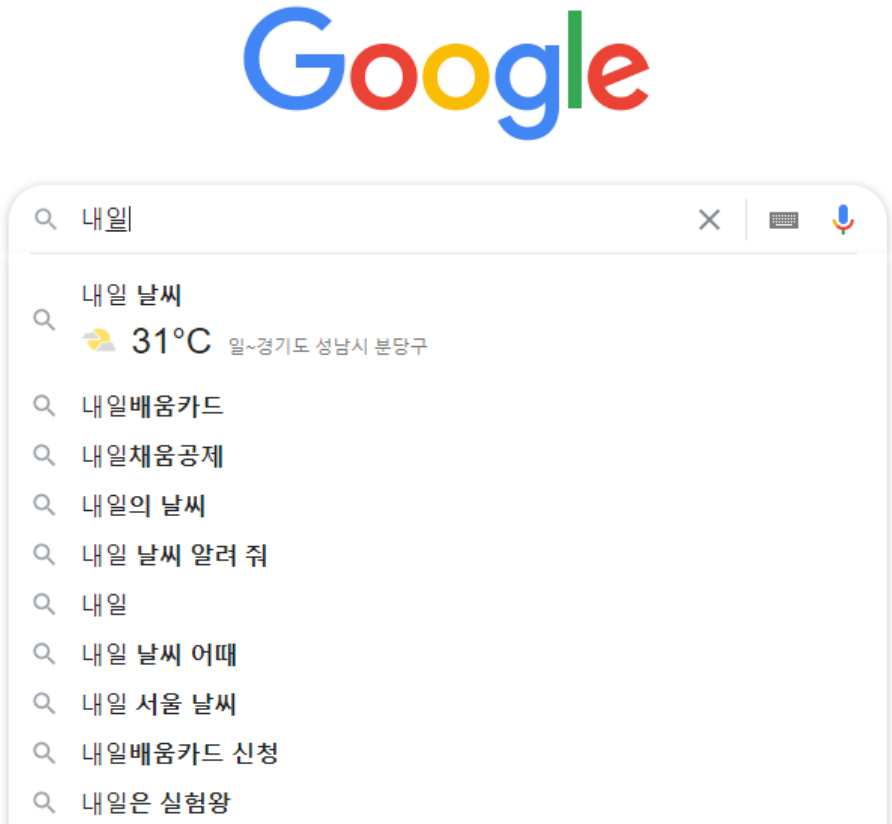
언어 모델(Language Model)

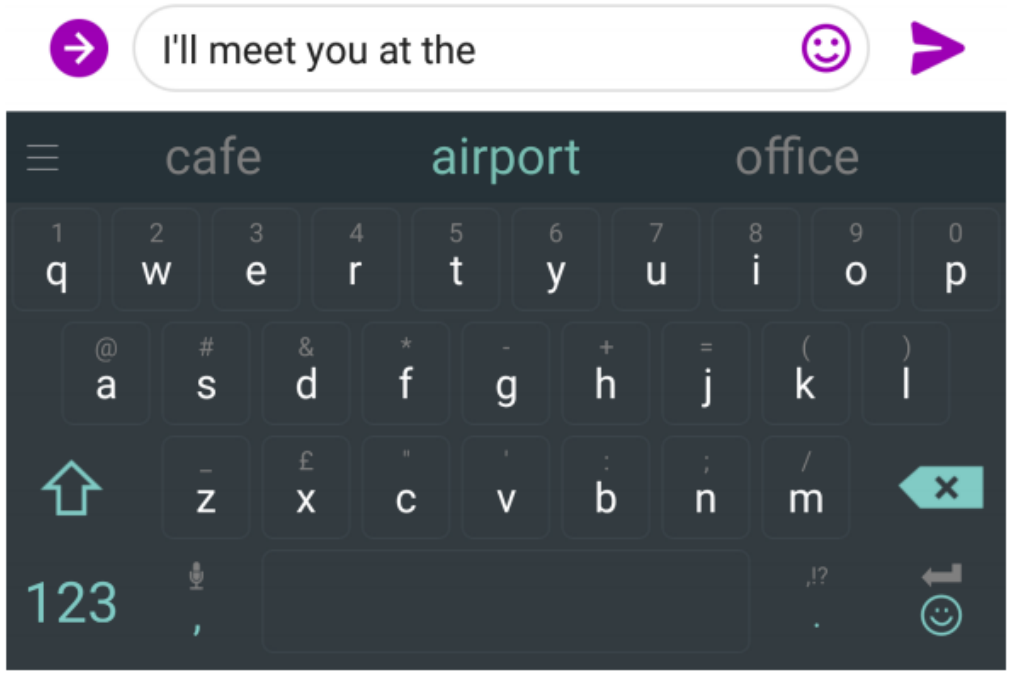
박경규

그림출처 : https://blog.naver.com/yn8

언어 모델(Language Model)

- 언어모델 : 언어 모델이란 **"자연어의 법칙을 컴퓨터로 모사한 모델"**을 의미합니다.
- 주어진 단어들로부터 그 **다음에 등장한 단어의 확률을 예측**하는 방식으로 학습합니다. (이전 state로 미래 state 예측)
- 다음의 등장할 단어를 잘 예측하는 모델은 그 **언어의 특성이 잘 반영된 모델**, **문맥을 잘 계산하는 좋은 언어 모델**





https://kpiswon.github.io/language/model/2020/09/19/Language-Model.html

언어 모델의 진화

1950

Fig. Evolution of NLP Models

2020

https://www.freshgravity.com/evolution-of-natural-language-processing/

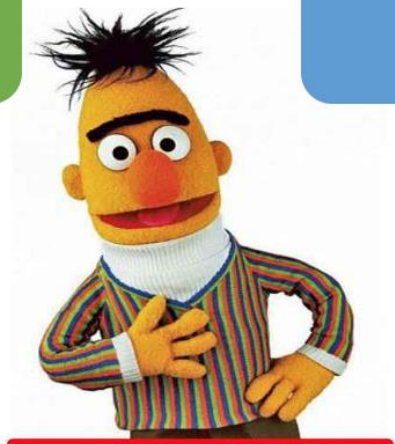
언어 모델의 진화

THE TRANSFORMER

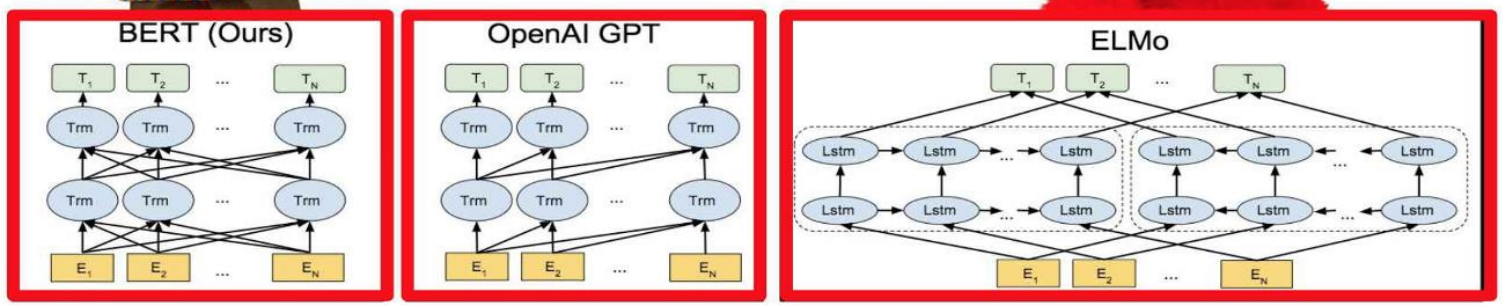
BERT

ELMo

Word2Vec



BERT



Popular algorithms in NLP

https://www.freshgravity.com/evolution-of-natural-language-processing/

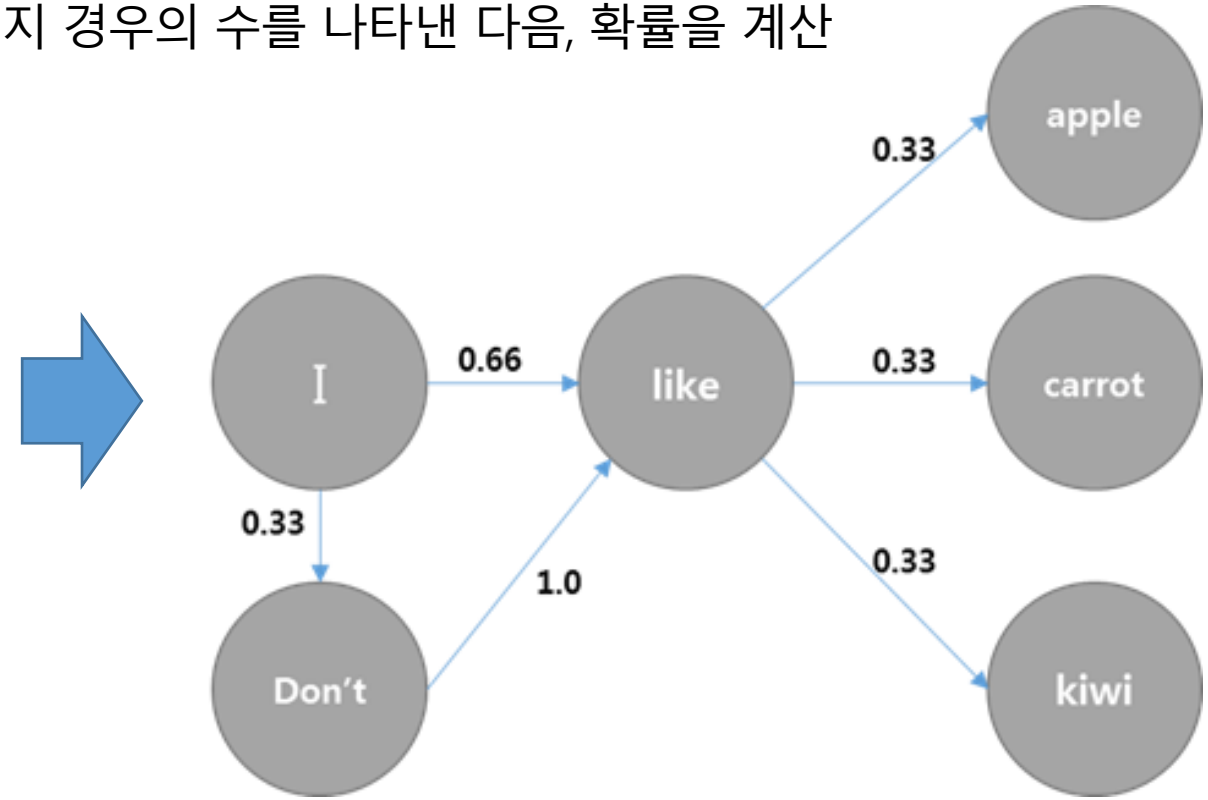
마코프 체인(Markov Chain)기반 언어 모델

- 마코프 체인 모델 (Markov Chain Model) : 미래 상태의 조건부 확률 분포는 과거 상태와는 독립적으로 , 현재 상태에 의해서만 결정
- 초기의 언어 모델은 다음의 단어나 문장이 나올 확률을 통계와 단어의 n-gram을 기반으로 계산
- 현재 단어 뒤에 미래의 단어가 어떤 것이 오는 지 경우의 수를 나타낸 다음, 확률을 계산

1) I like apple

2) I don't like carrot

3) I like kiwi



https://data-traveler.tistory.com/37

마코프 체인(Markov Chain)기반 언어 모델

	I	don't	like	apple	carrot	kiwi
I	-	0.33	0.66	-	-	-
don't	-	-	1.0	-	-	-
like	-	-	-	0.33	0.33	0.33
apple	-	-	-	-	-	-
carrot	-	-	-	-	-	-
kiwi	-	-	-	-	-	-

1) I like apple
0.66 0.33 = 0.22

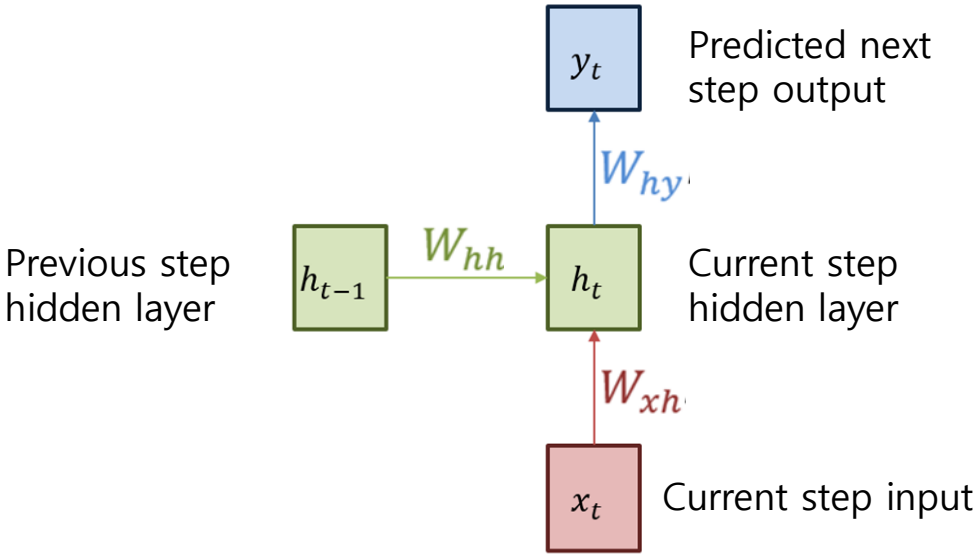
2) I don't like carrot
0.33 1.0 0.33 = 0.11

3) I like kiwi
0.66 0.33 = 0.22

https://data-traveler.tistory.com/37

RNN(Recurrent Neural Network) 기반 언어모델

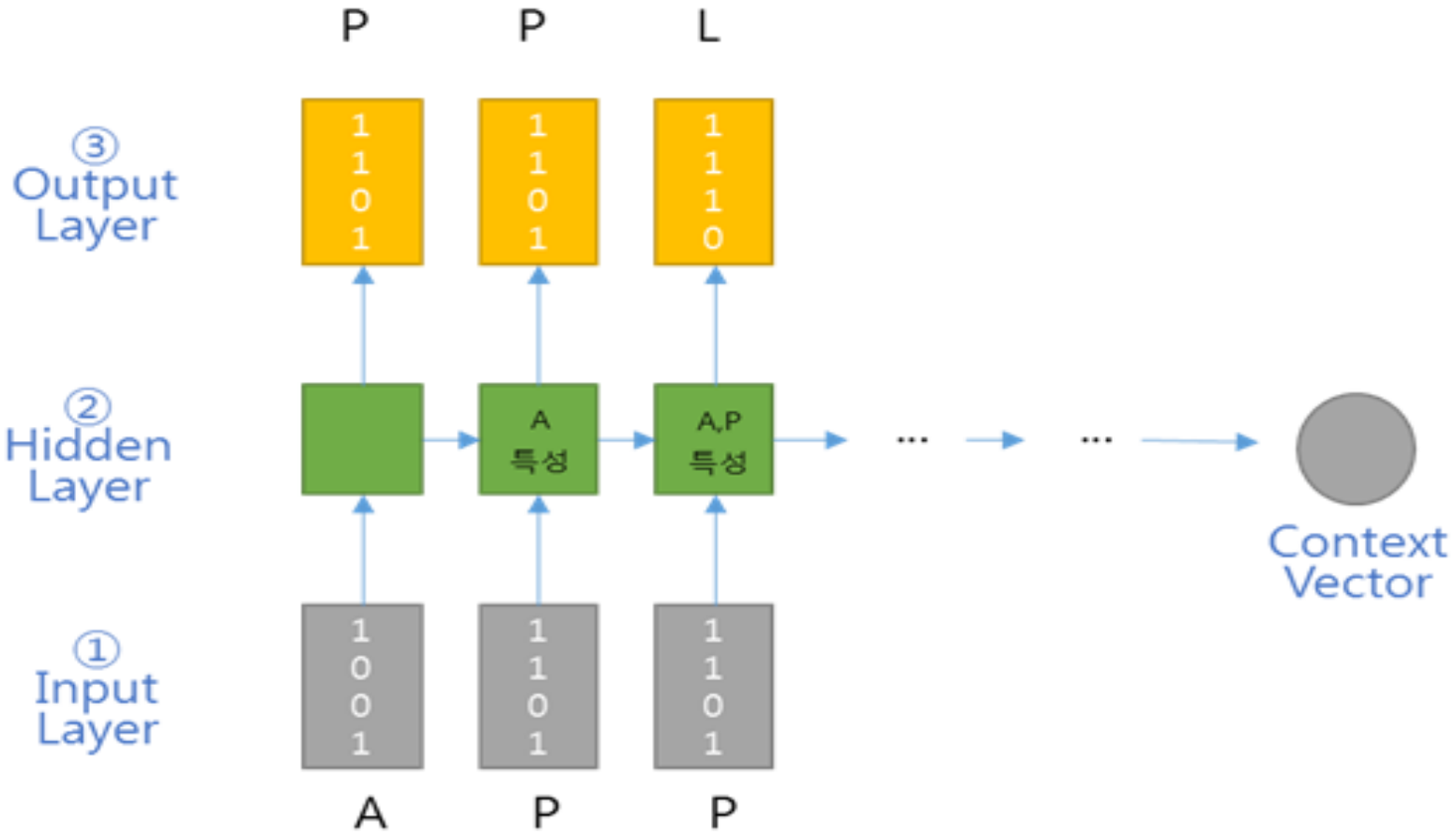
- Markov 확률 기반의 언어 모델은 과거의 문맥을 반영하기 어려우므로, 보완하는 방법으로 RNN 기반 언어모델 등장
- RNN은 순환구조 인공신경망으로 이전 state 정보가 다음 state 를 예측하는데 사용되어 시계열 데이터 처리에 특화



$$y_t = W_{hy} h_t + b_y$$

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

RNN(Recurrent Neural Network) 기반 언어모델



① Input Layer : 현재의 값
② Hidden Layer : 다음을 예측하기 위해 이전 값의 특성을 담는 곳
③ Output Layer : 현재를 기반으로 예측된 다음의 값

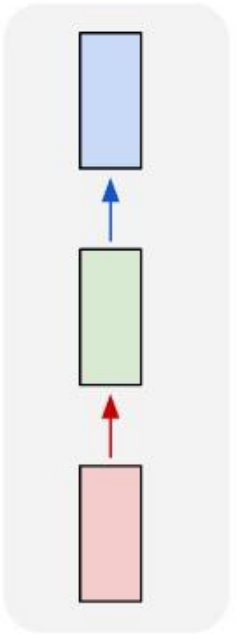
https://data-traveler.tistory.com/37

RNN 언어모델 활용

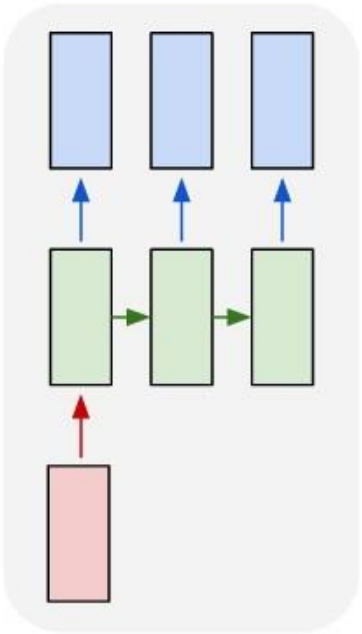
예. **감성 분류**
일련의 단어들 -> 감성

예. **기계 번역**
일련의 단어들 -> 일련의 단어들

one to one

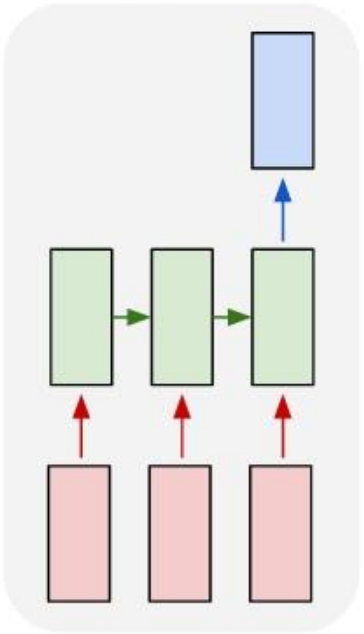


one to many



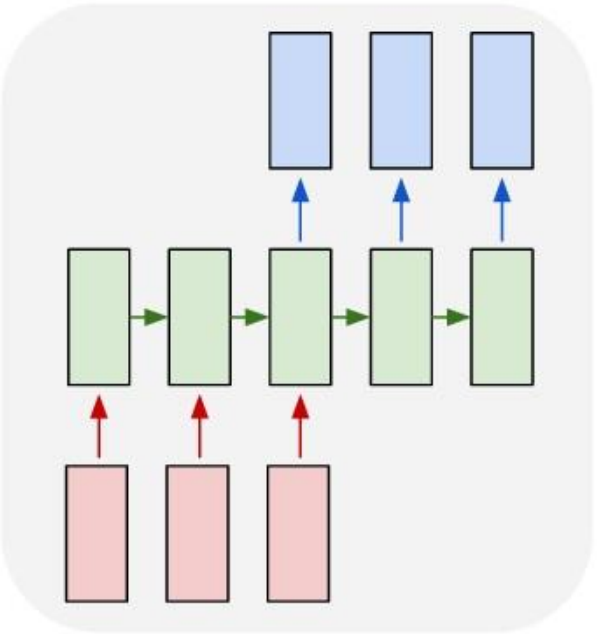
예) 이미지 Captioning
이미지 → 일련의 단어들

many to one



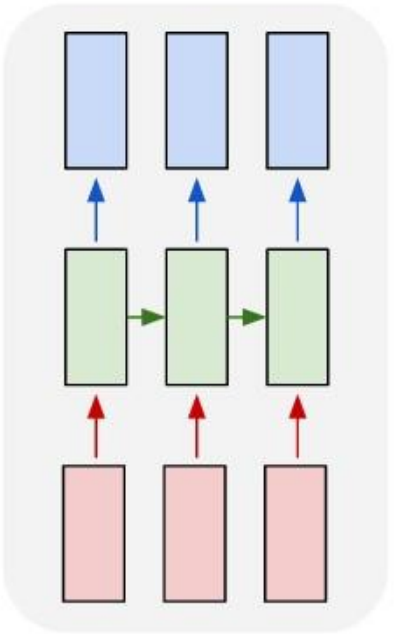
예) 감성 분류
일련의 단어들 → 감성

many to many



예) 기계 번역
일련의 단어들 → 일련의 단어들

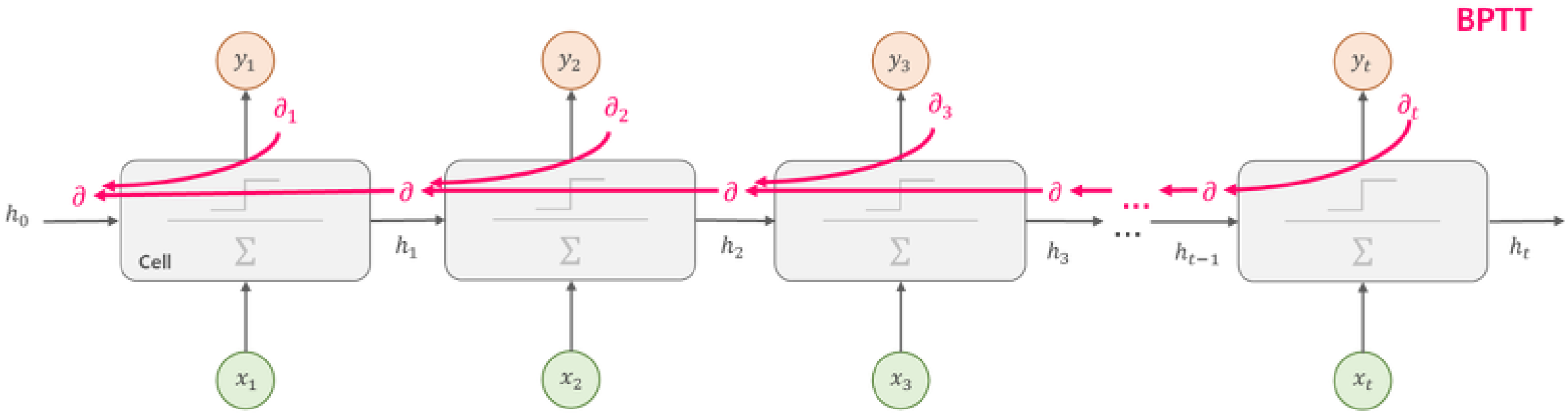
many to many



예) 프레임 수준에서 비디오 분류

BPTT (BackPropagation Through Time) 문제점

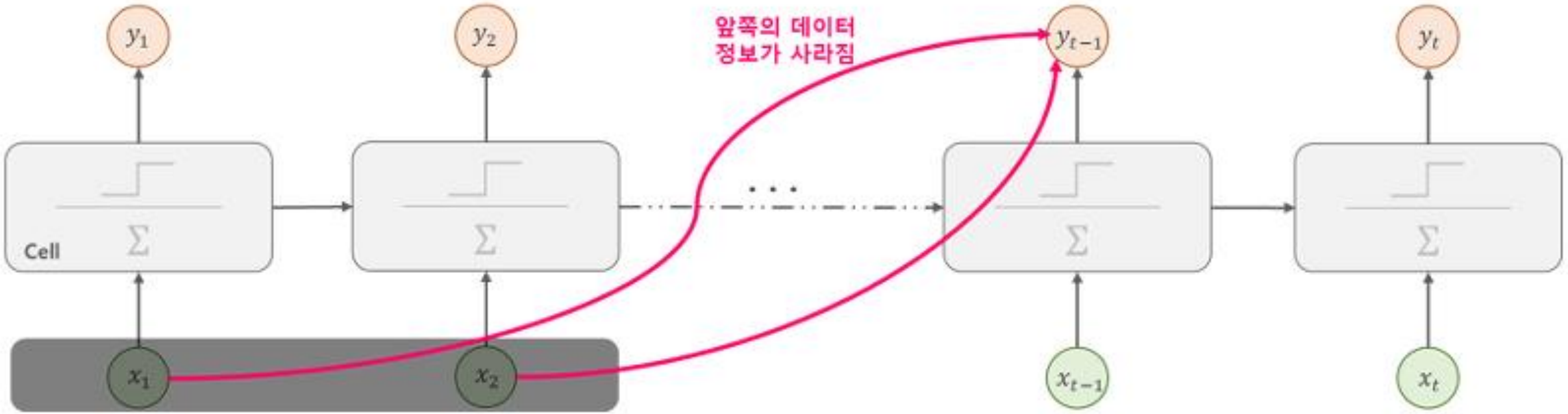
- RNN에서의 역전파 방법인 BPTT은 아래의 그림과 같이 모든 타임스텝마다 처음부터 끝까지 역전파합니다.
- 타임 스텝이 클 경우, 그림과 같이 RNN을 펼치게(unfold)되면 매우 깊은 네트워크가 될 것이며,
  이러한 네트워크는 그래디언트 소실 및 폭주(vanishing & exploding gradient) 문제가 발생할 가능성이 큽니다.
- 그리고, 계산량 또한 많기 때문에 한번 학습하는데 아주 오랜 시간이 걸리는 문제가 있습니다.



https://excelsior-cjh.tistory.com/185

장기 의존성(Long-Term Dependency) 문제

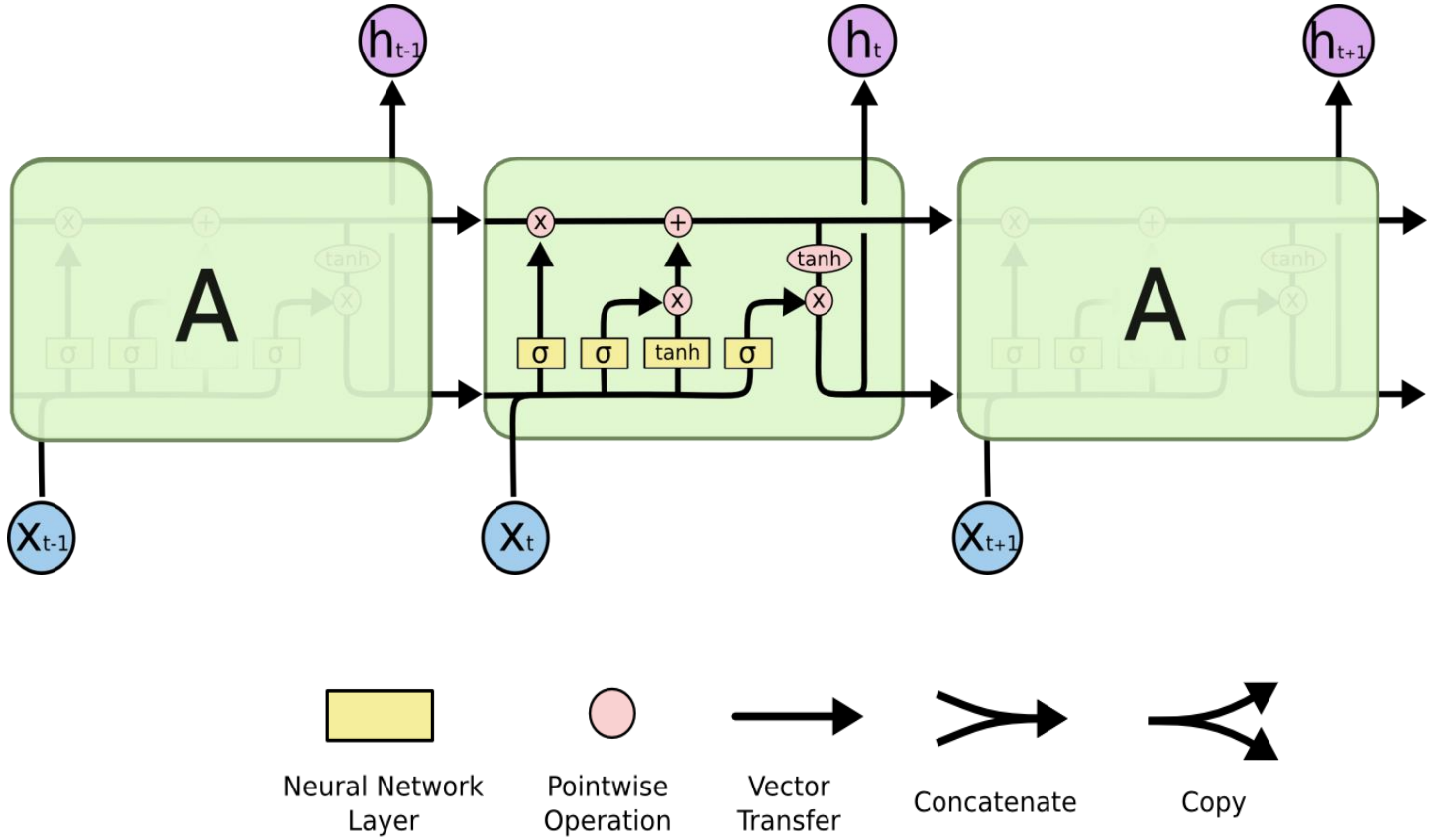
- RNN은 이전 타임 스텝의 상태를 입력으로 받는 구조로, 이전의 정보가 현재의 타임 스텝 에 영향을 줍니다.
- RNN은 이론적으로 모든 이전 타임 스텝이 영향을 주지만, 앞쪽의 타임 스텝은 타임 스텝이 길어질 수록 영향을 주지 못하는 문제가 발생하는데 이를 장기 의존성(Long-Term Dependency) 문제라고 합니다.



https://excelsior-cjh.tistory.com/185

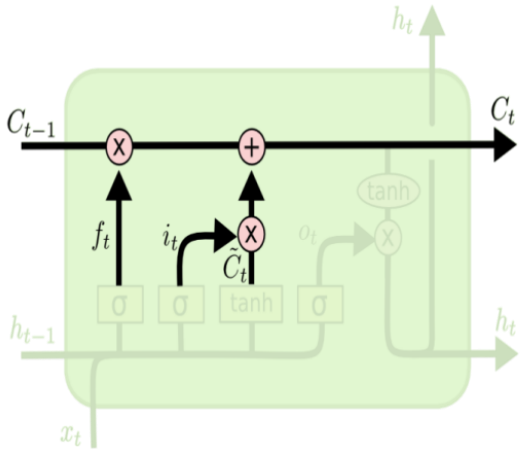
LSTM (Long-Short Term Memory)

- LSTM 네트워크는 장기적인 종속성을 학습할 수 있는 특수한 종류의 RNN입니다.
- LSTM은 RNN과 동일하게 입력과 출력사이 신경망이 재귀하는 구조를 갖고 있습니다.
- RNN은 재귀를 통한 정보전이 및 전파가 하나의 레이어로 제어되는 반면,
  LSTM은 Forget gate, Input gate, Output gate를 통한 정보전이 및 전파를 제어합니다.



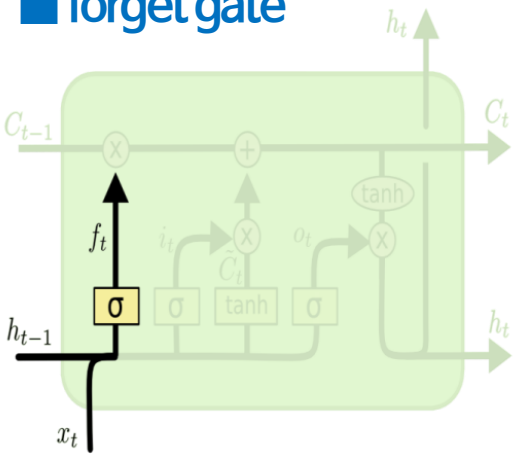
LSTM (Long-Short Term Memory)

■ Cell State(장기 상태)



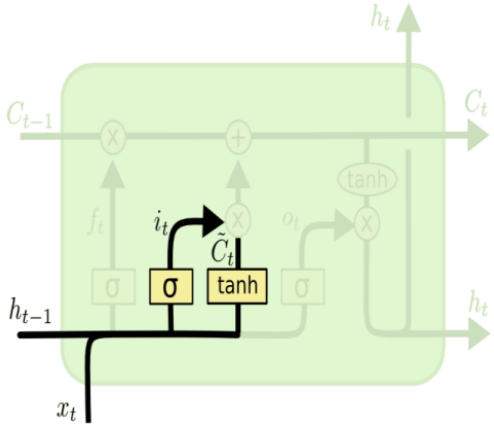
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

■ forget gate



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

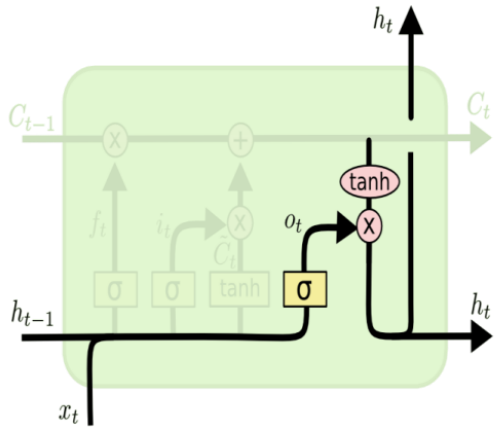
■ input gate



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

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

■ output gate, hidden state(단기 상태)



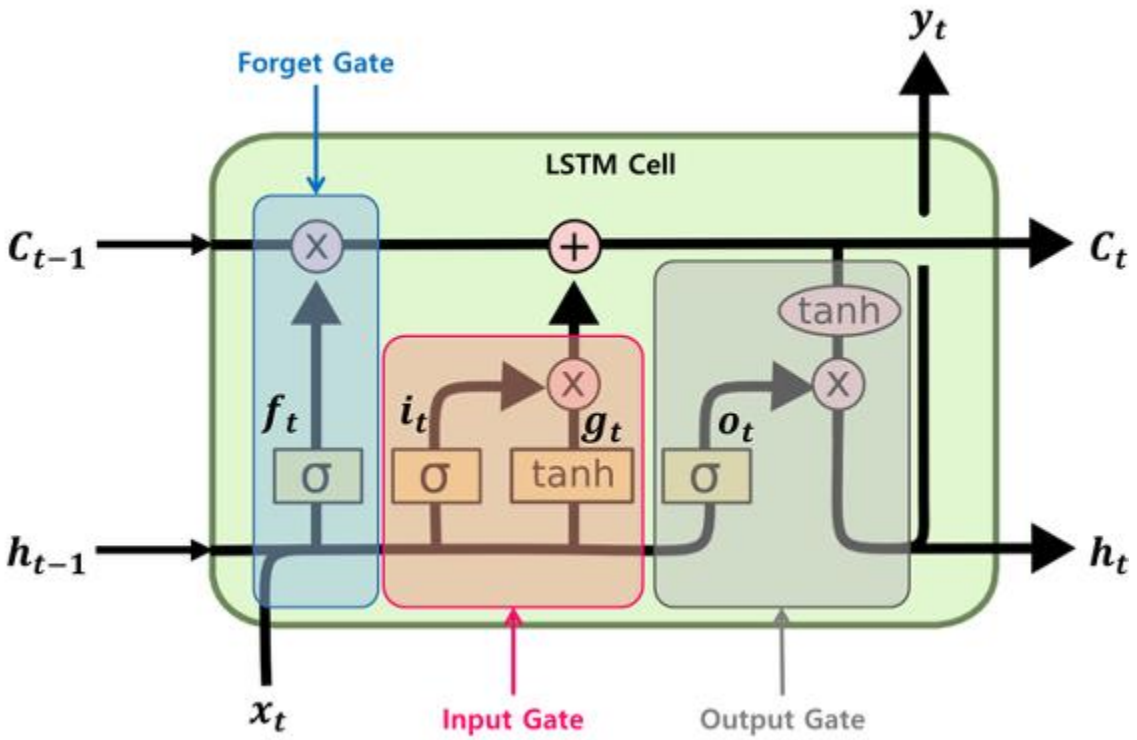
$$o_t = \sigma\left(W_o \left[h_{t-1}, x_t\right] \; + \; b_o\right)$$

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

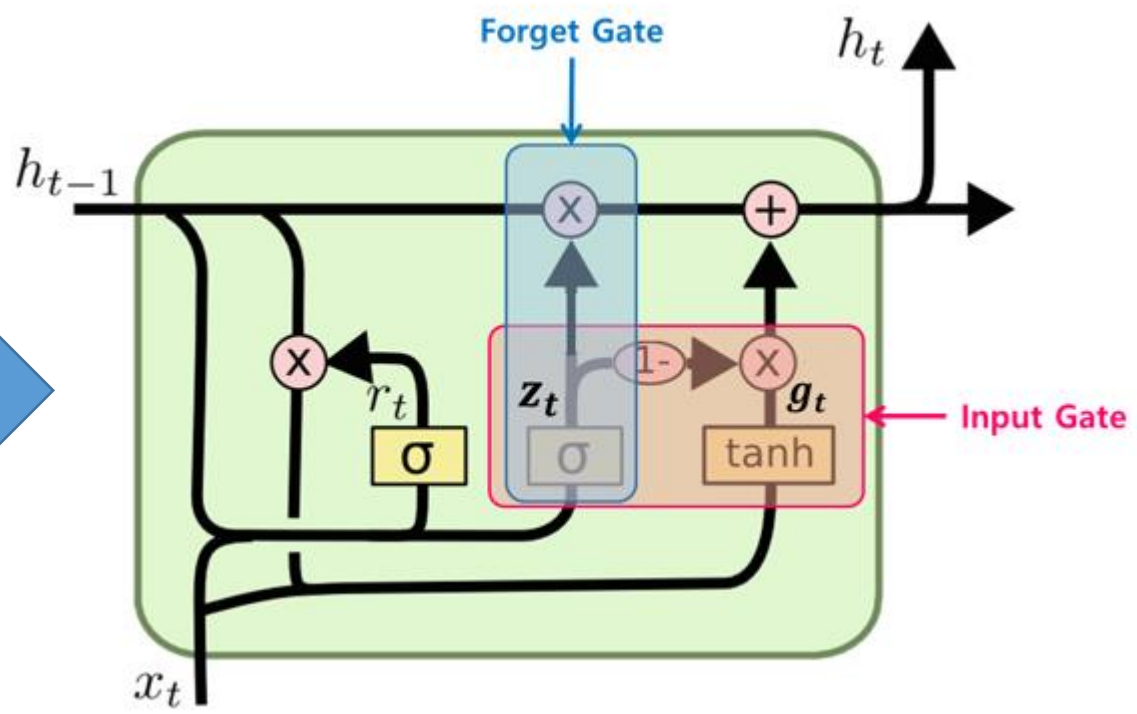
GRU (Gated Recurrent Unit)

- LSTM Cell에서의 두 상태 벡터 c_t 와 h_t 가 하나의 벡터 로 합쳐진, LSTM 간소화 버전입니다.
- 하나의 gate controller인 z_t가 forget과 input 게이트(gate)를 모두 제어합니다.

■ LSTM



■ GRU

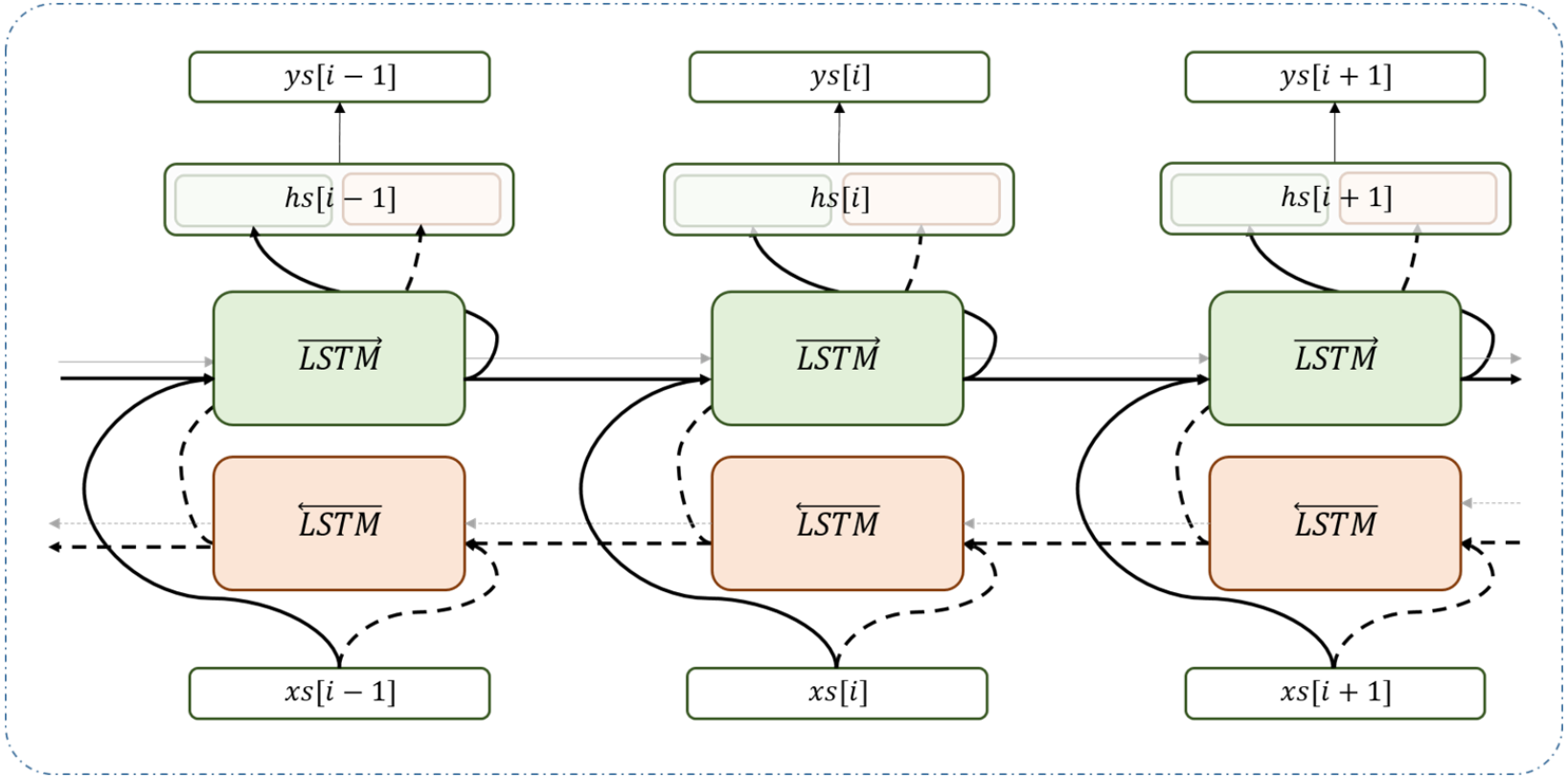


https://excelsior-cjh.tistory.com/185

Bidirectional LSTM

- 텍스트 데이터는 정방향 추론 뿐만이 아니라, 역방향 추론도 유의미한 결과가 존재
- LSTM을 2층을 쌓고, 서로 반대방향에서 연산 후, 붙이는 방식으로 합쳐줌

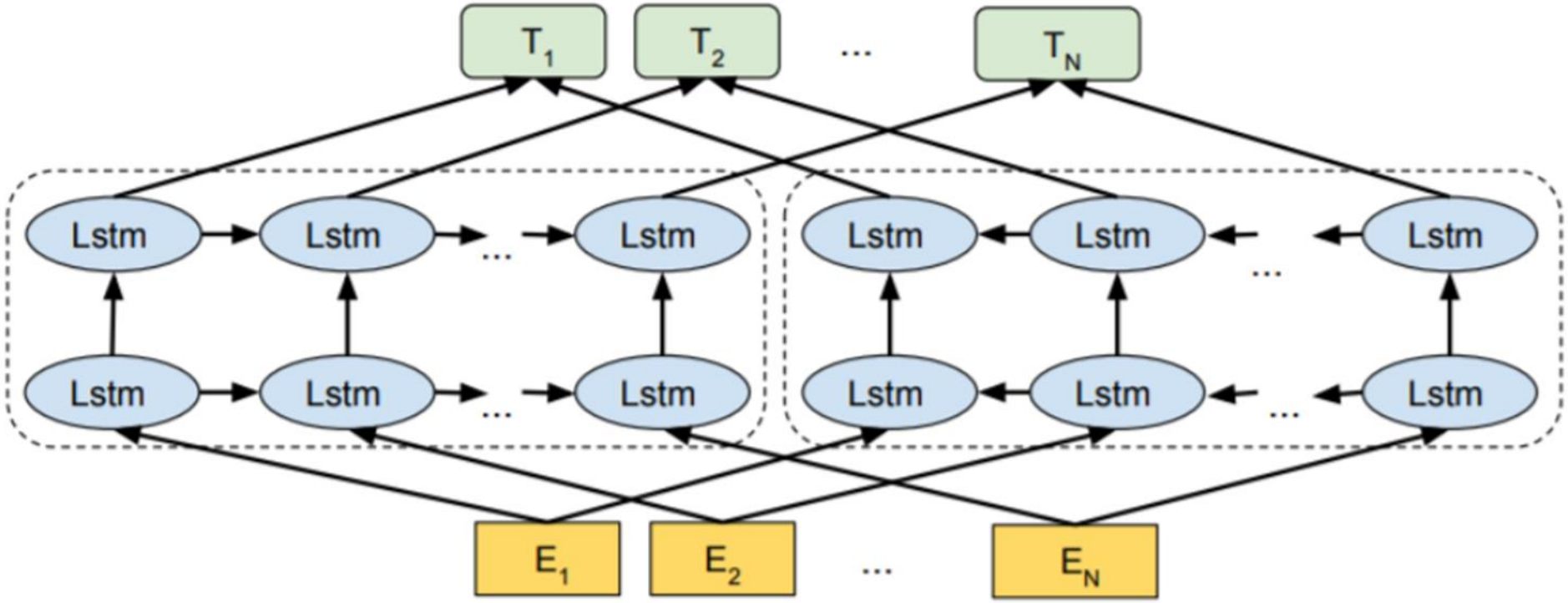
나는 ___를 뒤집어 쓰고 펑펑 울었다.



http://intelligence.korea.ac.kr/members/wschoi/nlp/deeplearning/Bidirectional-RNN-and-LSTM/

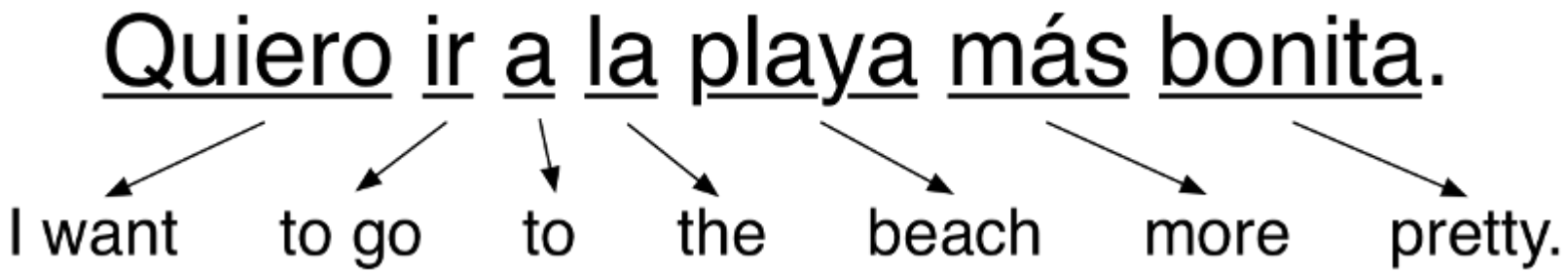
ELMo(Embedding from Language Models)

- ELMo(Embedding from Language Models) 언어모델로 하는 임베딩입니다.
- 2 Layer LSTM 을 이용하는 word embedding으로 각 Layer 의 hidden vector 결합을 embedding vector로 이용
- 앞/뒤, 문장 전체의 단어를 고려하는 hidden vector 를 단어 벡터로 이용함으로써 문맥을 표현할 수 있습니다.
- 단어 Tk의 임베딩 벡터는 모든 hidden vectors 의 선형 결합입니다.
- ELMo의 가장 큰 특징은 **Pre-trained language model**을 사용하는 것이며, 이름에 LM이 들어간 이유입니다.

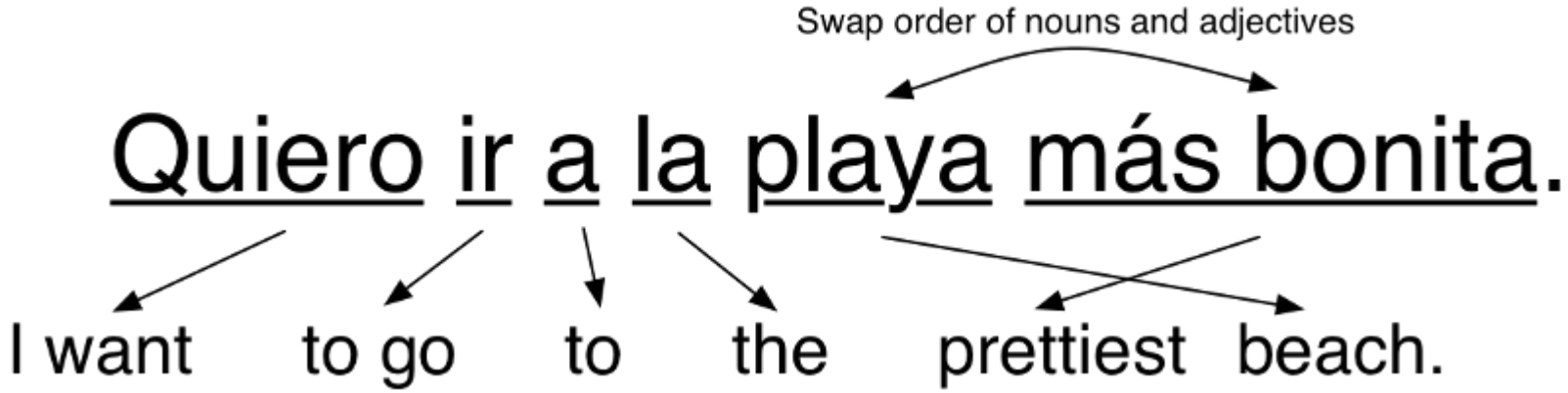


Seq2Seq(Sequence-to-Sequence)

■ 사전 기반 기계번역



■ 규칙 기반 기계번역



https://bit.ly/3eFiguY

Seq2Seq(Sequence-to-Sequence)

■ 병렬 코퍼스(parallel corpora)

English	Spanish
Resumption of the session I declare resumed the session of the European Parliament adjourned on Friday 17 December 1999, and I would like once again to wish you a happy new year in the hope that you enjoyed a pleasant festive period.	Reanudación del período de sesiones Declaro reanudado el período de sesiones del Parlamento Europeo, interrumpido el viernes 17 de diciembre pasado, y reitero a Sus Señorías mi deseo de que hayan tenido unas buenas vacaciones.

■ 확률 및 통계 기반 기계번역

https://bit.ly/3eFiguY

Seq2Seq(Sequence-to-Sequence)

■ 인코딩

Input Image

128 Measurements Generated from Image

0.097496084868908	0.045223236083984	-0.1281466782093	0.032084941864014
0.12529824674129	0.060309179127216	0.17521631717682	0.020976085215807
0.030809439718723	-0.01981477253139	0.10801389068365	-0.00052163278451189
0.036050599068403	0.065554238855839	0.0731306001544	-0.1318951100111
-0.097486883401871	0.1226262897253	-0.029626874253154	-0.0059557510539889
-0.0066401711665094	0.036750309169292	-0.15958009660244	0.043374512344599
-0.14131525158882	0.14114324748516	-0.031351584941149	-0.053343612700701
-0.048540540039539	-0.061901587992907	-0.15042643249035	0.078198105096817
-0.12567175924778	-0.10568545013666	-0.12728653848171	-0.076289616525173
-0.061418771743774	-0.074287034571171	-0.065365232527256	0.12369467318058
0.046741496771574	0.0061761881224811	0.14746543765068	0.056418422609568
-0.12113650143147	-0.21055991947651	0.0041091227903962	0.089727647602558
0.061606746166945	0.11345765739679	0.021352224051952	-0.0085843298584223
0.061989940702915	0.19372203946114	-0.086726233363152	-0.022388197481632
0.10904195904732	0.084853030741215	0.09463594853878	0.020696049556136
-0.019414527341723	0.0064811296761036	0.21180312335491	-0.050584398210049
0.15245945751667	-0.16582328081131	-0.035577941685915	-0.072376452386379
-0.12216668576002	-0.0072777755558491	-0.036901291459799	-0.034365277737379
0.083934605121613	-0.059730969369411	-0.070026844739914	-0.045013956725597
0.087945111095905	0.11478432267904	-0.089621491730213	-0.013955107890069
-0.021407851949334	0.14841195940971	0.078333757817745	-0.17898085713387
-0.018298890441656	0.049525424838066	0.13227833807468	-0.072600327432156
-0.011014151386917	-0.051016297191381	-0.14132921397686	0.0050511928275228
0.0093679334968328	-0.062812767922878	-0.13407498598099	-0.014829395338893
0.058139257133007	0.0048638740554452	-0.039491076022387	-0.043765489012003
-0.024210374802351	-0.11443792283535	0.071997955441475	-0.012062266469002
-0.057223934680223	0.014683869667351	0.05228154733777	0.012774495407939
0.023535015061498	-0.081752359867096	-0.031709920614958	0.069833360612392
-0.0098039731383324	0.037022035568953	0.11009479314089	0.11638788878918
0.020220354199409	0.12788131833076	0.18632389605045	-0.015336792916059
0.0040337680839002	-0.094398014247417	-0.11768248677254	0.10281457751989
0.051597066223621	-0.10034311562777	-0.040977258235216	-0.082041338086128

https://bit.ly/3eFiguY

Seq2Seq(Sequence-to-Sequence)

■ 인코딩

Input Sentence

"Machine Learning is Fun!"

Measurements Generated from Sentence

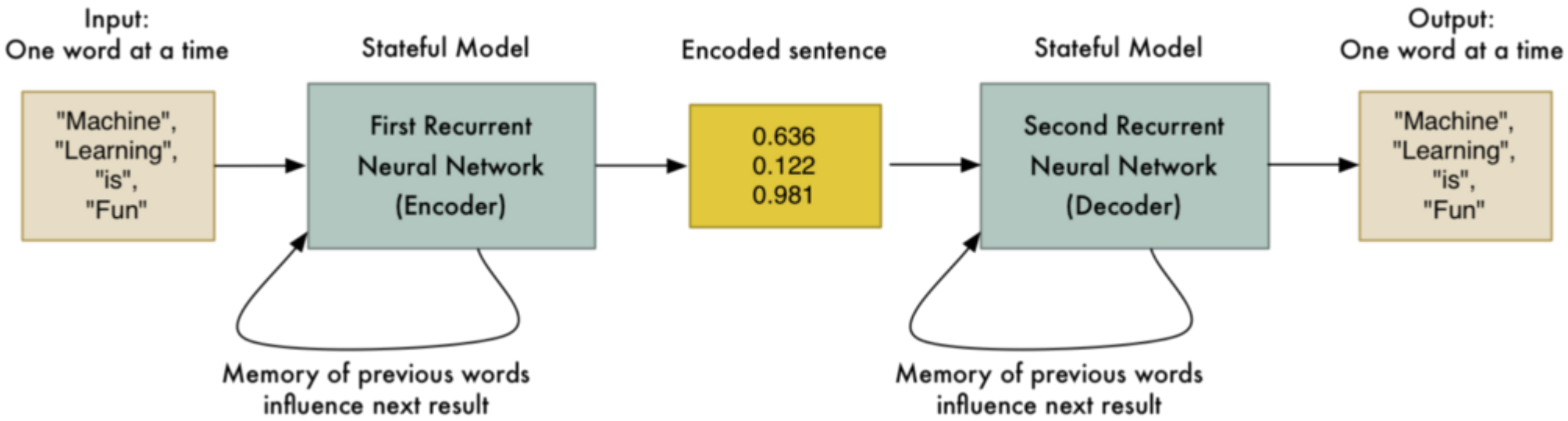
0.097496084868908	0.045223236083984	-0.1281466782093	0.032084941864014
0.12529824674129	0.060309179127216	0.17521631717682	0.020976085215807
0.030809439718723	-0.01981477253139	0.10801389068365	-0.00052163278451189
0.036050599068403	0.065554238855839	0.0731306001544	-0.1318951100111
-0.097486883401871	0.1226262897253	-0.029626874253154	-0.0059557510539889
-0.0066401711665094	0.036750309169292	-0.15958009660244	0.043374512344599
-0.14131525158882	0.14114324748516	-0.031351584941149	-0.053343612700701
-0.048540540039539	-0.061901587992907	-0.15042643249035	0.078198105096817
-0.12567175924778	-0.10568545013666	-0.12728653848171	-0.076289616525173
-0.061418771743774	-0.074287034571171	-0.065365232527256	0.12369467318058
0.046741496771574	0.0061761881224811	0.14746543765068	0.056418422609568
-0.12113650143147	-0.21055991947651	0.0041091227903962	0.089727647602558
0.061606746166945	0.11345765739679	0.021352224051952	-0.0085843298584223
0.061989940702915	0.19372203946114	-0.086726233363152	-0.022388197481632
0.10904195904732	0.084853030741215	0.09463594853878	0.020696049556136
-0.019414527341723	0.0064811296761036	0.21180312335491	-0.050584398210049
0.15245945751667	-0.16582328081131	-0.035577941685915	-0.072376452386379
-0.12216668576002	-0.0072777755558491	-0.036901291459799	-0.034365277737379
0.083934605121613	-0.059730969369411	-0.070026844739914	-0.045013956725597
0.087945111095905	0.11478432267904	-0.089621491730213	-0.013955107890069
-0.021407851949334	0.14841195940971	0.078333757817745	-0.17898085713387
-0.018298890441656	0.049525424838066	0.13227833807468	-0.072600327432156
-0.011014151386917	-0.051016297191381	-0.14132921397686	0.0050511928275228
0.0093679334968328	-0.062812767922878	-0.13407498598099	-0.014829395338893
0.058139257133007	0.0048638740554452	-0.039491076022387	-0.043765489012003
-0.024210374802351	-0.11443792283535	0.071997955441475	-0.012062266469002
-0.057223934680223	0.014683869667351	0.05228154733777	0.012774495407939
0.023535015061498	-0.081752359867096	-0.031709920614958	0.069833360612392
-0.0098039731383324	0.037022035568953	0.11009479314089	0.11638788878918
0.020220354199409	0.12788131833076	0.18632389605045	-0.015336792916059
0.0040337680839002	-0.094398014247417	-0.11768248677254	0.10281457751989
0.051597066223621	-0.10034311562777	-0.040977258235216	-0.082041338086128

https://bit.ly/3eFiguY

Seq2Seq(Sequence-to-Sequence)

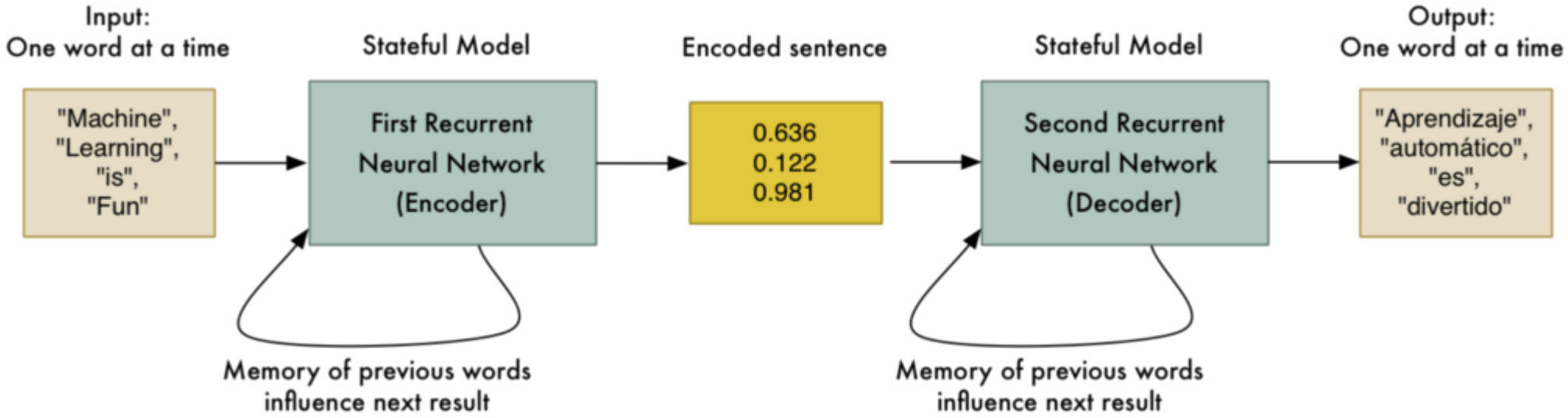
■ Encoder - Decoder

- Encoder layer: RNN 구조를 통해 Context vector 를 획득
- Decoder layer: 획득된 Context vector를 입력으로 출력을 예측

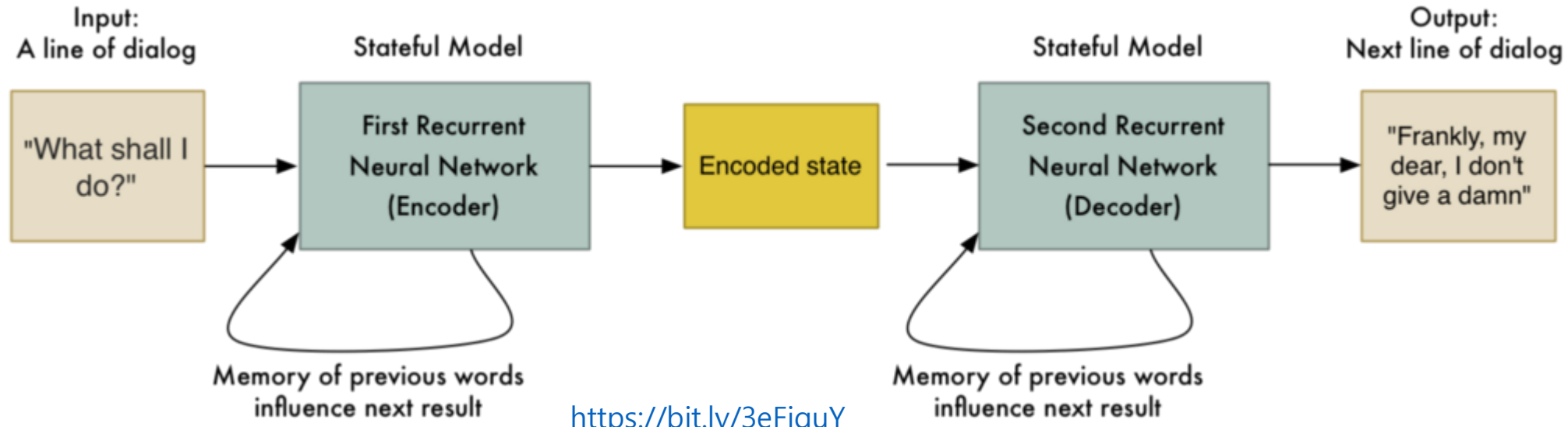


Seq2Seq(Sequence-to-Sequence)

■ 기계번역



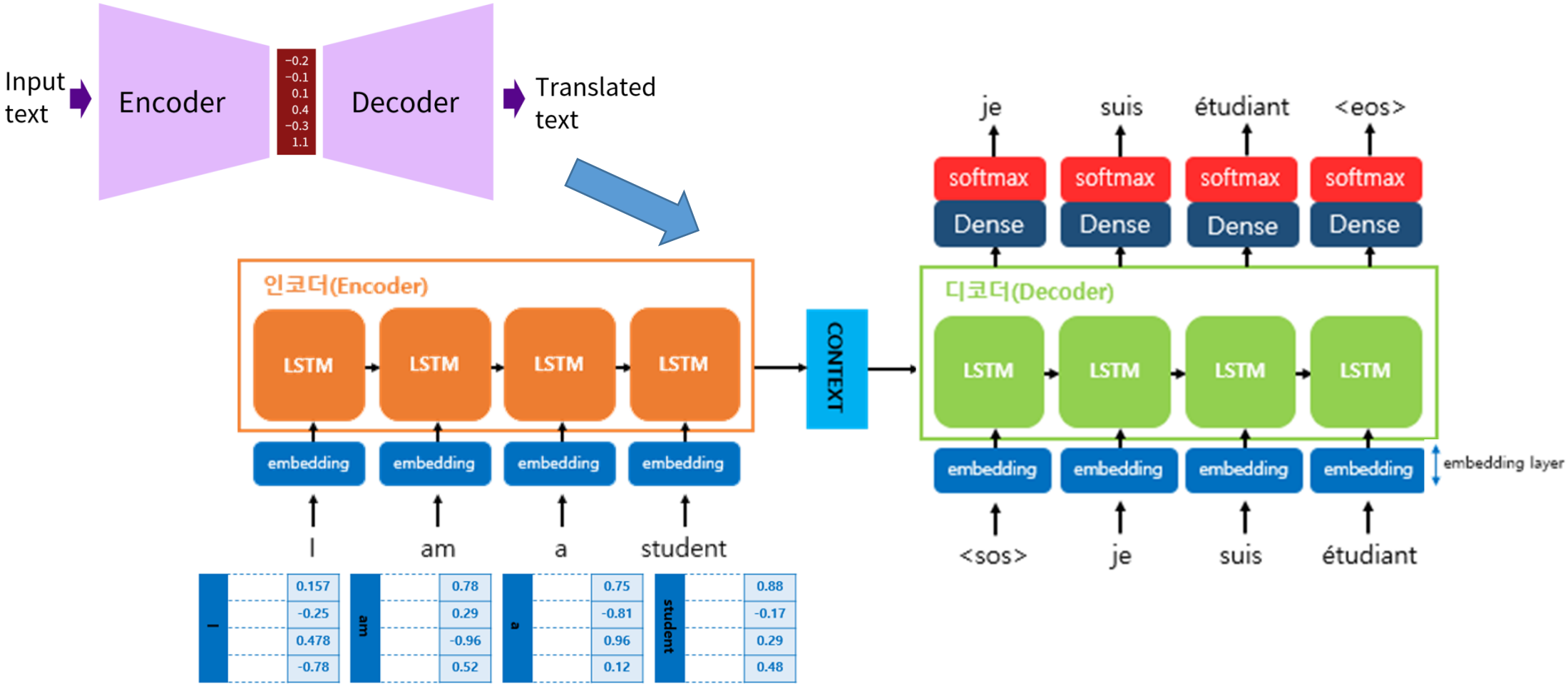
■ AI 챗봇



https://bit.ly/3eFiguY

Seq2Seq(Sequence-to-Sequence)

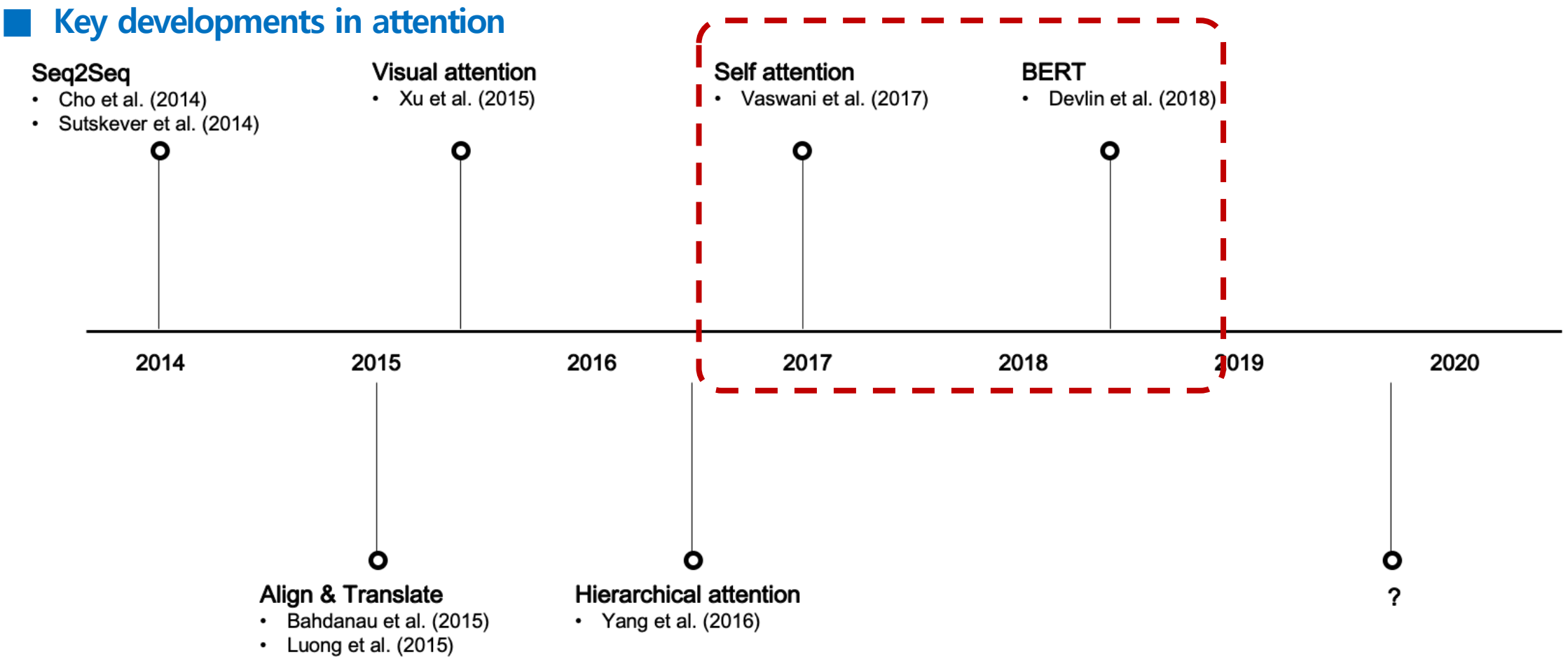
Encoder Layer에서는 Context Vector를 만들고 Decoder Layer에서는 Context Vector 를 입력으로 출력을 예측



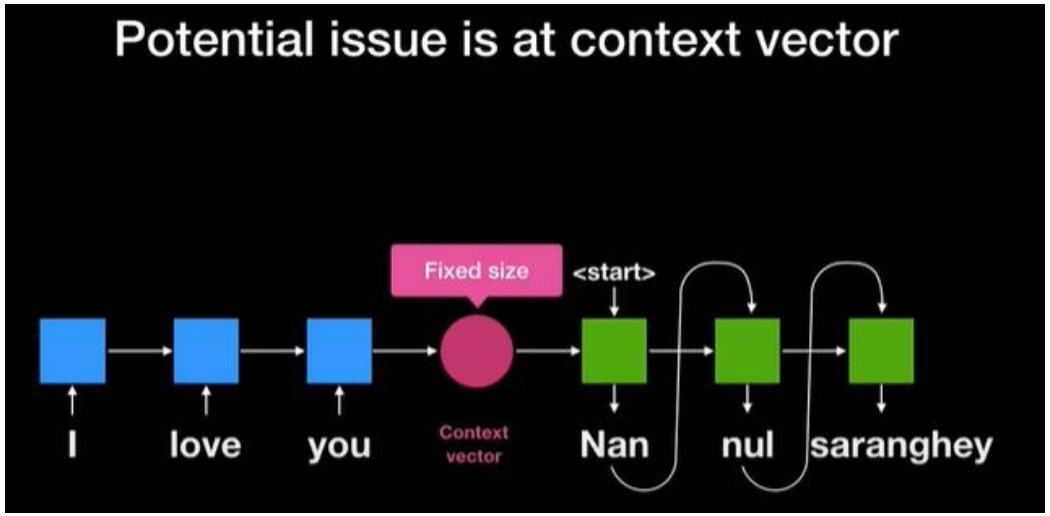
https://wikidocs.net/24996

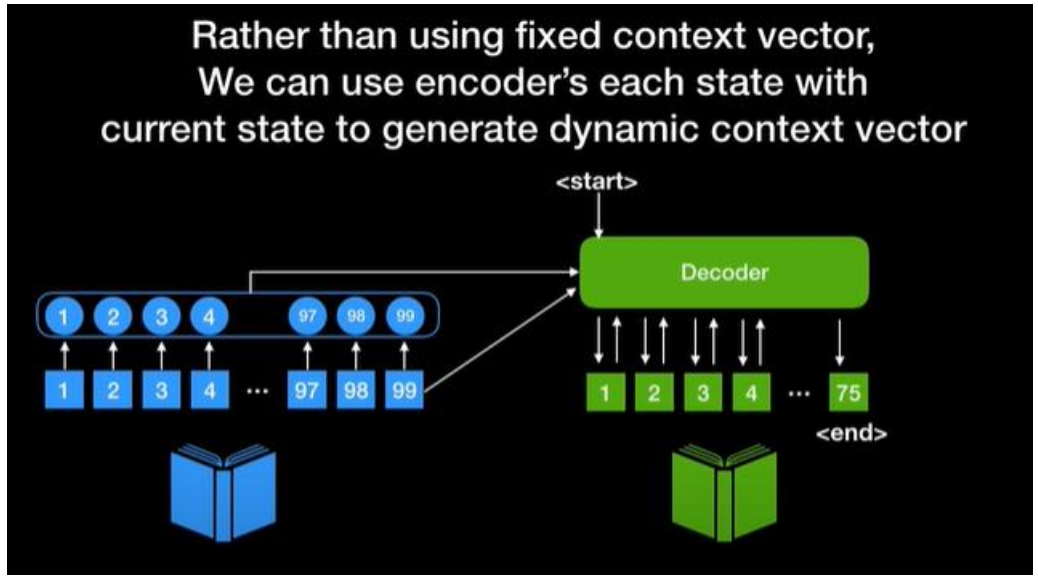
Attention 모델

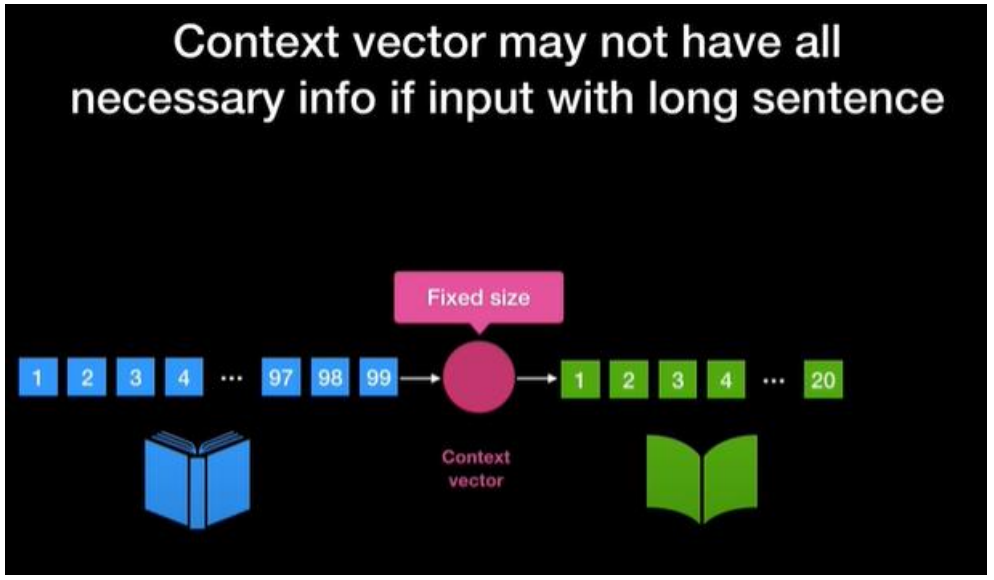
- 사람은 모든 sequence를 고려하면서 정보처리를 하는 것이 아니며, 중요한 feature를 더 중요하게 고려함
- Seq2Seq에서는 인코더 최종 output인 Context vector만 활용하지만, Attention에서는 인코더셀 각output을 활용
- Decoder에서는 매 스텝마다 인코더 셀의 output을 이용해 다이나믹 하게 Context vector를 생성



Attention 모델

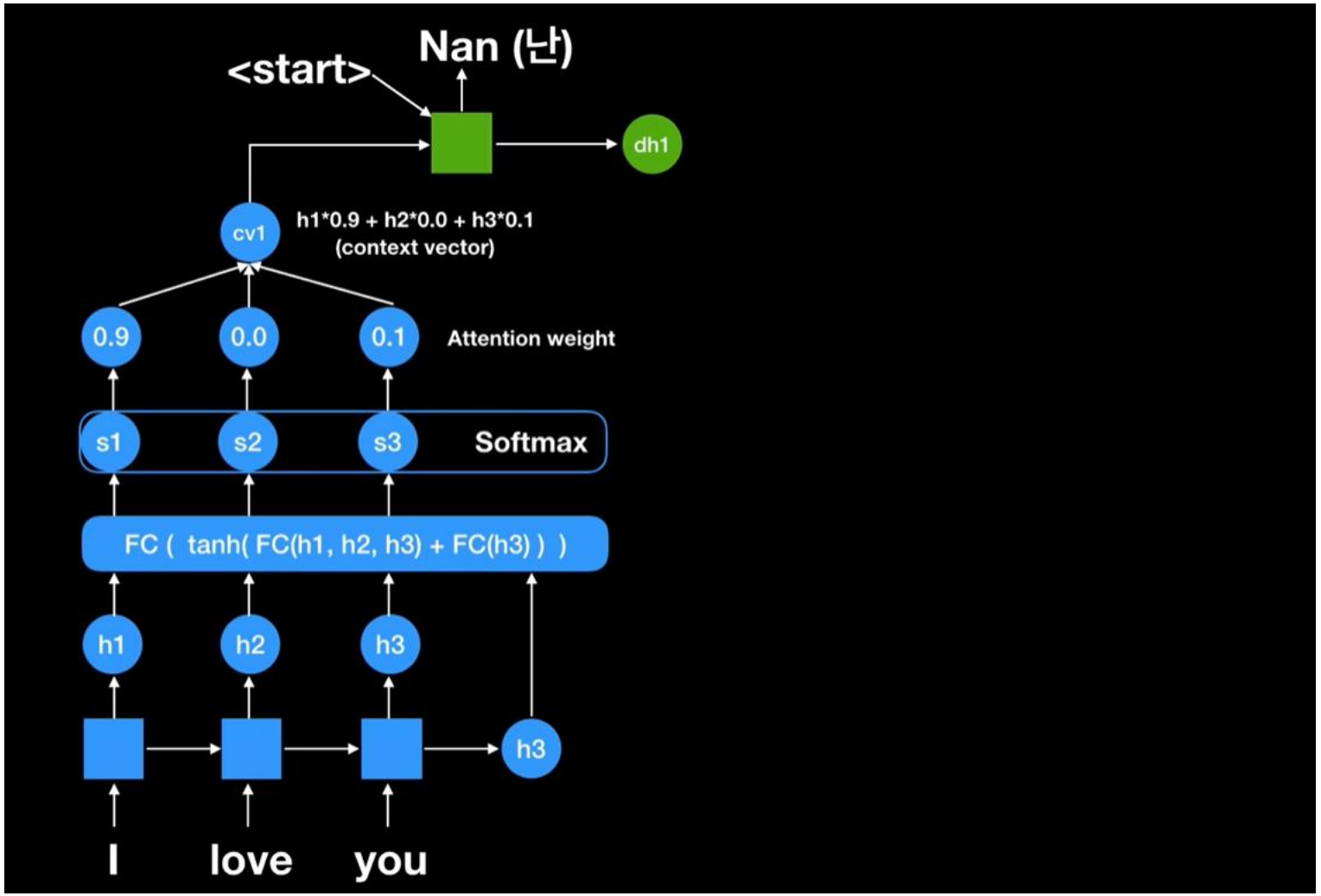






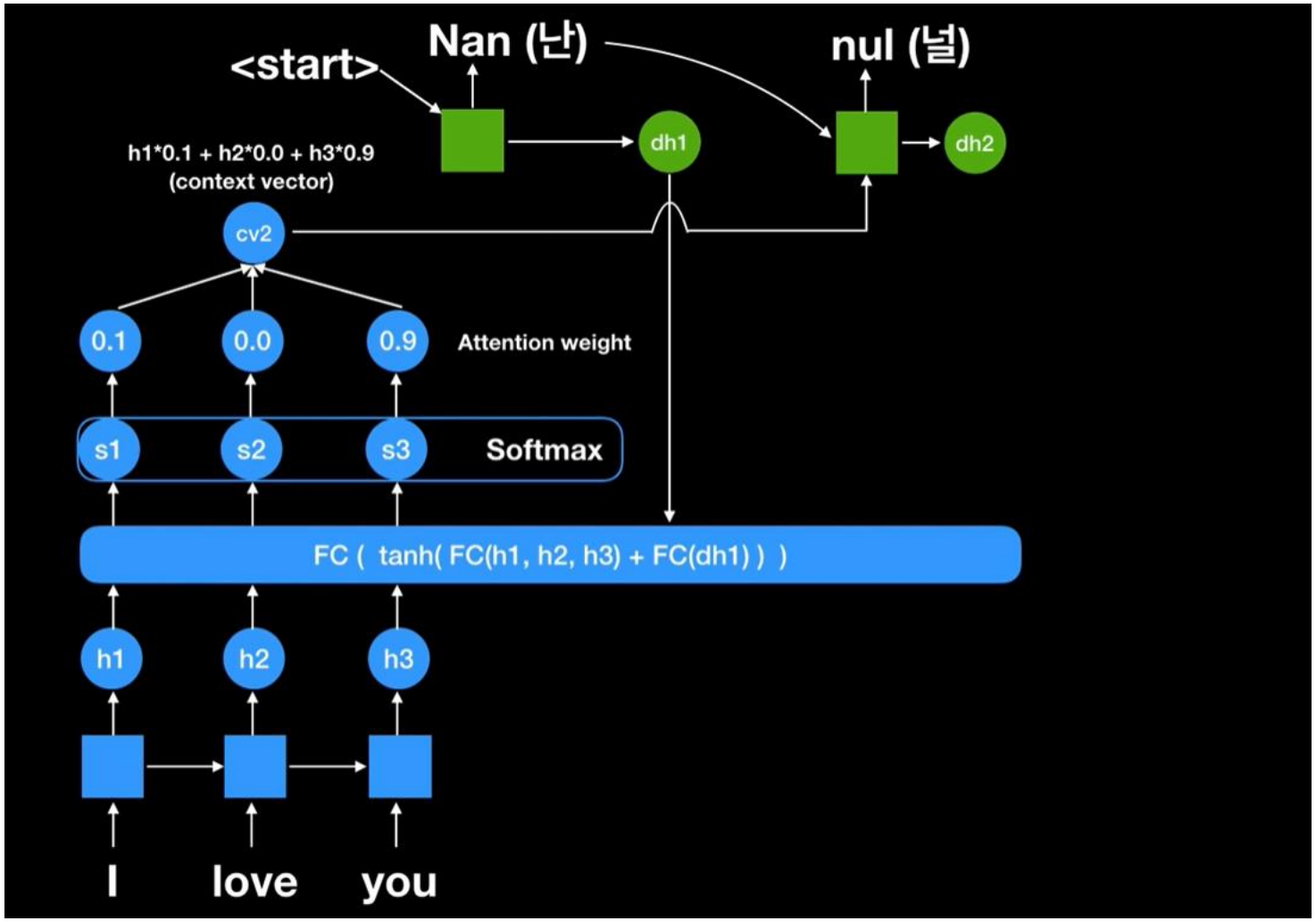
https://www.youtube.com/watch?v=WsQLdu2JMgI

Attention 모델



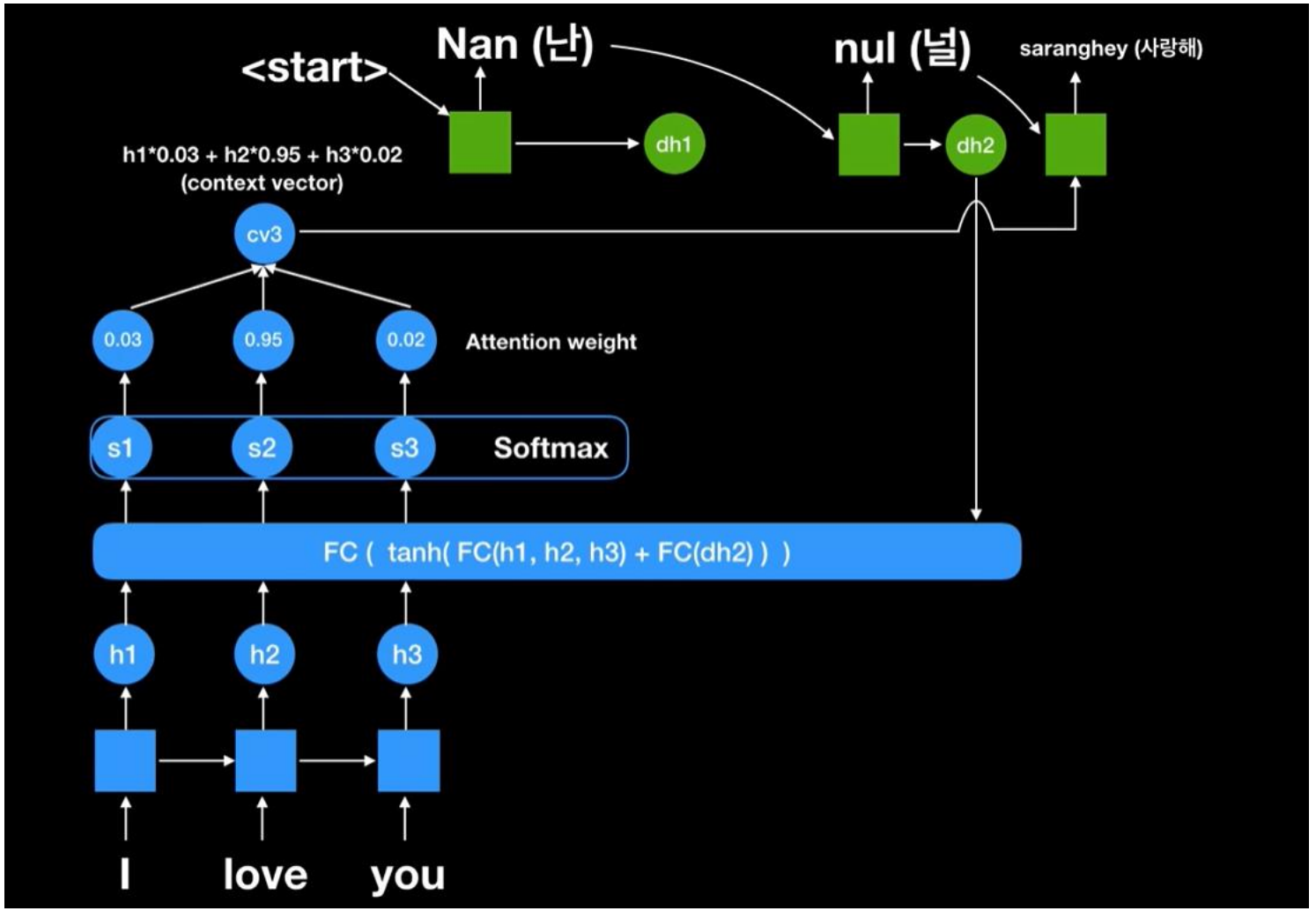
https://www.youtube.com/watch?v=WsQLdu2JMgI

Attention 모델



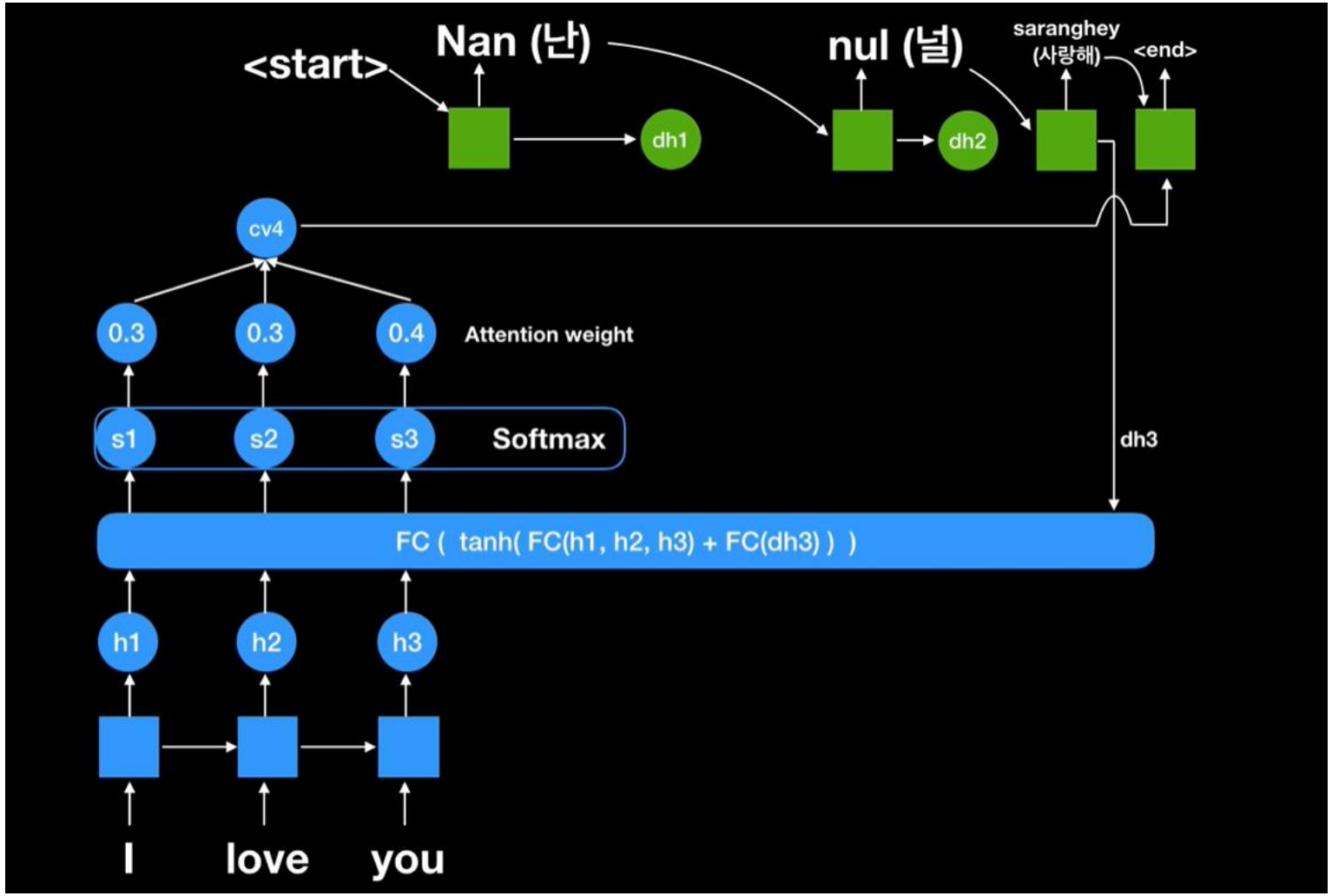
https://www.youtube.com/watch?v=WsQLdu2JMgI

Attention 모델



https://www.youtube.com/watch?v=WsQLdu2JMgI

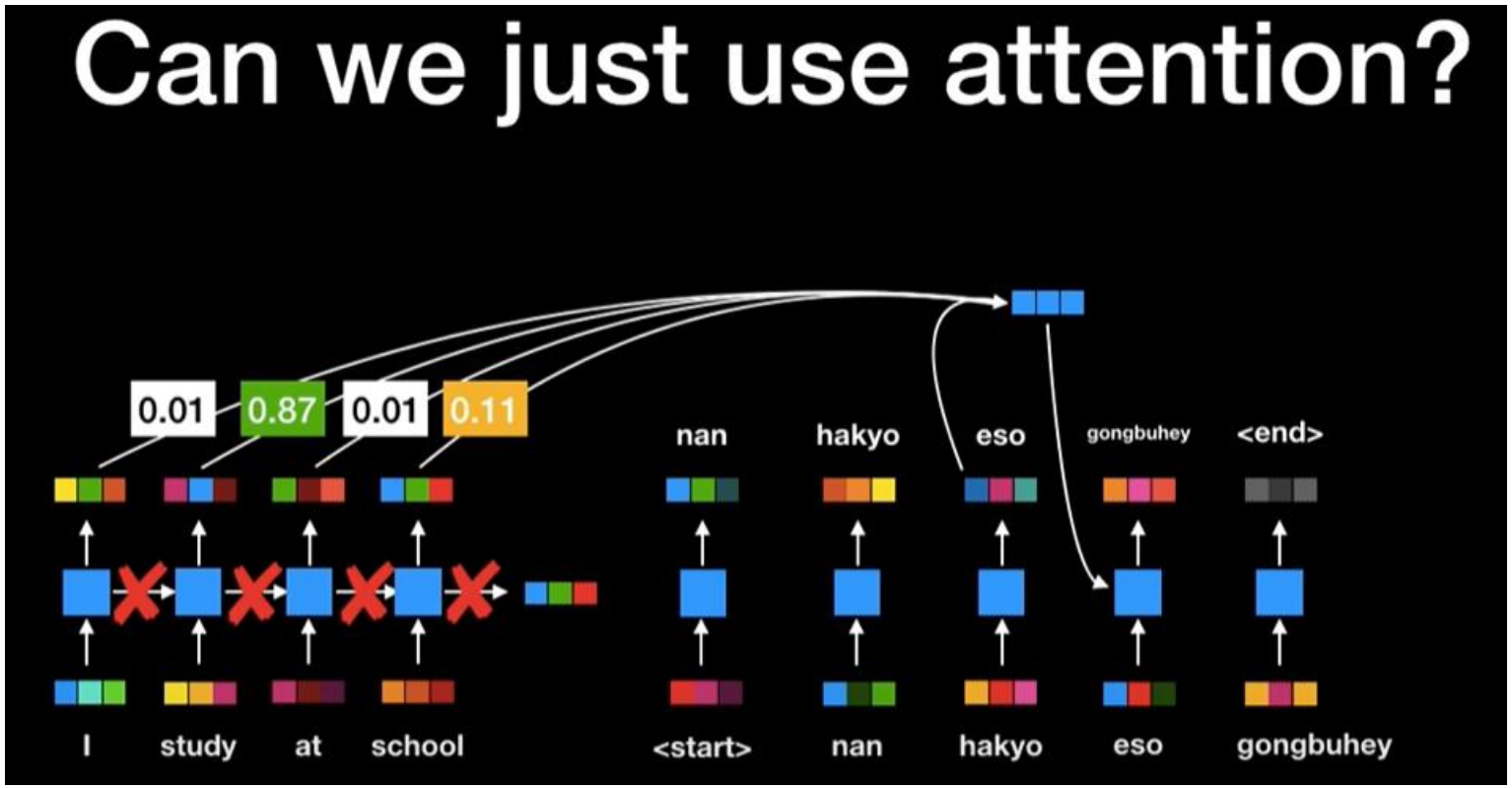
Attention 모델



https://www.youtube.com/watch?v=WsQLdu2JMgI

Attention 모델

- Attention 모델은 Seq2Seq의 Encoder, Decoder 성능을 향상시켰으나, RNN 순차 연산으로 연산 속도가 느림
- RNN 계열은 hi계산을 위하여 $hi-1$ 가 필요하고, 순차적인 계산으로 병렬화(parallelization)가 안 됨
- LSTM도 long dependency 가 잘 학습되지 않음



https://www.youtube.com/watch?v=mxGCEWOxfe8

Transformer

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszk Google Rese usz@google

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

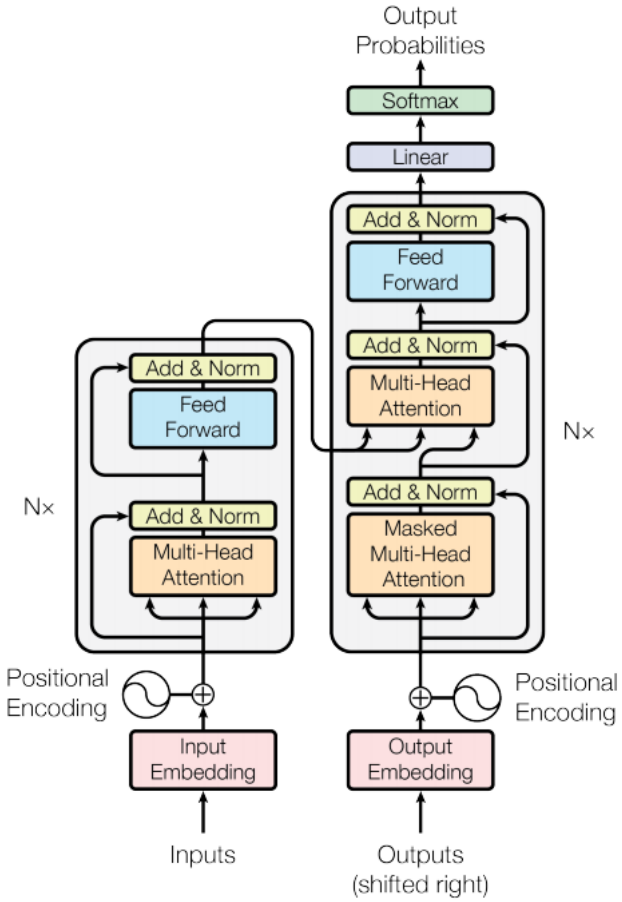
Łukasz Kaiser* Google Brain lukaszkaiser@google.c

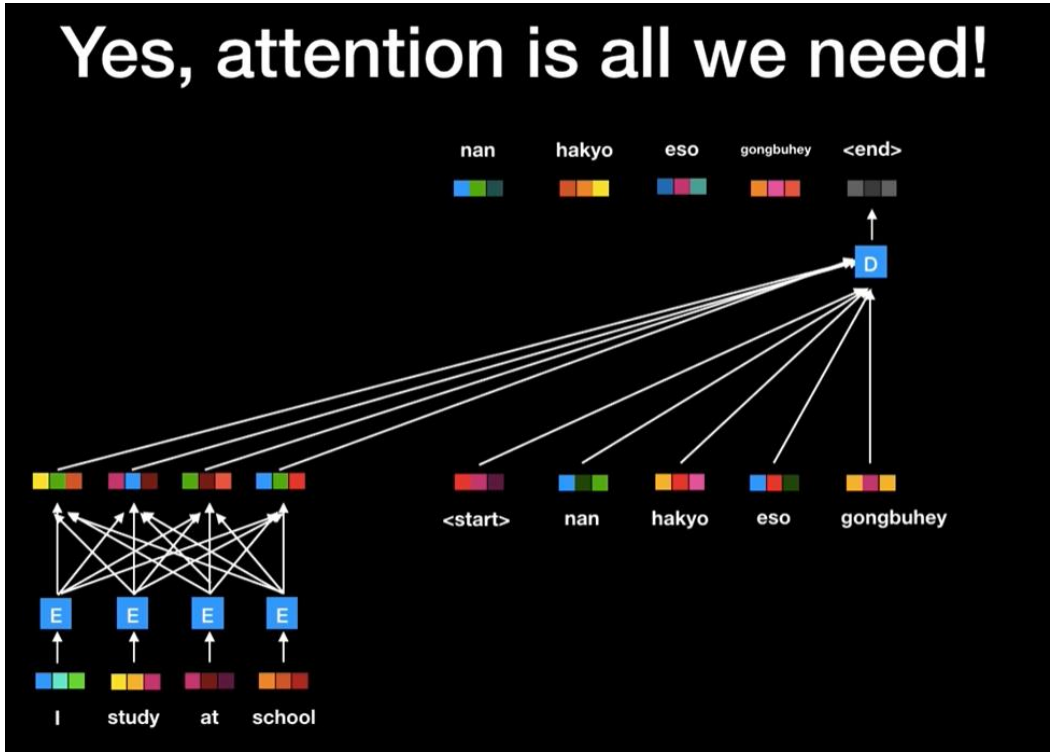
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

Introduction

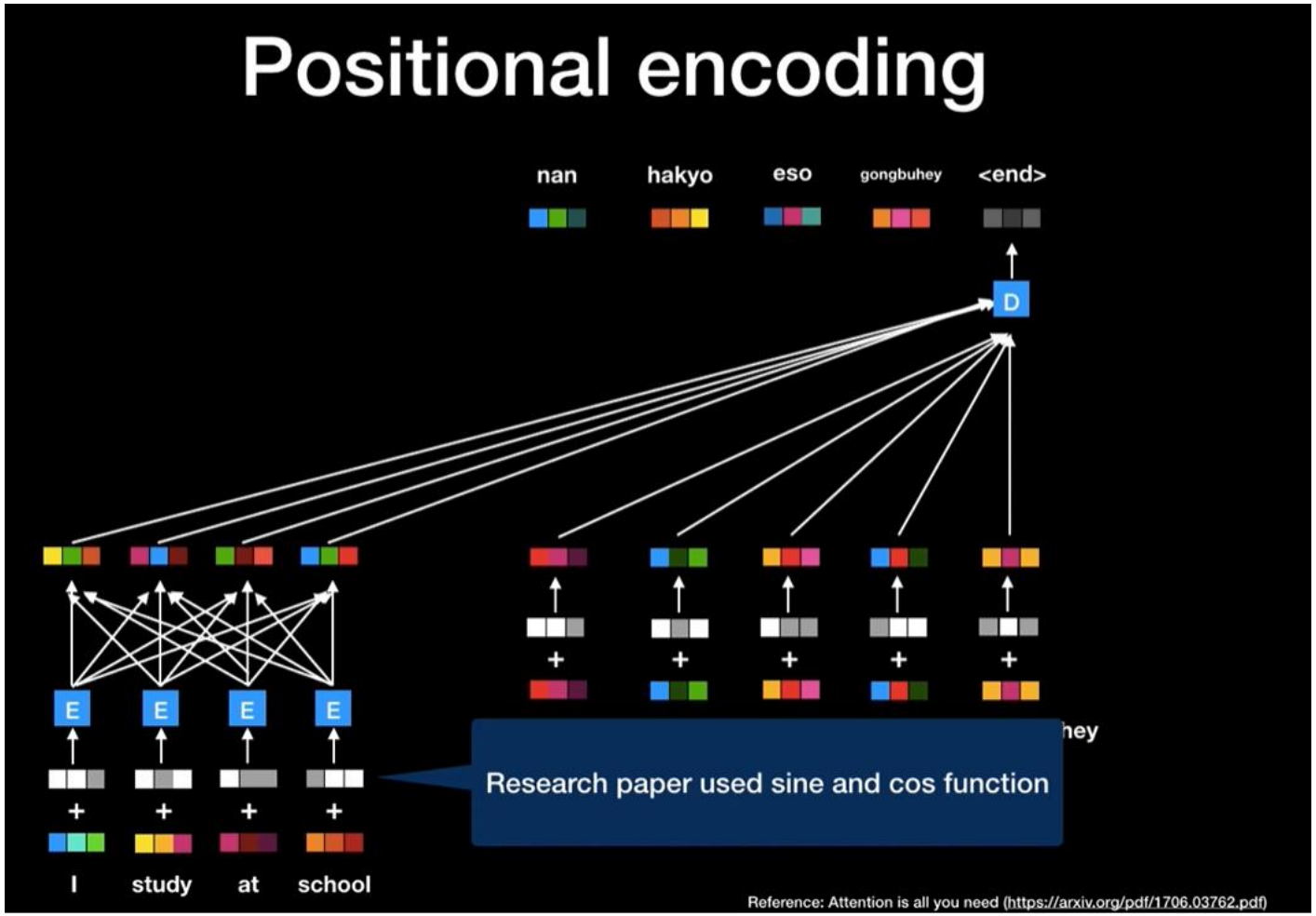




https://arxiv.org/pdf/1706.03762.pdf

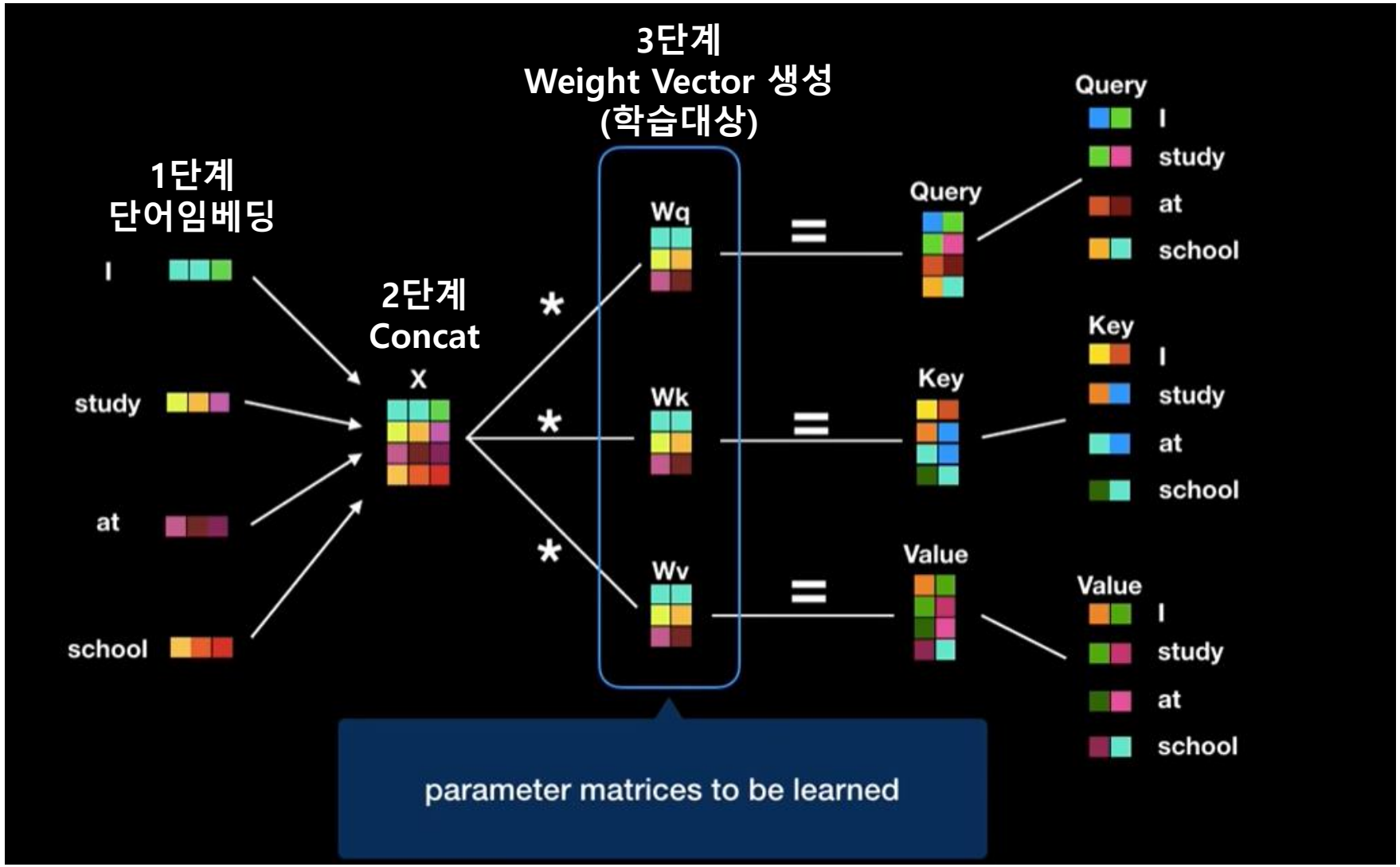
https://www.youtube.com/watch?v=mxGCEWOxfe8

Self Attention 모델



https://www.youtube.com/watch?v=mxGCEWOxfe8

Self Attention 모델



Input 문장의 token을
3 종류의 vector로 표현

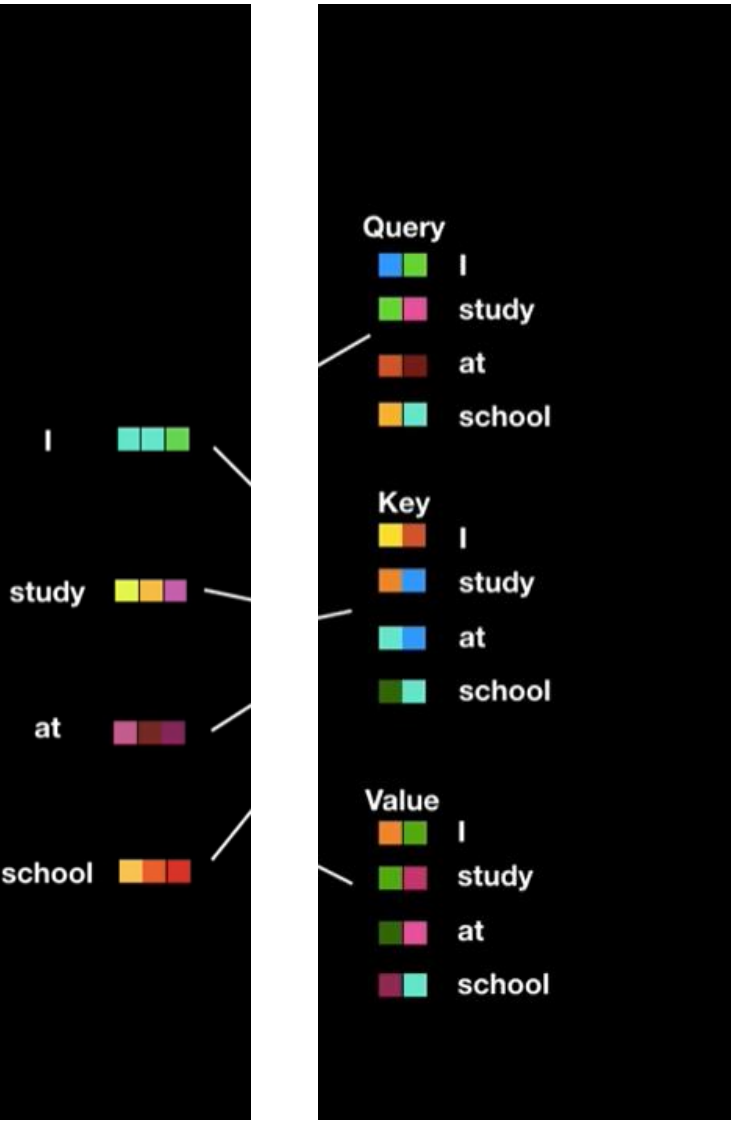
얻고자 하는
Context Vector

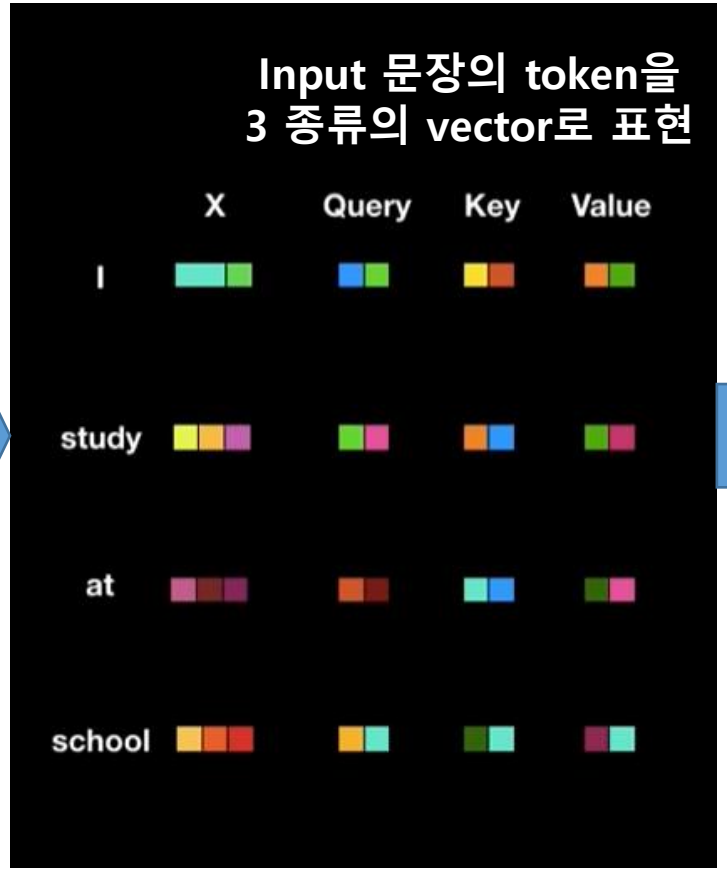
Query와의 상관관계를
나타내고자 하는 대상

Query와 Key의
상관관계 값

https://www.youtube.com/watch?v=mxGCEWOxfe8

Self Attention 모델





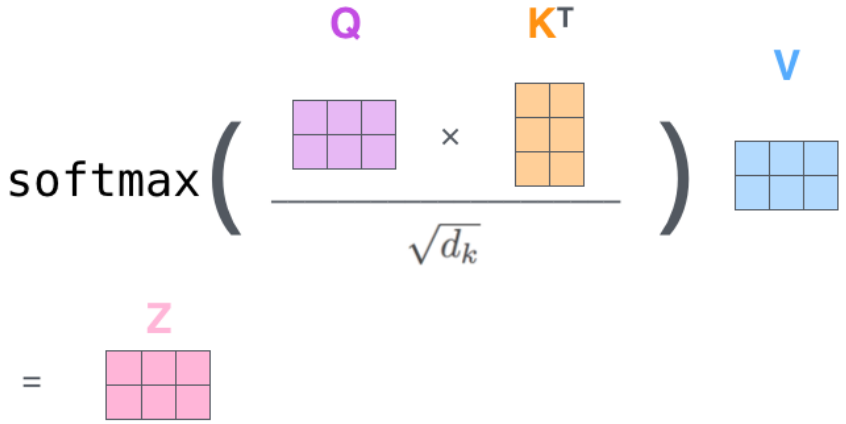
얻고자 하는
Context Vector

	Query * Key^T	Score	Softmax	Value
I	I * I	= 130	0.92	I
study	I * study	= 50	0.05	study
at	I * at	= 20	0.02	at
school	I * school	= 10	0.01	school

https://www.youtube.com/watch?v=mxGCEWOxfe8

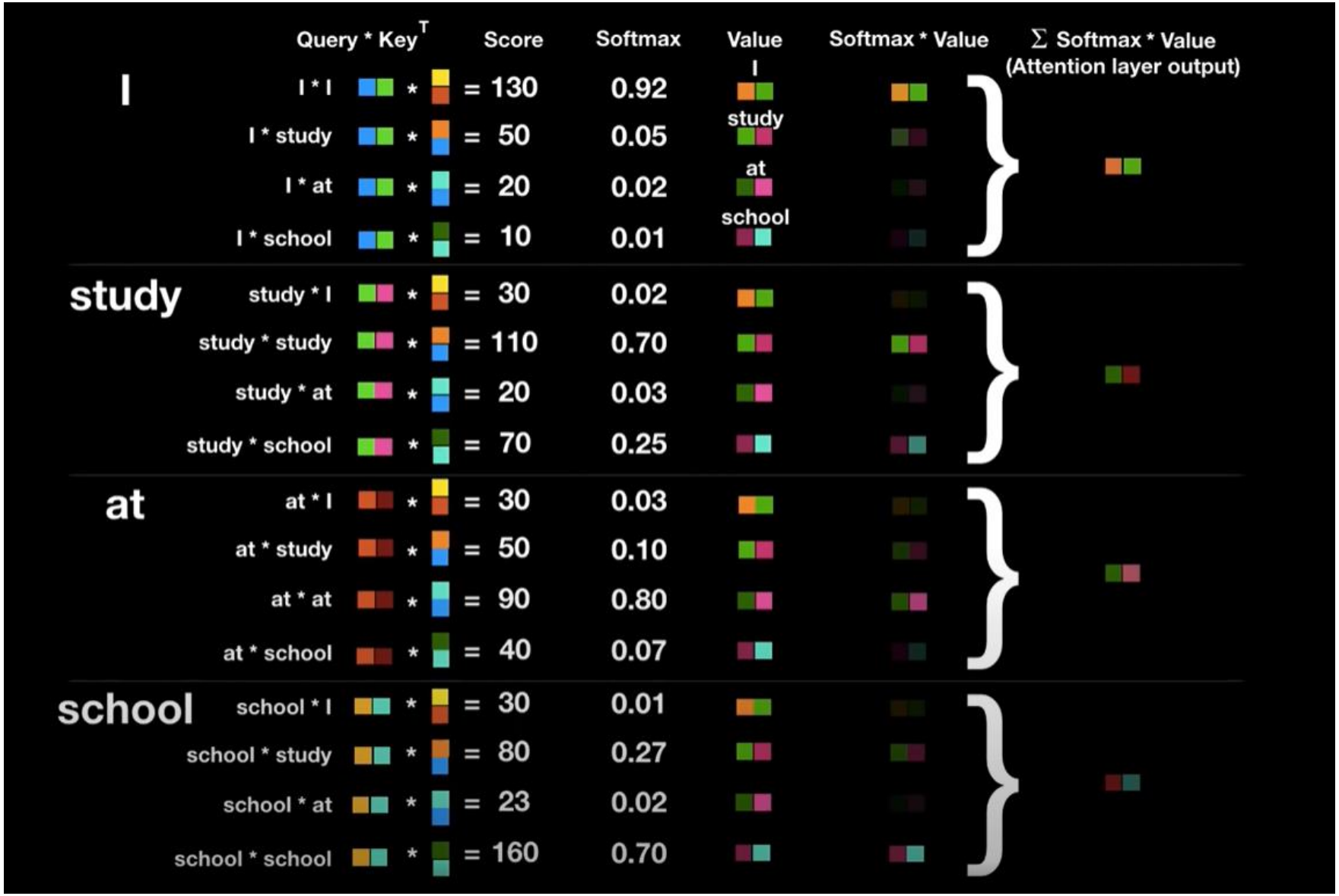
Self Attention 모델

self-attention 계산



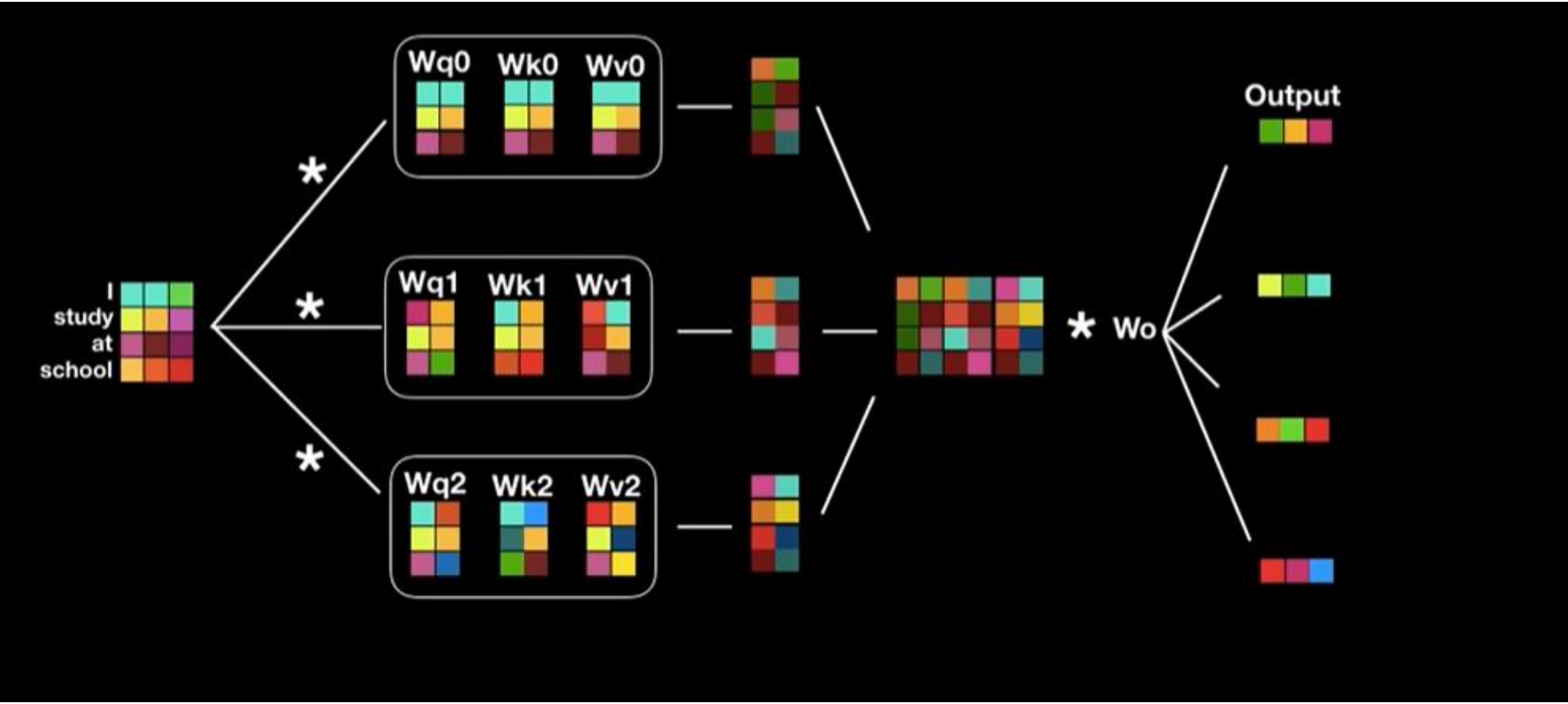
https://www.youtube.com/watch?v=mxGCEWOxfe8

Self Attention 모델



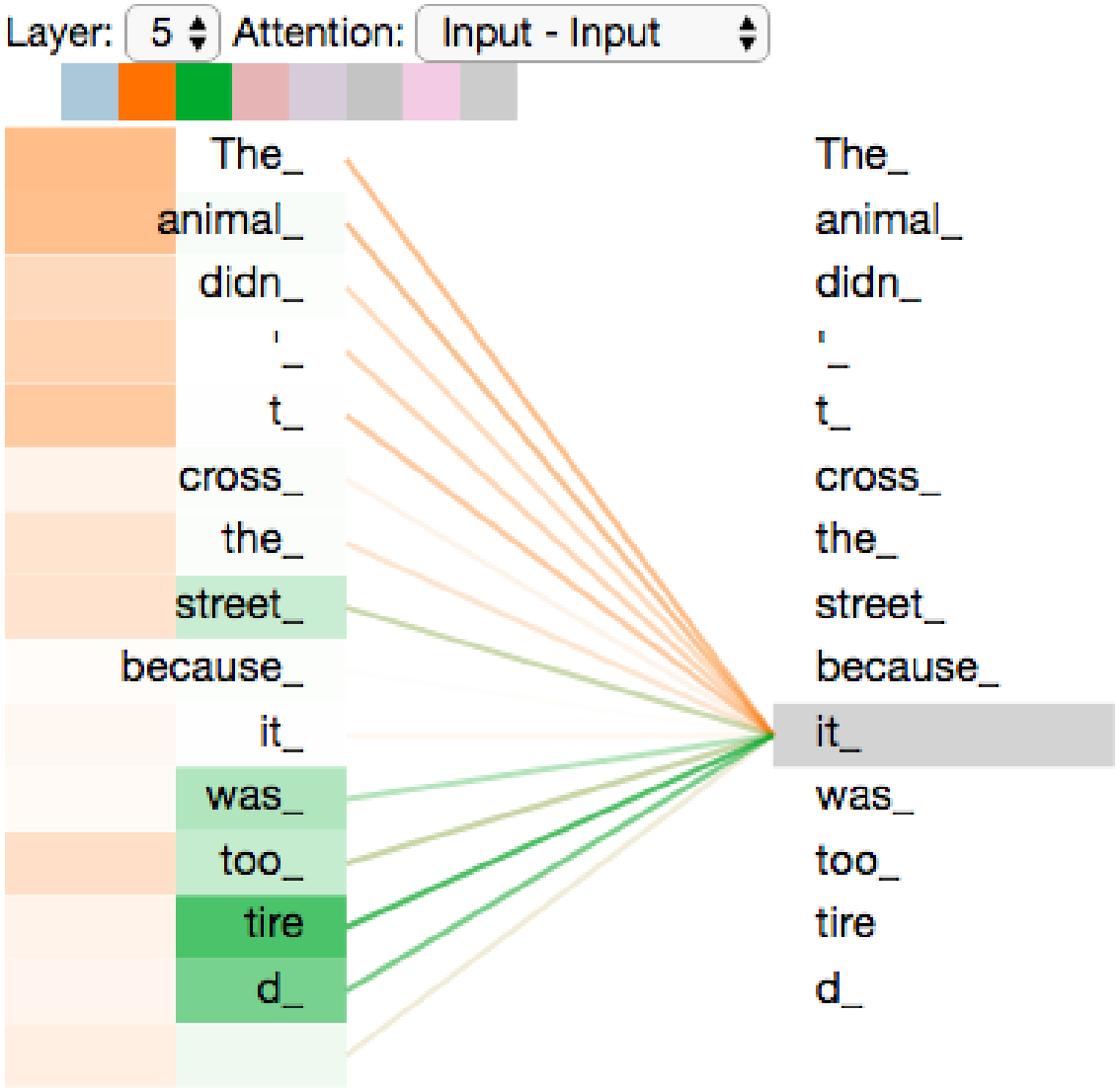
https://www.youtube.com/watch?v=mxGCEWOxfe8

Multi Head Self Attention 모델



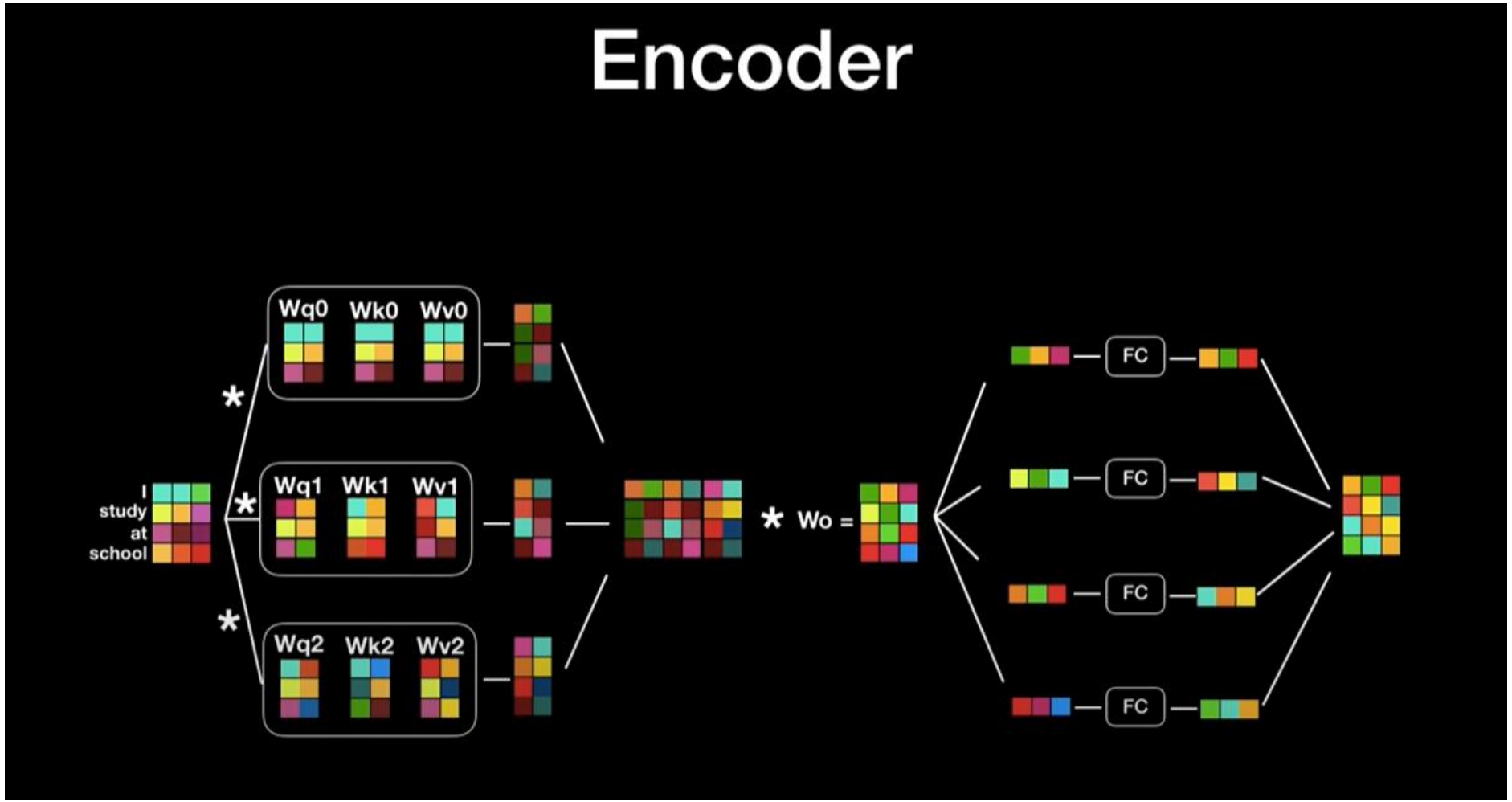
https://www.youtube.com/watch?v=mxGCEWOxfe8

Multi Head Self Attention 모델



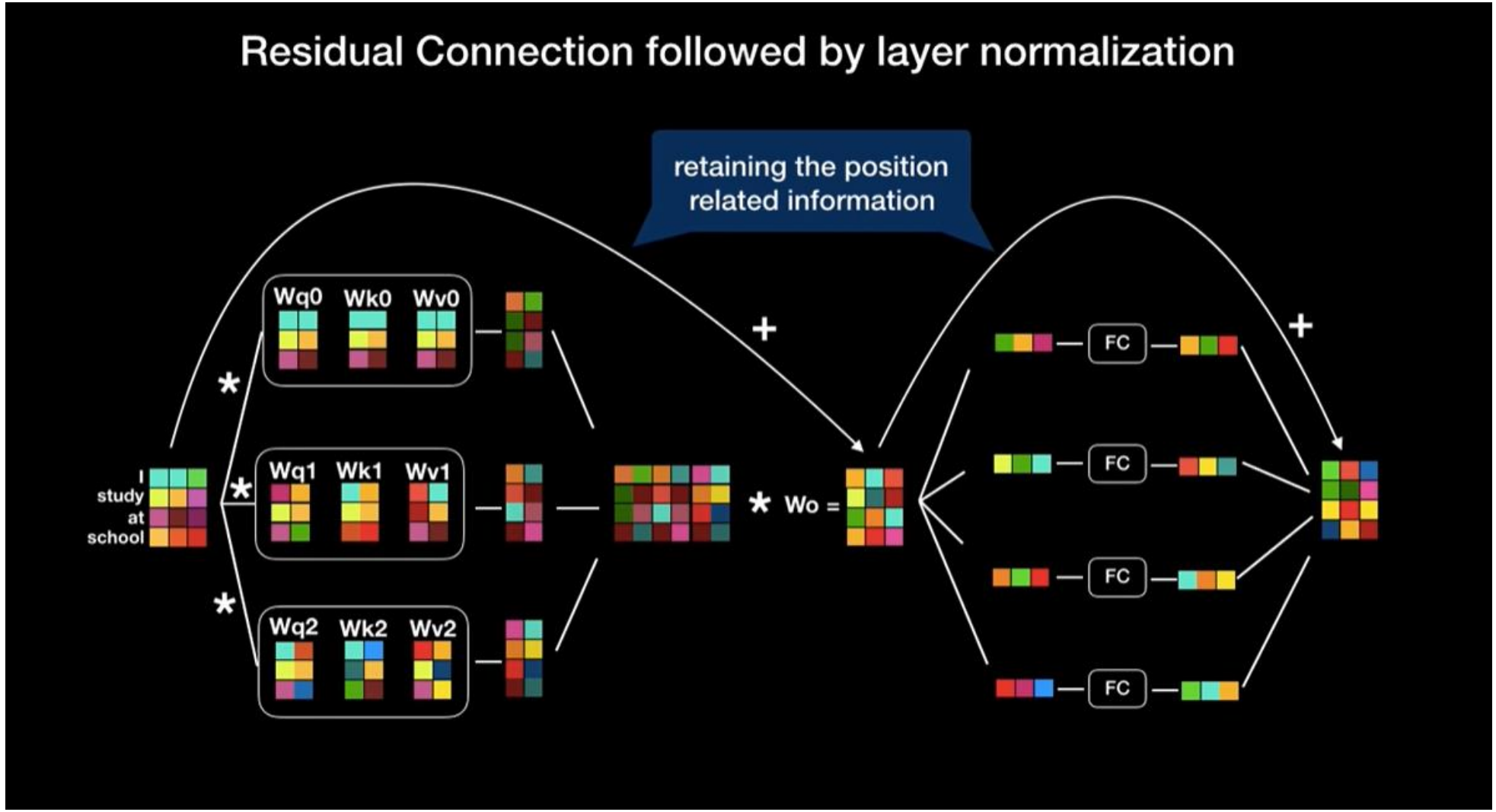
https://nlpinkorean.github.io/illustrated-transformer/

Transformer Encoder Layer



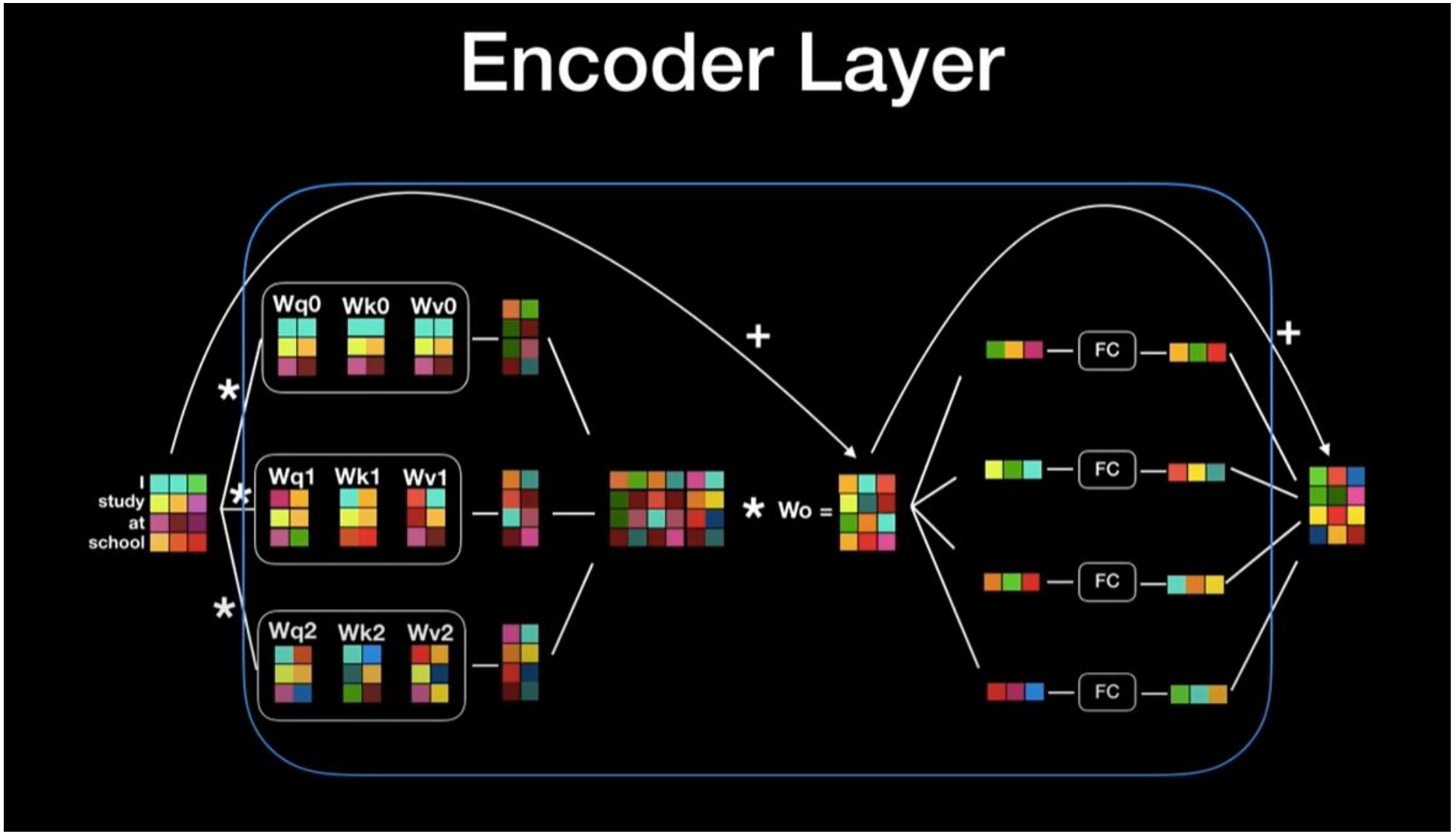
https://www.youtube.com/watch?v=mxGCEWOxfe8

Transformer Encoder Layer



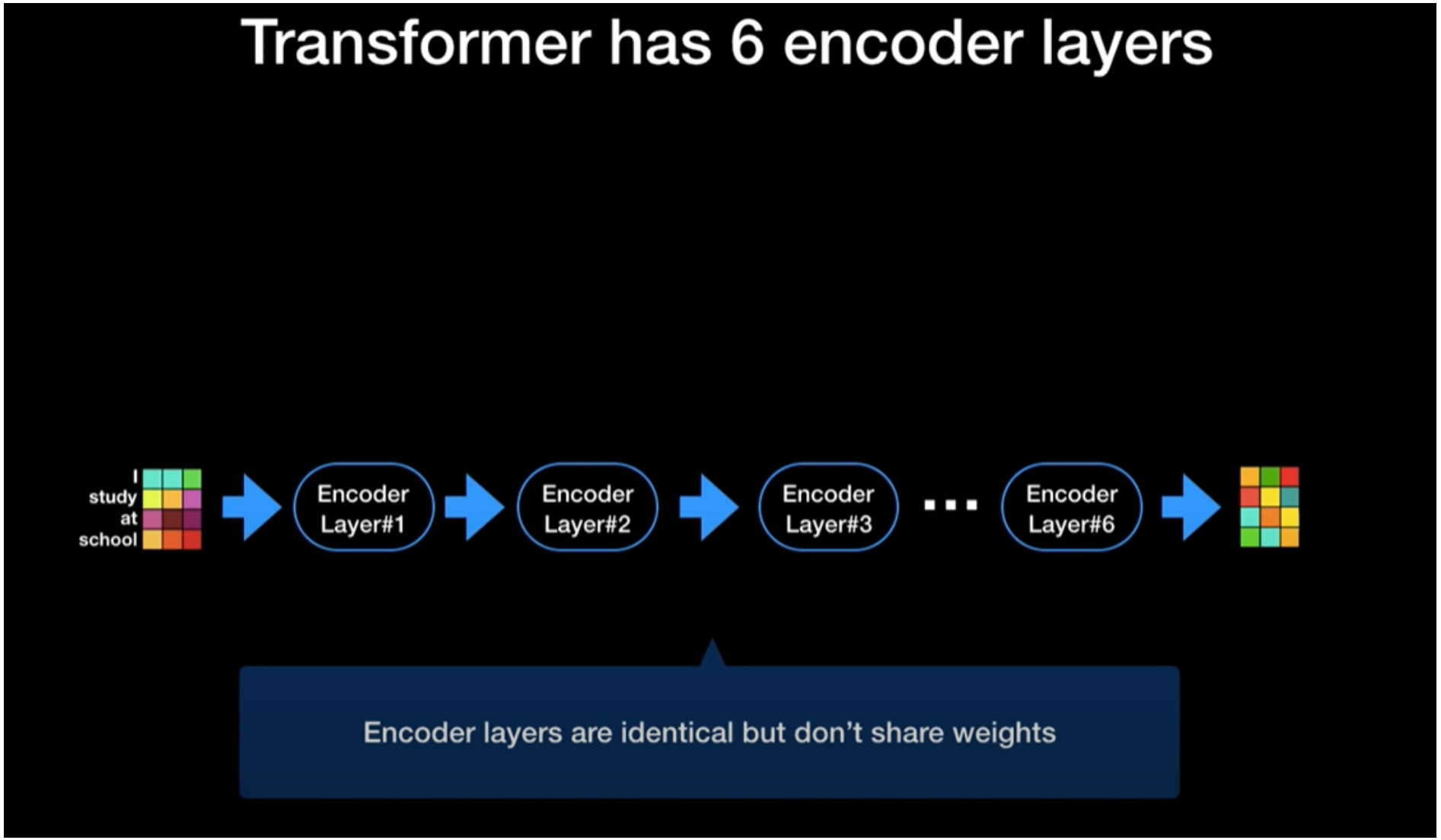
https://www.youtube.com/watch?v=mxGCEWOxfe8

Transformer Encoder Layer



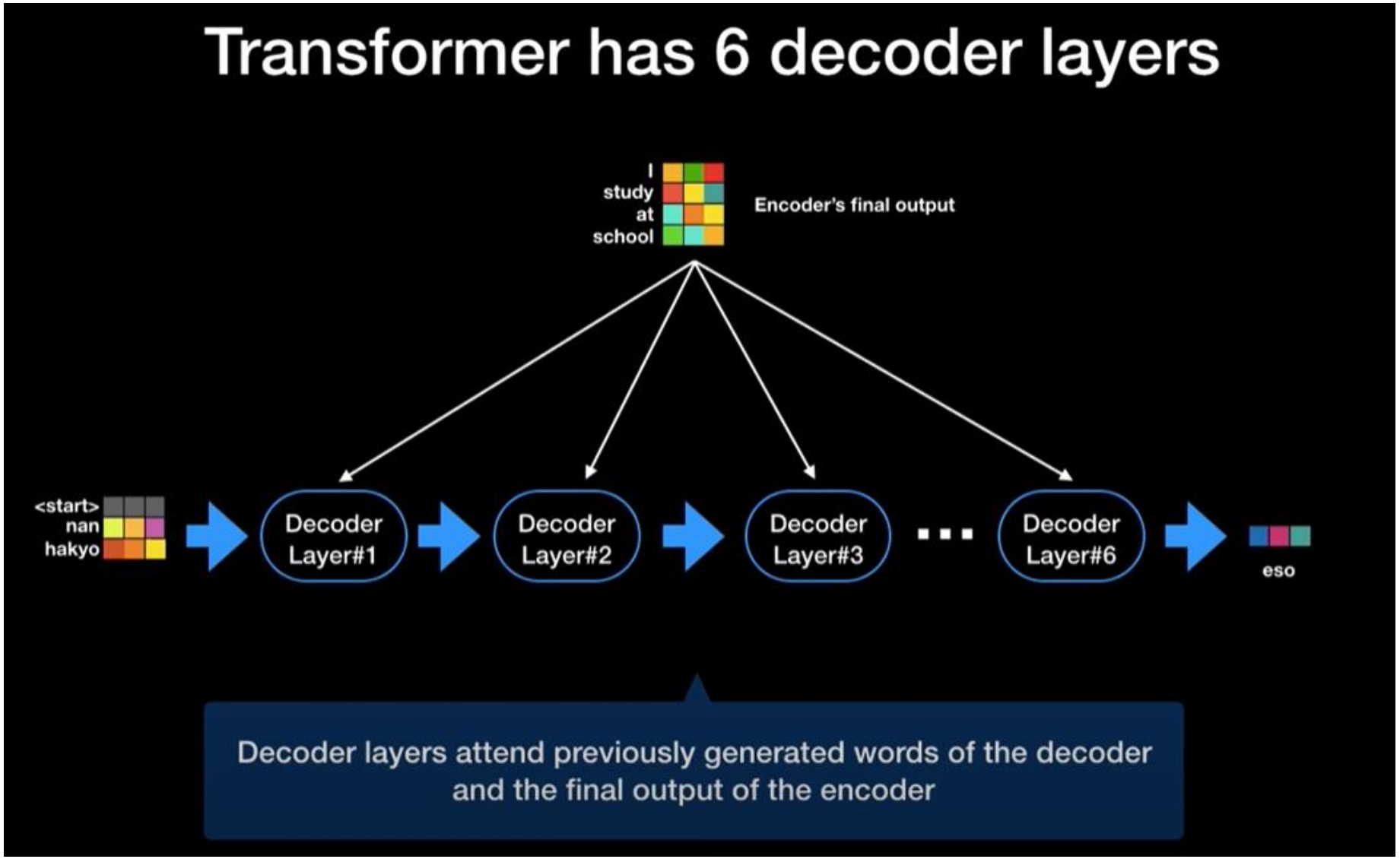
https://www.youtube.com/watch?v=mxGCEWOxfe8

Transformer Encoder Layer



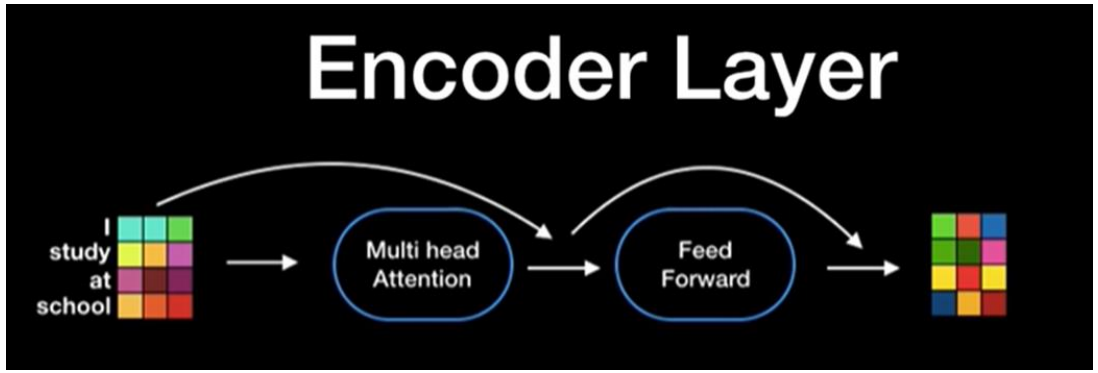
https://www.youtube.com/watch?v=mxGCEWOxfe8

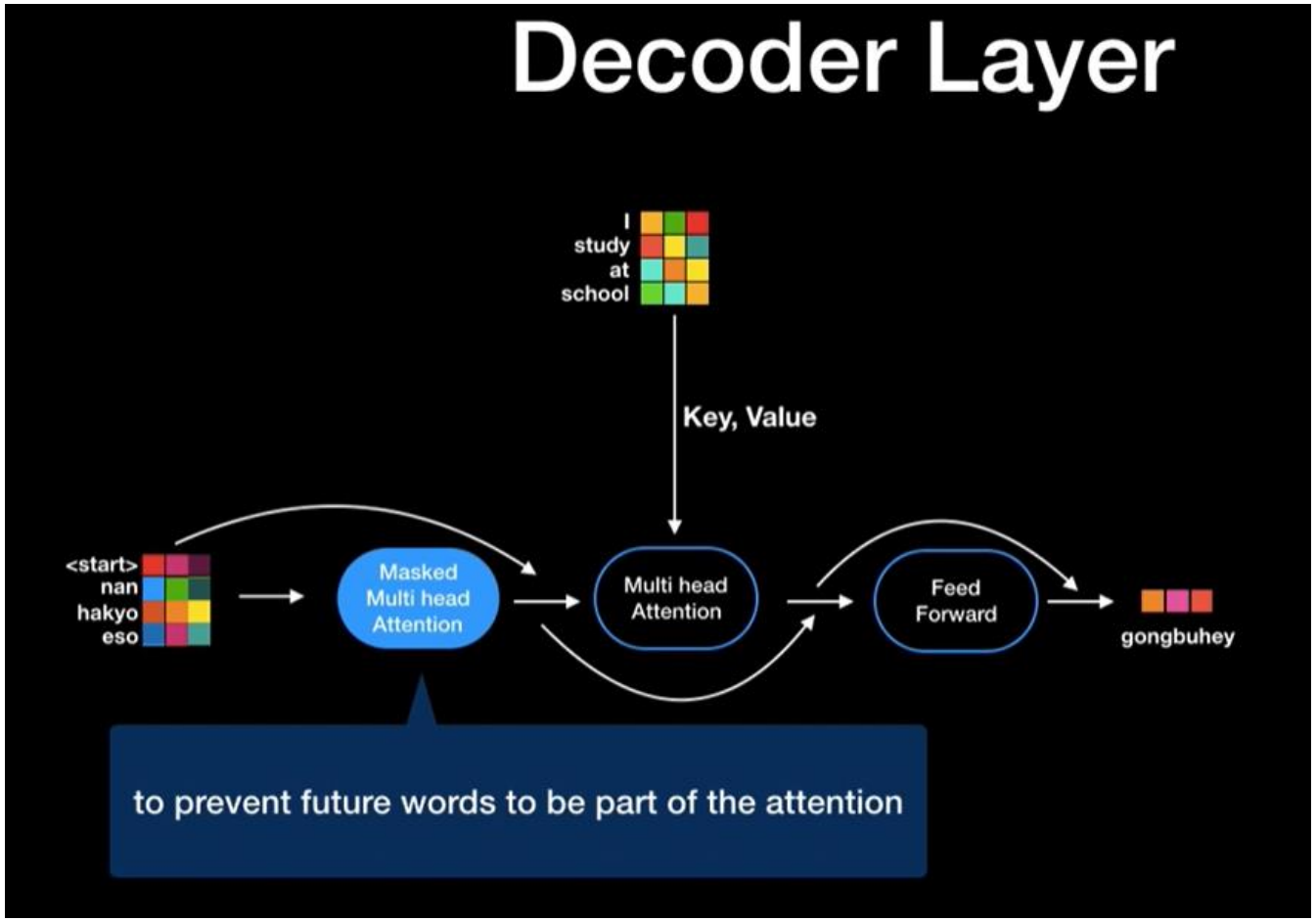
Transformer Decoder Layer



https://www.youtube.com/watch?v=mxGCEWOxfe8

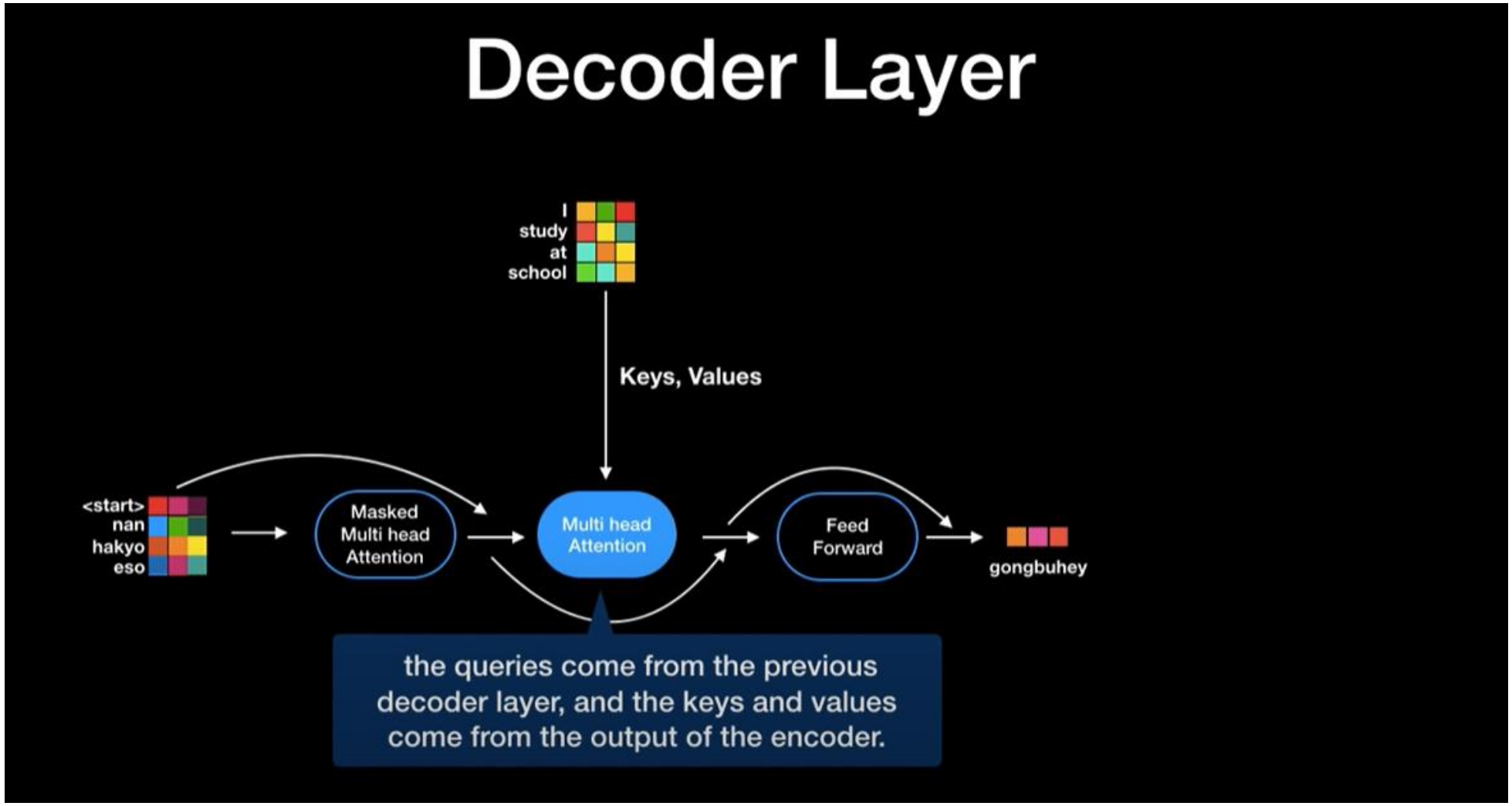
Transformer Decoder Layer





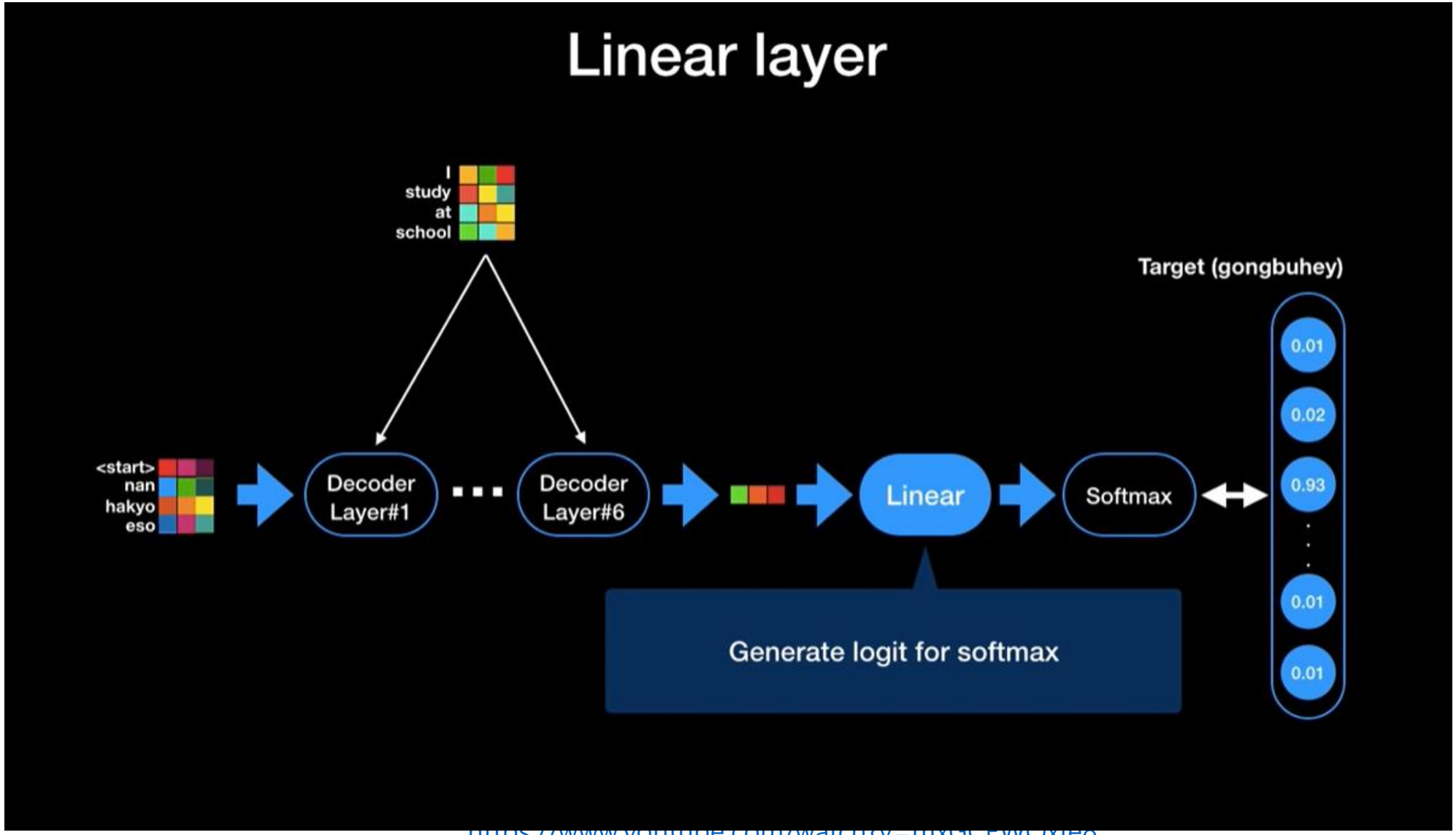
https://www.youtube.com/watch?v=mxGCEWOxfe8

Transformer Decoder Layer



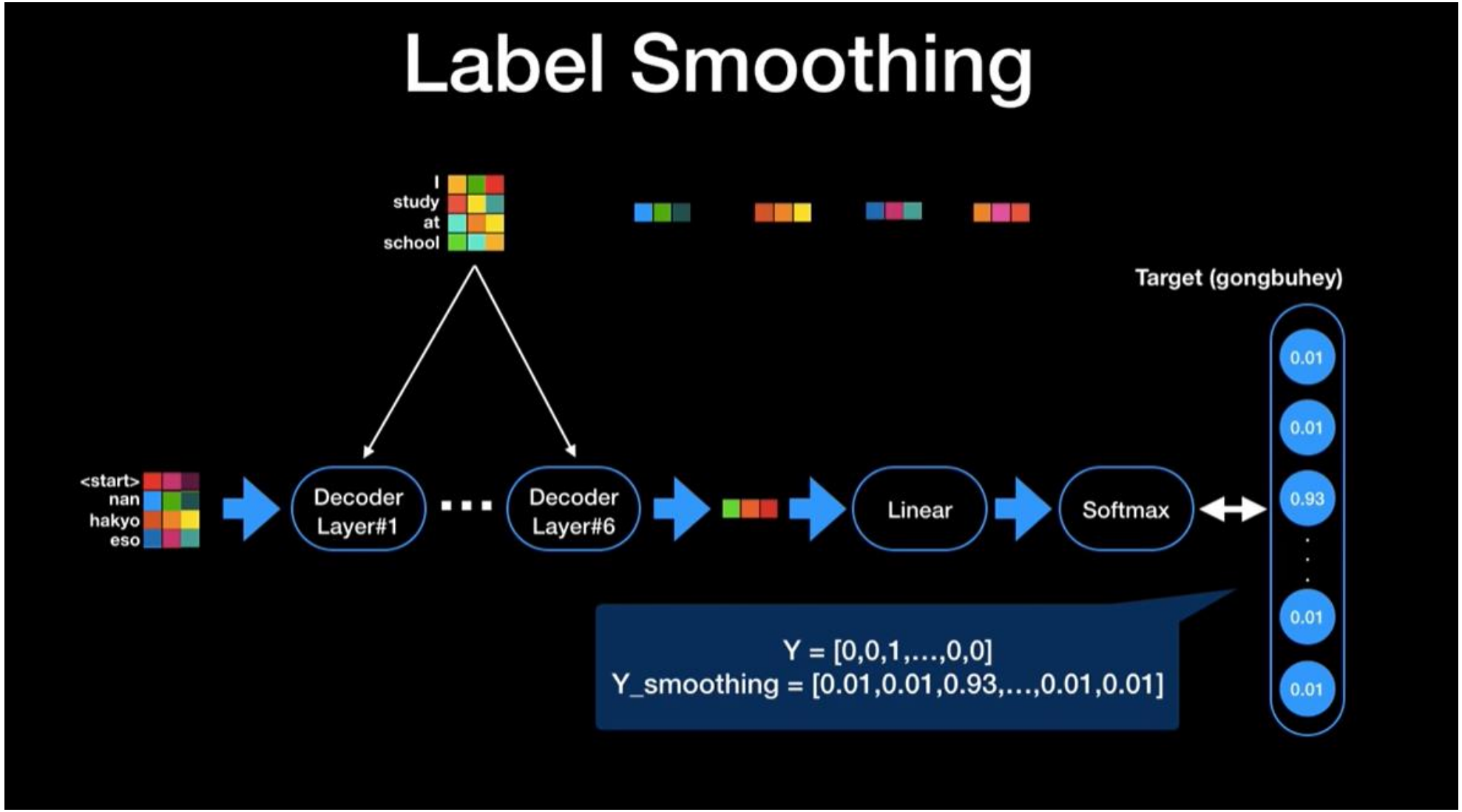
https://www.youtube.com/watch?v=mxGCEWOxfe8

Transformer Decoder Layer



https://www.youtube.com/watch?v=mxGCEWOxfe8

Transformer Decoder Layer



https://www.youtube.com/watch?v=mxGCEWOxfe8

Transformer 참고 자료

Transformer 모델 시각화

https://nlpinkorean.github.io/illustrated-transformer/

어텐션 메커니즘과 transfomer(self-attention)

https://bit.ly/3vreqwi

SEQUENCE TO SEQUENCE 네트워크와 ATTENTION을 이용한 번역

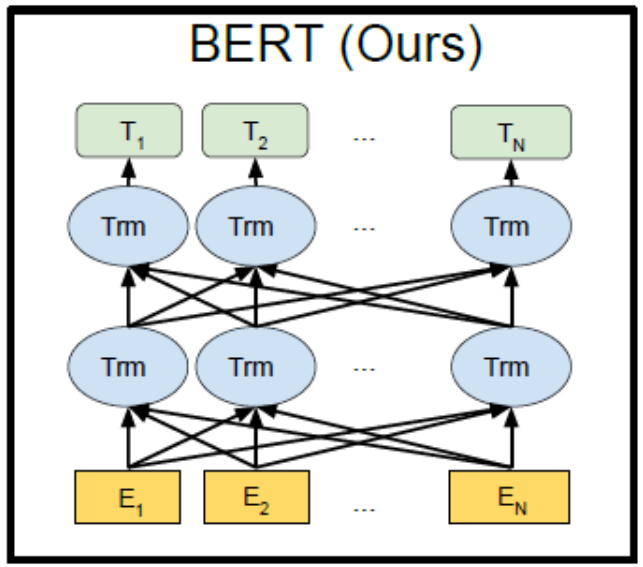
https://tutorials.pytorch.kr/intermediate/seq2seq_translation_tutorial.html

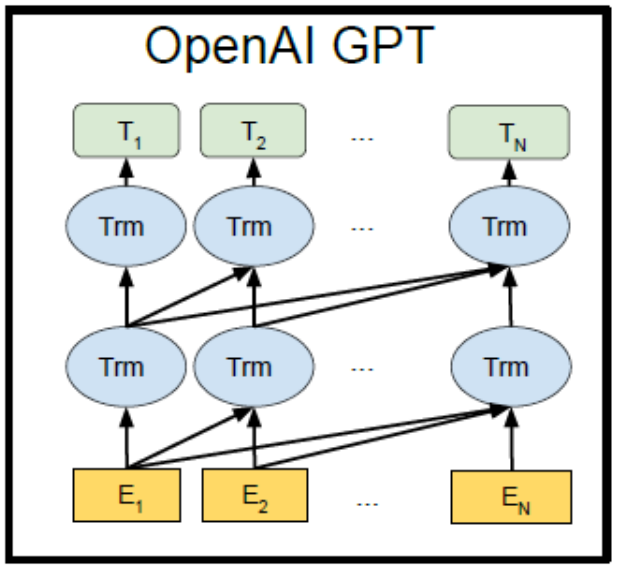
TRANSFORMER 와 TORCHTEXT 로 시퀀스-투-시퀀스(SEQUENCE-TO-SEQUENCE) 모델링하기

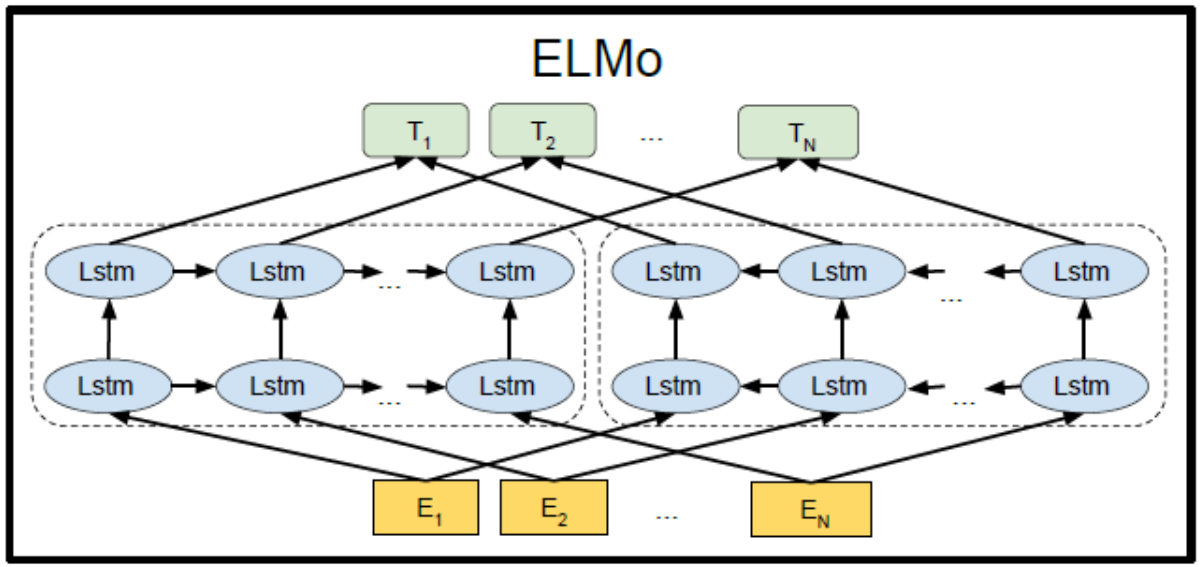
https://tutorials.pytorch.kr/beginner/transformer_tutorial.html

BERT

- BERT(Bidirectional Encoder Representations from Transformers) 는 wiki, book data와 같은 대용량 데이터로 모델을 사전 훈련(pre-train) 시킨 후, 특정 태스크를 가지고 있는 labeled data로 전이학습(transfer learning)을 하는 모델
- BERT의 성공비결은 성능이 검증된 Transformer Encoder 블록을 사용하고, 모델의 속성이 양방향을 지향한다는 점
- GPT(Generative Pre-Training)는 단 방향 아키텍처이고 ELMo 는 Bi-LSTM 상단은 양방향이지만 중간 레이어는 단방향

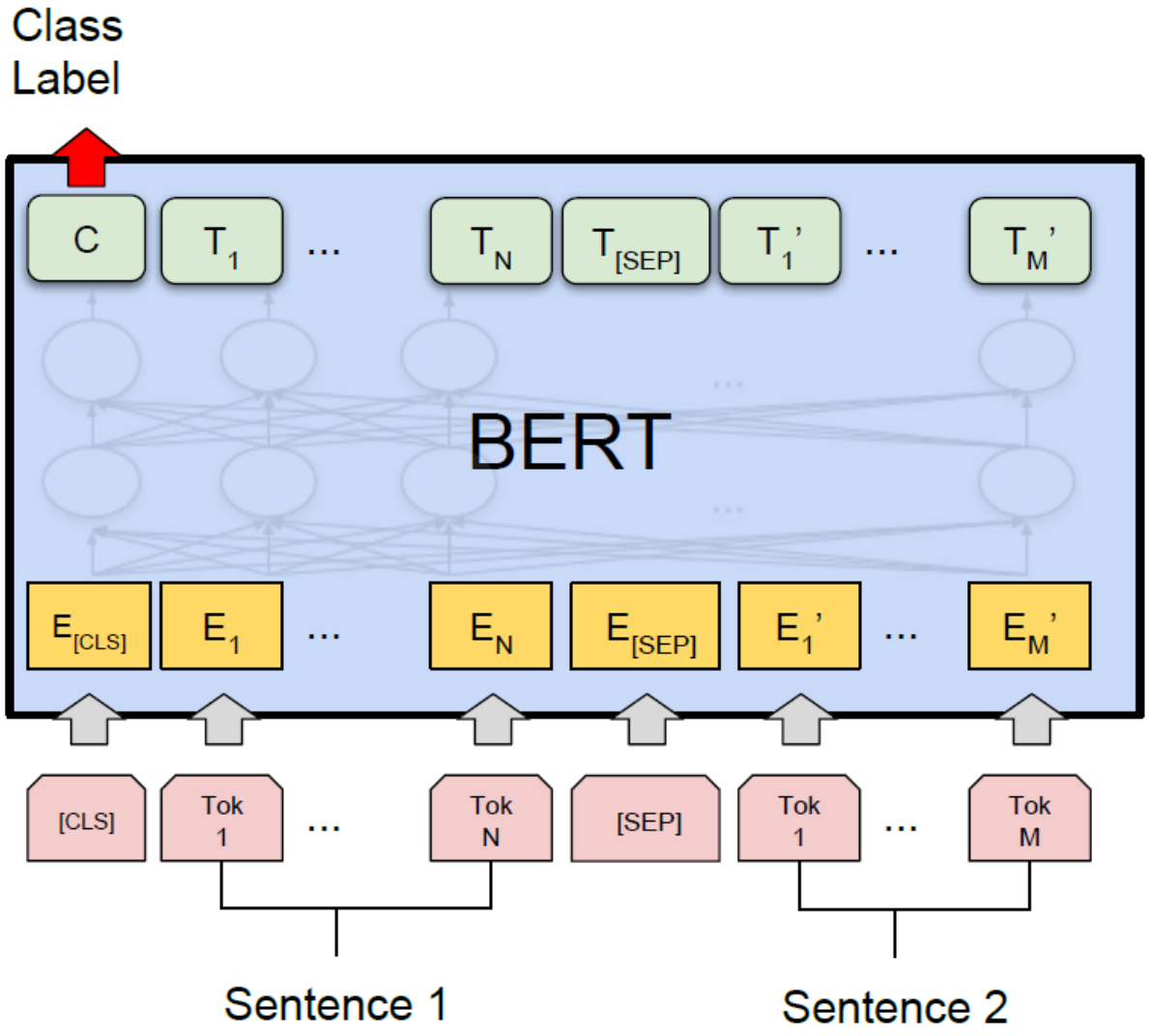






https://arxiv.org/pdf/1810.04805.pdf

BERT 모델 구조



Contextual representations of token

Transformer layer

Input embedding layer

구분	BERT_base	BERT_large
Layer	12	24
Hidden size	768	1024
Self-attention Heads	12	16
Total Parameters	110M	340M

https://arxiv.org/pdf/1810.04805.pdf

BERT 프리트레인(Pre-train) 데이터

■ 사전 훈련 데이터(대용량 코퍼스)

- Books Corpus (800M words) (Zhu et al.,2015)
- English Wikipedia (2,500M words).

https://googlebooks.byu.edu/

8억 단어

https://corpus.byu.edu/wiki/

25억 단어

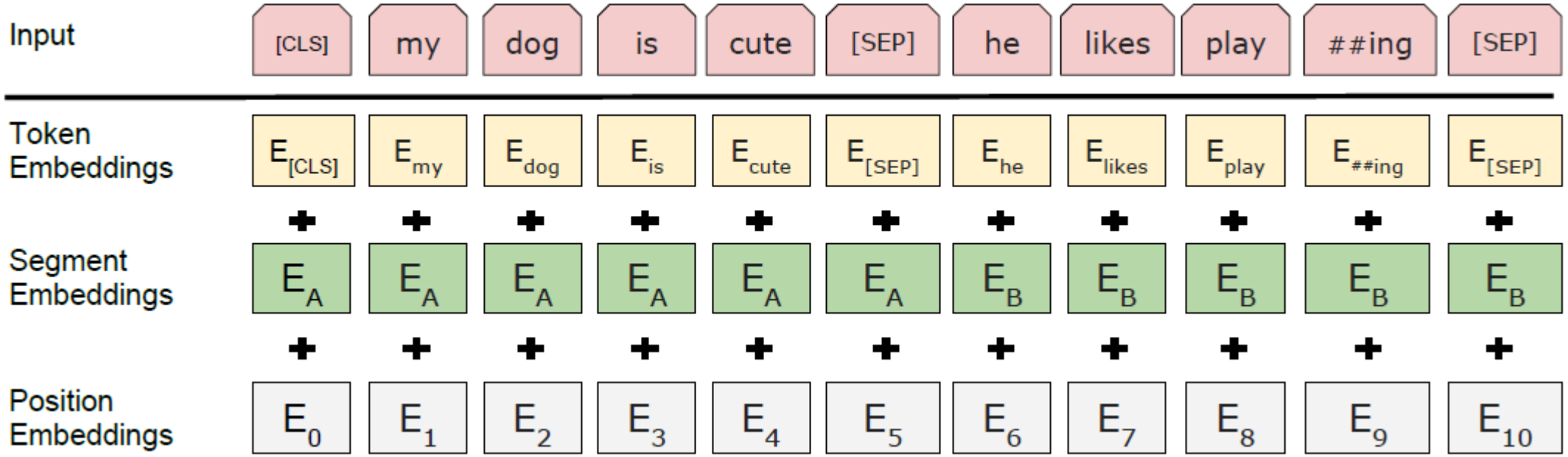
(리스트, 표, 헤더 등 모두 무시하고 텍스트만 추출)

■ Tokenizing : WordPiece

- 단어보다 더 작은 단위로 쪼개는 서브워드 토크나이저 WordPiece 사용
  He likes playing a He likes play **##ing**
- 글자로부터 서브워드들을 병합해가는 방식으로 Vocab을 만드는 것으로 BPE(Byte Pair Encoding)와 유사
- 자주 등장하지 않는 단어는 더 작은 단위인 서브워드로 분리되어 서브워드들을 Vocab에 추가

BERT 입력

- BERT의 입력은 토큰 임베딩, 세그먼트 임베딩, 포지션 임베딩 3개의 합으로 구성됩니다.
- 입력 문장(sentence)의 첫번째 토큰은 [CLS](special classification token) 입니다.
- [CLS] 토큰은 transformer 전체층을 거치고 나면 token sequence의 결합된 의미를 가지게 되며, 여기에 classifier를 붙이면 classification을 할 수 있게 됩니다. classification 태스크가 아니라면 이 토큰은 무시하면 됩니다.
- 문장은 페어(pair)로 합쳐져서 입력되며, 두 개의 문장을 구분하기 위해, [SEP] 토큰과 세그먼트 임베딩 사용합니다.

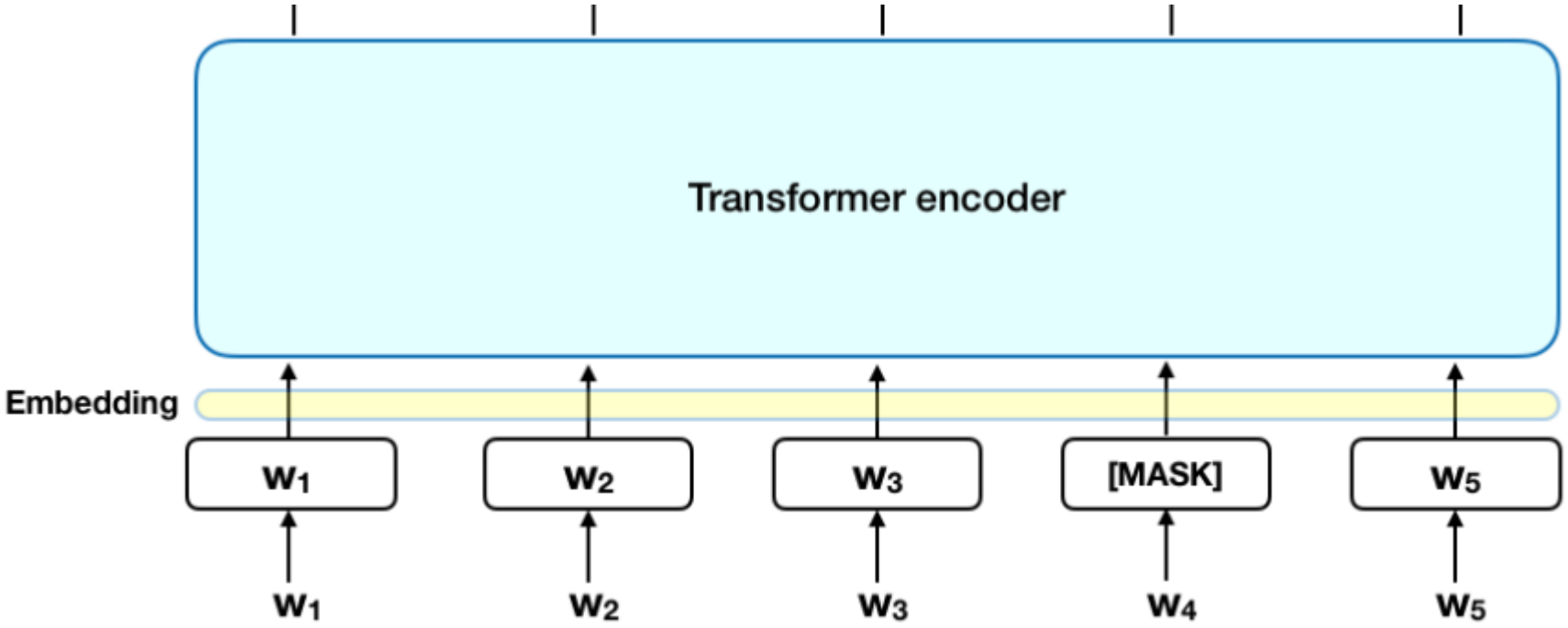


https://arxiv.org/pdf/1810.04805.pdf

BERT 사전 훈련(Pre-training)

■ Task #1: Masked LM

- 입력 단어 중의 일부를(15%) MASK 시킵니다. MASK 대상 중 80%는 [MASK]로 , 10%는 random word로 바꾸고, 10%는 token을 원래의 단어로 그대로 놔둡니다.
- [MASK] token 만을 predict하는 pre-training task를 수행합니다.
- [MASK] token은 pre-training에만 사용되고, fine-tuning시에는 사용되지 않습니다.
- 해당 token을 맞추어 내는 task를 수행하면서, BERT는 문맥을 파악하는 능력을 길러내게 됩니다.



https://bit.ly/2RhO80C

BERT 사전 훈련(Pre-training)

■ Task #2: Next Sentence prediction

- Next Sentence prediction pre-training task 수행하는 이유는, NLP task중에 QA(Question & Answering)나 NLI(Natural Language Inference)에서는 두 문장 사이의 관계를 이해하는 것이 중요하기 때문입니다.

- Corpus에서 두 문장을 이어 붙여, 이것이 원래의 corpus에서 바로 이어 붙여져 있던 문장인지를 맞추는 binarized next sentence prediction task를 수행합니다.

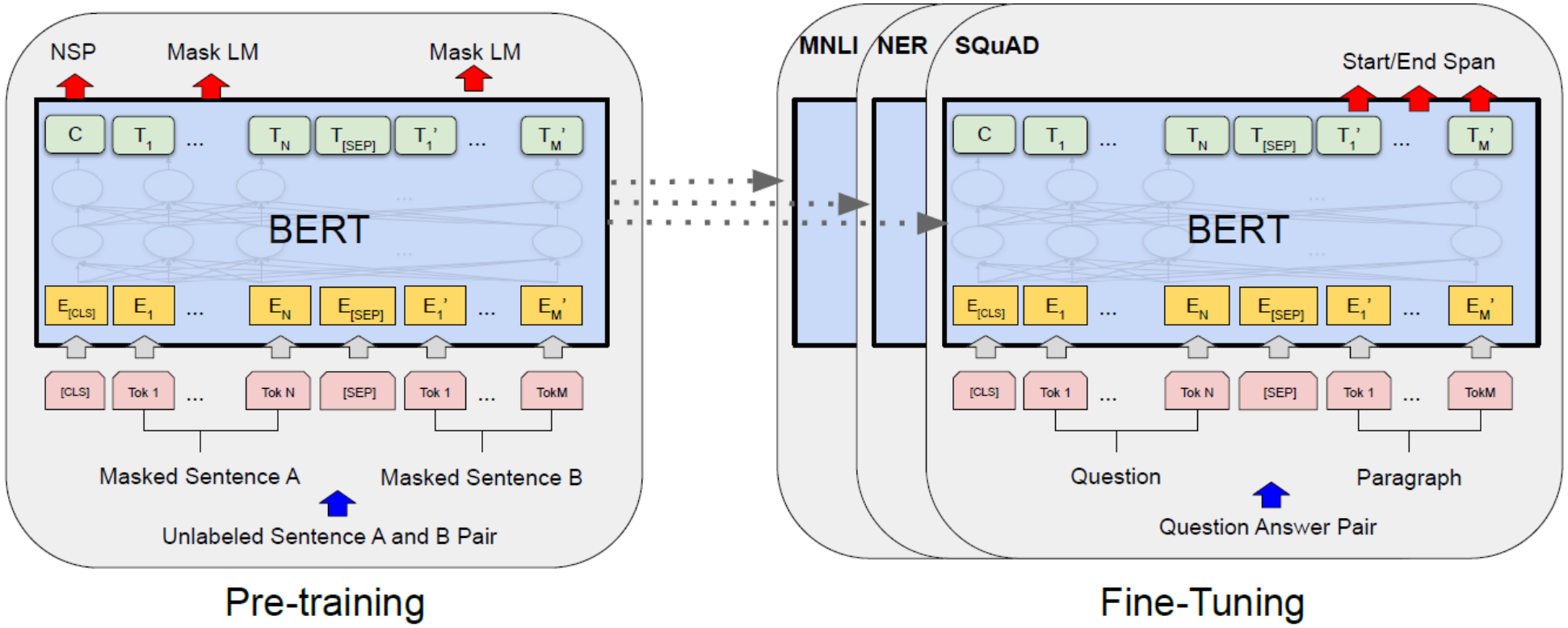
  Input = [CLS] the man went to [MASK] store [SEP] he bought a gallon [MASK] milk [SEP]
  LABEL = IsNext

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

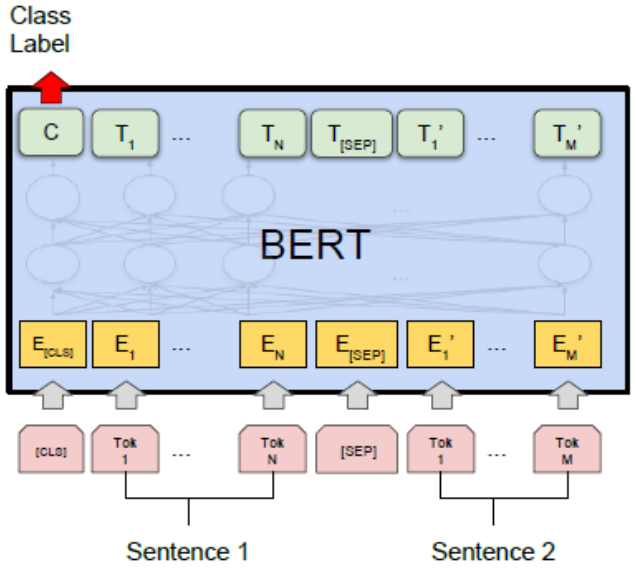
https://bit.ly/2RhO80C

BERT 파인 튜닝(Fine-tuning)

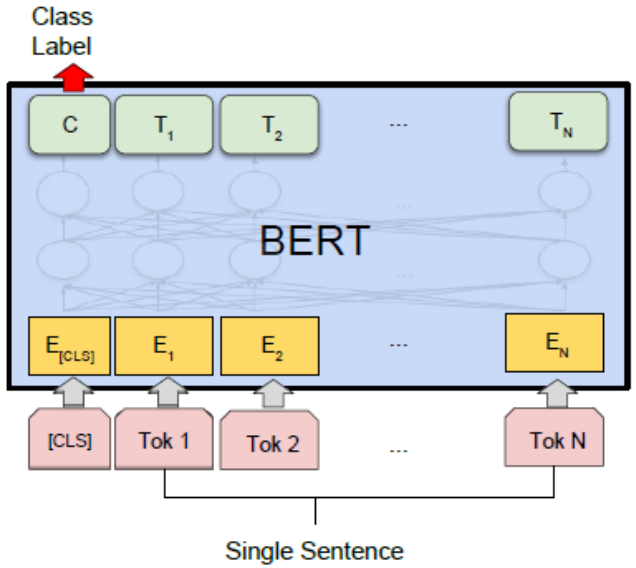
- 프리트레인을 마친 임베딩은 말뭉치의 의미적, 문법적 정보를 충분히 담고 있습니다.
- 하지만, 그 자체로는 다운스트림 태스트를 수행하기 어려우며, 파인 튜닝을 해야 원하는 성능이 나옵니다.
- 파인 튜닝은 프리트레인 이후 추가 학습을 시행해 임베딩을 다운스트림에 맞게 업데이트 하는 것입니다.
- 출력 레이어를 제외하고, 사전 학습과 파인 튜닝 에 동일한 아키텍처가 사용됩니다.



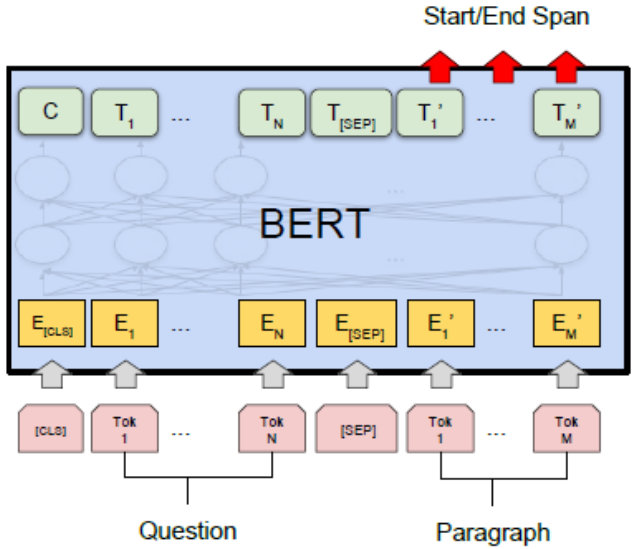
BERT 파인 튜닝



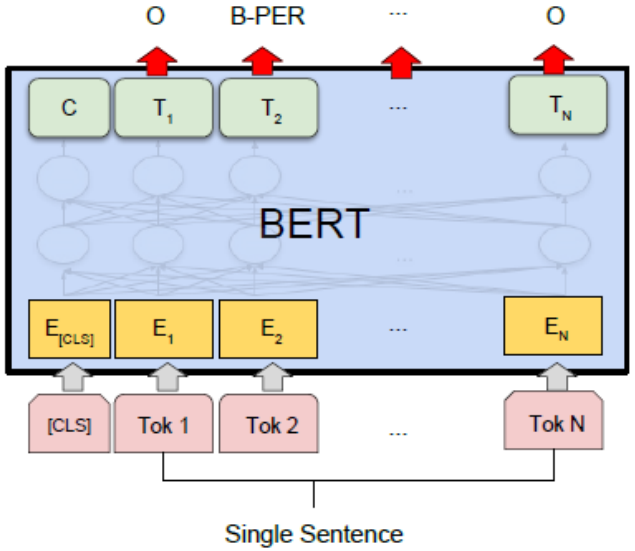
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG



(b) Single Sentence Classification Tasks: SST-2, CoLA

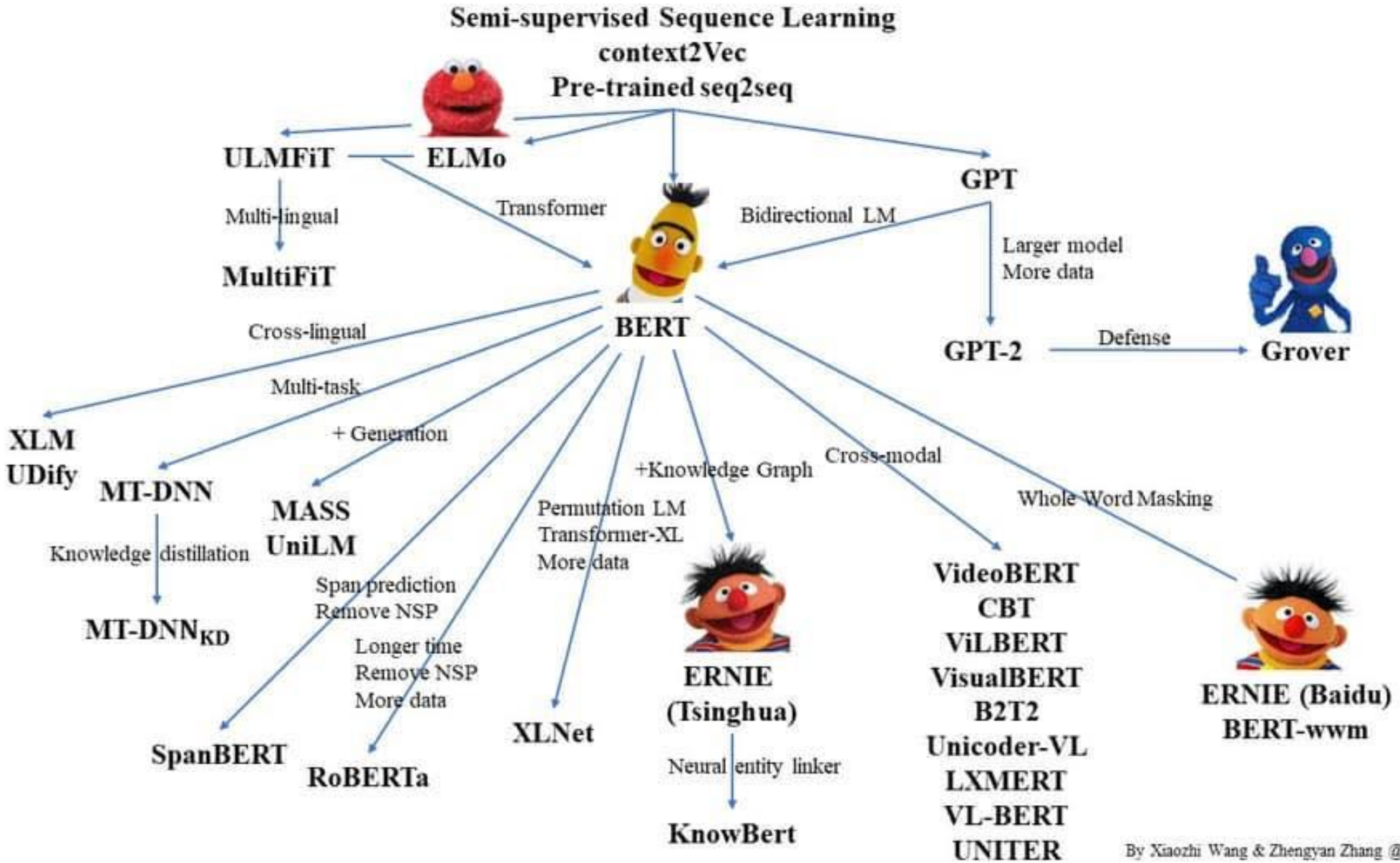


(c) Question Answering Tasks: SQuAD v1.1



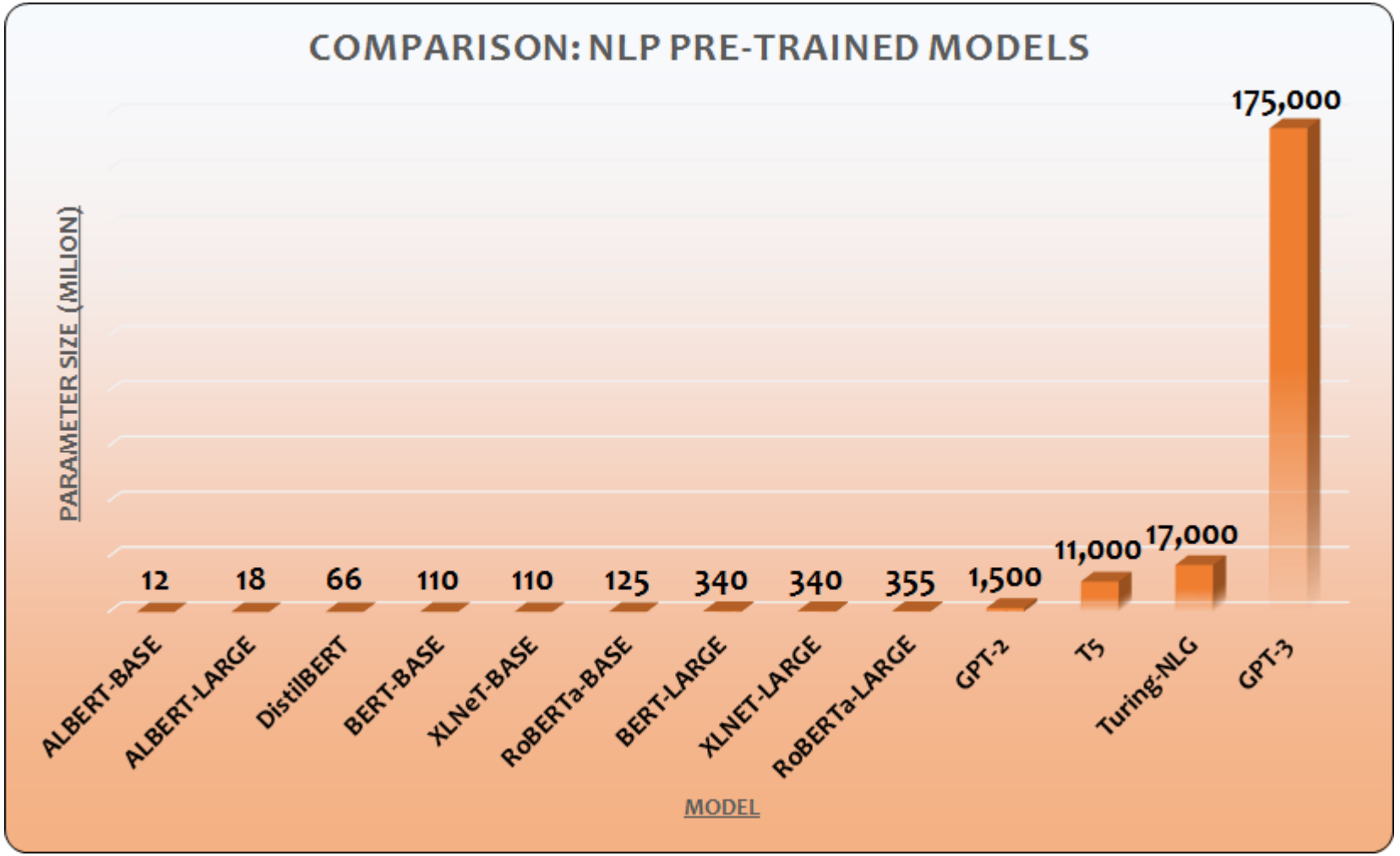
(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

언어모델 트렌드


Semi-supervised Sequence Learning
context2Vec
Pre-trained seq2seq
ELMo
ULMFiT
Multi-lingual
MultiFiT
Transformer
Bidirectional LM
GPT
Larger model
More data
GPT-2
Defense
Grover
BERT
Cross-lingual
XLM
UDify
Multi-task
MT-DNN
Knowledge distillation
MT-DNN$_{KD}$
+ Generation
MASS
UniLM
Span prediction
Remove NSP
SpanBERT
Longer time
Remove NSP
More data
RoBERTa
Permutation LM
Transformer-XL
More data
XLNet
+Knowledge Graph
ERNIE
(Tsinghua)
Neural entity linker
KnowBert
Cross-modal
VideoBERT
CBT
ViLBERT
VisualBERT
B2T2
Unicoder-VL
LXMERT
VL-BERT
UNITER
Whole Word Masking
ERNIE (Baidu)
BERT-wwm
By Xiaozhi Wang & Zhengyan Zhang @THUNLP

언어모델 트렌드

■ 모델 크기 증가

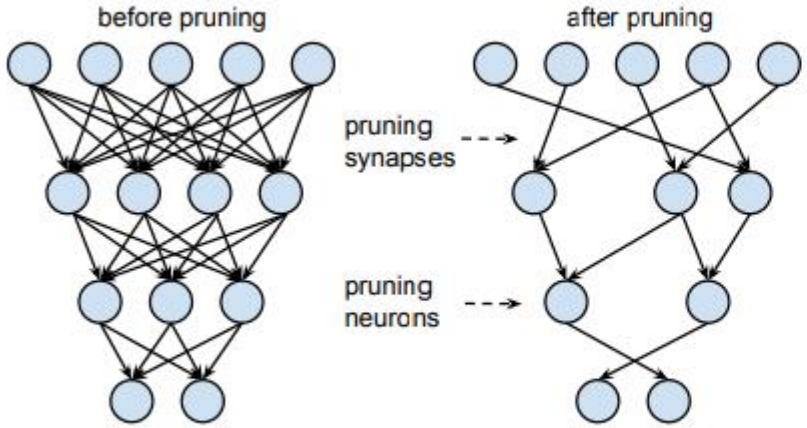


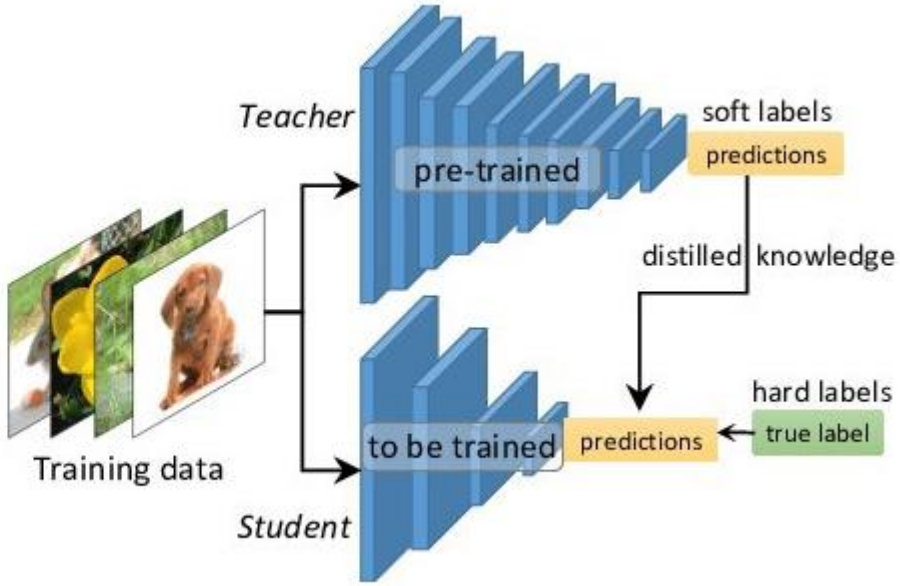
https://bit.ly/3t6whHn

언어모델 트렌드 - 모델 압축

■ 모델 압축

- 가지치기 (Pruning) : 학습 후에 불필요한 부분을 제거하는 방식
- 가중치 분해 (Weight Factorization) : 가중치 행렬을 분해하여 두 개의 작은 행렬의 곱으로 근사하는 방법
- 지식 증류 (Knowledge Distillation) : 미리 잘 학습된 큰 네트워크(Teacher network)로부터 실제로 사용하고자 하는 작은 네트워크(Student network)를 학습시키는 방식
- 가중치 공유 (Weight Sharing) : 모델의 일부 가중치들을 다른 파라미터들과 공유하는 방식, ALBERT
- 양자화 (Quantization) : 부동 소수점 값을 잘라내서 더 적은 비트만을 사용하는 방식

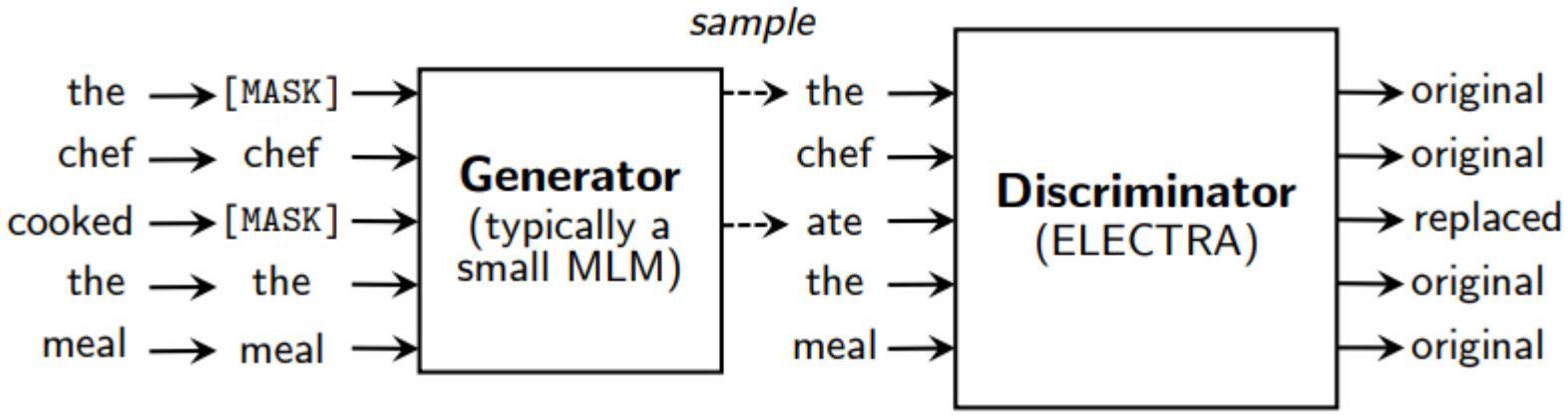




https://bit.ly/3ue9gU3

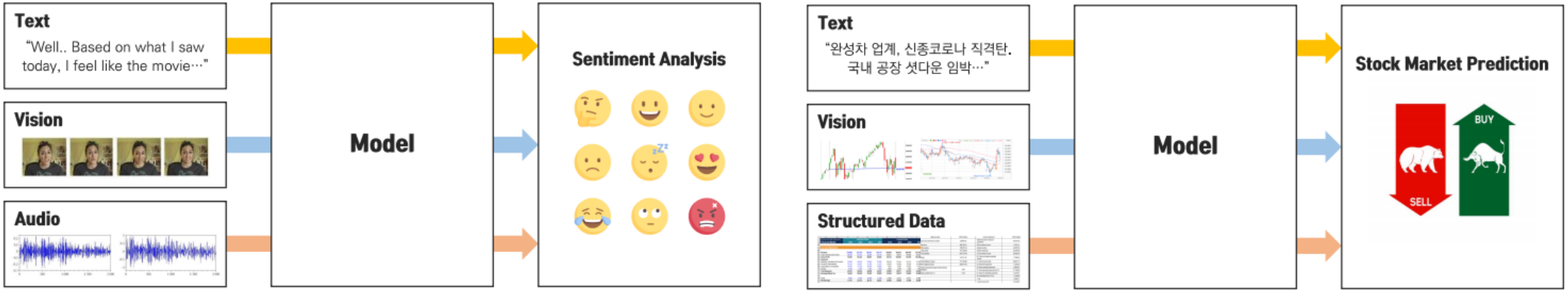
언어모델 트렌드

■ GAN 방식을 차용한 언어모델 : ELECTRA



https://littlefoxdiary.tistory.com/41

■ 다양한 소스의 데이터 사용 : MULTIMODAL TRANSFORMER



https://littlefoxdiary.tistory.com/41

BERT 실습

https://github.com/deepseasw/bert-naver-movie-review

https://huggingface.co/transformers/notebooks.html

https://github.com/monologg/KoELECTRA-Pipeline

BERT 참고 자료

BERT 모델 시각화

https://nlpinkorean.github.io/illustrated-bert/

KcBERT: Korean comments BERT

https://github.com/Beomi/KcBERT

KcELECTRA: Korean comments ELECTRA

https://github.com/Beomi/KcELECTRA

GPT3 - Language Models are Few-Shot Learners

https://littlefoxdiary.tistory.com/44

ratsgo's NLPBOOK

https://ratsgo.github.io/nlpbook/

Thank you

kgpark88@gmail.com