Language Model의 표준 evaluation metric은 perplexity

$$\text{perplexity} = \prod_{t=1}^{T} \left(\frac{1}{P_{\text{LM}}(\boldsymbol{x}^{(t+1)} | \boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(1)})} \right)^{1/T}$$

Normalized by number of words

Inverse probability of corpus, according to Language Model

- 이것은 cross-entropy loss인 $J(\theta)$의 exponential과 같음

$$= \prod_{t=1}^{T} \left(\frac{1}{\hat{\boldsymbol{y}}^{(t)}_{\boldsymbol{x}_{t+1}}} \right)^{1/T} = \exp\left(\frac{1}{T} \sum_{t=1}^{T} -\log \hat{\boldsymbol{y}}^{(t)}_{\boldsymbol{x}_{t+1}} \right) = \exp(J(\theta))$$

따라서 perplexity 낮을수록 좋음!

n-gram model

Increasingly complex RNNs

Model	Perplexity
Interpolated Kneser-Ney 5-gram (Chelba et al., 2013)	67.6
RNN-1024 + MaxEnt 9-gram (Chelba et al., 2013)	51.3
RNN-2048 + BlackOut sampling (Ji et al., 2015)	68.3
Sparse Non-negative Matrix factorization (Shazeer et al., 2015)	52.9
LSTM-2048 (Jozefowicz et al., 2016)	43.7
2-layer LSTM-8192 (Jozefowicz et al., 2016)	30
Ours small (LSTM-2048)	43.9
Ours large (2-layer LSTM-2048)	39.8

Perplexity improves (lower is better)

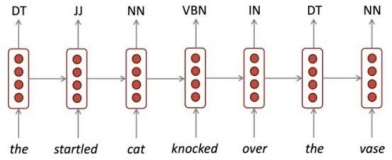
2. Other uses of RNNs

정리

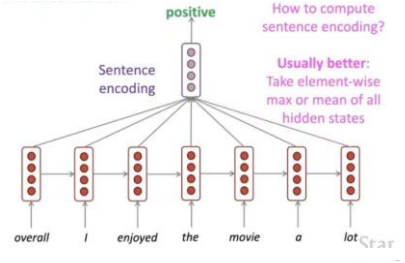
- Language Modeling은 많은 NLP task 중 하나
- (generating text, estimating the probability of text)
- RNN은 Language Model을 구축하기 위한 좋은 방법 중 하나
- RNN은 다른 곳에서도 사용됨

2. Other uses of RNNs

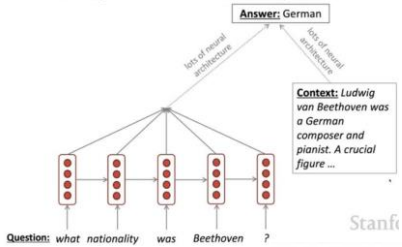
1. Sequence tagging
Ex) part-of-speech tagging, named entity recognition



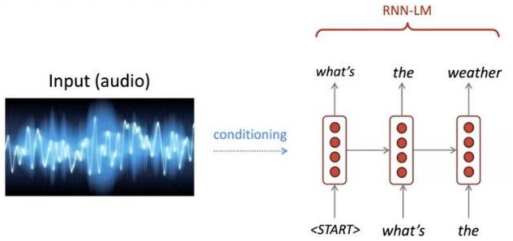
2. Sentence classification
Ex) sentiment classification



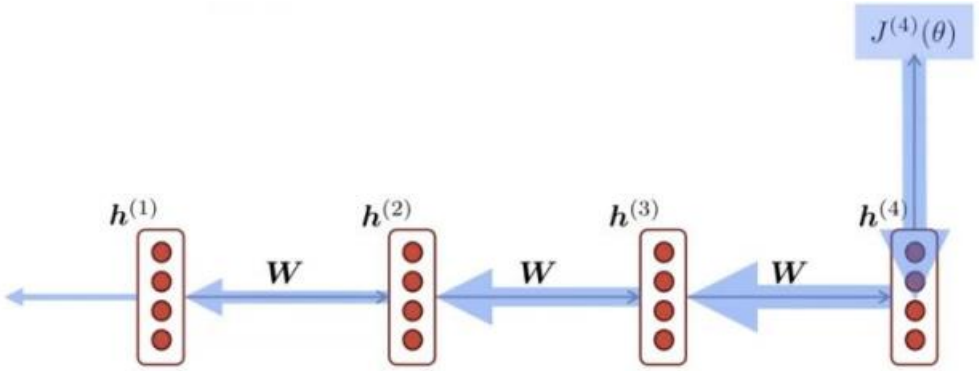
3. Language encoder module
Ex) question answering, machine translation



4. Generate text
Ex) speech recognition, machine translation, summarization



3. Exploding and vanishing gradients



$$\frac{\partial J^{(t)}(\theta)}{\partial W_h} = \frac{\partial J^{(t)}(\theta)}{\partial \hat{y}^{(t)}} \cdot \frac{\partial \hat{y}^{(t)}}{\partial h^{(t)}} \cdot \frac{\partial h^{(t)}}{\partial W_h} + \frac{\partial J^{(t)}(\theta)}{\partial \hat{y}^{(t)}} \cdot \frac{\partial \hat{y}^{(t)}}{\partial h^{(t)}} \cdot \boxed{\frac{\partial h^{(t)}}{\partial h^{(t-1)}}}_{W_h} \cdot \frac{\partial h^{(t-1)}}{\partial W_h} \cdots + \frac{\partial J^{(t)}(\theta)}{\partial \hat{y}^{(t)}} \cdot \frac{\partial \hat{y}^{(t)}}{\partial h^{(t)}} \cdot \boxed{\frac{\partial h^{(t)}}{\partial h^{(t-1)}} \cdot \cdots \cdot \frac{\partial h^{(2)}}{\partial h^{(1)}}}_{(W_h)^{t-1}} \cdot \frac{\partial h^{(1)}}{\partial W_h}$$

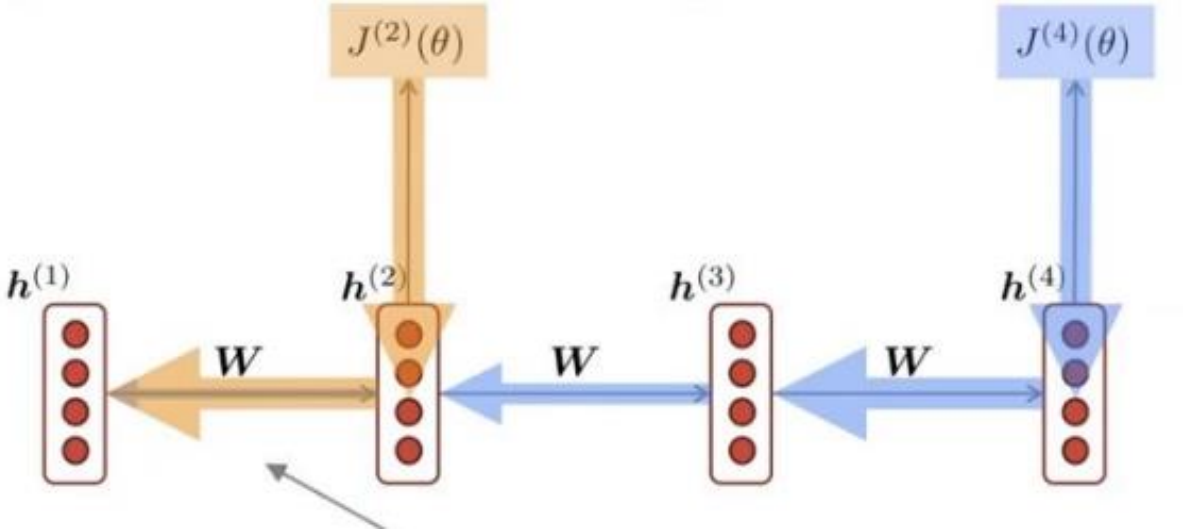
동일한 가중치(W) 공유

→ W < 1, 반복적으로 곱해지는 값이 0에 가까워져 gradient vanishing

→ W > 1, 반복적으로 곱해지는 값이 기하급수적으로 커져 gradient exploding

3. Exploding and vanishing gradients

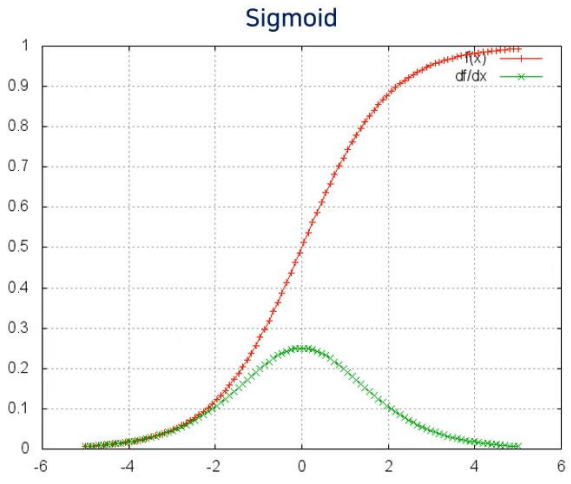
Vanishing gradient의 문제점

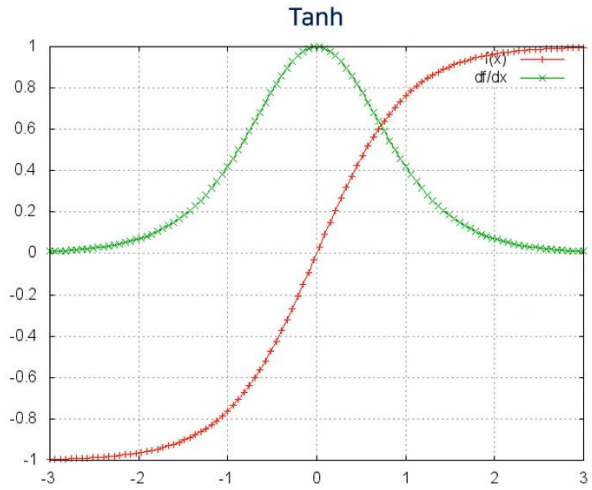


멀리 떨어져 있는 gradient signal이 소실됨 → long term dependency를 잘 반영하지 못함

3. Exploding and vanishing gradients

Vanishing gradient 완화





Hidden states 도출 시 activation function으로 Tanh 사용
(Sigmoid : 기울기 0~0.25 / Tanh : 기울기 0~1) → gradient vanishing에 더 강함

3. Exploding and vanishing gradients

Exploding gradient의 문제점

$$\theta^{new} = \theta^{old} - \overbrace{\alpha}^{\text{learning rate}} \underbrace{\nabla_\theta J(\theta)}_{\text{gradient}}$$

Gradient가 커지면 bad 업데이트 → loss 커짐 → 최악의 경우 Inf or NaN에 도달할 수도 있음

3. Exploding and vanishing gradients

Exploding gradient 해결 : Gradient clipping

Algorithm 1 Pseudo-code for norm clipping

$\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta}$

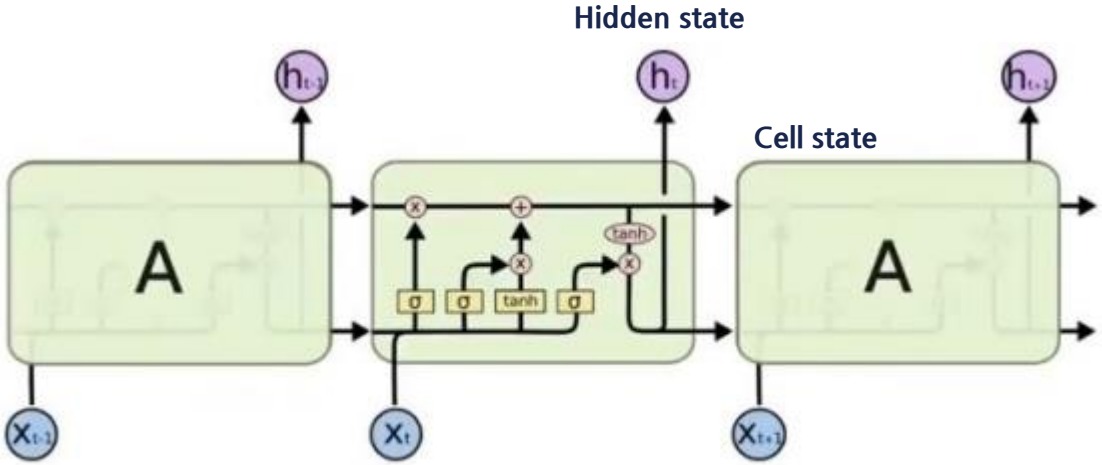
if $\|\hat{\mathbf{g}}\| \geq threshold$ **then**

$\hat{\mathbf{g}} \leftarrow \frac{threshold}{\|\hat{\mathbf{g}}\|}\hat{\mathbf{g}}$

end if

Gradient의 norm이 threshold 보다 크면, SGD update를 적용하기 전에 작게 조정해주는 방법
같은 방향이지만 작은 step으로 업데이트 되도록

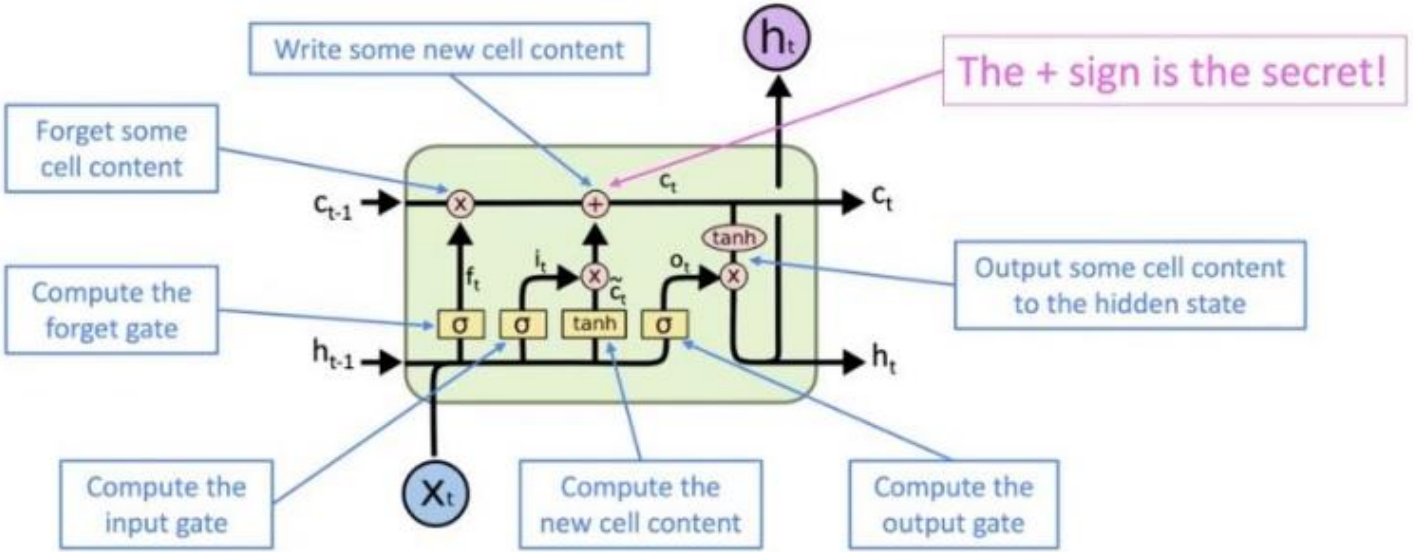
4. LSTMs



- Hidden state를 통해 short term memory 조절
- Cell state를 통해 long term memory 보존
- Forget, input, output, 3개의 gate를 통해 매 time step의 cell state와 hidden state, input에서 취할 정보의 양 결정

4. LSTMs

Architecture



https://youtu.be/RM4SR430boI?list=PLetSlH8YjIfVk0G_lQfVCCOJ4M_iMZCJQ

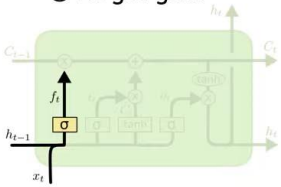
4. LSTMs

Architecture

04 LSTM: Long Short-Term Memory RNN

Architecture

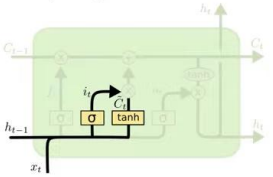
① forget gate



$$f_t = \sigma(W_f h^{(t-1)} + U_f x^{(t)} + b_f)$$

- 입력 정보(새로운 입력 시퀀스와 이전 시점의 hidden state)에 시그모이드를 취함

② input gate, new cell content



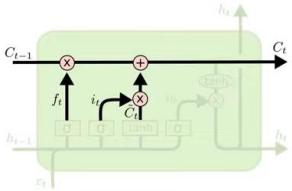
$$i_t = \sigma(W_i h^{(t-1)} + U_i x^{(t)} + b_i)$$

- input gate: 입력 정보에 시그모이드를 취함

$$\tilde{C}_t = \tanh(W_c h^{(t-1)} + U_c x^{(t)} + b_c)$$

- new cell content: 입력 정보에 tanh를 취함

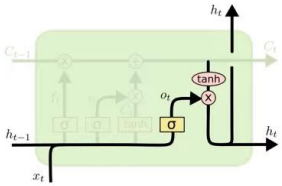
③ Cell state



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

- forget gate: 이전 시점의 cell state에서 어느 정도의 정보를 가져갈 것인지 결정
- input gate: 입력 정보에서 장기 기억으로 가져갈 정보의 양 결정
- element wise product

④ Output gate, hidden state



$$o_t = \sigma(W_o h^{(t-1)} + U_o x^{(t)} + b_o)$$

- output gate: 입력 정보에 시그모이드를 취함

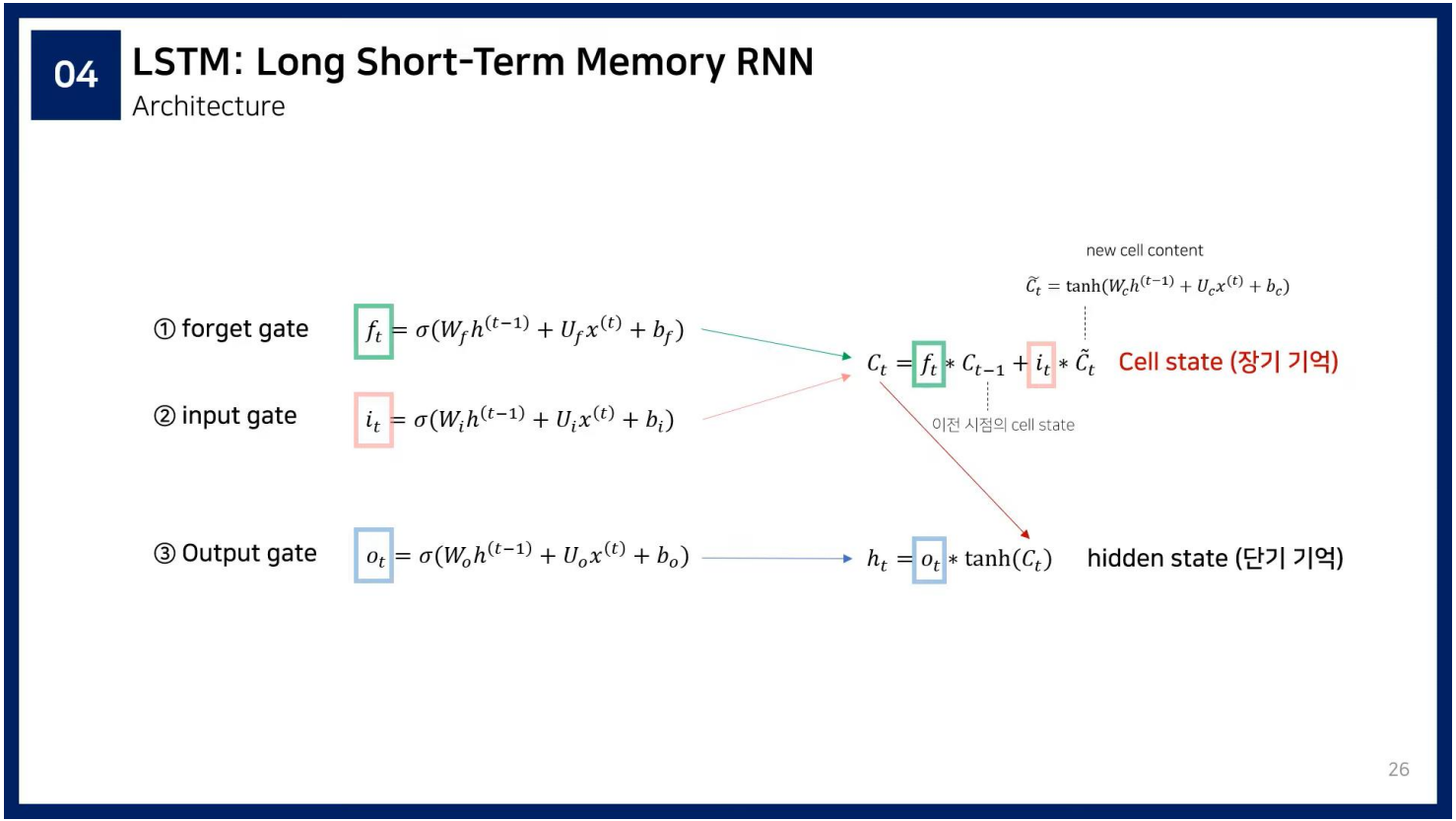
$$h_t = o_t * \tanh(C_t)$$

- hidden state: 현재 입력과 대비해서 장기 기억에서 어느 정도의 정보를 단기 기억으로 사용할지 결정

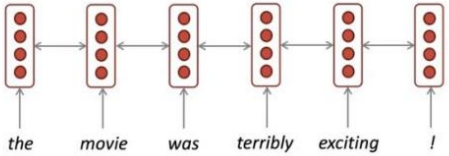
https://youtu.be/RM4SR430boI?list=PLetSlH8YjIfVk0G_lQfVCCOJ4M_iMZCJQ

4. LSTMs

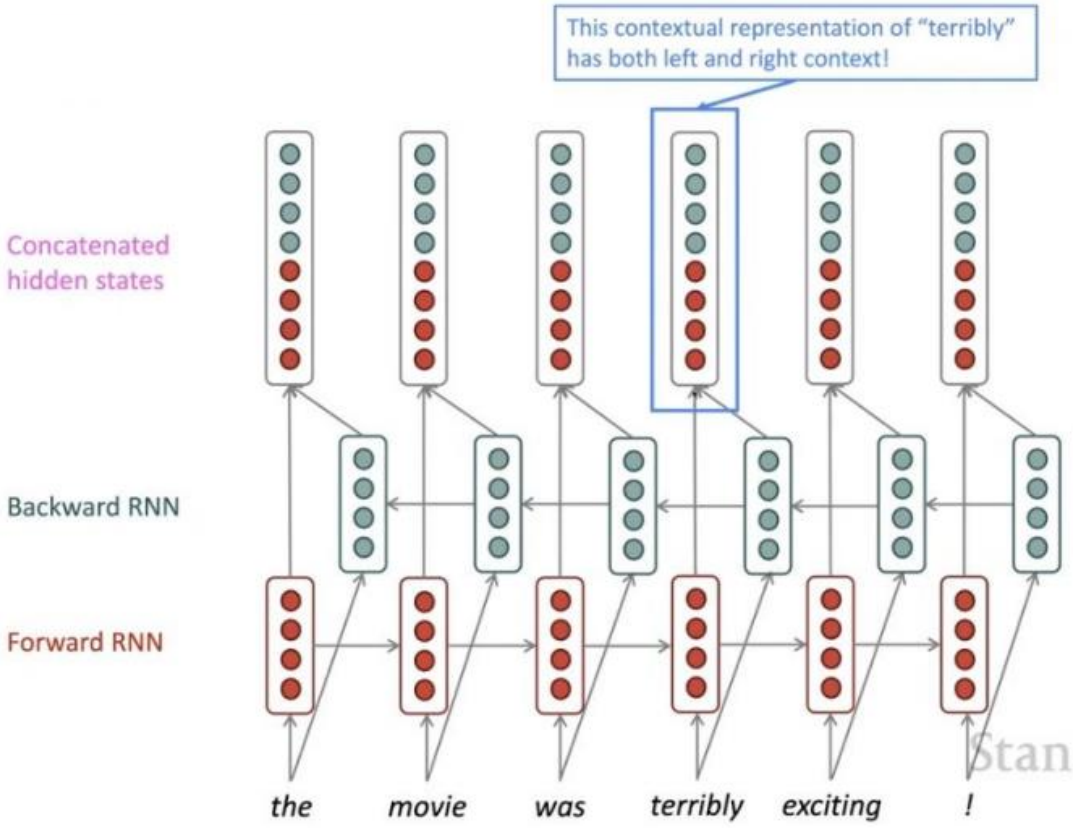
Architecture



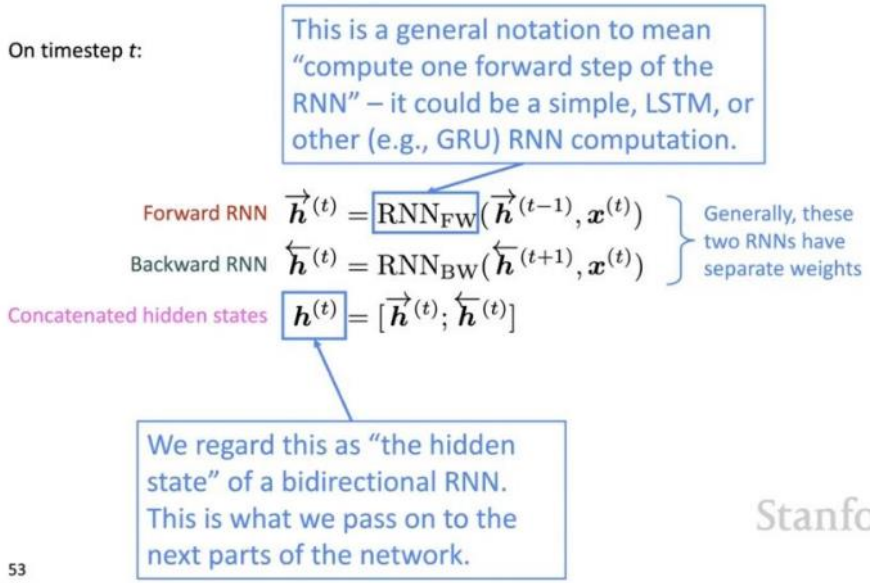
5. Bidirectional and multi-layer RNNs



Simplified diagram



5. Bidirectional and multi-layer RNNs



Bidirectional RNN은 entire input sequence에 접근하려 할 때에만 적용 가능

Lecture 7
: Translation, Seq2Seq, Attention

Lecture 7 Index

1. Machine Translation
2. Sequence-to-sequence (Seq2Seq)
3. Attention

Machine Translation은 Seq2Seq의 대표적인 use-case이고, Attention을 통해 성능이 개선됨

1. Machine Translation

x: *L'homme est né libre, et partout il est dans les fers*

y: *Man is born free, but everywhere he is in chains*

Machine Translation : source language의 문장 x를 target language의 문장 y로 번역하는 것

1. Machine Translation

Pre-Neural Machine Translation

1950s : The early history of Machine Translation

- 군사 목적으로 개발되기 시작함 (Russian → English)
- 같은 뜻의 단어를 대체하는 단순한 방식을 사용

1990s - 2010s : Statistical Machine Translation

- 확률 모델을 활용하기 시작
- Translation Model + Language Model (Bayes Rule)
- 입력 문장 x에 대해 conditional prob.를 최대화하는 번역 문장 y를 찾고자 함

$$\text{argmax}_y P(y|x) = \text{argmax}_y P(x|y)P(y)$$

각 단어와 구가 어떻게 번역되어야 하는지 parallel data로부터 학습

Translation Model

Models how words and phrases should be translated (*fidelity*). Learnt from parallel data.

Language Model

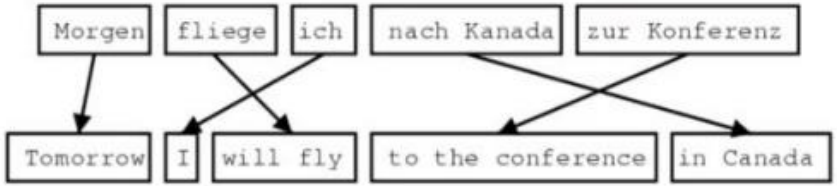
Models how to write good English (*fluency*). Learnt from monolingual data.

가장 그럴듯한 target 문장을 생성하도록 단일 언어 data로부터 학습

1. Machine Translation

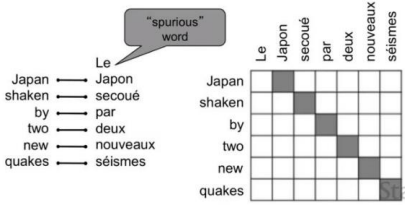
Statistical Machine Translation (SMT)

- 많은 양의 parallel data가 필요함
- Source-target sentence 간의 correspondence를 mapping하는 alignment 사용

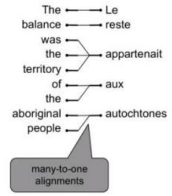


1:1 mapping

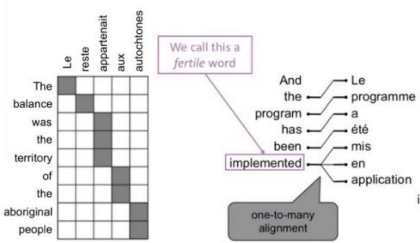
- But, 1:1 mapping이 불가능한 경우가 많음 → EM algorithm



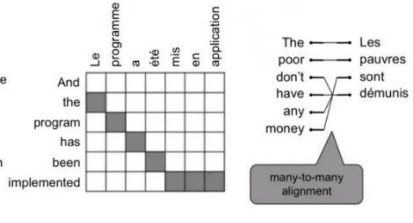
No counterpart

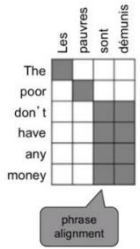


Many-to-one



One-to-many

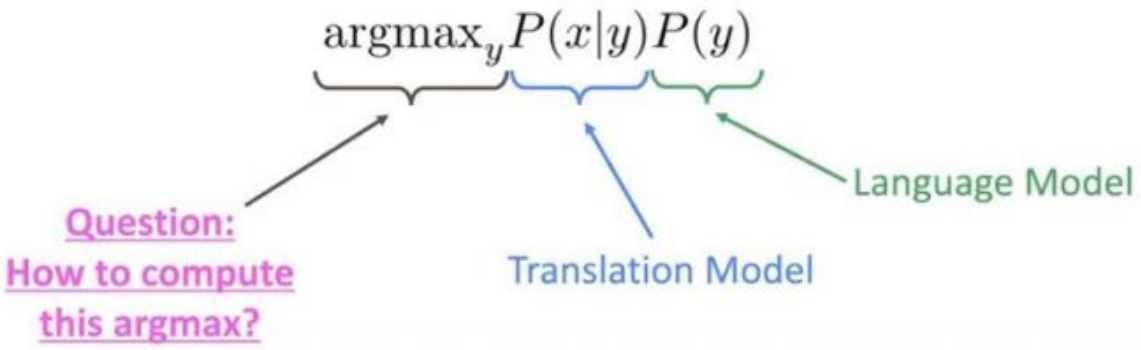




Many-to-many

1. Machine Translation

Decoding for SMT



- 가능한 모든 y를 비교 → 계산량 너무 많음
- 따라서 Dynamic Programming
- 여러 factor로 분리하고 각각의 최적해 조합으로 global optimal 찾기

1. Machine Translation

Neural Machine Translation

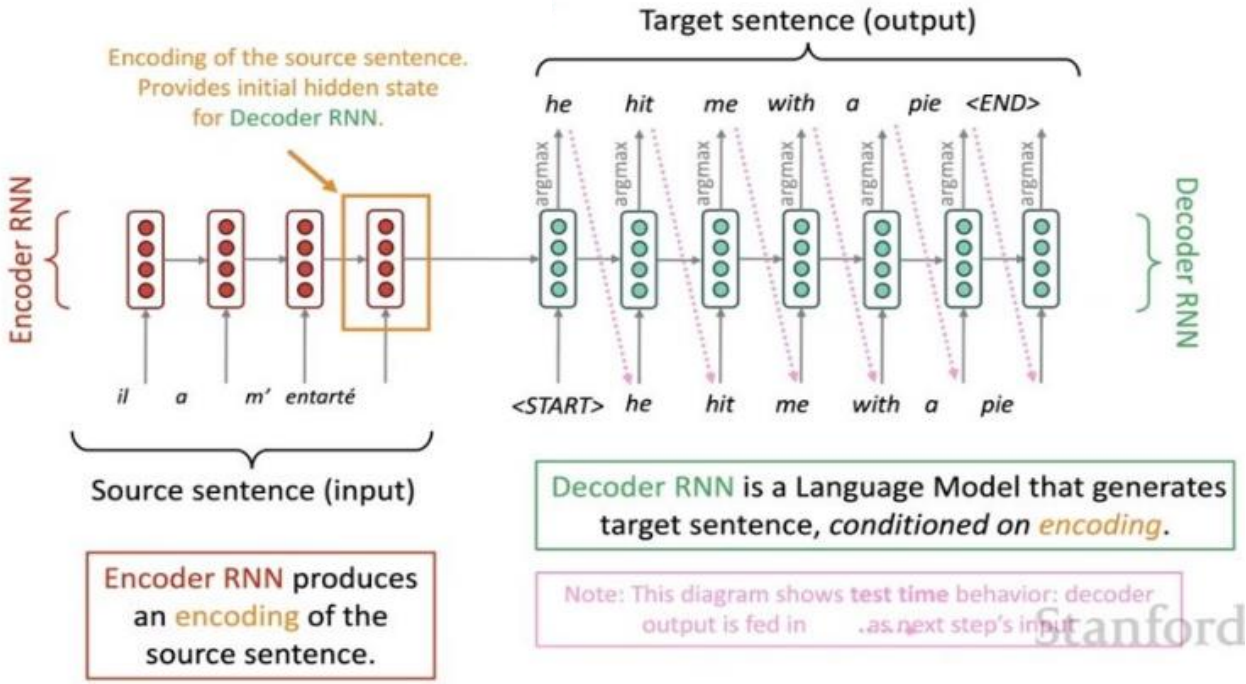
Statistical Machine Translation (SMT)

- 많은 하위 시스템으로 전체 시스템을 설계해야 함
- Feature engineering이 필요함
- 성능을 위한 추가 resource 필요 (phrase table)
- 사람의 노력이 많이 필요함 (paired dataset, subcomponent 학습)

Neural Machine Translation (NMT, 2014~)

- Single end-to-end neural network → Subcomponents 따로 최적화할 필요 X
- RNN 2개로 구성된 Seq2Seq model
- SMT보다 뛰어난 성능
- 사람의 노력 줄어듦 (No feature engineering, 모든 source-target pair에 대한 동일한 method 가능)
- Interpretability가 부족
- Rule, guideline 적용이 쉽지 않음

2. Sequence-to-sequence (Seq2Seq)



Train : Target sentence의 timestep 별 word → 다음 timestep의 word를 예측하기 위한 입력

Test : Decoder RNN의 timestep 별 output → 다음 timestep의 입력

2. Sequence-to-sequence (Seq2Seq)

- Machine Translation이 가장 대표적인 use-case
- 이외에도 많은 NLP task들이 Seq2Seq 활용
  - ✓ Summarization (long text → short text)
  - ✓ Dialogue (previous utterances → next utterance)
  - ✓ Parsing (input text → output parse as sequence)
  - ✓ Code generation (natural language → Python code)

- Conditional probability 직접 계산 (Bayes Rule 없이)

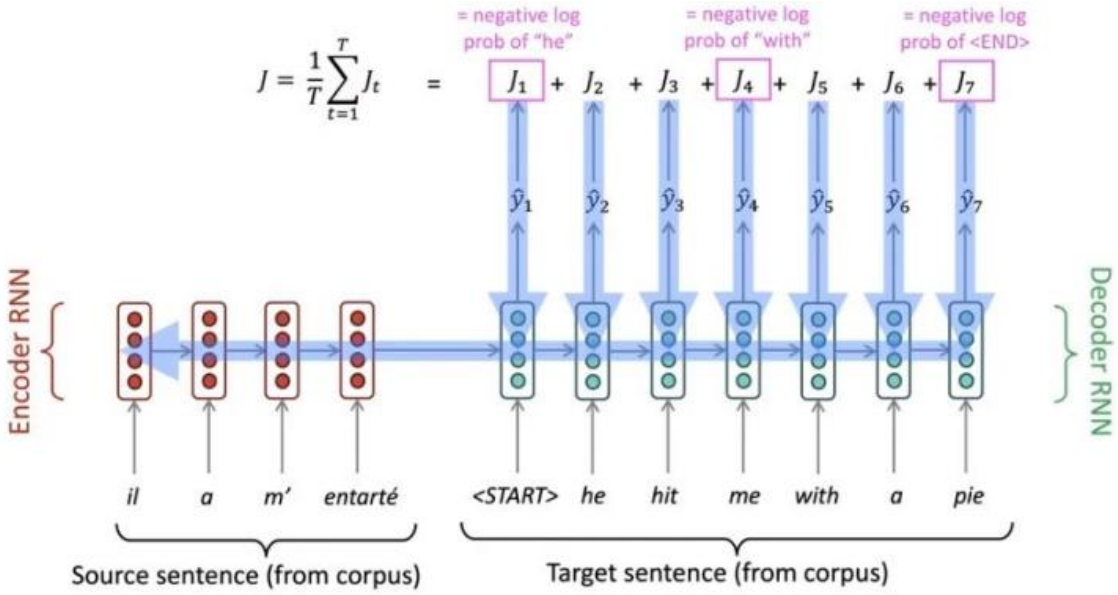
$$P(y|x) = P(y_1|x)\,P(y_2|y_1,x)\,P(y_3|y_1,y_2,x)\ldots\underbrace{P(y_T|y_1,\ldots,y_{T-1},x)}$$

Probability of next target word, given target words so far and source sentence *x*

→ End-to-end 학습 가능하지만, 여전히 parallel corpus가 필요

2. Sequence-to-sequence (Seq2Seq)

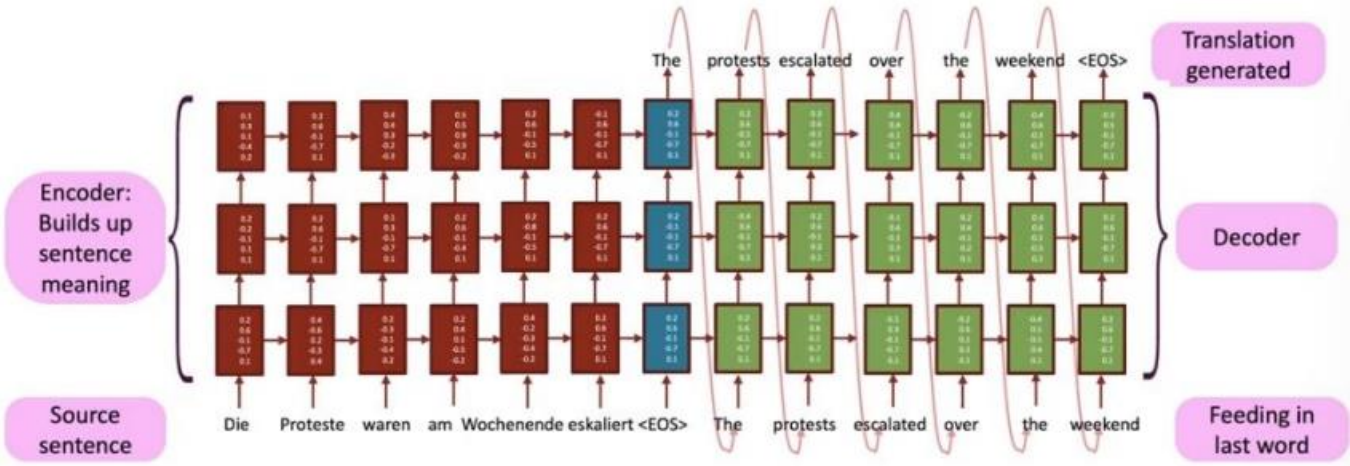
Training a Neural Machine Translation system



NLL loss : $L(y) = -\frac{1}{K}\sum_{k=1}^{K} \log(y)$, probability를 [0,∞]로 매핑하여 loss 최소화 문제로 정의함

2. Sequence-to-sequence (Seq2Seq)

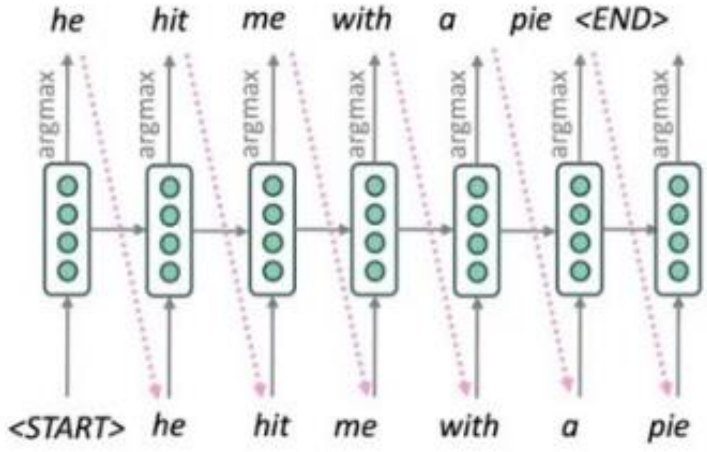
Multi-layer RNNs



- Timestep의 진행과 평행하게 layer를 추가 → higher-lever feature 학습 가능
- 좋은 성능을 내는 RNN 모델은 대부분 Multi-layer RNNs 구조를 사용
  - ✓ Encoder : 2-4 layers / Decoder : 4 layers
  - ✓ Transformer-based : 12 or 24 layers

2. Sequence-to-sequence (Seq2Seq)

Greedy Decoding



- Input: *il a m'entarté* *(he hit me with a pie)*
- → *he* ____
- → *he hit* ____
- → *he hit a* ____ *(whoops! no going back now...)*

- Exhaustive search 방식 계산량 많음 → Greedy search
- Step 별 argmax word를 선택하는 방식
- 부자연스러운 문장이 생성될 수 있음 (최적해 보장 X)

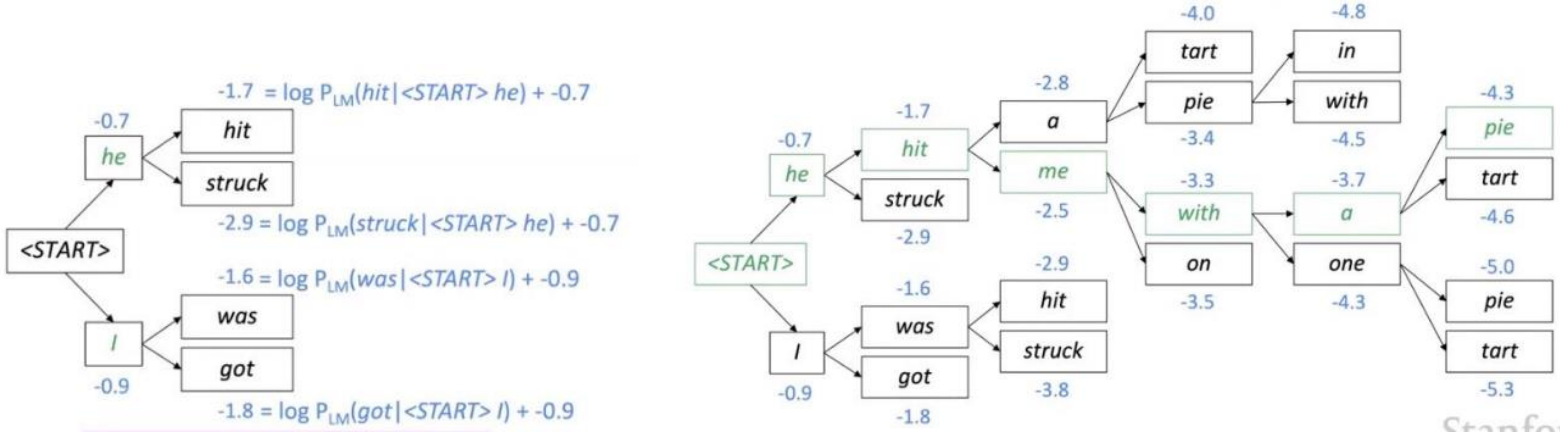
$$P(y|x) = P(y_1|x)\,P(y_2|y_1,x)\,P(y_3|y_1,y_2,x)\ldots,P(y_T|y_1,\ldots,y_{T-1},x)$$
$$= \prod_{t=1}^{T} P(y_t|y_1,\ldots,y_{t-1},x)$$

Exhaustive search

2. Sequence-to-sequence (Seq2Seq)

Beam Searching Decoding

- k개의 best translation을 유지하며 decoding (k : beam size, 보통 5-10)
- 최적해 보장하진 않지만 exhaustive보다 효율적이고 greedy보다 더 자연스러움



k^2의 hypotheses 비교하여 k개의 best hypotheses를 선택하는 과정 반복

EOS 토큰의 등장이나 사전 정의된 max timestep T, 또는 n개의 완성된 문장을 종료조건으로 함

NLL loss 합을 생성된 문장의 길이로 나눠서 normalize

https://youtu.be/gyTm_s3-Ofo?list=PLetSlH8YjIfVk0G_lQfVCCOJ4M_iMZCJQ

2. Sequence-to-sequence (Seq2Seq)

Evaluation for Machine Translation : BLEU (BiLingual Evaluation Understudy)

02

Sequence-to-sequence

Neural Machine Translation

Evaluation for Machine Translation : BLEU

- Target sentence 와 machine-written translation 을 비교
- N-gram precision 기반 similarity score + penalty for too short translation

- `예측된 sentence` : **빛이 쐬는** 노인은 **완벽한** 어두운곳에서 **잠든 사람과 비교할 때** 강박증이 **심해질** 기회가 **훨씬 높았다**
- `true sentence` : **빛이 쐬는** 사람은 **완벽한** 어둠에서 **잠든 사람과 비교할 때** 우울증이 **심해질** 가능성이 **훨씬 높았다**

$$BLEU = min(1, \frac{output\ length(\text{예측 문장})}{reference\ length(\text{실제 문장})})(\prod_{i=1}^{4} precision_i)^{\frac{1}{4}}$$

짧은 문장에 대한 penalty — N-gram precision 의 기하평균

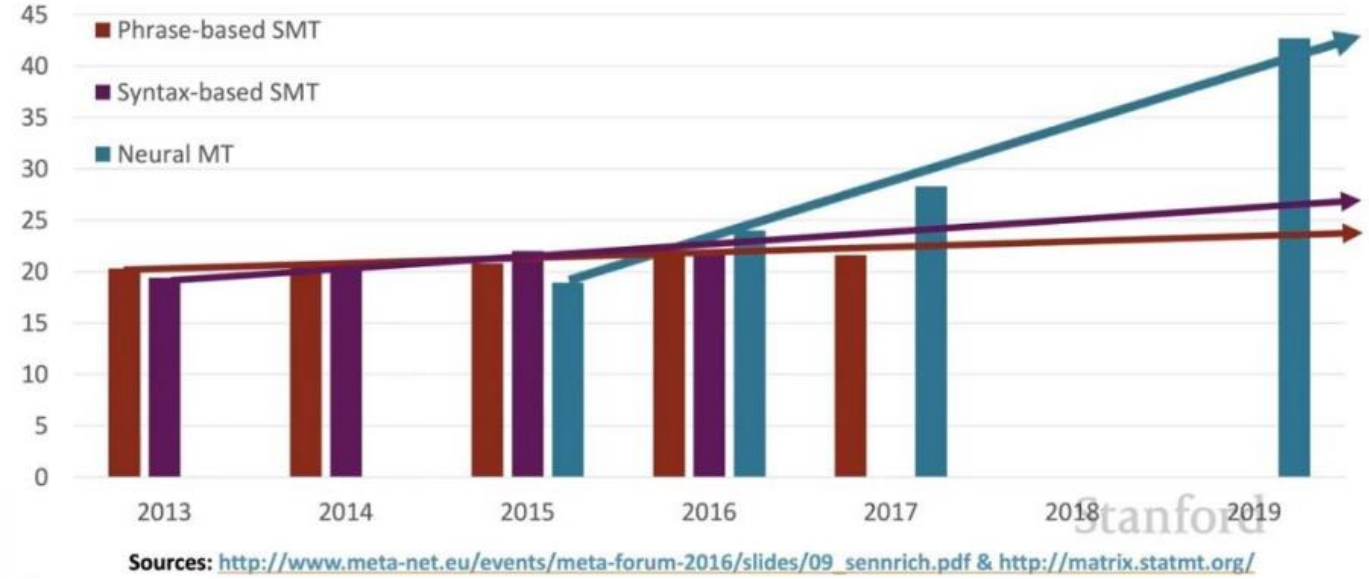
https://donghwa-kim.github.io/BLEU.html

- 1-gram precision: $\frac{\text{일치하는}1-gram\text{의 수(예측된 }sentence\text{중에서)}}{\text{모든}1-gram\text{쌍 (예측된 }sentence\text{중에서)}} = \frac{10}{14}$
- 2-gram precision: $\frac{\text{일치하는}2-gram\text{의 수(예측된 }sentence\text{중에서)}}{\text{모든}2-gram\text{쌍 (예측된 }sentence\text{중에서)}} = \frac{5}{13}$
- 3-gram precision: $\frac{\text{일치하는}3-gram\text{의 수(예측된 }sentence\text{중에서)}}{\text{모든}3-gram\text{쌍 (예측된 }sentence\text{중에서)}} = \frac{2}{12}$
- 4-gram precision: $\frac{\text{일치하는}4-gram\text{의 수(예측된 }sentence\text{중에서)}}{\text{모든}4-gram\text{쌍 (예측된 }sentence\text{중에서)}} = \frac{1}{11}$

$$(\prod_{i=1}^{4} precision_i)^{\frac{1}{4}} = (\frac{10}{14} \times \frac{5}{13} \times \frac{2}{12} \times \frac{1}{11})^{\frac{1}{4}}$$

2. Sequence-to-sequence (Seq2Seq)

Evaluation for Machine Translation : BLEU (BiLingual Evaluation Understudy)

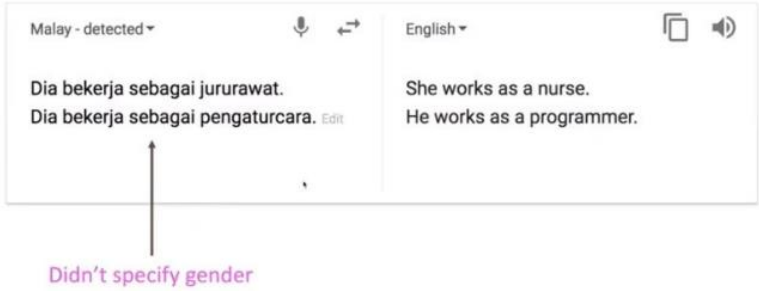


2. Sequence-to-sequence (Seq2Seq)

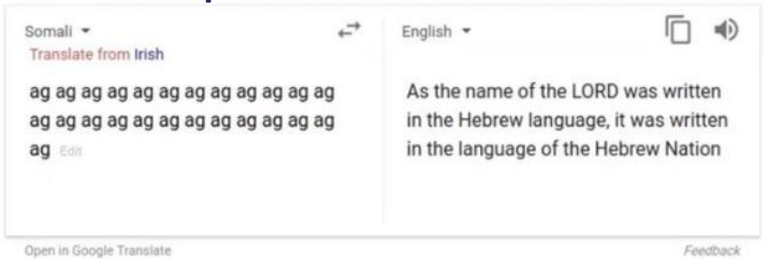
Remained difficulties

- 관용 표현

- Bias 학습

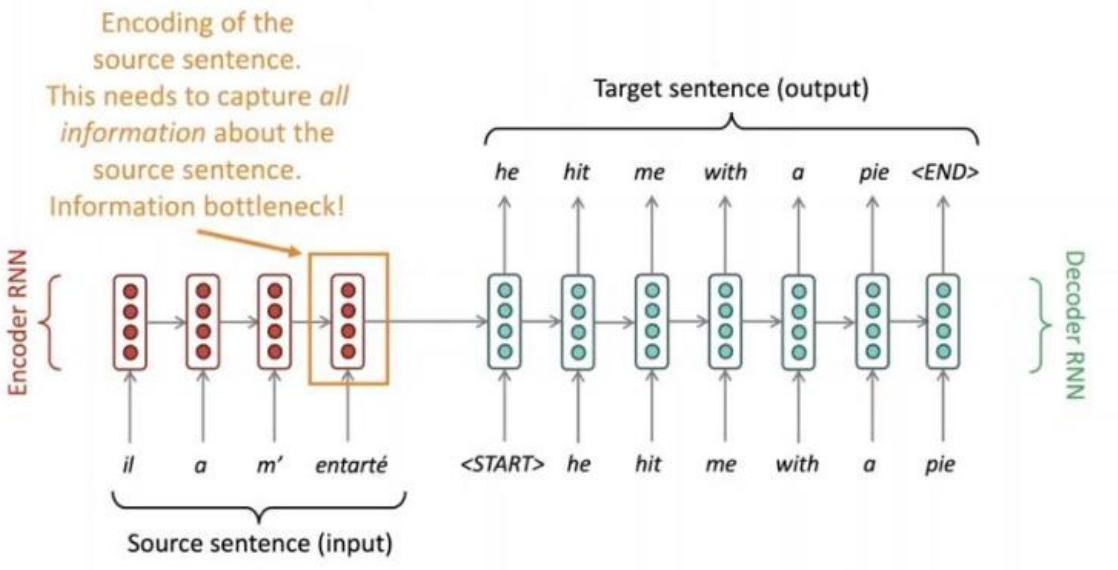


- Uninterpretable



3. Attention

Bottleneck Problem



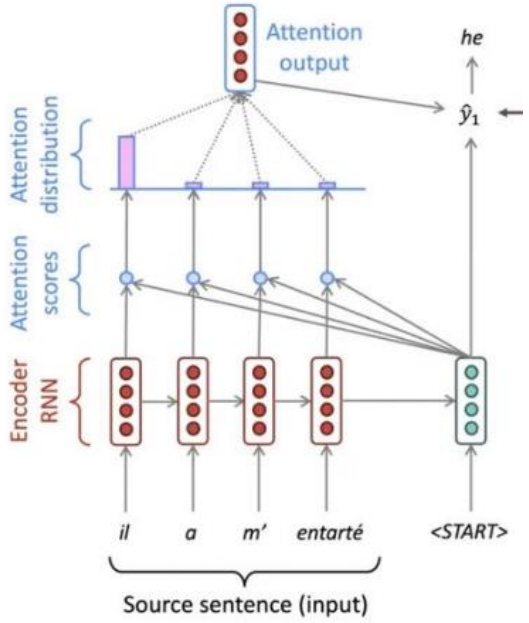
Source sentence의 마지막 encoding만을 decoder의 입력으로 사용

→ Encoding이 source sentence에 필요한 모든 정보 담고 있어야 함

→ Bottleneck Problem!

3. Attention

- Bottleneck problem 해결하기 위해 제안됨
- Decoder의 각 step에서 source sentence의 특정 부분에 집중할 수 있도록 connection 추가



감사합니다☺