Paper Review:
Language Representation with Neural Networks (Many Legends)

Presentation: Jeiyoon Park
NMT Team / Research Engineer

LLSOLLU

Plan for Today

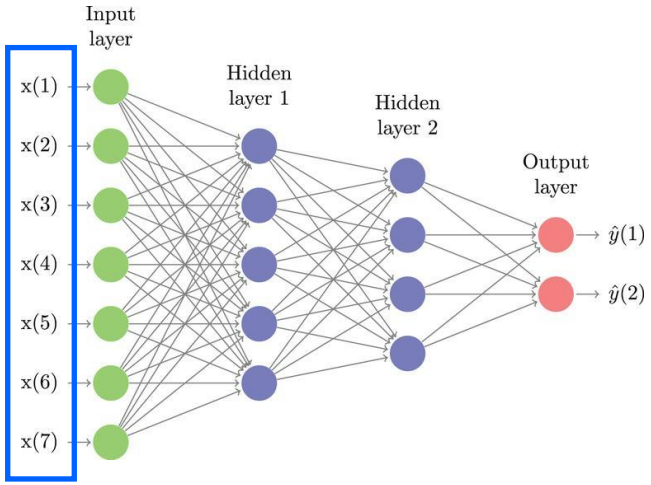
1. Neural Networks?
2. Part 1
3. Transformer
4. Part 2
5. Conclusion

Plan for Today

1. Neural Networks?
2. Part 1
3. Transformer
4. Part 2
5. Conclusion

Detour: Neural Networks (NNs)

- 딥러닝: “데이터” & “모델(NNs)” & “손실함수”



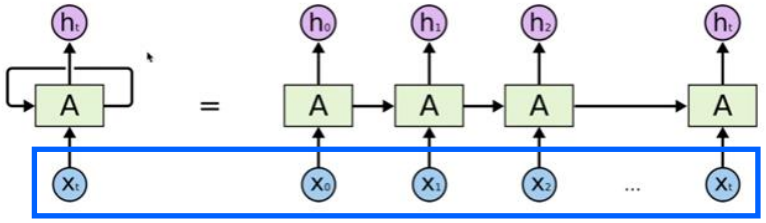
Model

Input data

Loss function

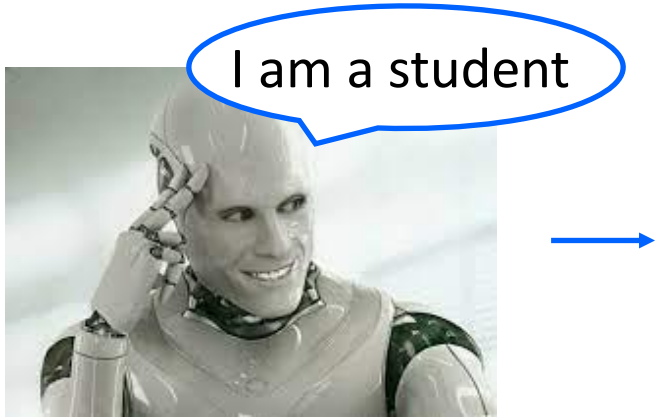
Detour: Recurrent Neural Networks (RNNs)

- RNNs: Hidden 노드가 순환구조를 이루는 신경망



"저는 학생 입니다"

(sequential)
Input data



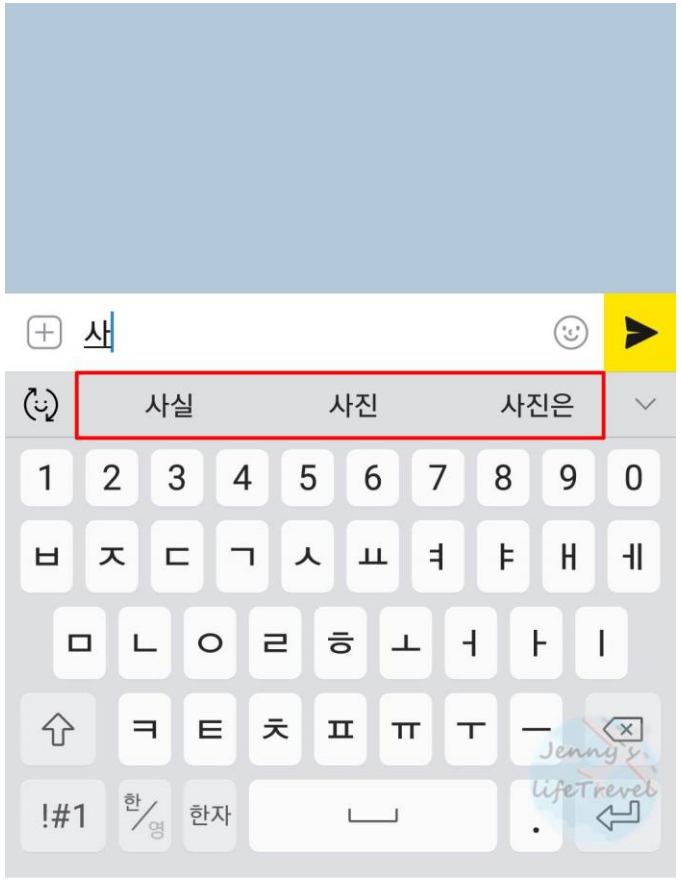
Model

Loss function

Detour: Recurrent Neural Networks (RNNs)

- Language Model (LM): 이전 단어들로 다음 단어를 예측

Input output

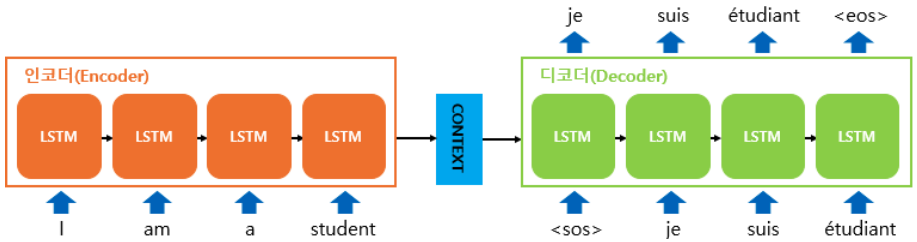


Plan for Today

1. Neural Networks?
2. Part 1
3. Transformer
4. Part 2
5. Conclusion

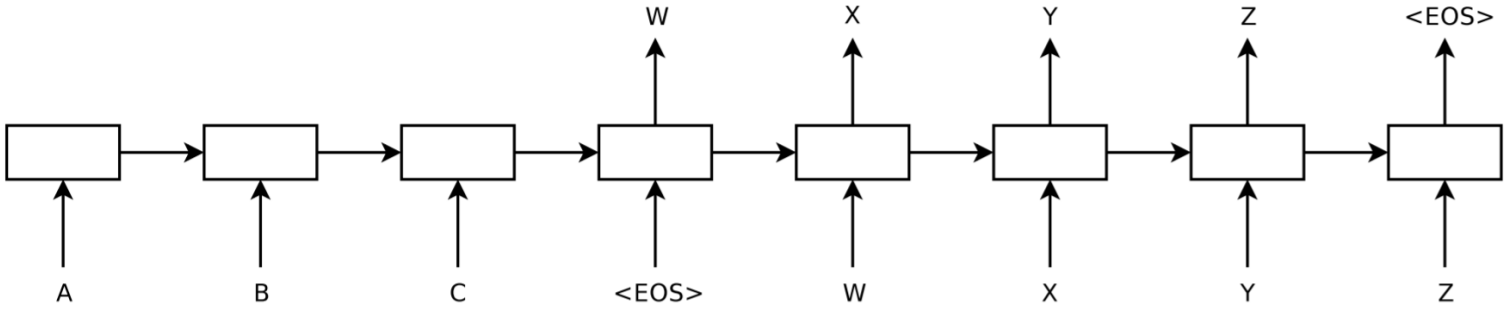
https://papers.nips.cc/paper/2014/file/a14ac55a4f27472c5d894ec1c3c743d2-Paper.pdf

1. Sequence to Sequence Learning with Neural Networks (Sutskever et al., NIPS 2014)



1) Contribution

- Two different LSTMS(RNN): 하나는 input data에, 다른 하나는 output data에
- LSTM with four layers & fixed length context vector
- 입력 문장의 순서를 뒤집어서 사용하면 성능이 오르는걸 발견함
(즉, “student a am I” 와 “je suis etudiant”를 매핑)



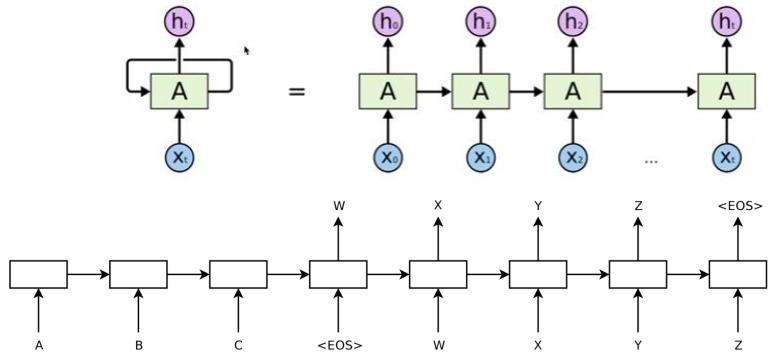
ABC: an input sentence

WXYZ: the output sentence

<EOS>: “end-of-sentence token”

https://papers.nips.cc/paper/2014/file/a14ac55a4f27472c5d894ec1c3c743d2-Paper.pdf

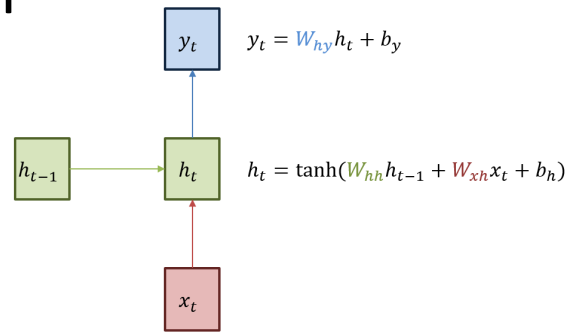
1. Sequence to Sequence Learning with Neural Networks (Sutskever et al., NIPS 2014)



2) Notations

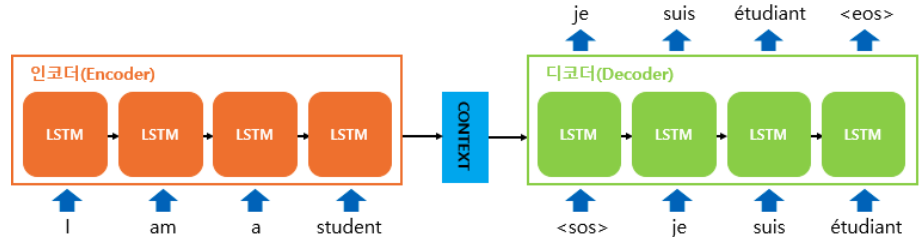
- Given a sentence of inputs $(x_1, \ldots, x_T)$, RNN computes a sequence of outputs $(y_1, \ldots, y_{T'})$, where T *and* T': length

$$h_t = \text{sigm}\left(W^{\text{hx}} x_t + W^{\text{hh}} h_{t-1}\right)$$
$$y_t = W^{\text{yh}} h_t$$



$$p(y_1, \ldots, y_{T'} | x_1, \ldots, x_T) = \prod_{t=1}^{T'} p(y_t | v, y_1, \ldots, y_{t-1})$$

goal — context (fixed length)

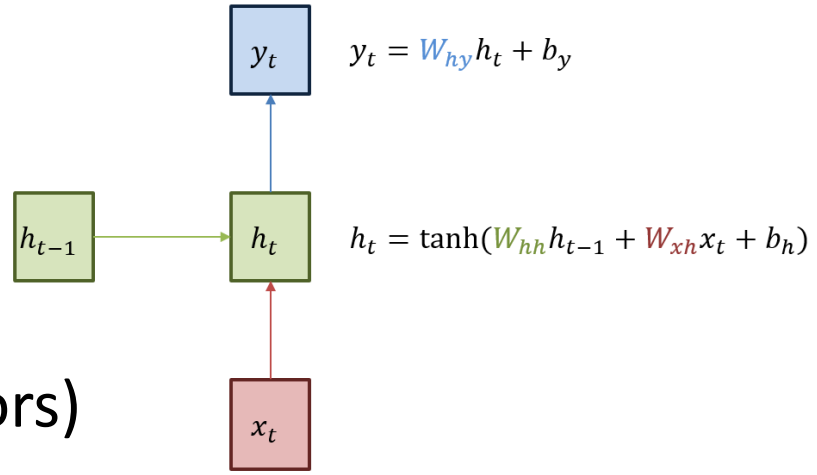


https://arxiv.org/pdf/1409.0473.pdf

2. Neural Machine Translation by Jointly Learning To Align and Translate (Bahdanau et al., ICLR 2015)

1) Contribution

- Encoder: “양방향” RNN. 왜 why? 양방향 context를 반영하기 위해서
- Decoder: “Alignment model”
- Fixed-length vector



2) Notations

- $x = (x_1, \dots x_{T_x})$: input sentence (a sequence of vectors)
- $h_t = f(x_t, h_{t-1})$: hidden state
- $c = q(\{h_1, \dots, h_{T_x}\})$: context vector
- q: activation function (nonlinear)

https://arxiv.org/pdf/1409.0473.pdf

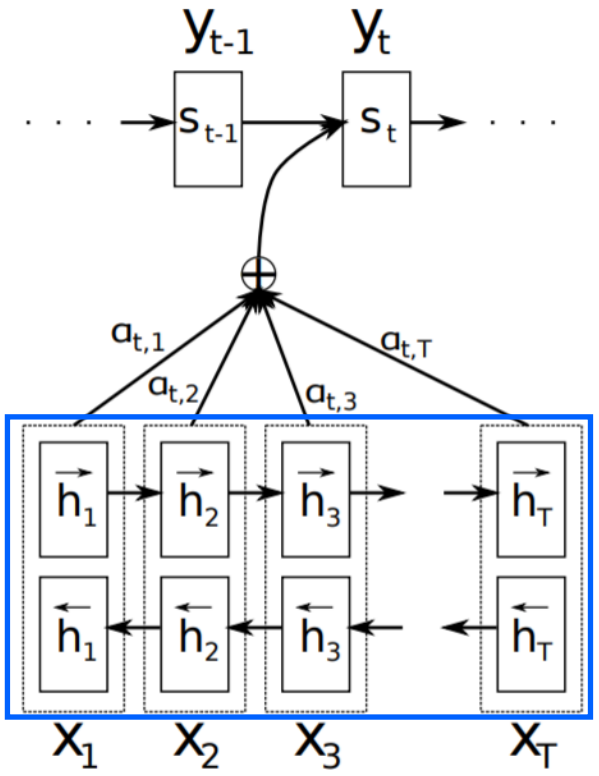
2. Neural Machine Translation by Jointly Learning To Align and Translate (Bahdanau et al., ICLR 2015)

3) Encoder

- Forward RNN $\vec{f} = (\overrightarrow{h_1}, \ldots, \overrightarrow{h_{T_x}})$

- Backward RNN $\overleftarrow{f} = (\overleftarrow{h_1}, \ldots, \overleftarrow{h_{T_x}})$

- $h_j = [\overrightarrow{h_j^T}; \overleftarrow{h_j^T}]$



https://arxiv.org/pdf/1409.0473.pdf

2. Neural Machine Translation by Jointly Learning To Align and Translate (Bahdanau et al., ICLR 2015)

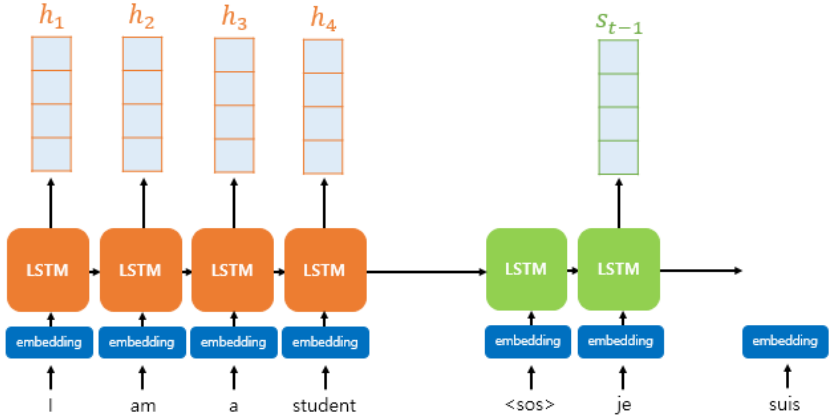
4) Decoder

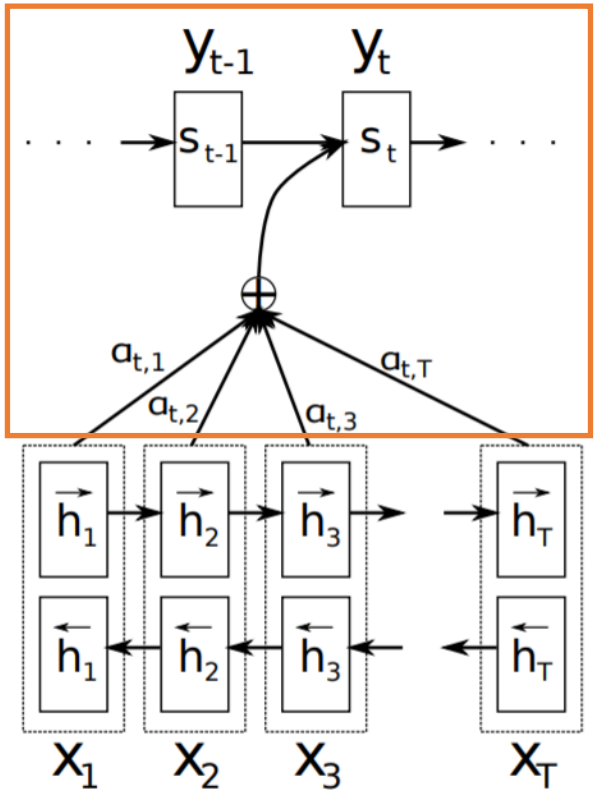
$$e_{ij} = a(s_{i-1}, h_j) \longleftarrow h_j = \left[\overrightarrow{h}_j^\top ; \overleftarrow{h}_j^\top\right]$$

Feed forward network

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$$

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$





$$s_i = f(s_{i-1}, y_{i-1}, c_i) \longrightarrow p(y_i | y_1, \ldots, y_{i-1}, \mathbf{x}) = g(y_{i-1}, s_i, c_i)$$

https://arxiv.org/pdf/1508.04025.pdf

3. Effective Approaches to Attention-based Neural Machine Translation (Minh et al., EMNLP 2015)

1) Contribution

- Attention 구조에 대한 연구
- Global approach: It always attends to all source words
- Local approach: It looks at a subset of source word at time

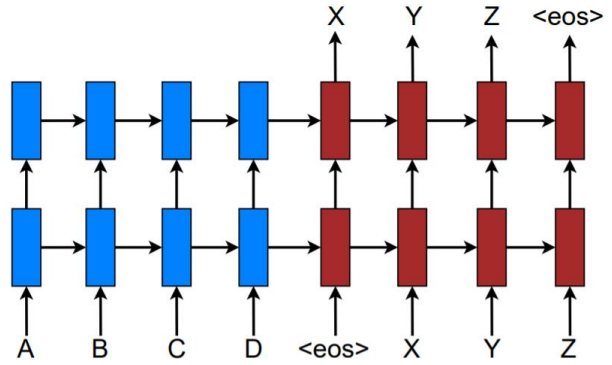


Figure 1: **Neural machine translation** – a stacking recurrent architecture for translating a source sequence A B C D into a target sequence X Y Z. Here, <eos> marks the end of a sentence.

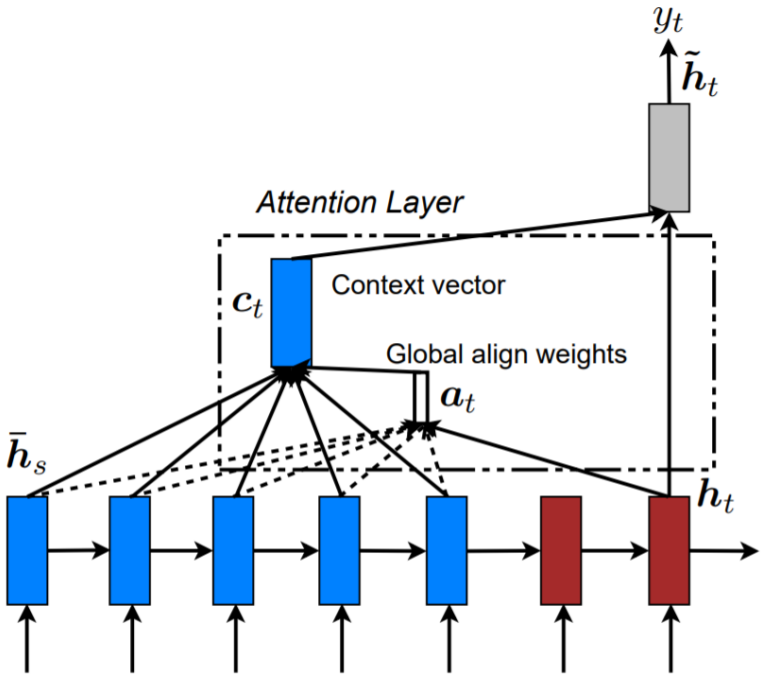
https://arxiv.org/pdf/1508.04025.pdf

3. Effective Approaches to Attention-based Neural Machine Translation (Minh et al., EMNLP 2015)

2) Global approach: alignment weight vector a_t 를 만들때 encode의 모든 hidden state 들을 고려함 (h_t: target hidden, $\overline{h_s}$: source hidden)

$$\boldsymbol{a}_t(s) = \text{align}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)$$
$$= \frac{\exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)\right)}{\sum_{s'} \exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_{s'})\right)}$$

$$\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \bar{\boldsymbol{h}}_s & \textit{dot} \\ \boldsymbol{h}_t^\top \boldsymbol{W}_{\boldsymbol{a}} \bar{\boldsymbol{h}}_s & \textit{general} \\ \boldsymbol{v}_a^\top \tanh\left(\boldsymbol{W}_{\boldsymbol{a}}[\boldsymbol{h}_t; \bar{\boldsymbol{h}}_s]\right) & \textit{concat} \end{cases}$$



https://arxiv.org/pdf/1508.04025.pdf

3. Effective Approaches to Attention-based Neural Machine Translation (Minh et al., EMNLP 2015)

3) Local approach: target word마다 source의 subset을 사용함

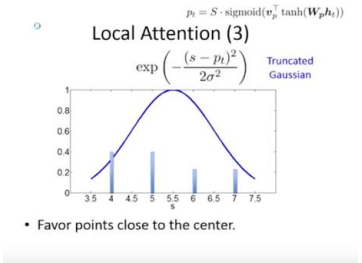
- Global approach의 단점: 하나의 target word마다 모든 source side word들을 attend함 (비싸면서 long sequence에 취약함)

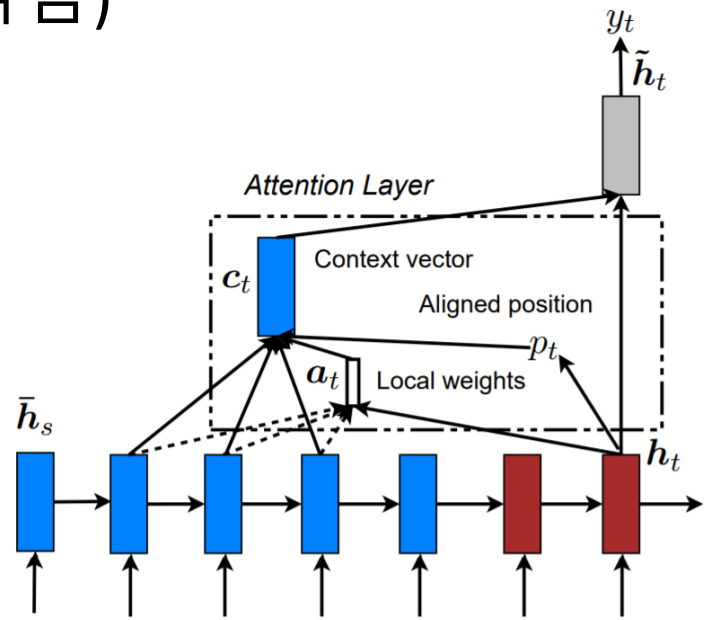
$$p_t = S \cdot \text{sigmoid}(\boldsymbol{v}_p^\top \tanh(\boldsymbol{W}_p \boldsymbol{h}_t))$$

Aligned position for target word → Encoder의 hidden들 중 window 만큼 고려해서 context vector c_t 생성

$[p_t - D, p_t + D]$; D is empirically selected.

$$\boldsymbol{a}_t(s) = \text{align}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) \exp\left(-\frac{(s - p_t)^2}{2\sigma^2}\right)$$





4. 그 외에 읽어보면 좋은 논문들

- Guided Alignment Training for Topic-Aware Neural Machine Translation (Chen et al. 2016)
- Linguistic Input Features Improve Neural Machine Translation (Sennrich et al. 2016)
- Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation (Wu et al. 2016)
- A Convolutional Encoder Model for Neural Machine Translation (Gehring et al. 2016)
- MS-UEdin Submission to the WMT2018 APE Shared Task: Dual-Source Transformer for Automatic Post-Editing (Junczys-Dowmunt et al. 2018)
- Scaling Neural Machine Translation (Ott et al. 2018)
- The Best of Both Worlds: Combining Recent Advances in Neural Machine Translation (Chen et al. 2018)
- Self-Attention with Relative Position Representations (Shaw et al. 2018)

Plan for Today

1. Neural Networks?
2. Part 1
3. Transformer
4. Part 2
5. Conclusion

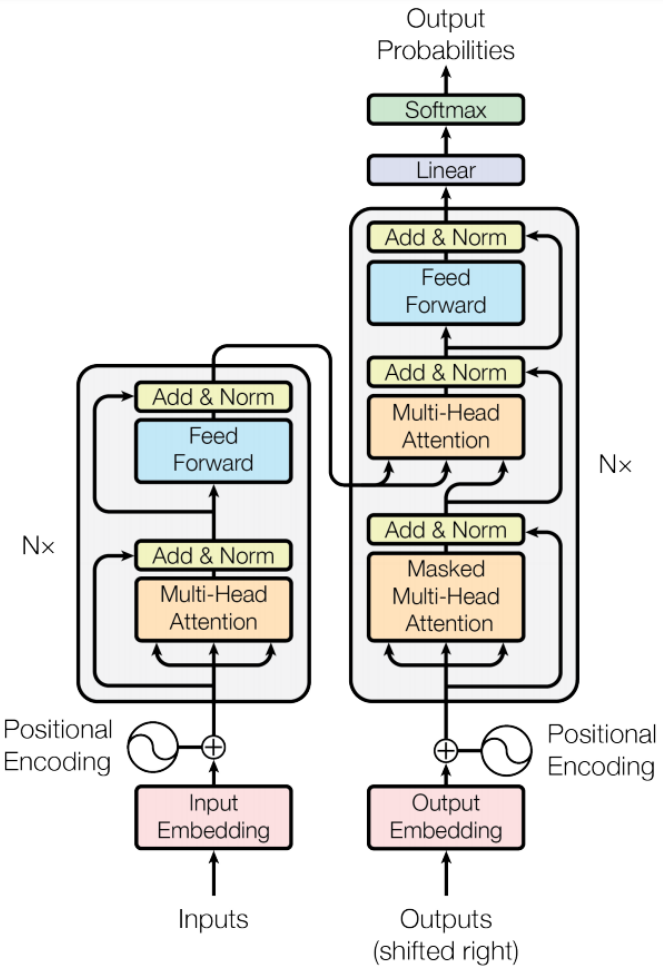
https://arxiv.org/pdf/1706.03762.pdf
http://jalammar.github.io/illustrated-transformer/

1. Attention is all you need (Vaswani et al., NIPS 2017)

1) Contribution

- ~~존재 그 자체~~ 복잡한 CNN or RNN가 없는 구조
- Encoder & Decoder

2) Transformer: A high-level look

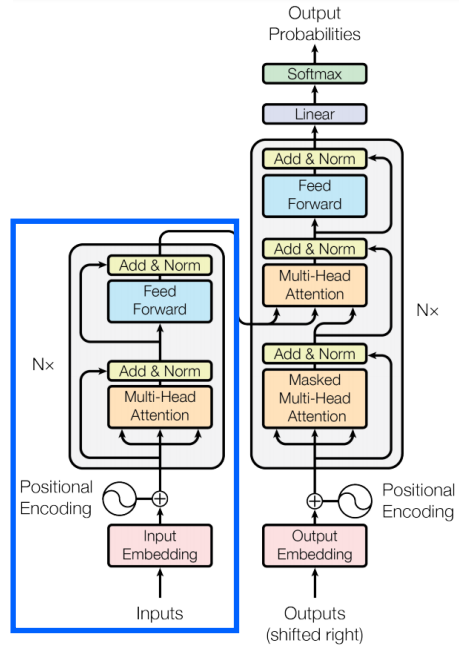


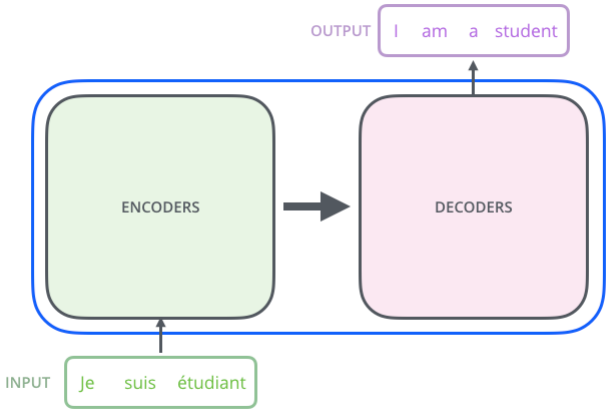
https://arxiv.org/pdf/1706.03762.pdf
http://jalammar.github.io/illustrated-transformer/

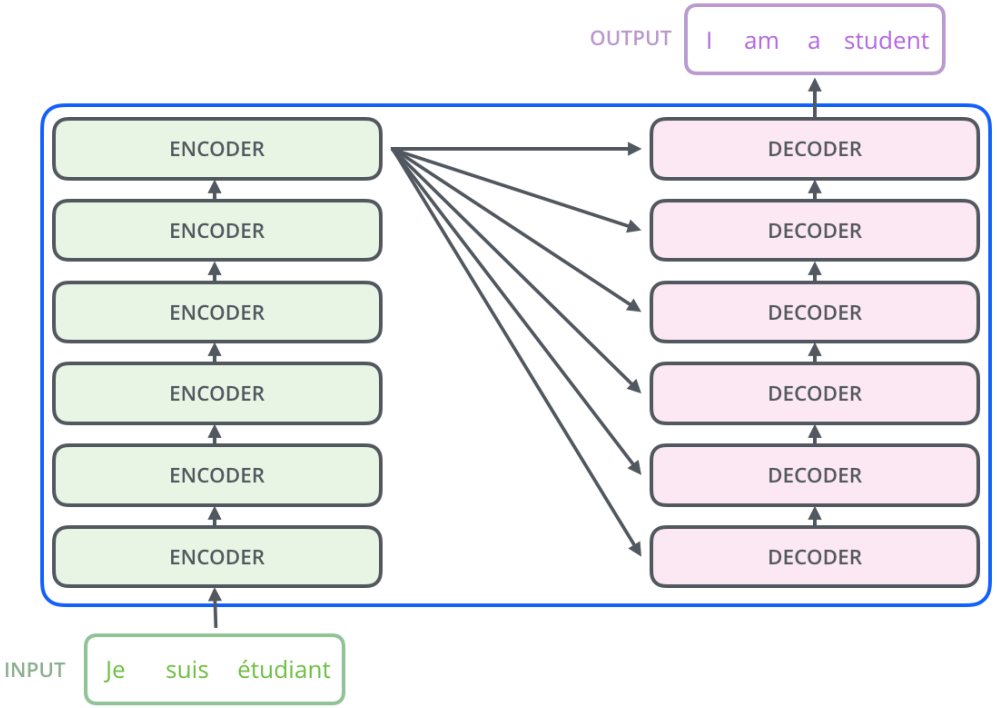
1. Attention is all you need (Vaswani et al., NIPS 2017)

2) Transformer: A high-level look

- 기존 방법들과 같이 encoder & decoder 구조를 가짐
- 각 encoder와 decoder는 6개가 stacked되어 사용됨

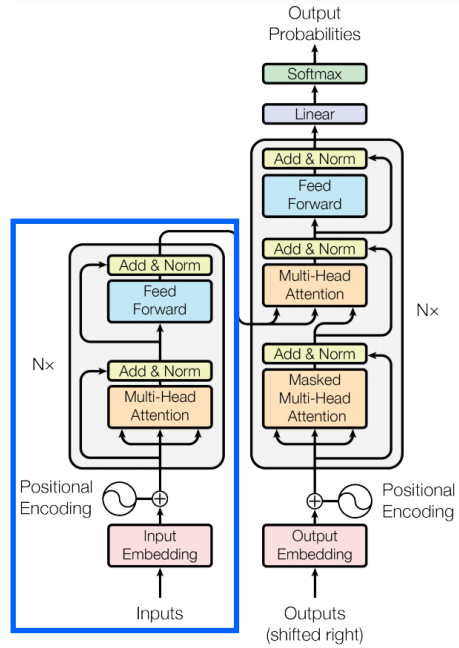






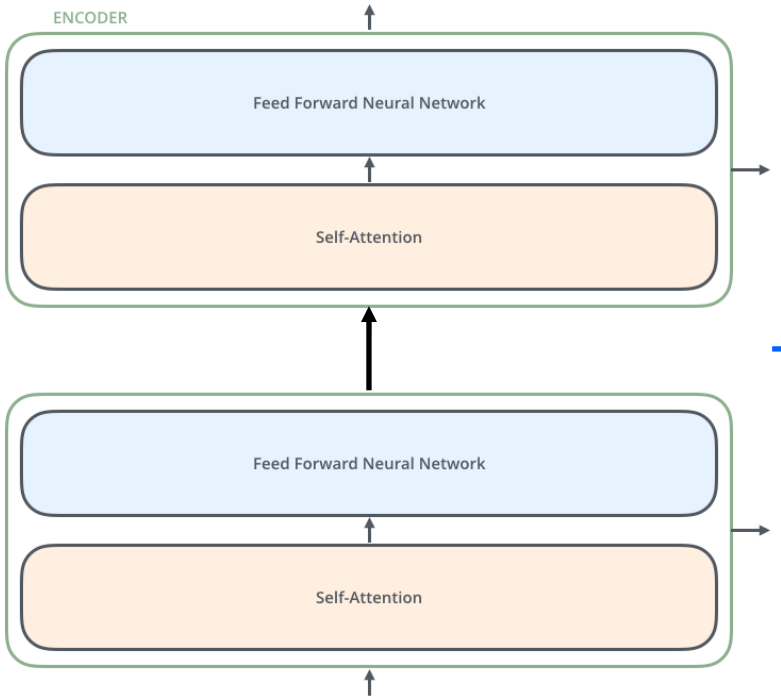
https://arxiv.org/pdf/1706.03762.pdf
http://jalammar.github.io/illustrated-transformer/

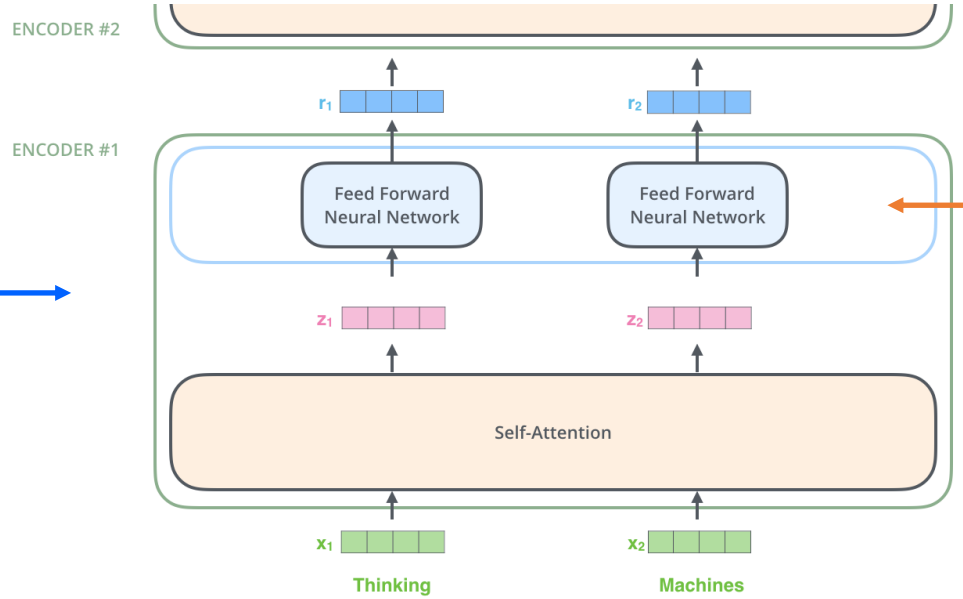
1. Attention is all you need (Vaswani et al., NIPS 2017)



3) Encoder

- Encoder는 Self-attention layer와 FFN layer로 구성됨
- Self-attention layer의 output z가 FFN의 input으로 사용됨

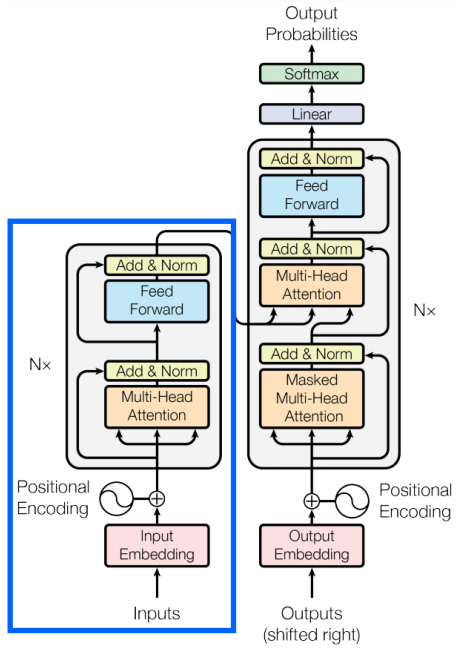




The exact same network with each vector flowing through it separately!

https://arxiv.org/pdf/1706.03762.pdf
http://jalammar.github.io/illustrated-transformer/

1. Attention is all you need (Vaswani et al., NIPS 2017)



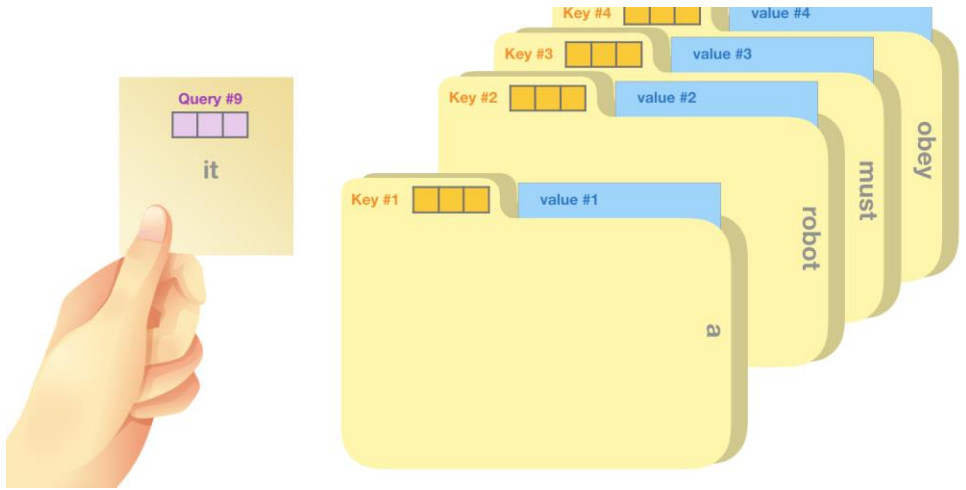
3) Encoder: Self-Attention

- 우선 input을 세가지 vector형태로 나타냄(Q, K, V)

(1) Query: 현재 단어 (다른 단어(Key)들과 score를 계산하는데 사용)
(2) Key: 모든 단어의 label 역할
(3) Value: 실제 word representation

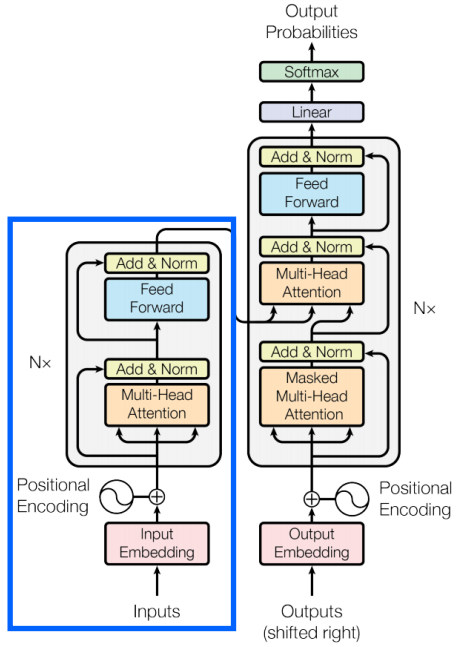
e.g.) <s> a robot must obey the orders given it

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



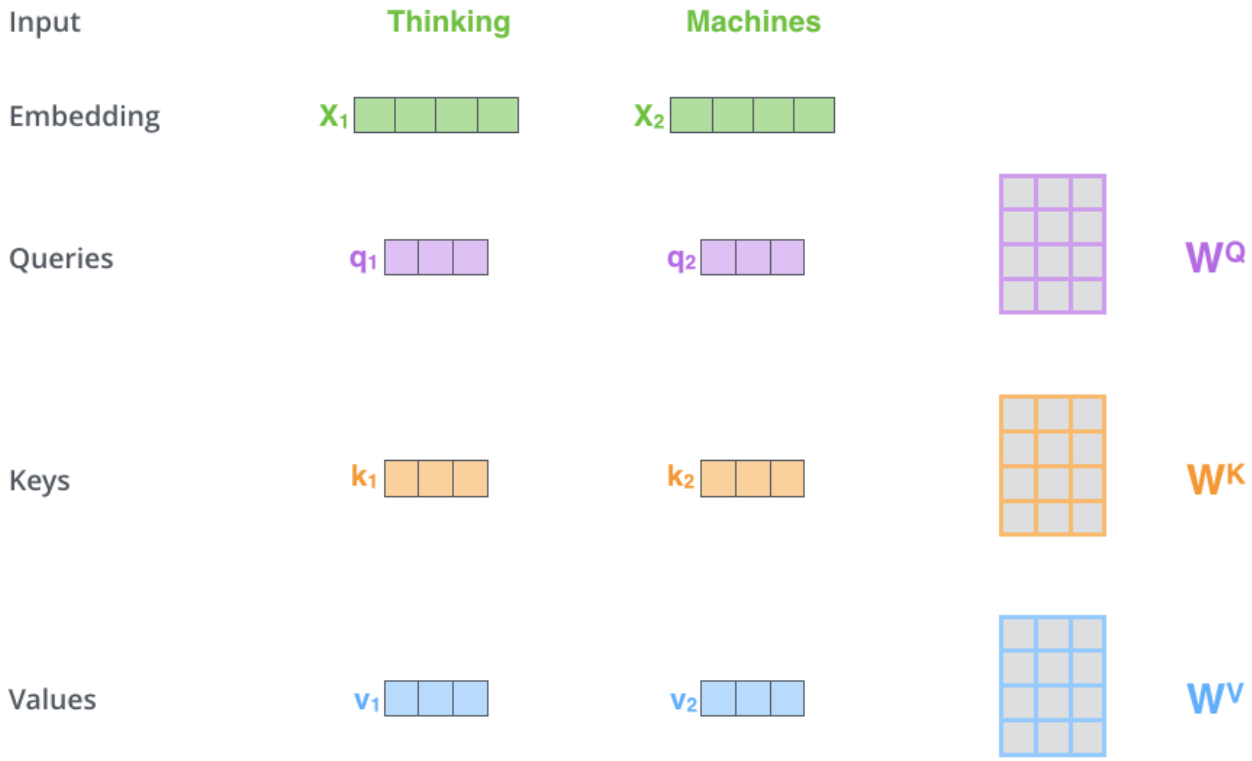
https://arxiv.org/pdf/1706.03762.pdf
http://jalammar.github.io/illustrated-transformer/

1. Attention is all you need (Vaswani et al., NIPS 2017)



3) Encoder: Self-Attention

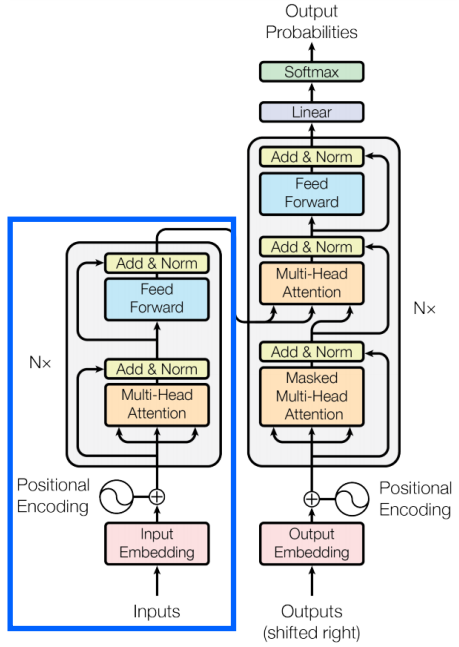
- 우선 input을 세가지 vector형태로 나타냄(Q, K, V)



$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

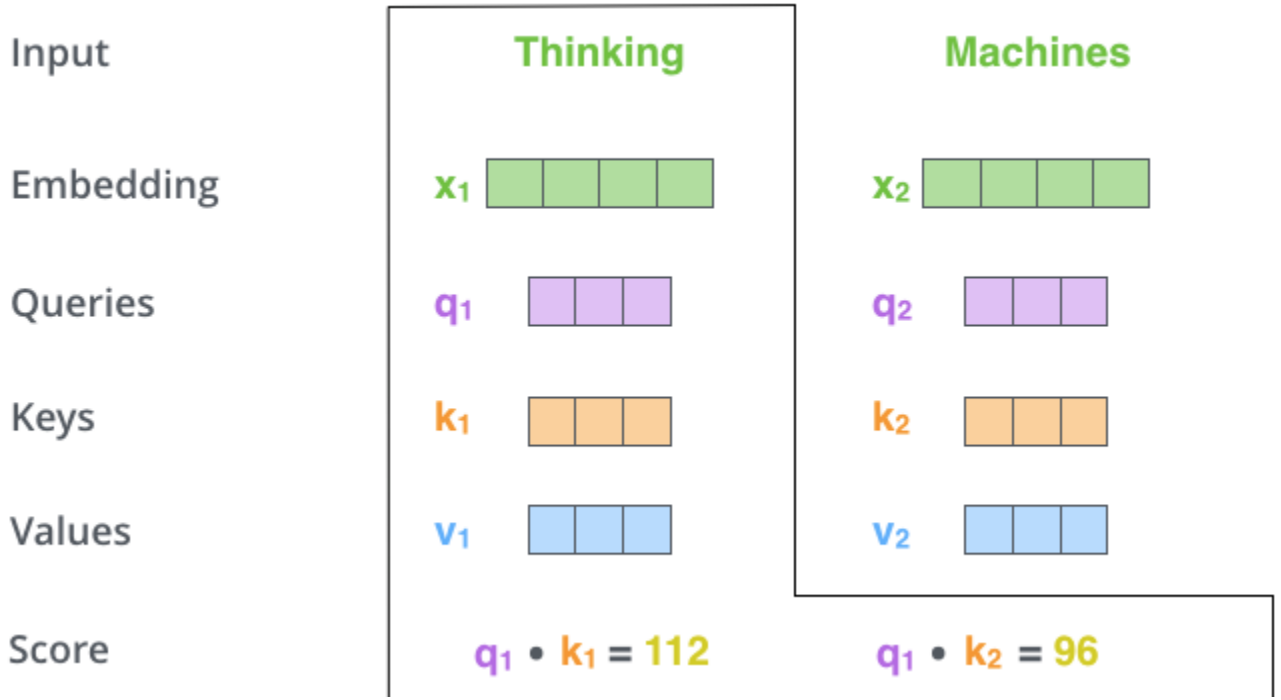
https://arxiv.org/pdf/1706.03762.pdf
http://jalammar.github.io/illustrated-transformer/

1. Attention is all you need (Vaswani et al., NIPS 2017)



3) Encoder: Self-Attention

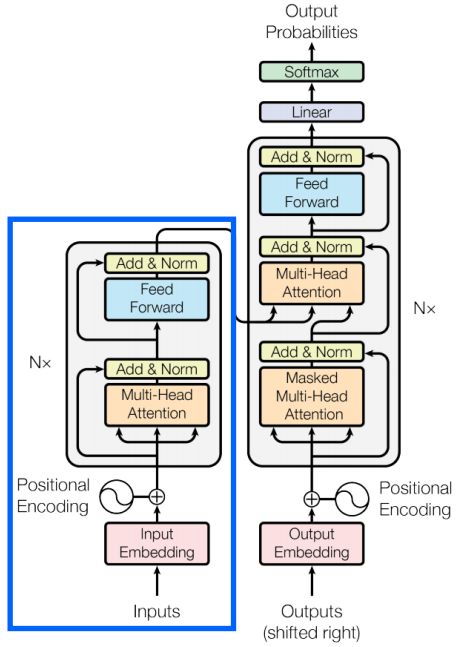
- Q와 K를 곱해서 attention score를 계산함



$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

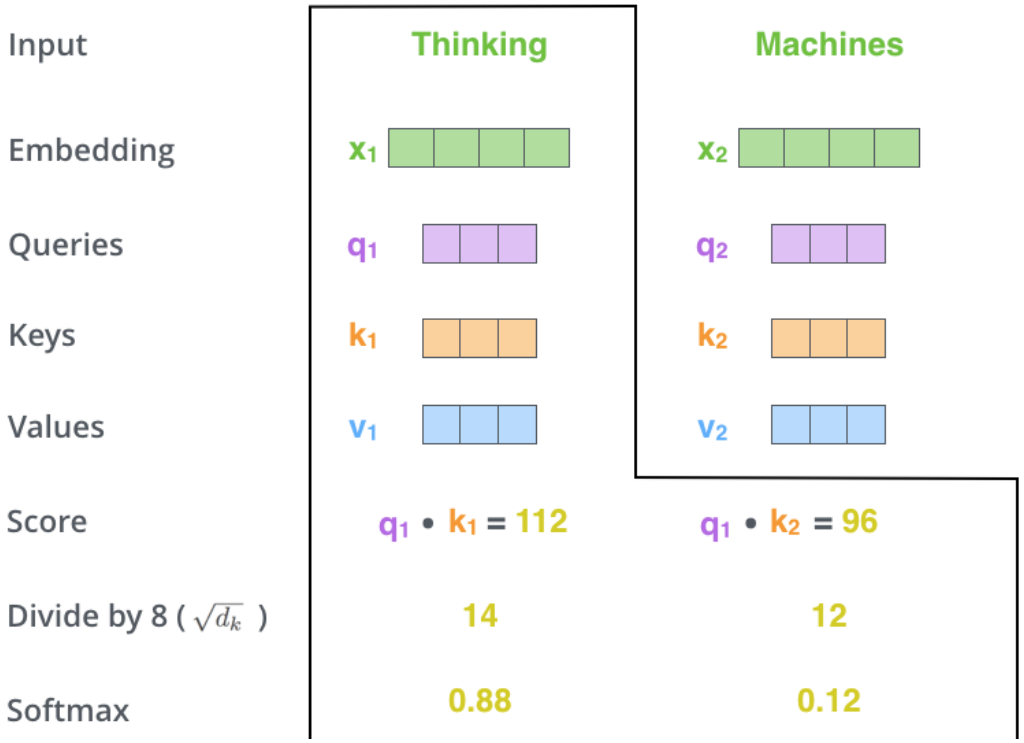
https://arxiv.org/pdf/1706.03762.pdf
http://jalammar.github.io/illustrated-transformer/

1. Attention is all you need (Vaswani et al., NIPS 2017)



3) Encoder: Self-Attention

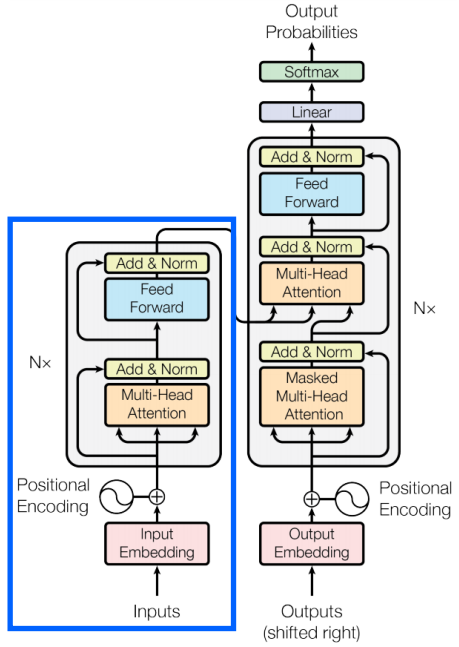
- K vector의 dimension의 square root로 나눠줌
- 학습시 gradient의 안정성 증가



$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

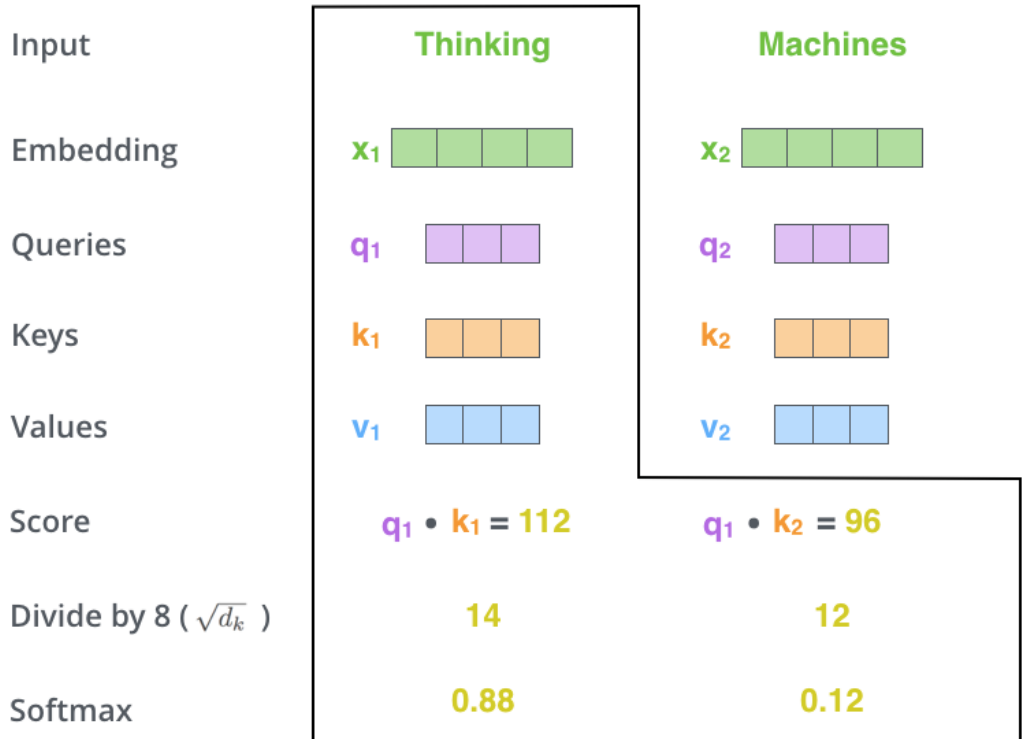
https://arxiv.org/pdf/1706.03762.pdf
http://jalammar.github.io/illustrated-transformer/

1. Attention is all you need (Vaswani et al., NIPS 2017)



3) Encoder: Self-Attention

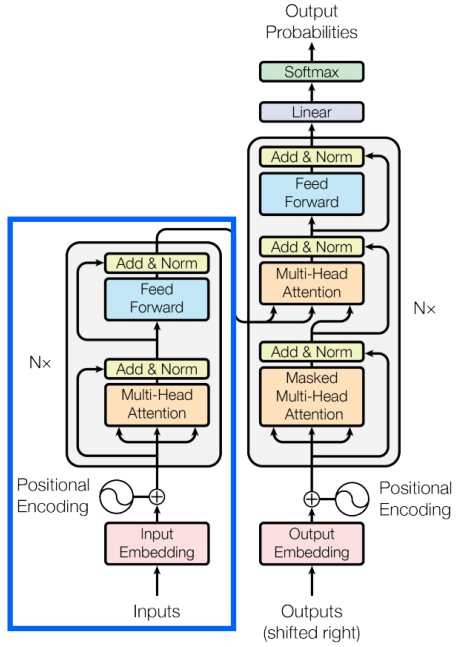
- Softmax함수로 score를 normalize 해줌 (합 = 1)
- score에 V를 곱해줌 (각각의 단어를 intact하게 유지하겠다)



$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

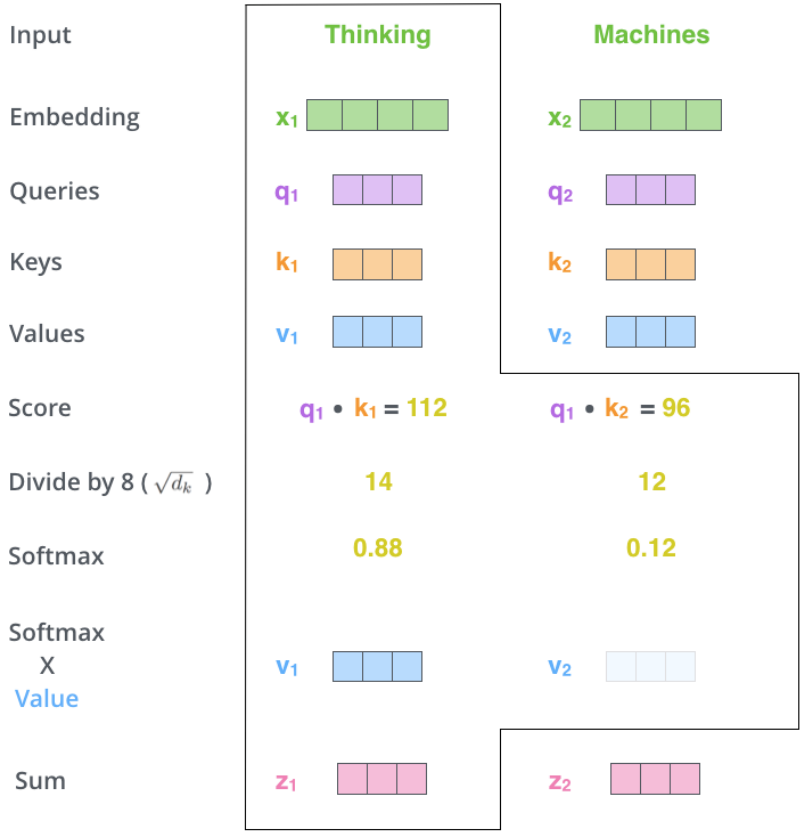
https://arxiv.org/pdf/1706.03762.pdf
http://jalammar.github.io/illustrated-transformer/

1. Attention is all you need (Vaswani et al., NIPS 2017)

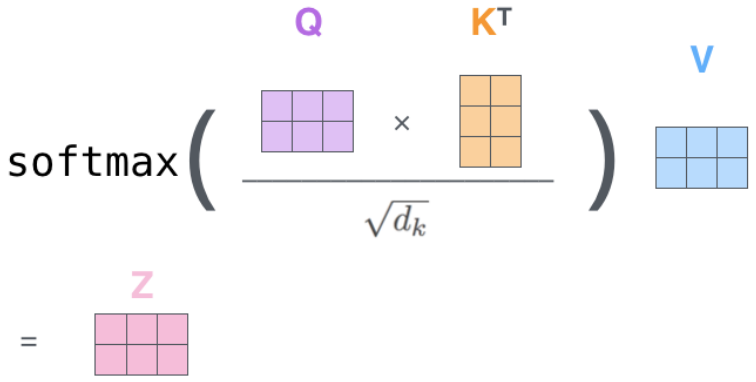


3) Encoder: Self-Attention

- Weighted vector들의 합 z를 구함

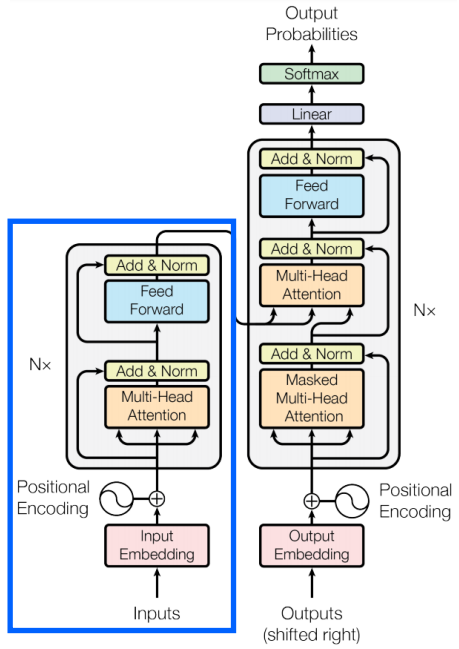


$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



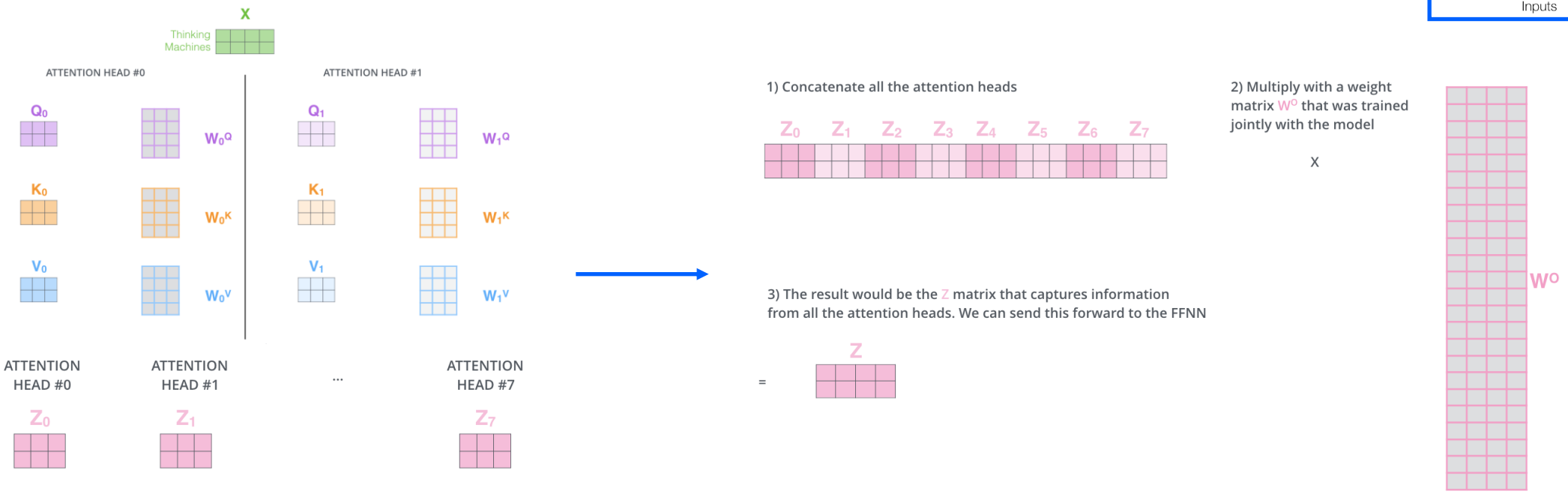
https://arxiv.org/pdf/1706.03762.pdf
http://jalammar.github.io/illustrated-transformer/

1. Attention is all you need (Vaswani et al., NIPS 2017)



3) Encoder: Attention with Multi-heads

- 모델이 다양한 측면을 보게해서 모델의 성능을 올림

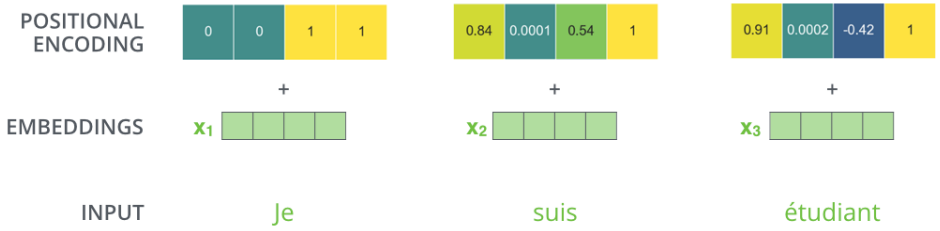


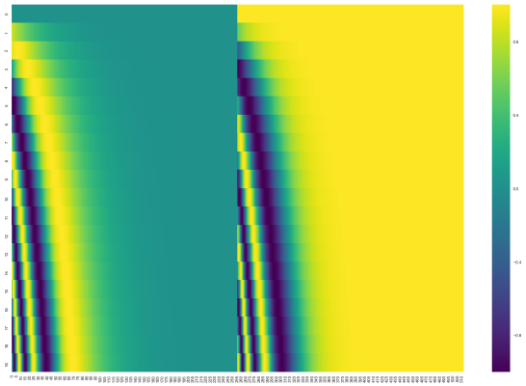
https://arxiv.org/pdf/1706.03762.pdf
http://jalammar.github.io/illustrated-transformer/

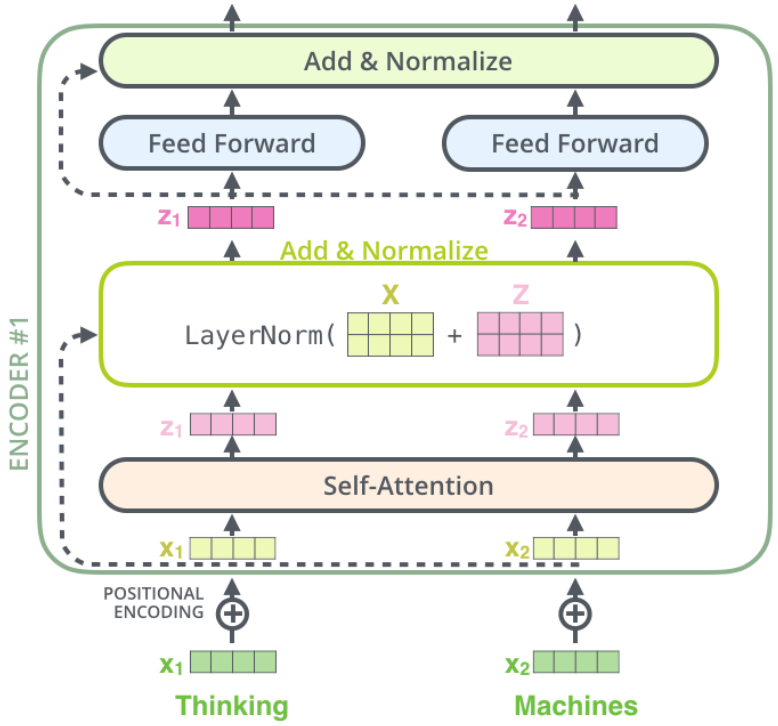
1. Attention is all you need (Vaswani et al., NIPS 2017)

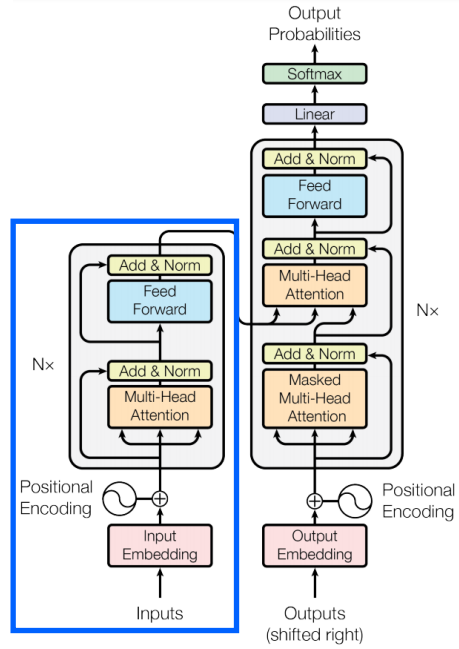
3) Encoder: Positional Encoding & Residuals

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$





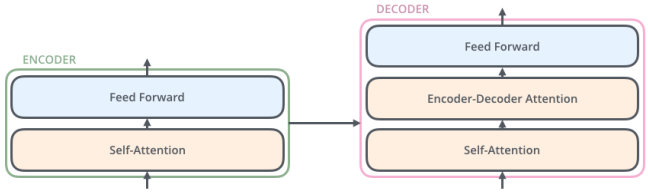


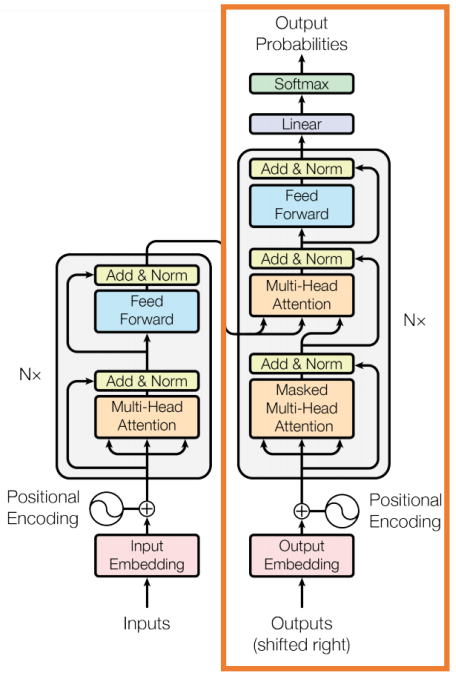


https://arxiv.org/pdf/1706.03762.pdf
http://jalammar.github.io/illustrated-transformer/

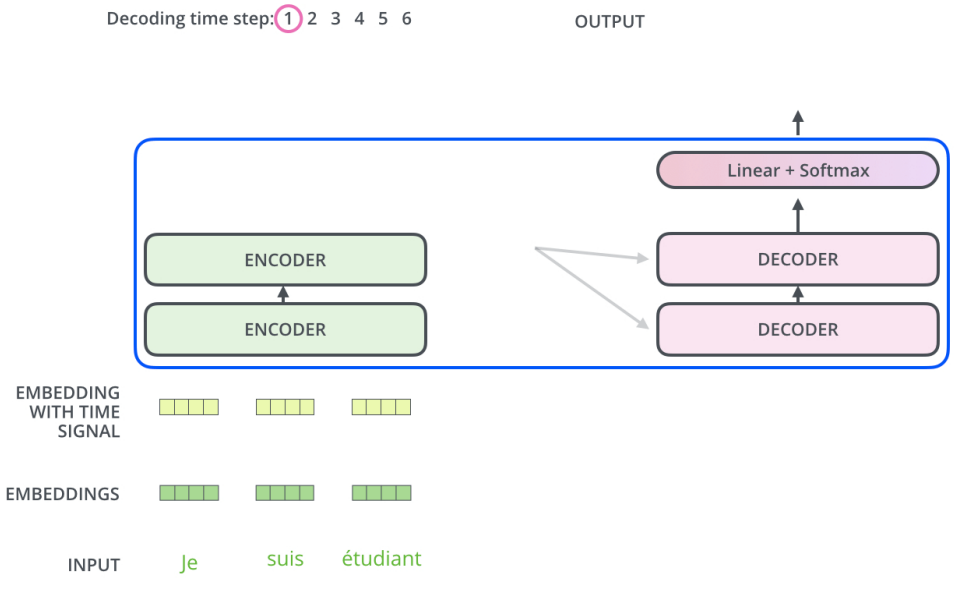
1. Attention is all you need (Vaswani et al., NIPS 2017)

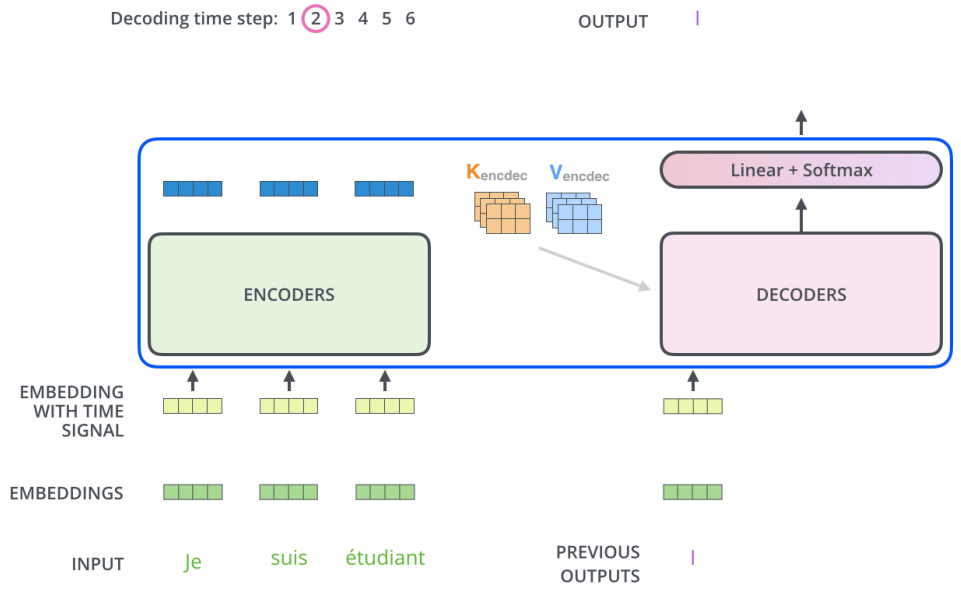
4) Decoder





- Encoder + 가운데에 Encoder-Decoder Attention layer
- K, V를 사용해서 decoder가 적절한 input sequence를 학습하게함





Plan for Today

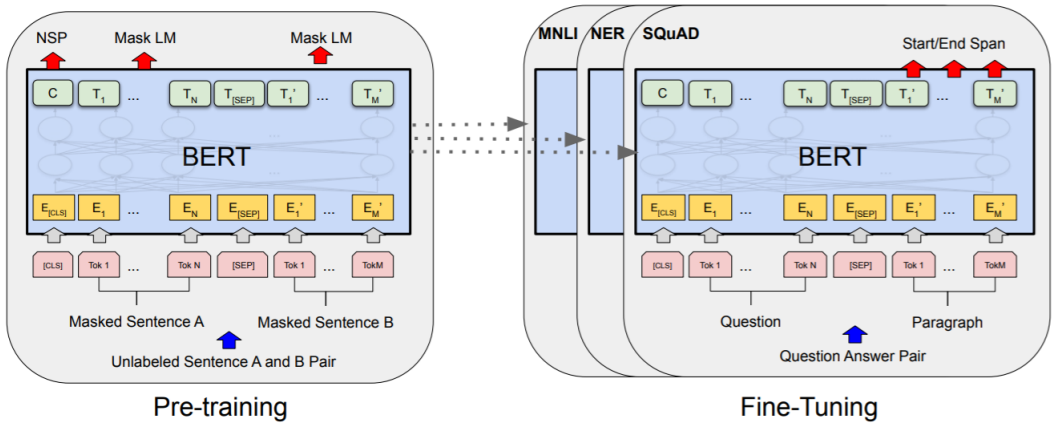
1. Neural Networks?
2. Part 1
3. Transformer
4. Part 2
5. Conclusion

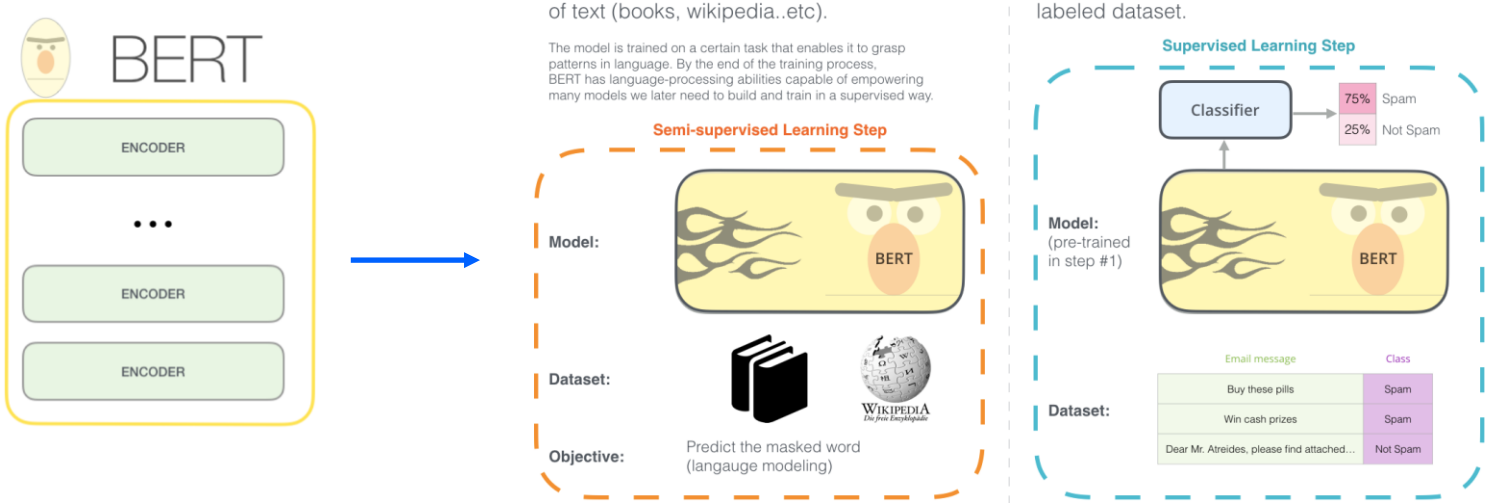
https://arxiv.org/pdf/1810.04805.pdf
http://jalammar.github.io/illustrated-bert/

1. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (Devlin et al., NACCL 2019)

1) Contribution

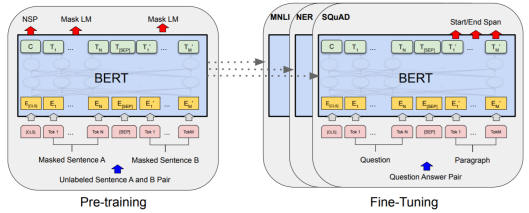
- Transformer encoder를 양방향으로 쌓음
- Pretrain & Fine-tuning 구조
- 각 task 마다 모델 디자인을 할 필요가 없음





https://arxiv.org/pdf/1810.04805.pdf
http://jalammar.github.io/illustrated-bert/

1. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (Devlin et al., NACCL 2019)



2) Pretrain

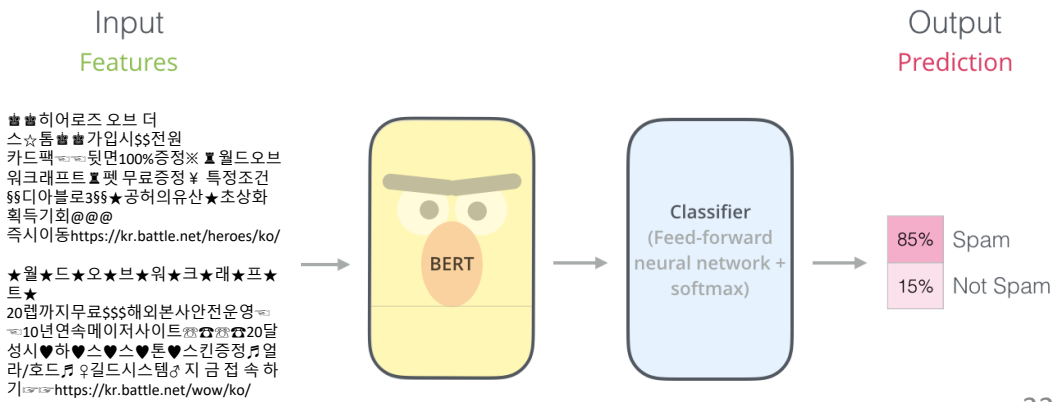
- Masked language model: 15% 확률로 랜덤하게 세가지 방식으로 mask를 씌우고 맞추는 방식으로 학습함

```
Input Sequence  : The man went to [MASK] store with [MASK] dog
Target Sequence :                  the               his
```

- Next sentence prediction: 각 샘플은 두 개의 문장으로 구성. 50% 확률로 문장 뒤의 문장을 다른 문장으로 바꾸고 맞추는 방식으로 학습함

```
Input = [CLS] the man went to [MASK] store [SEP]
        he bought a gallon [MASK] milk [SEP]
Label = IsNext

Input = [CLS] the man [MASK] to the store [SEP]
        penguin [MASK] are flight ##less birds [SEP]
Label = NotNext
```



- Masked LM loss와 NSP loss를 더해서 학습함

https://arxiv.org/pdf/1907.11692.pdf

2. RoBERTa: A Robustly Optimized BERT Pretraining Approach (Liu et al., arXiv 2019)

1) Contribution

- BERT의 단점은 a) 비싼 학습 cost, 그리고 b) hyperparameter에 따라 결과 차이가 매우 큼
- 다양한 실험을 통해 BERT가 완전하지 않음을 확인함

2) 더 많은 data, 더 큰 batch size: 적합한 양을 찾아냄

bsz	steps	lr	ppl	MNLI-m	SST-2
256	1M	1e-4	3.99	84.7	92.7
2K	125K	7e-4	**3.68**	**85.2**	**92.9**
8K	31K	1e-3	3.77	84.6	92.8

Table 3: Perplexity on held-out training data (*ppl*) and development set accuracy for base models trained over BOOKCORPUS and WIKIPEDIA with varying batch sizes (*bsz*). We tune the learning rate (*lr*) for each setting. Models make the same number of passes over the data (epochs) and have the same computational cost.

https://arxiv.org/pdf/1907.11692.pdf

2. RoBERTa: A Robustly Optimized BERT Pretraining Approach (Liu et al., arXiv 2019)

3) NSP task 없애봄: NSP loss 빼면 성능 올라감

Model	SQuAD 1.1/2.0	MNLI-m	SST-2	RACE
Our reimplementation (with NSP loss):				
SEGMENT-PAIR	90.4/78.7	84.0	92.9	64.2
SENTENCE-PAIR	88.7/76.2	82.9	92.1	63.0
Our reimplementation (without NSP loss):				
FULL-SENTENCES	90.4/79.1	84.7	92.5	64.8
DOC-SENTENCES	90.6/79.7	84.7	92.7	65.6
$\text{BERT}_{\text{BASE}}$	88.5/76.3	84.3	92.8	64.3
$\text{XLNet}_{\text{BASE}}$ (K = 7)	–/81.3	85.8	92.7	66.1
$\text{XLNet}_{\text{BASE}}$ (K = 6)	–/81.0	85.6	93.4	66.7

4) Masking pattern 다양하게 해봄: 살짝 좋음

Masking	SQuAD 2.0	MNLI-m	SST-2
reference	76.3	84.3	92.8
Our reimplementation:			
static	78.3	84.3	92.5
dynamic	78.7	84.0	92.9

https://arxiv.org/pdf/1910.13461.pdf

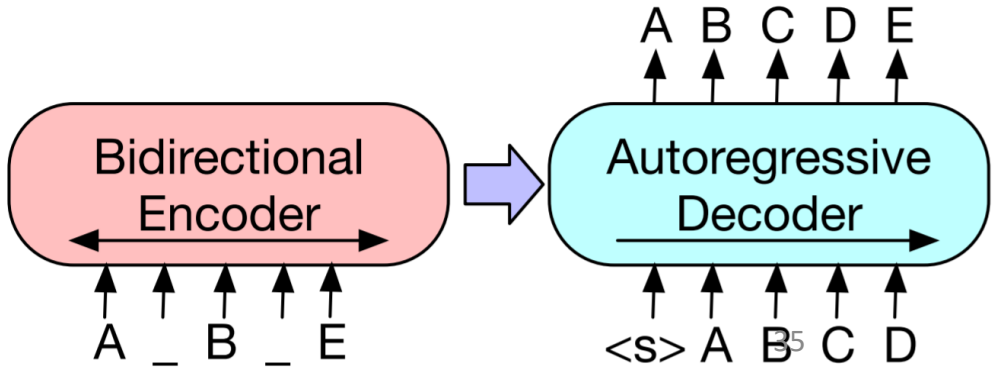
3. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension (Lewis et al., ACL 2020)

1) Contribution

- 기존에 Masked Language Model(MLM)들이 특정 유형 task에만 집중하여서 응용력이 떨어짐 (e.g. Natural Language Understanding)

- Bidirectional Encoder + Auto-regressive Decoder (like “seq2seq” model)

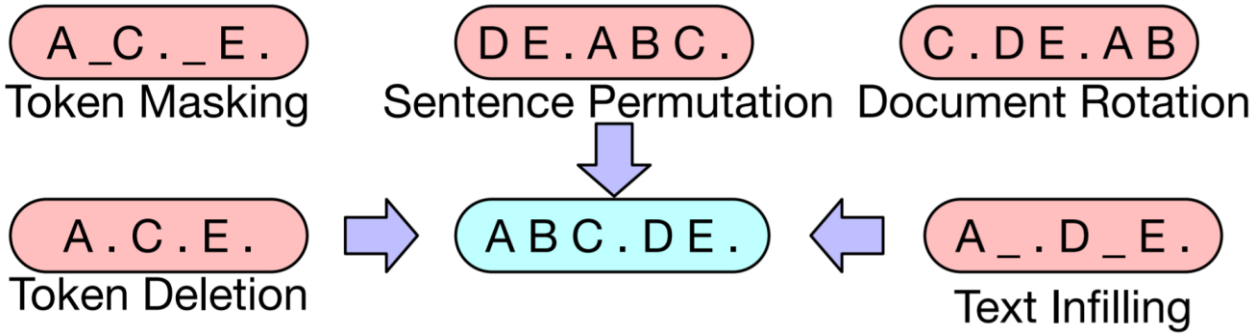
- 다섯가지 noise로 손상시킨 데이터로 pretrain



https://arxiv.org/pdf/1910.13461.pdf

3. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension (Lewis et al., ACL 2020)

2) Five different noises



- Token Masking: BERT처럼 랜덤으로 masking하고 맞추기
- Token Deletion: 랜덤 토큰을 삭제하고 복구하기. Masking과 다르게 어떤 위치의 토큰이 지워졌는지 알 수 없음
- Text Infilling: 매번 임의의 길이 (0-6) 만큼 토큰들이 하나의 mask 토큰으로 대체됨. 모델은 얼마나 많은 수의 토큰들이 대체되었는지 예측함
- Sentence Permutation: Document를 문장 단위로 나눠서 섞음
- Document Rotation: 임의의 토큰을 선택하고 이 토큰이 document의 시작이 되도록 문장 순서를 바꿈. 모델은 document의 시작을 예측하도록 훈련됨

3) Fine-tuning BART: Five different tasks

https://arxiv.org/pdf/2003.10555.pdf

4. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators (Clark et al., ICLR 2020)

0) Drawbacks

- 기존의 input sequence 토큰들을 15% 랜덤하게 masking 하고 복원하는 방식으로 학습하는 masked language model (MLM)은 하나의 example에 대해 15% 밖에 학습하지 못함

- 학습이 굉장히 비효율적이고 비쌈

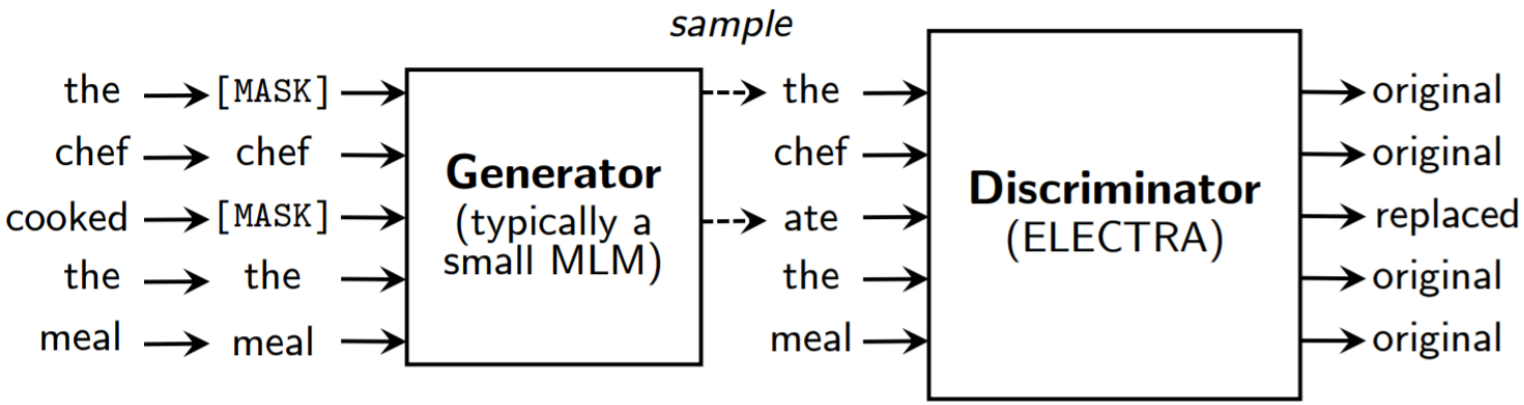
- 학습시에는 모델이 mask 토큰을 같이 학습하지만 실제 inference시에는 mask 토큰이 없음…

https://arxiv.org/pdf/2003.10555.pdf

4. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators (Clark et al., ICLR 2020)

1) Contribution

- Replaced Token Detection (RTD)라는 새로운 task를 제안함
- Generator가 실제 입력의 일부 토큰을 있을법한 토큰으로 치환하고 Discriminator가 이 토큰이 original인지 replaced인지 맞추도록 학습함
- 15%가 아니라 모든 input에 대해 masking을 적용하기 때문에 훨씬 효율적으로 학습하며 학습 속도도 빠름



https://arxiv.org/pdf/2003.10555.pdf

4. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators (Clark et al., ICLR 2020)

2) ELECTRA substantially outperforms MLM-based methods

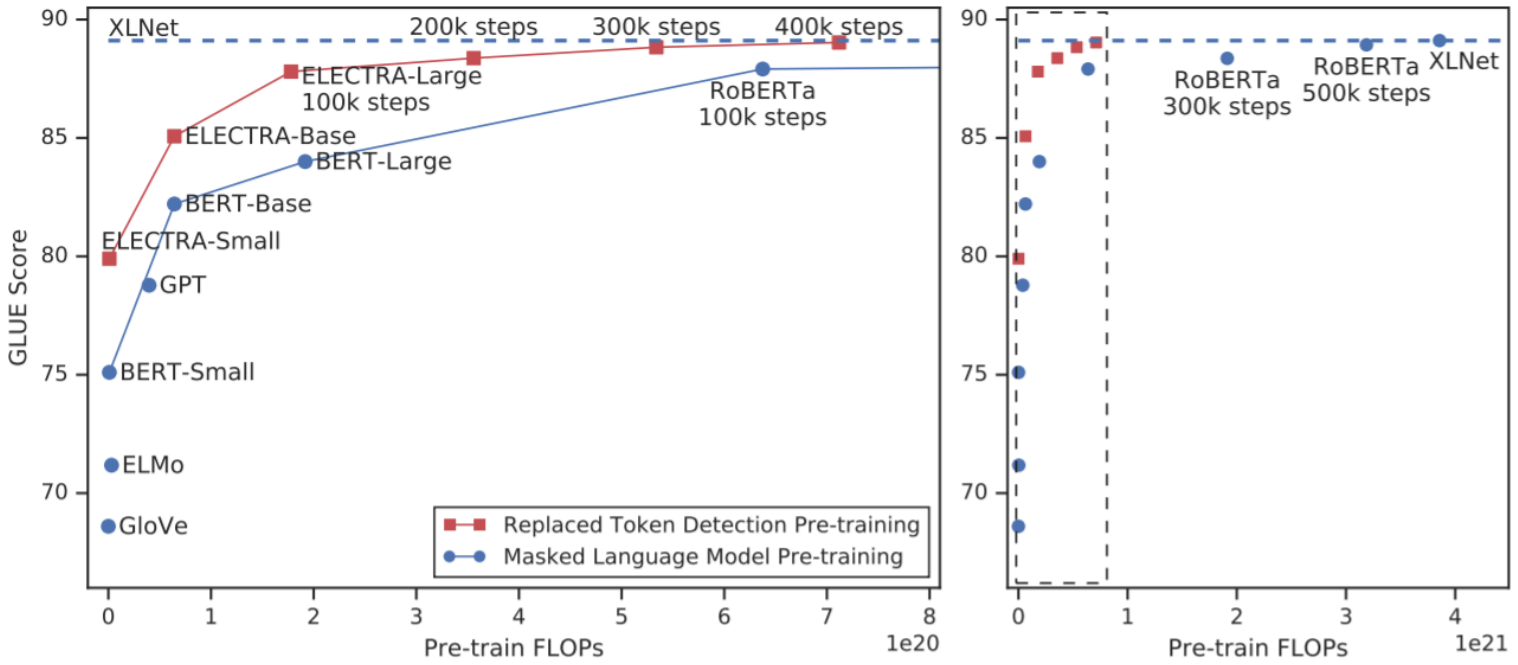


Figure 1: Replaced token detection pre-training consistently outperforms masked language model pre-training given the same compute budget. The left figure is a zoomed-in view of the dashed box.

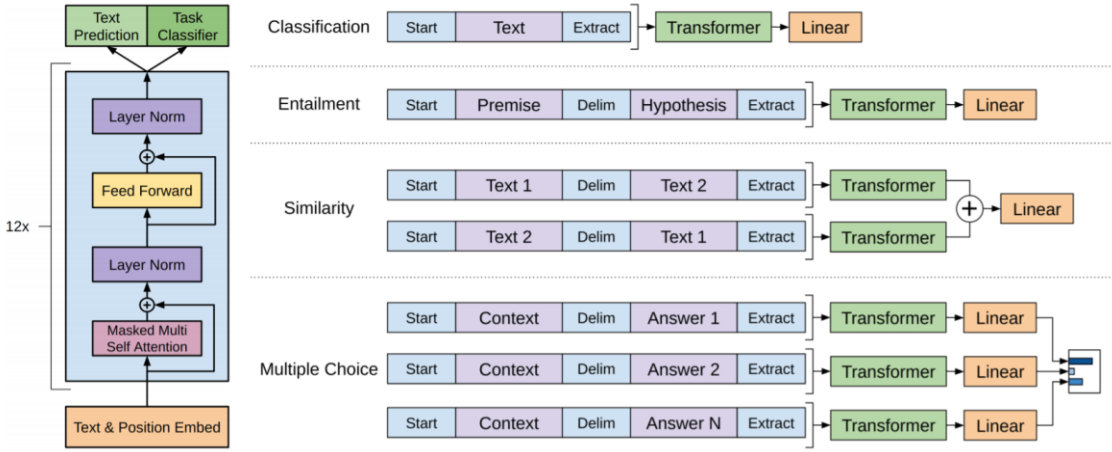
https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

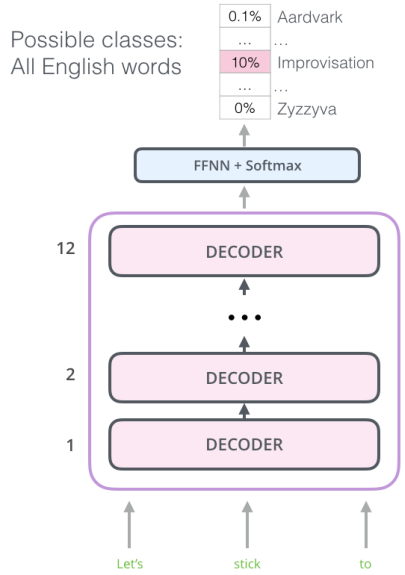
5. GPT-n

1. GPT-1 *(Improving Language Understanding by Generative Pre-Training (Radford and Sutskever et al., 2018 arXiv)*

1) Contribution

- 해결하려는 문제: scarcity of labeled data & Task-aware input representation during fine-tuning
- Transformer decoder를 이용한 순방향 언어모델
- Unsupervised pre-training using unlabeled text
- Supervised fine-tuning using manually annotated dataset





https://d4mucfpksywv.cloudfront.net/better-language-models/language_models_are_unsupervised_multitask_learners.pdf
https://jalammar.github.io/illustrated-gpt2/

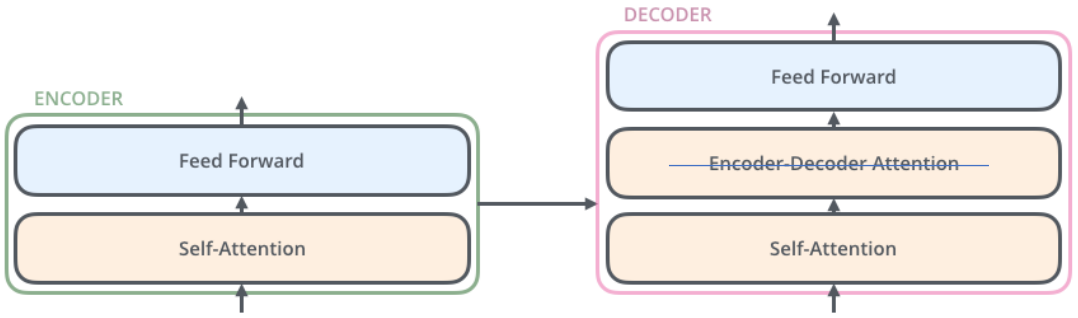
5. GPT-n

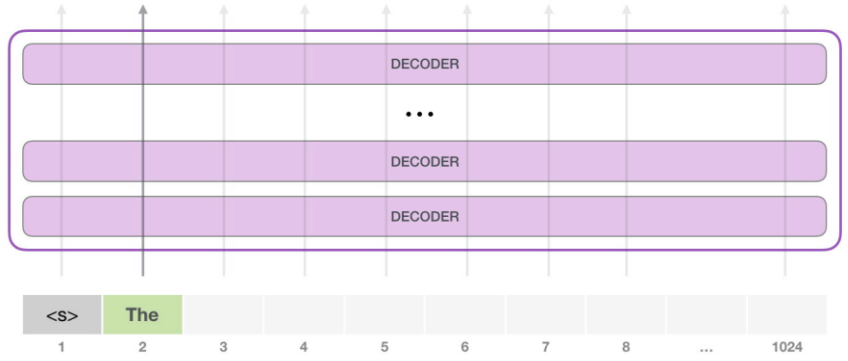
2. GPT-2 *(Language Models are Unsupervised Multitask Learners (Radford and Wu et al., 2019 arXiv)*

1) Contribution

- 해결하려는 문제: Zero-shot task transfer, When only minimal or no supervised data is available, and Ability to optimize the unsupervised objective to convergence

2) Recap: Transformer decoder



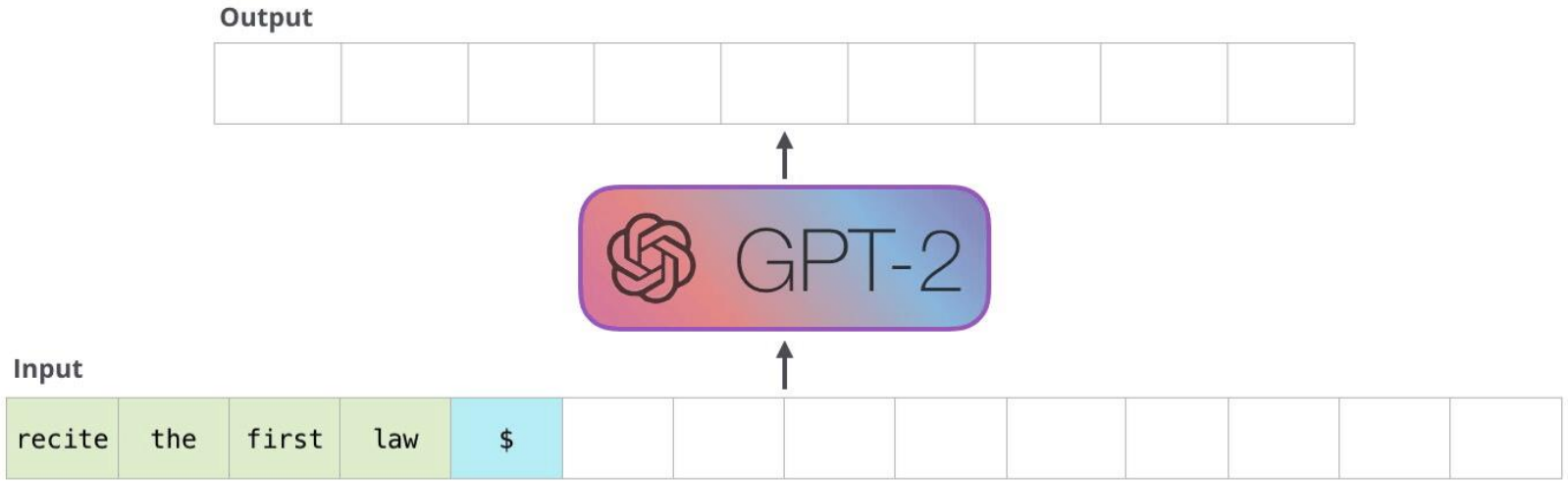


https://d4mucfpksywv.cloudfront.net/better-language-models/language_models_are_unsupervised_multitask_learners.pdf
https://jalammar.github.io/illustrated-gpt2/

5. GPT-n

2. GPT-2 *(Language Models are Unsupervised Multitask Learners (Radford and Wu et al., 2019 arXiv)*

3) Auto-regressive model

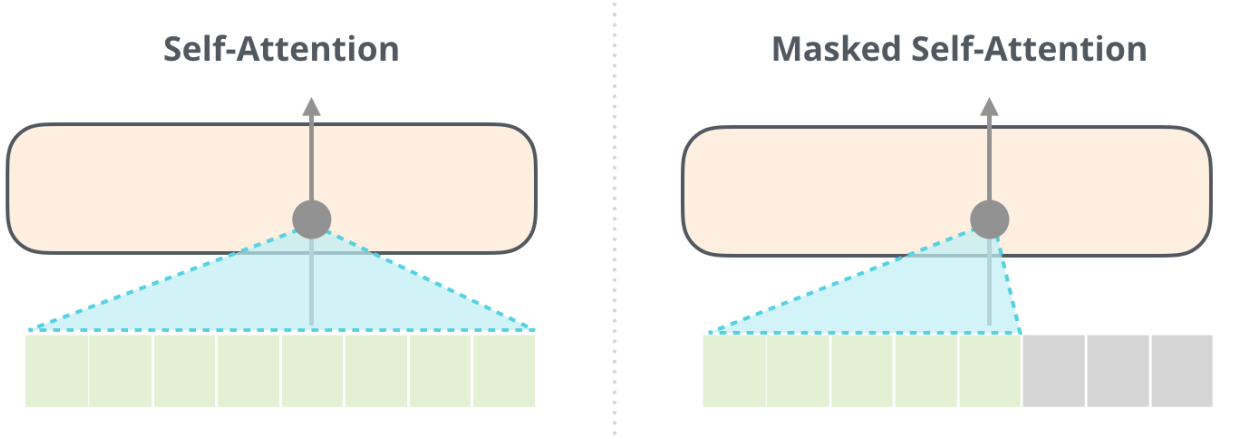


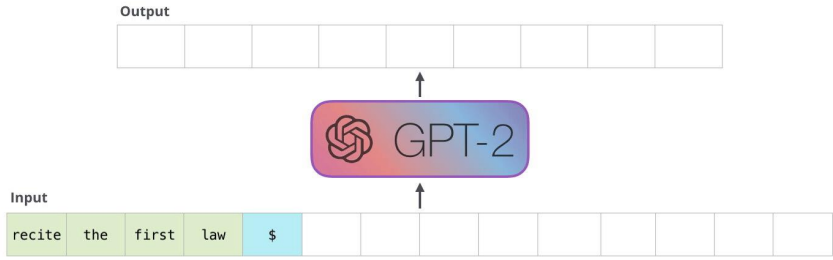
https://d4mucfpksywv.cloudfront.net/better-language-models/language_models_are_unsupervised_multitask_learners.pdf
https://jalammar.github.io/illustrated-gpt2/

5. GPT-n

2. GPT-2 *(Language Models are Unsupervised Multitask Learners (Radford and Wu et al., 2019 arXiv)*

4) Masked Self-Attention





https://arxiv.org/pdf/2005.14165.pdf

5. GPT-n

3. GPT-3 *(Language Models are Few-Shot Learners (Tom B et al., 2020 arXiv)*

1) Contribution: 파라메터: 1750억개… >>> BERT 3억개

- Few-shot learning: 풀고자 하는 문제에 대해 몇 개의 예시만 보고 문제를 푸는 task

Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

```
1  Translate English to French:        ← task description
2  sea otter => loutre de mer          ← examples
3  peppermint => menthe poivrée        ←
4  plush girafe => girafe peluche      ←
5  cheese =>                           ← prompt
```

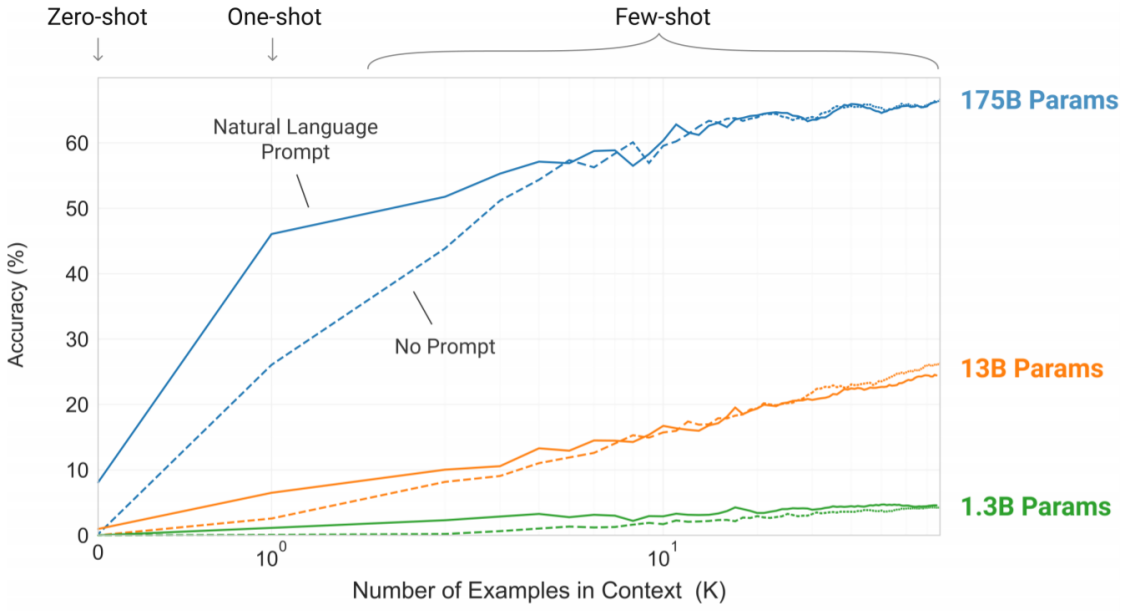
https://arxiv.org/pdf/2005.14165.pdf

5. GPT-n

3. GPT-3 *(Language Models are Few-Shot Learners (Tom B et al., 2020 arXiv)*

1) Contribution

- one-shot learning: 하나의 예제만을 허용
- zero-shot learning: 예제는 사용하지 않음. Task에대한 설명, 지시사항만을 모델에게 줌 (prompt)



1. Task에 대한 prompt는 모델성능을 향상시킴

2. 모델의 context window에 대 많은 예제를 놓을수록 성능이 향상 (K에 비례)

3. 큰 모델일수록 in-context 정보를 잘 활용함

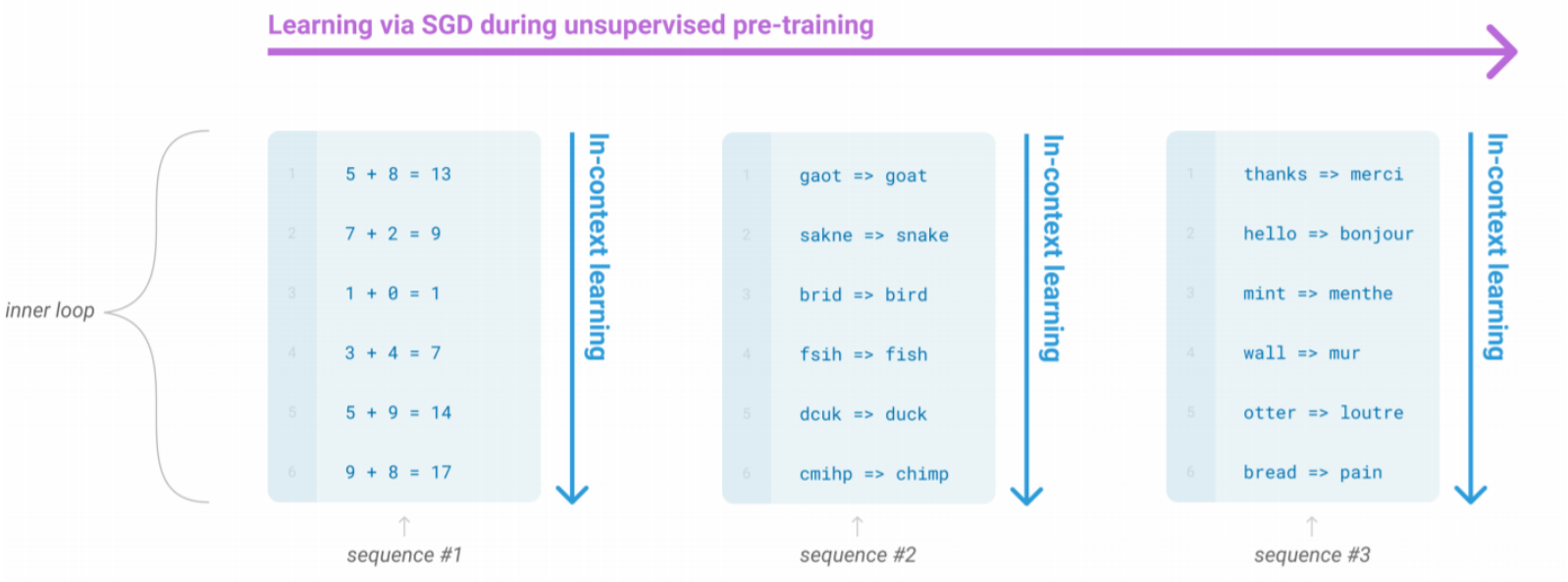
https://arxiv.org/pdf/2005.14165.pdf

5. GPT-n

3. GPT-3 *(Language Models are Few-Shot Learners (Tom B et al., 2020 arXiv)*

2) In-context 정보: pretrained model에 풀고자 하는 task를 text input으로 넣는 방식

- 각 시퀀스들을 돌면서 시퀀스 안에 세자릿수 덧셈, 오탈자 교정, 번역등과 같이 데이터가 학습됨. 컨텍스트 안에서 학습된다는 의미

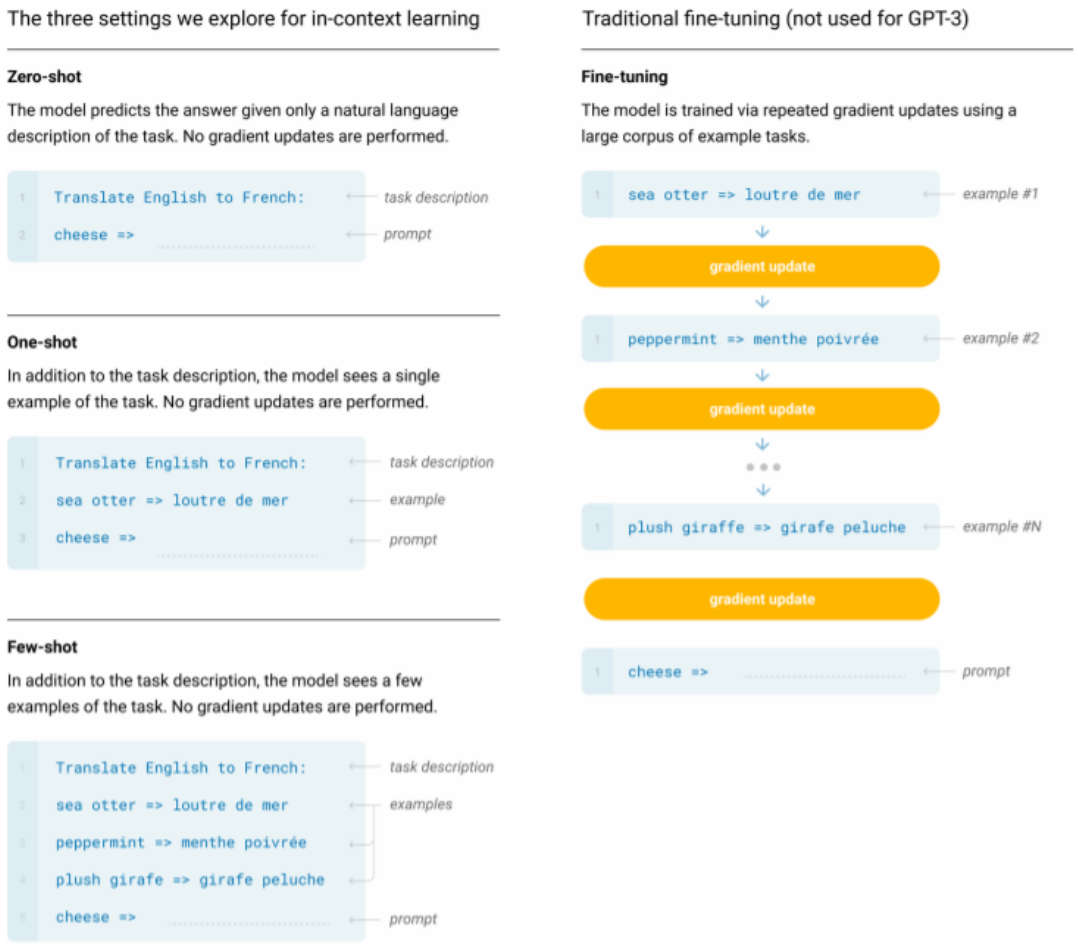


https://arxiv.org/pdf/2005.14165.pdf

5. GPT-n

3. GPT-3 *(Language Models are Few-Shot Learners (Tom B et al., 2020 arXiv)*

3) prompt



Plan for Today

1. Neural Networks?
2. Part 1
3. Transformer
4. Part 2
5. Conclusion

https://arxiv.org/pdf/2102.12092.pdf

1. DALL·E: Zero-Shot Text-to-Image Generation (Ramesh et al., 2021 arXiv)

TEXT PROMPT an illustration of a baby daikon radish in a tutu walking a dog

AI-GENERATED IMAGES

Edit prompt or view more images↓

TEXT PROMPT an armchair in the shape of an avocado. . . .

AI-GENERATED IMAGES

Edit prompt or view more images↓

Thank you

https://jeiyoon.github.io/

LLSOLLU