

Advanced DeepLearning for NLP

BERT 실습 코드: <https://bit.ly/3yM7wGU>

GPT 실습 코드: <https://bit.ly/3FWM8Ai>

220604

고우영

Contents

- 언어모델

- BERT
- GPT
- ELECTRA
- MASS

- 언어모델 활용

- 개체명인식
- 문서 요약
- 번역
- QA

자연어처리

- **자연어처리, Natural Language Processing(NLP)**

- 컴퓨터가 인간의 언어를 이해/모사 하도록 하는 학문
- 이를 바탕으로 각종 정보처리
- 언어학+컴퓨터공학+인공지능
- for 구조화 되지 않은 비정형 텍스트 데이터

- **NLU(Natural Language Understanding)**

- 자연어 형태의 문장을 이해하는 기술
- 문서 분류, [개체명인식](#), 형태소분류

- **NLG(Natural Language Generation)**

- 자연어 문장을 생성하는 기술
- [번역](#), [문서 요약](#), 챗봇, ..

언어 모델링

Language Modeling
(BERT, GPT, ELECTRA, MASS)

Embedding

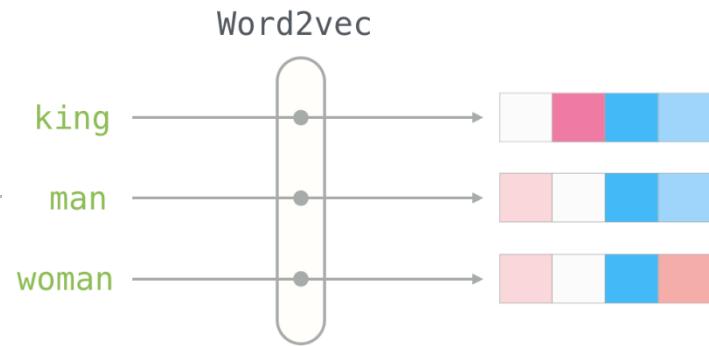
- 컴퓨터는 인간이 사용하는 언어를 그대로 이해하는게 아니다!
- 표현력이 무한한 언어를 벡터(숫자의 나열)로 변형하여 계산 => '임베딩'
- 품질 좋은 임베딩 → 좋은 NLP 성능
- 학습 수렴도 빠름
- 사람도 책을 읽을 때 기본 배경지식, 의미, 문법 정보를 가지고 독해

Embedding

▪ 단어 수준 임베딩

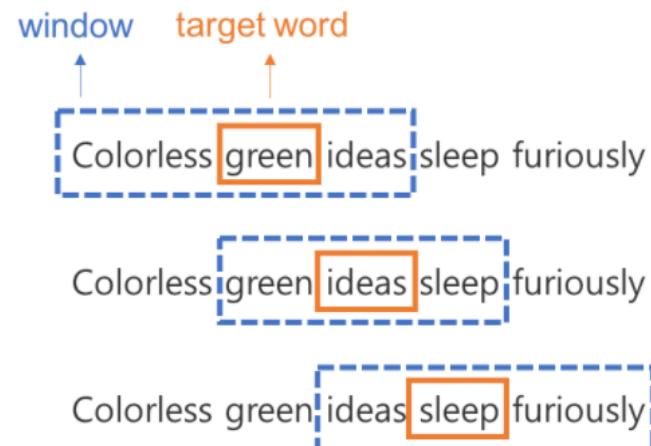
- 단어의 순서 정보
- 예측기반 임베딩(Word2Vec, FastText) 행렬 분해방법(Glove/Swivel)
- But, 같은 단어도 문맥에 따라 다른 의미가 될 수 있음
- 동의어 문제, 문맥상 의미 차이 -> 해결 불가

ex) 中 : 중국, 가운데, 중소기업 등



▪ 문장 수준 임베딩

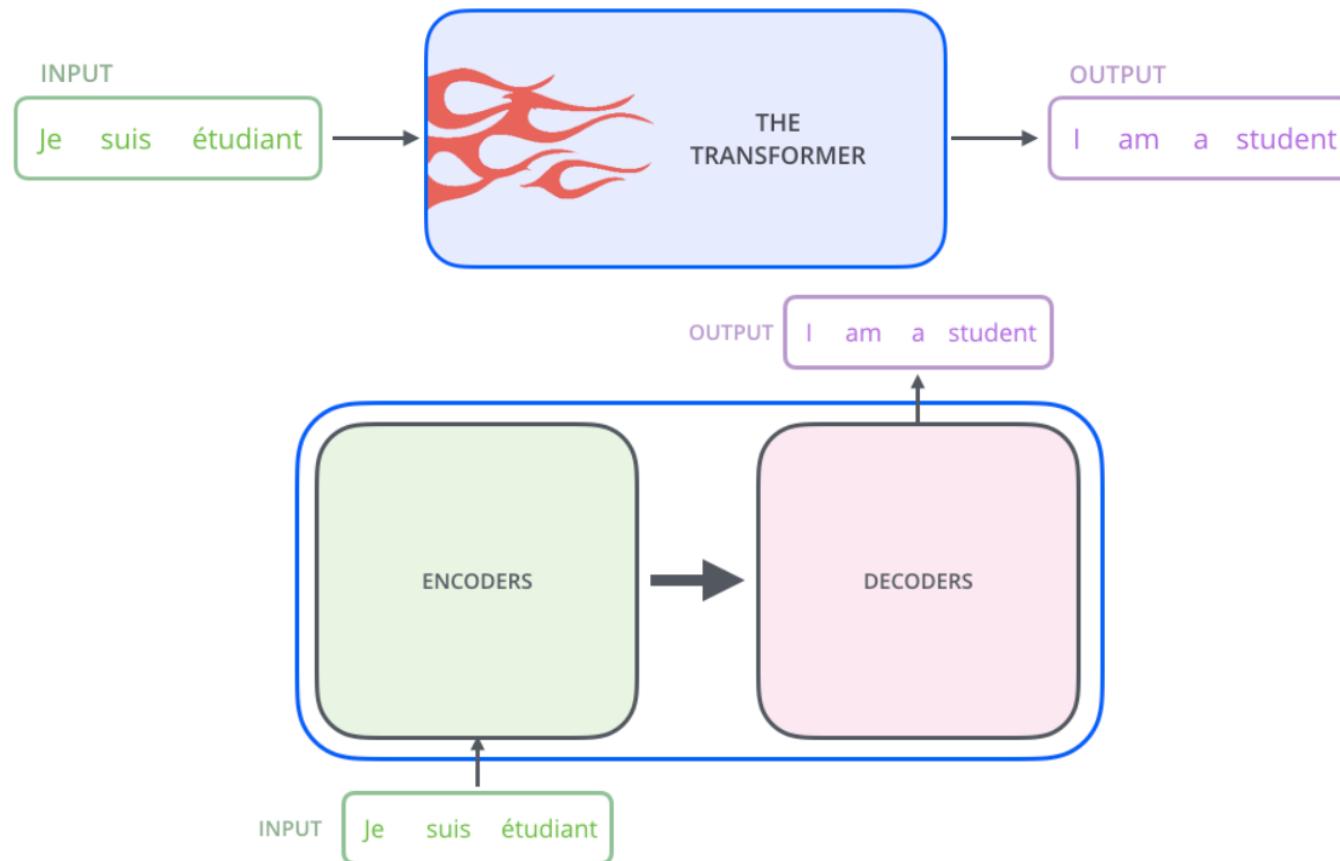
- Elmo, Bert, Electra, GPT 등



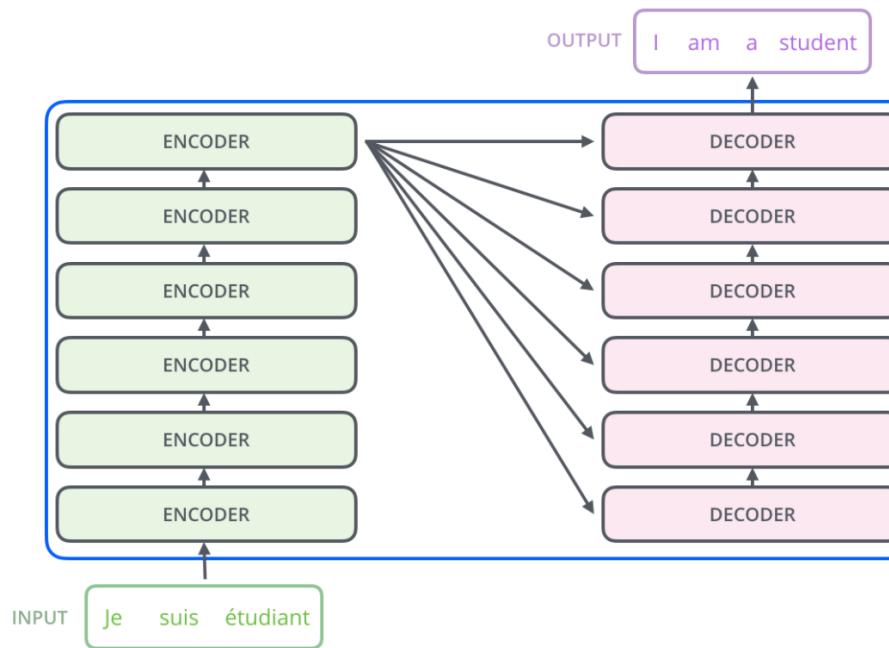
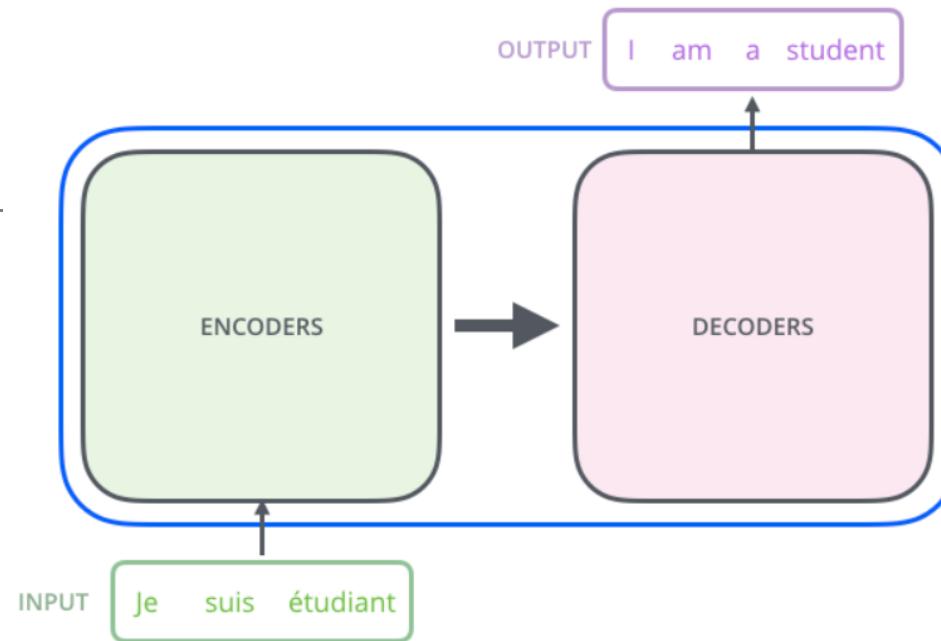
Seq2Seq

Seq2Seq

Encoder-Decoder, NMT

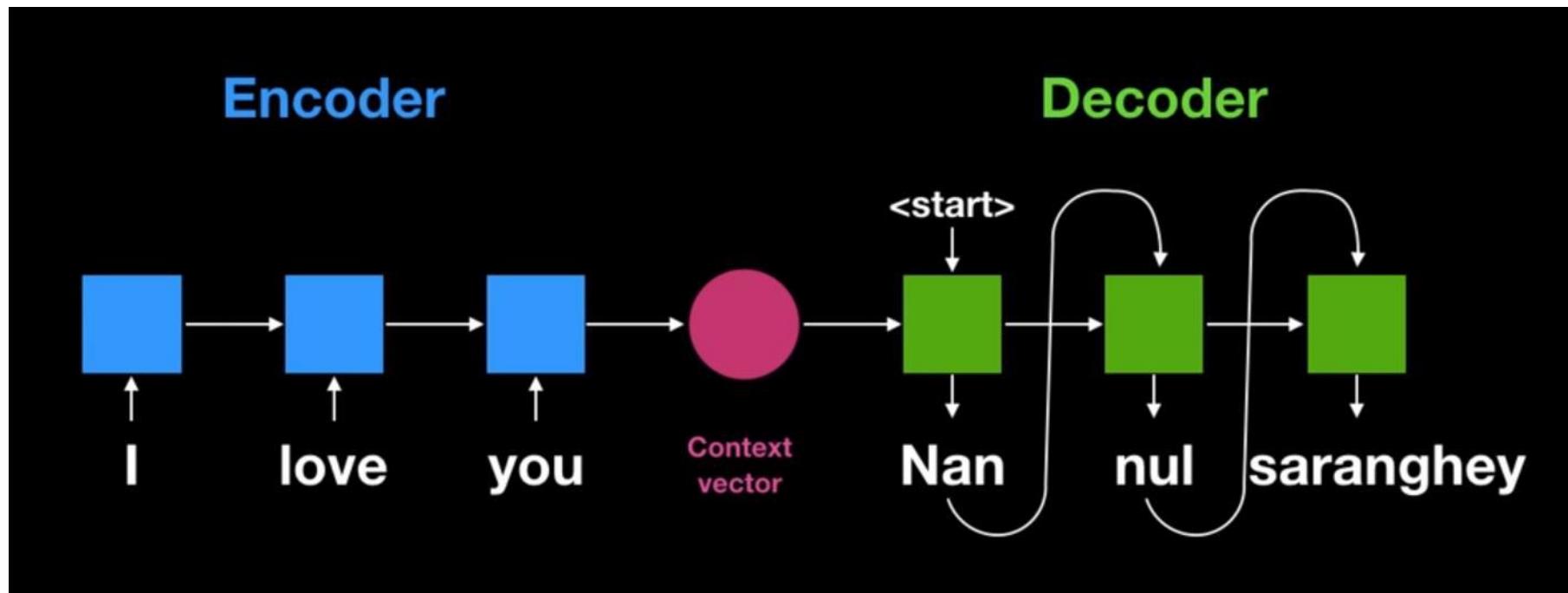
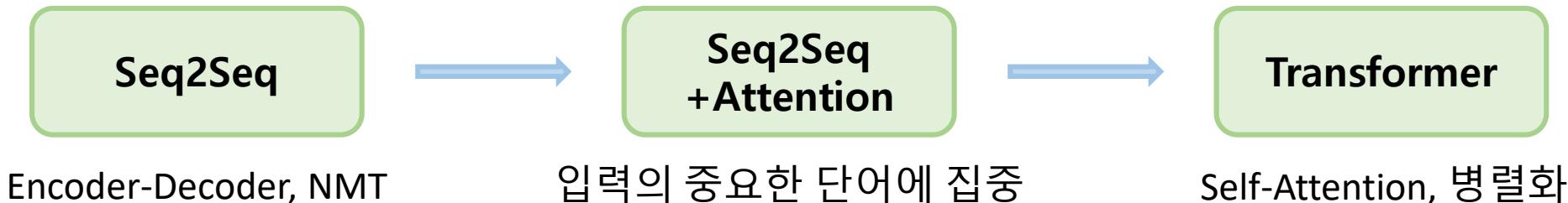


Seq2Seq



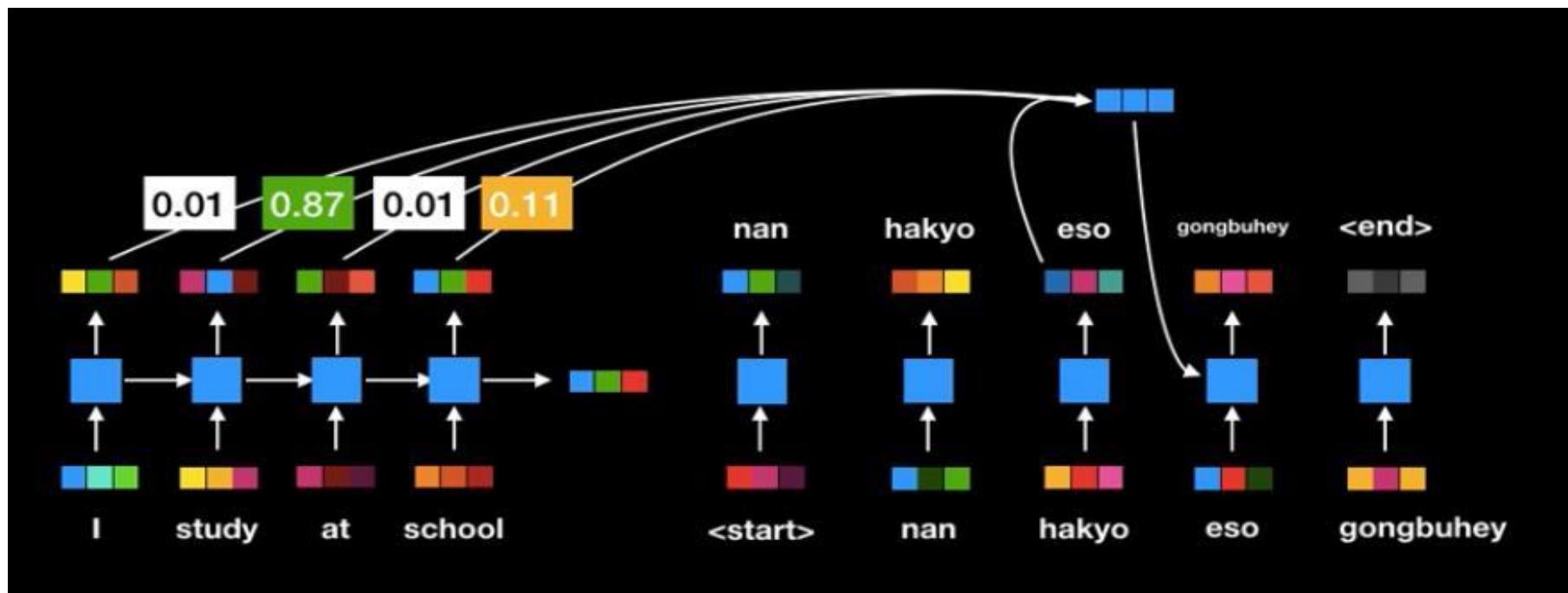
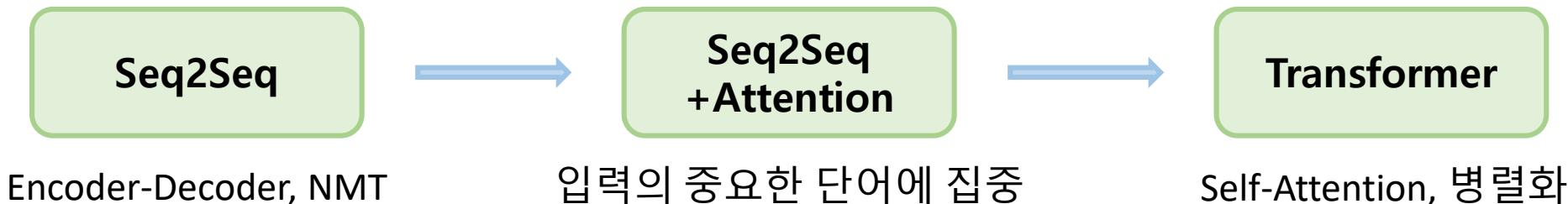
Seq2Seq 모델 구조

Seq2Seq



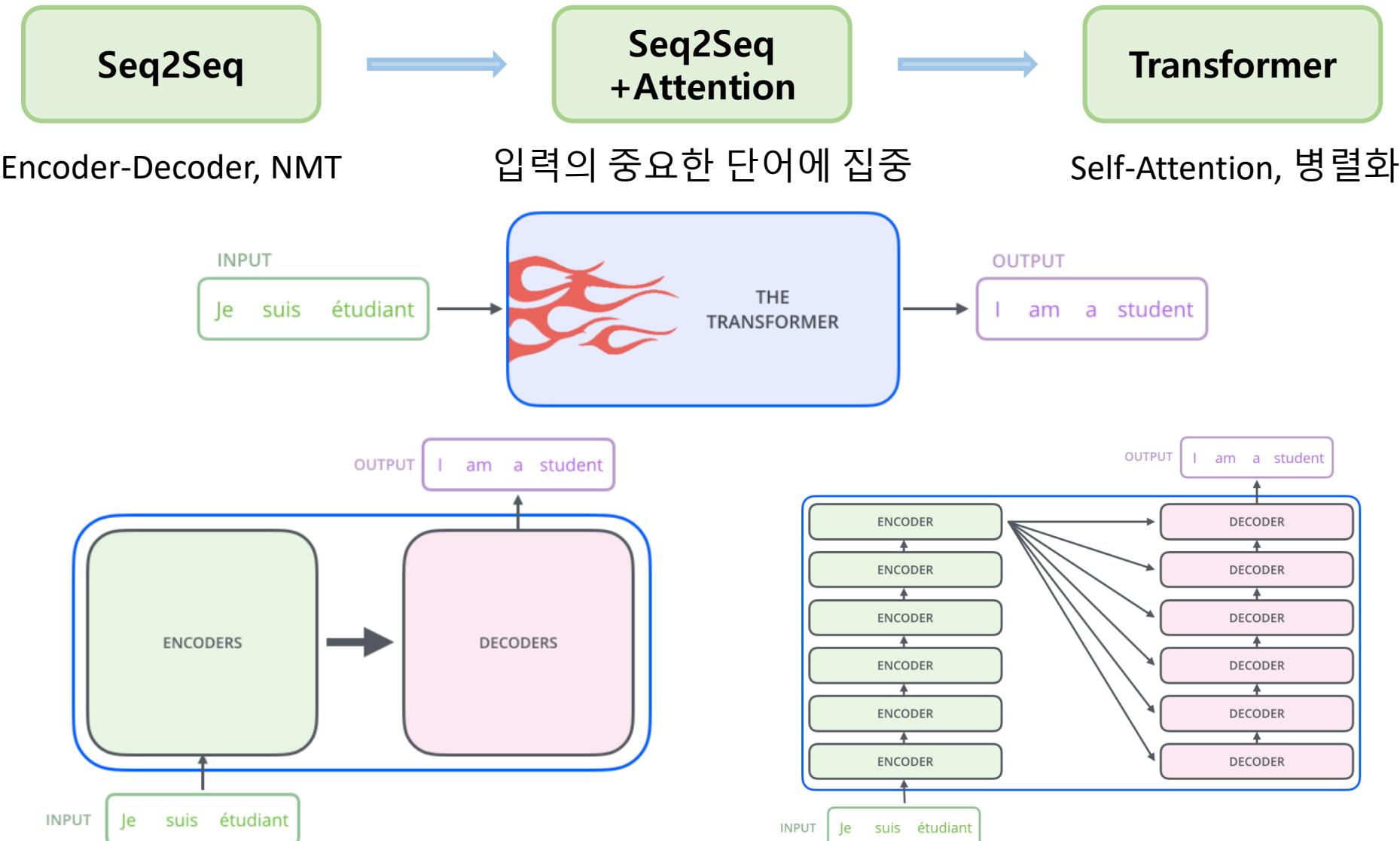
Seq2Seq 모델 구조

Seq2Seq

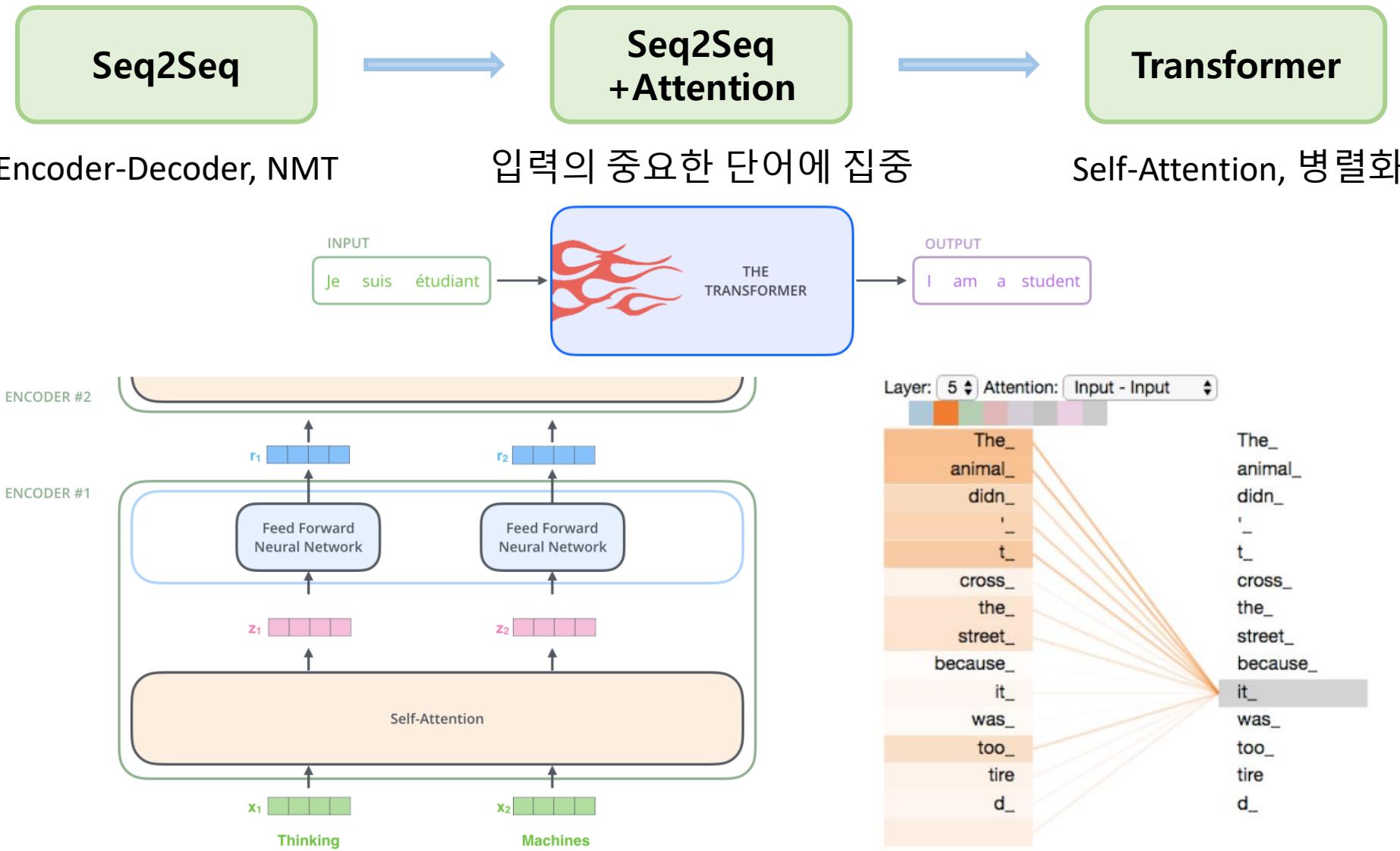


Attention+Seq2Seq 모델 구조

Seq2Seq



Seq2Seq



Transformer

Seq2Seq

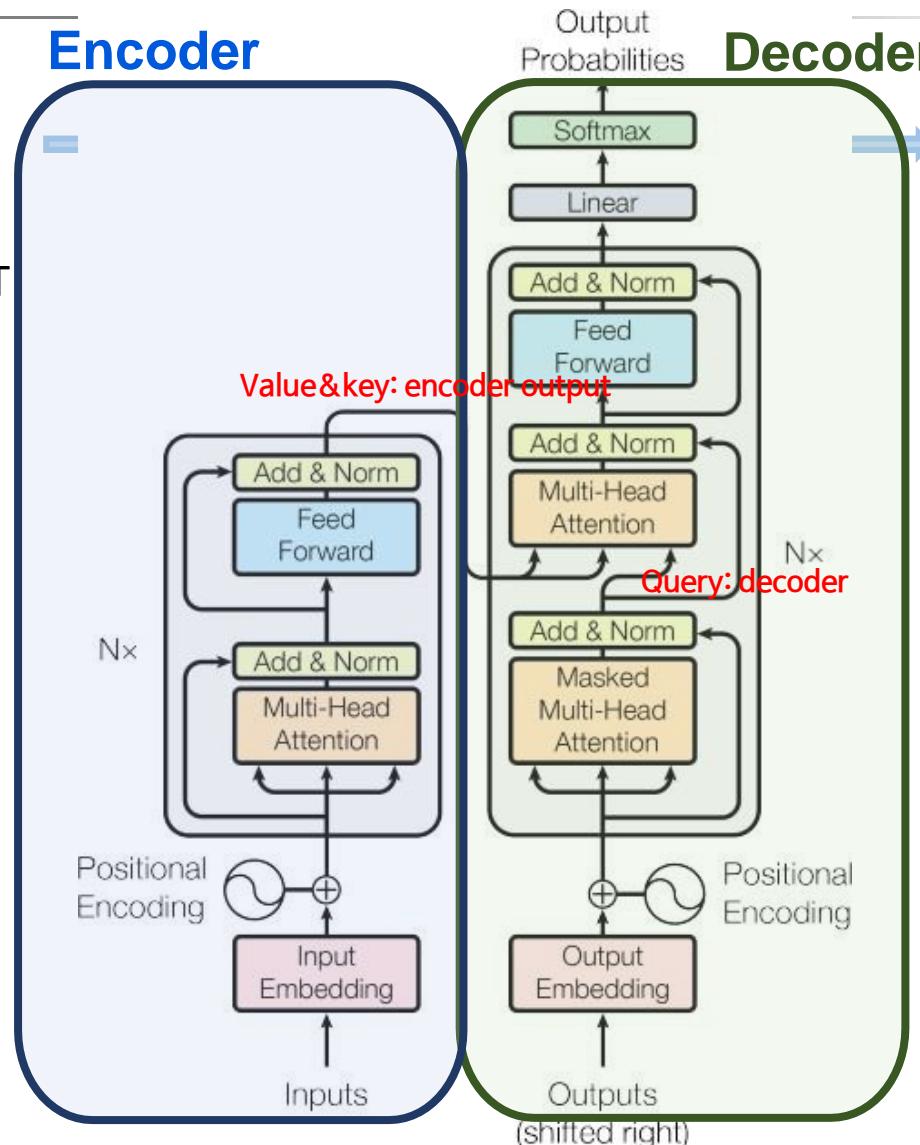
Encoder-Decoder, NMT

Encoder

Decoder

Transformer

Self-Attention, 병렬화



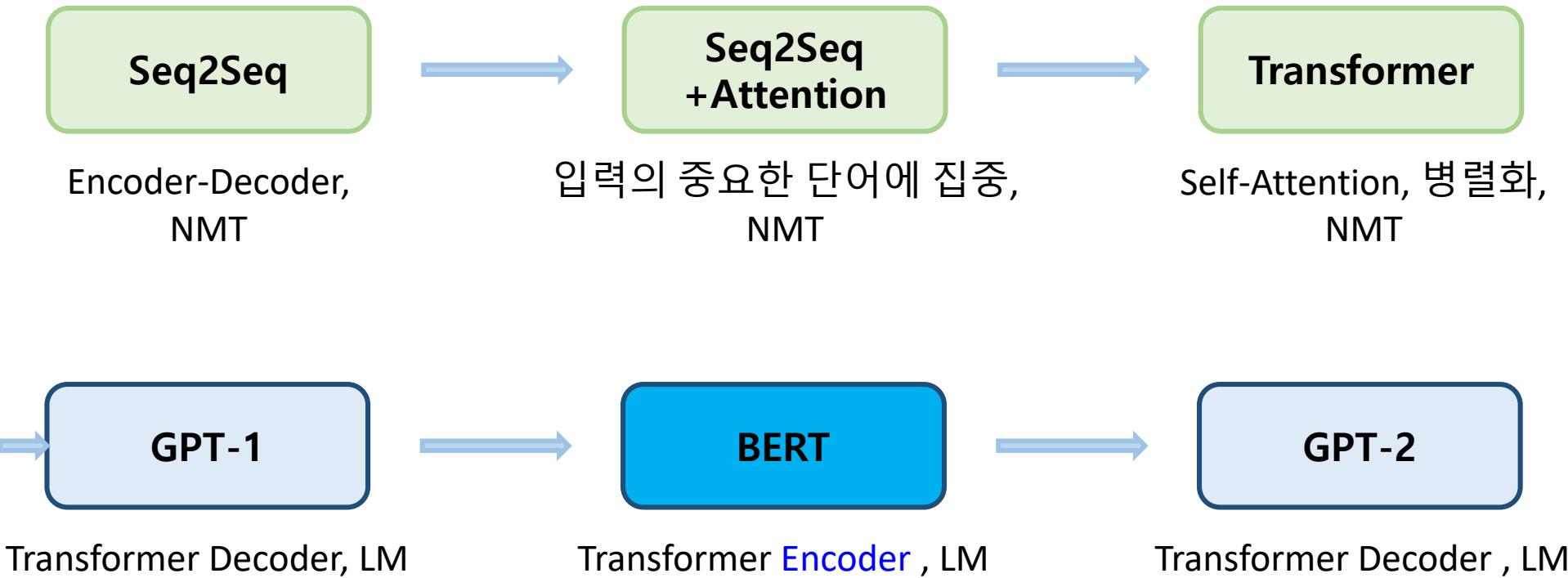
I study at school

<start> -> 나는 -> 학교 -> 에서 -> 공부 -> 한다
13/20
<http://jalammar.github.io/illustrated-gpt2/>

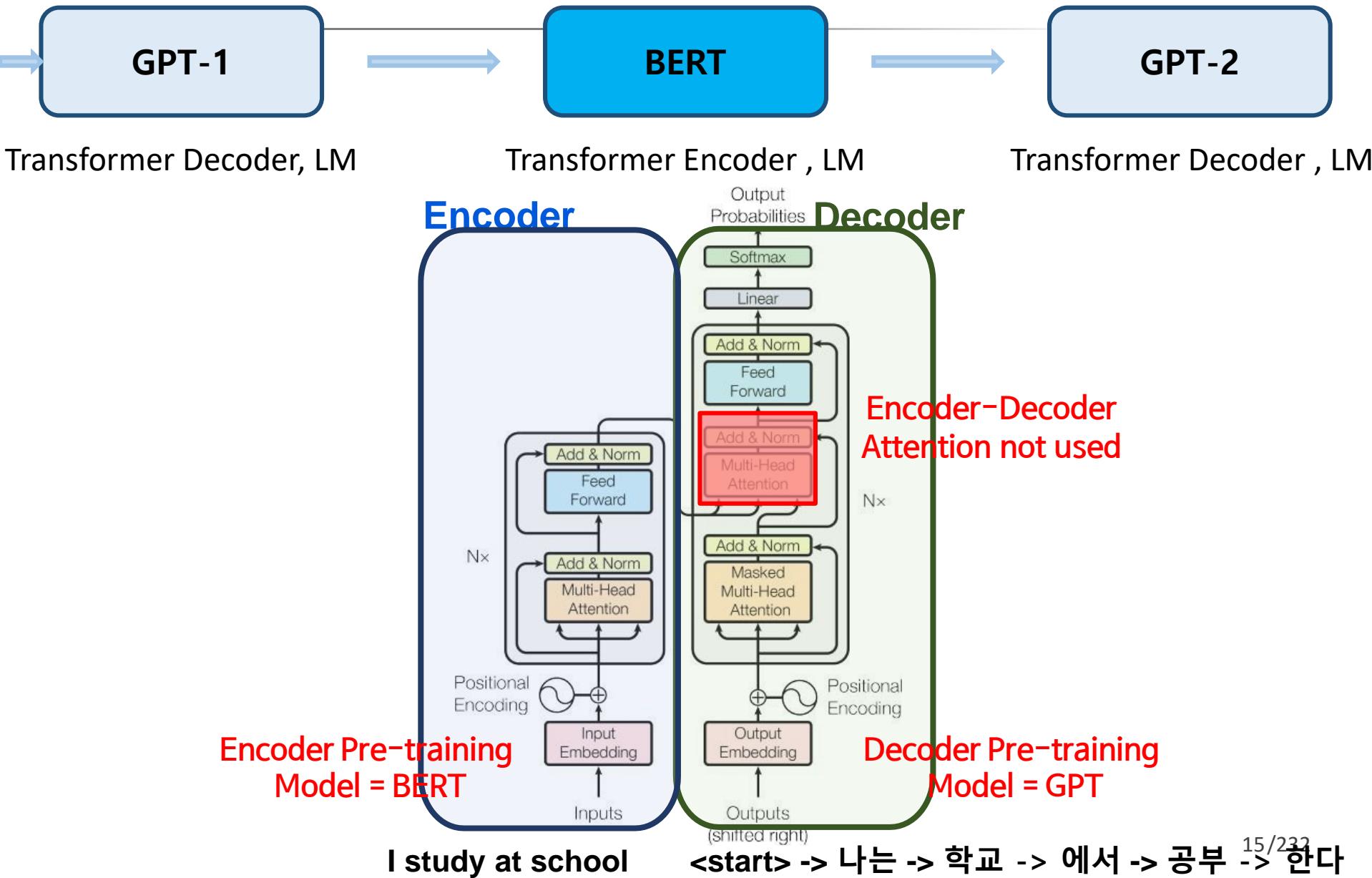
Transformer 장점

- 1) long-term dependency problem
- 2) Parallelization

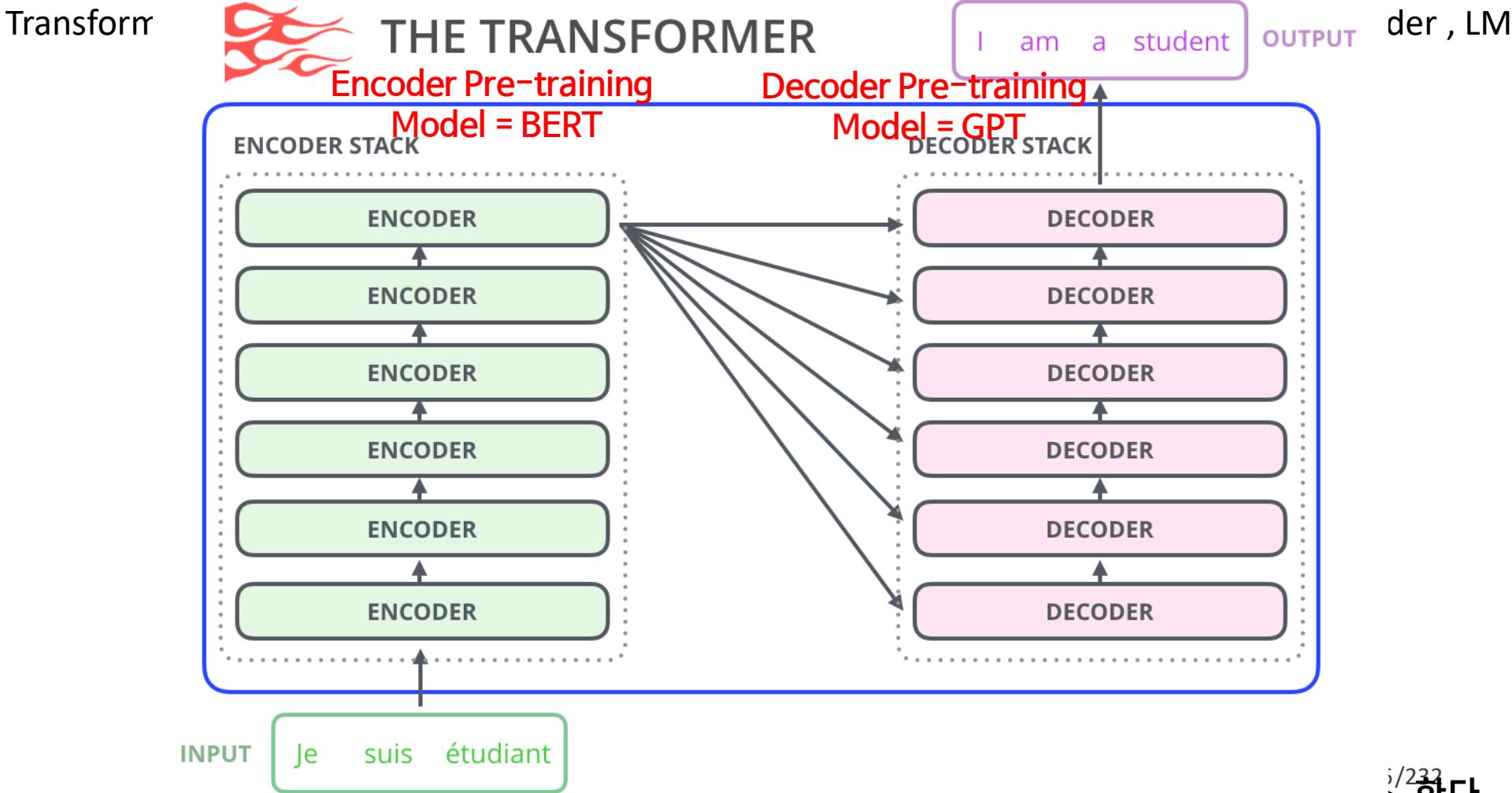
After Transformer



After Transformer

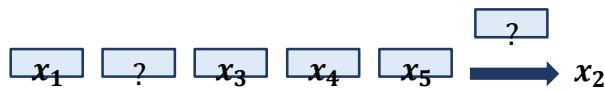


After Transformer



Language Model : Auto-Encoding VS Auto-Regressive

Auto Encoding



입력 문장

$$\bar{x} = [x_1, x_2, x_3, x_4, x_5]$$

오염된 문장

$$\hat{x} = [x_1, [MASK], x_3, x_4, x_5]$$

likelihood

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

우리 점심 뭐 먹을까?

오전 8:23

오전 8:25

몰라 대충 먹자

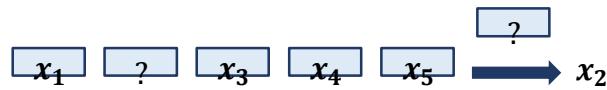
나는 중국음식 먹고 싶어

오전 8:26

그래 그럼 난 짜장

Language Model : Auto-Encoding VS Auto-Regressive

Auto Encoding



입력 문장

$$\bar{x} = [x_1, x_2, x_3, x_4, x_5]$$

오염된 문장

$$\hat{x} = [x_1, [MASK], x_3, x_4, x_5]$$

likelihood

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

우리

뭐 먹을까?

오전 8:23

몰라 대충 먹자

나는 중국음식

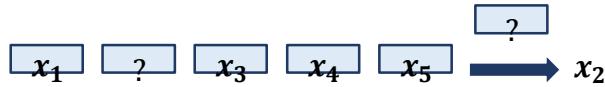
싶어

오전 8:26

그래 그럼 난

Language Model : Auto-Encoding VS Auto-Regressive

Auto Encoding



입력 문장

$$\bar{x} = [x_1, x_2, x_3, x_4, x_5]$$

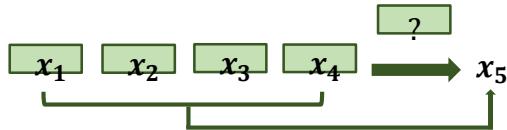
오염된 문장

$$\hat{x} = [x_1, [MASK], x_3, x_4, x_5]$$

likelihood

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

Auto Regressive



입력 문장

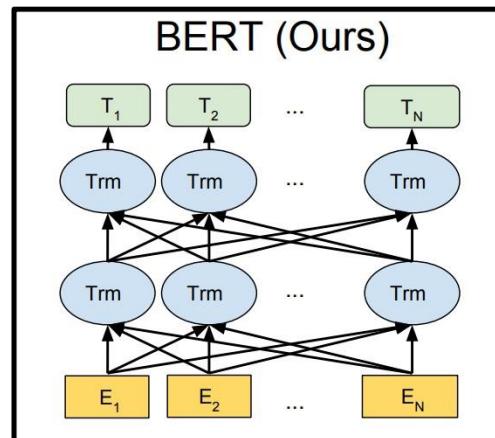
$$\bar{x} = [x_1, x_2, x_3, x_4]$$

다음 단어(정답)

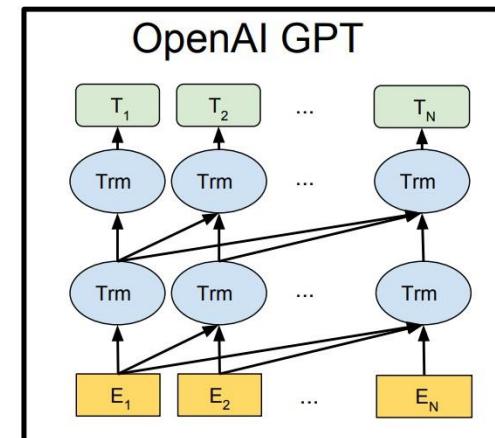
$$x = x_5$$

likelihood

$$p(x) \approx \prod_{t=1}^T p(x_t|x_{<t})$$



BERT (Ours)



OpenAI GPT

언어모델 Final

- 그래서 입출력이 뭐지??

입력

北	정보탈취	전문	해킹조직	'APT37'	전모	공개
---	------	----	------	---------	----	----



언어모델

BERT/GPT/ELECTRA

언어모델

입력 차원 : 문장개수x문장길이

출력 차원 : 문장개수x문장길이x768

$1 \times 7 \rightarrow 1 \times 7 \times 768$

출력

base: 768

large: 1024

0.01
0.2
0.05
0.4
...

0.04
0.6
0.02
0.1
...

0.02
0.5
0.08
0.3
...

0.13
0.21
0.01
0.32
...

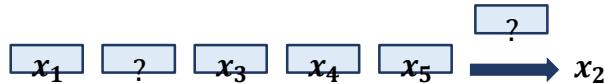
0.53
0.01
0.05
0.2
...

0.1
0.22
0.05
0.34
...

0.07
0.15
0.07
0.25
...

Language Model : Auto-Encoding VS Auto-Regressive

Auto Encoding



입력 문장

$$\bar{x} = [x_1, x_2, x_3, x_4, x_5]$$

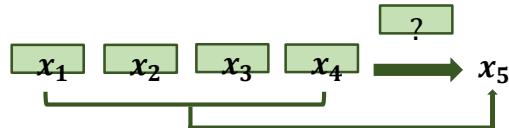
오염된 문장

$$\hat{x} = [x_1, [MASK], x_3, x_4, x_5]$$

likelihood

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

Auto Regressive



입력 문장

$$\bar{x} = [x_1, x_2, x_3, x_4]$$

다음 단어(정답)

$$x = x_5$$

likelihood

$$p(x) \approx \prod_{t=1}^T p(x_t | x_{<t})$$

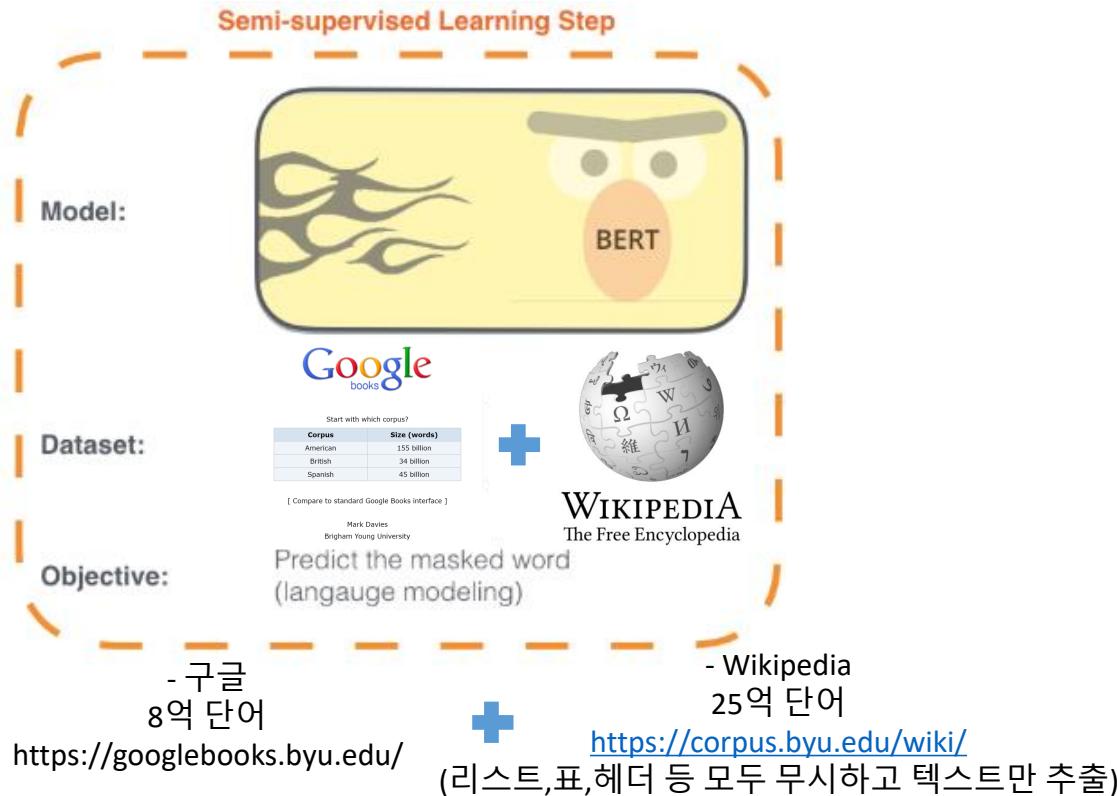
	BERT	GPT
입력/출력	입력: N개의 단어 열 출력: N개 단어의 인코딩 벡터	입력: N개의 단어 열 출력: N+1번째 단어
모델 특징	양방향 문맥 정보 활용	단방향 문맥 정보 활용
사전학습 태스크	입력: 임의의 단어 masking 출력: 주위 문맥으로 해당 단어 맞추기	입력: 이전 단어 열 출력: 다음 단어 예측
비고	동일 문장이라도 random masking 위치에 따라 서로 다른 정보 학습 (→ random masking을 수 차례 반복하여 학습)	동일 문장이면 동일 정보 학습
한계	Downstream task에 Fine-tuning 시 [MASK] 토큰이 등장하지 않음, Pre-training과 Fine-tuning의 학습 목표 불일치	단방향 정보만으로 학습 가능함, 양방향 정보 이용 불가

언어 모델링

BERT

BERT

- **BERT**(Pre-training of Deep Bidirectional Transformers for Language Understanding)
- 구글 + wikipedia(33억 단어) 대용량 데이터로 언어 학습



BERT

- **BERT**(Pre-training of Deep Bidirectional Transformers for Language Understanding)
- 구글 + wikipedia(33억 단어) 대용량 데이터로 언어 학습

1. 다음 글이가 요청하는 것으로 가장 적절한 것은? [18]¹

Thank you for deciding to send your child to Gibbons Summer Camp. We have found over the past few years that text messages are the most reliable form of communication, so we are asking for your permission to contact your child. By completing the form, you will be giving us permission to contact your child via text over the summer. We understand if you do not wish to grant this permission. But we ask that you complete this form indicating your preference and have your child return it no later than May 10, 2019.

- ① to let them know how important the text messages are
- ② to contact their child by sending an email
- ③ to ask to fill out an application form
- ④ to gain permission from them to contact their child
- ⑤ to thank them for participating in Gibbons Summer Camp

2. 다음 글의 내용과 일치하지 않는 것은? [18]²

Thank you for deciding to send your child to Gibbons Summer Camp. We have found over the past few years that text messages are the most reliable form of communication, so we are asking for your permission to contact your child. By completing the form, you will be giving us permission to contact your child via text over the summer. We understand if you do not wish to grant this permission. But we ask that you complete this form indicating your preference and have your child return it no later than May 10, 2019.

- ① Gibbons Summer Camp is for children.
- ② They seem to have tried the best method of communication for many years.
- ③ When the form is completed, they will be texting

3. (A)(B)(C)의 각 네보 밑에서 어법에 맞는 표현으로 가장 적절한 것은? [18]³

Thank you for deciding to send your child to Gibbons Summer Camp. We have found over the past (A) [few / a few] years that text messages are the most reliable form of communication, so we are asking for your permission to contact your child. By completing the form, you will be giving us permission to contact your child via text over the summer. We understand (B) [if / that] you do not wish to grant this permission. But we ask that you complete this form indicating your preference and have your child return it (C) [until / by] May 10, 2019.

- ① a few - if - until
- ② few - if - by
- ③ few - that - until
- ④ few - that - by
- ⑤ few - if - by

수능 문제만 열심히 푼 학생

VS

영어 소설, 뉴스 3.3억 개로 공부
+ 수능문제 공부



Start with which corpus?	
Corpus	Size (words)
American	155 billion
British	31 billion
Spanish	45 billion

[Compare to standard Google Books interface]

Mark Davies
Brigham Young University



WIKIPEDIA
The Free Encyclopedia

BERT

- **BERT**(Pre-training of Deep Bidirectional Transformers for Language Understanding)
- 구글 + wikipedia(33억 단어) 대용량 데이터로 언어 학습

Model	F1
Flair embeddings (Akbik et al., 2018)♦	93.09
BERT Large (Devlin et al., 2018)	92.8
CVT + Multi-Task (Clark et al., 2018)	92.61
BERT Base (Devlin et al., 2018)	92.4
BiLSTM-CRF+ELMo (Peters et al., 2018)	92.22
Peters et al. (2017) ♦	91.93
CRF + AutoEncoder (Wu et al., 2018)	91.87
Bi-LSTM-CRF + Lexical Features (Ghaddar and Langlais 2018)	91.73
Chiu and Nichols (2016) ♦	91.62
HSCRF (Ye and Ling, 2018)	91.38
IXA pipes (Agerri and Rigau 2016)	91.36
NCRF++ (Yang and Zhang, 2018)	91.35
LM-LSTM-CRF (Liu et al., 2018)	91.24
Yang et al. (2017) ♦	91.26
Ma and Hovy (2016)	91.21
LSTM-CRF (Lample et al., 2016)	90.94

Rank	Name	Model	URL	Score	CoLA	SST-2	MRPC	STS-B	QQP	MNLI-m	MNLI-mm	QNLI	RTE	WNLI	AX
1	Jacob Devlin	BERT: 24-layers, 1024-hidden, 16-heads		80.4	60.5	94.9	89.3/85.4	87.6/86.5	72.1/89.3	86.7	85.9	91.1	70.1	65.1	39.6
2	Jason Phang	GPT on STILTs		76.9	47.2	93.1	87.7/83.7	85.3/84.8	70.1/88.1	80.7	80.6	87.2	69.1	65.1	29.4
3	Alec Radford	Singletask Pretrain Transformer		72.8	45.4	91.3	82.3/75.7	82.0/80.0	70.3/88.5	82.1	81.4	88.1	56.0	53.4	29.8
4	Samuel Bowman	BiLSTM+ELMo+Attn		70.5	36.0	90.4	84.9/77.9	75.1/73.3	64.8/84.7	76.4	76.1	79.9	56.8	65.1	26.5
5	GLUE Baselines	BiLSTM+ELMo+Attn		68.9	18.9	91.6	83.5/77.3	72.8/71.1	63.3/83.5	75.6	75.9	81.7	61.2	65.1	22.6



Rank	Model	EM	F1
	Human Performance Stanford University (Rajpurkar & Jia et al. '18)	86.831	89.452
1 Nov 25, 2018	PAML+BERT (ensemble) PINGAN GammaLab	83.435	85.992
2 Nov 16, 2018	AoA + DA + BERT (ensemble) Joint Laboratory of HIT and iFLYTEK Research	82.374	85.310
3 Nov 27, 2018	PAML+BERT (single model) PINGAN GammaLab	81.347	84.560
4 Nov 16, 2018	AoA + DA + BERT (single model) Joint Laboratory of HIT and iFLYTEK Research	81.178	84.251
5 Nov 08, 2018	BERT (single model) Google AI Language	80.005	83.061

모든 분야 최고 성능 기록

BERT

▪ BERT (Pre-training)

▪ 구글 + will

Model	F1
Flair embeddings (Akbik et al., 2018)♦	93.09
BERT Large (Devlin et al., 2018)	92.8
CVT + Multi-Task (Clark et al., 2018)	92.61
BERT Base (Devlin et al., 2018)	92.4
BiLSTM-CRF+ELMo (Peters et al., 2018)	92.22
Peters et al. (2017) ♦	91.93
CRF + AutoEncoder (Wu et al., 2018)	91.87
Bi-LSTM-CRF + Lexical Features (Ghaddar and Langlais 2018)	91.73
Chiu and Nichols (2016) ♦	91.62
HSCRF (Ye and Ling, 2018)	91.38
IXA pipes (Agerri and Rigau 2016)	91.36
NCRF++ (Yang and Zhang, 2018)	91.35
LM-LSTM-CRF (Liu et al., 2018)	91.24
Yang et al. (2017) ♦	91.26
Ma and Hovy (2016)	91.21
LSTM-CRF (Lample et al., 2016)	90.94

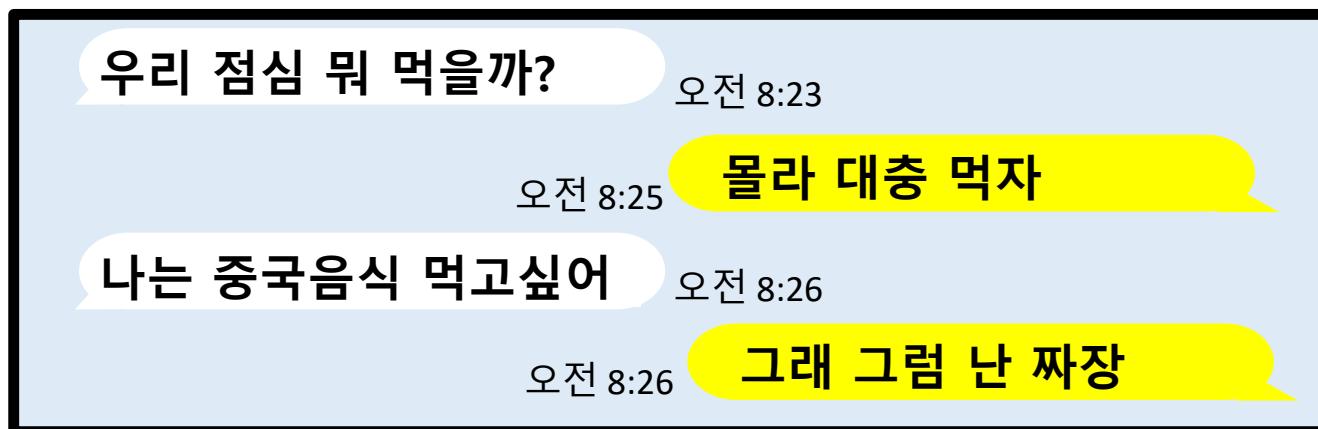
Rank	Name	Model
1	HFL iFLYTEK	MacALBERT + DKM
2	Alibaba DAMO NLP	StructBERT + TAPT
3	PING-AN Omni-Sinitic	ALBERT + DAAF + NAS
4	ERNIE Team - Baidu	ERNIE
5	T5 Team - Google	T5
6	Microsoft D365 AI & MSR AI & GATECHMT-DNN-SMART	
7	Zihang Dai	Funnel-Transformer (Ensemble B10)
8	ELECTRA Team	ELECTRA-Large + Standard Tricks
9	Huawei Noah's Ark Lab	NEZHA-Large
10	Microsoft D365 AI & UMD	FreeLB-RoBERTa (ensemble)
11	Junjie Yang	HIRE-RoBERTa
12	Facebook AI	RoBERTa
13	Microsoft D365 AI & MSR AI	MT-DNN-ensemble
14	GLUE Human Baselines	GLUE Human Baselines
15	Stanford Hazy Research	Snorkel MeTaL
16	XLM Systems	XLM (English only)

NLI-m	MNLI-mm	QNLI	RTE	WNLI	AX
86.7	85.9	91.1	70.1	65.1	39.6
80.7	80.6	87.2	69.1	65.1	29.4
82.1	81.4	88.1	56.0	53.4	29.8
76.4	76.1	79.9	56.8	65.1	26.5
75.6	75.9	81.7	61.2	65.1	22.6

▪ 최고 성능 기록

BERT

- 학습방법 1 : 빈칸 맞추기 문제(Masked Language Modeling)
- 전체 단어에서 15%를 랜덤하게 삭제 후 빈칸 맞추기



BERT

- 학습방법 1 : 빈칸 맞추기 문제(Masked Language Modeling)
- 전체 단어에서 15%를 랜덤하게 삭제 후 빈칸 맞추기



BERT

- 학습방법 1. 빠칸 맞추기 문제(Masked Language Modeling)
- 전체 퀴즈

1-1 빠칸 맞추기

Due to the fact that people tend to favor more (A) outputs, fossil fuels are more (B) than renewable energy alternatives in regards to the distance between inputs and outputs.

(A)

(B)

- | | | |
|--------------|-------|-------------|
| ① immediate | | competitive |
| ② available | | expensive |
| ③ delayed | | competitive |
| ④ convenient | | expensive |
| ⑤ abundant | | competitive |

BERT

- 학습방법 2 : 연속 대화 여부(Next Sentence Prediction)
- 두 문장이 연속된 문장인지, 상관없는 문장인지 맞추기

우리 점심 뭐 먹을까?

오전 8:23

몰라 대충 먹자

오전 8:25

나는 중국음식 먹고 싶어

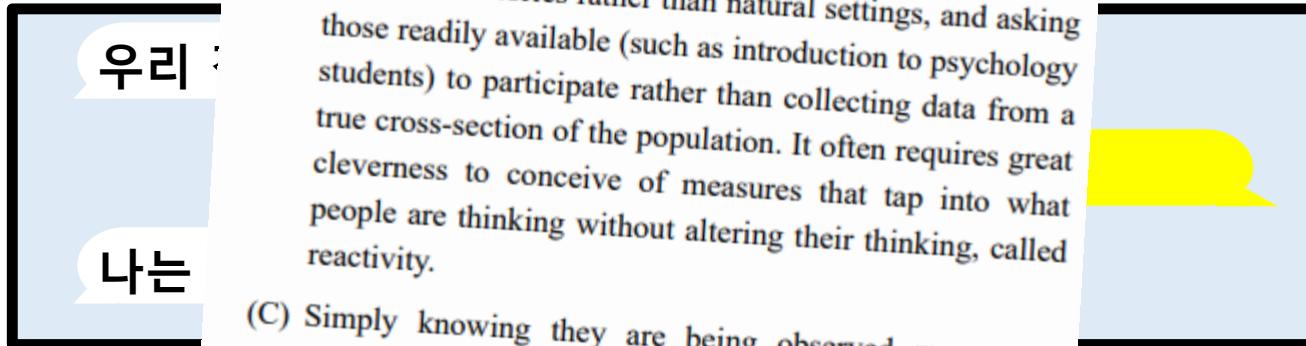
오전 8:26

문맥에 맞는지? Yes or No

그래 그럼 난 짜장

Researchers in psychology follow the scientific method to perform studies that help explain and may predict human behavior. This is a much more challenging task than studying snails or sound waves.

- 학습방법 2 : ↗
- 두 문장이 연관되는지 여부를 판별하는 문제입니다.



- (A) But for all of these difficulties for psychology, the payoff of the scientific method is that the findings are replicable; that is, if you run the same study again following the same procedures, you will be very likely to get the same results.
- (B) It often requires compromises, such as testing behavior within laboratories rather than natural settings, and asking those readily available (such as introduction to psychology students) to participate rather than collecting data from a true cross-section of the population. It often requires great cleverness to conceive of measures that tap into what people are thinking without altering their thinking, called reactivity.
- (C) Simply knowing they are being observed may cause people to behave differently (such as more politely!). People may give answers that they feel are more socially desirable than their true feelings.

- ① (A) – (C) – (B)
- ③ (B) – (C) – (A)
- ⑤ (C) – (B) – (A)

* replicable: 반복 가능한

- ② (B) – (A) – (C)
- ④ (C) – (A) – (B)

언어모델 Final

- 그래서 입출력이 뭐지??

입력

北	정보탈취	전문	해킹조직	'APT37'	전모	공개
---	------	----	------	---------	----	----



언어모델

BERT/GPT/ELECTRA

언어모델

입력 차원 : 문장개수x문장길이

출력 차원 : 문장개수x문장길이x768

$1 \times 7 \rightarrow 1 \times 7 \times 768$

출력

base: 768

large: 1024

0.01
0.2
0.05
0.4
...

0.04
0.6
0.02
0.1
...

0.02
0.5
0.08
0.3
...

0.13
0.21
0.01
0.32
...

0.53
0.01
0.05
0.2
...

0.1
0.22
0.05
0.34
...

0.07
0.15
0.07
0.25
...

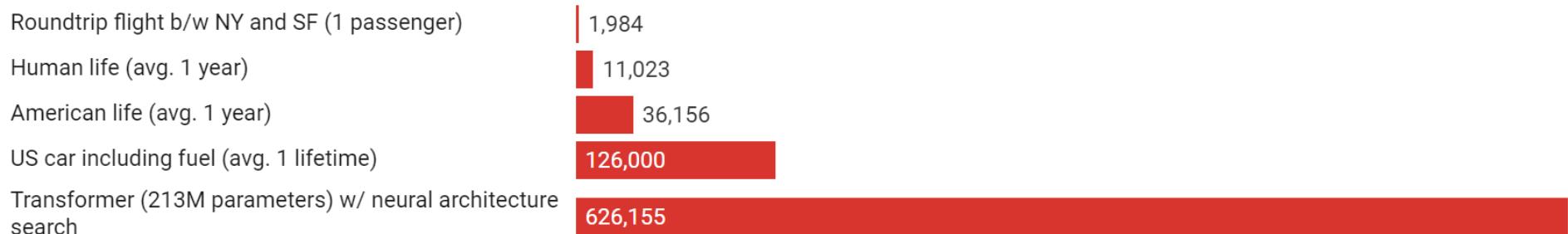
Training Time

Comparison	BERT October 11, 2018	RoBERTa July 26, 2019	DistilBERT October 2, 2019	ALBERT September 26, 2019
Parameters	Base: 110M Large: 340M	Base: 125 Large: 355	Base: 66	Base: 12M Large: 18M
Layers / Hidden Dimensions / Self-Attention Heads	Base: 12 / 768 / 12 Large: 24 / 1024 / 16	Base: 12 / 768 / 12 Large: 24 / 1024 / 16	Base: 6 / 768 / 12	Base: 12 / 768 / 12 Large: 24 / 1024 / 16
Training Time	Base: 8 x V100 x 12d Large: 280 x V100 x 1d	1024 x V100 x 1 day (4-5x more than BERT)	Base: 8 x V100 x 3.5d (4 times less than BERT)	[not given] Large: 1.7x faster
Performance	Outperforming SOTA in Oct 2018	88.5 on GLUE	97% of BERT-base's performance on GLUE	89.4 on GLUE
Pre-Training Data	BooksCorpus + English Wikipedia = 16 GB	BERT + CCNews + OpenWebText + Stories = 160 GB	BooksCorpus + English Wikipedia = 16 GB	BooksCorpus + English Wikipedia = 16 GB
Method	Bidirectional Transformer, MLM & NSP	BERT without NSP, Using Dynamic Masking	BERT Distillation	BERT with reduced parameters & SOP (not NSP)

Training Cost

Date of original paper	Energy consumption (kWh)	Carbon footprint (lbs of CO2e)	Cloud compute cost (USD)
Transformer (65M parameters)	27	26	\$41-\$140
Transformer (213M parameters)	201	192	\$289-\$981
ELMo	275	262	\$433-\$1,472
BERT (110M parameters)	1,507	1,438	\$3,751-\$12,571 1700만
Transformer (213M parameters) w/ neural architecture search	656,347	626,155	\$942,973-\$3,201,722 41억
GPT-2	-	-	\$12,902-\$43,008 5500만

Training Time



NLP for general domains

- BERT(Pre-training of Deep Bidirectional Transformers for Language Understanding)
 - 대용량 데이터와 Transformer의 Encoder를 이용한 언어모델
 - 특정 분야에 국한된 기술이 아니라 모든 자연어 응용 분야에서 좋은 성능을 내는 범용 모델
 - 많은 자연어 처리 분야(GLUE)에서 기존 모델들을 가볍게 누르며 1위를 차지

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	Average -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
BERT _{BASE}	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
BERT _{LARGE}	86.7/85.9	72.1	92.7	94.9	60.5	86.5	89.3	70.1	82.1

NLP for specific domain

- 꽤 많은 텍스트 비정형 데이터는 general domain LM로 해결 가능
- But, 현실은??
- 의료, 제약, 법률, 특허, 보안 등 specific domain(특정 분야)에서는??
- SNS(tweet), 채팅대화(kakao), 댓글, 다크웹 에서는??
- Domain-specific NLP??

NLP for specific domain

- 쓸아지는 방대한 ** 데이터는 대부분 **중복 정보 or 필요 없는 정보**가 대부분
- But, 소수의 중요한 정보는 대부분의 필요 없는 정보에 가려 **활용되지 못하고 있음**

□ [한국거래소 디도스 공격 받아.2시간가량 접속 제한](#) 2020.08.26. 오후 6:01

한국거래소 홈페이지가 디도스(DDoS·분산서비스거부) 공격으로 접속 지연 현상을 겪었다. 26일 한국거래소에 따르면 한국거래소 홈페이지는 이날 오후 12시45분쯤 디도스 공격을 받았다. 거래소가 급히 복구작업에 나서면서 접속 지연은 공격 후 3시간가량 이어졌다. 한국거래소 관계자는 "디도스 공격으로 인해 오후 2시쯤부터 4시 25분까지 홈페이지 접속이..."

- ↳ [한국거래소, 디도스 공격에 한때 홈페이지 접속 중단](#) 2020.08.26. 오후 6:01
- ↳ [한국거래소 디도스 공격.약 2시간 접속 제한](#) 2020.08.28. 오후 4:10
- ↳ [한국거래소, 홈페이지 디도스 공격받아.한때 접속 멈통](#) 2020.08.30. 오후 7:17
- ↳ [한국거래소, 디도스 공격에 홈페이지 한때 '멎통'](#) 2020.08.27. 오후 12:00
- ↳ [한국거래소도 디도스 공격 받았다.카뱅·신한은행 등 이달만 4번째](#) 2020.08.27. 오후 5:30
- ↳ [금융권 잇단 디도스 공격에 거래소도 노출."같은 조직 소행 추정"](#) 2020.08.26. 오전 10:39

BERT for specific domain

- Wikipedia + Google 데이터로 학습된 Bert embedding
- 아무리 많은 데이터(33억)로 학습했고
- 일반 NLP에서 좋은 성능을 내지만
- 과연 특정 분야 보고서에도 잘 동작할까?

BERT for specific domain

■ **분야 자연어처리의 어려움

- CVE-2015-2545 EPS exploit ([image1.eps](#), MD5 **a90a329335fa0af64d8394b28e0f86c1**)
- The decryption function is 1-byte XOR with key from "\x00" to "\xff" and replacement of the Odd byte
- MD5 : 4de21c3af64b3b605446278de92dfff4
SHA1 : 8180c24445b162ce3338ee2ce77053acc08cda88
File Size : 65,536 bytes
Timestamp : 2018:05:20 00:47:58
File Type : PE32 executable (DLL) (GUI) Intel 80386, for MS Windows
Known file name : **core.dll**
Original file name : **OneDII.dll**
- **try{
 log.open("GET", "http://suppcrt-seourity.esy[.]es/update/templates/index.php?v=pp", false);
 log.send();
}
catch(e){}**

BERT

- **분야

- CVE-2015-

- The decry

- byte

- MD5 : 4de

- SHA1 : 818

- File Size : 0

- Timestamp

- File Type : target domain

- Known file name : core.dll

- Original file name : OneDII.dll

- try{

- log.open("GET", "http://suppcrt-security.esy[.]es/update/templates/inaox.php?v=pp", false);

- log.send();

- }catch(e){}

도메인 데이터

일반 데이터

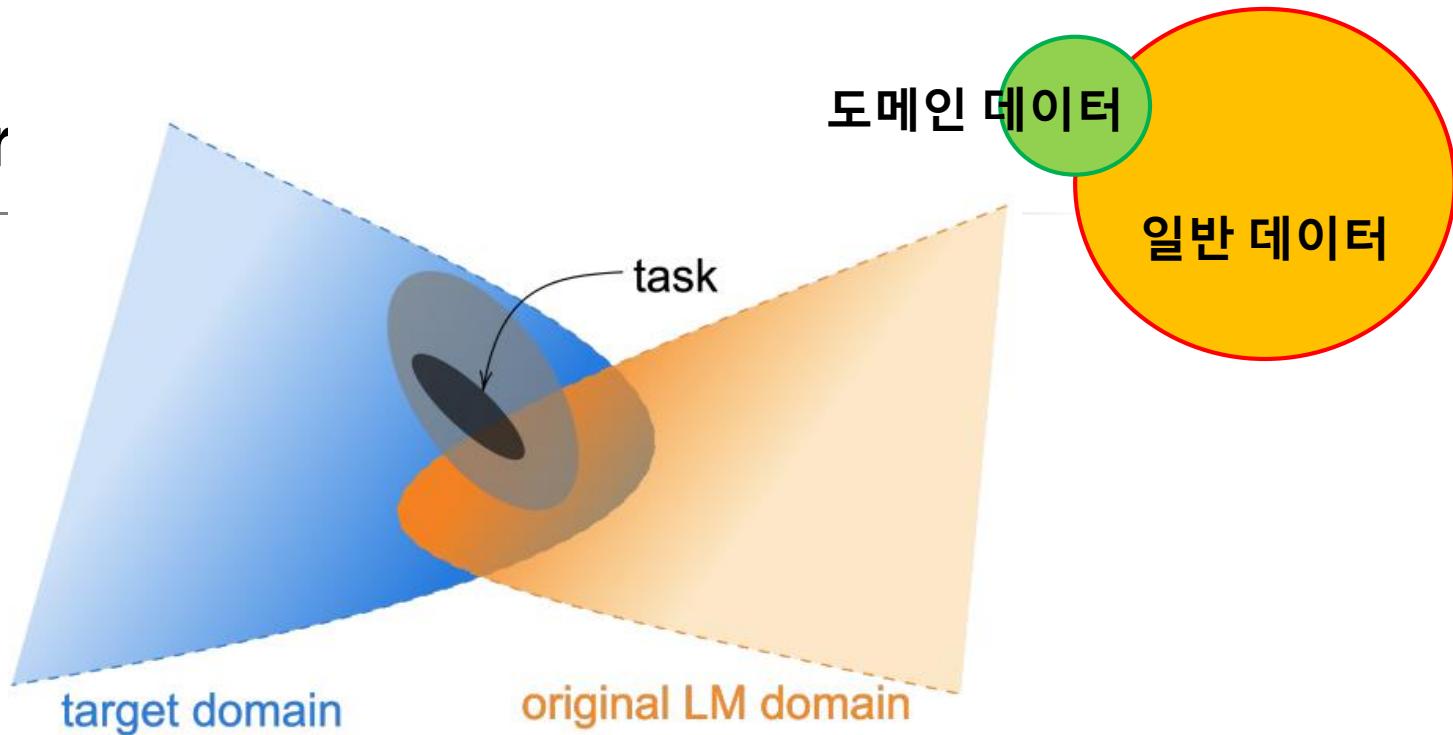
task

of the Odd

original LM domain

**분야에서만 사용되는
단어, 파일명, 16진수, 코드는
일반 언어로 학습한 BERT로는
이해하기 어려움

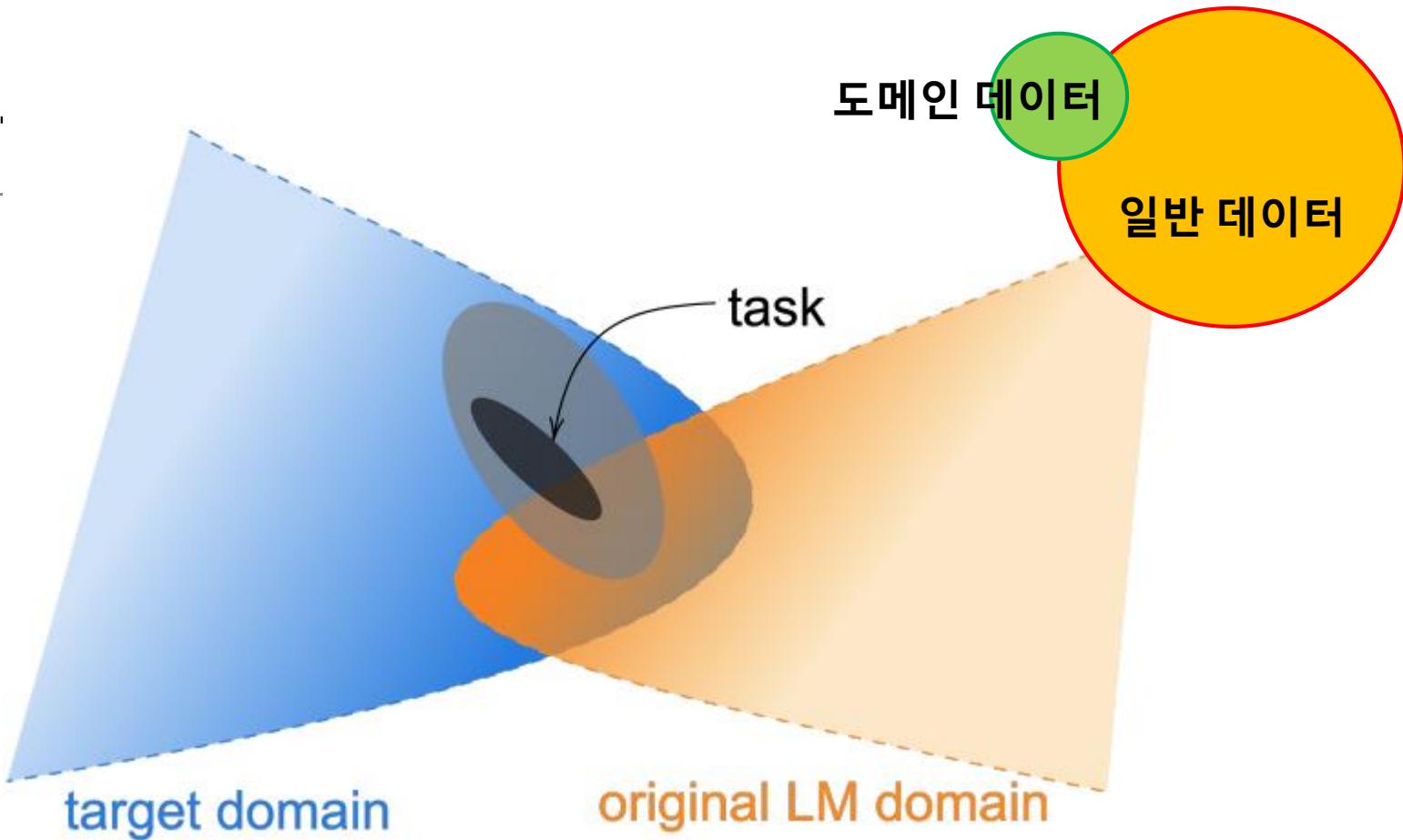
BERT for



RoBERTa	100.0	54.1	34.5	27.3	19.2
뉴스	54.1	100.0	40.0	24.9	17.3
아마존 리뷰	34.5	40.0	100.0	18.3	12.7
생명공학 논문	27.3	24.9	18.3	100.0	21.4
컴퓨터공학 논문	19.2	17.3	12.7	21.4	100.0
	RoBERTa	뉴스	아마존 리뷰	컴퓨터공학 논문	생명공학 논문

도메인 간 Vocabulary overlap

BERT



- General하게 다양한 도메인을 '어느 정도' 커버할 수 있는 PLM도 물론 좋고
- 사용 환경에 따라 특정 도메인만 커버할 수 있지만
- 해당 도메인에서는 General-domain PLM 보다 좋은 성능을 보일 수 있는 모델 필요

Domain-Specific Bert

BioBERT: a pre-trained biomedical language representation model for biomedical text mining

Jinhyuk Lee^{1,1,4}, Wonjin Yoon^{1,1}, Sungdong Kim², Donghyeon Kim¹,
Sunkyu Kim¹, Chan Ho So³ and Jaewoo Kang^{1,3,*}

¹Department of Computer Science and Engineering, Korea University, Seoul 02841, Korea

²Clova AI Research, Naver Corp, Seong-Nam 13561, Korea

³Interdisciplinary Graduate Program in Bioinformatics, Korea University, Seoul 02841, Korea

⁴These authors contributed equally to this work.

¹Most work was done while the author was an intern at Clova AI Research.

*To whom correspondence should be addressed.

Associate Editor: XXXXXXX

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Abstract

Motivation: Biomedical text mining is becoming increasingly important as the number of biomedical documents rapidly grows. With the progress in machine learning, extracting valuable information from the biomedical literature has gained popularity among researchers, and deep learning has boosted the development of effective biomedical text mining models. However, as deep learning models require a large amount of training data, applying deep learning to biomedical text mining is often unsuccessful due

Table 1. List of text corpora used for BioBERT

Corpus	# of words (B)	Domain
English Wikipedia	2.5B	General
BooksCorpus	0.8B	General
PubMed Abstracts	4.5B	Biomedical
PMC Full-text articles	13.5B	Biomedical

Table 2. Combinations of text corpora for pre-training BioBERT. Wiki denotes English Wikipedia, Books denotes BooksCorpus, PubMed denotes PubMed abstracts, and PMC denotes PMC full-text articles.

Corpus	Domain
Wiki + Books	General
Wiki + Books + PubMed	General + Biomedical
Wiki + Books + PMC	General + Biomedical
Wiki + Books + PubMed + PMC	General + Biomedical

SCI-BERT: Pretrained Contextualized Embeddings for Scientific Text

Iz Beltagy Arman Cohan Kyle Lo
Allen Institute for Artificial Intelligence, Seattle, WA, USA
{beltagy, armanc, kylel}@allenai.org

Abstract

Obtaining large-scale annotated data for NLP tasks in the scientific domain is challenging and expensive. We release SCIBERT, a new pre-trained contextualized language model based on BERT (Devlin et al., 2018) to address the lack of high-quality, large-scale labeled scientific data. SCIBERT leverages unsupervised pretraining on a large multi-domain corpus of scientific publications to improve performance in two state-of-the-art tasks: evaluation of scientific text in multiple genres, tagging, sentence classification and dependency parsing, with datasets from a variety of scientific domains. We demonstrate statistically significant improvements over BERT and achieve new state-of-the-art results on several of these tasks.

the success of these models across a variety of NLP tasks, leveraging unsupervised pretraining has become useful especially when task-specific annotations are difficult to obtain. Yet while both BERT and SCIBERT are pre-trained on large amounts of unlabeled data, they are still trained on general domain corpora such as news articles and Wikipedia.

In this work, we make the following contributions:

(i) We release SCIBERT, a new resource to address the lack of annotated data for NLP tasks in the scientific domain. SCIBERT is based on BERT trained on a large corpus of scientific text. The code and pre-trained models are available at <https://github.com/allenai/scibert>.

(ii) We evaluate SCIBERT against BERT on a variety of tasks in the scientific domain, including

FinBERT: Financial Sentiment Analysis with Pre-trained Language Models

Dogu Tan Araci
dogu.araci@student.uva.nl
University of Amsterdam
Amsterdam, The Netherlands

ABSTRACT

Financial sentiment analysis is a challenging task due to the specialized language and lack of labeled data in that domain. General-purpose models are not effective enough because of specialized language used in financial context. We hypothesize that pre-trained language models can help with this problem because they require few labeled examples and then can further train on domain specific corpora. We introduce FinBERT, a language model based on BERT, to tackle NLP tasks in financial domain. Our results show improvement in every measured metric on current state-of-the-art results for two financial sentiment analysis datasets. We find that FinBERT with a small finetuning achieves better results than BERT initialized with a large finetuning, using only a part of the data. FinBERT is a form of domain specific machine learning methods.

NLP transfer learning methods look like a promising solution to both of the challenges mentioned above, and are the focus of this work. The core idea behind these methods is that by training language models on very large corpora and then initializing down-stream models with the weights learned from the language modeling task, a much better performance can be achieved. The initialized layers can range from the single word embedding layer [23] to the whole model [5]. This approach should, in theory, be an answer to the scarcity of labeled data problem. Language models don't require any labels, since the task is predicting the next word. They can learn how to represent the semantic information. That leaves the fine-tuning on labeled data only the task of predicting the next word. We can learn how to represent the semantic information. That leaves the fine-tuning on labeled data only the task of predicting the next word.

One particular component of the transfer learning methods is the ability to further train the language model on domain specific

BIO, Science, Financial 분야 데이터셋으로 BERT 언어모델 학습
NER, 주제분류, OA 등 모든 자연언어처리 대회 최고성능
특정 도메인은 도메인 특화된 언어모델 개발 필요

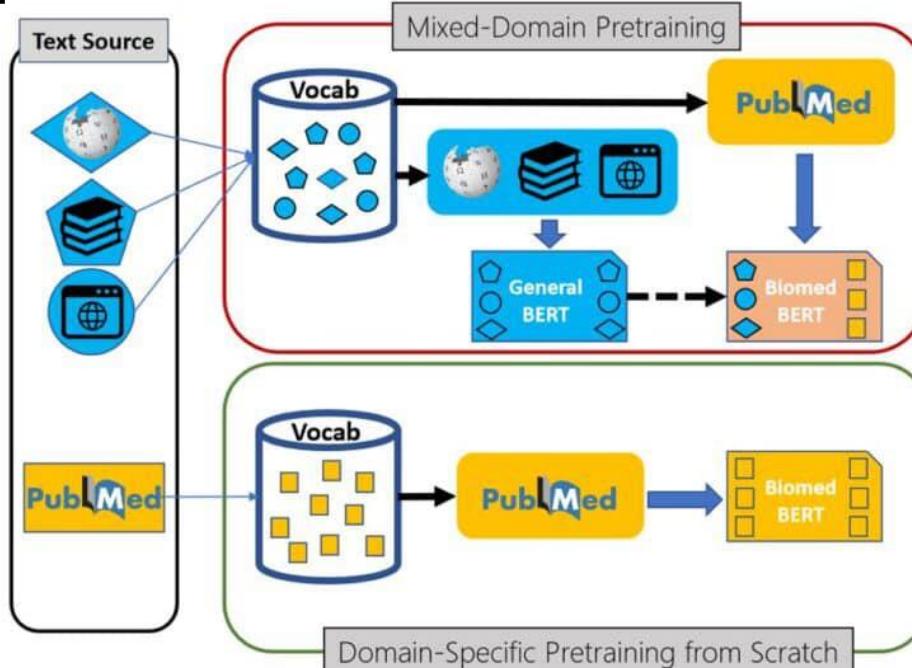
	Corpus	# of words
BERT	English Wiki	2.5B
	BooksCorpus	0.8B
SCIBERT	Biomedical	2.5B
	Computer Science	0.6B

Table 1: Size of pretraining corpora

Domain-Specific PLM

도메인 데이터

일반 데이터



▪ Continual Pretraining(Domain-Adaptive Pretraining)

- General LM에 Domain 데이터로 추가 학습
- General 코퍼스에서도 분명 모델이 배울 것들이 있다!(많은 관심)
- 저렴한 리소스로 학습
- But, vocab 정보를 공유하는 환경으로 도메인 한정적인 고유명사의 커버리지가 낮음
 - `log.open("GET", "http://suppcrt-seecurity.esy[.]es/update/templates/index.php?v=pp", false);`

▪ Domain-specific 코퍼스만을 활용하여 사전학습

Domain-Specific Bert

도메인 데이터

일반 데이터

■ **분야에 특화된 언어모델

- ** 관련 문서 fine-tuning
- ** 보고서, 논문 4만 문서 추가 학습



Start with which corpus?

Corpus	Size (words)
American	155 billion
British	34 billion
Spanish	45 billion

[Compare to standard Google Books interface]

Mark Davies
Brigham Young University



WIKIPEDIA
The Free Encyclopedia

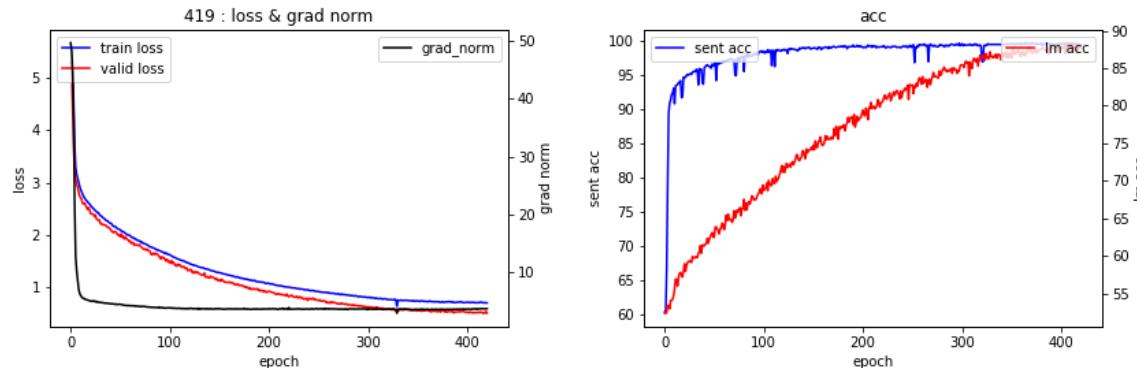


** 비정형 데이터(4만)

논문, 보고서, 블로그, 뉴스, ...

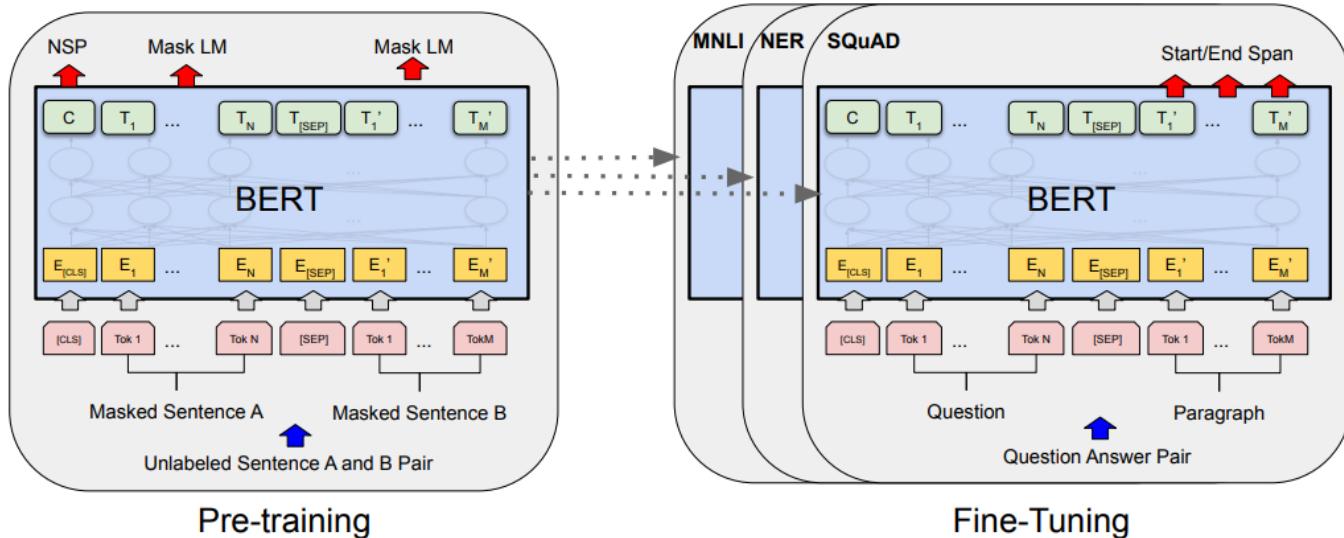
Domain-Specific Bert 구축의 어려움

- ** 분야 데이터셋 구축을 위해 크롤링.....
 - arxiv 논문 크롤링 중 arxiv로 부터 ban을 당해 실원 전체가 반년간 arxiv 접근 못함.....꼭 VPN 사용 추천
 - 논문 내용엔 수식, 기호, 그래프가 많아 introduction만 사용
- 모든 **보고서 텍스트화 후 완벽한 본문만 뽑아야 함
- 보고서마다 형식이 달라서 엄청난 노가다
 - Caption, page, table, 개행 문자 등등 처리
 - PDF, HTML 12종에 대해 각각 12개의 parser 개발(개발 기간의 절반)
- 학습 코드 부재
 - PPL 기반 학습코드는 존재했지만 분야에 특정분야 데이터로 추가학습이기 때문에 오버피팅 발생
 - MLM, NSP의 학습 추이를 보면 best epoch 선정 코드 설계



Domain-Specific Bert

- ** 데이터의 의미적, 문법적 정보를 충분히 담도록 Pre-training 완료
- But, 이 자체로는 downstream-task 수행 불가
- Fine-tuning 필요
 - Pre-training 이후 추가 학습을 수행해 임베딩을 특정 task에 맞게 업데이트
ex) 번역/요약/개체명인식/QA ..

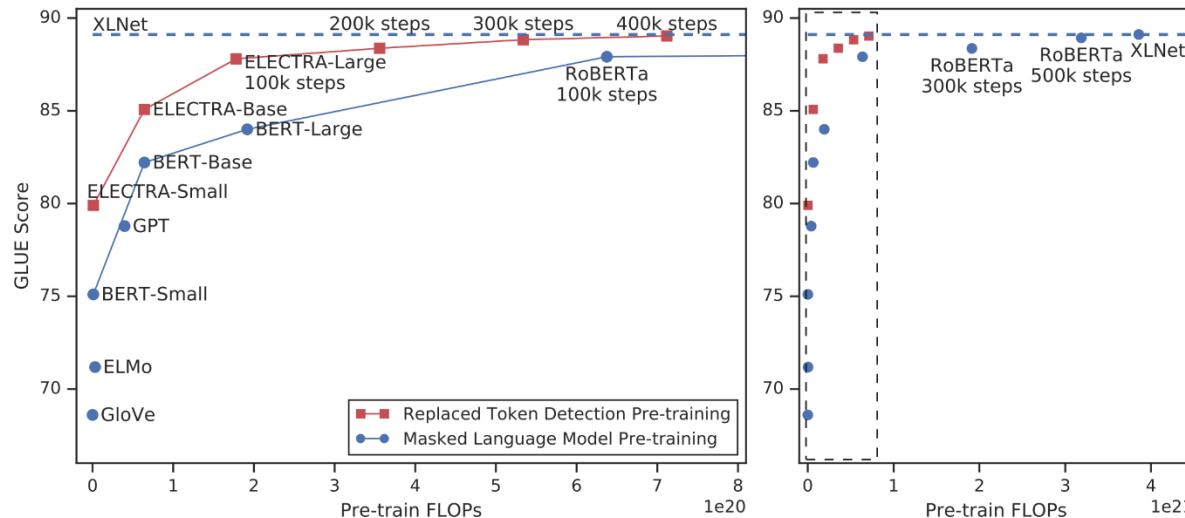
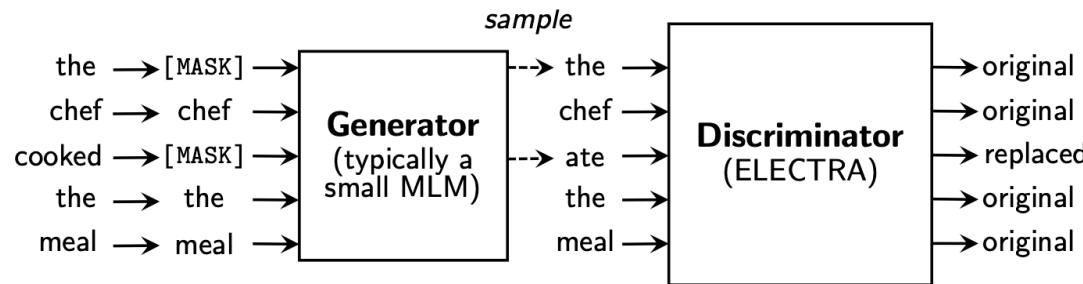


언어 모델링

ELECTRA, MASS

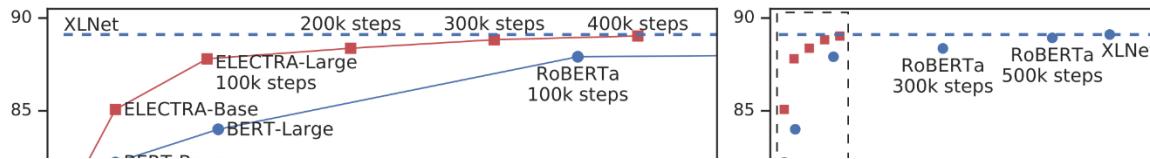
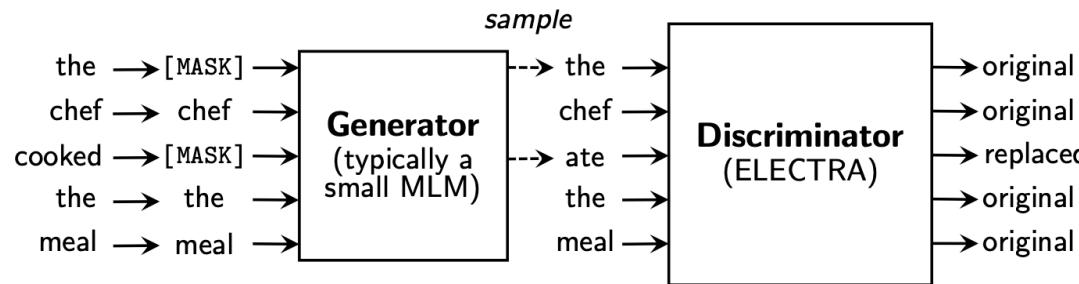
ELECTRA

- 가장 좋은 성능을 보이는 언어모델
- 적은 학습시간으로 더 좋은 성능
- 한국어 뉴스(37G)+모두의말뭉치(20G)로 학습



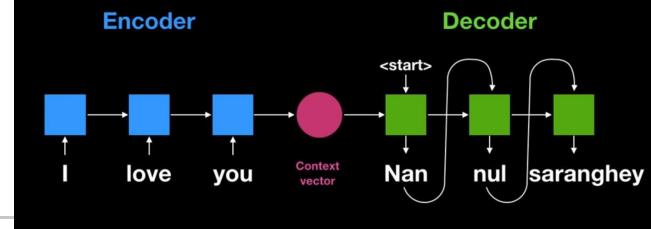
ELECTRA

- 가장 좋은 성능을 보이는 언어모델
- 적은 학습시간으로 더 좋은 성능
- 한국어 뉴스(37G)+모두의말뭉치(20G)로 학습

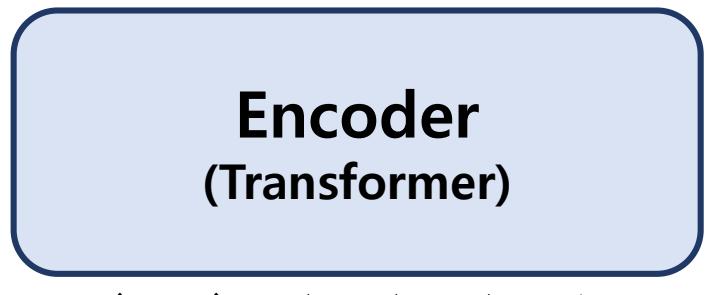


Model	Train FLOPs	CoLA	SST	MRPC	STS	QQP	MNLI	QNLI	RTE	WNLI	Avg.*	Score
BERT	1.9e20 (0.06x)	60.5	94.9	85.4	86.5	89.3	86.7	92.7	70.1	65.1	79.8	80.5
RoBERTa	3.2e21 (1.02x)	67.8	96.7	89.8	91.9	90.2	90.8	95.4	88.2	89.0	88.1	88.1
ALBERT	3.1e22 (10x)	69.1	97.1	91.2	92.0	90.5	91.3	–	89.2	91.8	89.0	–
XLNet	3.9e21 (1.26x)	70.2	97.1	90.5	92.6	90.4	90.9	–	88.5	92.5	89.1	–
ELECTRA	3.1e21 (1x)	71.7	97.1	90.7	92.5	90.8	91.3	95.8	89.8	92.5	89.5	89.4

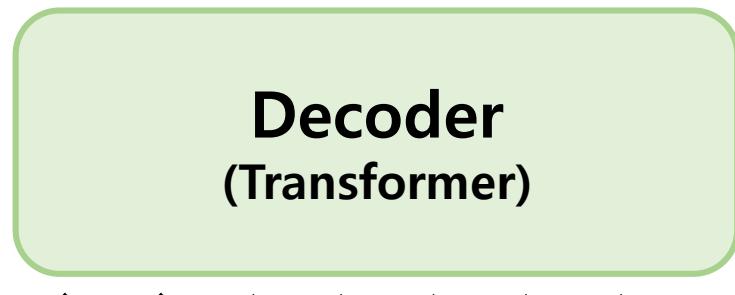
MASS



나는 중국 음식 먹고 싶어 !



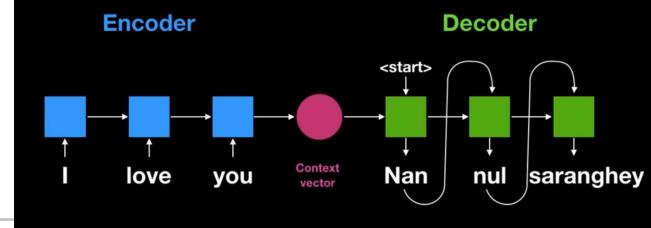
Attention
→



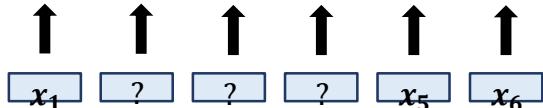
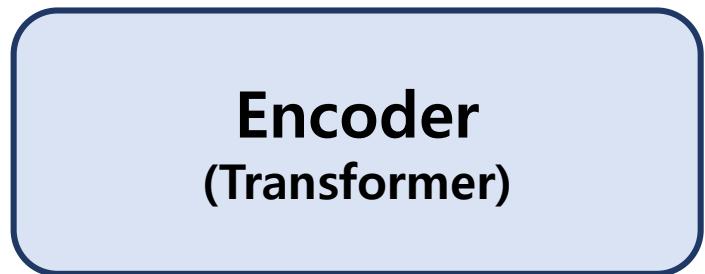
x_1 x_2 x_3 x_4 x_5 x_6

$[BOS]$ x_1 x_2 x_3 x_4 x_5 x_6

MASS

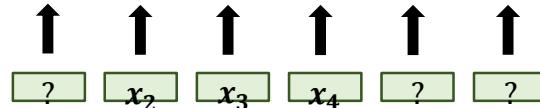
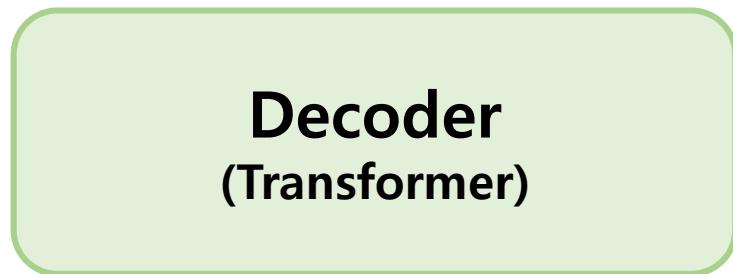


나는 중국 음식 먹고 싶어 !



나는 ? ? ? 싶어 !

Attention →



중국 음식 먹고
 x_2 x_3 x_4

↑ ↑ ↑

Method	en-fr	fr-en	en-de	de-en	en-ro	ro-en
Artetxe et al. (2017)	2-layer RNN	15.13	15.56	6.89	10.16	-
Lample et al. (2017)	3-layer RNN	15.05	14.31	9.75	13.33	-
Yang et al. (2018)	4-layer Transformer	16.97	15.58	10.86	14.62	-
Lample et al. (2018)	4-layer Transformer	25.14	24.18	17.16	21.00	21.18
XLM (Lample & Conneau, 2019)	6-layer Transformer	33.40	33.30	27.00	34.30	33.30
MASS	6-layer Transformer	37.50	34.90	28.30	35.20	33.10

Method	en-fr	fr-en	en-de	de-en	en-ro	ro-en
<i>BERT+LM</i>	33.4	32.3	24.9	32.9	31.7	30.4
<i>DAE</i>	30.1	28.3	20.9	27.5	28.8	27.6
MASS	37.5	34.9	28.3	35.2	35.2	33.1

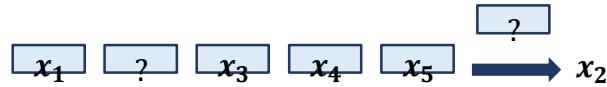
언어 모델링

GPT

ref: <https://jalammar.github.io/illustrated-gpt2/>

Language Model : Auto-Encoding VS Auto-Regressive

Auto Encoding



입력 문장

$$\bar{x} = [x_1, x_2, x_3, x_4, x_5]$$

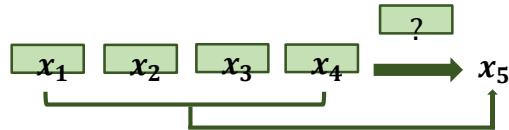
오염된 문장

$$\hat{x} = [x_1, [MASK], x_3, x_4, x_5]$$

likelihood

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

Auto Regressive



입력 문장

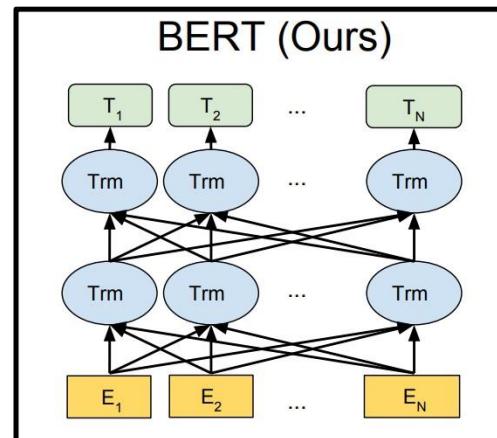
$$\bar{x} = [x_1, x_2, x_3, x_4]$$

다음 단어(정답)

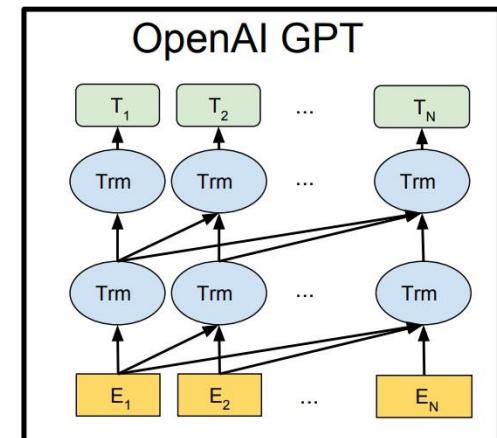
$$x = x_5$$

likelihood

$$p(x) \approx \prod_{t=1}^T p(x_t|x_{<t})$$



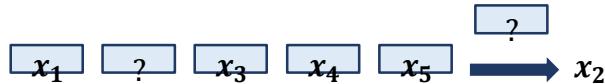
BERT (Ours)



OpenAI GPT

Language Model : Auto-Encoding VS Auto-Regressive

Auto Encoding



입력 문장

$$\bar{x} = [x_1, x_2, x_3, x_4, x_5]$$

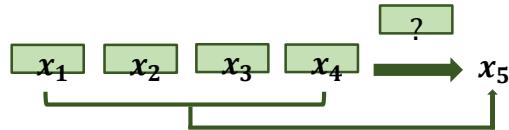
오염된 문장

$$\hat{x} = [x_1, [MASK], x_3, x_4, x_5]$$

likelihood

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

Auto Regressive



입력 문장

$$\bar{x} = [x_1, x_2, x_3, x_4]$$

다음 단어(정답)

$$x = x_5$$

likelihood

$$p(x) \approx \prod_{t=1}^T p(x_t | x_{<t})$$

	BERT	GPT
입력/출력	입력: N개의 단어 열 출력: N개 단어의 인코딩 벡터	입력: N개의 단어 열 출력: N+1번째 단어
모델 특징	양방향 문맥 정보 활용	단방향 문맥 정보 활용
사전학습 태스크	입력: 임의의 단어 masking 출력: 주위 문맥으로 해당 단어 맞추기	입력: 이전 단어 열 출력: 다음 단어 예측
비고	동일 문장이라도 random masking 위치에 따라 서로 다른 정보 학습 (→ random masking을 수 차례 반복하여 학습)	동일 문장이면 동일 정보 학습
한계	Downstream task에 Fine-tuning 시 [MASK] 토큰이 등장하지 않음, Pre-training과 Fine-tuning의 학습 목표 불일치	단방향 정보만으로 학습 가능함, 양방향 정보 이용 불가

언어모델 Final

- 그래서 입출력이 뭐지??

입력

北	정보탈취	전문	해킹조직	'APT37'	전모	공개
---	------	----	------	---------	----	----



언어모델

BERT/GPT/ELECTRA

언어모델

입력 차원 : 문장개수x문장길이

출력 차원 : 문장개수x문장길이x768

$1 \times 7 \rightarrow 1 \times 7 \times 768$

출력

base: 768

large: 1024

0.01
0.2
0.05
0.4
...

0.04
0.6
0.02
0.1
...

0.02
0.5
0.08
0.3
...

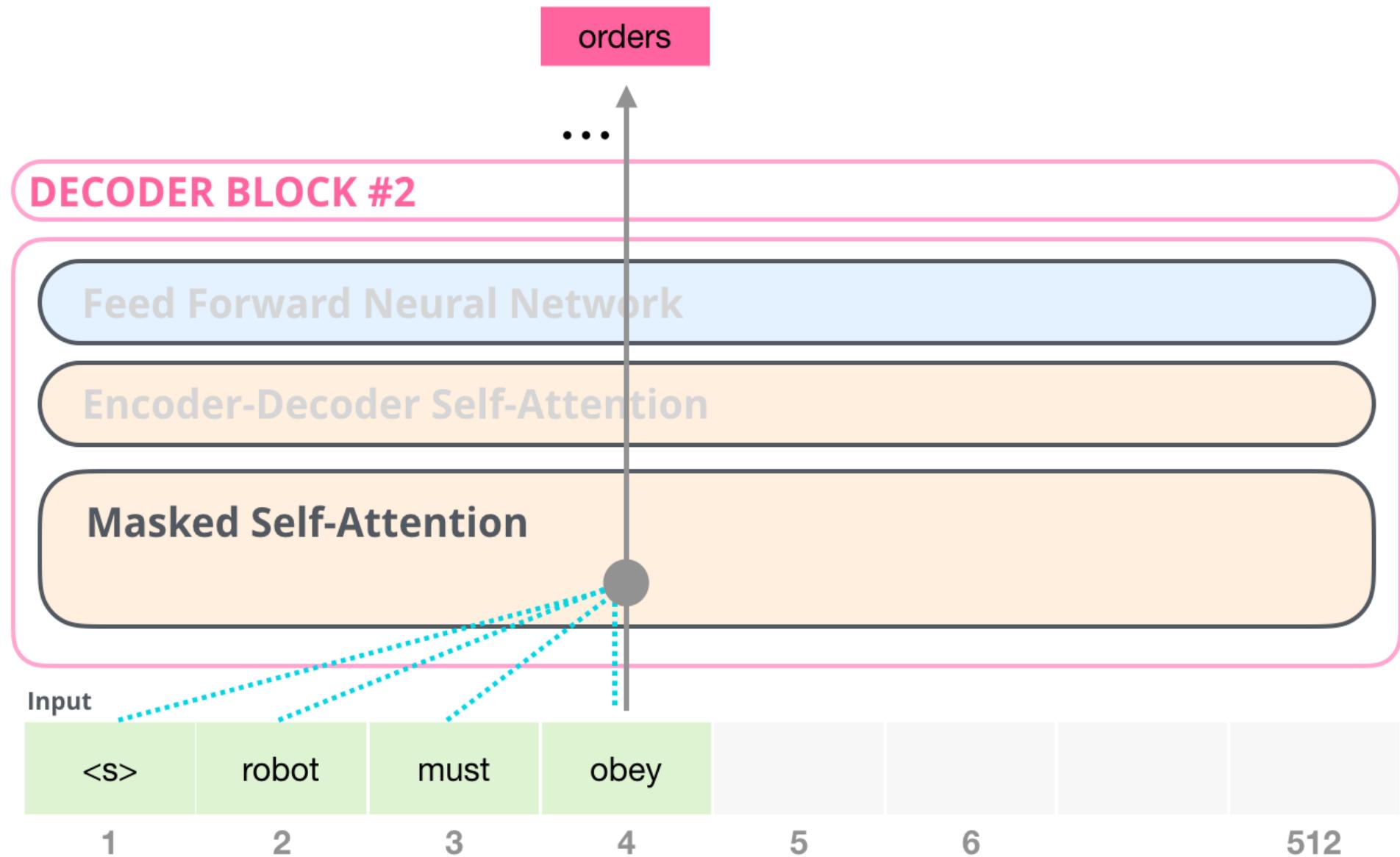
0.13
0.21
0.01
0.32
...

0.53
0.01
0.05
0.2
...

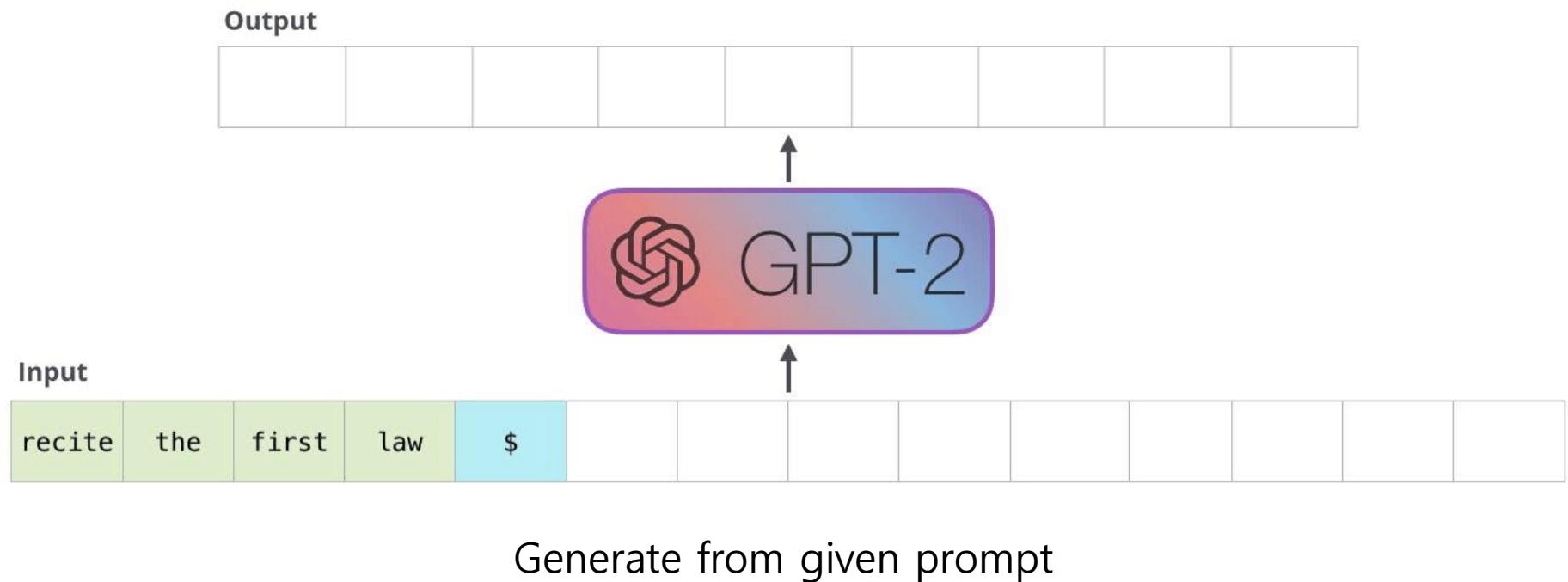
0.1
0.22
0.05
0.34
...

0.07
0.15
0.07
0.25
...

Language Model : Auto-Encoding VS Auto-Regressive



GPT



단순, ##함, ##을, 얻기란, 복잡함, ##을, 얻기, 보다, 어렵다 [SEP]



[CLS], 단순, ##함, ##을, 얻기란, 복잡함, ##을, 얻기, 보다, 어렵다



Auto Regressive Model



[CLS] 단순, ##함, ##을 얻기란, 복잡함, ##을, 얻기, 보다, 어렵다 [SEP]



단순함을 얻기란 복잡함을 얻기보다 어렵다.

GPT



117M Parameters

1.1억

345M Parameters

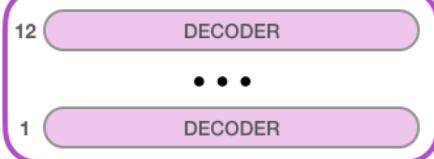
762M Parameters

1,542M Parameters
15억 61/232

GPT



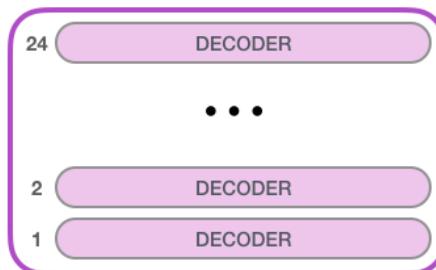
GPT-2
SMALL



Model Dimensionality: 768



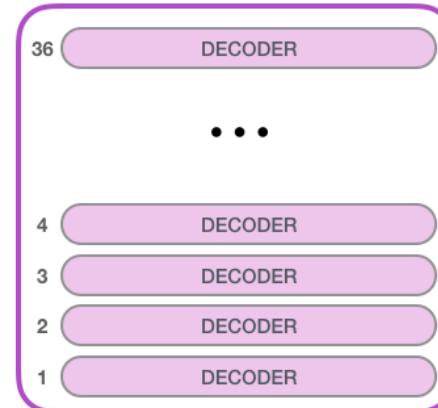
GPT-2
MEDIUM



Model Dimensionality: 1024



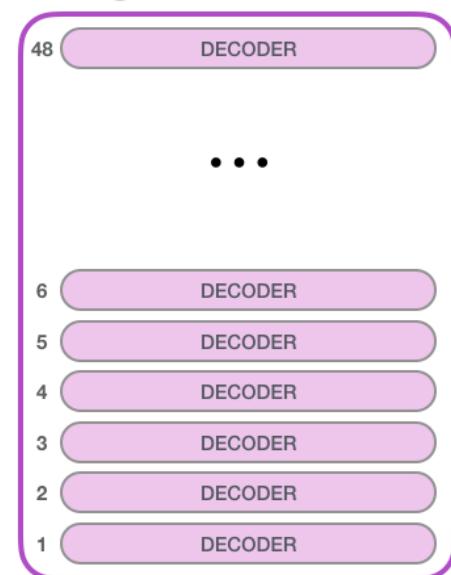
GPT-2
LARGE



Model Dimensionality: 1280



GPT-2
EXTRA
LARGE



Model Dimensionality: 1600

GPT

Model Name	n_{params}
GPT-3 Small	125M
GPT-3 Medium	350M
GPT-3 Large	760M
GPT-3 XL	1.3B
GPT-3 2.7B	2.7B
GPT-3 6.7B	6.7B
GPT-3 13B	13.0B
GPT-3 175B or “GPT-3”	175.0B

15억 VS 1750억



117M Parameters

1.1억



345M Parameters



762M Parameters



1,542M Parameters

15억
63/232

GPT-2 vs GPT-3

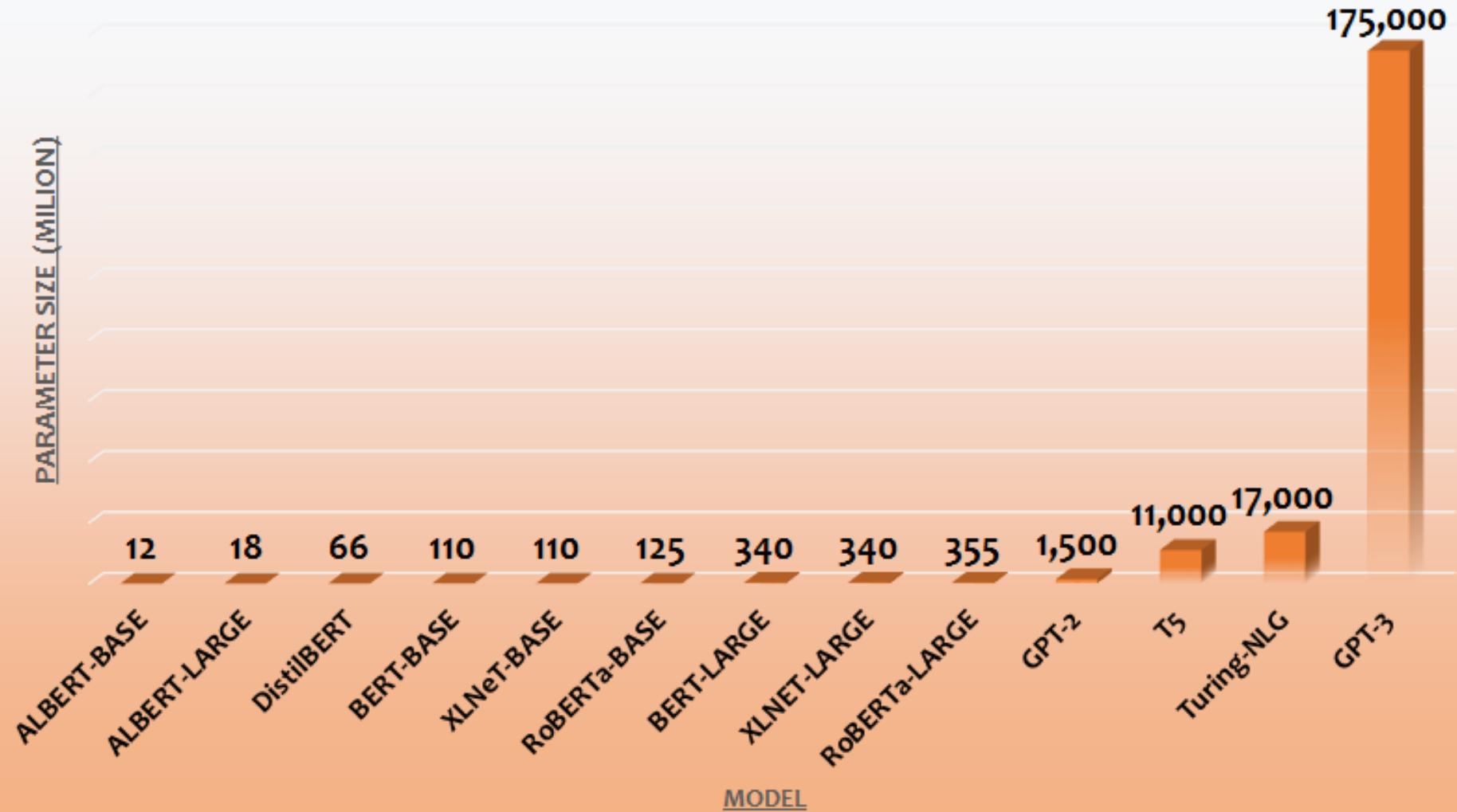
Model Name	n_{params}
GPT-3 Small	125M
GPT-3 Medium	350M
GPT-3 Large	760M
GPT-3 XL	1.3B
GPT-3 2.7B	2.7B
GPT-3 6.7B	6.7B
GPT-3 13B	13.0B
GPT-3 175B or "GPT-3"	175.0B

15억 VS 1750억

GPT-3

GPT-2
EXTRA LARGE

COMPARISON: NLP PRE-TRAINED MODELS



언어모델 개발 로드맵

Language Modeling

(BERT, GPT, ELECTRA, MASS)

언어모델 개발 로드맵

- 1) 대량 코퍼스 수집
- 2) 코퍼스 정제
- 3) 토크나이저 학습
- 4) 정제된 코퍼스 토크나이징

- 5) 모델 선정
- 6) 모델 입력데이터 형식으로 변환 코드 작성
- 7) 학습데이터 Pre-generation

- 8) 학습 테스트
 - 모델 저장 경로, interval, 저장 용량 체크
- 9) 학습!
 - 끊임없이 모니터링, 용량체크, 중단체크, 에러체크

한글 코퍼스 수집

■ 뉴스 크롤링

- 네이버 24개 신문사 + ** 뉴스
- 52개 모든 경우를 커버하는 파서를 만드는건 무리 => 파싱이 가능한 24개로 축소
- 신문사 별, 기자 별 포맷이 다 다르다!!! 파싱 기준이 상이하여 통일된 본문 파싱.....하....
- 뉴스 본문은 마지막 '~다.' 까지로 파싱
 - but, 이렇게 해서 안되는 경우가 있음

서울 종로구 옛 주한일본대사관 앞의 소녀상 앞에서 열린 1440차 일본군 위안부 문제 해결을 위한 정기 수요시위에서 일본군성노예제 문제해결을 위한 정의기억연대 이사회 입장문을 발표하고 있다 / 2020.05.20.

misocamera@newsis.com

[사진 영상 제보받습니다] 공감언론 뉴시스가 독자 여러분의 소중한 제보를 기다립니다. 뉴스 가치나 화제성이 있다고 판단되는 사진 또는 영상을 뉴시스 사진영상부(n-photo@newsis.com, 02-721-7470)로 보내주시면 적극 반영하겠습니다.

앞으로 1년간 소비자물가상승률 전망을 나타내는 기대인플레이션율은 한 달 전과 같은 1.7%다. /손철기자 runiron@sedaily.com

[서울경제 바로가기]

- ▶ 네이버 채널에서 '서울경제' 구독해주세요!
- ▶ 세상 모든 것에 대해 던지는 질문, 왜(WHY)?

저작권자 © 서울경제, 무단 전재 및 재배포 금지

한편 전국 엑서스 공식 딜러 서비스 센터에서 진행 중인 '고객 안심 서비스 캠페인'은 6월 13일까지 기간을 연장해 실시한다.

김지희 기자 ways@asiae.co.kr

- ▶ 소름 짹! 2020년 내 대운 시기 확인하기
- ▶ 네이버에서 아시아경제 뉴스를 받아보세요 ▶ 놀 준비 되었다면 드루와! 드링킷!

<©경제를 보는 눈, 세계를 보는 창 아시아경제 무단전재 배포금지>

울산시 교육청은 A씨를 모든 업무에서 배제하고, 담임교사도 바꾸도록 했다. 또 경찰에 해당 사실을 신고했으며 조사가 끝나면 징계 여부를 결정할 방침이다.

박지혜 (noname@edaily.co.kr)

네이버에서 '이데일리' 구독하기▶
청춘뉘우스~ 스냅타임▶

<©종합 경제정보 미디어 이데일리 - 무단전재 & 재배포 금지>

두산그룹이 제출한 자구안에는 두산중공업에 대한 유상증자와 자산매각, 비용 축소 등 자구노력을 통해 3조원 이상을 확보하는 방안이 담겼다. 특히 두산 오너들은 두산중공업에 사재를 출연할 방침이다.

김도윤 기자 justice@

- ▶ 졸리아 투자노트
- ▶ 조 변호사의 가정상담소 ▶ 머니투데이 구독하기

<저작권자 © '돈이 보이는 리얼타임 뉴스' 머니투데이, 무단전재 및 재배포 금지>

뉴스 크롤링

▪ 뉴스 크롤링 데이터 정제

- 데이터 정제는 끝이 없다. 가장 힘들고 긴 시간이 걸리지만 중요한 작업
 - 어디까지, 얼마나 할 것인가 X
 - 기간을 정해두고 언제까지 하겠다고 결정하기를 추천
 - 10일동안만 데이터 정제를 한다! 그러고도 정제되지 않은 것들은 그냥 노이즈라 여기고 갖고 가자!

뉴스 크롤링

▪ 뉴스 크롤링 데이터 정제

- 데이터 정제 방식
 - 사람마다 모두 기준과 방식이 다름
 - 정규표현식으로 숫자, 영어, 한글, 개행 + 키보드로 입력할 수 있는 특수문자는 살리고 나머지 문자는 추후 결정
 - pattern = '\[^ 0-9a-zA-Z가-힣 \~`!@#\$%^&*\(\)-_=+\[\{\}\]\}|;\'",.\<\>/?.\t\n]+'
 - test = re.sub(pattern , ", lines)
 - 한달간 크롤링한 뉴스로 걸러지는 특수문자 체크
 - 400개의 키보드로 입력할 수 없는 특수문자 확인
 - 절반이 한자
 - 삭제 시 문서 본문의 의미에 영향을 미칠만한 특수문자들이 많은 것을 확인
 - 단위, 주요 한자(중국, 미국), 증가/감소

↑ : 457,
→ : 439,
↓ : 188.

뉴스 크롤링

▪ 뉴스 크롤링 데이터 정제

- 데이터 정제 방식

- 한달간 크롤링한 뉴스로 걸러지는 특수문자 체크

- 400개의 키보드로 입력할수 없는 특수문자 확인
 - 절반이 한자
 - 삭제 시 문서 본문의 의미에 영향을 미칠만한 특수문자들이 많은 것을 확인
 - 단위, 주요 한자(중국, 미국), 증가/감소
 - 의미 있는 특수문자는 replace로 살리고 삭제

```
lines = lines.replace('↑', '증가').replace('↓', '하락').replace('→', '→').replace('㈜', '주식회사')
lines = lines.replace('外', '외신').replace('比', '대비').replace('新', '신').replace('韓', '한국').replace('史', '역사')
lines = lines.replace('金', '김').replace('女', '여자').replace('年', '년').replace('李', '이').replace('男', '남자')
# 이상한 문자들 교체
lines = lines.replace('%', '%').replace('&', '&').replace('+', '+').replace(' ', ' ').replace('-', '-')
# 로마 숫자
lines = lines.replace('Ⅰ', '1.').replace('Ⅱ', '2.').replace('Ⅲ', '3.').replace('Ⅳ', '4.').replace('Ⅴ', '5.')
lines = lines.replace('@', '1.').replace('@', '2.').replace('@', '3.').replace('@', '4.').replace('@', '5.')
lines = lines.replace('₩', '₩').replace('₩', '₩').replace('₩', '₩').replace('₩', '₩').replace('₩', '₩')
lines = lines.replace('-', '-').replace('-', '-').replace('·', '·').replace('·', '·').replace('·', '·')
lines = lines.replace('₩', '₩').replace('₩', '₩').replace('₩', '₩').replace('₩', '₩').replace('₩', '₩')
```

2) 코퍼스 정제

2) 코퍼스 정제

```
# '₩', '₩', '₩', '₩', '₩' 는 유일함
lines = lines.replace('₩', '₩').replace('₩', '₩').replace('₩', '₩').replace('₩', '₩').replace('₩', '₩')

lines = lines.replace('↔', '↔').replace('↔', '↔')
lines = lines.replace('↑', '↑').replace('↑', '↑').replace('↑', '↑').replace('↑', '↑').replace('↑', '↑')
lines = lines.replace('↓', '↓').replace('↓', '↓').replace('↓', '↓')
lines = lines.replace('⊕', '⊕').replace('⊕', '⊕').replace('⊕', '⊕')
lines = lines.replace('⊖', '⊖')
lines = lines.replace('⌚', '⌚').replace('⌚', '⌚').replace('⌚', '⌚').replace('⌚', '⌚')
lines = lines.replace('⌚', '⌚')

# 풀임말: 온점 2개짜리 풀임말로 통일
lines = re.sub('...', '...', lines)
lines = re.sub('...{3,}', '...', lines) # 3개 이상은 구분자가 아니고 풀임말임 -> ... 로 처리
lines = re.sub('...{3,}', '...', lines) # 3개 이상 ...은 ... 2개로 풀여줌

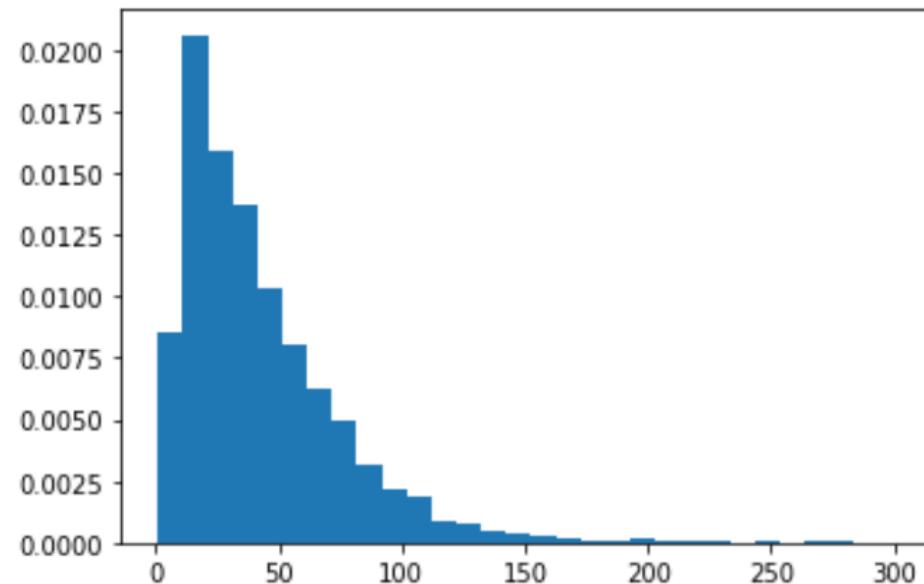
#####
## 2차 전처리 - 교체
#####
lines = lines.replace(''''', '''').replace(''''', '''').replace('---', '--').replace('...', '...').replace('...', '...')

lines = lines.replace('%', '%')
#####
## 원래 문장 제거용 코드
#####
# 원래 문장이 둘째로 사라지는 것인데 그럴수 없으니 해당 문자열만 삭제하기
remove_string = '\r|\x97|\ufe0b1|\ufe0d5|\ufe0df|\ufe028|\ufe029|\ufe02b|\ufe02d|\ufe03d|\ufe040|\ufe044|\ufe05b|\ufe05d|\ufe05e'
remove_string_pattern = re.compile(remove_string)
lines = remove_string_pattern.sub('', lines).strip()

print('정제 완료')
return lines
```

2) 코퍼스 정제

```
total      : 12815
mean  +-  sd  : 43  +-  32
median (IQR) : 34    (39)
25per       < 19
75per       < 58
90per       < 84
95per       < 104
98per       < 130
99per       < 154
max        : 304
```



3) 토크나이저 학습

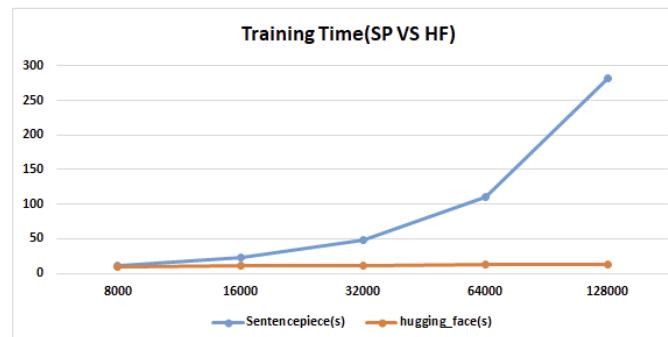
<https://keep-steady.tistory.com/37>

4.1. 학습시간 비교

아래 표는 vocab 사이즈 별 학습시간 비교 표이다. SK의 vocab은 8000이므로 8000부터 2의 배수씩 128000까지 실험하였다. SentencePiece는 15만 문장을 학습시킬 때 vocab_size에 비례하여 학습시간이 늘어났다. 아래 그래프를 보면 더 잘 볼 수 있는데 vocab_size가 제곱으로 늘어날수록 시간도 제곱으로 늘어난다. 즉 vocab을 얼마나 잡느냐가 학습시간이다. 하지만 huggingface는 vocab_size가 증가해도 학습시간이 크게 바뀌지 않았다. NSMC 코퍼스를 12초 내로 학습할 수 있었다. 확실히 학습 속도에서는 huggingface가 SentencePiece를 압도했다.

Subword Tokenizer 학습 시간 비교					
vocab_size	8000	16000	32000	64000	128000
Sentencepiece(s)	11	22	48	110	282
hugging_face(s)	10	11	11	12	12

SentencePiece VS huggingface 학습시간 비교표



SentencePiece VS huggingface 학습시간 그래프

4.2. 분절 시간 비교

NSMC 코퍼스로 학습된 각 모델들로 추론시간을 비교해봤다. 진짜 huggingface가 엄청나게 빠른가?? 의아했다. 아래 표는 15만 문장을 100번 분절하여 걸리는 시간의 평균 표이다. 추론 시 속도 차이는 크지 않았다. huggingface는 학습 시에도 vocab_size에 큰 영향을 받지 않았는데, 추론 시에도 비슷한 속도로 관찰된다. SentencePiece는 vocab_size가 크면 약간 속도가 느려졌다. 하지만 광고하는 것처럼 huggingface의 추론 속도가 SentencePiece에 비해 빠르진 않았다. 오히려 vocab이 적을 땐 SentencePiece이 빨랐다. SentencePiece도 C로 짜여있으므로 추론 시 비슷하다는 결론이다.

vocab_size	8000	128000
Sentencepiece(s)	4.5	4.93
hugging_face(s)	4.9	4.97

4) 정제된 코퍼스 토크나이징

형태소 단위 학습 포함? 미포함? -> mecab

3.6. WordPiece 분절 결과 비교

1) Without Mecab

"나는 오늘 아침밥을 먹었다." =>

- tokens: ['나는', '오늘', '아침', '##밥', '##을', '먹', '##었다', '.']
- idx : [6227, 6390, 7860, 4476, 3291, 1474, 5870, 18]
- offset: [(0, 2), (3, 5), (6, 8), (8, 9), (9, 10), (11, 12), (12, 14), (14, 15)]
- decode: 나는 오늘 아침밥을 먹었다.

2) With Mecab

"나는 오늘 아침밥을 먹었다." =>

- tokens: ['나', '##는', '오늘', '아침', '##밥', '##을', '먹', '##었다', '.']
- idx : [875, 3391, 5446, 6142, 3488, 3421, 1474, 17168, 18]
- offset: [(0, 1), (1, 2), (3, 5), (6, 8), (8, 9), (9, 10), (11, 12), (12, 14), (14, 15)]
- decode: 나는 오늘 아침밥을 먹었다.

5) 모델 선정

언어모델 Final

- 그래서 입출력이 뭐지??

입력

北	정보탈취	전문	해킹조직	'APT37'	전모	공개
---	------	----	------	---------	----	----



언어모델

BERT/GPT/ELECTRA

언어모델

입력 차원 : 문장개수x문장길이

출력 차원 : 문장개수x문장길이x768

$1 \times 7 \rightarrow 1 \times 7 \times 768$

출력

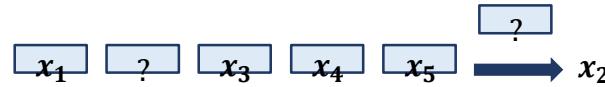
base: 768
large: 1024

0.01	0.04	0.02	0.13	0.53	0.1	0.07
0.2	0.6	0.5	0.21	0.01	0.22	0.15
0.05	0.02	0.08	0.01	0.05	0.05	0.07
0.4	0.1	0.3	0.32	0.2	0.34	0.25
...

5) 모델 선정

Language Model : Auto-Encoding VS Auto-Regressive

Auto Encoding



입력 문장

$$\bar{x} = [x_1, x_2, x_3, x_4, x_5]$$

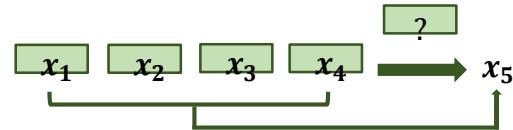
오염된 문장

$$\hat{x} = [x_1, [MASK], x_3, x_4, x_5]$$

likelihood

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

Auto Regressive



입력 문장

$$\bar{x} = [x_1, x_2, x_3, x_4]$$

다음 단어(정답)

$$x = x_5$$

likelihood

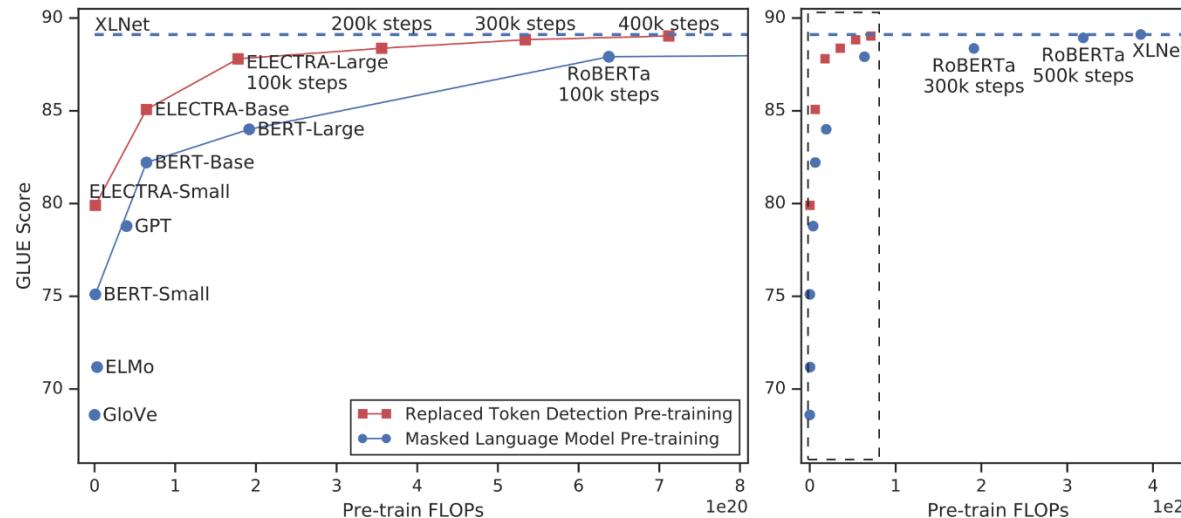
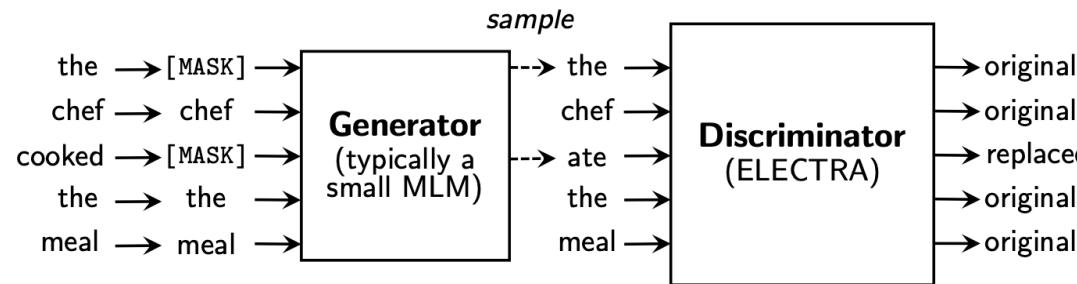
$$p(x) \approx \prod_{t=1}^T p(x_t|x_{<t})$$

	BERT	GPT
입력/출력	입력: N개의 단어 열 출력: N개 단어의 인코딩 벡터	입력: N개의 단어 열 출력: N+1번째 단어
모델 특징	양방향 문맥 정보 활용	단방향 문맥 정보 활용
사전학습 태스크	입력: 임의의 단어 masking 출력: 주위 문맥으로 해당 단어 맞추기	입력: 이전 단어 열 출력: 다음 단어 예측
비교	동일 문장이라도 random masking 위치에 따라 서로 다른 정보 학습 (→ random masking을 수 차례 반복하여 학습)	동일 문장이면 동일 정보 학습
한계	Downstream task에 Fine-tuning 시 [MASK] 토큰이 등장하지 않음, Pre-training과 Fine-tuning의 학습 목표 불일치	단방향 정보만으로 학습 가능함, 양방향 정보 이용 불가

5) 모델 선정

ELECTRA

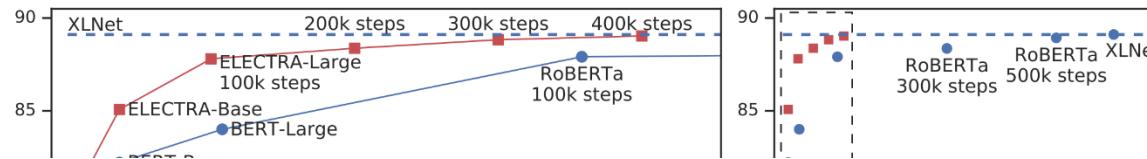
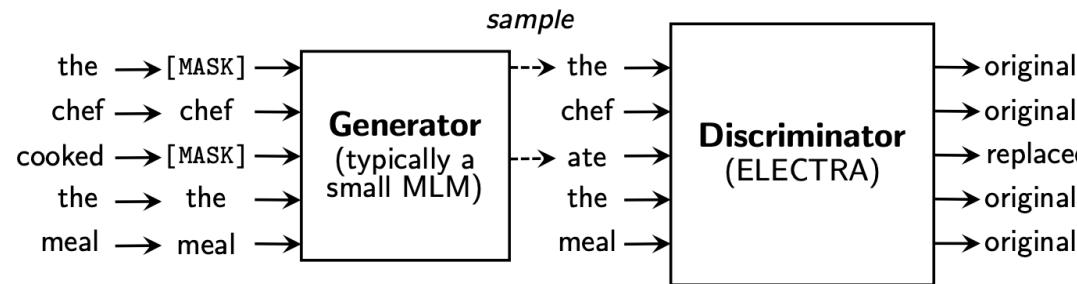
- 가장 좋은 성능을 보이는 언어모델
- 적은 학습시간으로 더 좋은 성능
- 한국어 뉴스(37G)+모두의말뭉치(20G)로 학습



5) 모델 선정

ELECTRA

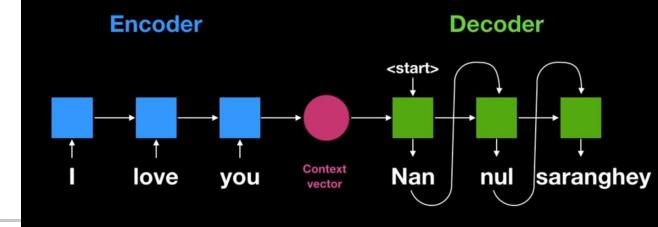
- 가장 좋은 성능을 보이는 언어모델
- 적은 학습시간으로 더 좋은 성능
- 한국어 뉴스(37G)+모두의말뭉치(20G)로 학습



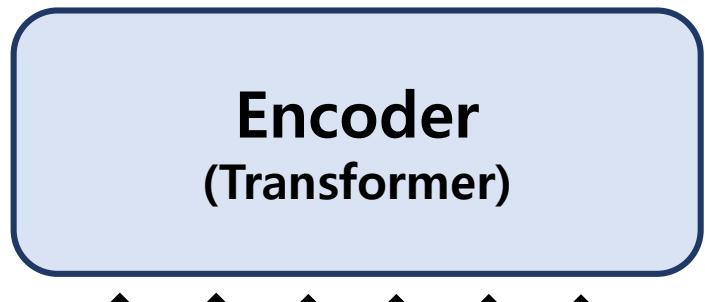
Model	Train FLOPs	CoLA	SST	MRPC	STS	QQP	MNLI	QNLI	RTE	WNLI	Avg.*	Score
BERT	1.9e20 (0.06x)	60.5	94.9	85.4	86.5	89.3	86.7	92.7	70.1	65.1	79.8	80.5
RoBERTa	3.2e21 (1.02x)	67.8	96.7	89.8	91.9	90.2	90.8	95.4	88.2	89.0	88.1	88.1
ALBERT	3.1e22 (10x)	69.1	97.1	91.2	92.0	90.5	91.3	–	89.2	91.8	89.0	–
XLNet	3.9e21 (1.26x)	70.2	97.1	90.5	92.6	90.4	90.9	–	88.5	92.5	89.1	–
ELECTRA	3.1e21 (1x)	71.7	97.1	90.7	92.5	90.8	91.3	95.8	89.8	92.5	89.5	89.4

5) 모델 선정

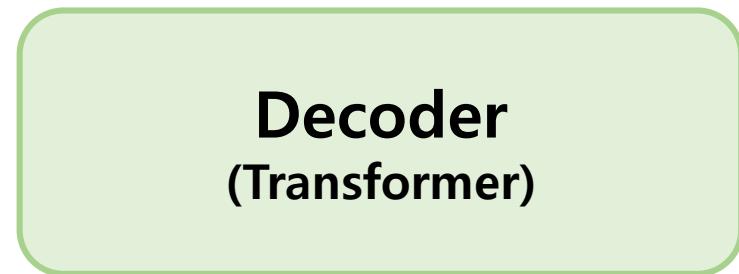
MASS



나는 중국 음식 먹고 싶어 !

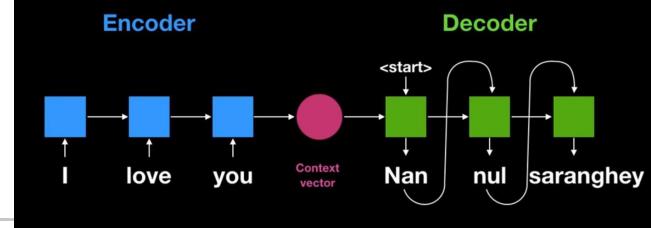


Attention

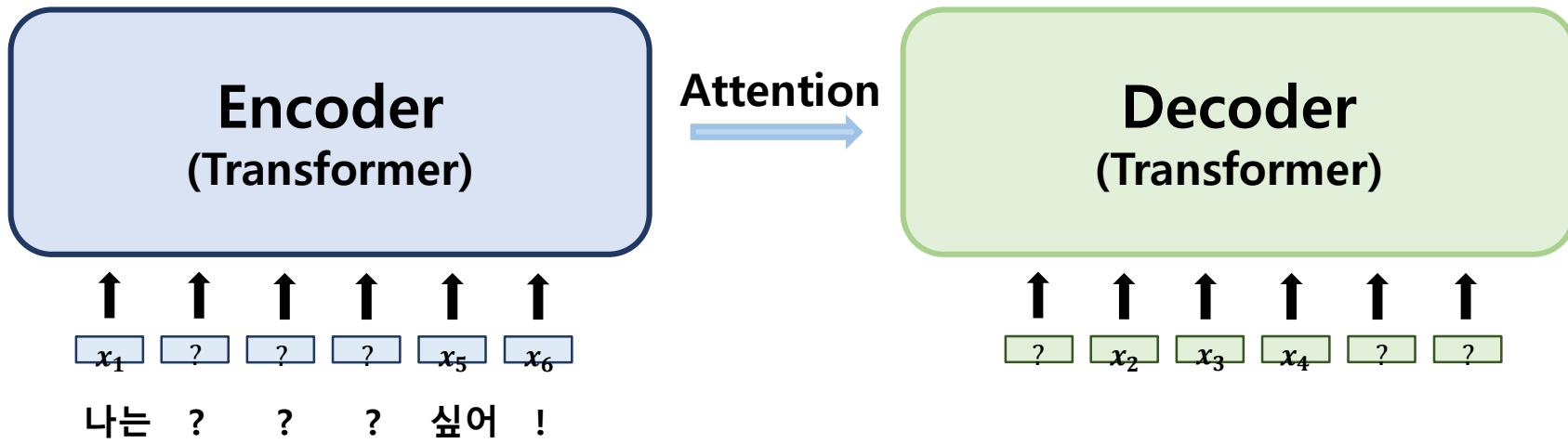


5) 모델 선정

MASS



나는 중국 음식 먹고 싶어 !

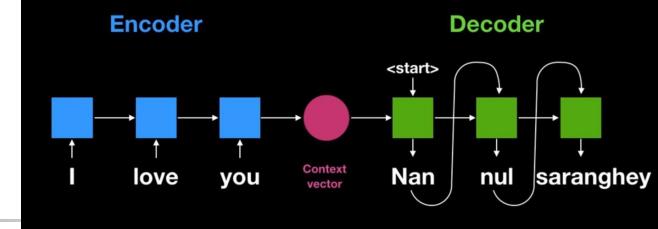


Method	Setting	en - fr	fr - en	en - de	de - en	en - ro	ro - en
Artetxe et al. (2017)	2-layer RNN	15.13	15.56	6.89	10.16	-	-
Lample et al. (2017)	3-layer RNN	15.05	14.31	9.75	13.33	-	-
Yang et al. (2018)	4-layer Transformer	16.97	15.58	10.86	14.62	-	-
Lample et al. (2018)	4-layer Transformer	25.14	24.18	17.16	21.00	21.18	19.44
XLM (Lample & Conneau, 2019)	6-layer Transformer	33.40	33.30	27.00	34.30	33.30	31.80
MASS	6-layer Transformer	37.50	34.90	28.30	35.20	35.20	33.10

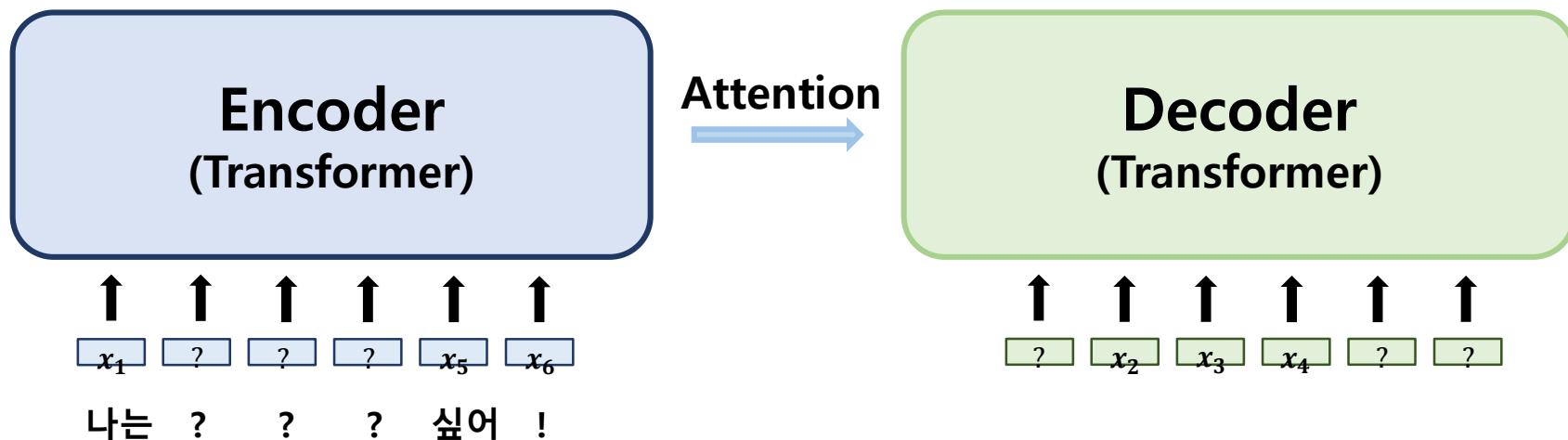
Method	en-fr	fr-en	en-de	de-en	en-ro	ro-en
<i>BERT+LM</i>	33.4	32.3	24.9	32.9	31.7	30.4
<i>DAE</i>	30.1	28.3	20.9	27.5	28.8	27.6
MASS	37.5	34.9	28.3	35.2	35.2	33.1

5) 모델 선정

BART



나는 중국 음식 먹고 싶어 !



A _ C . _ E .
Token Masking

D E . A B C .
Sentence Permutation

C . D E . A B
Document Rotation

A . C . E .
Token Deletion

A B C . D E .

A _ . D _ E .
Text Infilling

6) 모델 입력데이터 형식으로 변환 코드 작성

7) 학습데이터 Pre-generation

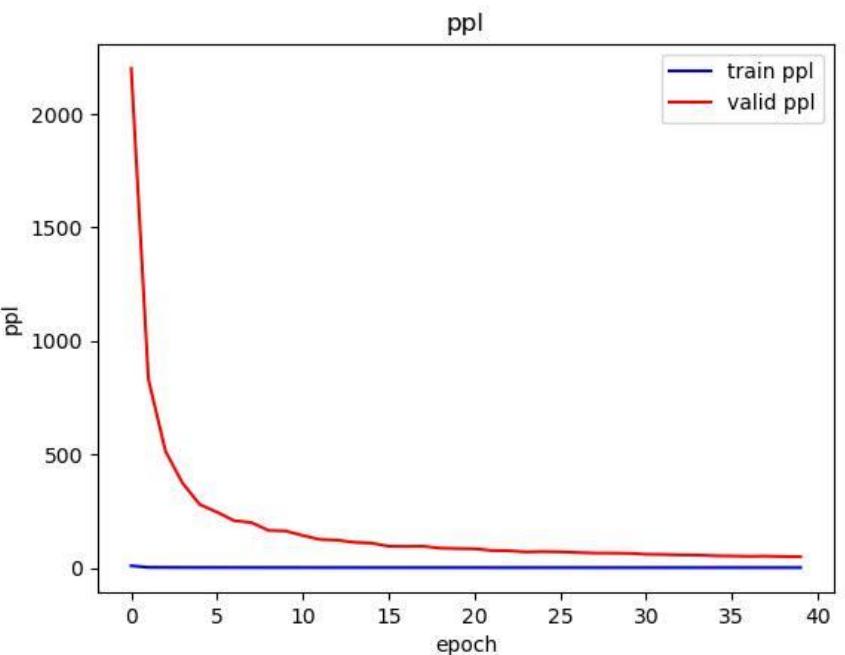
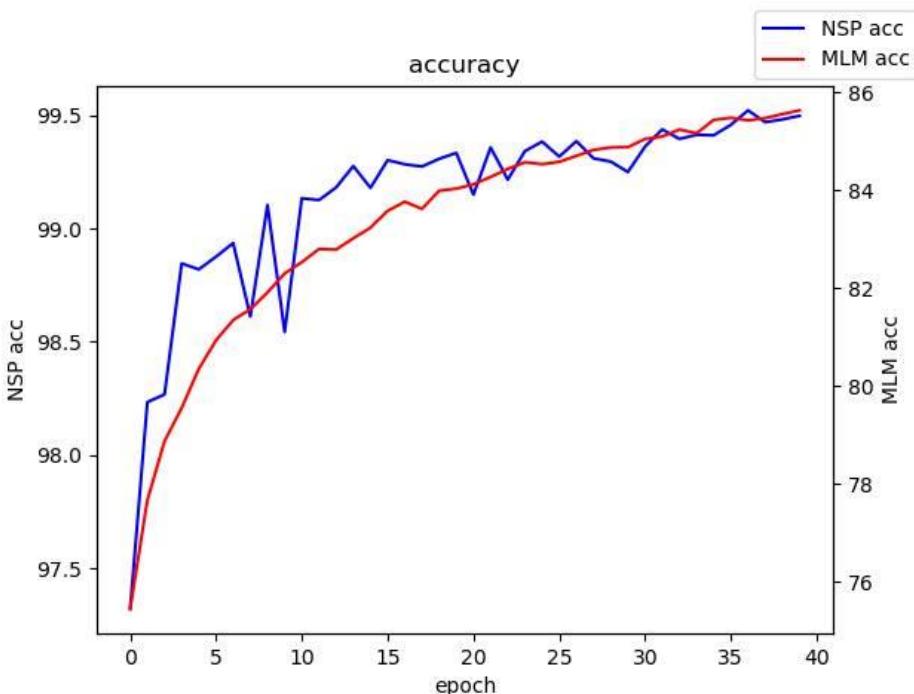
- 목적에 따라 언어모델 선택
- 언어 이해 VS 생성
- 모델은 다 거기서 거기
- 입출력만 다르지 똑같다!

8) 학습 테스트

- 모델 저장 경로
- Interval
- 저장 용량 체크

9) 학습!

- 끊임없이 모니터링
- 용량체크
- 중단체크
- 에러체크



언어모델 활용 I

개체명 인식

Named Entity Recognition(NER)

자연어처리 기술을 이용, 문맥 상 의미를 파악하여 meta data 추출

개체명인식

- 기존 ** 분야에서 텍스트 분석 연구는 자주 사용되는 키워드나 핵심 문장 추출에 불과(word cloud)
 - 하지만 이러한 연구는 현재 어떤 단어가 이슈인지 정도의 트렌드 분석만 가능
 - 원하는 정보는 얻을 수 없음



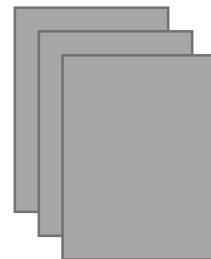
개체명인식

- 수만 개의 다양한 비정형 텍스트 존재
- 직접 읽지 않으면??
- 각 회사의 문서들마다 형식이 다름
- 특정 ** 보고서에서 '중국', '국방부' 이 두 단어를 키워드로 추출
- But, 같은 단어도 문맥에 따라 다른 의미가 될 수 있음
 - 중국 : 공격을 실행한 국가? 당한 국가?
 - 국방부 : 공격을 지시한 기관? 공격을 당한 기관?
- 보고서로부터 문맥을 파악하여 중요한 **meta data**(공격자, 공격 그룹, 공격 방법, 공격 대상 등)를 추출하는 기술이 필요

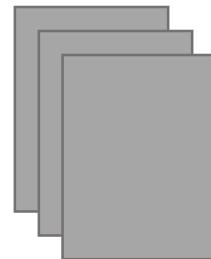
In fact, the Chinese **NORP** market has the **three CARDINAL** most influential names of the retail and tech space – **Alibaba GPE**, **Baidu ORG**, and **Tencent PERSON** (collectively touted as **BAT ORG**), and is betting big in the global **AI GPE** in retail industry space. The **three CARDINAL** giants which are claimed to have a cut-throat competition with the **U.S. GPE** (in terms of resources and capital) are positioning themselves to become the 'future **AI PERSON** platforms'. The trio is also expanding in other **Asian NORP** countries and investing heavily in the **U.S. GPE** based **AI ORG** startups to leverage the power of **AI GPE**. Backed by such powerful initiatives and presence of these conglomerates, the market in APAC AI is forecast to be the fastest-growing **one CARDINAL**, with an anticipated **CAGR PERSON** of **45% PERCENT** over **2018 - 2024 DATE**.

To further elaborate on the geographical trends, **North America LOC** has procured **more than 50% PERCENT** of the global share in **2017 DATE** and has been leading the regional landscape of **AI GPE** in the retail market. The **U.S. GPE** has a significant credit in the regional trends with **over 65% PERCENT** of investments (including M&As, private equity, and venture capital) in artificial intelligence technology. Additionally, the region is a huge hub for startups in tandem with the presence of tech titans, such as **Google ORG**, **IBM ORG**, and **Microsoft ORG**.

개체명인식



구조화된 보고서



구조화 되지 않은 보고서

Meta data 변환



Metadata 항목	
취약점 번호	2018-2628
공격 국가	북한
공격 목적	정보 탈취
공격 그룹	APT37

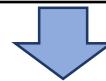
개체명인식

■ 개체명 인식

- 입력데이터의 문맥을 파악하여 각각의 어휘가 어떤 메타데이터에 해당하는지를 추정하는 기능
- 구조화되지 않은 일반 문서가 주어졌을 때, 사람 이름, 장소, 시간 등 미리 정의된 특성에 해당하는 단어 또는 구절을 찾는 기술
- 직접 텍스트를 읽지 않고도 문맥 정보가 고려된 중요 정보를 추출 가능

입력

北	정보탈취	전문	해킹조직	'APT37'	전모	공개
---	------	----	------	---------	----	----



Deep Learning Model



출력

공격 국가	공격 목적	None	None	공격 그룹	None	None
-------	-------	------	------	-------	------	------

언어모델 Final

- 그래서 입출력이 뭐지??

입력

北	정보탈취	전문	해킹조직	'APT37'	전모	공개
---	------	----	------	---------	----	----



언어모델

BERT/GPT/ELECTRA

언어모델

입력 차원 : 문장개수x문장길이

출력 차원 : 문장개수x문장길이x768

$1 \times 7 \rightarrow 1 \times 7 \times 768$

출력

base: 768

large: 1024

0.01
0.2
0.05
0.4
...

0.04
0.6
0.02
0.1
...

0.02
0.5
0.08
0.3
...

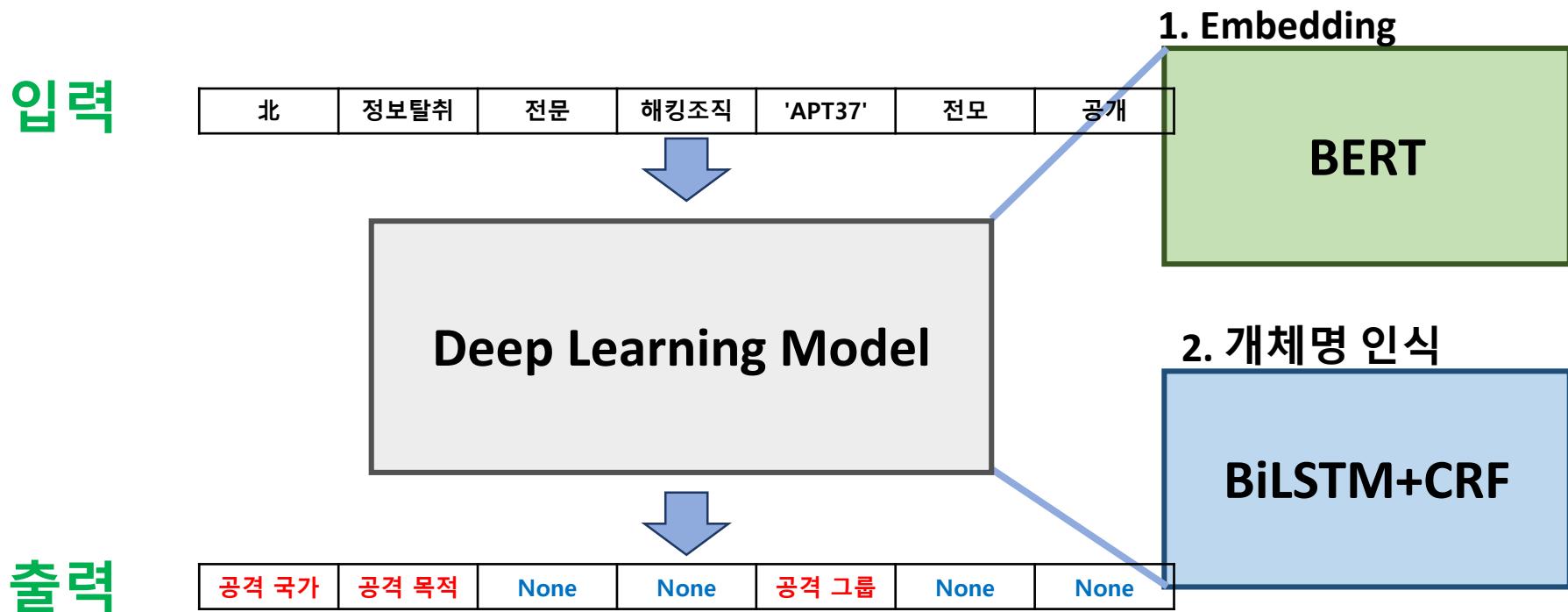
0.13
0.21
0.01
0.32
...

0.53
0.01
0.05
0.2
...

0.1
0.22
0.05
0.34
...

0.07
0.15
0.07
0.25
...

개체명인식



입력

北	정보탈취	전문	해킹조직	'APT37'	전모	공개
---	------	----	------	---------	----	----



언어모델

BERT/GPT/ELECTRA

언어모델

입력 차원 : 문장개수x문장길이

출력 차원 : 문장개수x문장길이x768

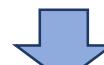
$1 \times 7 \rightarrow 1 \times 7 \times 768$

언어모델 출력

base: 768

large: 1024

0.01	0.04	0.02	0.13	0.53	0.1	0.07
0.2	0.6	0.5	0.21	0.01	0.22	0.15
0.05	0.02	0.08	0.01	0.05	0.05	0.07
0.4	0.1	0.3	0.32	0.2	0.34	0.25
...



개체명 인식

BiLSTM+CRF

개체명인식

입력 차원 : 문장개수x문장길이x768

출력 차원 : 문장개수x문장길이x개체명개수

$1 \times 7 \times 768 \rightarrow 1 \times 7 \times 4$

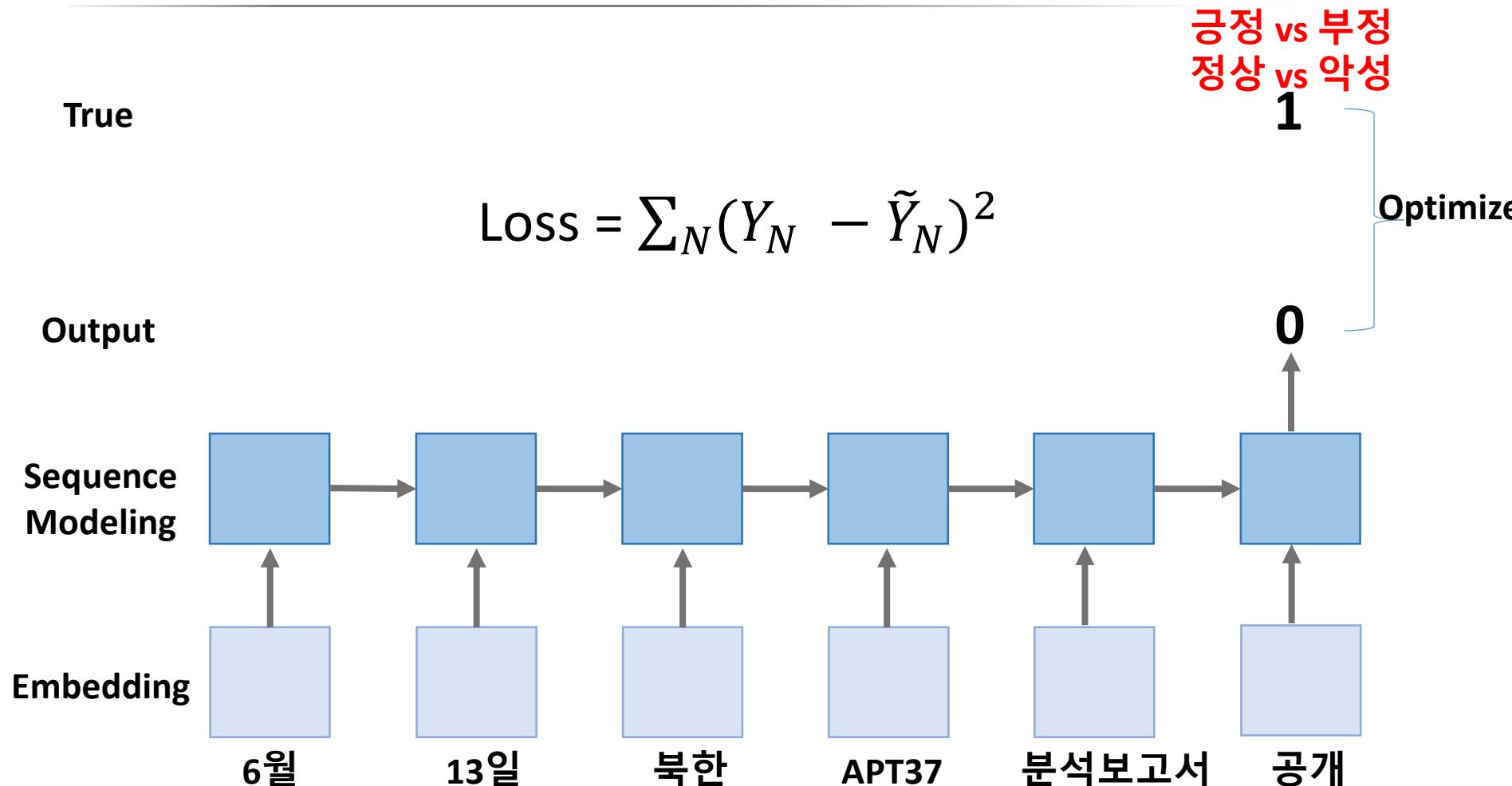
개체명인식 출력

base: 768

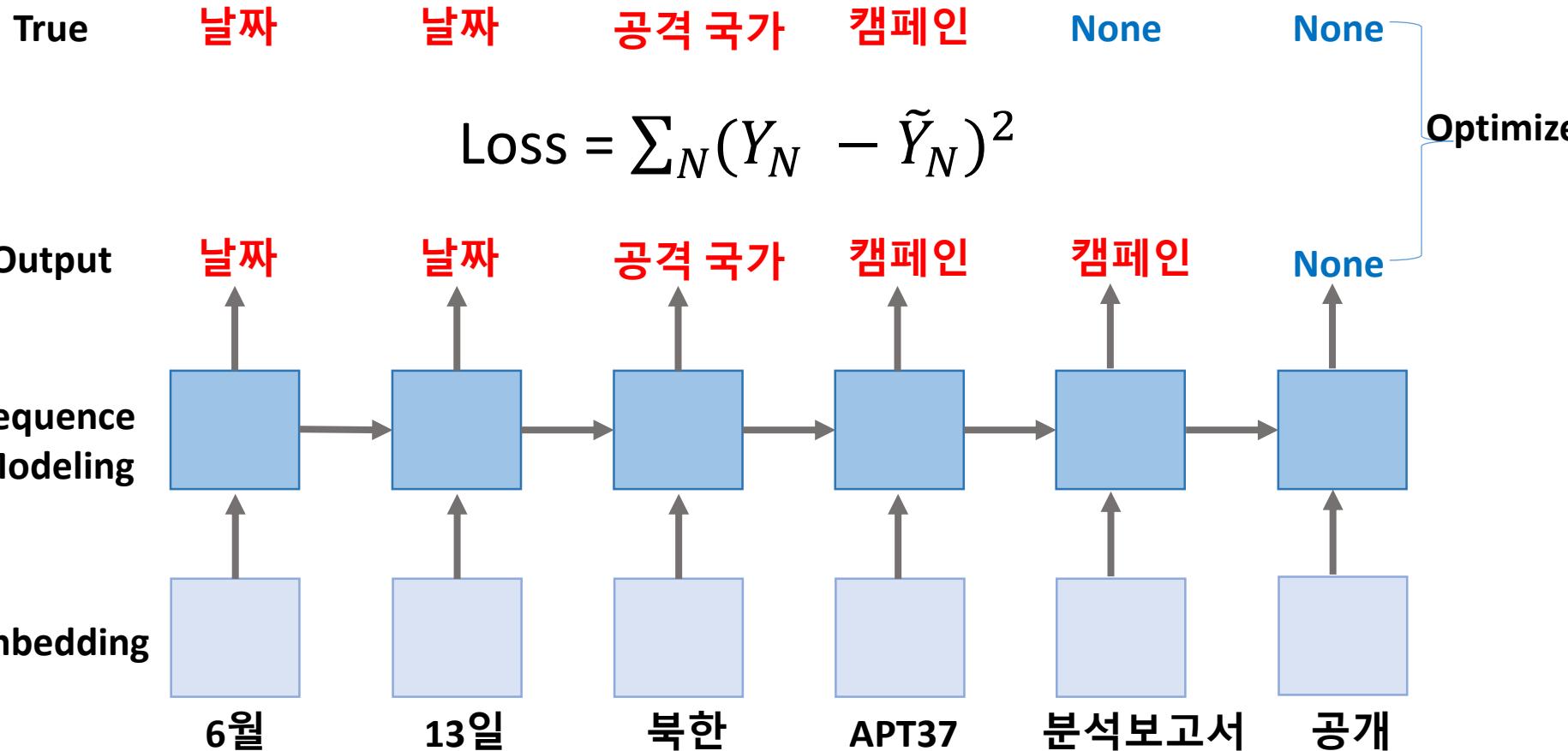
large: 1024

공격 국가	공격 목적	None	None	공격 그룹	None	None
0.01	0.02	0.97	0.97	0.01	0.97	0.97
0.2	0.91	0.02	0.01	0.2	0.12	0.02
0.05	0.03	0.05	0.01	0.91	0.05	0.25
0.74	0.04	0.04	0.01	0.4	0.13	0.04

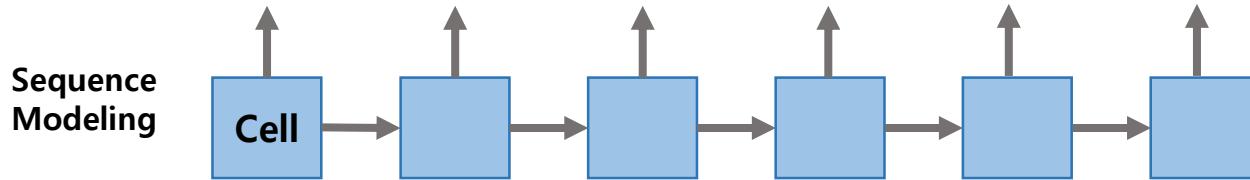
일반적인 NLP 문제(N to 1)



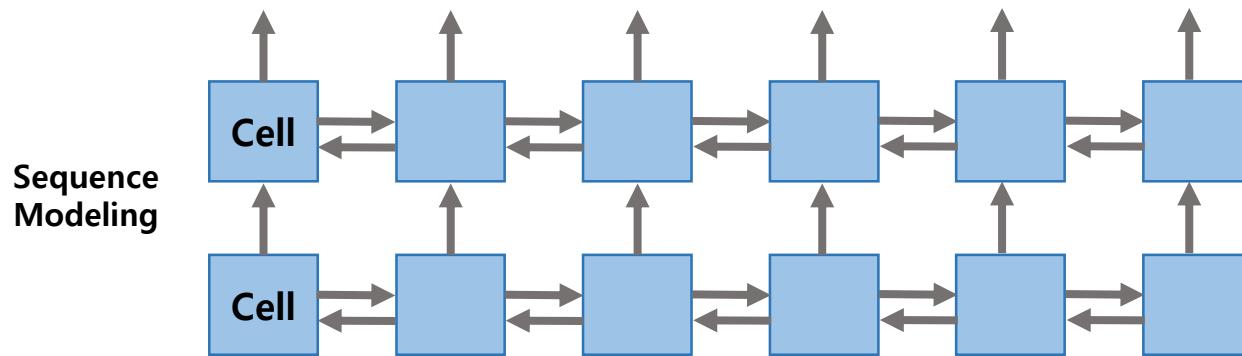
NER기본 구조 – Sequence labeling(N to N)



NER기본 구조 – Sequence labeling(N to N)



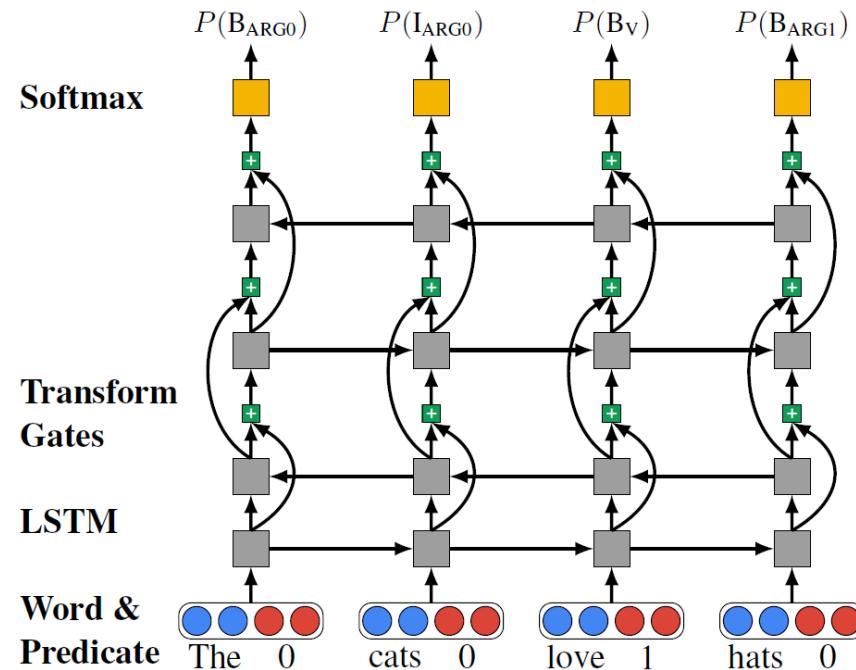
1. Cell 변경 : RNN, LSTM, GRU
2. 층 추가
3. 방향 추가: 단방향, 양방향(Bi-LSTM)



NER기본 구조 – Sequence labeling(N to N)

4. HighWay-LSTM

- He, Luheng, et al. "Deep semantic role labeling: What works and what's next." ACL. 2017.



NER기본 구조 – Sequence labeling(N to N)

NER

수상구분	Rank	Date	팀명	구성원	F1
대상	1	2018.12.14 07:45	State_Of_The_Art	박동주 (광주과학기술원)	90.4219
우수	2	2018.12.14 00:29	cheap_learning	박광현, 이영훈 (전북대학교)	90.2417
우수	3	2018.12.14 22:46	nlp_pln	이신의, 박장원, 박종성 (연세대학교 데이터공학연구실)	89.7830
장려	4	2018.12.14 15:18	Sogang_Alzzam	박찬민, 박영준 (서강대학교 자연어처리연구실)	88.8506
장려	5	2018.12.14 23:51	ner_master	조민수, 박찬희, 박진욱 (연세대학교 데이터공학연구실)	88.5818
장려	6	2018.12.13 19:28	bible	현청천 (HELLO NMS)	88.3348

→ BiLSTM + CRF

→ BiLSTM + CRF

→ Transformer

SRL

수상구분	Rank	Date	팀명	구성원	F1
대상	1	2018.12.14 22:01	Sogang_Alzzam	박찬민, 박영준 (서강대학교 자연어처리연구실)	77.6628
우수	2	2018.12.14 17:44	KANE_team	함영균, 김동환, 최기선 (KAIST SWRC)	76.3328
우수	3	2018.12.10 19:53	cheap_learning	박광현, 이영훈 (전북대학교)	76.2543
장려	4	2018.12.08 22:06	OnlyOne	김영천	75.4308
장려	5	2018.12.14 17:43	nlp_pln	이신의, 박장원, 박종성 (연세대학교 데이터공학연구실)	74.8749
장려	6	2018.12.14 09:17	kozistr_team	김형찬 (한국기술교육대학교)	74.7695

→ Elmo + BiLSTM + CRF

→ BiLSTM + CRF

모델 구조 : BiLSTM + CRF를 공통적으로 사용

NER ranking

Model	F1	Paper / Source
Flair embeddings (Akbik et al., 2018)♦	93.09	Contextual String Embeddings for Sequence Labeling
BERT Large (Devlin et al., 2018)	92.8	BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding
CVT + Multi-Task (Clark et al., 2018)	92.61	Semi-Supervised Sequence Modeling with Cross-View Training
BERT Base (Devlin et al., 2018)	92.4	BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding
BiLSTM-CRF+ELMo (Peters et al., 2018)	92.22	Deep contextualized word representations
Peters et al. (2017) ♦	91.93	Semi-supervised sequence tagging with bidirectional language models
CRF + AutoEncoder (Wu et al., 2018)	91.87	Evaluating the Utility of Hand-crafted Features in Sequence Labelling
Bi-LSTM-CRF + Lexical Features (Ghaddar and Langlais 2018)	91.73	Robust Lexical Features for Improved Neural Network Named-Entity Recognition
Chiu and Nichols (2016) ♦	91.62	Named entity recognition with bidirectional LSTM-CNNs
HSCRF (Ye and Ling, 2018)	91.38	Hybrid semi-Markov CRF for Neural Sequence Labeling
IXA pipes (Agerri and Rigau 2016)	91.36	Robust multilingual Named Entity Recognition with shallow semi-supervised features
NCRF++ (Yang and Zhang, 2018)	91.35	NCRF++: An Open-source Neural Sequence Labeling Toolkit
LM-LSTM-CRF (Liu et al., 2018)	91.24	Empowering Character-aware Sequence Labeling with Task-Aware Neural Language Model
Yang et al. (2017) ♦	91.26	Transfer Learning for Sequence Tagging with Hierarchical Recurrent Networks
Ma and Hovy (2016)	91.21	End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF
LSTM-CRF (Lample et al., 2016)	90.94	Neural Architectures for Named Entity Recognition

모델 구조 : BiLSTM + CRF를 공통적으로 사용

차이는???

=> Embedding!!!

NER 데이터셋 구축



1) 문서 PDF/HTML

September 07, 2016 07:43:00 AM~

U.S. Investigating Potential Covert Russian Plan to Disrupt November Elections.

According to intelligence and Congressional officials, U.S. law enforcement and intelligence agencies are investigating what they think is a broad Russian operation to instill public distrust prior to the November election.~

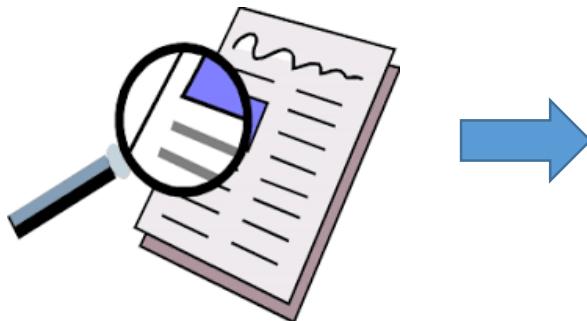
The Russian campaign allegedly uses cyber tools to infiltrate systems to spread false information.~

ANALYST COMMENT : Throughout the past three months, FireEye iSIGHT Intelligence has reported on cyber espionage campaigns conducted by Tsar Team(APT28) as well as TEMP.Monkey(APT29) against political interests related to the upcoming U.S. elections.~

Historically, Tsar Team campaign targets indicate specific interest in the collection of information intended to provide political and military advantage to Russian interests.~

Additionally, disinformation operations have often been a preferred method for Russian-based cyber espionage actors.~

2) 텍스트 추출 및 정제



[@September 07, 2016 07:43:00 AM#Time_Published*]

[@U.S.#Country*] Investigating Potential Covert Russian Plan to Disrupt [@November Elections#Victim_Target*]

According to intelligence and Congressional officials, [\$U.S.#Country*] law enforcement and intelligence agencies are investigating what they think is a broad Russian operation to instill public distrust prior to the November election.~

The Russian campaign allegedly uses cyber tools to infiltrate systems to [@spread false information#Attack_Objective*].~

ANALYST COMMENT : Throughout the past three months, [@FireEye#Company*] iSIGHT Intelligence has reported on cyber espionage campaigns conducted by [@Tsar Team#Threat_Actor*]([@APT28#Threat_Actor*]) as well as [@TEMPMonkey#Threat_Actor*]([@APT29#Threat_Actor*]) against political interests related to the upcoming [@U.S.#Country*] [@elections#Target_Industry*].~

Historically, [@Tsar Team#Threat_Actor*] campaign targets indicate specific interest in [@the collection of information intended to provide political and military advantage to Russian interests#Attack_Objective*].~

Additionally, disinformation operations have often been a preferred method for Russian-based cyber espionage actors.~

3) Annotation 작업

Metadata 항목	
취약점 번호	2018-2628
공격 국가	북한
공격 목적	정보 탈취
공격 그룹	APT37

NER 데이터셋 구축

- NER 데이터 태깅을 위해 YEDDA 사용

- 태깅 시 [@개체명 단어#metadata*]로 저장

ex) [@US#Location*]

- 문제점

- 태깅시 '#'이 들어간 단어에 태깅이 된다면??
 - 한 문구에 여러개의 태깅이 되야한다면??
 - 태깅 숫자가 너무 많으면?? => 단축기가 모자람
 - Labeling 수행자들 간 통일성 문제....심각
 - 수행자의 일관성 문제

=> 어느 날은 공격방법, 어느 날은 공격도구로 태깅, 성능저하....

- CONLL2003 포맷으로 변환

word	POS	Syntactic chunk	tag	named entity
U.N.	NNP	I-NP		I-ORG
official	NN	I-NP		O
Ekeus	NNP	I-NP		I-PER
heads	VBZ	I-VP		O
for	IN	I-PP		O
Baghdad	NNP	I-NP		I-LOC
.	.	O		O

동의어 사전 구축

- 같은 단어가 다양하게 사용되는 경우가 너무 많음

- APT37
- Scarcruft
- Group123
- APT38
- Lazarus
- Bluenoroff
- NK
- North Korea
- N-Korea

동의어 사전 구축

- 같은 단어가 다양하게 사용되는 경우가 너무 많음

- Wcry
- WanaCrypt0r
- WanaCrypt0r 2.0
- WanaCryptor
- WannaCry
- WannaCry 2.0
- WannaCryptor
- Wannacrypt

개체명인식 개발 로드맵

자연어처리 기술을 이용, 문맥 상 의미를 파악하여 meta data 추출

NER 개발 로드맵

- 1) 데이터 정제
- 2) 데이터 Labeling
- 3) Labeling 데이터 체크
- 4) 학습형태로 변환
- 5) 모델 선정
- 6) 학습
- 7) 학습결과 확인
- 8) 추론 코드 작성
- 9) 추론서비스 테스트
- 9) 서비스

1) 데이터 정제

- 보유한 텍스트에서 불필요한 metatext 제거

2) 데이터 Labeling

3) Labeling 데이터 체크

▪ Label 결정

- 개수에 따라 성능이 달라짐
- 중간에 바뀌면 안됨

▪ Labeling tool 결정

- Doccano 추천!
- <https://github.com/doccano/doccano>

▪ 가이드라인 작성

▪ 크로스체킹

▪ 제일 중요한 작업!!

Barack Hussein Obama II (born August 4, 1961) is an American attorney and politician who served as the 44th President of the United States from January 20, 2009, to January 20, 2017. A member of the Democratic Party, he was the first African American to serve as president. He was previously a United States Senator from Illinois and a member of the Illinois State Senate.

Person p Loc l Org o Event e Date d Other z

1/6

Prev Next

4) 학습형태로 변환

- **NER format**

- CoNLL2003 format이 많이 이용
- POS tagging이 필요

- **NER 학습을 위해 단어단위 분류형태로 변환**

- Doccano format -> CoNLL format

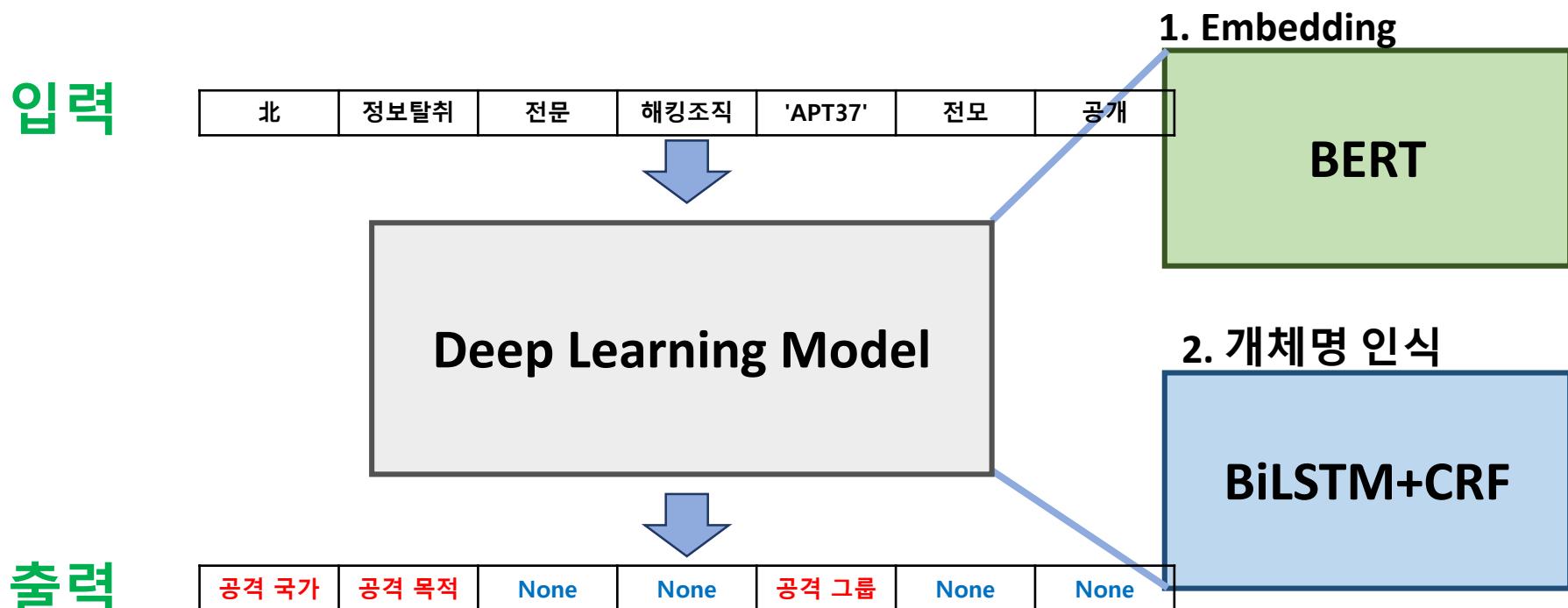
U.N.	NNP	I-NP	I-ORG
official	NN	I-NP	O
Ekeus	NNP	I-NP	I-PER
heads	VBZ	I-VP	O
for	IN	I-PP	O
Baghdad	NNP	I-NP	I-LOC
.	.	O	O

5) 모델 선정

6) 학습

7) 학습결과 확인

- Token classification model 선정
- 단어단위 -> 토큰단위 변환
 - 단어단위로 달려있는 label을 token 단위로 잘개 쪼개주는 과정에서 많은 난관...



- 8) 추론 코드 작성
- 9) 추론서비스 테스트
- 9) 서비스

- 텍스트 입력 -> 정제 -> NER 수행 -> 단어단위로 변환 -> HTML 형태로 출력

1/6

Person p Loc l Org o Event e Date d Other z

Barack Hussein Obama II (born August 4, 1961) is an American attorney and politician who served as the 44th President of the United States from January 20, 2009, to January 20, 2017. A member of the Democratic Party, he was the first African American to serve as president. He was previously a United States Senator from Illinois and a member of the Illinois State Senate.

◀ Prev

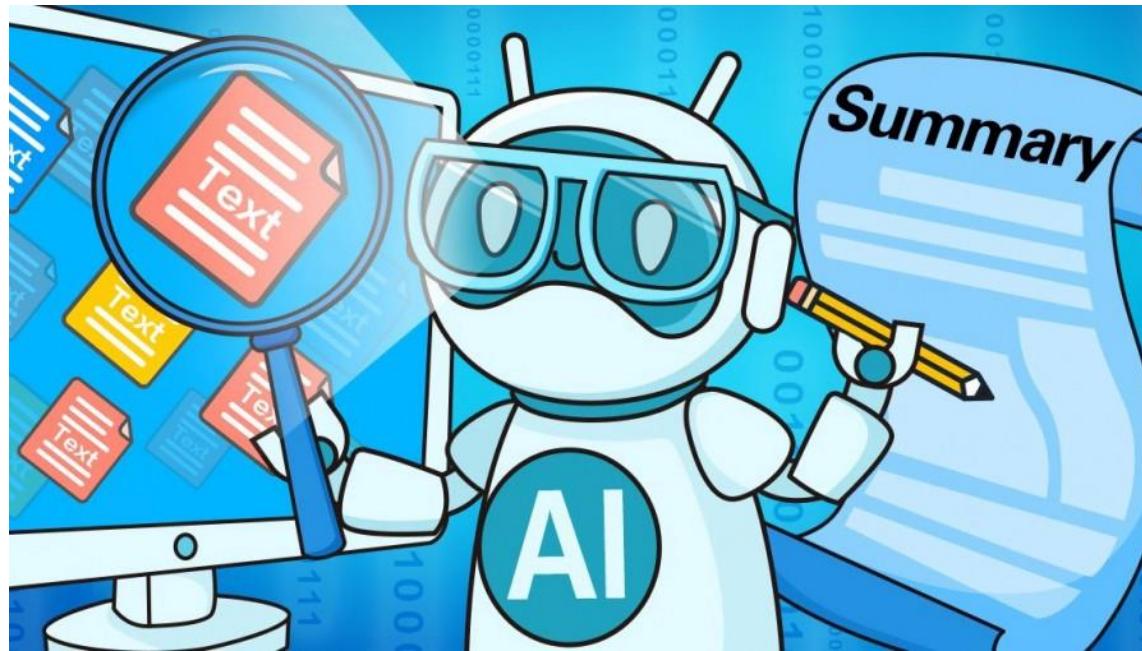
Next ▶

언어모델 활용 Ⅱ

자동 문서 요약

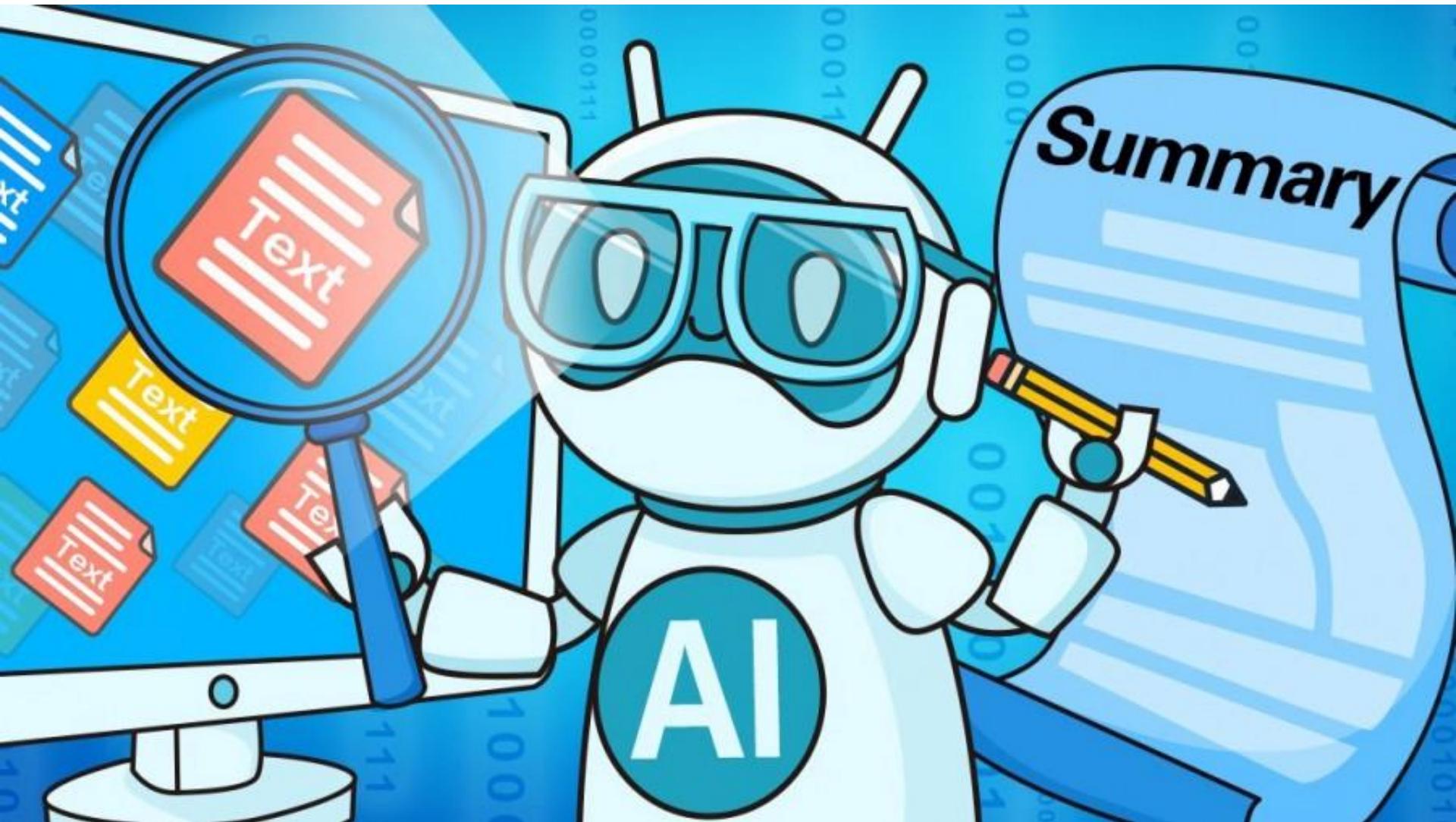
Abstract Summarization

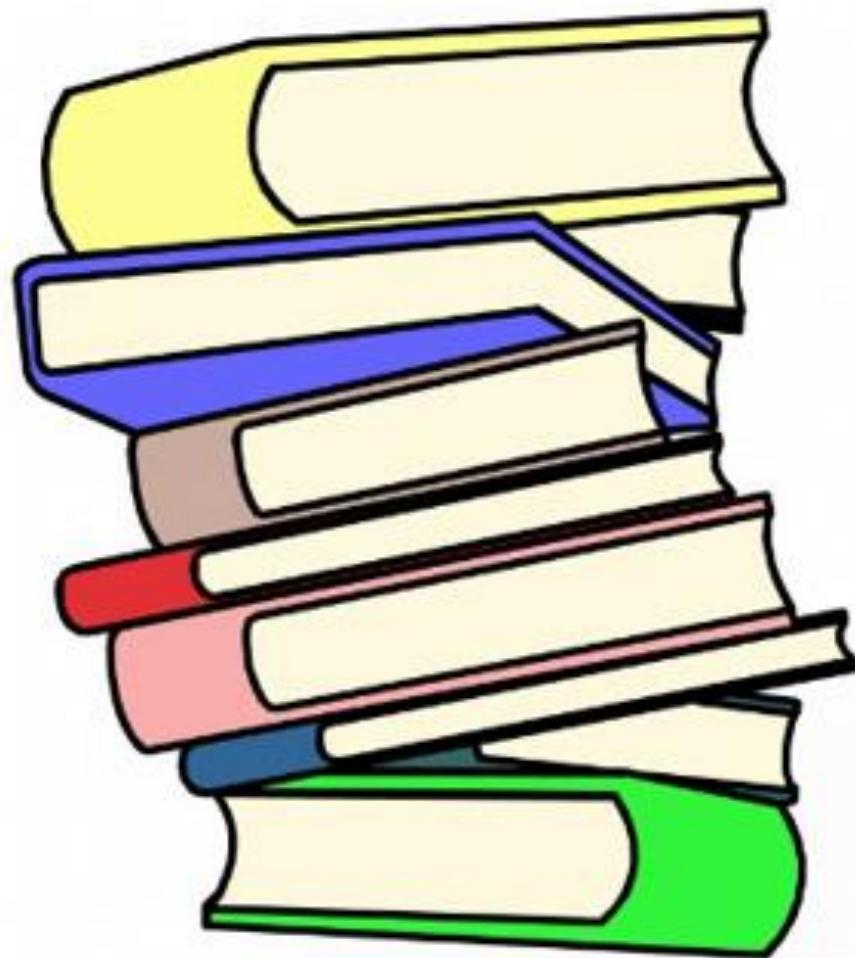
** 분야 전문 **요약** 모델



I just need
the main ideas







Summary



문서 요약

- 중복 뉴스 제거
 - 문장 유사도 비교 기술
- 문서 요약 기술

중복 뉴스 제거 : 문장 유사도 비교 기술

▪ 중복 뉴스 제거 이슈

- 네이버 뉴스에서 관련 뉴스를 둑어주는 기능 존재. 하지만 빠지는게 많음



【이슈】 인공지능과 클라우드로 최초의 무관중 'US 오픈' 위한 새로운 패 경험...

인공지능신문 | 1일 전 |

새로운 솔루션 중 두 가지는 멀티 클라우드에서 워크로드를 실행하면서 다양한 데이터 세트를 가져다 쓰는 IBM 맞손의 자연어 처리(Natural Language Processing, NLP) 기술을 기반으로 한다. IBM의 자연어 처리 기술은 차트...

↳ IBM, 무관중 'US 오픈'에 AI 및 하이… CIO Korea | 1일 전

↳ 무관중 US오픈, 클라우드 기반 AI 가… 헬로티 | 1일 전

↳ IBM, 최초의 무관중 'US 오픈' 위해 AI… 디지털조선일보 | 1일 전

관련뉴스 4건 전체보기 >



신세계아이앤씨, 포티투마루와 자연어 처리·기계독해 AI 공동 개발

디지털투데이 | 2020.08.21. |

손정현 신세계아이앤씨(신세계&C) IT사업부 전무는 "리테일 산업에서 AI기술의 활용이 전방위적으로 확대되면서, 사용자의 니즈는 다양해지고, 처리해야 하는 데이터 역시 방대해지고 있다"며 "자연어처리...

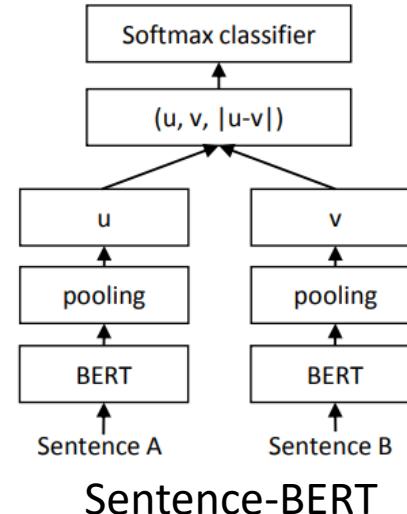
↳ 신세계아이앤씨, 포티투마루와 AI 언… ZDNet Korea | 2020.08.21. | 네이버뉴스

↳ 신세계&C, AI 스타트업 '포티투마루'… 이데일리 | 2020.08.21. | 네이버뉴스

↳ 신세계아이앤씨, 포티투마루, AI 기반… 데이비넷 | 2020.08.21.

↳ 신세계아이앤씨, 고객응대·사내업무 … 아주경제 | 2020.08.21.

관련뉴스 11건 전체보기 >



- 검색된 뉴스에서 문장 유사도 비교 기술을 통해 둑어주자
- 근데 중복 뉴스 라벨링은 어떻게 달지?
- 유사도 비교 기술 알고리즘은? Sentence-BERT? 속도는 어쩌지??
- 성능측정은?
 - ROUGE(N-gram 일치도) 수치 조합을 통한 문장 유사도 비교 기술
- 제약 : 연산량이 $O(n^2)$ 기 때문에 속도가 빠르면서 중복을 잘 제거해야 함
- 유사도 = $W1*ROUGE1 + W2*ROUGE2 + W3*ROUGE3 + W4*ROUGE$ (only 명사인 경우)
- 유사도 > threshold 일 때 중복 뉴스라고 판단하는 최적의 threshold 구함

중복 뉴스 제거 : 문장 유사도 비교 기술

유사도 = $W1*ROUGE1 + W2*ROUGE2 + W3*ROUGE3 + W4*ROUGE$ (only 명사)

▪ 지니언스, 하이트진로에 EDR 솔루션 공급

- ==> 지니언스, 하이트진로에 EDR 솔루션 공급
- ==> 지니언스, 하이트진로에 EDR 솔루션 공급.. 악성코드.이상행위 관리
- ==> 지니언스, 하이트진로에 EDR 솔루션 공급."악성코드 관리"
- ==> 지니언스, 하이트진로에 EDR 솔루션 공급.악성코드 등 통합관리
- ==> 지니언스, 하이트진로에 EDR 공급."악성코드.이상행위 관리"

▪ 미, 북 사이버위협에 주의보 발령."국제금융망에 상당한 위협"

- ==> 미국 "북한 사이버활동, 국제금융시스템에 중대 위협"
- ==> 미, 북 사이버 위협에 주의보 발령."국제금융망에 상당한 위협"
- ==> 미국, 북한 사이버위협 주의보 발령."국제 금융시스템에 위협"
- ==> 미국, 북한 사이버 위협에 주의보."국제금융시스템에 중대한 위협"
- ==> 미국, '북한 사이버해킹 주의보' 발령."국제금융시스템에 큰 위협"
- ==> 미국 '북한 사이버위협' 주의보 내려. "국제금융망에 위협"

▪ 송파구, 사이버보안 전문가 양성

- ==> 송파구 "사이버보안 전문가 되세요"
- ==> 서울 송파구, 사이버보안 무료 교육과정 수강생 20명 모집
- ==> 사이버보안 직종 전문가 양성 위해 송파구·KISIA 뭉쳤다

중복 뉴스 제거 : 문장 유사도 비교 기술

유사도 = $W1*ROUGE1 + W2*ROUGE2 + W3*ROUGE3 + W4*ROUGE$ (only 명사)

▪ 지니언스, 하이트진로에 EDR 솔루션 공급

- ==> 지니언스, 하이트진로에 EDR 솔루션 공급
- ==> 지니언스, 하이트진로에 EDR 솔루션 공급.. 악성코드.이상행위 관리
- ==> 지니언스, 하이트진로에 EDR 솔루션 공급."악성코드 관리"
- ==> 지니언스, 하이트진로에 EDR 솔루션 공급.악성코드 등 통합관리
- ==> 지니언스, 하이트진로에 EDR 공급."악성코드.이상행위 관리"

▪ 미, 북 사이버위협에 주의보 발령."국제금융망에 상당한 위협"

- ==> 미국 "북한 사이버활동, 국제금융시스템에 중대 위협"
- ==> 미, 북 사이버 위협에 주의보 발령."국제금융망에 상당한 위협"
- ==> 미국, 북한 사이버위협 주의보 발령."국제 금융시스템에 위협"
- ==> 미국, 북한 사이버 위협에 주의보."국제금융시스템에 중대한 위협"
- ==> 미국, '북한 사이버해킹 주의보' 발령."국제금융시스템에 큰 위협"
- ==> 미국 '북한 사이버위협' 주의보 내려. "국제금융망에 위협"

겹치는 명사 개수 비교만으로 해결!!

▪ 송파구,사이버보안 전문가 양성

- ==> 송파구 "사이버보안 전문가 되세요"
- ==> 서울 송파구, 사이버보안 무료 교육과정 수강생 20명 모집
- ==> 사이버보안 직종 전문가 양성 위해 송파구·KISIA 뭉쳤다

딥러닝이 항상 해답은 아님

문서 요약 데이터셋

■ ** 요약모델 데이터셋 구축

- 2010~2020.4 5,551개 *** 일일동향 파싱
- 2만개 본문-요약 데이터셋 구축
- 보안뉴스(3줄요약) 1,697개
- 총 2,2000개 뉴스로 학습

<<doc>>_180105_1##0_1. 도메인관리 1위업체 가비아, 디도스로 한때 서버 다운

오후 4시19분부터 약 40분간 접속 장애...중소 소핑몰 피해
국내 도메인 등록 접유율 1위 업체인 가비아[079940](www.gabia.co.kr)의 도메인네임서버(DNS)가 4일 오후 디도스(DDoS·부산서비스거부) 공격을 받아 다운됐다.

가비아에 따르면 네임서버를 노린 디도스 공격으로 이날 오후 4시19분부터 오후 5시까지 접속 장애가 발생했다.

이 때문에 이 업체를 통해 도메인네임을 관리하던 많은 기업의 웹사이트가 제대로 열리지 않아 인터넷 이용자들과 기업들이

불편을 겪었다.

대부분 중소쇼핑몰을 업체인 것으로 알려졌다.

DNS는 숫자로 된 IP주소(예를 들어 211.234.118.50)를 문자로 된 도메인네임(예를 들어 ns.gabia.co.kr)과 연결해 주는

역할을 한다.

만약 어떤 웹사이트가 이용하는 DNS의 작동에 문제가 생기면 이용자가 인터넷 주소창에 주소(도메인명)를 입력하더라도

DNS가 이를 연결해 주지 못해 웹사이트에 접속되지 않는다.

이날 가비아 홈페이지 자체도 접속되지 않았으며, 소핑몰 등 가비아 고객들은 로그인하지 못해 서비스를 제대로 이용하지

못했다.

가비아는 홈페이지 공지를 통해 "디도스 공격으로 사이트 접속이 안 되는 장애가 발생했지만 정상 복구됐다"며 "서비스

운영에 불편을 끼쳐 사과드린다"고 밝혔다.

한국인터넷진흥원(KISA)은 공격에 동원된 IP주소를 추적하는 등 조사에 나섰다.

1998년 설립된 가비아는 국내 도메인 등록 접유율 1위 업체이며, 2005년에 업계 최초로 코스닥에 상장했다.

@highlight

국내 도메인 등록 접유율 1위 업체인 가비아의 도메인네임서버(DNS)가 4일 오후 디도스(DDoS·부산서비스거부) 공격을
받아 다운돼

@highlight

가비아에 따르면 네임서버를 노린 디도스 공격으로 이날 오후 4시19분부터 오후 5시까지 접속 장애가 발생했으며,
한국인터넷진흥원(KISA)은 공격에 동원된 IP주소를 추적하는 등 조사에 나서

APT 단체들, 옛 기법과 신기술 혼합해 방어선 흐트러트려

좋아요 7개 | 입력: 2020-11-04 12:22



#정보보호 #정보보안 #IT보안 #사이버보안

3사분기 동안 여러 가지 행동 패턴 보여준 APT 단체들...활동량 늘어나

옛 기법 고수하는 단체도 있고 신기술 적용하는 단체도 있고...방어 혼란 가중

[보안뉴스 문가용 기자] APT 그룹들이 지정학과 외교 상황이라는 안개 속에 숨어서 자신들의 목적을 꾸준히 달성하고 있다. 그런 와중에 다양한 전략과 기법을 선보이며 수사와 추적에 혼란을 주고 있기도 하다. 이런 APT 단체들이 3사분기 동안 보여준 행동 패턴들을 보안 업체 카스퍼스키(Kaspersky)가 정리해 발표했다.

3줄 요약

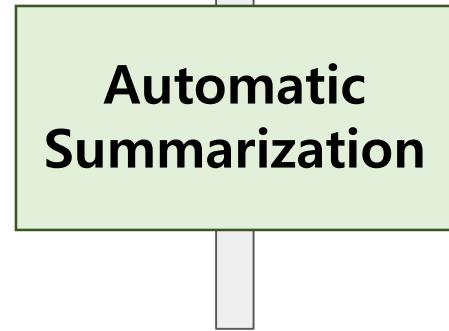
1. 신기술 도입하기 시작한 APT 단체들, 방어에 혼란 가져옴.
2. 특히 감염 방식, C&C 연결 방식, 정상적인 서비스 활용 등에서 혁신이 눈에 띌.
3. 과거에 괜찮았던 서비스나 컴퓨팅 요소들도 이제 다시 확인해야 함.

[국제부 문가용 기자(globoan@boannews.com)]

문서 요약 기술

▪ Source text x (문서)가 주어졌을 때 target text y (요약문)을 생성

- Target text y 는 짧아야 하며, source text x 의 중요한 정보를 포함해야 함
- Source text x 는 한개의 문서 or 여러 개의 문서일 수 있음
- 요약 평가는 요약 문장 y 와 생성된 문장 \hat{y} 간 N-gram 일치도(ROUGE, 겹치는 단어 혹은 구의 수)로 판단



Document (News): “또한, 개인정보가 유출된 것으로 추정되는 고객에게는 이메일과 문자 메시지로 개인정보 유출사고에 관해 안내했다고 설명하면서, 안내를 받은 고객들은 회원계정의 안전한 보호를 위해 가급적 비밀번호를 변경해 달라고 당부했다.”

문서 요약 기술

■ Source text x (문서)가 주어졌을 때 target text y (요약문)을 작성

- Target text y 는 짧아야 하며, source text x 의 중요한 정보를 포함해야 함
- Source text x 는 한개의 문서이거나 여러 개의 문서일 수 있음
- 요약 평가는 요약 문장 y 와 생성된 문장 \hat{y} 간 N-gram 일치도(ROUGE, 겹치는 단어 혹은 구의 수)로 판단

■ 추출 요약(Extractive Summarization)



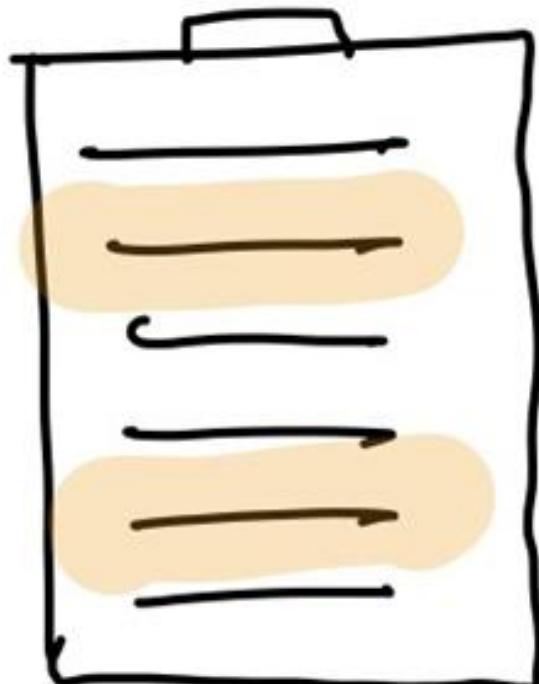
- Source text x 에서 중요한 문장을 선택
- 쉽지만 source 내에 존재하는 문장만 사용해야 하므로 제한적, 부자연스러움
- 그래프 랭킹 알고리즘 주로 사용(ex. PageRank)

■ 생성 요약(Abstractive Summarization)

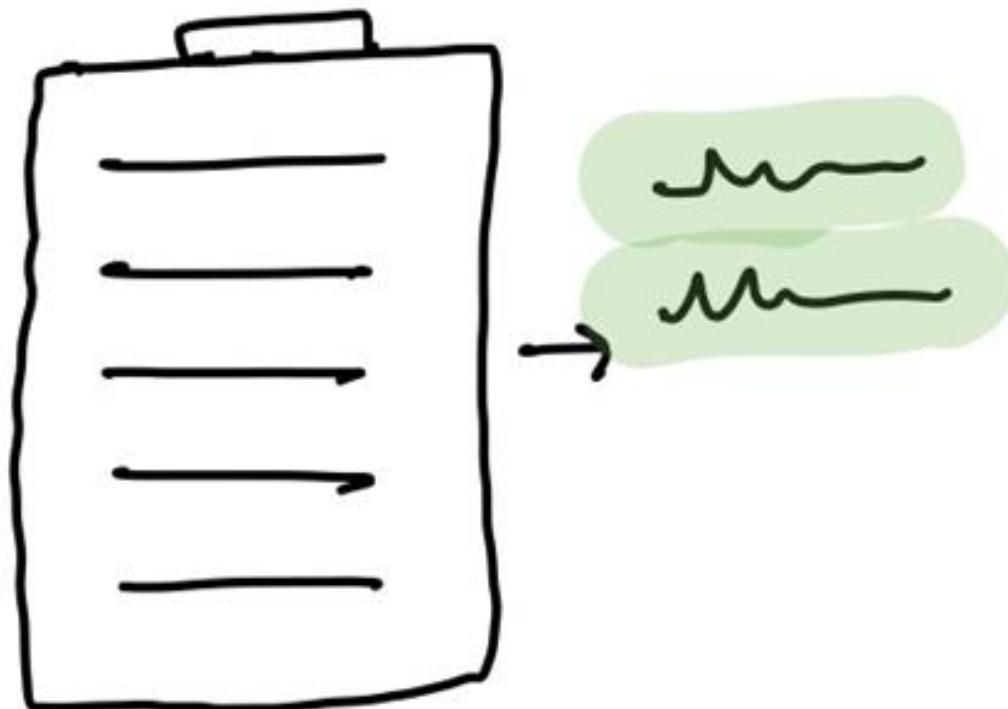


- NLG를 이용해 새로운 문장을 생성
- NLG는 앞 문맥을 보고 모든 생성 가능한 단어(3,4000개) 중 하나씩 순서대로 생성
- 어렵지만 사람이 직접 요약을 하는 것과 유사(e.g. 사람이 문서를 읽고 남에게 짧게 설명함)

문서 요약 기술



Extractive



Abstractive

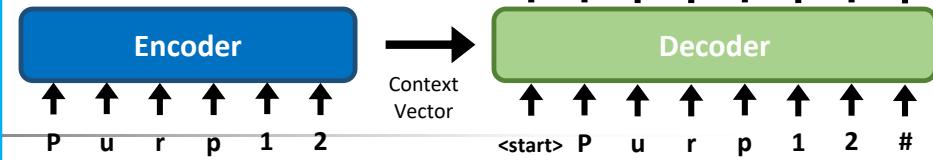
문서 요약 기술

- 어떤 모델을 쓸까??

- Seq2Seq 모델이니 Decoder까지 학습하는 BART, Pegasus가 SOTA
- But, 잘 살펴보니 BERT-large 모델 기준
- large 모델은 너무 커서 학습/추론이 무리
- 동일한 크기의 bert-base 모델 기준으로 비교해서 Presumm으로 결정

문서 요약 기술

Seq2Seq model outline



▪ Presumm : BERT + Transformer

- BERT를 이용하여 뉴스 본문 임베딩(벡터화)
- Transformer를 이용하여 요약문 생성
- BUT!! 파라미터 숫자가 엄청 많아 느리다!!

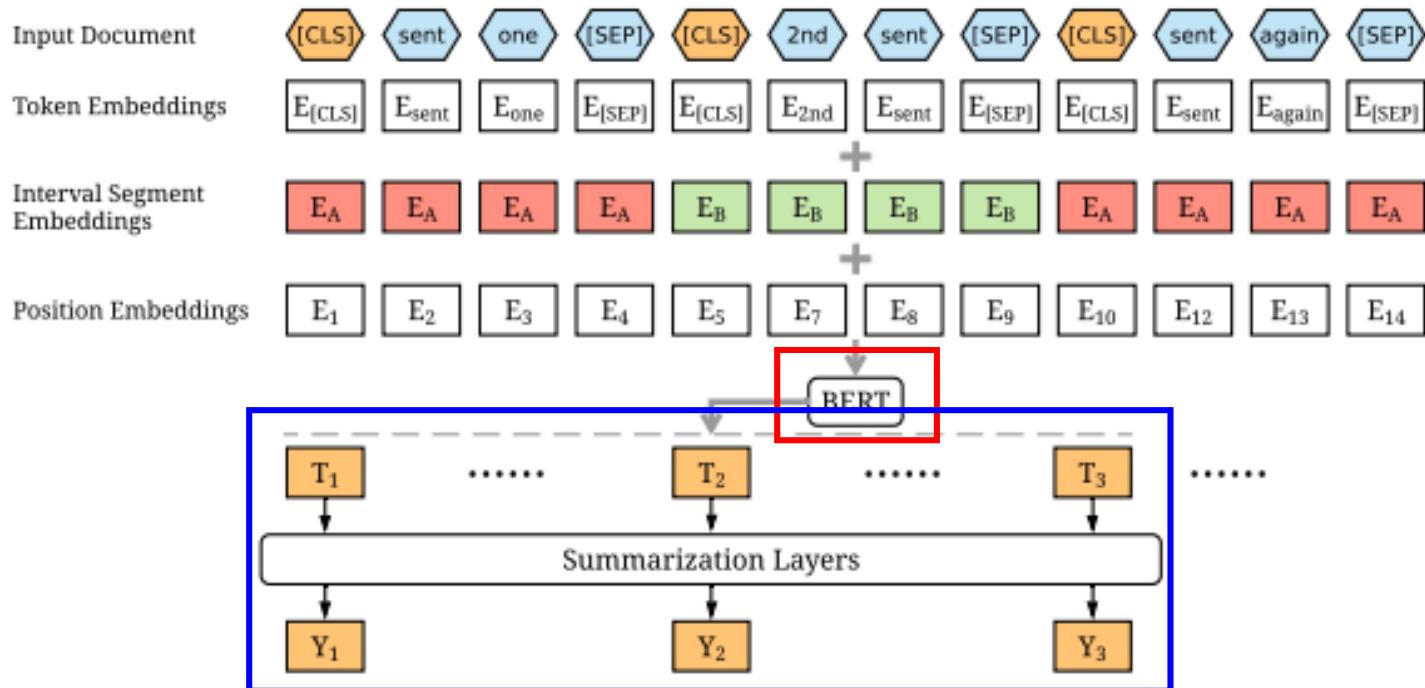
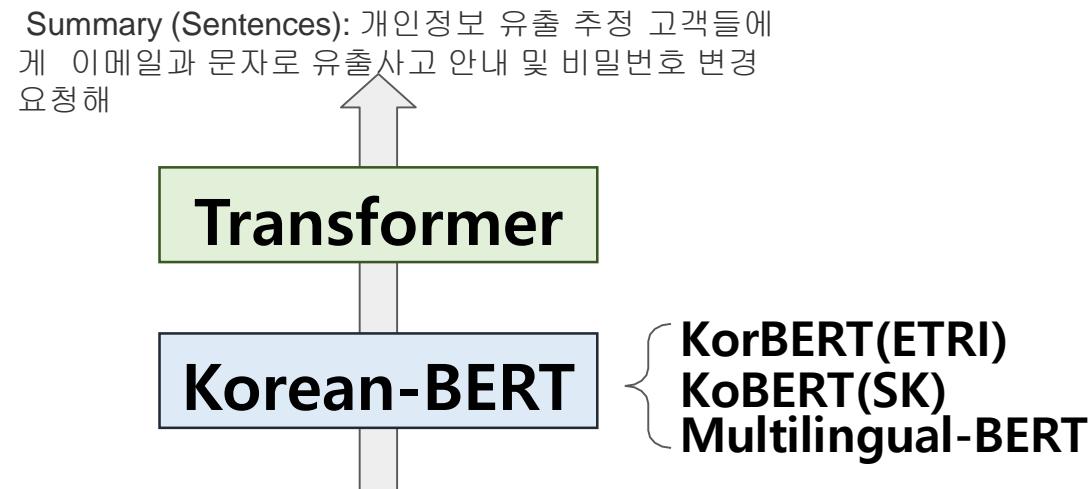


Figure 1: The overview architecture of the BERTSUM model.

문서 요약 기술

▪ Presumm : BERT + Transformer

- BERT를 이용하여 뉴스 본문 임베딩(벡터화)
- Transformer를 이용하여 요약문 생성
- BUT!! 파라미터 숫자가 엄청 많아 느리다!!
- 한국어 요약을 위해 한국어 BERT 적용
- 학습 시 굉장히 높은 ROUGE 수치 획득



Document (News): “또한, 개인정보가 유출된 것으로 추정되는 고객에게는 이메일과 문자 메시지로 개인정보 유출사고에 관해 안내했다고 설명하면서, 안내를 받은 고객들은 회원계정의 안전한 보호를 위해 가급적 비밀 번호를 변경해 달라고 당부했다.”

언어모델 Final

입력

北	정보탈취	전문	해킹조직	'APT37'	전모	공개
---	------	----	------	---------	----	----



언어모델
BERT/GPT/ELECTRA

언어모델

입력 차원 : 문장개수x문장길이

출력 차원 : 문장개수x문장길이x768

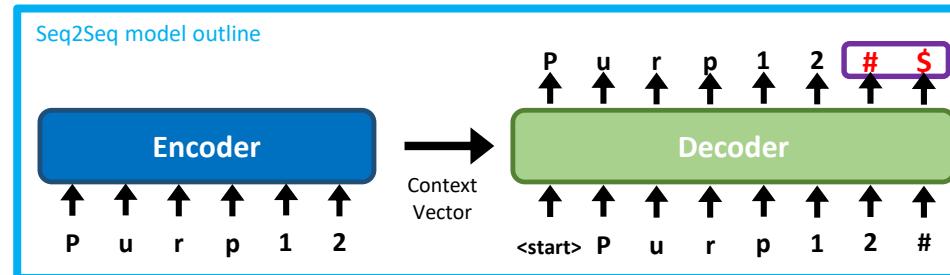
$1 \times 7 \rightarrow 1 \times 7 \times 768$

출력

base: 768

large: 1024

0.01	0.04	0.02	0.13	0.53	0.1	0.07
0.2	0.6	0.5	0.21	0.01	0.22	0.15
0.05	0.02	0.08	0.32	0.05	0.05	0.07
0.4	0.1	0.3	...	0.2	0.34	0.25
...



문서 요약기술

▪ Presumm 실제 적용 문제점 1)

- 추론 시간이 너무 오래 걸림
 - 서비스 고려 X, 배치 처리 구현 X
 - 뉴스 한개 요약 시 **18초**???????
 - 배치 처리가 없어서 10개 요약 시 180초
 - 다 버리고 코드 처음부터 다시 시작
 - 추론 서버 쪽에서 모델 및 api import **미리** 수행
 - 요약 시 요약 함수만 수행하도록 2단계 코드 분리
 - 18 → 4초로 시간단축!!!!

1. 라이브러리 호출(1s)
ex) import torch, ...

2. Load model, Tokenizer(14s)
ex) import torch, ...

3. 추론 실행(4s)

- 이동통신 3사 본인인증 애플리케이션(앱) 기반 사설인증서인 '패스(point) 인증서'가 오는 9월 시중은행에 처음으로 적용돼
- 이번 협약을 바탕으로 농협은행의 생활금융 플랫폼 '율원뱅크'와 패스 앱을 연계해 회원가입과 인증 절차를 편리하게 개선하며 농협은행 금융상품 관련 공동 마케팅을 추진하는 등 고객 편의 향상에도 적극 나설 계획
- 각종 사이버전자인증서인인증 사이버테크인증서비스를 위한 업무협약

이통3사-NH농협은행-아동, 비대면 금융 서비스 강화 위한 업무협약 체결 9월부터 '율원뱅크' 앱 회원가입, 출금이체등록, 추가인증 등에 '패스' 앱 활용 이동통신 3사 본인인증 애플리케이션(앱) 기반 사설인증서인 '패스(PASS) 인증서'가 오는 9월 시중은행에 처음으로 적용된다. SK텔레콤-KT-LG유플러스 등 이통 3사와 NH농협은행, 팁테크 보...

- ↳ 앞으로 은행에서도 공인인증서 대신 '패스' 쓴다 2020.07.13. 오전 9:28
- ↳ 이통3사 '패스' 첫발...공인인증서 없이 '율원뱅크' 쓴다 2020.07.13. 오전 9:35
- ↳ '패스' 인증, 시중은행에서 쓴다.농협앱서 9월부터 적용 2020.07.13. 오전 10:31
- ↳ 이제 농협서 'PASS' 쓴다.이통3사, '패스 인증' 시중은행에 첫 적용 2020.07.13. 오전 10:26
- ↳ 시중은행도 이제 이통3사 '패스 인증서' 쓴다 2020.07.13. 오전 10:51

문서 요약기술

■ Presumm 실제 적용 문제점 1)

- 근데 왜 이렇게 오래 걸리지?
- 영어 요약 모델 구현 후 테스트
- Multilingual BERT VS BERT 차이!
- BERT: Only for English, 110M parameters - 3.4만개 단어사전 => 2억개
- Multilingual BERT: 104 languages, 110M parameters - 12만개 단어사전 => 4억개
- 단어 개수가 많을수록 모델이 커지고 속도가 오래 걸림, 8배!!!!
- vocab 개수가 추론 속도에 큰 영향을 미친다
- multi-lingual BERT는 버리자!
- BERT 기반 seq2seq은 서비스엔 **무리**가 있다

뉴스개수	CPU(sec)	GPU(P100)
1	10	4
10	91	20
20	183	39
50	450	93
90	720	170

Presumm(vocab=120,000)

뉴스개수	CPU(sec)	GPU(P100)
1	9	0.6
10	43	2.5
20	58	5
50	105	11
90	150	20

Presumm(vocab=3,4000)

문서 요약기술

▪ Presumm 실제 적용 문제점 2)

- ROUGE 수치는 높으니 요약 결과도 좋겠지?? → 정성평가 실시
- But, 제법 인간을 따라하기 하는데... 뭔가 아쉽네??
- 문장 끝맺음이 부족하다!!

ex1)

- 미국 대형 제약회사가 랜섬웨어 공격으로 내부정보가 유출
- 공격이 처음 이뤄진 당시 알려진 바에 따르면 유출된 정보는 사회보장번호와 유사한 역할), 금융정보, 운전면허 정보, 여권 번호 등 민감한 정보가 다수 포함
- 이에 미국연방수사국(FBI)가 수사에 나선 상황이라고 강조

ex2)

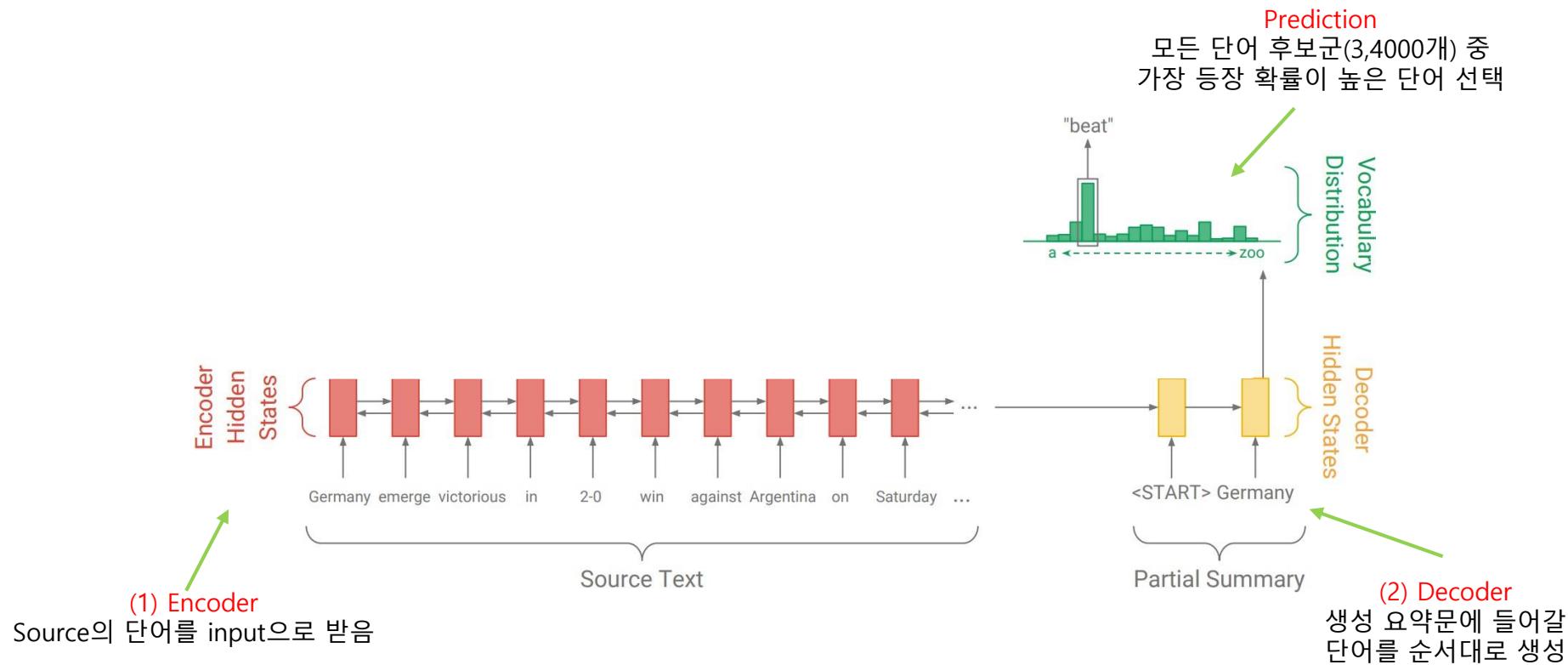
- 메타넷이 개발한 '솔메 클라우드'는 개인 PC, 이메일, 파일서버 등 다양한 방식으로 분산 관리되던 기업 내부 자료를 중앙 클라우드에 자동 보관·통합 관리해 기업 문서를 자산화
- 원격 근무 환경에서도 회사 주요 정보에 편리하게 접근해 안전한 내부 자료 공유가 가능하고 기업 협업 협업 환경에 최적화
- 기업 협업 내 문서관리 혁신·스마트 업무 환경 조성

ex3)

- 데이터 보호업체 아크서브(한국대표 유준철)의 어플라이언스 제품이 최근 2020 사이버 시큐리티 어워드 우수상을 수상
- 이번 우수상을 수상한 아크서브 어플라이언스는 세계적 보안회사인 소포스의 랜섬웨어 방어기술을 탑재한 일체형 백업 제품이라고 보안 분야 선도 기업의 솔루션을 일체형으로 적용해 랜섬웨어 보안에 대한 수준을 한 단계 더 높인 것으로 130/232
- **유준을 한 단계로 낮은 것**

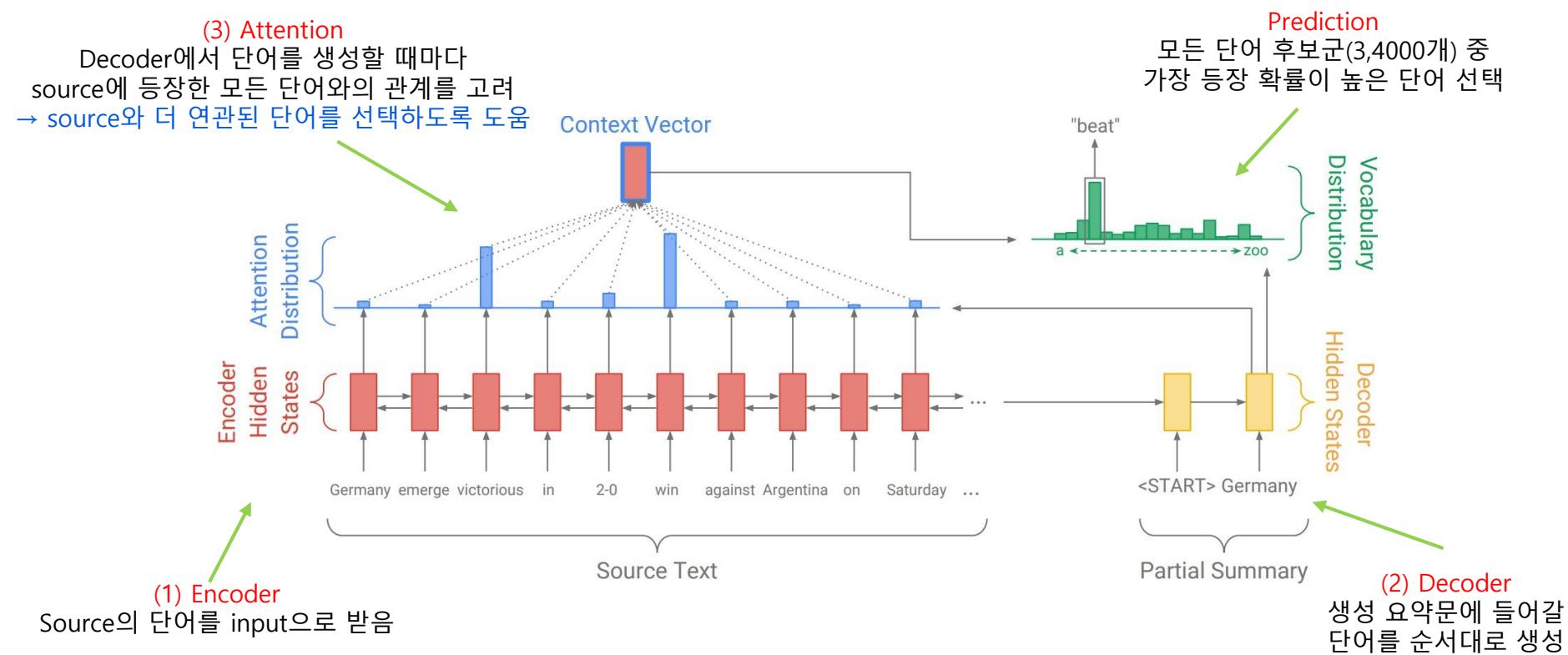
Automatic Summarization

- General Sequence-to-sequence(Encoder-Decoder) Summarization



Automatic Summarization

▪ Attention Sequence-to-sequence(Encoder-Decoder) Summarization



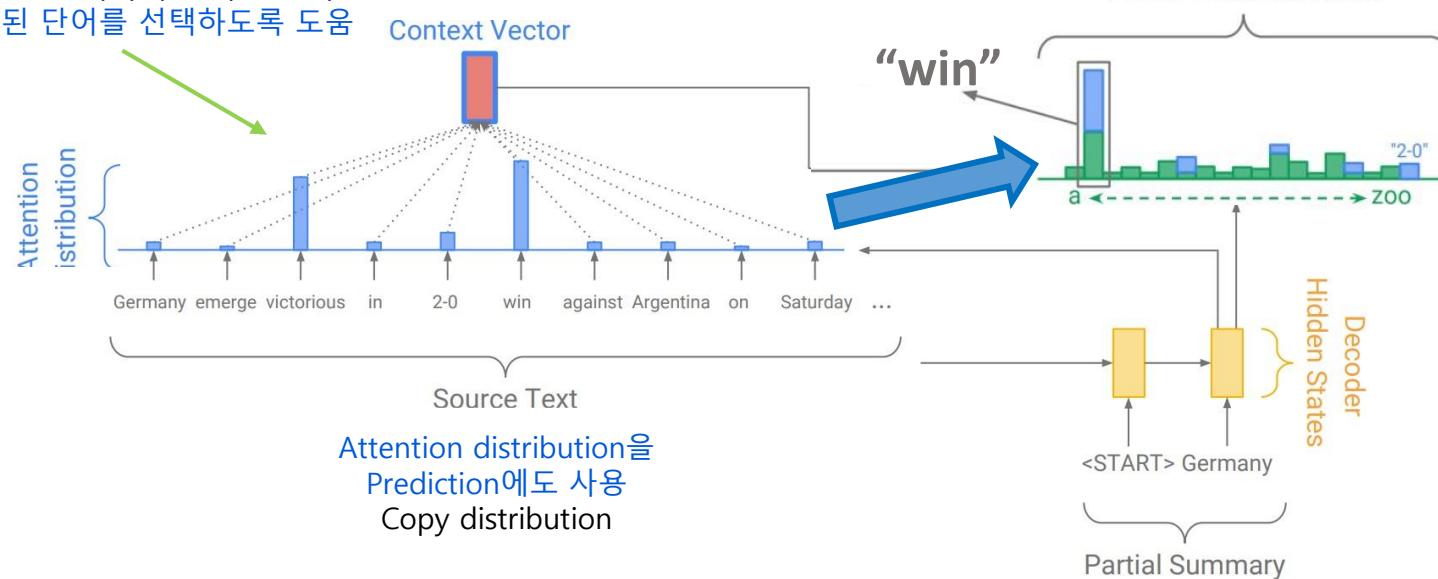
Automatic Summarization

▪ Copy mechanism - Pointer Generator(2017)

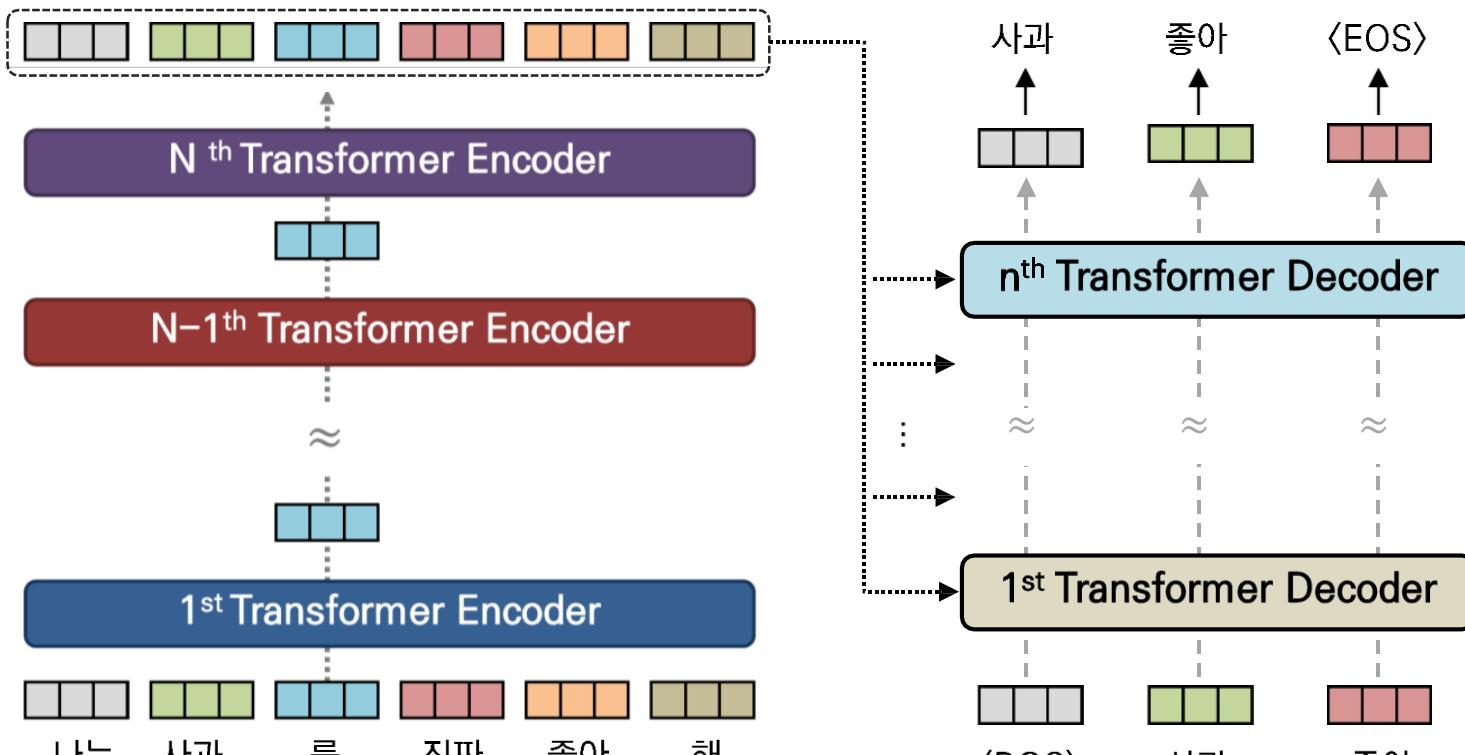
- Attention Sequence-to-sequence(Encoder-Decoder)를 통해 요약문 생성
- 단어를 예측할 때 source x 단어를 참조할 확률(attention)을 단어 예측에도 사용

(3) Attention

Decoder에서 단어를 생성할 때마다
source에 등장한 모든 단어와의 관계를 고려
→ source와 더 연관된 단어를 선택하도록 도움



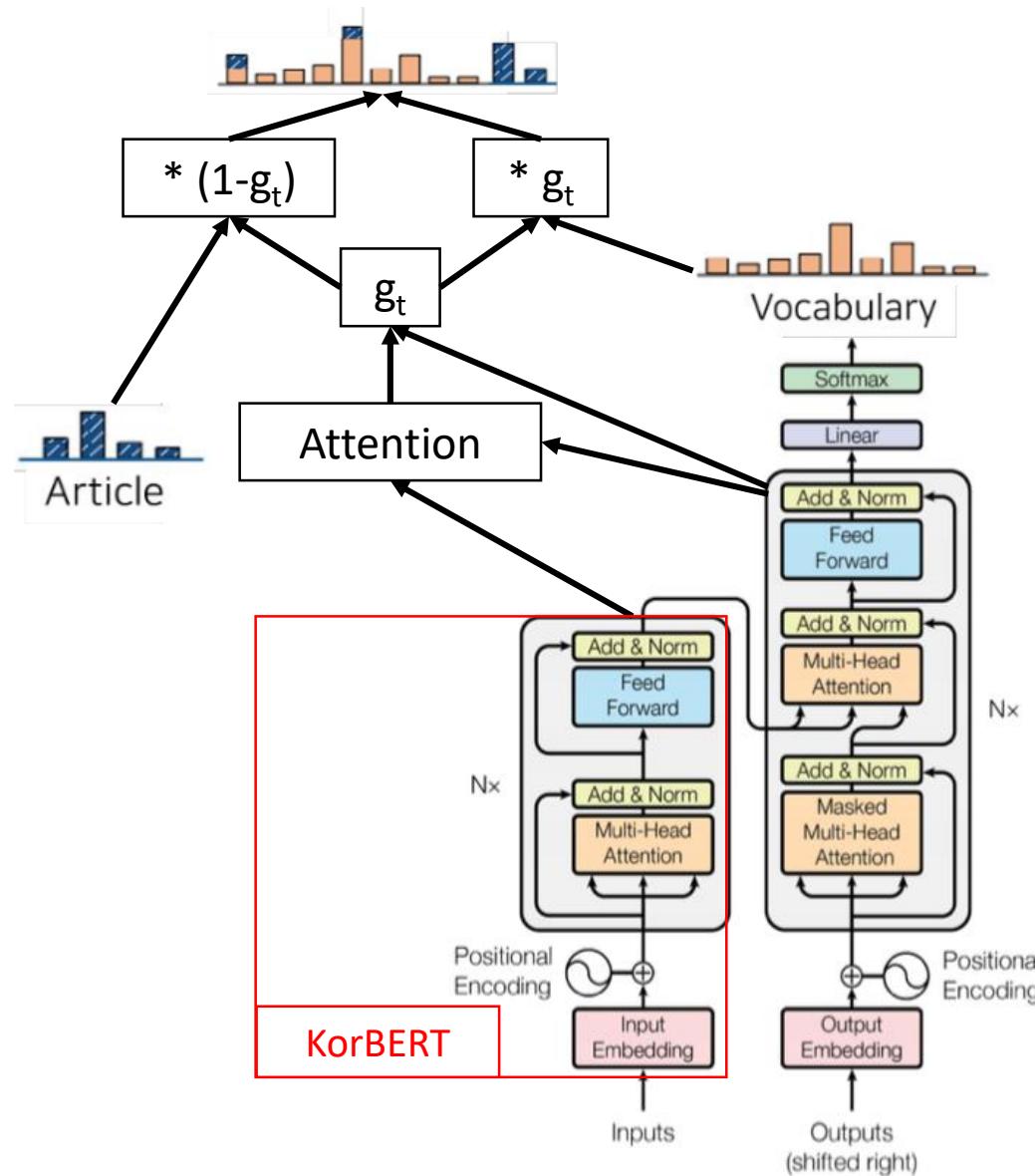
Copy mechanism



Encoder Learning Rate : $2e^{-3}$

Decoder Learning Rate : 0.1

Copy mechanism



문서 요약 기술

▪ Presumm 실제 적용 문제점 2)

- ROUGE 수치는 높으니 요약 결과도 좋겠지?? 정성평가 실시
- But, 제법 인간을 따르는 하는데 뭔가 아쉽네??
- 문장 끝맺음이 부족하다!!

▪ 문서 요약 성능 개선

- + Copy mechanism 도입 : 길고 어려운 단어 기억 문제 해결 => **BEST!!**
- + 강화학습 : 학습이 10배 느려짐, 미세한 성능 향상 or 하락 => **역효과!! 버리자!!**

	RL ratio	Cov.	Beam	R1	R2	RL
BERT + Transformer decoder	0 (b=4, ep=10)	-	1			
			5	22.00	6.29	14.99
BERT + Transformer decoder + Copy mechanism	0 (b=4, ep=5)	-	1	52.10	41.07	46.32
			5	52.74	42.30	46.70
BERT + Transformer decoder + Copy mechanism + Reinforcement	0.05 (b=1, ep=1)	-	1	52.01	41.18	46.26
			5	51.39	41.23	45.79

문서 요약 예제-1

▪ Ref (1)

- 지난 1 월 초 , 국내 특정 기관 에서 공지 한 ' 북한 2018 년 신년사 분석 ' 내용 으로 사칭 한 HWP 취약점 기반 악성 파일 이 발견 된 이후 유사 한 형태 의 스피어 피싱 공격 이 지속 적 으로 발견 되고 있 기 때문 에 이용자 들 의 각별 한 주의 가 요구 되고 있 어 이스트 시큐리티 관계자 는 현재 악용 되고 있는 고스트 스크립트 기반 취약점 은 한글 최신 버전 에서 취약점 이 제거 된 보안 업데이트 가 제공 되고 있 기 때문에 이 용자 들 은 반드시 최신 버전 으로 업데이트 해야 하 고 파일 리스 기반 의 지능 적 보안 위협 에 노출 되지 않 도록 각별 한 주의 가 필요 하 다고 전해

▪ BERT + Transformer decoder

- Beam=5: 국내 공공 기관 을 사칭 한 스피어 피싱 이 발견 되어 이용자 들 의 주의 가 요구 돼 이번에 발견 된 악성 파일 은 ' 2014 년 상반기 국내 기관 및 기관 ' 에서 유포 되고 있 으며 , 해당 악성 코드 는 지난 1 월 [UNK] [UNK] [UNK] [UNK]

▪ BERT + Transformer decoder + Copy mechanism

- Beam=1: 지난 1 월 초 , 국내 특정 기관 에서 공지 한 ' 북한 2018 년 신년사 분석 ' 내용 으로 사칭 한 HWP 취약점 기반 악성 파일 이 발견 돼 해당 문서 파일 포맷 내부 에는 밀봉 형 포스트스크립트 인 ' BIN 000 4 .eps ' 데이터 가 포함 되어 있 으며 , 복호화 가 완료 되면 다음 과 같은 한국 의 특정 웹 사이트 에 등록 된 PNG 형식 의 파일 을 로드 하 는 것 을 확인 할 수 있 어
- Beam=5: 북한 2018 년 신년사 분석 ' 내용 으로 사칭 한 HWP 취약점 기반 악성 파일 이 발견 돼 해당 문서 파일 은 zlib 공개 압축 라이브러리 를 사용 하 고 있 으며 , 압축 된 부분 을 해제 하 면 아래 비교 화면 과 같이 내부 의 코드 를 확인 할 수 있 어

문서 요약 예제-2

- **Ref (1)**
 - '아시아의 알프스'라고 불리는 일본 혼카이도의 명물 '르타오 치즈 케익'이 봄딸기를 만났다.
- **BERT + Transformer decoder**
 - 르타오 코리아가 봄시즌을 맞아 새콤달콤한 딸기가 듬뿍 올라간 '베리밀크프로프로그래'를 출시했다.
 - 베리밀크프로프로그래 (x) → 베리밀크프로마쥬 (o)
- **BERT + Transformer decoder + Copy mechanism**
 - 봄을 맞아 고소한 딸기를 가득 담은 신제품 '베리밀크프로마쥬'가 출시되어 눈길을 끌었다.
- **BERT + Transformer decoder + Copy mechanism + Reinforcement**
 - 딸기 덕후들의 취향을 저격할 르타오케익 신제품이 출시되었다.
- **정성 평가 결과 "BERT + Transformer decoder + Copy mechanism"이 더 우수**

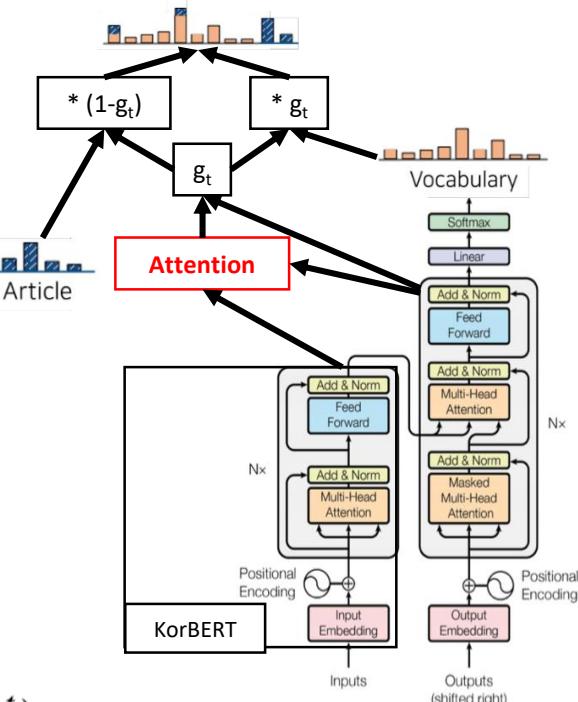
Coverage Mechanism

- Copy mechanism을 위한 Coverage Mechanism 모델 개발

- “BERT+Transformer decoder+Copy mechanism” 모델에서는 Copy mechanism이 주로 사용됨
- Copy mechanism 이용 시에 **반복 생성 현상**이 발생함
- Transformer decoder는 multi-head attention을 사용하기 때문에 coverage mechanism을 적용하기 어려움
- Coverage vector for Copy
 - The sum of copy attention distributions $c^t = \sum_{t'=0}^{t-1} a^{t'}$
- Extra input to the copy attention
- Coverage loss
 - Penalize repeatedly attending (copying) to the same locations

$$\text{covloss}_t = \sum_i \min(a_i^t, c_i^t)$$

$$\text{loss}_t = -\log P(w_t^*) + \lambda \sum_i \min(a_i^t, c_i^t)$$



문서 요약 예제-3

▪ 원문 (83)

- 소셜 네트워크 (SNS Social Network or Service) 트위터 가 **보이스 피싱** 범죄 예방에 큰 몫을 하고 있는 것으로 나타났다. 지식 경제부 우정사업본부 (본부장 김명룡) 트위터 (@koreapost)에 올라 있는 **보이스 피싱** 범죄 사례와 예방법이 팔로어들에게 실시간으로 빠르게 확산되면서 **보이스 피싱** 범죄 예방에 큰 효과를 보고 있다. **보이스 피싱** 사기 범이 우체국콜센터 번호 (1588-1900)를 가장해 신용 카드가 잘못 발급됐다고 속이고 있다는 글은 14만4천명이 리트윗했다. 또 우체국 홈페이지에 나오는 직원의 실명을 앞세워 계좌번호와 비밀번호를 묻고 돈을 가로채는 사기 수법에 대한 주의 글은 빠르게 리트윗되면서 무려 27만8천여명이나 읽었다. 우정사업본부는 **보이스 피싱**으로 국민들의 피해가 커짐에 따라 트위터를 통해 범죄 사례와 예방법을 알렸다. 올해만 검찰, 경찰, 우체국 등 사칭사례와 금융정보 유출, 자녀 납치 가장 등 사기 수법과 관련해 10건의 글을 올렸다. 이 글은 모두 60만5천명이 읽은 것으로 나타났다. 실제로 올해 우체국 사칭 **보이스 피싱** 피해 건수가 지난해보다 10% 이상 감소하면서 우정사업본부 트위터가 큰 몫을 했다는 평가를 받고 있다. 우정사업본부에 따르면 올 4월까지 **보이스 피싱** 피해 신고 건수는 345건으로 지난해 같은 기간 412건에 비해 17% 감소했다. 피해 신고로 지급 정지된 우체국 계좌 금액도 14억 원에서 7억 원으로 50% 감소했다.

▪ Ref

- 우정사업본부 트위터 (@koreapost)에 올라 있는 보이스 피싱 범죄 사례와 예방법이 팔로어들에게 실시간으로 빠르게 확산되면서 보이스 피싱 범죄 예방에 큰 효과를 보고 있어

▪ BERT + Transformer decoder + **Copy mechanism**

- Beam=5: 지식 경제부 **우정사업본부** 트위터 (@koreapost)에 올라 있는 보이스 피싱 범죄 사례와 예방법이 팔로어들에게 실시간으로 빠르게 확산되면서 **보이스 피해자들의 주의를 보고 있다고 밝혀 우정사업[UNK] [UNK] [UNK]**

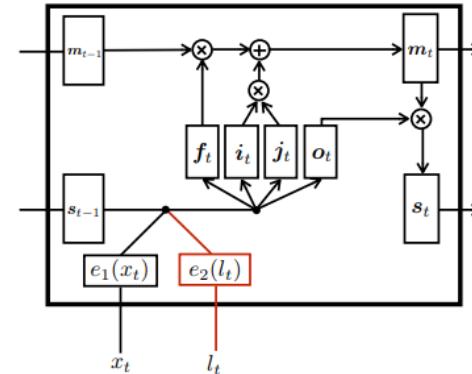
▪ BERT + Transformer decoder + **Copy mechanism** + **Coverage mechanism**

- Beam=5: 지식 경제부 우정사업본부 트위터에 올라 있는 보이스 피싱 범죄 사례와 예방법이 팔로어들에게 실시간으로 빠르게 확산되면서 **우체국 계좌 금액도 14억 원에서 7억 원으로 50% 감소**

요약문 길이 조절

■ 관련연구:

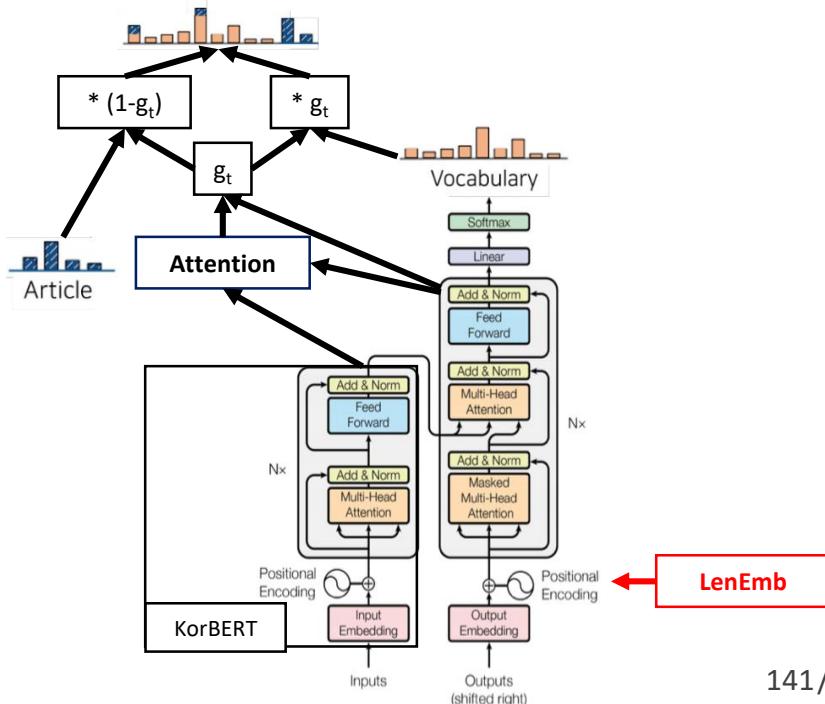
- Controlling Output Length in Neural Encoder-Decoders (EMNLP 16)
 - LenEmb model



■ 구현

- Transformer decoder에 Length Embedding 추가

- Length embedding 값을 word embedding에 더함
- Length = 원하는 요약문의 길이 - 현재까지 생성한 요약문의 길이



문서 요약 예제-4

▪ Ref (43)

- 트렌드 마이크로 가 배쉬 (Bash) 취약점에 대한 뉴스가 나오고 얼마 지나지 않아 관련 공격이 출현했다고 밝혀 트렌드 마이크로는 해당 취약점을 이용한 악성코드 (ELFBASHLITE . A)는 무차별 대입 공격 (Brute Force Attack)으로 사용자 계정 정보 획득과 DDoS 공격 기능을 보유하고 있으며 C & amp; C 서버에도 접속한다고 전해

▪ BERT + Transformer decoder + Copy/Coverage + LenEmb

- **Beam=5, out_len=정답길이**: 트렌드 마이크로 가 배쉬 (Bash) 취약점에 대한 뉴스가 나온지 채 몇 시간 지나지 않아 벌써 관련 공격이 출현했다고 보도해 트렌드 마이크로는 자사 서버 보안 솔루션인 '딥 시큐리티' (Deep Security)의 경우 배쉬운 취약점을 가진 패치되지 않은 서버를 보호하며 가상 패치 기능을 통해 취약점이 있는 서버가 패치될 때까지 서버를 취해
- **Beam=5, out_len=50**: 트렌드 마이크로 가 배쉬 (Bash) 취약점을 이용한 코드가 포함된 악성코드 샘플에 주목하고 탐지명 'ELFBASHLITE . A'으로 나타나
- **Beam=5, out_len=100**: 트렌드 마이크로 가 배쉬 (Bash) 취약점에 대한 뉴스가 나온지 채 몇 시간 지나지 않아 벌써 관련 공격이 출현했다고 밝혀 트렌드 마이크로는 자사 서버 보안 솔루션인 '딥 시큐리티' (Deep Security)의 경우 배쉬운 악성코드 샘플에 주목하고 탐지명 'ELFBASHLITE . A'으로 알려
- **Beam=5, out_len=150**: 트렌드 마이크로 가 배쉬 (Bash) 취약점에 대한 뉴스가 나온지 채 몇 시간 지나지 않아 벌써 관련 공격이 출현했다고 밝혀 트렌드 마이크로는 실제로 해당 취약점을 이용한 코드가 포함된 악성코드 샘플에 주목하고 탐지명 'ELFBASHLITE . A'으로 알려진 이 취약점은 DDoS 공격 수행이 가능하며 무차별 대입 공격 (Brute Force Attack) 기능도 있어 공격자가 사용자 계정 정보를 획득하는 것에서부터 웹사이트 자체를 취약점으로부터 보호하고 있다고 설명해

MASS 기반 한국어 문서 요약

- 국보연 데이터 학습 결과

모델		R1	R2	RL
Transformer (형태소 단위 BPE)	형태소(태그x)	47.55	20.98	30.14
Transformer + Copy (형태소 단위 BPE)	형태소(태그x)	50.97	28.22	37.95
BERT + Transformer decoder + Copy (max target len=200)	형태소	49.80	39.43	43.96
BERT + Transformer decoder + Copy/Coverage	형태소	50.28	39.85	44.41
MASS (형태소 단위 BPE)	형태소(태그x)	57.35	44.99	51.44
MASS + Copy (Copy no pre-training) (형태소 BPE)	형태소(태그x)	58.06	45.65	51.73
MASS + Copy (Copy pre-training) (형태소 BPE)	형태소(태그x)	57.96	46.51	52.64
MASS + Copy (Copy pre-training) (어절 BPE)	형태소(태그x)	58.56	46.89	52.90
MASS + Copy + Coverage (Copy pre-training)(어절 BPE)	형태소(태그x)	58.88	47.12	53.25
MASS + Copy + Coverage + LenEmb(Copy pre-training) (어절 BPE)	형태소 (태그x)	LenEmb 30	41.18	32.28
		LenEmb 60	53.99	41.30
		LenEmb 90	56.80	43.31
				49.91

문서 요약 결론

- ROUGE가 높은 SOTA 모델일 지라도 정성평가 전까진 신뢰할 수 없다
- NLU에선 좋은 임베딩(BERT or BART)도 중요하지만 디코더의 후처리(**Copy/Coverage mechanism**)가 훨씬 큰 영향을 미친다
- BERT를 쓸 경우 추론 속도가 매우 매우 느리다
 - 실제 서비스를 고려한다면 성능을 조금 포기하고 Transformer를 선택

문서 요약 결론

- ROUGE가 높은 SOTA 모델일 지라도 정성평가 전까진 신뢰할 수 없다
- NLU에선 좋은 임베딩(BERT or BART)도 중요하지만 디코더의 후처리(**Copy/Coverage mechanism**)가 훨씬 큰 영향을 미친다
- BERT를 쓸 경우 추론 속도가 매우 매우 느리다
 - 실제 서비스를 고려한다면 성능을 조금 포기하고 Transformer를 선택

국립대병원 해킹·랜섬웨어 대응 정보보호 강화방안 마련

(2022-06-03 07:47:49, 아시아경제)

- 교육부는 국립대병원을 노린 랜섬웨어 공격 등을 막기 위해 정보보호 강화방안을 국가정보원과 합동으로 마련했다고 3일 밝혀
- 교육부는 사이버 위협 사전 방지 방안으로 국가사이버 공격을 사전에 막기 위해 국립대학병원 기반시설의 백업 시스템 구축과 위기 발생 시 복구방안 등 보호 대책을 점검해

[자세히 보기 >](#)

OT 네트워크에도 랜섬웨어 심는 개념 증명 공격 성공해

(2022.06.02 11:51, 보안뉴스)

- 보안 업체 포스카웃은 개념 증명 차원에서 실시한 S4IoT 랜섬웨어 공격을 선보임으로써, 앞으로 OT 망을 노리는 랜웨어 공격웨어 공격자들의 움직임이 있을 것이라고 경고해
- 이번 실험용 공격의 특징은 사물인터넷 장비로 최초 침투하고, 기존에 익숙한 취약점이 발굴되거나 익스플로잇 되지 않았고, 기존에 알려진 취약점과 해킹 전략들만으로 공격이 구성됐다고 전해

[자세히 보기 >](#)

강병탁 에이아이스페라 대표 "OSINT와 CTI 활용한 보안위협 탐지"

(2022.06.02 16:38, 데일리시큐)

- 제10회 CPS 보안워크숍이 지난 5월 26일 제주 메종글리드호텔에서 온오프라인 병행해서 성황리에 개최돼
- 강병탁 대표는 OSINT와 CTI를 활용한 보안위협 탐지 방법에 대해 설명했고, 사이버상 IP 정보를 모두 모아서 사이버위협 탐지에 활용할 수 있는 시스템 Criminal IP도 소개해

[자세히 보기 >](#)

'사이버 안보동맹' 뒷받침 급하다

(2022-06-02 11:21:09, 문화일보)

- 지난달 21일 열린 한·미 정상회담에서 사이버 안보가 회담의 핵심 어젠다로 등장한 이유는 몇 가지로 요약된다.
- 양국 정상은 공동성명을 통해 양국이 "국가 배후의 사이버 공격 등을 포함해 북한으로부터의 다양한 사이버 위협에 대응하기 위한 협력을 대폭 확대"하고, "핵심·신흥 기술과 사이버 안보 협력을 심화하고 확대"해 나갈 것이라고 밝혀

[자세히 보기 >](#)

"나토, '중국의 위협' 새로운 전략 개념에 사상 처음으로 포함"

(2022-06-02 18:20:43, 뉴시스)

- 미국 디펜스원에 따르면 줄리안 스미스 나토 주재 미국 대사는 1일 오는 29~30일 스페인 마드리드에서 열리는 나토 정상회의에서 이 같은 내용을 담은 전략 개념 문건이 공개될 것이라고 밝혀
- 스미스 대사는 이번 나토 정상회의가 러시아의 우크라이나 침공과 스웨덴·핀란드의 나토 가입 신청에 초점을 맞추겠지만 새로 도입할 전략 개념은 중국의 위협, 사이버 공격, 신기술 등 앞으로 10년의 문제를 아울러야 한다고 전해

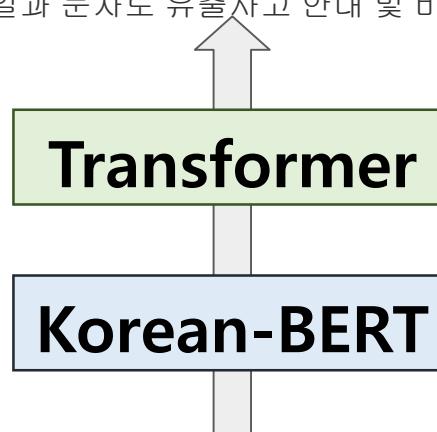
[자세히 보기 >](#)

요약모델 개발 로드맵

요약모델 개발 로드맵

- 1) 데이터 정제
- 2) 요약 데이터셋 구축
- 5) 모델 선정
- 6) 학습
- 7) 학습결과 확인
- 8) 추론 코드 작성
- 9) 추론서비스 테스트
- 9) 서비스

Summary (Sentences): 개인정보 유출 추정 고객들에게 이메일과 문자로 유출사고 안내 및 비밀번호 변경 요청해



Document (News): “또한, 개인정보가 유출된 것으로 추정되는 고객에게는 이메일과 문자 메시지로 개인정보 유출사고에 관해 안내했다고 설명하면서, 안내를 받은 고객들은 회원계정의 안전한 보호를 위해 가급적 비밀번호를 변경해 달라고 당부했다.”

언어모델 활용 Ⅲ

번역

Neural Machine Translation

** 분야 전문 **번역** 모델

Domain specific NMT

- 구글/파파고 번역기는 일반 번역은 정말 잘됨
- 하지만 특정 도메인의 전문 용어(도메인 고유명사)들은 번역이 잘 안됨
- ** 분야는 URL, filename, ip등이 많고 전문 용어들이 많아 구글 번역이 소화 못함
 - ex) CVE-2015-2545 EPS exploit ([image1.eps](#), MD5 [a90a329335fa0af64d8394b28e0f86c1](#))
 - ex) The decryption function is 1-byte XOR with key from " `\x00` " to " `\xff` " and replacement of the Odd byte
- **분야 특화된 한/영 번역기 개발 필요

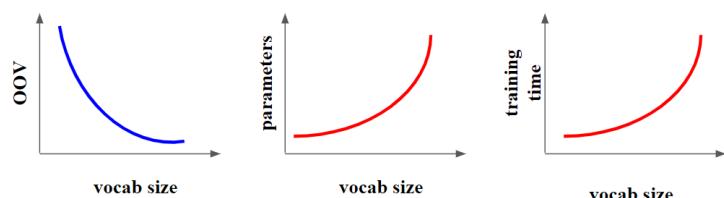
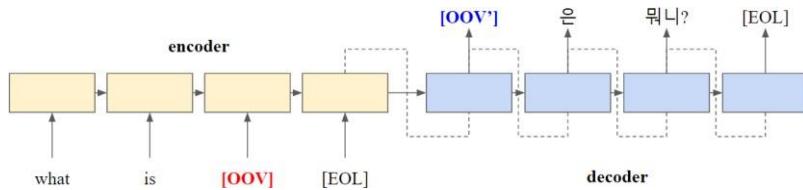
NMT 학습 데이터

- 괜찮은 성능의 번역기 개발을 위해 최소 300만 문장 쌍 이상의 데이터가 필요
- N사, S사는 양질의 800만 문장 쌍으로 학습
- 병렬 문장 쌍을 얻기 위해 번역 데이터 구매
 - 특정 도메인 전문 번역 의뢰 : 1문장에 대략 1400원 이상, 10만 문장에 **1.4억!!**
 - 번역 단가: 번역가(700원), 감수자(200원), PM(200원), 회사(300원)
 - 이미 번역 된 문장 : 1문장에 200원정도, 300만 문장에 **6억!!!**
 - Open된 한영 병렬 데이터셋
 - AiHub의 Korean-English parallel corpus(160만)
 - ~~OpenSubtitles의 Korean-English subtitles parallel corpus~~
 - IwsIt 등
 - 데이터 정제 필요
 - 구어체/대화체/문어체 중, 만들고자 하는 번역기의 입력과 비슷한 셋만 취하자
 - 아니면 노이즈!

NMT 학습 데이터

• Rare Word Problem

- Out-of-vocabulary (OOV) 문제
- 희기 단어, 이름, 숫자
- 단어장에 없는 단어에 대한 학습, 번역의 어려움



• Word Piece Model

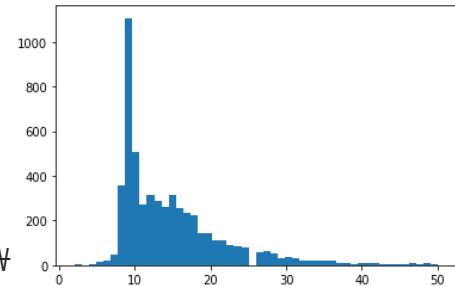
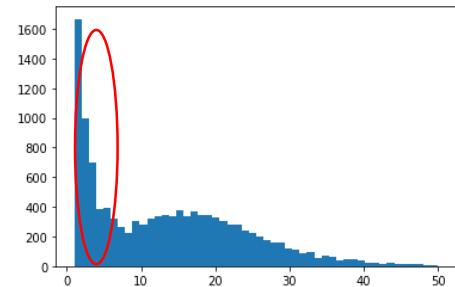
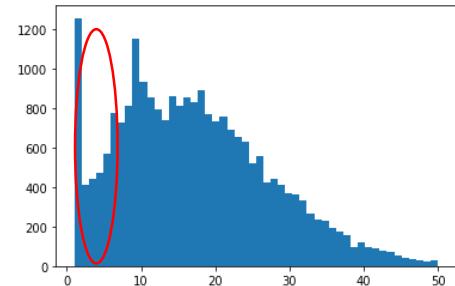
- Sub-word 단위 분절(Byte Pair Encoding)
- 단어는 의미를 가진 더 작은 sub 단어의 조합
- UNK 처리에 효과적
- 정확도 상승 효과



현재 TED 웹 사이트에 는 1 000 개 가 넘는 TED 강연 들이 있습니다. 여기 계신 여러분의 대다수 는 정말 대단 한 일 이라고 생각 하시겠죠. 전 다릅니다. 전 그렇게 생각 하지 않아요. 저는 여기 한 가지 문제점 이 있 다고 생각 합니다. 왜냐하면 강연 이 1 000 개 라는 것 은 공유 할 만한 아이디어 들이 1 000 개 이상 이라는 뜻 이 되기 때문 이죠. 도 대체 무슨 수로 1 000 개 나 되는 아이디어 를 널리 알릴 건가요? 1 000 개의 TED 영상 전부 를 보면서 그 모든 아이디어 들을 머리 속에 집어 넣으려고 해도 250 시간 이상의 시간 이 필요 할 겁니다. 250 시간 이상의 시간 이 필요 할 겁니다. 간단 한 계산 을 해 봤는데요. 정말 그렇게 하 는 경우 1 인 당 경제 적 순실 은 15 000 달러 정도 가 됩니다.

번역 학습데이터 구축-1. 번역 데이터 선정

	문서 개수	문장수	평균 단어	중복제거문장	max/min token
malware_TM	1752	5442	14	1243	3~50
Blog_SM	489	23343	16	7512	3~50
Blog_FE	116	8988	16	704	3~50
Blog_RF	119	6145	19		4~35
Blog_TM	1614	58237	16	4668	4~35
TR_FE	59	112	15	294	4~35
TR_RF	1792	2414	18	282	4~35
NEWS_TM	155	1339	18	19	4~35
총합	6096	106020			



- 3~50 word, 직역+의역(적절히), 중복제거완료
 - ['\\Wxad', '—', '\"\\Wxa0', '|', 'хакер', '●\\Wxa0\\Wxa0\\Wxa0\\Wxa0', "...", '→', 'â', '\\Wxa0\\Wxa0\\Wxa0\\Wxa0\\Wxa0', "", "", '年员工工资性津贴额统计报告', '\\Wxa0\\Wxa0\\Wxa0\\Wxa0\\Wxa0', '|', '\\Wxa0\\Wxa0\\Wxa0\\Wxa0', '\\Wxa0', '\"\\Wxa0\\Wxa0\\Wxa0\\Wxa0', "", "", '一带一路', 'Wu2028', "", "", '\\Wxa0\\Wxa0\\Wxa0\\Wxa0\\Wxa0\\Wxa0\\Wxa0\\Wxa0\\Wxa0\\Wxa0\\Wxa0', 'â', 'у', '\\Wxa0\\Wxa0\\Wxa0\\Wxa0', 'Wxa0—', '\"\\Wxa0\\Wxa0\\Wxa0\\Wxa0', 'Wxa0', '|', '○\\Wxa0\\Wxa0\\Wxa0\\Wxa0', 'â', 'â', 'Д', "", 'Wxa0\\Wxa0\\Wxa0\\Wxa0\\Wxa0\\Wxa0\\Wxa0\\Wxa0', 'Wxa0\\Wxa0\\Wxa0\\Wxa0\\Wxa0\\Wxa0', 'Wuf0b7Wuf020', '北京研创达科技有限公司', "...", 'ê', 'Wxa0\\Wxa0', 'ЦНИИХМ', '|', 'ó', "", 'Wxa0\\Wxa0\\Wxa0\\Wxa0', 'Wxa0\\Wxa0\\Wxa0', 'Wxa0\\Wxa0', 'Wxa0', '—Wxa0']

번역 학습데이터 구축-1. 번역 데이터 선정

마일리지	번역	번역	번역
B	62c3b52b5310393dbf0590bc246161249632a1d2f21c3aa7fb779dc8018a0edf5078a0940abc31a7fa271483ac345044a91a0e21c517bceb85091cd3fca310f70c77b260ee3fd2754cd4f289efce709519aad34fa3cb84663655a6240e459731ab8feef7d7f3706a42f996a3291d24a7ab2c5eb67d98236eb73995d587576ad3ecb650c471d7c8291d084ffffd634da0eddc9a473d29792d5033fe5fdcbf4ddd64d187bed40d023e14d41b1a80d528f5c12dcf743fc84de91530567d3244e09e77689e7752470501d26cf8a5e2eb9b4e1ac372b27b2151268e0acf024e355f9981dab2787f72997afb09fb98ada159f78c3e93f9d3fa83f844e580620d08322a87fb207ae29baaa300c2377625b745667a516e2243e1904ef81b4f7b97b5da1b09875c102bbe89ad636096efca6b04d6b843529eb9717d822f7b0b42a087c7332a0170a01e656cf7089a0d68a1803c3e2ba64ba8996c8eb5fffa0898940cb4c0ecb9b4511065cb56bd162e143c22cf3e3ee6617ba5a4852182cb0781f18f1c6f43bedad8b0c3f60d71a2a6c1fab297e144483f17deeb5150bdb6c73755a4d746e41e18bb637062881aca207186dc3d005e79c857e025f89ce2a1b3e52ecfd9eedee0541b9a5baf2cb2b1915aef1d034efd4edd4b3c030b508669da1e2aaef2db627ff946ff64910cf909c81ae51294c4bb6477ee2c620aae1d0f7a7208b6b5f4909c420e208e4728116e8b0f4254c9f741d864f9618cddbe3f51b71f602066fa2993f2455971244350178008cc671fb739b53d79b594c80e69047421ce1666bf480a5862210b9e033f270379bb95c1d1fadd16bf0d21db5bfb9268ae595ac	중	hxxp://107.170.215.53/woxkt/trkmix.php?device=desktop&cou=e=Austria&browser=ie&browserversion=11&carrier=%3F&cost=0.0=Mozilla%2F5.0+%28Windows+NT+6.1%3B+WOW64%3B+Trident%2F7.09718dbaf2fda5d19bab0d7c1866e0664035ETW
B		Acknowledgement	
B		RDP	
B		HKEY_CURRENT_USER\Software\SimonTatham\PutTY	
		HKEY_CURRENT_USER\Software\SimonTatham\PutTY\SSHHostKeys	
		HKEY_CURRENT_USER\SYSTEM\CurrentControlSet\Services\PortP	
		%systemroot%\Windows\System32\winevt\Logs\Microsoft-Termi	
		%systemroot%\Windows\System32\winevt\Logs\Security.evtx	
		Amcache	
		Prefetch	
		Cyber	
		Trojan	
		Url	
		ii.	
		Appendix	
		Methodology	
		{0000002F-0000-0000-C000-000000000046}	
		{00000300-0000-0000-C000-000000000046}	
		{00000301-A8F2-4877-BA0A-FD2B6645FB94}	
		{00000303-0000-0000-C000-000000000046}	
		{00000304-0000-0000-C000-000000000046}	
		{00000305-0000-0000-C000-000000000046}	
		{00000306-0000-0000-C000-000000000046}	
		{00000308-0000-0000-C000-000000000046}	
		{00000309-0000-0000-C000-000000000046}	
		{0000030B-0000-0000-C000-000000000046}	
		{00000315-0000-0000-C000-000000000046}	
		{00000316-0000-0000-C000-000000000046}	
		\$handle.Start([ref]\$True)	
		\$instance.Connect()	

번역 학습데이터 구축-1. ** 용어 사전 정의

A 구분	B 단어	C 해석
2 TM	malware	멀웨어
3 TM	This malware	이 멀웨어는
4 TM	This malware has been reportedly used in targeted attacks.	이 멀웨어는 표적 공격에 사용된 것으로 알려져 왔다
5 TM	targeted attacks	표적 공격
6 TM	has been reportedly	알려져 왔다
7 TM	~targeted attacks against financial institutions by the Lazarus group.	Lazarus 그룹에 의해 금융 기관들을 대상으로 한 표적 공격에 ~
8 TM	~used in attacks targeting Saudi Arabia Government.	사우디아라비아 정부를 표적으로 한 공격에 사용된~
9 TM	~arrives on a system	시스템에 침투한다
10 TM	personal data	개인 정보
11 TM	contact a remote server	원격 서버와 통신하다
12 TM	report infection	감염을 알리다
13 TM	It contacts its control server to report its infection	그것은 감염을 알리기 위해 컨트롤 서버와 통신한다
14 TM	control server	컨트롤 서버
15 TM	custom payload	커스텀 페이로드
16 TM	The firewall may be set to a permissive/passive state, or may be disabled all together.	방화벽은 허용/수동 상태로 설정되거나 모두 비활성화 상태로 설정될 수 있다.
17 TM	permissive/passive state	허용/수동 상태
18 TM	disabled	비활성화
19 TM	Other security features of Windows may be disabled as well.	원도우의 다른 보안 기능을 또한 비활성화될 수 있다
20 TM	credentials	크레덴셜 정보
21 TM	attempts to locate and exfiltrate ~	~ 유출을 시도하다
22 TM	The malware arrives to a victim's system	멀웨어는 피해 시스템에 침투하다
23 TM	arrives to system	시스템에 침투하다
24 TM	The macro embedded in the Microsoft document	MS 문서에 임베디드된 매크로
25 TM	Microsoft	MS
26 TM	embedded	임베디드된
27 TM	extracts an embedded ZIP file ~	임베디드된 ZIP 파일을 추출하다
28 TM	The embedded shellcode	임베디드된 셀코드
29 TM	configuration settings	환경 설정값들
30 TM	This malware contacts a remote server to receive configuration settings, and can connect to an IRC server to accept commands.	이 멀웨어는 환경 설정값들을 받기 위해 원격 서버와 통신하고, 추가 명령들을 수신하기 위해 IRC 서버와 연결할 수 있다.

A 구분	B 단어	C 해석
2 FE	~with version 4.2, when a major build took place.	중요 빌드가 수행된 버전 4.2
3 FE	invoke OS functionality	OS 기능 호출
4 FE	own custom plugin.	자체 커스텀 플러그인
5 FE	rely on standards-compliant API bindings	표준-호환 API 바인딩
6 FE	wrapper	래퍼
7 FE	standards-compliant	표준-호환
8 FE	API bindings	API 바인딩
9 FE	app developer or the SDK creator	앱 개발자 또는 SDK 제작자
10 FE	Monitor and upload device location	기기의 위치 정보 감시 및 업로드
11 FE	These apps are still available in the App Store as of May 25, 2016.	이러한 앱들은 2016년 5월 25일 현재 여전히 앱스토어에 있다
12 FE	change log, changelog	변경 로그
13 FE	malicious capabilities	악성 기능들
14 FE	The delivery of the abovementioned malicious capabilities	위에 명시된 악성 기능들의 전달
15 FE	community contributed framework	커뮤니티 참여
16 FE	view controller class	뷰 컨트롤러 클래스
17 FE	view controller hierarchy	뷰 컨트롤러 계층
18 FE	workflow of Vpon Cordova plugin	Vpon Cordova 플러그인의 워크플로우
19 FE	Apache repository	Apache 레파지토리
20 FE	querying	쿼리를 요청하다
21 FE	DNS resolution	DNS 주소변환
22 FE	~resolve a C2 domain~	C2 도메인으로 주소변환하다
23 FE	adversary	공격자
24 FE	operator / developer	운영자 / 개발자
25 FE	artifact	아티팩트
26 FE	iterate	반복하다
27 FE	the latest version at the time of posting	포스팅하는 시점에서의 최신 버전
28 FE	Reboots the operating system	OS 재부팅
29 FE	SysInternals pipelist program	SysInternals pipelist 프로그램
30 FE	command-line syntax	명령줄 구문
31 FE	Visual Basic Command Line Compiler	VB 명령줄 컴파일러
32 FE	Windows Native API	Windows Native API
33 FE	form data	폼 데이터
34 FE	lateral movement	전이 공격
35 FE	infrastructure	인프라
36 FE	padding bytes	패딩 바이트
37 FE	intelligence	인텔리전스
38 FE	threat hunting	위협 헌팅

번역 학습데이터 구축-2. 상세한 feedback

- 1. to 부정사를 좀 명확히 해석할 필요가 있을 듯 합니다.
ex> The malware exploits CVE-1111-1111 to perform blah~
멀웨어는 blah 를 수행하기 위해 CVE-111-1111을 익스플로잇 한다 가 좀 더 정확할 것 같은데
멀웨어는 CVE-1111-1111을 익스플로잇 하여 blah를 수행한다 로 해석된 문장이 자주 보임
- 2. may, can 등과 같은 조동사 뉘앙스를 안 살리는 경우가 좀 있던데, 이에 유의 필요
- 3. 앞문장에 malware 이름이 있고, 뒤에서 대명사 등으로 받을 경우.. 번역에서 앞문장의 malware 고유 이름을 쓰는 경우가 가끔 보입니다.
- 4. control server 를 “제어 서버” 가 아니라, “컨트롤 서버” 로 해석
- 5. collected data에서 “수집한” 또는 “수집된” 으로 혼용 해석
- 6. that targets Windows platform은 항상 “윈도우 플랫폼 목표용” 이라고 해석, “윈도우 플랫폼을 목표로 하는” 또는 “대상으로 하는” 이 더 맞을 거 같은데..
- 7. The malware reportedly has been used in 이라는 문장이 “사용되어 왔다고 알려 졌다”로 해석되는데, 보고서 상에서 문장 해석에서는 단순히 알려진 거 라기보다는 “보고에 의하면, 사용 되어 왔다” 라고 명확하게 쓰는 것이 나을 것 같기도 합니다.
- 8. its 라는 대명사를 거의 해석하지 않고 뛰어 넘고 있는데, 이것을 “자신의” 정도로 해석이 필요하다고 봐요~~!!

번역 학습데이터 구축-2. 상세한 feedback

- 1. to 부정사를 좀 명확히 해석할 필요가 있을 듯 합니다.

ex> The malware exploits CVE-1111-1111 to perform blah~

멀웨어는 blah 를 수행하기 위해 CVE-111-1111을 익스플로잇 한다 가 좀 더 정확할 것 같은데

멀웨어는 CVE 1111 1111을 이스프로이 하여 blah를 수행하다 고 해서 되 므자이 자즈

				dropped by ~를 ~에 의해 드롭되거나 맞지 않을까?
48	This Trojan Spy arrives on a system as a file dropped by other malware or as a file downloaded unknowingly by users when visiting malicious sites.	이 트로이 목마 스파이는 다른 멀웨어가 드롭하였거나 악의적인 사이트 방문 시에 사용자 모르게 다운로드되는 파일을 통해 시스템에 침투한다.	이 트로이 목마 스파이는 다른 멀웨어에 의해 파일이 드롭되거나 악의적인 사이트 방문 시에 사용자 모르게 다운로드되는 파일을 통해 시스템에 침투한다.	
49	It may be downloaded by other malware/grayware from remote sites. It connects to certain websites to send and receive information.	이 트로이 목마 스파이는 원격 사이트에서 다른 멀웨어/그레이웨어에 의해 다운로드 될 수 있다. 이 트로이 목마 스파이는 특정 웹사이트들과 통신하여 정보를 전송하고 수신한다.		
50	<<doc>>_malware_TM_015ce9f6-026d-4a6f-aa4e-9cff527efb78			
51	Worm:Linux.Jkirabot.A is a Bot agent that targets Linux operating systems running on ARM, MIPS, and x86 architecture.	Worm:Linux.Jkirabot.A는 ARM, MIPS, x86 아키텍처에서 실행되는 리눅스 OS 목표용 봇 애이전트이다		
52	The malware can exploit JAWS Webserver unauthenticated shell command execution vulnerability to propagate.	이 멀웨어는 JAWS 웹서버의 인증되지 않은 쉘 커맨드를 실행 취약점을 익스플로잇하여 전파한다	쉘 커맨드를 실행 취약점=> 쉘 커맨드 실행 취약점	
53	It also can exploit the vulnerability of Huawei router known as CVE-2017-17215 to propagate.	이것은 또한 CVE-2017-17215로 알려진 화웨이 라우터의 취약점을 익스플로잇하여 전파한다	또한 CVE-2017-17215로 알려진 화웨이 라우터의 취약점을 이용한다	
54	It can kill the processes on the list of the pre-defined names.	사전 정의된 이름 리스트에 있는 프로세스를 종료할 수 있다		
55	Moreover, the malware accepts commands to conduct various types of DoS attacks against a given target.	나아가 이 멀웨어는 설정된 표적 대상으로 다양한 도스 공격들을 수행하기 위한 명령들을 수신한다	나아가 이 멀웨어는 주어진 목표에 따라 다양한 도스 공격들을 수행하기 위한 명령들을 수신한다	given이 설정된? 주어진?

- 6. that targets Windows platform은 항상 "윈도우 플랫폼 목표용" 이라고 해석, "윈도우 플랫폼을 목표로 하는" 또는 "대상으로 하는" 이 더 맞을 거 같은데..
- 7. The malware reportedly has been used in 이라는 문장이 "사용되어 왔다고 알려 졌다"로 해석되는데, 보고서 상에서 문장 해석에서는 단순히 알려진 거 라기보다는 "보고에 의하면, 사용 되어 왔다" 라고 명확하게 쓰는 것이 나을 것 같기도 합니다.
- 8. its 라는 대명사를 거의 해석하지 않고 뛰어 넘고 있는데, 이것을 "자신의" 정도로 해석이 필요하다고 봐요~~!!

번역모델 개발

- 번역모델은 요약모델과 똑같다! 데이터만 차이
- OpenNMT 기반 Transformer 사용
- BERT+Transformer는 너무 느리다
 - 번역은 여러 문장을 한번에 처리해야 해서 성능보다 속도에 중점
- Shared Vocab을 사용하려면 BERT를 쓸 수 없다
- 가벼운 모델로 최대 성능을 끌어내기 위한 전처리&학습 기법 개발

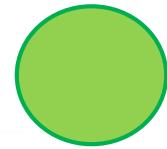
Summary (Sentences): 개인정보 유출 추정 고객들에게 이메일과 문자로 유출사고 안내 및 비밀번호 변경 요청해



target: 북한이 남한에 또 사이버 공격을 했다

Document (News): “또한, 개인정보가 유출된 것으로 추정되는 고객에게는 이메일과 문자 메시지로 개인정보 유출사고에 관해 안내했다고 설명하면서, 안내를 받은 고객들은 회원계정의 안전한 보호를 위해 가급적 비밀번호를 변경해 달라고 당부했다.”

번역모델 개발



도메인 데이터

■ 문장 분절에 따른 실험결과 비교

- Share-Sentencepiece 인 경우 영어 코퍼스 + 한글 코퍼스

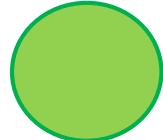
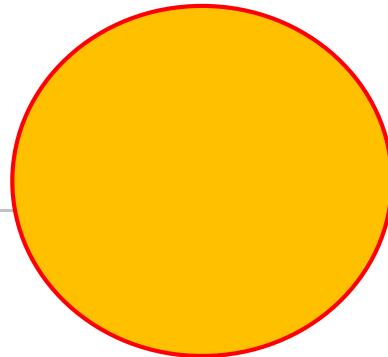
일반 데이터

Experiment	IWSLT 16	IWSLT 17
Word	7.98	7.16
char	16.39	17.06
BPE	20.25	17.84
ShareBPE	17.47	15.42
ShareSentencepiece	21.63	19.11
ko-MECAB + en-sentencepiece	19.61	17.08
ko-MECAB&Sentencepiece + en-sentencepiece	19.78	17.49
ShareSentencepiece(ko-MECAB + en)	24.95	22.58

조사	어미의 변형		
(a) 한국어	(b) 영어	(c) 한국어	(d) 영어
핑퐁은	Pingpong is	먹다	eat
핑퐁이	Pingpong does	먹는다	to eat
핑퐁을	Pingpong have	먹고	eating
핑퐁에게	to Pingpong	먹어서	ate
핑퐁한테	of Pingpong	먹어	eaten
핑퐁의	with Pingpong	먹어라	have eaten
핑퐁과		먹는데	have eating
핑퐁이랑		먹지만	had eaten

#	Origin Sentence	SentencePiece Tokenizer	Mecab Tokenizer
1	엄청 빨리끝나는거같네	엄청 V 빨리 V 끝나 V 는 V 거 V 같 V 네	엄청 V 빨리 V 끝나 V 는 V 거 V 같 V 네
2	차량운행할때 불편하겠 다 조심해요	차량 V 운 V 행 V 할때 V 불편하 겠다 V 조심해요	차량 V 운행 V 할 V 때 V 불편 V 하 V 겠 다 V 조심 V 해요
3	출석부르기전에 들어가 면 되지않을까	출석 V 부르 V 기전에 V 들어가 면 V 되지 V 않 V 을까	출석 V 부르 V 기 V 전 V 에 V 들어가 V 면 V 되 V 지 V 않 V 을까
4	잘못될까무섭고	잘못 V 될 V 까 V 무 V 섭 V 고	잘못 V 될까 V 무섭 V 고
5	피스타치오향 약간 새 로운맛이야	피 V 스타 V 치 V 오 V 향 V 약 간 V 새로운 V 맛이야	피스타치오 V 향 V 약간 V 새로운 V 맛 V 이 V 야
6	후리스따뜻해?	후리스 V 따뜻해 V ?	후 V 리스 V 따뜻 V 해 V ?
7	응 당직은 칼퇴	응 V 당직 V 은 V 칼퇴	응 V 당직 V 은 V 칼 V 퇴
8	자꾸자꾸 심쿵하네	자꾸자꾸 V 심쿵 V 하네	자꾸 V 자꾸 V 심 V 쿵 V 하 V 네
9	칭구들이랑 가죠 뭐	칭구들이랑 V 가죠 V 뭐	칭 V 구 V 들 V 이랑 V 가 V 죠 V 뭐

번역모델 개발



도메인 데이터

일반 데이터

- Domain-specific NMT 학습방법
 - 데이터를 합쳐서 학습
 - 단순히 general + domain을 합쳐서 데이터로 사용
 - 데이터의 비율을 달리 하여 학습
 - 한 배치 안에서 general 코퍼스와 Domain-specific 코퍼스의 비율을 달리 구성하여 학습
 - 추가 학습(Continual-learning)
 - general 데이터로 학습된 모델에 domain 데이터로 추가 학습

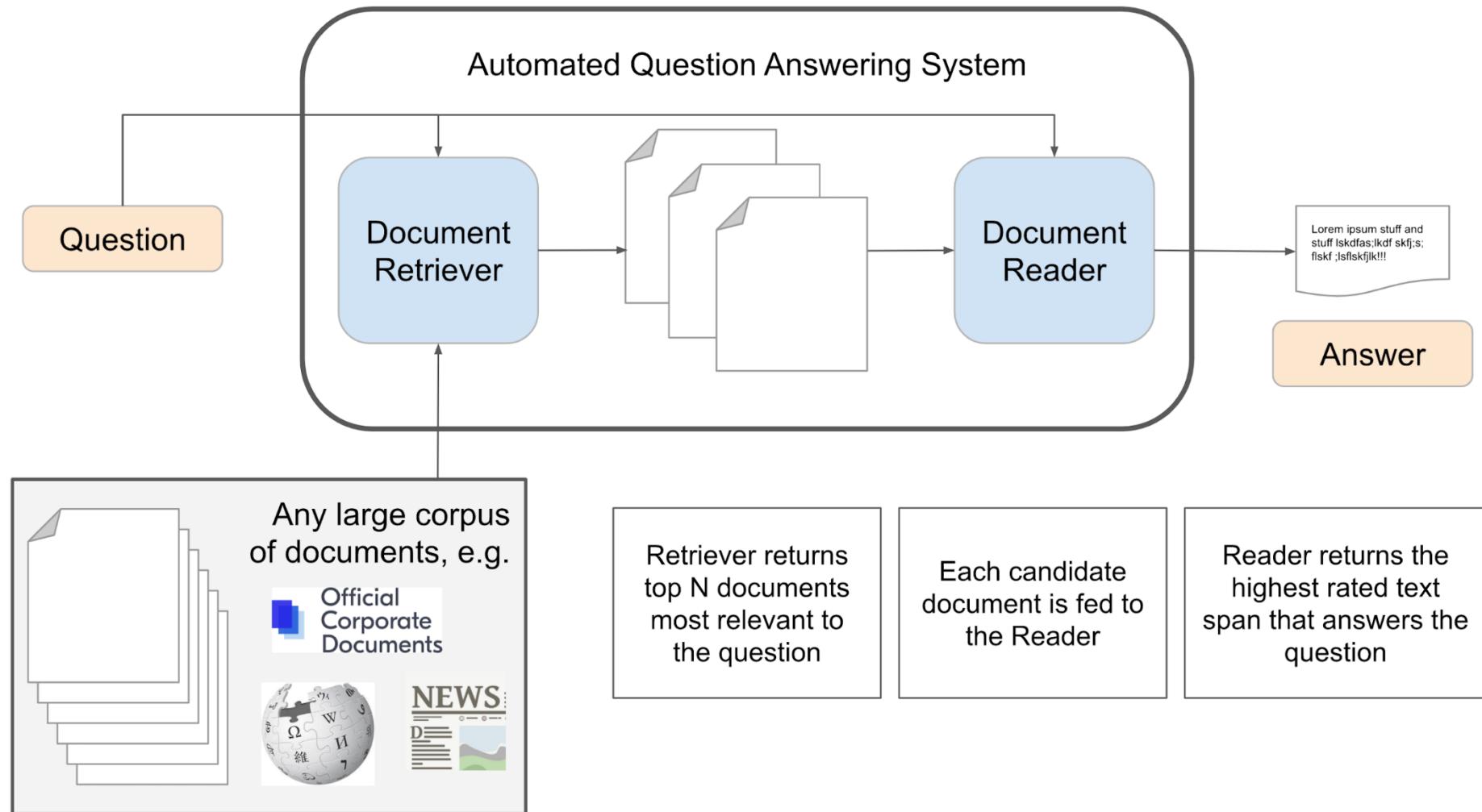
General-Domain : Specific-Domain	BLEU
Combine Augmentation (220만+2000)	64.64
상대적 비율 (1:1)	60.46
상대적 비율 (2:1)	64.91
상대적 비율 (1:2)	55.38
상대적 비율 (3:1)	65.27
상대적 비율 (4:1)	66.02
상대적 비율 (5:1)	65.14
상대적 비율 (10:1)	67.91
상대적 비율 (30:1)	68.41
상대적 비율 (50:1)	68.72
상대적 비율 (100:1)	68.92

Domain-specific test	Domain test set (687)
ShareSentencepiece(ko-MECAB + en)	27.06
Incremental Training / Re-training	
(+) 2000개로 파인튜닝 + 2000 step	69.19
(+) 2000개로 파인튜닝 + 4000 step	69.13
(+) 2000개로 파인튜닝 + 6000 step	68.63
(+) 2000개로 파인튜닝 + 8000 step	68.73
(+) 2000개로 파인튜닝 + 10000 step	68.09
(+) 2000개로 파인튜닝 + 20000 step	67.34
(+) 2000개로 파인튜닝 + 30000 step	66.88

감사합니다

ODQA

Open-Domain Question Answering



MRC

- 주어진 문서에서 질문의 답을 찾는 문제
- Input: text
- Output: span(start, end)
 - Ex) (15, 24)
- Token classification model

In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under **gravity**. The main forms of precipitation include drizzle, rain, sleet, snow, **graupel** and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals **within a cloud**. Short, intense periods of rain in scattered locations are called "showers".

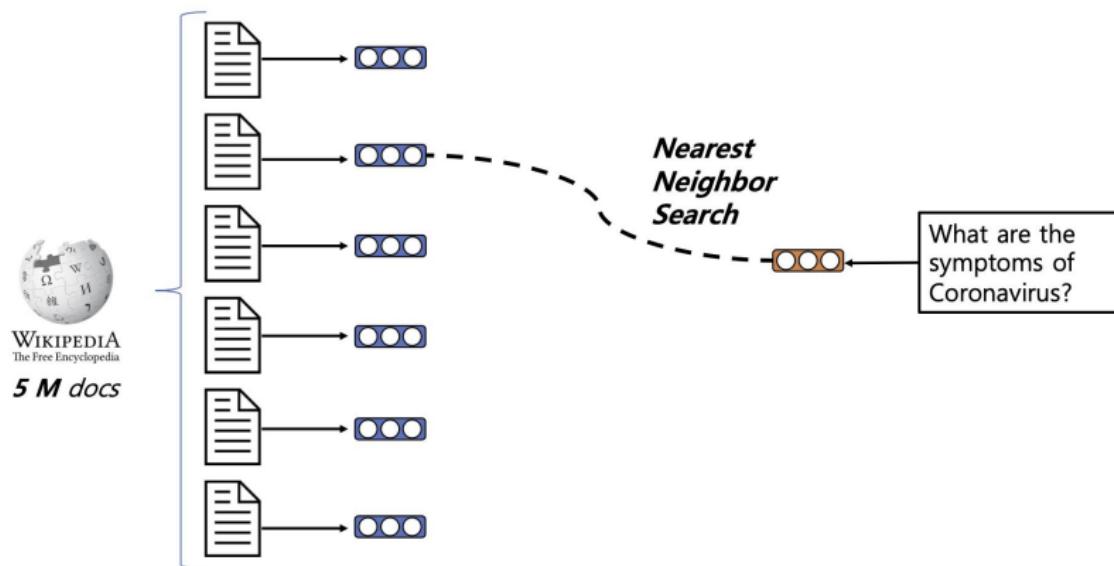
What causes precipitation to fall?
gravity

What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?
graupel

Where do water droplets collide with ice crystals to form precipitation?
within a cloud

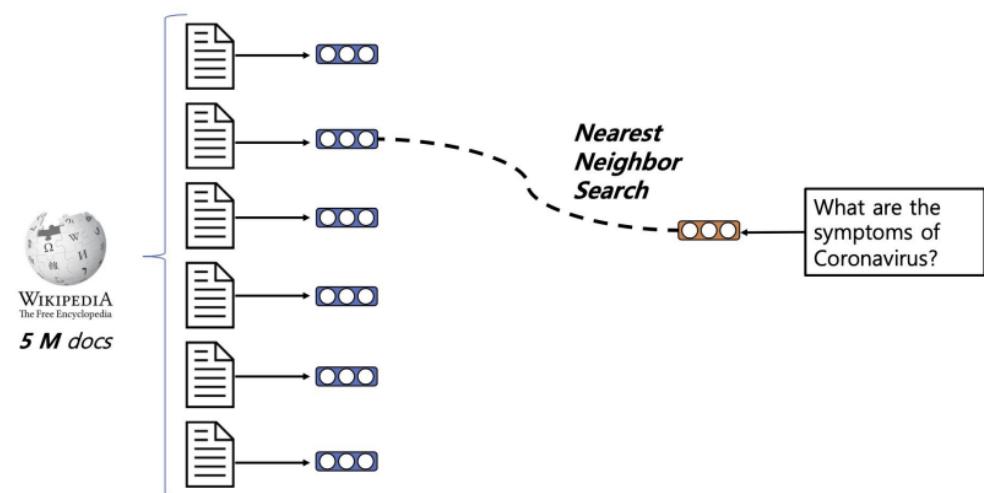
ODQA

- 전체 DB에서 후보 문서를 찾고, 그 문서에서 질문의 답을 찾는 문제
- Input: DB
- Output: span(start, end)
 - Ex) (15, 24)
- Document retrieval & Token classification model

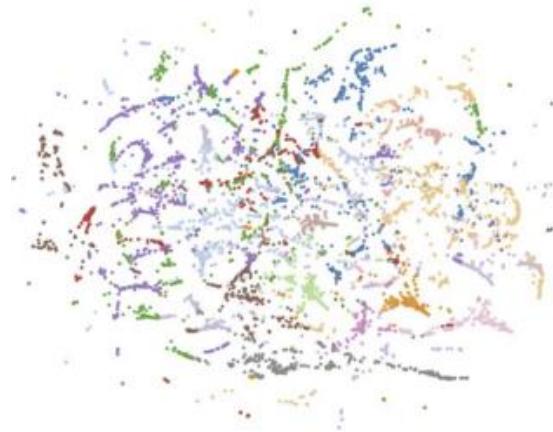


Document retrieval

- 전체 DB에서 후보 문서 찾기
- Input: DB
- Output: Document ranking idx
 - Ex) [4, 17, 6778, 11523, ...]



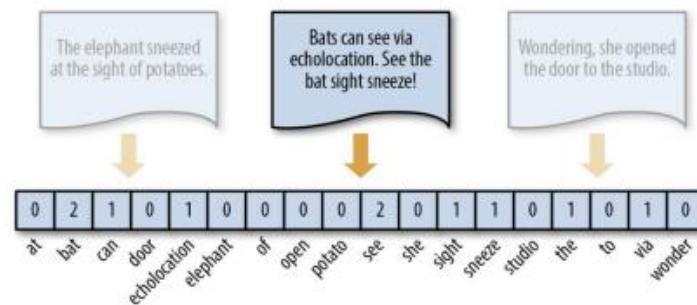
Sparse Retrieval



- **Good** for capturing **syntactic** and **semantic** information
- **Difficult** to encode precise **lexical** information

TF-IDF, BM25

Dense Retrieval

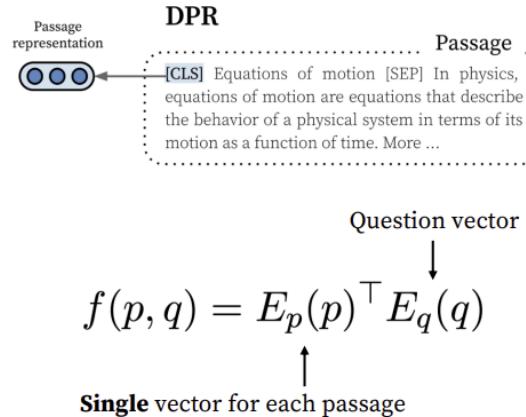


- **Good** for capturing **lexical** information
- **Difficult** to encode **syntactic** or **semantic** information

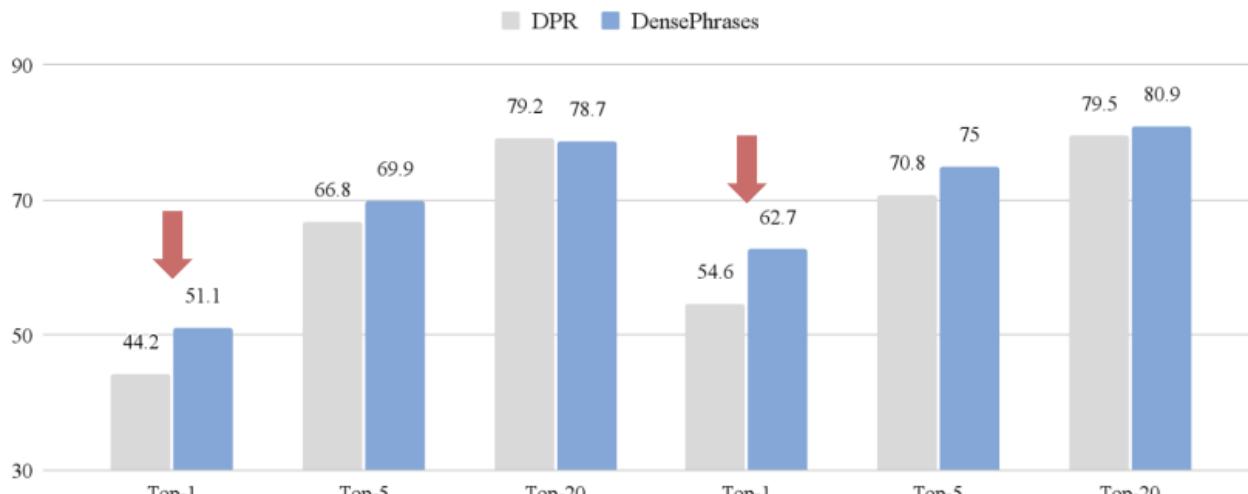
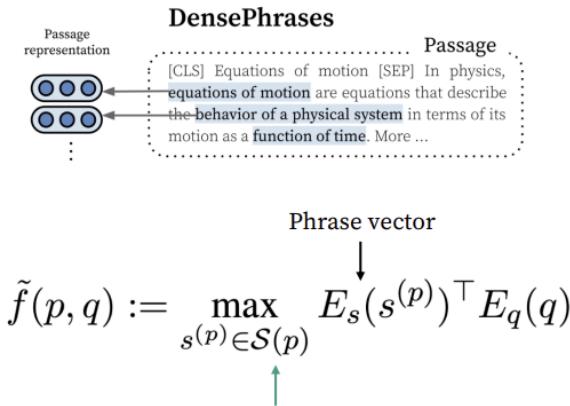
DPR, Phraset Retrieval

Document retrieval

Passage Retrieval



Phrase-based Passage Retrieval



BERT Code Review

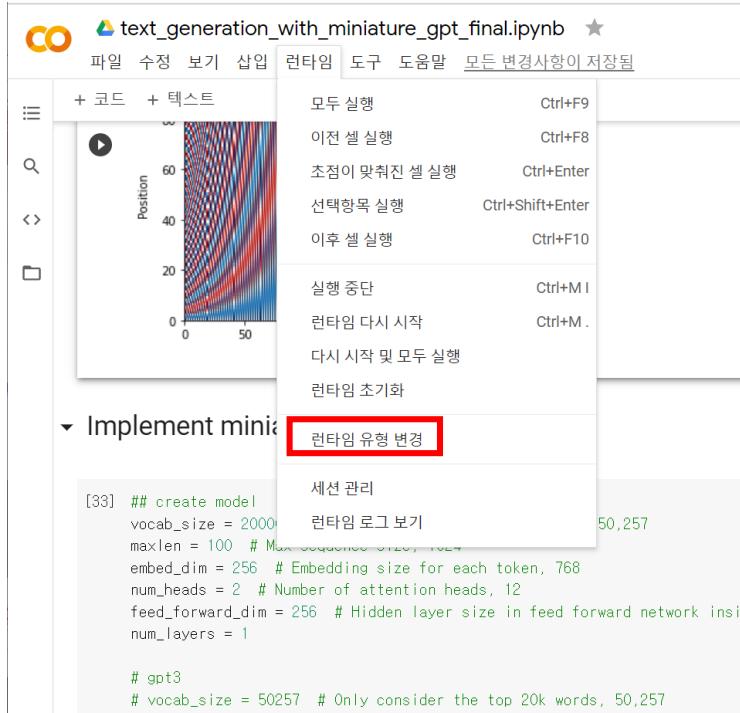
Keras Learning Day: <https://youtu.be/AS-7fQiV6LE>

Colab 설정

- 인터넷에서 아래 URL로 접근

- https://colab.research.google.com/drive/1aBXnwBMukb_AcDh7aBFjQY9SoNwnOxfh?usp=sharing

- 런타임->런타임 유형변경->GPU 설정



text_generation_with_miniature_gpt_final.ipynb

파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

+ 코드 + 텍스트

모두 실행 Ctrl+F9

이전 셀 실행 Ctrl+F8

초점이 맞춰진 셀 실행 Ctrl+Enter

선택항목 실행 Ctrl+Shift+Enter

이후 셀 실행 Ctrl+T0

실행 중단 Ctrl+M I

런타임 다시 시작 Ctrl+M .

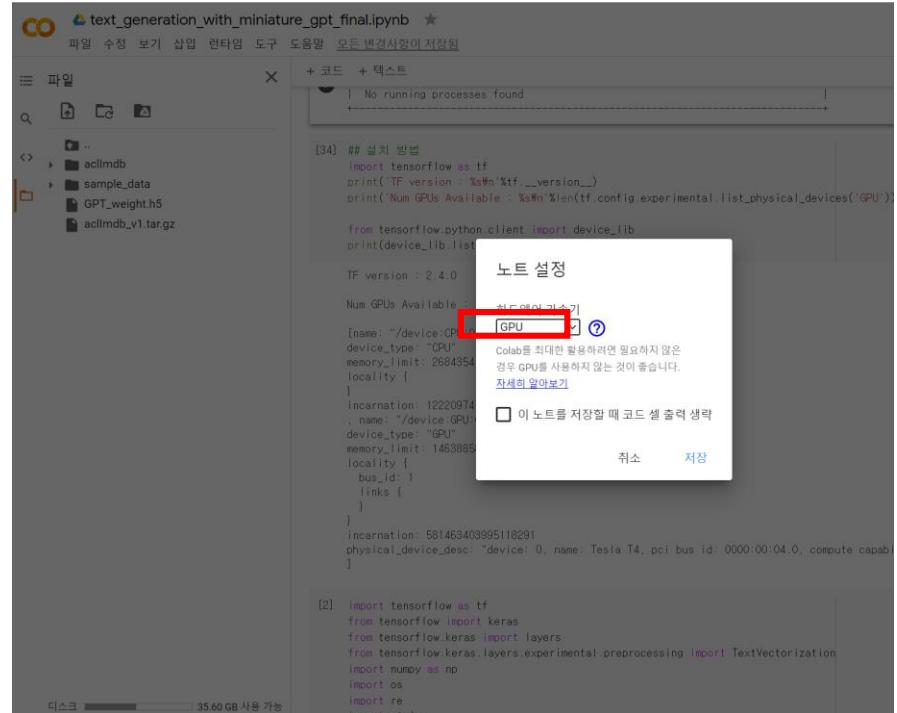
다시 시작 및 모두 실행

런타임 초기화

런타임 유형 변경

```
[33] ## create model
vocab_size = 2000
maxlen = 100 # Max sequence size, 1024
embed_dim = 256 # Embedding size for each token, 768
num_heads = 2 # Number of attention heads, 12
feed_forward_dim = 256 # Hidden layer size in feed forward network inside
num_layers = 1

# gpt3
# vocab_size = 50257 # Only consider the top 20k words, 50,257
```



text_generation_with_miniature_gpt_final.ipynb

파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

+ 코드 + 텍스트

No running processes found

```
[34] ## 설치 방법
import tensorflow as tf
print('TF version : %s' % tf.__version__)
print('Num GPUs Available: %s' % len(tf.config.experimental.list_physical_devices('GPU')))

from tensorflow.python.client import device_lib
print(device_lib.list_local_devices())
```

TF version : 2.4.0

Num GPUs Available: 1

[name: "/device:CPU:0", device_type: "CPU", memory_limit: 2684954, locality: {}, incarnation: 12220974, name: "/device:GPU:0", device_type: "GPU", memory_limit: 1463889, locality: {bus_id: 1, links: []}], incarnation: 581463403995110291, physical_device_desc: "device: 0, name: Tesla T4, pci bus id: 0000:00:04.0, compute capability: 7.5"

[2] import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.layers.experimental.preprocessing import TextVectorization
import numpy as np
import os
import re
import string

노트 설정

코드에서 GPU 사용 [GPU]

Colab은 최대한 활용하려면 필요하지 않은 경우 GPU를 사용하지 않는 것이 좋습니다. 자세히 알아보기

이 노트를 저장할 때 코드 셀 출력 생략

취소 저장

Colab 설정

- >> !nvidia-smi 명령어로 GPU 설정 상태 확인
- Tesla T4(16G), 400만원 상당의 GPU 12시간 동안 사용 가능!!

```
▶ !nvidia-smi
```

Fri Dec 18 07:22:39 2020

NVIDIA-SMI 455.45.01			Driver Version: 418.67		CUDA Version: 10.1		
GPU	Name	Persistence-M	Bus-Id	Disp.A	Volatile	Uncorr. ECC	
Fan	Temp	Perf	Pwr:Usage/Cap	Memory-Usage	GPU-Util	Compute M.	
					MIG M.		
0	Tesla T4	Off	00000000:00:04.0	Off	0		
N/A	78C	P0	33W / 70W	14251MiB / 15079MiB	0%	Default	
						ERR!	

+

Processes:					
GPU	GI	CI	PID	Type	Process name
ID			ID		GPU Memory Usage
No running processes found					



Dell 16GB NVIDIA Tesla T4 GPU 그래픽 카드

NVIDIA Tesla GPU 가속기로 요구 사양이 높은 HPC, 하이퍼스케일, 엔터프라이즈 데이터 센터 작업을 가속화하세요.

이제 아티지 탑사에서 머신 러닝에 이르기까지, 다양한 분야의 과학자들이 GPU를 사용하는 것보다 빠른 속도로 페터바이트 단위의 데이터를 조사할 수 있습니다. 또한 Tesla 가속기는 다양한 시뮬레이션을 이전보다 빠른 속도로 실행할 수 있는 성능을 제공합니다. VDI를 배포한 기업에게 있어 Tesla 가속기는 모든 사용자에게 어디서 ... [더 보기](#)

3,969,636 원 773,600 원
할인 3,813,964 원
10% 부가세 포함

[장바구니에 담기](#)

BERT-code review

- Tensorflow 2.8 버전 사용 >> pip install tensorflow==2.8.0
- Colab에 기본적으로 설치되어 있음 -> 설치 필요 X

```
## TF 버전 확인
import tensorflow as tf
print('TF version : %s\n'%tf.__version__)
print('Num GPUs Available : %s\n'%len(tf.config.experimental.list_physical_devices('GPU')))

from tensorflow.python.client import device_lib
print(device_lib.list_local_devices())

TF version : 2.8.0
Num GPUs Available : 1

[name: "/device:CPU:0"
device_type: "CPU"
memory_limit: 268435456
locality {
}
incarnation: 9241766196942243098
xla_global_id: -1
, name: "/device:GPU:0"
device_type: "GPU"
memory_limit: 14465892352
locality {
  bus_id: 1
  links {
  }
}
incarnation: 17601612635952061443
physical_device_desc: "device: 0, name: Tesla T4, pci bus id: 0000:00:04.0, compute capability: 7.5"
xla_global_id: 416903419
]

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.layers.experimental.preprocessing import TextVectorization
import numpy as np
import os
import re
import string
import random
```

- tf-2.8.0 버전 체크
- 2.8버전이 아니라면 아래 명령어로 설치
 - pip install tensorflow==2.8.0
- Tf2.8의 내장 MultiHeadAttention 함수가 import 되면 OK
- 각종 library import

BERT-code review

Load the data

We will first download the IMDB data and load into a Pandas dataframe.

```
%%time
!curl -O https://ai.stanford.edu/~amaas/data/sentiment/aclImdb_v1.tar.gz
!tar -xf aclImdb_v1.tar.gz

def get_text_list_from_files(files):
    text_list = []
    for name in files:
        with open(name, encoding='UTF-8') as f:
            for line in f:
                text_list.append(line)
    return text_list

def get_data_from_text_files(folder_name):
    pos_files = glob.glob("aclImdb/" + folder_name + "/pos/*.txt")
    pos_texts = get_text_list_from_files(pos_files)
    neg_files = glob.glob("aclImdb/" + folder_name + "/neg/*.txt")
    neg_texts = get_text_list_from_files(neg_files)
    df = pd.DataFrame(
    {
        "review": pos_texts + neg_texts,
        "sentiment": [0] * len(pos_texts) + [1] * len(neg_texts),
    })
    df = df.sample(len(df)).reset_index(drop=True)
    return df

train_df = get_data_from_text_files("train")
test_df = get_data_from_text_files("test")

all_data = train_df.append(test_df)

all_data.head()
```

- **Imdb 데이터셋 다운로드(80MB)**

- **압축해제 (210MB)**



- **데이터 load&처리**

	review	sentiment
0	Well, if you are open-minded enough to have li...	0
1	If another Hitler ever arises, it will be than...	1
2	This show is wonderful. It has some of the bes...	0
3	This is one of the funniest and most excellent...	0
4	This kid is rather bad, but in no way do they ...	1 232

Setup configuration

```
@dataclass
class Config:
    MAX_LEN = 256
    BATCH_SIZE = 32
    LR = 0.001
    VOCAB_SIZE = 30000
    EMBED_DIM = 128
    NUM_HEAD = 8 # used in bert model
    FF_DIM = 128 # used in bert model
    NUM_LAYERS = 1

config = Config()
```

#L	#H	#A	LM (ppl)	Hyperparams		Dev Set Accuracy	
				MNLI-m	MRPC	SST-2	
▪ 각종 파	3	768	12	5.84	77.9	79.8	88.4
	6	768	3	5.24	80.6	82.2	90.7
	6	768	12	4.68	81.9	84.8	91.3
	12	768	12	3.99	84.4	86.7	92.9
	12	1024	16	3.54	85.7	86.9	93.3
	24	1024	16	3.23	86.6	87.8	93.7

▪ MAX_LEN

- 처리하는 문장의 최대길이, BERT는 512로 제한
- 최대 고정 길이 지정으로 BERT의 한계

▪ VOCAB_SIZE

- 한 언어에 32000정도가 가장 좋다고 알려짐
- WordPiece로 분절된 Subword tokenization 방식 사용

▪ EMBED_DIM

- Token을 몇 차원으로 나타낼지 설정

▪ NUM_HEAD

- Multi-head Self-Attention의 head 개수

▪ FF_DIM

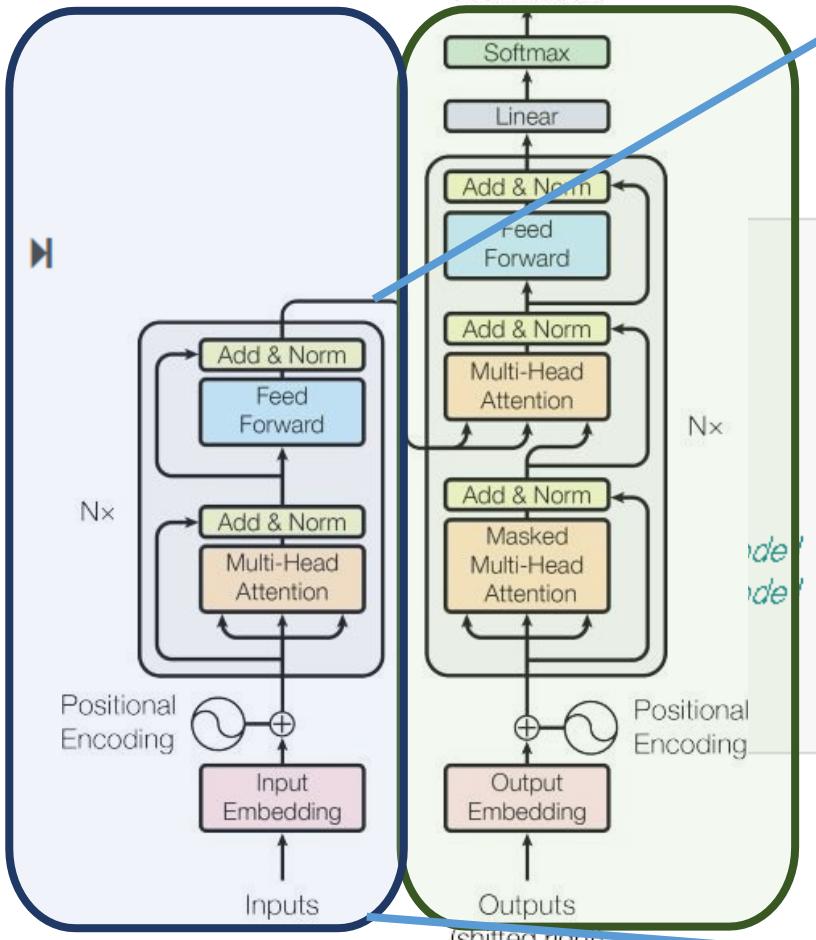
- Multi-head Self-Attention의 Feed forward dimension

▪ NUM_LAYERS

- BERT Encoder의 Transformer block layer 32 몇 층 쌓을지 설정

Setup configuration

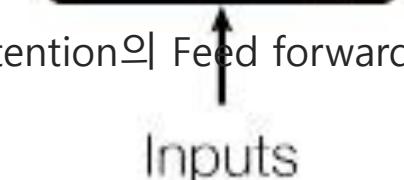
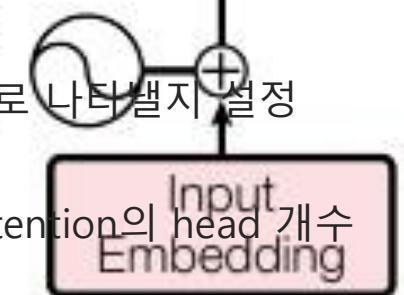
Encoder



Decoder

- 각종 파라미터 정의
- **MAX_LEN** $N \times$
 - 처리하는 문장의 최대길이, BERT는 512로 제한
 - 최대 고정 길이 지정으로 BERT의 한계
- **Self-attention block**
 - VOCAB_SIZE
 - 한 언어에 32000정도가 가장 좋다고 알려짐
 - WordPiece로 본질된 Subword tokenization 방식 사용
- **EMBED_DIM**
 - Token을 몇 차원으로 나타낼지 설정
- **NUM_HEAD**
 - Multi-head Self-Attention의 head 개수
- **FF_DIM**
 - Multi-head Self-Attention의 Feed forward dimension
- **NUM_LAYERS**
 - BERT Encoder의 Transformer block layer 12가 몇 층 쌓을지 설정

Positional
Embedding



Setup configuration

- 각종 파라미터 정의

```

@dataclass
class Config:
    MAX_LEN = 256
    BATCH_SIZE = 32
    LR = 0.001
    VOCAB_SIZE = 30000
    EMBED_DIM = 128
    NUM_HEAD = 8 # used in bert model
    FF_DIM = 128 # used in bert model
    NUM_LAYERS = 1

config = Config()

```

- MAX_LEN

- 처리하는 문장의 최대 고정 길이
- 최대 고정 길이 지정으로 bert는 한 문장

- VOCAB_SIZE

- 한 언어에 32000정도가 가장 좋다고 알려짐
- WordPiece로 분절된 Subword tokenization 방식 사용

- EMBED_DIM

- Token을 몇 차원으로 나타낼지 설정

- NUM_HEAD

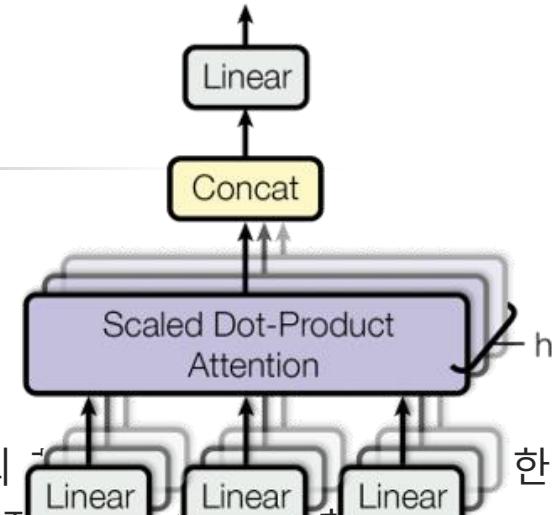
- Multi-head Self-Attention의 head 개수

- FF_DIM

- Multi-head Self-Attention의 Feed forward dimension

- NUM_LAYERS

- BERT Encoder의 Transformer block layer 32개 몇 층 쌓을지 설정



Dataset Preparation -1

▪ `get_vectorize_layer` - TextVectorization

- Text를 Vector로 변환하여 입력형식으로 만들어주는 모듈

```
def custom_standardization(input_data):  
    lowercase = tf.strings.lower(input_data) # 소문자화  
    stripped_html = tf.strings.regex_replace(lowercase, "<br />", " ") # html 제거  
    # 특수문자 제거  
    return tf.strings.regex_replace(stripped_html, "[%s]" % re.escape("!#$%&`()*+,-./;:<=>?@#!^_`{|}~")  
  
def get_vectorize_layer(texts, vocab_size, max_seq, special_tokens=["[MASK]"]):  
    """Build Text vectorization layer  
  
    Args:  
        texts (list): List of string i.e input texts  
        vocab_size (int): vocab size  
        max_seq (int): Maximum sequence lenght.  
        special_tokens (list, optional): List of special tokens. Defaults to ['[MASK]'].  
  
    Returns:  
        layers.Layer: Return TextVectorization Keras Layer  
    """  
    # 텍스트를 벡터화 하는 함수  
    # ex) ['i have a dream'] -> [10, 25, 4, 1040]  
    vectorize_layer = TextVectorization(  
        max_tokens=vocab_size,  
        output_mode="int",  
        standardize=custom_standardization, # 따로 정의한 정제함수  
        output_sequence_length=max_seq,  
    )  
    vectorize_layer.adapt(texts) # texts 적용  
  
    # Insert mask token in vocabulary  
    vocab = vectorize_layer.get_vocabulary()  
    vocab = vocab[2 : vocab_size - len(special_tokens)] + ["[mask]"] # [mask] token 추가  
    vectorize_layer.set_vocabulary(vocab)  
    return vectorize_layer
```

Dataset Preparation -1

▪ **get_vectorize_layer - TextVectorization**

- Text를 Vector로 변환하여 입력형식으로 만들어주는 모듈

```
vectorize_layer = get_vectorize_layer(all_data.review.values.tolist(),
                                       config.VOCAB_SIZE,
                                       config.MAX_LEN,
                                       special_tokens=["[mask]"])
```

```
# Get mask token id for masked language model
mask_token_id = vectorize_layer(["[mask]"]).numpy()[0][0]
print('[mask] token id : %s' % mask_token_id)
print(vectorize_layer(['i have a dream']))
```

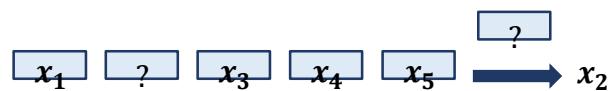
[mask] token id : 29999

tf.Tensor([[10 25 4 1040 0]], shape=(1, 5), dtype=int64)

Dataset Preparation -2. MLM 학습데이터 생성

- MLM : 15%의 [MASK] token을 생성

Auto Encoding



입력 문장

$$\bar{x} = [x_1, x_2, x_3, x_4, x_5]$$

오염된 문장

$$\hat{x} = [x_1, [MASK], x_3, x_4, x_5]$$

likelihood

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

Dataset Preparation -2. MLM 학습데이터 생성

■ MLM : 15%의 [MASK] token을 생성

```
encoded_texts
```

```
array([[ 5, 259, 10, 57, 25],  
       [ 56, 11, 277, 28, 5],  
       [ 1, 1596, 3, 2, 188]], dtype=int64)
```



the man went to the [MASK] to buy a [MASK] of milk

```
# 15% BERT masking  
inp_mask = np.random.rand(*encoded_texts.shape) < 0.15  
# Do not mask special tokens  
inp_mask[encoded_texts <= 2] = False  
print(inp_mask)
```

```
[[False True False True False]  
 [ True False False False True]  
 [False False False False False]]
```

```
# Set targets to -1 by default, it means ignore  
labels = -1 * np.ones(encoded_texts.shape, dtype=int)  
print(labels)
```

```
[[ -1 -1 -1 -1 -1]  
 [ -1 -1 -1 -1 -1]  
 [ -1 -1 -1 -1 -1]]
```

```
# Set labels for masked tokens  
labels[inp_mask] = encoded_texts[inp_mask]  
print(labels)
```

```
[[ -1 259 -1 57 -1]  
 [ 56 -1 -1 -1 5]  
 [ -1 -1 -1 -1 -1]]
```

Dataset Preparation -2. MLM 학습데이터 생성

■ MLM : 15%의 [MASK] token을 생성

- 80%의 경우 : token을 [MASK]로 대체
 - eg., my dog is hairy -> my dog is [MASK]
- 10%의 경우 : token을 원래의 단어로 그대로 놔둠
 - 이는 실제 관측된 단어에 대한 표상을 bias해주기 위해 실시합니다.
 - eg., my dog is hairy -> my dog is [MASK] -> my dog is hairy

```
# Set input to [MASK] which is the last token for the 90% of tokens
# This means leaving 10% unchanged
inp_mask_2mask = inp_mask & (np.random.rand(*encoded_texts.shape) < 0.90)
encoded_texts_masked[inp_mask_2mask] = mask_token_id # mask token is the last in the dict
```

`[[False True False True False] & [[False True False True False] & [[True True False True True]
[False False False False True] = [True False False False True] & [False True True True True]
[False False False False False]] & [False False False False False]] & [True True True True True]]`

`[[5 259 10 57 25]
[56 11 277 28 5] → [[5 29999 10 29999 25]
[1 1596 3 2 188]] [56 11 277 28 29999]
[1 1596 3 2 188]]`

Dataset Preparation -2. MLM 학습데이터 생성

■ MLM : 15%의 [MASK] token을 생성

- 80%의 경우 : token을 [MASK]로 대체
 - eg., my dog is hairy -> my dog is [MASK]
- 10%의 경우 : token을 원래의 단어로 그대로 놔둠
 - 이는 실제 관측된 단어에 대한 표상을 bias해주기 위해 실시합니다.
 - eg., my dog is hairy -> my dog is [MASK] -> my dog is hairy
- 10%의 경우 : token을 random word 대체
 - eg., my dog is hariy -> my dog is apple

```
## 3. 10%는 아무 단어로 변환
```

```
# Set 10% to a random token
```

```
inp_mask_2random = inp_mask_2mask & (np.random.rand(*encoded_texts.shape) < 1 / 9)
encoded_texts_masked[inp_mask_2random] = np.random.randint(3, mask_token_id, inp_mask_2random.sum())
```

```
[[False False False True False]
 [False False False False False]
 [False False False False False]]
```

```
[[ 5 29999 10 29999 25]
 [ 56 11 277 28 29999]
 [ 1 1596 3 2 188]]
```



```
[[ 5 29999 10 22697 25]
 [ 56 11 277 28 29999]
 [ 1 1596 3 2 188]] 180/232
```

Dataset Preparation -2. MLM 학습데이터 생성

■ MLM : 15%의 [MASK] token을 생성

```
# Prepare sample_weights to pass to .fit() method
sample_weights = np.ones(labels.shape)
sample_weights[labels == -1] = 0

# y_labels would be same as encoded_texts i.e input tokens
y_labels = np.copy(encoded_texts)
```

$\begin{bmatrix} [1. 1. 1. 1. 1.] \\ [1. 1. 1. 1. 1.] \\ [1. 1. 1. 1. 1.] \end{bmatrix}$ & $\begin{bmatrix} [-1 \textcolor{blue}{259} -1 \textcolor{blue}{57} -1] \\ [\textcolor{blue}{56} -1 -1 -1 \textcolor{blue}{5}] \\ [-1 -1 -1 -1 -1] \end{bmatrix}$ \Rightarrow $\begin{bmatrix} [0. \textcolor{blue}{1.} 0. \textcolor{blue}{1.} 0.] \\ [\textcolor{blue}{1.} 0. 0. 0. \textcolor{blue}{1.}] \\ [0. 0. 0. 0. 0.] \end{bmatrix}$

$\begin{bmatrix} [5 \textcolor{blue}{29999} 10 \textcolor{green}{22697} 25] \\ [\textcolor{red}{56} 11 277 28 \textcolor{blue}{29999}] \\ [1 1596 3 2 188] \end{bmatrix}$

y_labels

Dataset Preparation -2. MLM 학습데이터 생성

■ MLM 입력데이터 생성

```
# Prepare data for masked language model
x_all_review = encode(all_data.review.values)
x_masked_train, y_masked_labels, sample_weights = get_masked_input_and_labels(x_all_review)

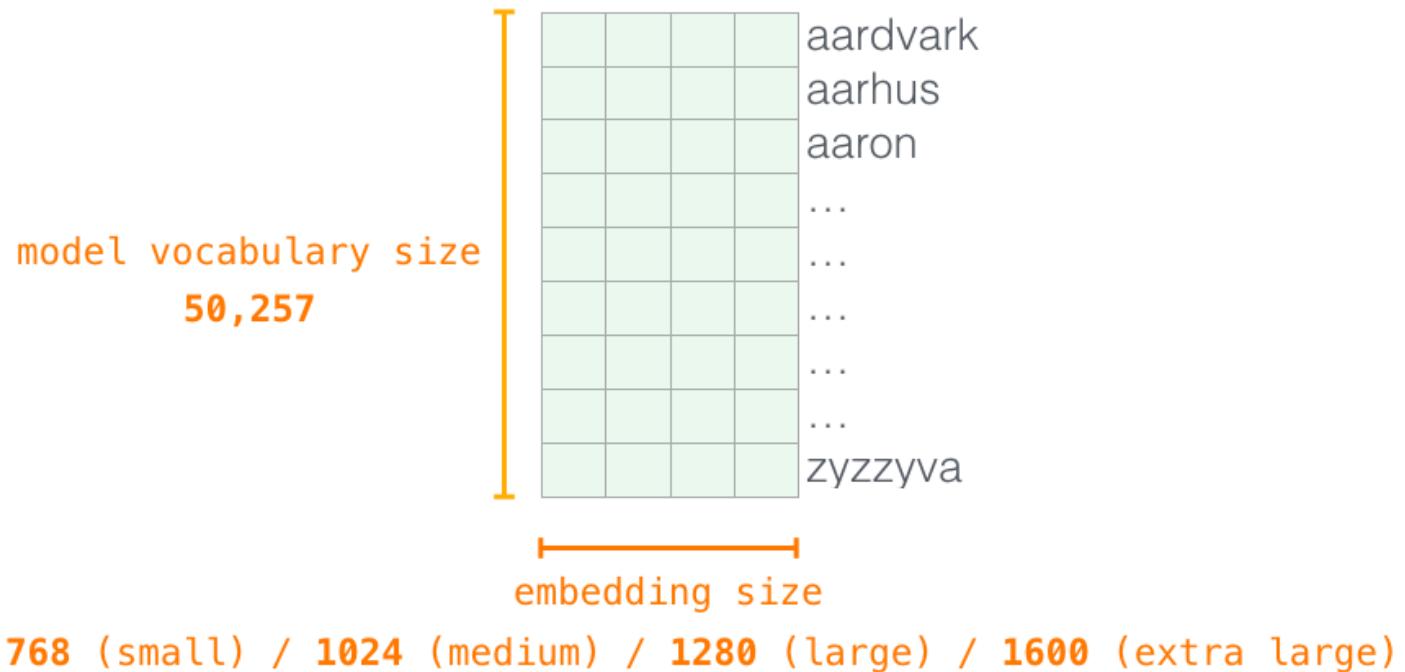
mlm_ds = tf.data.Dataset.from_tensor_slices((x_masked_train, y_masked_labels, sample_weights))
mlm_ds = mlm_ds.shuffle(1000).batch(config.BATCH_SIZE)
```

[[5 259 10 57 25] [56 11 277 28 5] [1 1596 3 2 188]]	[[5 29999 10 22697 25] [56 11 277 28 29999] [1 1596 3 2 188]]	[[-1 259 -1 57 -1] [56 -1 -1 -1 5] [-1 -1 -1 -1 -1]]	[[0. 1. 0. 1. 0.] [1. 0. 0. 0. 1.] [0. 0. 0. 0. 0.]]
x_all_review	x_masked_train	y_masked_labels	sample_weights



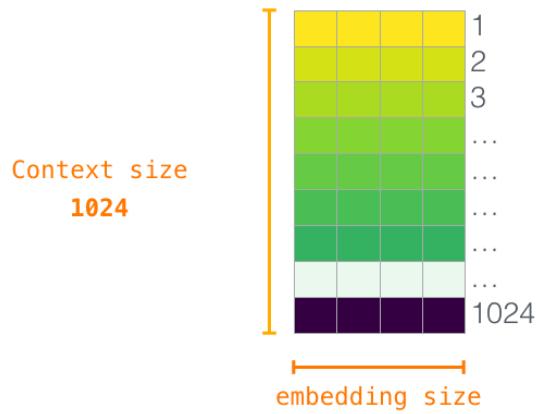
Input Embedding

- **Embedding – 1) Token Embedding**



Input Embedding

▪ Embedding – 2) Position Encoding

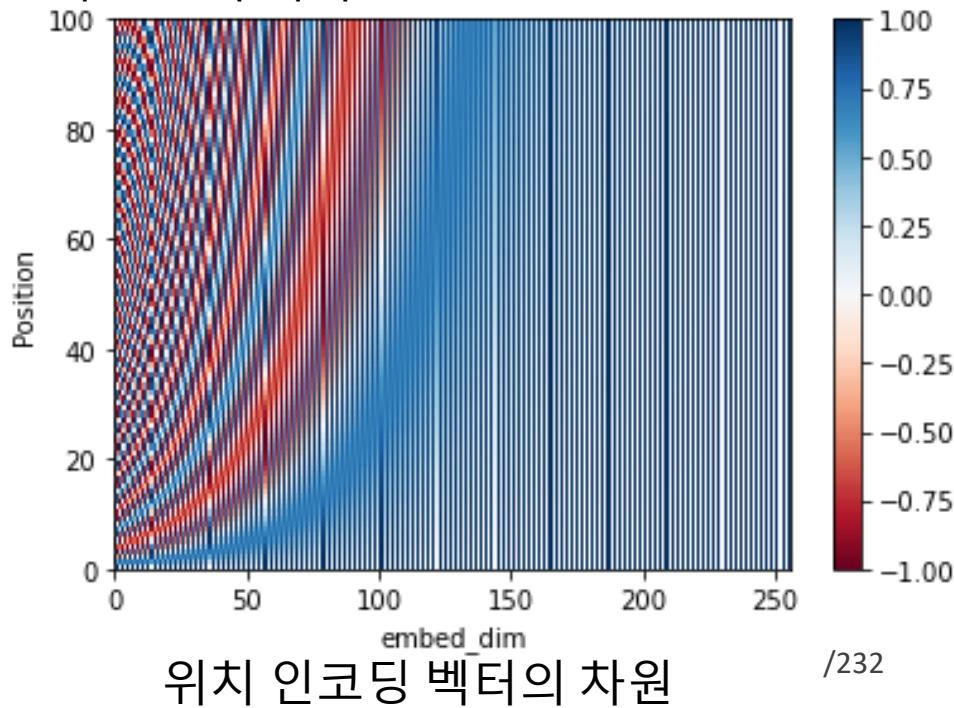


768 (small) / 1024 (medium) / 1280 (large) / 1600 (extra large)

0 :	0 0 0 0	8 :	1 0 0 0
1 :	0 0 0 1	9 :	1 0 0 1
2 :	0 0 1 0	10 :	1 0 1 0
3 :	0 0 1 1	11 :	1 0 1 1
4 :	0 1 0 0	12 :	1 1 0 0
5 :	0 1 0 1	13 :	1 1 0 1
6 :	0 1 1 0	14 :	1 1 1 0
7 :	0 1 1 1	15 :	1 1 1 1

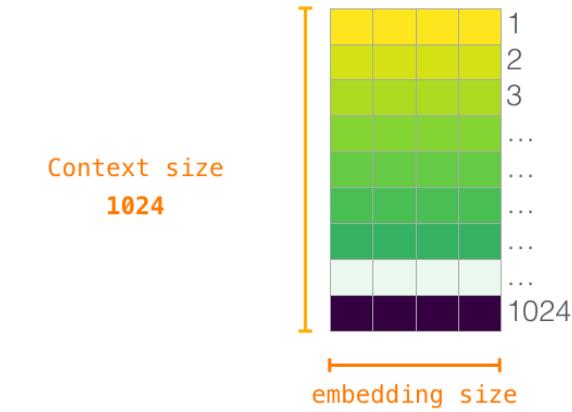
$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{\text{model}}})$$

Sequence 내 토큰의 위치 t



Input Embedding

▪ Embedding – 2) Position Encoding



768 (small) / 1024 (medium) / 1280 (large) / 1600 (extra large)

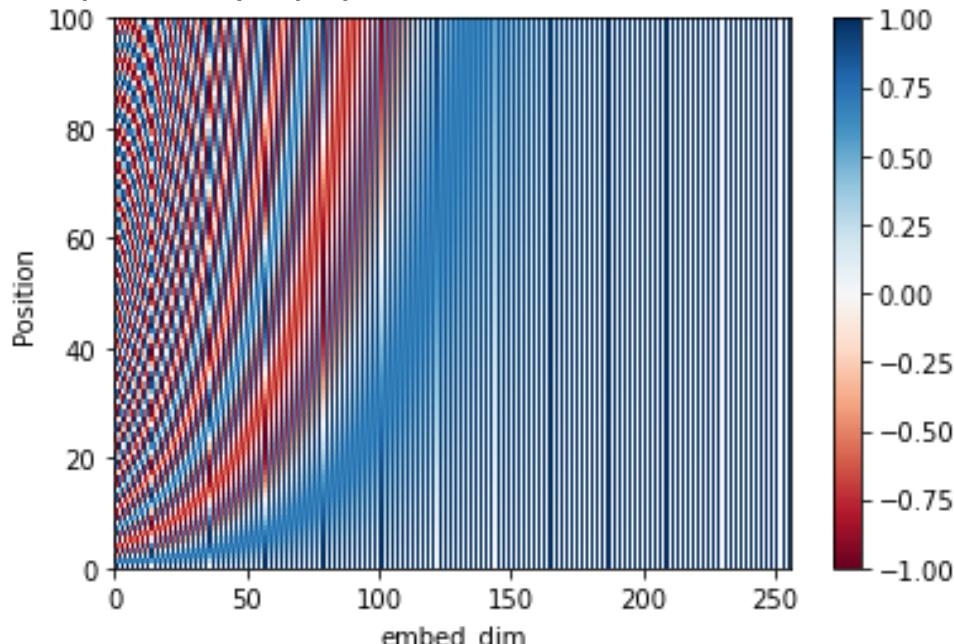
0:	0 0 0 0	8:	1 0 0 0
1:	0 0 0 1	9:	1 0 0 1
2:	0 0 1 0	10:	1 0 1 0
3:	0 0 1 1	11:	1 0 1 1
4:	0 1 0 0	12:	1 1 0 0
5:	0 1 0 1	13:	1 1 0 1
6:	0 1 1 0	14:	1 1 1 0
7:	0 1 1 1	15:	1 1 1 1

```
def get_pos_encoding_matrix(max_len, d_emb):  
    pos_enc = np.array([  
        [pos / np.power(10000, 2 * (j // 2) / d_emb) for j in range(d_emb)]  
        if pos != 0  
        else np.zeros(d_emb)  
        for pos in range(max_len)  
    ])  
    pos_enc[1::2, 0::2] = np.sin(pos_enc[1::2, 0::2]) # dim 2i  
    pos_enc[1::2, 1::2] = np.cos(pos_enc[1::2, 1::2]) # dim 2i+1  
    return pos_enc
```

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

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

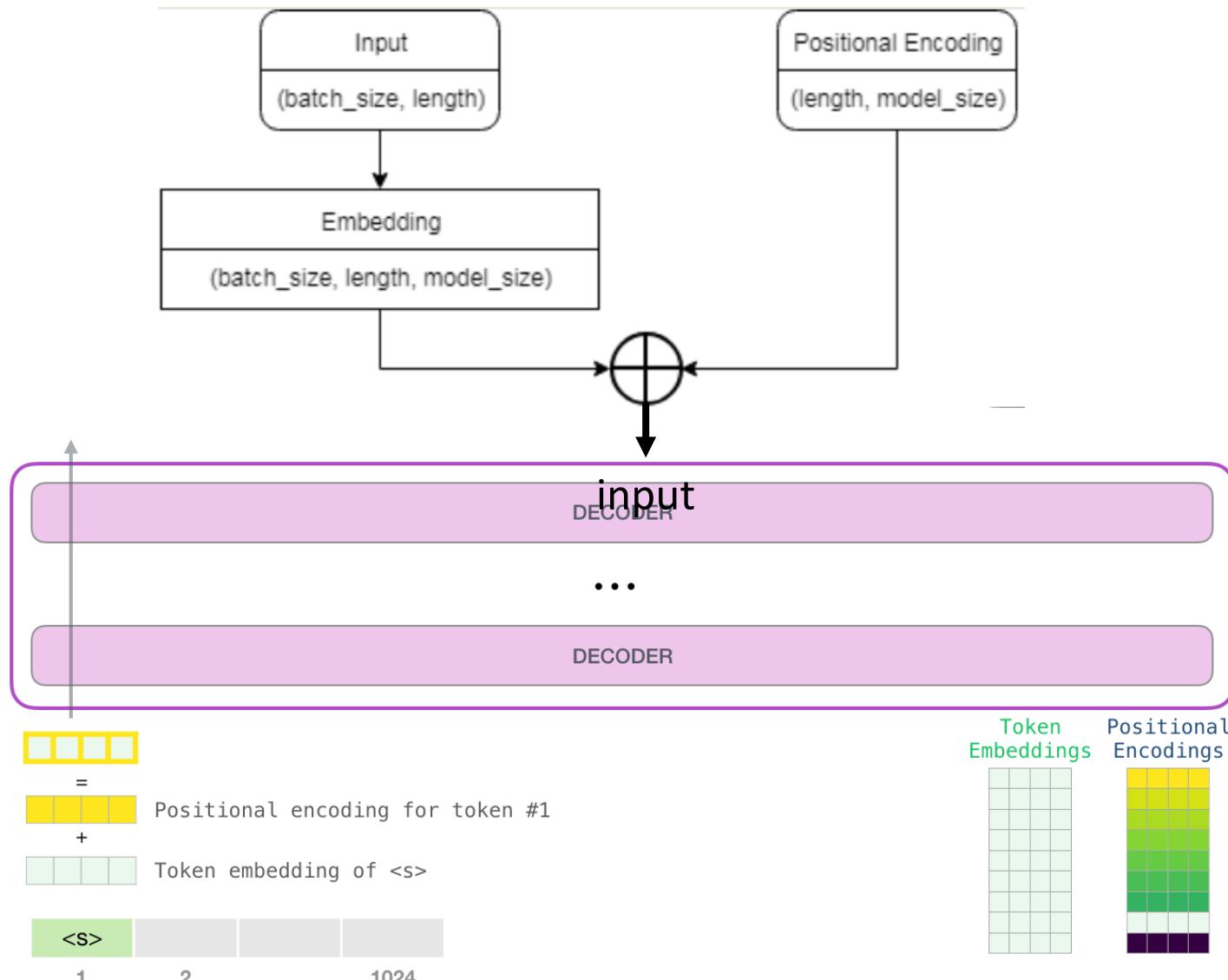
Sequence 내 토큰의 위치 t



위치 인코딩 벡터의 차원

Input Embedding

- Embedding – input : Token Embedding + Position Encoding

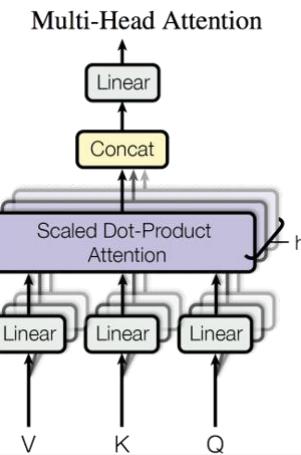
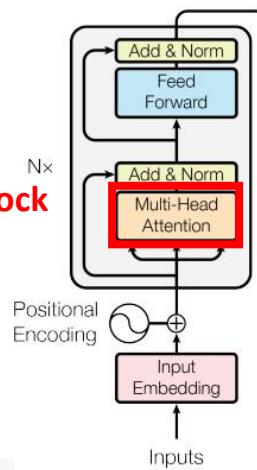


Transformer Block-1

■ MultiHeadAttention

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

Self-attention block

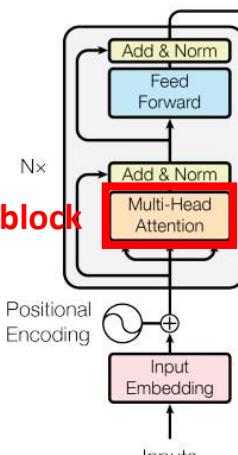


```
def Transformer_block(query, key, value, i):
    # Multi headed self-attention
    attention_output = layers.MultiHeadAttention(num_heads=config.NUM_HEAD,
                                                key_dim=config.EMBED_DIM // config.NUM_HEAD,
                                                name="encoder_{}/multiheadattention".format(i),
                                                )(query, key, value)
    attention_output = layers.Dropout(0.1, name="encoder_{}/att_dropout".format(i))(attention_output)
    attention_output = layers.LayerNormalization(epsilon=1e-6, name="encoder_{}/att_layernormali
    query + attention_output)

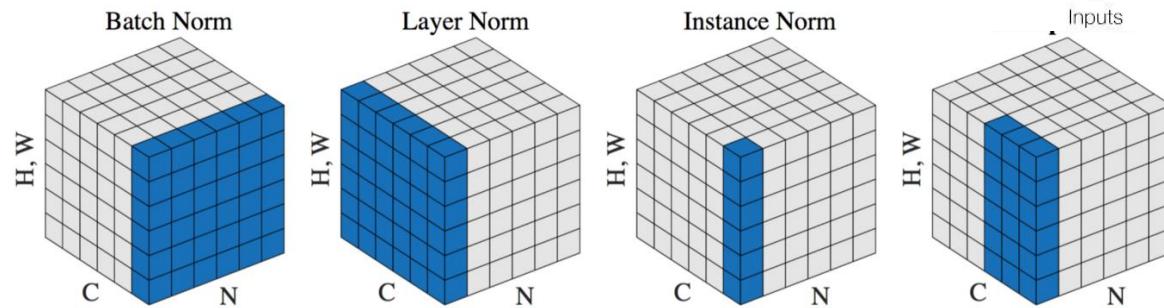
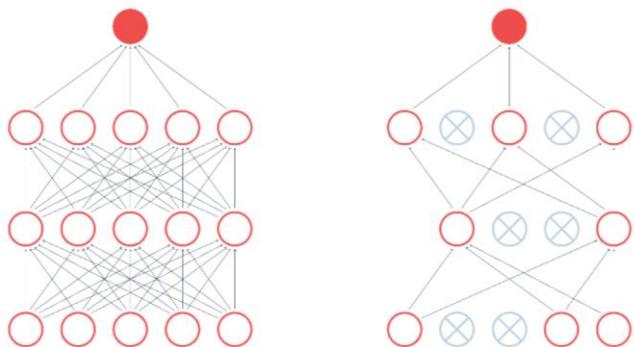
    # Feed-forward layer
    ffn = keras.Sequential([layers.Dense(config.FF_DIM, activation="relu"),
                           layers.Dense(config.EMBED_DIM),
                           name="encoder_{}/ffn".format(i)])
    ffn_output = ffn(attention_output)
    ffn_output = layers.Dropout(0.1, name="encoder_{}/ffn_dropout".format(i))(ffn_output)
    sequence_output = layers.LayerNormalization(epsilon=1e-6, name="encoder_{}/ffn_layernormaliz
    attention_output + ffn_output)
    return sequence_output
```

Transformer Block-2

Self-attention block



■ Dropout & LayerNormalization

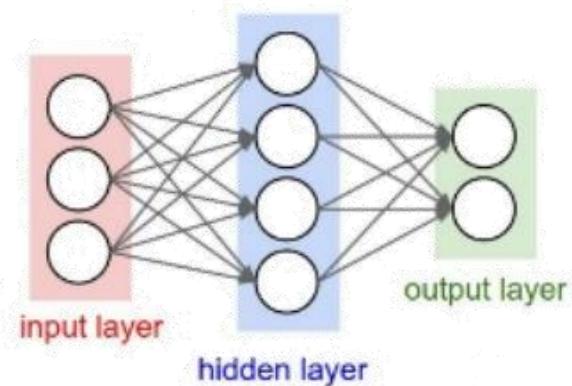


```
def Transformer_block(query, key, value, i):
    # Multi headed self-attention
    attention_output = layers.MultiHeadAttention(num_heads=config.NUM_HEAD,
                                                key_dim=config.EMBED_DIM // config.NUM_HEAD,
                                                name="encoder_{}/multiheadattention".format(i),
                                                )(query, key, value)
    attention_output = layers.Dropout(0.1, name="encoder_{}/att_dropout".format(i))(attention_output)
    attention_output = layers.LayerNormalization(epsilon=1e-6, name="encoder_{}/att_layernormali
                                                query + attention_output)

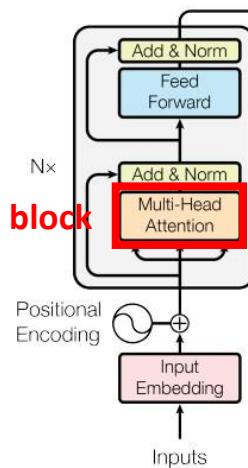
    # Feed-forward layer
    ffn = Sequential([layers.Dense(config.FF_DIM, activation="relu"),
                      layers.Dense(config.EMBED_DIM),
                      name="encoder_{}/ffn".format(i)])
    ffn_output = ffn(attention_output)
    ffn_output = layers.Dropout(0.1, name="encoder_{}/ffn_dropout".format(i))(ffn_output)
    sequence_output = layers.LayerNormalization(epsilon=1e-6, name="encoder_{}/ffn_layernormaliz
                                                attention_output + ffn_output)
    return sequence_output
```

Transformer Block-3

▪ Linear layer or Dens layer



Self-attention block

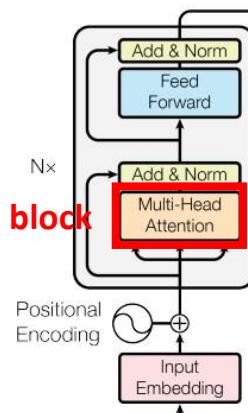


```
def Transformer_block(query, key, value, i):
    # Multi headed self-attention
    attention_output = layers.MultiHeadAttention(num_heads=config.NUM_HEAD,
                                                key_dim=config.EMBED_DIM // config.NUM_HEAD,
                                                name="encoder_{}/multiheadattention".format(i),
                                                )(query, key, value)
    attention_output = layers.Dropout(0.1, name="encoder_{}/att_dropout".format(i))(attention_output)
    attention_output = layers.LayerNormalization(epsilon=1e-6, name="encoder_{}/att_layernormali
                                                query + attention_output)

    # Feed-forward layer
    ffn = keras.Sequential([layers.Dense(config.FF_DIM, activation="relu"),
                           layers.Dense(config.EMBED_DIM),
                           name="encoder_{}/ffn".format(i)])
    ffn_output = ffn(attention_output)
    ffn_output = layers.Dropout(0.1, name="encoder_{}/ffn_dropout".format(i))(ffn_output)
    sequence_output = layers.LayerNormalization(epsilon=1e-6, name="encoder_{}/ffn_layernormaliz
                                                attention_output + ffn_output)
    return sequence_output
```

Model 선언-1

Self-attention block



▪ BERT 모델 생성

```
def create_masked_language_bert_model():
    inputs = layers.Input((config.MAX_LEN,), dtype=tf.int64)

    word_embeddings = layers.Embedding(config.VOCAB_SIZE, config.EMBED_DIM, name="word_embedding")(
        inputs)
    position_embeddings = layers.Embedding(input_dim=config.MAX_LEN,
                                            output_dim=config.EMBED_DIM,
                                            weights=[get_pos_encoding_matrix(config.MAX_LEN, config.EMBED_DIM)],
                                            name="position_embedding")(
        tf.range(start=0, limit=config.MAX_LEN, delta=1))
    embeddings = word_embeddings + position_embeddings

    encoder_output = embeddings
    for i in range(config.NUM_LAYERS):
        encoder_output = Transformer_block(encoder_output, encoder_output, encoder_output, i)

    mlm_output = layers.Dense(config.VOCAB_SIZE, name="mlm_cls", activation="softmax")(
        encoder_output)

    mlm_model = MaskedLanguageModel(inputs, mlm_output, name="masked_bert_model")

    optimizer = keras.optimizers.Adam(learning_rate=config.LR)
    mlm_model.compile(optimizer=optimizer)
    return mlm_model
```

Model 선언-2

▪ BERT 모델 학습 설정

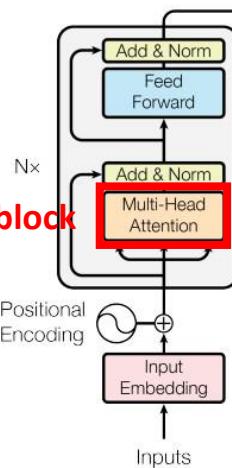
```
class MaskedLanguageModel(tf.keras.Model):  
  
    loss_fn = keras.losses.SparseCategoricalCrossentropy(reduction=tf.keras.losses.Reduction.NONE)  
    loss_tracker = tf.keras.metrics.Mean(name="loss")  
  
    def train_step(self, inputs):  
        if len(inputs) == 3:  
            features, labels, sample_weight = inputs  
        else:  
            features, labels = inputs  
            sample_weight = None  
  
        with tf.GradientTape() as tape:  
            predictions = self(features, training=True)  
            loss = loss_fn(labels, predictions, sample_weight=sample_weight)  
  
            # Compute gradients  
            trainable_vars = self.trainable_variables  
            gradients = tape.gradient(loss, trainable_vars)  
  
            # Update weights  
            self.optimizer.apply_gradients(zip(gradients, trainable_vars))  
  
            # Compute our own metrics  
            loss_tracker.update_state(loss, sample_weight=sample_weight)  
  
            # Return a dict mapping metric names to current value  
        return {"loss": loss_tracker.result()}
```

Model 선언-3

■ BERT 모델 선언

```
bert_masked_model = create_masked_language_bert_model()  
bert_masked_model.summary()
```

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 256)]	0	
word_embedding (Embedding)	(None, 256, 128)	3840000	input_1[0] [0]
tf.__operators__.add (TFOpLambd (None, 256, 128))	0		word_embedding[0] [0]
encoder_0/multiheadattention (M (None, 256, 128))	66048		tf.__operators__.add[0] [0] tf.__operators__.add[0] [0] tf.__operators__.add[0] [0]
encoder_0/att_dropout (Dropout) (None, 256, 128)	0		encoder_0/multiheadattention[0] [0]
tf.__operators__.add_1 (TFOpLam (None, 256, 128))	0		tf.__operators__.add[0] [0] encoder_0/att_dropout[0] [0]
encoder_0/att_layernormalizatio (None, 256, 128)	256		tf.__operators__.add_1[0] [0]
encoder_0/ffn (Sequential)	(None, 256, 128)	33024	encoder_0/att_layernormalization[
encoder_0/ffn_dropout (Dropout) (None, 256, 128)	0		encoder_0/ffn[0] [0]
tf.__operators__.add_2 (TFOpLam (None, 256, 128))	0		encoder_0/att_layernormalization[encoder_0/ffn_dropout[0] [0]
encoder_0/ffn_layernormalizatio (None, 256, 128)	256		tf.__operators__.add_2[0] [0]
mlm_cls (Dense)	(None, 256, 30000)	3870000	encoder_0/ffn_layernormalization[
<hr/>			
Total params: 7,809,584			
Trainable params: 7,809,584			
Non-trainable params: 0			
<hr/>			



BERT Train – 14분 소요

Train and Save

▶ %%time

```
bert_masked_model.fit(mlm_ds, epochs=5, callbacks=[generator_callback])
bert_masked_model.save("bert_mlm_imdb.h5")
```

```
Epoch 1/5
1563/1563 [=====] - 164s 103ms/step - loss: 6.9995
Epoch 2/5
1563/1563 [=====] - 160s 103ms/step - loss: 6.3822
Epoch 3/5
1563/1563 [=====] - 160s 103ms/step - loss: 5.6972
Epoch 4/5
1563/1563 [=====] - 160s 102ms/step - loss: 5.0691
Epoch 5/5
1563/1563 [=====] - 160s 102ms/step - loss: 4.7139
```

- **input_text': i have watched this [mask] and it was awesome**

- Epoch 1) [mask] : This, a, i, to, movie
- Epoch 5) [mask] : movie, film, show, time, review

Fine-tune a sentiment classification model

	review	sentiment
0	Well, if you are open-minded enough to have li...	0
1	If another Hitler ever arises, it will be than...	1
2	This show is wonderful. It has some of the bes...	0
3	This is one of the funniest and most excellent...	0
4	This kid is rather bad, but in no way do they ...	1

```
# Load pretrained bert model
mlm_model = keras.models.load_model("bert_mlm_imdb.h5", custom_objects={"MaskedLanguageModel": MaskedLanguageModel})
pretrained_bert_model = tf.keras.Model(mlm_model.input, mlm_model.get_layer("encoder_0/ffn_layernormalization").output)

# Freeze it
pretrained_bert_model.trainable = False

def create_classifier_bert_model():
    inputs = layers.Input((config.MAX_LEN,), dtype=tf.int64)
    sequence_output = pretrained_bert_model(inputs)
    pooled_output = layers.GlobalMaxPooling1D()(sequence_output)
    hidden_layer = layers.Dense(64, activation="relu")(pooled_output)
    outputs = layers.Dense(1, activation="sigmoid")(hidden_layer)
    classifier_model = keras.Model(inputs, outputs, name="classification")
    optimizer = keras.optimizers.Adam()
    classifier_model.compile(optimizer=optimizer, loss="binary_crossentropy", metrics=["accuracy"])
    return classifier_model

classifier_model = create_classifier_bert_model()
classifier_model.summary()
```

pretrained_bert_model.trainable = False		
Layer (type)	Output Shape	Param #
input_2 (InputLayer)	[(None, 256)]	0
model (Functional)	(None, 256, 128)	3939584
global_max_pooling1d (Global)	(None, 128)	0
dense_2 (Dense)	(None, 64)	8256
dense_3 (Dense)	(None, 1)	65
Total params: 3,947,905		
Trainable params: 8,321		
Non-trainable params: 3,939,584		

pretrained_bert_model.trainable = True		
Layer (type)	Output Shape	Param #
input_3 (InputLayer)	[(None, 256)]	0
model_1 (Functional)	(None, 256, 128)	3939584
global_max_pooling1d_1 (Global)	(None, 128)	0
dense_4 (Dense)	(None, 64)	8256
dense_5 (Dense)	(None, 1)	65
Total params: 3,947,905		
Trainable params: 3,947,905		
Non-trainable params: 0		

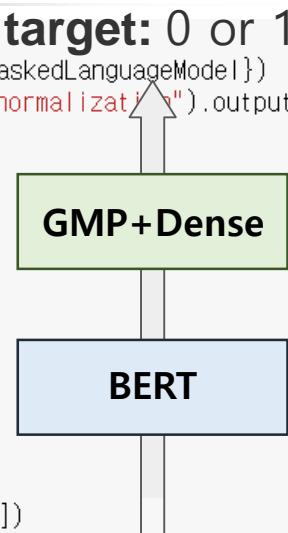
Fine-tune a sentiment classification model

```
# Load pretrained bert model
mlm_model = keras.models.load_model("bert_mlm_imdb.h5", custom_objects={"MaskedLanguageModel": MaskedLanguageModel})
pretrained_bert_model = tf.keras.Model(mlm_model.input, mlm_model.get_layer("encoder_0/ffn_layernorm").output)

# Freeze it
pretrained_bert_model.trainable = False

def create_classifier_bert_model():
    inputs = layers.Input((config.MAX_LEN,), dtype=tf.int64)
    sequence_output = pretrained_bert_model(inputs)
    pooled_output = layers.GlobalMaxPooling1D()(sequence_output)
    hidden_layer = layers.Dense(64, activation="relu")(pooled_output)
    outputs = layers.Dense(1, activation="sigmoid")(hidden_layer)
    classifier_model = keras.Model(inputs, outputs, name="classification")
    optimizer = keras.optimizers.Adam()
    classifier_model.compile(optimizer=optimizer, loss="binary_crossentropy", metrics=["accuracy"])
    return classifier_model

classifier_model = create_classifier_bert_model()
classifier_model.summary()
```



target: 0 or 1

SOURCE: "This was no Trainspotting or Guy Fawkes
itchie film. It was a big wannabee."

pretrained_bert_model.trainable = False

Layer (type)	Output Shape	Param #
input_2 (InputLayer)	[(None, 256)]	0
model (Functional)	(None, 256, 128)	3939584
global_max_pooling1d (Global)	(None, 128)	0
dense_2 (Dense)	(None, 64)	8256
dense_3 (Dense)	(None, 1)	65
<hr/>		
Total params:	3,947,905	
Trainable params:	8,321	
Non-trainable params:	3,939,584	

pretrained_bert_model.trainable = True

Layer (type)	Output Shape	Param #
input_3 (InputLayer)	[(None, 256)]	0
model_1 (Functional)	(None, 256, 128)	3939584
global_max_pooling1d_1 (Global)	(None, 128)	0
dense_4 (Dense)	(None, 64)	8256
dense_5 (Dense)	(None, 1)	65
<hr/>		
Total params:	3,947,905	
Trainable params:	3,947,905	
Non-trainable params:	0	

Fine-tune a sentiment classification model training

pretrained_bert_model.trainable = False		
Layer (type)	Output Shape	Param #
input_2 (InputLayer)	[(None, 256)]	0
model (Functional)	(None, 256, 128)	3939584
global_max_pooling1d (Global)	(None, 128)	0
dense_2 (Dense)	(None, 64)	8256
dense_3 (Dense)	(None, 1)	65
<hr/>		
Total params:	3,947,905	
Trainable params:	8,321	
Non-trainable params:	3,939,584	

pretrained_bert_model.trainable = True		
Layer (type)	Output Shape	Param #
input_3 (InputLayer)	[(None, 256)]	0
model_1 (Functional)	(None, 256, 128)	3939584
global_max_pooling1d_1 (Global)	(None, 128)	0
dense_4 (Dense)	(None, 64)	8256
dense_5 (Dense)	(None, 1)	65
<hr/>		
Total params:	3,947,905	
Trainable params:	3,947,905	
Non-trainable params:	0	

```
pretrained_bert_model.trainable = False
```

```
Epoch 1/5
782/782 [=====] - 12s 14ms/step - loss: 0.8259 - accuracy: 0.5303 - val_loss: 0.6677 - val_accuracy: 0.6018
Epoch 2/5
782/782 [=====] - 10s 13ms/step - loss: 0.6733 - accuracy: 0.6012 - val_loss: 0.6532 - val_accuracy: 0.6149
Epoch 3/5
782/782 [=====] - 10s 13ms/step - loss: 0.6462 - accuracy: 0.6323 - val_loss: 0.6948 - val_accuracy: 0.5790
Epoch 4/5
782/782 [=====] - 10s 13ms/step - loss: 0.6468 - accuracy: 0.6338 - val_loss: 0.6452 - val_accuracy: 0.6264
Epoch 5/5
782/782 [=====] - 10s 13ms/step - loss: 0.6415 - accuracy: 0.6321 - val_loss: 0.6411 - val_accuracy: 0.6312
```

```
pretrained_bert_model.trainable = True
```

```
Epoch 1/5
782/782 [=====] - 41s 51ms/step - loss: 0.5178 - accuracy: 0.7337 - val_loss: 0.3593 - val_accuracy: 0.8394
Epoch 2/5
782/782 [=====] - 39s 50ms/step - loss: 0.2812 - accuracy: 0.8820 - val_loss: 0.3431 - val_accuracy: 0.8612
Epoch 3/5
782/782 [=====] - 38s 49ms/step - loss: 0.1418 - accuracy: 0.9465 - val_loss: 0.5124 - val_accuracy: 0.8381
Epoch 4/5
782/782 [=====] - 38s 49ms/step - loss: 0.0555 - accuracy: 0.9782 - val_loss: 0.6536 - val_accuracy: 0.8366
Epoch 5/5
782/782 [=====] - 38s 49ms/step - loss: 0.0326 - accuracy: 0.9874 - val_loss: 0.7465 - val_accuracy: 0.8391
```

GPT Code Review

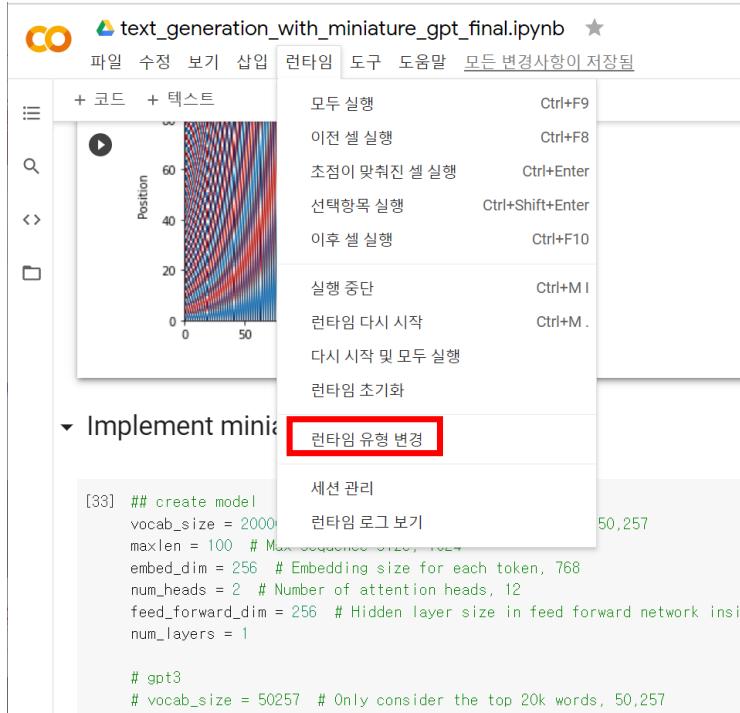
Keras Learning Day: <https://youtu.be/rGDLvSLO3Js>

Colab 설정

- 인터넷에서 아래 URL로 접근

- <https://colab.research.google.com/drive/1MbHsfbYJflyaE4yrzP7dg5iqXArs1YRb?usp=sharing>

- 런타임->런타임 유형변경->GPU 설정



text_generation_with_miniature_gpt_final.ipynb

파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

+ 코드 + 텍스트

모두 실행 Ctrl+F9

이전 셀 실행 Ctrl+F8

초점이 맞춰진 셀 실행 Ctrl+Enter

선택항목 실행 Ctrl+Shift+Enter

이후 셀 실행 Ctrl+F10

실행 중단 Ctrl+M I

런타임 다시 시작 Ctrl+M .

다시 시작 및 모두 실행

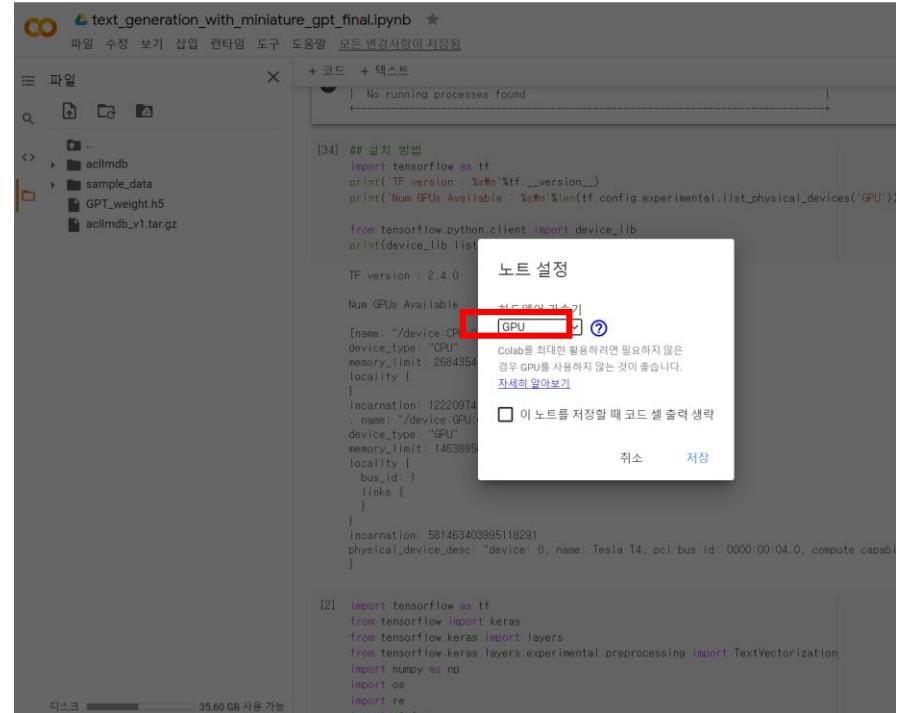
런타임 초기화

런타임 유형 변경

Implement miniature GPT

```
[33] ## create model
vocab_size = 2000
maxlen = 100 # Max sequence size, 1024
embed_dim = 256 # Embedding size for each token, 768
num_heads = 2 # Number of attention heads, 12
feed_forward_dim = 256 # Hidden layer size in feed forward network inside
num_layers = 1

# gpt3
# vocab_size = 50257 # Only consider the top 20k words, 50,257
```



text_generation_with_miniature_gpt_final.ipynb

파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

+ 코드 + 텍스트

No running processes found

[34] ## 설치 방법
import tensorflow as tf
print('TF version : %s' % tf.__version__)
print('Num GPUs Available: %s' % len(tf.config.experimental.list_physical_devices('GPU')))

from tensorflow.python.client import device_lib
print(device_lib.list_local_devices())

TF version : 2.4.0

Num GPUs Available: 1

[name: "/device:CPU:0", device_type: "CPU", memory_limit: 26849544, locality: {}, incarnation: 12220974, name: "/device:GPU:0", device_type: "GPU", memory_limit: 14638896, locality: {bus_id: 1, links: []}, incarnation: 581463403995110291, physical_device_desc: "device: 0, name: Tesla T4, pci bus id: 0000:00:04.0, compute capability: 7.5"]

[2] import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.layers.experimental.preprocessing import TextVectorization
import numpy as np
import os
import re
import string

노트 설정

런타임 유형 설정

[GPU]

Colab은 최대한 활용하려면 필요하지 않은 경우 GPU를 사용하지 않는 것이 좋습니다. 자세히 알아보기

이 노트를 저장할 때 코드 셀 출력 생략

취소 저장

Colab 설정

- >> !nvidia-smi 명령어로 GPU 설정 상태 확인
- Tesla T4(16G), 400만원 상당의 GPU 12시간 동안 사용 가능!!

```
▶ !nvidia-smi
```

Fri Dec 18 07:22:39 2020

NVIDIA-SMI 455.45.01			Driver Version: 418.67		CUDA Version: 10.1		
GPU	Name	Persistence-M	Bus-Id	Disp.A	Volatile	Uncorr. ECC	
Fan	Temp	Perf	Pwr:Usage/Cap	Memory-Usage	GPU-Util	Compute M.	
					MIG M.		
0	Tesla T4	Off	00000000:00:04.0	Off	0		
N/A	78C	P0	33W / 70W	14251MiB / 15079MiB	0%	Default	
						ERR!	

+

Processes:					
GPU	GI	CI	PID	Type	Process name
ID			ID		GPU Memory Usage
No running processes found					



Dell 16GB NVIDIA Tesla T4 GPU 그래픽 카드

NVIDIA Tesla GPU 가속기로 요구 사양이 높은 HPC, 하이퍼스케일, 엔터프라이즈 데이터 센터 작업을 가속화하세요.

이제 아티지 탑사에서 머신 러닝에 이르기까지, 다양한 분야의 과학자들이 GPU를 사용하는 것보다 빠른 속도로 페타바이트 단위의 데이터를 조사할 수 있습니다. 또한 Tesla 가속기는 다양한 시뮬레이션을 이전보다 빠른 속도로 실행할 수 있는 성능을 제공합니다. VDI를 배포한 기업에게 있어 Tesla 가속기는 모든 사용자에게 어디서 ... [더 보기](#)

3,969,636 원 7,773,600 원
할인 3,813,964 원
10% 부가세 포함

장바구니에 담기

GPT-code review

- Tensorflow 2.4 버전 사용 >> pip install tensorflow==2.4.0
- Colab에 기본적으로 설치되어 있음 -> 설치 필요 X

Setup

```
▶ !nvidia-smi  
▶ # 2.4.0버전으로 설치  
#!pip install tensorflow==2.4.0  
▶ ## 설치 방법  
import tensorflow as tf  
print('TF version : %s\n'%tf.__version__)  
print('Num GPUs Available : %s\n'%len(tf.config.experimental.list_physical_devices('GPU')))  
  
from tensorflow.python.client import device_lib  
# print(device_lib.list_local_devices())
```

TF version : 2.4.0

Num GPUs Available : 0

- TF 버전 확인
 - 2.4.0이면 OK

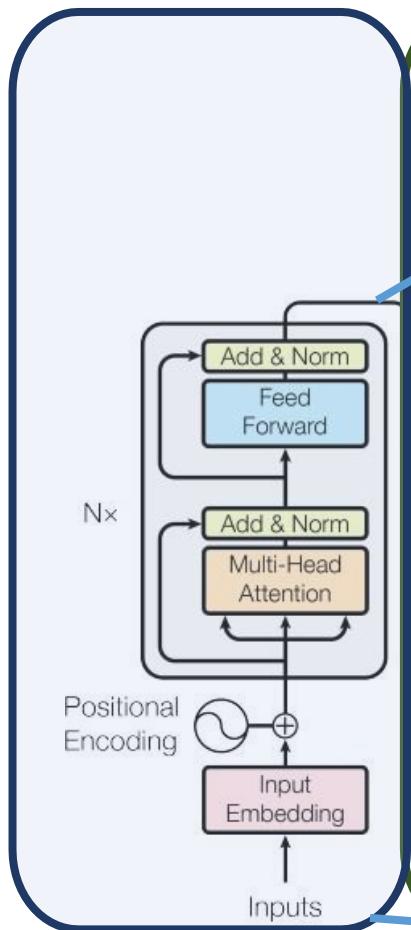
```
▶ import tensorflow as tf  
from tensorflow import keras  
from tensorflow.keras import layers  
from tensorflow.keras.layers.experimental.preprocessing import TextVectorization  
import numpy as np  
import os  
import re  
import string  
import random
```

- 각종 라이브러리 import

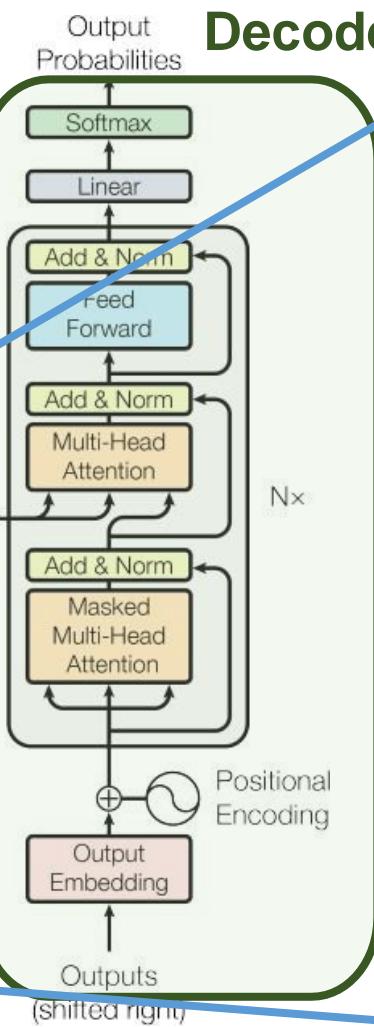
GPT-code review

- 2. Self-attention with causal masking

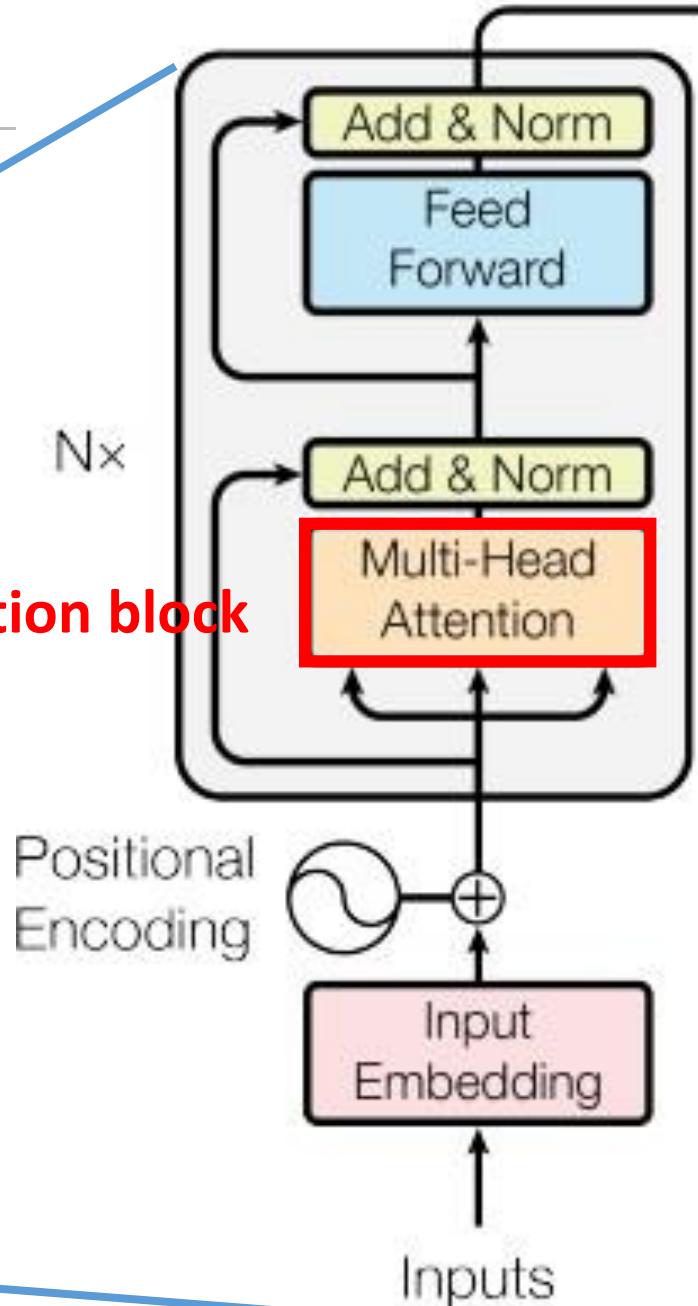
Encoder

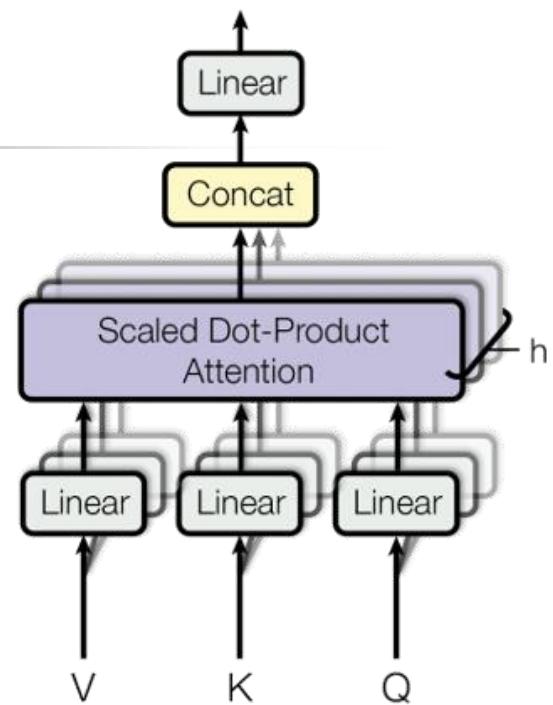


Decoder



Self-attention block

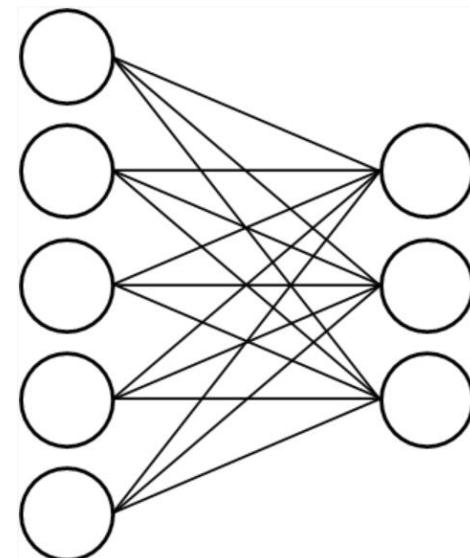




GPT-code review

▪ 2. Self-attention with causal masking

```
class MultiHeadSelfAttention(layers.Layer):
    def __init__(self, embed_dim, num_heads=8):
        super(MultiHeadSelfAttention, self).__init__()
        self.embed_dim = embed_dim
        self.num_heads = num_heads
        # embedding을 head 개수로 나눠서 계산하므로 예외처리
        if embed_dim % num_heads != 0:
            raise ValueError(
                f"embedding dimension = {embed_dim} should be divisible by number of heads = {num_heads}"
            )
        self.projection_dim = embed_dim // num_heads # 256 dimension을 8개의 head로 나눠서 계
        self.query_dense = layers.Dense(embed_dim)
        self.key_dense = layers.Dense(embed_dim)
        self.value_dense = layers.Dense(embed_dim)
        self.combine_heads = layers.Dense(embed_dim)
```

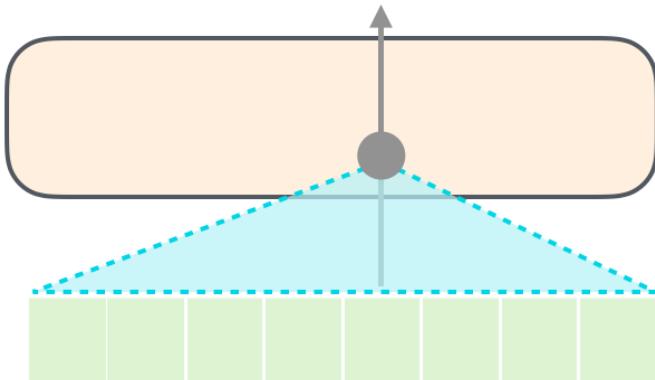


GPT-code review

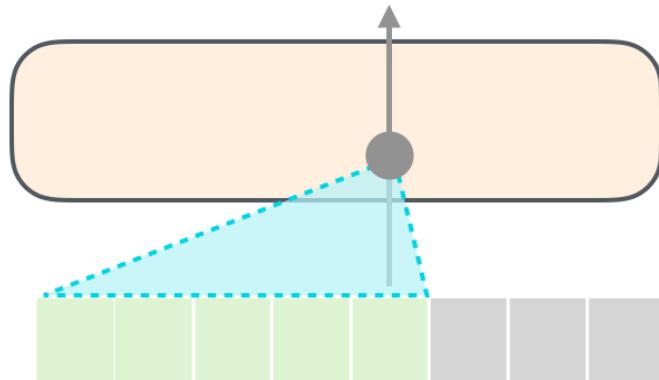
▪ 2. Self-attention with causal masking

```
@staticmethod
def causal_attention_mask(n_dest, n_src, dtype):
    # input : n_dest, n_src = seq_len
    # array([[[[[1., 0., 0., 0.],
    #           [1., 1., 0., 0.],
    #           [1., 1., 1., 0.],
    #           [1., 1., 1., 1.]]]])
    ...
    1's in the lower triangle, counting from the lower right corner.
    ...
    i = tf.range(n_dest)[:, None]
    j = tf.range(n_src)
    m = i >= j - n_src + n_dest
    return tf.cast(m, dtype) # true->1, false->2로 변환
```

Self-Attention



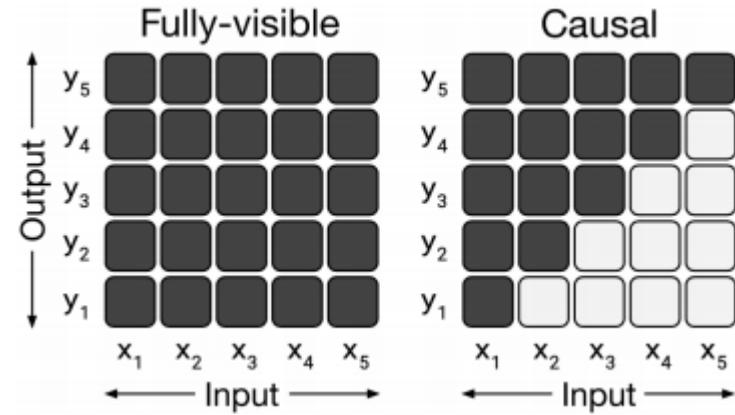
Masked Self-Attention



GPT-code review

▪ 2. Self-attention with causal masking

```
@staticmethod
def causal_attention_mask(n_dest, n_src, dtype):
    # input : n_dest, n_src = seq_len
    # array([[[[1., 0., 0., 0.],
    #          [1., 1., 0., 0.],
    #          [1., 1., 1., 0.],
    #          [1., 1., 1., 1.]]]])
    # ...
    # 1's in the lower triangle, counting from the lower right corner.
    # ...
    i = tf.range(n_dest)[:, None]
    j = tf.range(n_src)
    m = i >= j - n_src + n_dest
    return tf.cast(m, dtype) # true->1, false->2로 변환
```



Features					Labels	
position: 1		2	3	4		
Example:	robot	must	obey	orders	must	obey
1	robot	must	obey	orders		
2	robot	must	obey	orders		
3	robot	must	obey	orders		
4	robot	must	obey	orders		

GPT-code review

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

▪ 2. Self-attention with causal masking

```
def attention(self, query, key, value):
    # input
    # q, k, v : (batch_size, num_heads, seq_len, projection_dim)
    # output
    # 1. output : (batch_size, seq_len, num_heads, projection_dim)
    # 2. weights
    # score : (batch_size, seq_len, seq_len) = (seq_len, embed_dim)x(seq_len, embed_dim)
    score = tf.matmul(query, key, transpose_b=True) # (batch_size, seq_len, seq_len)
    dim_key = tf.cast(tf.shape(key)[-1], tf.float32) # embed_dim
    # scaled_score : (batch_size, seq_len, seq_len), sqrt(embed_dim)로 나눠서 normalize
    scaled_score = score / tf.math.sqrt(dim_key) # (batch_size, seq_len, seq_len)

    # prevent information flow from future tokens
    shape = tf.shape(scaled_score) # 4차원
    dim_dest, dim_src = shape[2], shape[3]
    # attention_mask
    # array([[[[1., 0., 0., 0.],
    #          [1., 1., 0., 0.],
    #          [1., 1., 1., 0.],
    #          [1., 1., 1., 1.]]]])
    attention_mask = self.causal_attention_mask(dim_dest, dim_src, scaled_score.dtype)
    attention_mask = tf.reshape(attention_mask, [1, 1, dim_dest, dim_src]) # why? 201?
    # 하삼각 행렬은 값을 갖고, 상삼각행렬은 -10000로 attention0/ 가해지지 않도록
    scaled_score = scaled_score * attention_mask - 1e4 * (1 - attention_mask)

    weights = tf.nn.softmax(scaled_score, axis=-1) # (batch_size, num_heads, seq_len, seq_len)
    output = tf.matmul(weights, value) # (batch_size, num_heads, seq_len, projection_dim)
    return output, weights
```

GPT-code review

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

▪ 2. Self-attention with causal masking

```
def attention(self, query, key, value):  
    # input  
    # q, k, v : (batch_size, num_heads, seq_len, projection_dim)  
    # output  
    # 1. output : (batch_size, seq_len, num_heads, projection_dim)
```

Scores
(before softmax)

0.11	0.00	0.81	0.79
0.19	0.50	0.30	0.48
0.53	0.98	0.95	0.14
0.81	0.86	0.38	0.90

Apply Attention
Mask

Masked Scores
(before softmax)

0.11	-inf	-inf	-inf
0.19	0.50	-inf	-inf
0.53	0.98	0.95	-inf
0.81	0.86	0.38	0.90

```
attention_mask = self.causal_attention_mask(dim_dest, dim_src, scaled_score.dtype)  
attention_mask = tf.reshape(attention_mask, [1, 1, dim_dest, dim_src]) # why? 201?  
# 하상각 행렬은 값을 갖고, 상상각행렬은 -10000로 attention0/ 가해지지 않도록  
scaled_score = scaled_score * attention_mask - 1e4 * (1 - attention_mask)
```

```
weights = tf.nn.softmax(scaled_score, axis=-1) # (batch_size, num_heads, seq_len, seq_len)  
output = tf.matmul(weights, value) # (batch_size, num_heads, seq_len, projection_dim)  
return output, weights
```

GPT-code review

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

▪ 2. Self-attention with causal masking

```
def attention(self, query, key, value):
    # input
    # q, k, v : (batch_size, num_heads, seq_len, projection_dim)
    # output
    # 1. output : (batch_size, seq_len, num_heads, projection_dim)
    # 2. weights
    # score : (batch_size, seq_len, seq_len) = (seq_len, embed_dim)x(seq_len, embed_dim)
    score = tf.matmul(query, key, transpose_b=True) # (batch_size, seq_len, seq_len)
    dim_key = tf.cast(tf.shape(key)[-1], tf.float32) # embed_dim
    # scaled_score : (batch_size, seq_len, seq_len), sqrt(embed_dim)로 나눠서 normalize
    scaled_score = score / tf.math.sqrt(dim_key) # (batch_size, seq_len, seq_len)
```

Masked Scores
(before softmax)

0.11	-inf	-inf	-inf
0.19	0.50	-inf	-inf
0.53	0.98	0.95	-inf
0.81	0.86	0.38	0.90

Softmax
(along rows)

Scores

1	0	0	0
0.48	0.52	0	0
0.31	0.35	0.34	0
0.25	0.26	0.23	0.26

```
weights = tf.nn.softmax(scaled_score, axis=-1) # (batch_size, num_heads, seq_len, seq_len)
output = tf.matmul(weights, value) # (batch_size, num_heads, seq_len, projection_dim)
return output, weights
```

GPT-code review

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

▪ 2. Self-attention with causal masking

```
def separate_heads(self, x, batch_size):
    x = tf.reshape(x, (batch_size, -1, self.num_heads, self.projection_dim))
    return tf.transpose(x, perm=[0, 2, 1, 3])

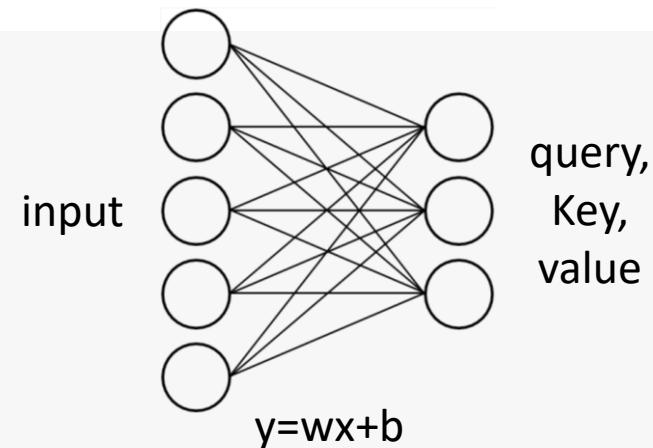
def call(self, inputs):
    # x.shape = [batch_size, seq_len, embedding_dim]
    batch_size = tf.shape(inputs)[0]

    # q, k, v 구하기
    query = self.query_dense(inputs) # (batch_size, seq_len, embed_dim)
    key = self.key_dense(inputs) # (batch_size, seq_len, embed_dim)
    value = self.value_dense(inputs) # (batch_size, seq_len, embed_dim)

    # multihead로 풀기, (batch_size, seq_len, embed_dim) -> (batch_size, num_heads, seq_len, projection_dim)
    query = self.separate_heads(query, batch_size) # (batch_size, num_heads, seq_len, projection_dim)
    key = self.separate_heads(key, batch_size) # (batch_size, num_heads, seq_len, projection_dim)
    value = self.separate_heads(value, batch_size) # (batch_size, num_heads, seq_len, projection_dim)

    # q,k,v의 attention 구하기
    attention, weights = self.attention(query, key, value) # attention : (batch_size, num_heads, seq_len, projection_dim)
    attention = tf.transpose(attention, perm=[0, 2, 1, 3]) # (batch_size, seq_len, num_heads, projection_dim)
    # multi-head attention을 하나로 합치기
    concat_attention = tf.reshape(attention, (batch_size, -1, self.embed_dim)) # (batch_size, seq_len, embed_dim)
    # 최종 dense layer
    output = self.combine_heads(concat_attention) # (batch_size, seq_len, embed_dim)

    # output : (batch_size, seq_len, embed_dim)
    return output
```



GPT-code review

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

▪ 2. Self-attention with causal masking

```
def separate_heads(self, x, batch_size):
    x = tf.reshape(x, (batch_size, -1, self.num_heads, self.projection_dim))
    return tf.transpose(x, perm=[0, 2, 1, 3])

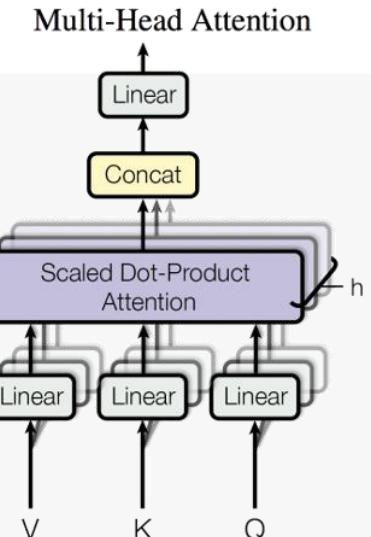
def call(self, inputs):
    # x.shape = [batch_size, seq_len, embedding_dim]
    batch_size = tf.shape(inputs)[0]

    # q, k, v 구하기
    query = self.query_dense(inputs) # (batch_size, seq_len, embed_dim)
    key = self.key_dense(inputs) # (batch_size, seq_len, embed_dim)
    value = self.value_dense(inputs) # (batch_size, seq_len, embed_dim)

    # multihead로 풀기, (batch_size, seq_len, embed_dim) -> (batch_size, num_heads, seq_len, projection_dim)
    query = self.separate_heads(query, batch_size) # (batch_size, num_heads, seq_len, projection_dim)
    key = self.separate_heads(key, batch_size) # (batch_size, num_heads, seq_len, projection_dim)
    value = self.separate_heads(value, batch_size) # (batch_size, num_heads, seq_len, projection_dim)

    # q,k,v의 attention 구하기
    attention, weights = self.attention(query, key, value) # attention : (batch_size, num_heads, seq_len, projection_dim)
    attention = tf.transpose(attention, perm=[0, 2, 1, 3]) # (batch_size, seq_len, num_heads, projection_dim)
    # multi-head attention을 하나로 합치기
    concat_attention = tf.reshape(attention, (batch_size, -1, self.embed_dim)) # (batch_size, seq_len, embed_dim)
    # 최종 dense layer
    output = self.combine_heads(concat_attention) # (batch_size, seq_len, embed_dim)

    # output : (batch_size, seq_len, embed_dim)
    return output
```

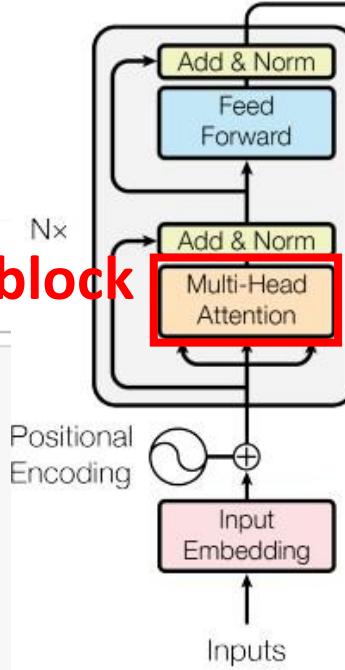


GPT-code review

Implement a Transformer block as a layer **Self-attention block**

```
▶ class TransformerBlock(layers.Layer):
    def __init__(self, embed_dim, num_heads, ff_dim, rate=0.1):
        # embed_dim=512
        # num_heads=8
        # ff_dim=2048
        super(TransformerBlock, self).__init__()
        self.att = MultiHeadSelfAttention(embed_dim, num_heads)
        self.ffn = keras.Sequential([
            layers.Dense(ff_dim, activation="relu"), ### # (batch_size, seq_len, ff_dim)
            layers.Dense(embed_dim) ### # (batch_size, seq_len, embed_dim)
        ])
        self.layernorm1 = layers.LayerNormalization(epsilon=1e-6)
        self.layernorm2 = layers.LayerNormalization(epsilon=1e-6)
        self.dropout1 = layers.Dropout(rate)
        self.dropout2 = layers.Dropout(rate)

    def call(self, inputs):
        attention_output = self.att(inputs)
        attention_output = self.dropout1(attention_output) ### # (batch_size, seq_len, embed_dim)
        out1 = self.layernorm1(inputs + attention_output) ### # (batch_size, seq_len, embed_dim)
        ffn_output = self.ffn(out1) ### # (batch_size, seq_len, embed_dim)
        ffn_output = self.dropout2(ffn_output)
        return self.layernorm2(out1 + ffn_output) ### # (batch_size, seq_len, embed_dim)
```



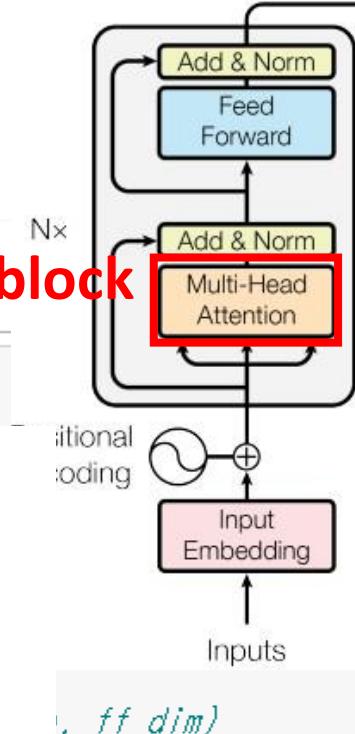
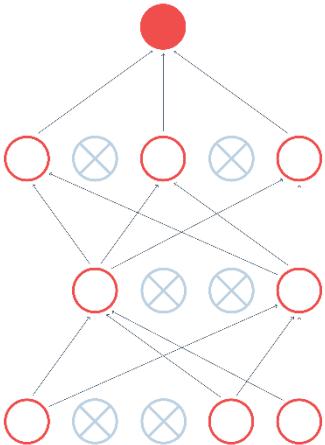
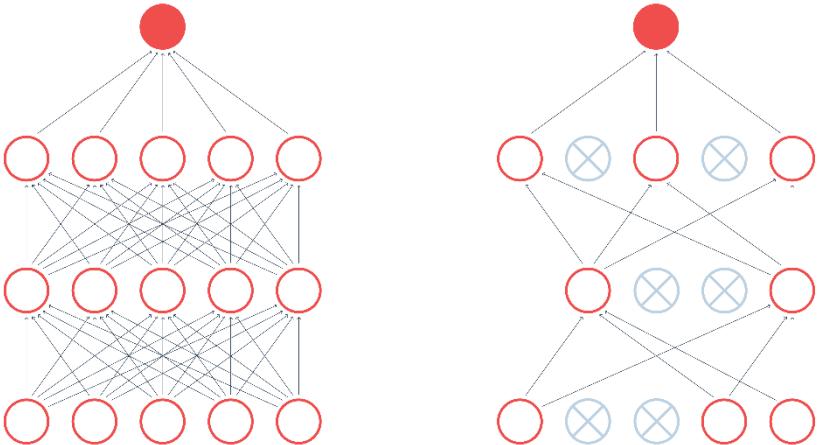
GPT-code review

Implement a Transformer block as a layer **Self-attention block**

```
class TransformerBlock(layers.Layer):
    def __init__(self, embed_dim, num_heads, ff_dim, rate=0.1):
        # em
        # nu
        # ff
        super()
        self
        self

    ])
    self
    self
    self
    self

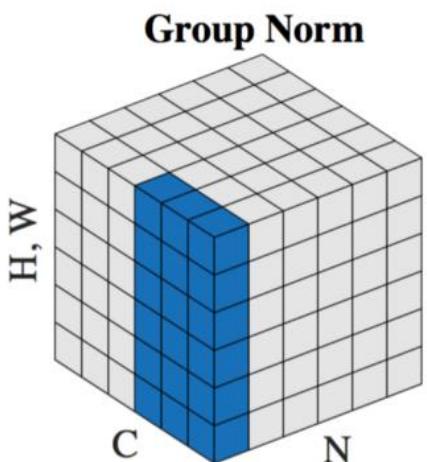
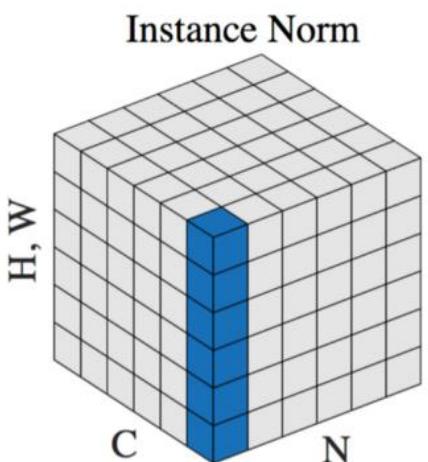
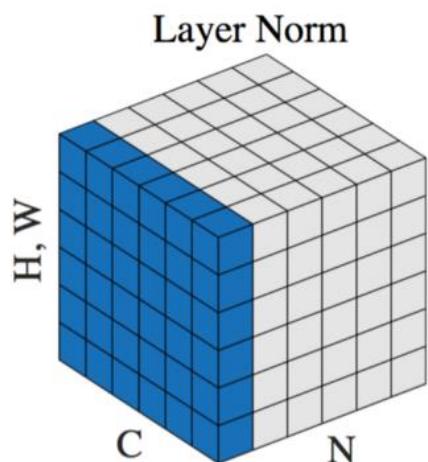
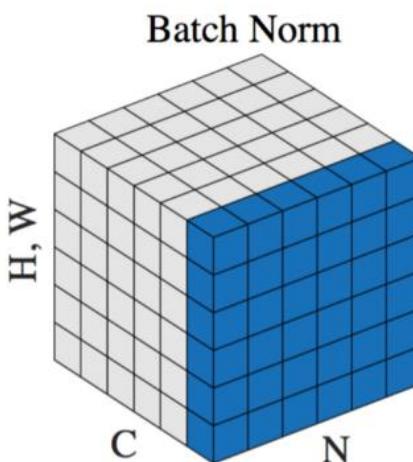
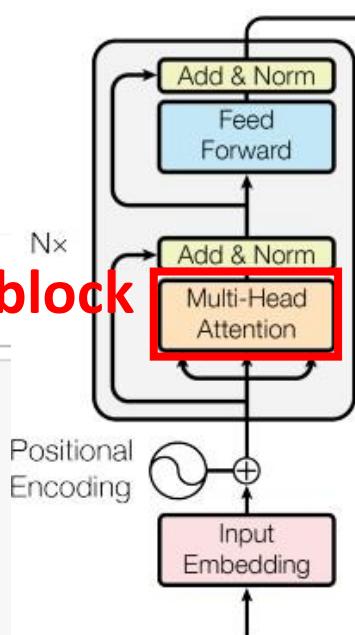
    def call
        atte
        attention_output = self.dropout1(attention_output) ### # (batch_size, seq_len, embed_dim)
        out1 = self.layernorm1(inputs + attention_output) ### # (batch_size, seq_len, embed_dim)
        ffn_output = self.ffn(out1) ### # (batch_size, seq_len, embed_dim)
        ffn_output = self.dropout2(ffn_output)
        return self.layernorm2(out1 + ffn_output) ### # (batch_size, seq_len, embed_dim)
```



GPT-code review

Implement a Transformer block as a layer **Self-attention block**

```
▶ class TransformerBlock(layers.Layer):  
    def __init__(self, embed_dim, num_heads, ff_dim, rate=0.1):  
        # embed_dim=512  
        # num_heads=8  
        # ff_dim=2048  
        super(TransformerBlock, self).__init__()
```

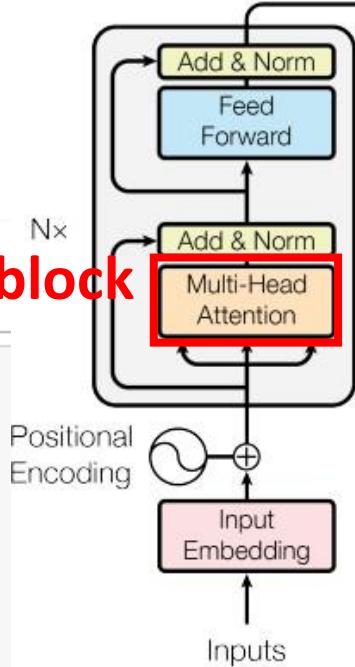
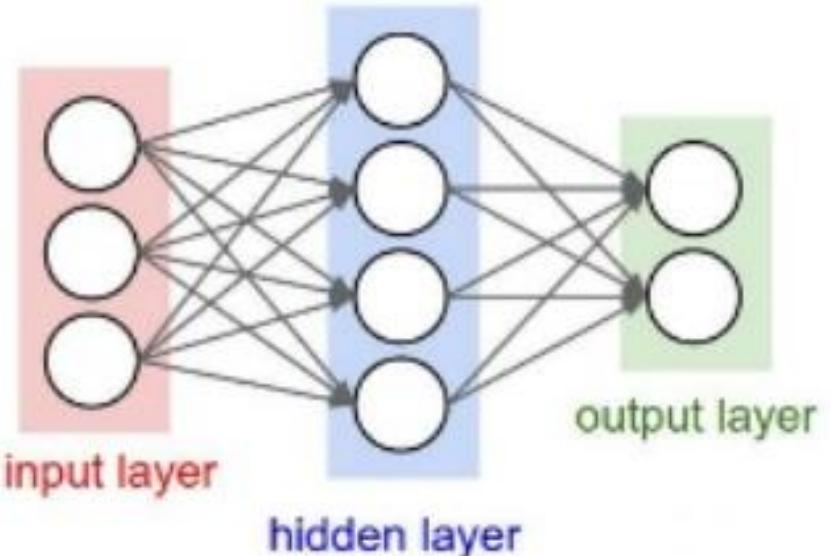


```
        out1 = self.layernorm1(inputs + attention_output) ### # (batch_size, seq_len, embed_dim)  
        ffn_output = self.ffn(out1) ### # (batch_size, seq_len, embed_dim)  
        ffn_output = self.dropout2(ffn_output)  
    return self.layernorm2(out1 + ffn_output) ### # (batch_size, seq_len, embed_dim)
```

GPT-code review

Implement a Transformer block as a layer **Self-attention block**

```
▶ class TransformerBlock(layers.Layer):  
    def __init__(self, embed_dim, num_heads, ff_dim, rate=0.1):  
        # embed_dim=512  
        # num_heads=8  
        # ff_dim=2048  
        super(Transf  
        self.att = M  
        self.ffn = k  
            layers.[  
            layers.[  
        ])  
        self.layernc  
        self.layernc  
        self.dropout  
        self.dropout  
  
    def call(self,  
            attention_out1,  
            attention_out2,  
            out1 = self.  
            ffn_output = self.ffn(out1) ### # (batch_size, seq_len, embed_dim)  
            ffn_output = self.dropout2(ffn_output)  
        return self.layernorm2(out1 + ffn_output) ### # (batch_size, seq_len, embed_dim)
```

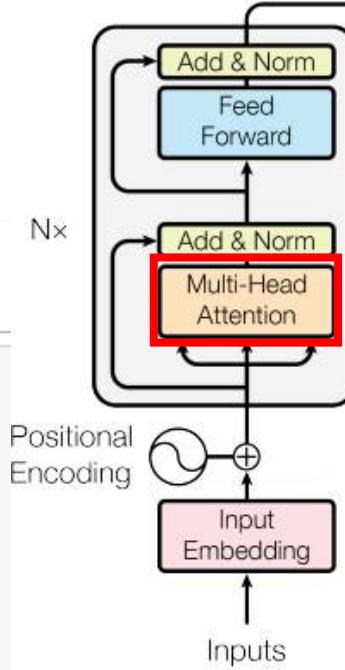


GPT-code review

Implement a Transformer block as a layer

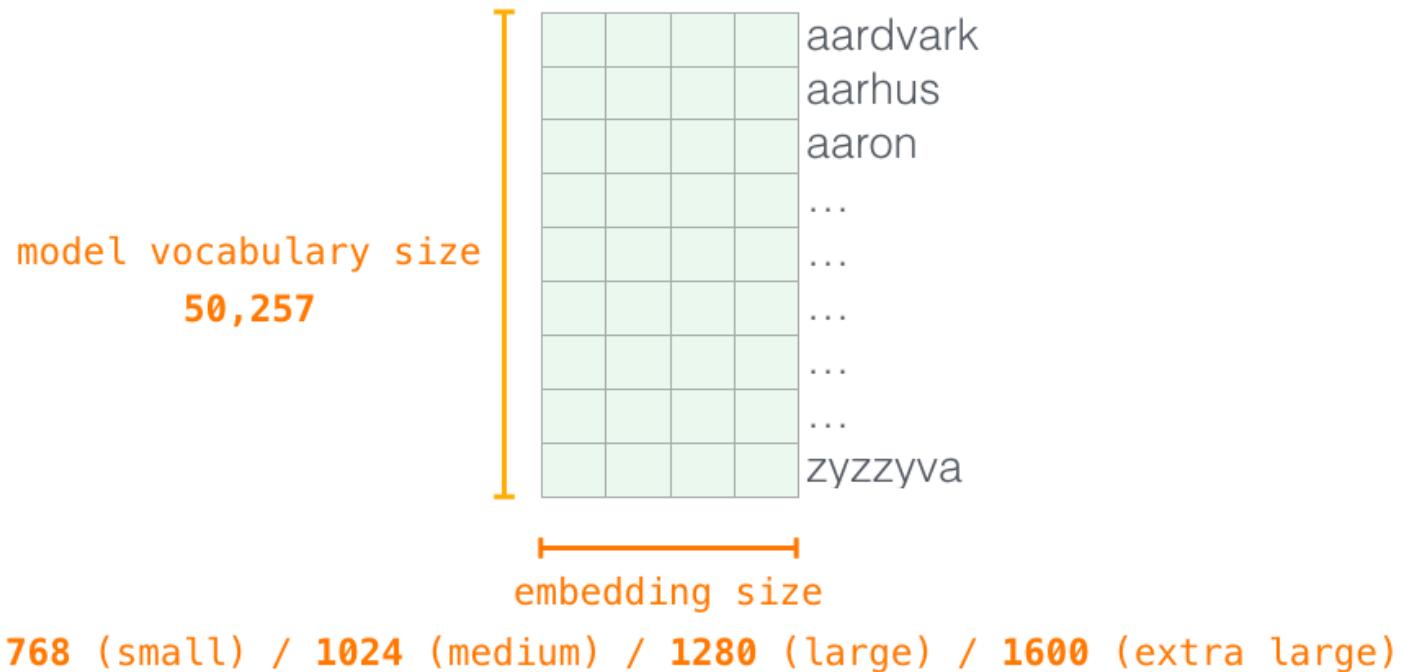
```
▶ class TransformerBlock(layers.Layer):
    def __init__(self, embed_dim, num_heads, ff_dim, rate=0.1):
        # embed_dim=512
        # num_heads=8
        # ff_dim=2048
        super(TransformerBlock, self).__init__()
        self.att = MultiHeadSelfAttention(embed_dim, num_heads)
        self.ffn = keras.Sequential([
            layers.Dense(ff_dim, activation="relu"), ### # (batch_size, seq_len, ff_dim)
            layers.Dense(embed_dim) ### # (batch_size, seq_len, embed_dim)
        ])
        self.layernorm1 = layers.LayerNormalization(epsilon=1e-6)
        self.layernorm2 = layers.LayerNormalization(epsilon=1e-6)
        self.dropout1 = layers.Dropout(rate)
        self.dropout2 = layers.Dropout(rate)

    def call(self, inputs):
        attention_output = self.att(inputs)
        attention_output = self.dropout1(attention_output) ### # (batch_size, seq_len, embed_dim)
        out1 = self.layernorm1(inputs + attention_output) ### # (batch_size, seq_len, embed_dim)
        ffn_output = self.ffn(out1) ### # (batch_size, seq_len, embed_dim)
        ffn_output = self.dropout2(ffn_output)
        return self.layernorm2(out1 + ffn_output) ### # (batch_size, seq_len, embed_dim)
```



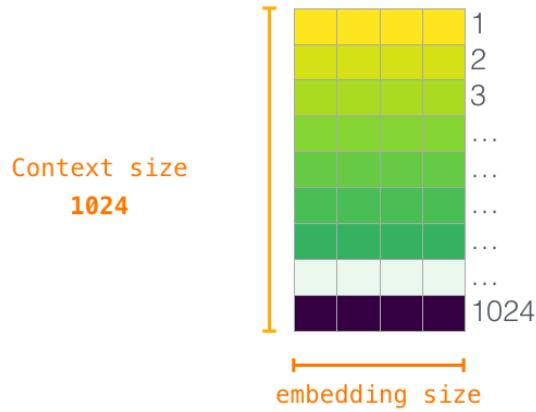
Input Embedding

- **Embedding – 1) Token Embedding**



Input Embedding

▪ Embedding – 2) Position Encoding

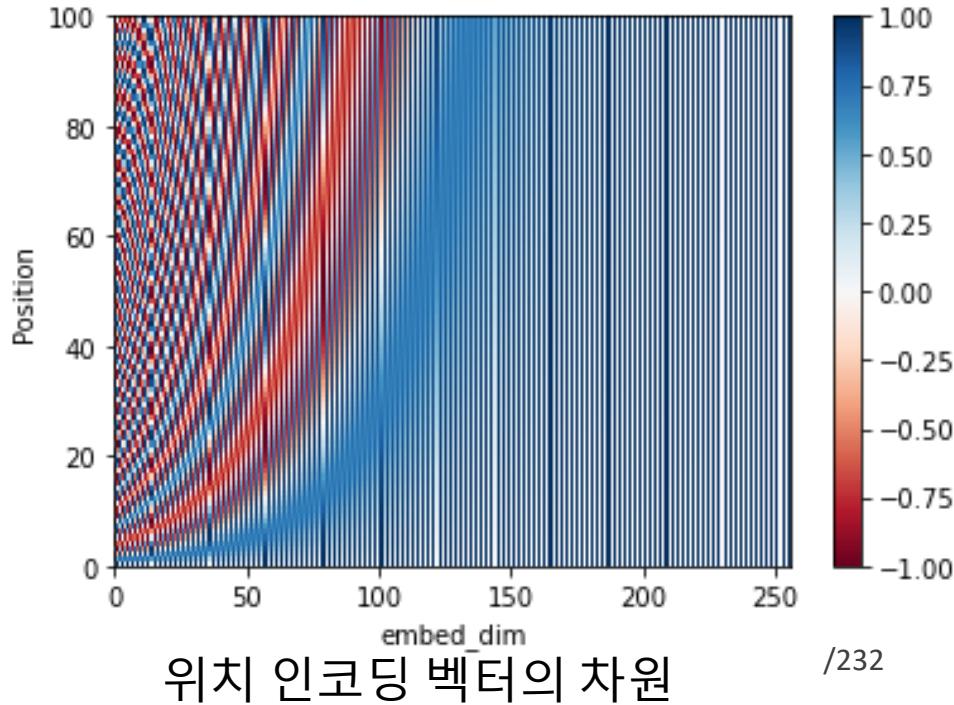


768 (small) / 1024 (medium) / 1280 (large) / 1600 (extra large)

0 :	0 0 0 0	8 :	1 0 0 0
1 :	0 0 0 1	9 :	1 0 0 1
2 :	0 0 1 0	10 :	1 0 1 0
3 :	0 0 1 1	11 :	1 0 1 1
4 :	0 1 0 0	12 :	1 1 0 0
5 :	0 1 0 1	13 :	1 1 0 1
6 :	0 1 1 0	14 :	1 1 1 0
7 :	0 1 1 1	15 :	1 1 1 1

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

Sequence 내 토큰의 위치 t



Input Embedding

▪ Embedding – 2) Position Encoding

```
# transformer 와 같은 positional encoding 방식
maxlen = 100
embed_dim = 256

def get_angles(pos, i, d_model):
    angle_rates = 1 / np.power(10000, (2 * (i//2)) / np.float32(d_model))
    return pos * angle_rates

def positional_encoding(position, d_model):
    angle_rads = get_angles(np.arange(position)[:, np.newaxis],
                           np.arange(d_model)[np.newaxis, :], d_model)

    # apply sin to even indices in the array; 2i
    angle_rads[:, 0::2] = np.sin(angle_rads[:, 0::2])

    # apply cos to odd indices in the array; 2i+1
    angle_rads[:, 1::2] = np.cos(angle_rads[:, 1::2])

    pos_encoding = angle_rads[np.newaxis, ...]

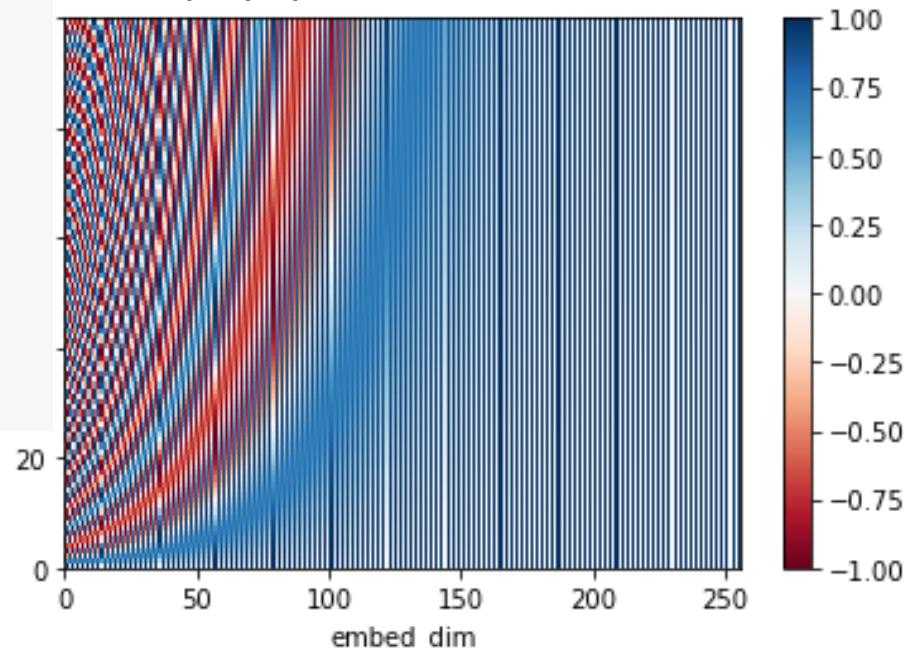
    return tf.cast(pos_encoding, dtype=tf.float32)

pos_encoding = positional_encoding(maxlen, embed_dim)
print (pos_encoding.shape)
```

4:	0 1 0 0	12:	1 1 0 0
5:	0 1 0 1	13:	1 1 0 1
6:	0 1 1 0	14:	1 1 1 0
7:	0 1 1 1	15:	1 1 1 1

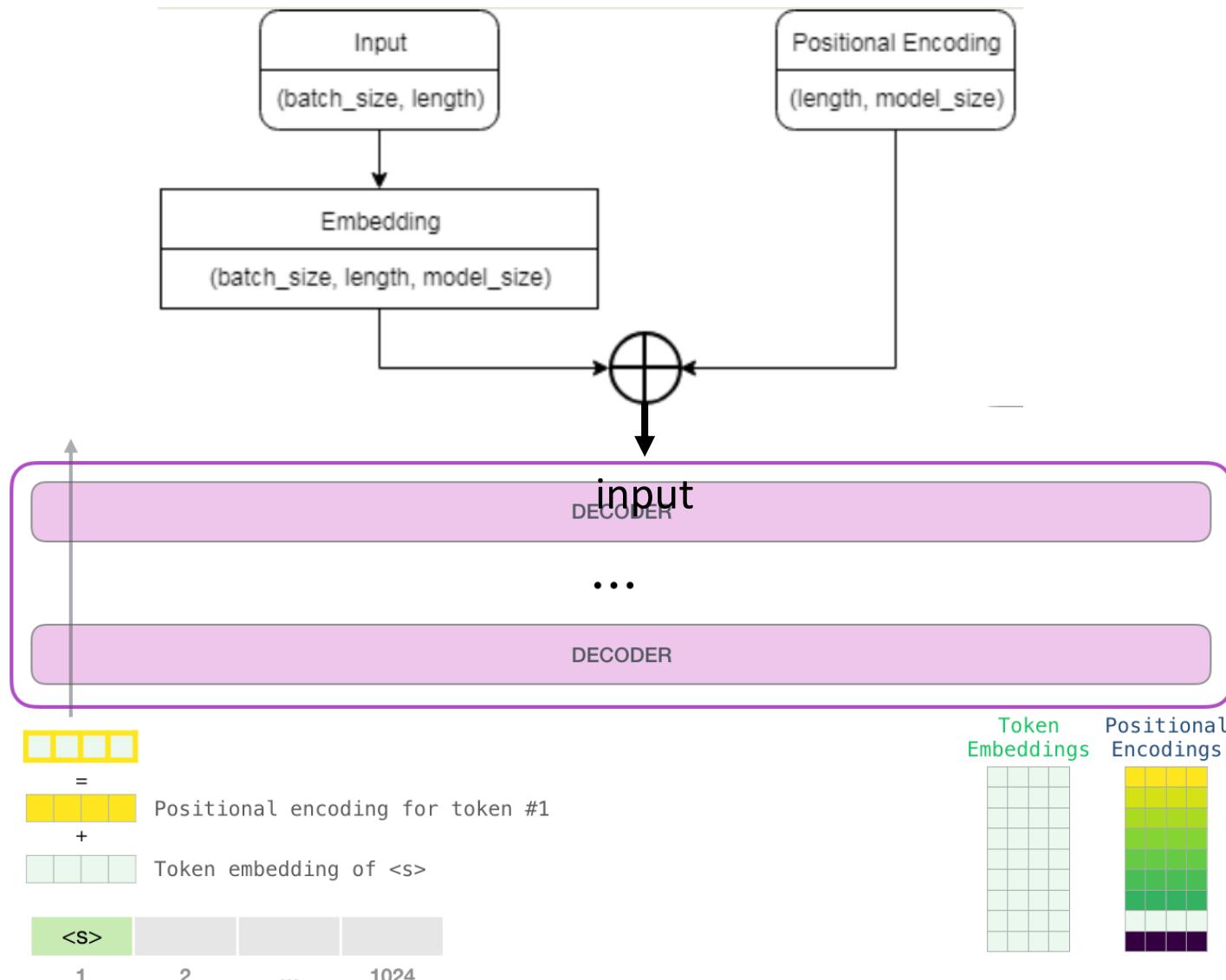
$$E_{(pos,2i)} = \sin(pos/10000^{2i/d_{\text{model}}})$$
$$E_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{\text{model}}})$$

토큰의 위치 t



Input Embedding

- Embedding – input : Token Embedding + Position Encoding



Input Embedding

Implement embedding layer

Two separate embedding layers, one for tokens, one for token index (positions).

```
▶ class TokenAndPositionEmbedding(layers.Layer):  
    # input : (batch_size, seq_len)  
    # output: (batch_size, seq_len, embed_dim)  
    def __init__(self, maxlen, vocab_size, embed_dim):  
        super(TokenAndPositionEmbedding, self).__init__()  
        self.token_emb = layers.Embedding(input_dim=vocab_size, output_dim=embed_dim)  
        self.pos_emb = layers.Embedding(input_dim=maxlen, output_dim=embed_dim)  
  
    def call(self, x):  
        # x : (batch_size, seq_len)  
        maxlen = tf.shape(x)[-1] # seq_len  
        positions = tf.range(start=0, limit=maxlen, delta=1) # [0, 1, ...seq_len]  
        positions = self.pos_emb(positions) # (seq_len, embed_dim)  
        x = self.token_emb(x) # (batch_size, seq_len, embed_dim)  
        return x + positions
```

본 실습코드에서는 Positional Encoding 사용 안함

모델 생성

▪ Create model

```
vocab_size = 100 # Only consider the top 20k words, 50,257
maxlen = 100 # Max sequence size, 1024
embed_dim = 256 # Embedding size for each token, 768
num_heads = 2 # Number of attention heads, 12
feed_forward_dim = 256 # Hidden layer size in feed forward network inside transformer, 1024 or 2048
num_layers = 1

def create_model():
    inputs = layers.Input(shape=(maxlen,), dtype=tf.int32)
    embedding_layer = TokenAndPositionEmbedding(maxlen, vocab_size, embed_dim) # (batch_size, input_seq_len) -> (batch_size, input_seq_len, embed_dim)
    x = embedding_layer(inputs)
    transformer_block = TransformerBlock(embed_dim, num_heads, feed_forward_dim) # (batch_size, input_seq_len, embed_dim) -> (batch_size, input_seq_len, embed_dim)
    transformer_block_list = [TransformerBlock(embed_dim, num_heads, feed_forward_dim) for _ in range(num_layers)]

    # x = transformer_block(x)
    ## 추가, multi-layers
    for i in range(num_layers):
        x = transformer_block_list[i](x)

    outputs = layers.Dense(vocab_size)(x)
    model = keras.Model(inputs=inputs, outputs=[outputs, x])
    # 훈련 데이터의 label(target)이 one-hot vector 이면 CategoricalCrossentropy
    # 훈련 데이터의 label(target)이 정수이면 SparseCategoricalCrossentropy
    loss_fn = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
    model.compile(
        "adam", loss=[loss_fn, None],
    ) # No loss and optimization based on word embeddings from transformer block
    return model

model = create_model()
model.summary()
```

모델 생성

▪ Create model

```
vocab_size = 100 # Only consider the top 20k words, 50,257
maxlen = 100 # Max sequence size, 1024
embed_dim = 256 # Embedding size for each token, 768
num_heads = 2 # Number of attention heads, 12
feed_forward_dim = 256 # Hidden layer size in feed forward network inside transformer, 1024 or 2048
num_layers = 1

def create_model():
    inputs = layers.Input(shape=(maxlen,), dtype=tf.int32)
    embedding_layer = TokenAndPositionEmbedding(maxlen, vocab_size, embed_dim) # (batch_size, input_seq_len) -> (batch_size, input_seq_len, embed_dim)
    x = embedding_layer(inputs)
    transformer_block = TransformerBlock(embed_dim, num_heads, feed_forward_dim) # (batch_size, input_seq_len, embed_dim) -> (batch_size, input_seq_len, embed_dim)
    transformer_block_list = [TransformerBlock(embed_dim, num_heads, feed_forward_dim) for _ in range(num_layers)]

    # x = transformer_block(x)
    ## 추가, multi-layers
    for i in range(num_layers):
        x = transformer_block_list[i](x)

    outputs = layers.Dense(vocab_size)(x)
    model = keras.Model(inputs=inputs, outputs=[outputs, x])
    # 훈련 데이터의 label(target)이 one-hot vector 이면 CategoricalCrossentropy
    # 훈련 데이터의 label(target)이 정수이면 SparseCategoricalCrossentropy
    loss_fn = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False)
    model.compile(
        "adam", loss=[loss_fn, None],
    ) # No loss and optimization based on word embeddings from transformer
    return model

model = create_model()
model.summary()
```

Layer (type)	Output Shape	Param #
input_2 (InputLayer)	[(None, 100)]	0
token_and_position_embedding	(None, 100, 256)	51200
transformer_block_3 (TransformerBlock)	(None, 100, 256)	395776
dense_25 (Dense)	(None, 100, 100)	25700
Total params: 472,676		
Trainable params: 472,676		
Non-trainable params: 0		

모델 생성

▪ Create model

```
vocab_size = 100 # Only consider the top 20k words, 50,257
maxlen = 100 # Max sequence size, 1024
embed_dim = 256 # Embedding size for each token, 768
num_heads = 2 # Number of attention heads, 12
feed_forward_dim = 256 # Hidden layer size in feed forward network inside transformer, 1024 or 2048
num_layers = 1
```

```
def create_model():
    inputs = layers.Input(shape=(maxlen,), dtype=tf.int32)
    embedding_layer = TokenAndPositionEmbedding(maxlen, vocab_size, embed_dim)
    x = embedding_layer(inputs)
    transformer_block = TransformerBlock(embed_dim, num_heads, feed_forward_dim)
    transformer_block_list = [TransformerBlock(embed_dim, num_heads, feed_forward_dim) for _ in range(num_layers)]
    for i in range(num_layers):
        x = transformer_block_list[i](x)

    outputs = layers.Dense(vocab_size)(x)
    model = keras.Model(inputs=inputs, outputs=[outputs, x])
    # 훈련 데잍의 label(target)이 one-hot vector 이면 CategoricalCrossentropy
    # 훈련 데잍의 label(target)이 정수이면 SparseCategoricalCrossentropy
    loss_fn = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
    model.compile(
        "adam", loss=[loss_fn, None],
    ) # No loss and optimization based on word embeddings from transformer
    return model
```

```
model = create_model()
model.summary()
```

Layer (type)	Output Shape	Param #
input_4 (InputLayer)	[(None, 100)]	0
token_and_position_embedding	(None, 100, 256)	51200
transformer_block_18 (TransformerBlock)	(None, 100, 256)	395776
transformer_block_19 (TransformerBlock)	(None, 100, 256)	395776
transformer_block_20 (TransformerBlock)	(None, 100, 256)	395776
transformer_block_21 (TransformerBlock)	(None, 100, 256)	395776
transformer_block_22 (TransformerBlock)	(None, 100, 256)	395776
transformer_block_23 (TransformerBlock)	(None, 100, 256)	395776
dense_147 (Dense)	(None, 100, 100)	25700
Total params: 2,451,556		
Trainable params: 2,451,556		
Non-trainable params: 0		

모델 생성 – GPT3

▪ Create model – GPT3

```
# gpt3
vocab_size = 50257 # Only consider the top 20k words, 50,257
maxlen = 2048 # Max sequence size, 1024
embed_dim = 12288 # Embedding size for each token, 768
num_heads = 96 # Number of attention heads, 12
feed_forward_dim = 12288 # Hidden layer size in feed forward network inside transformer, 1024 or 2048
num_layers = 96
```

```
def create_model():
    inputs = layers.Input(shape=(maxlen,), dtype=tf.int32)
    embedding_layer = TokenAndPositionEmbedding(maxlen, vocab_size, embed_dim) # (batch_size, input_seq_length, embed_dim)
    x = embedding_layer(inputs)
    transformer_block = TransformerBlock(embed_dim, num_heads, feed_forward_dim) # (batch_size, input_seq_length, embed_dim)
    transformer_block_list = [TransformerBlock(embed_dim, num_heads, feed_forward_dim) for _ in range(num_layers)]
    # x = transformer_block(x)
```

```
ResourceExhaustedError: OOM when allocating tensor with shape[12288,12288] and type float on
/job:localhost/replica:0/task:0/device:GPU:0 by allocator GPU_0_bfc [Op:Add]
```

```
outputs = layers.Dense(vocab_size)(x)
model = keras.Model(inputs=inputs, outputs=[outputs, x])
# 훈련 데이터의 label(target)이 one-hot vector 이면 CategoricalCrossentropy
# 훈련 데이터의 label(target)이 정수이면 SparseCategoricalCrossentropy
loss_fn = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
model.compile(
    "adam", loss=[loss_fn, None],
) # No loss and optimization based on word embeddings from transformer block
return model
```

```
model = create_model()
model.summary()
```

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}
GPT-3 Small	125M	12	768	12	64
GPT-3 Medium	350M	24	1024	16	64
GPT-3 Large	760M	24	1536	16	96
GPT-3 XL	1.3B	24	2048	24	128
GPT-3 2.7B	2.7B	32	2560	32	80
GPT-3 6.7B	6.7B	32	4096	32	128
GPT-3 13B	13.0B	40	5140	40	128
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128

Tesla T4 16G
GPT-3 XL까지 가능

데이터 준비

▪ Data down & processing

```
!curl -0 https://ai.stanford.edu/~amaas/data/sentiment/aclImdb_v1.tar.gz
!tar -xf aclImdb_v1.tar.gz

batch_size = 10

# The dataset contains each review in a separate text file
# The text files are present in four different folders
# Create a list all files
filenames = []
directories = [
    "aclImdb/train/pos",
    "aclImdb/train/neg",
    "aclImdb/test/pos",
    "aclImdb/test/neg",
]
for dir in directories:
    for f in os.listdir(dir)[:100]:
        filenames.append(os.path.join(dir, f))

print(f"{len(filenames)} files")

# Create dataset from text files
random.shuffle(filenames)
text_ds = tf.data.TextLineDataset(filenames)
text_ds = text_ds.shuffle(buffer_size=256)
text_ds = text_ds.batch(batch_size)

def custom_standardization(input_string):
    """ Remove html line-break tags and handle punctuation """
    lowercased = tf.strings.lower(input_string)
    stripped_html = tf.strings.regex_replace(lowercased, "<br />", " ")
    return tf.strings.regex_replace(stripped_html, f"([{string.punctuation}])", r" \1")

# Create vectorization layer and adapt it to the text
vectorize_layer = TextVectorization(
    standardize=custom_standardization,
    max_tokens=vocab_size - 1,
    output_mode="int",
    output_sequence_length=maxlen + 1,
)
vectorize_layer.adapt(text_ds)
vocab = vectorize_layer.get_vocabulary() # To get words back from token indices
```

▪ Imdb 데이터셋 다운로드(80MB)

▪ 압축해제 (210MB)



▪ 데이터 load&처리

학습 준비

- Data down & processing

```
# Tokenize starting prompt
word_to_index = {}
for index, word in enumerate(vocab):
    word_to_index[word] = index

start_prompt = "this movie is"
start_tokens = [word_to_index.get(_, 1) for _ in start_prompt.split()]
print(start_prompt, '=>', start_tokens)
num_tokens_generated = 40
text_gen_callback = TextGenerator(num_tokens_generated, start_tokens, vocab)
```

this movie is -> [13, 16, 9]

학습 준비

▪ Data down & processing

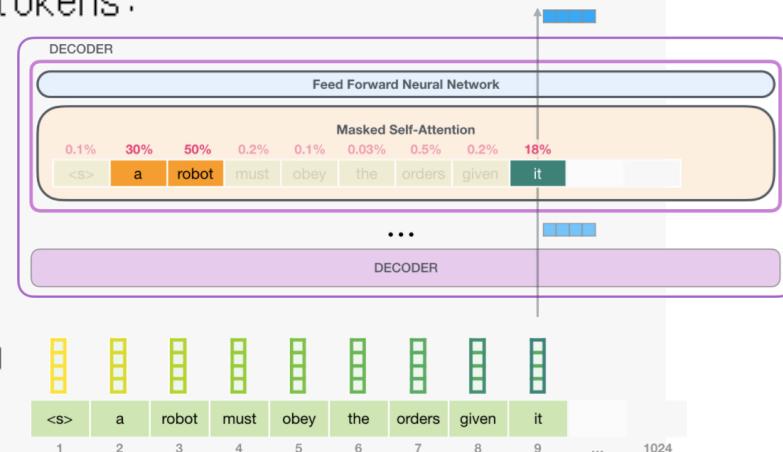
```
num_tokens_generated = 0
tokens_generated = []
while num_tokens_generated <= self.max_tokens:
    pad_len = maxlen - len(start_tokens)
    sample_index = len(start_tokens) - 1
    if pad_len < 0:
        x = start_tokens[:maxlen]
        sample_index = maxlen - 1
    elif pad_len > 0:
        x = start_tokens + [0] * pad_len
    else:
        x = start_tokens
    x = np.array([x])
    y, _ = self.model.predict(x)
    sample_token = self.sample_from(y[0][sample_index])
    tokens_generated.append(sample_token)
    start_tokens.append(sample_token)
    num_tokens_generated = len(tokens_generated)
    txt = " ".join([
        self.detokenize(_) for _ in self.start_tokens + tokens_generated
    ])
    print(txt)

    # Masked Self-Attention Diagram
    # DECODER
    # Feed Forward Neural Network
    # Masked Self-Attention
    # 0.1% 30% 50% 0.2% 0.1% 0.03% 0.5% 0.2% 18%
    # <s> a robot must obey the orders given it
    # ...
    # DECODER
    # 1 2 3 4 5 6 7 8 9 ...
    # <s> a robot must obey the orders given it
    # 1024
```

이전 단어들로 다음 단어 생성

생성된 단어가 다음 생성에 입력으로

'23



이전 단어들로 다음 단어 생성

생성된 단어가 다음 생성에 입력으로

Train : Colab에서 50분 소요

```
model.fit(text_ds, verbose=2, epochs=30, callbacks=[text_gen_callback])
model.save_weights("GPT_weight.h5") # 모델 save
model.load_weights("GPT_weight.h5") # 모델 load
```

Epoch 1/30

500/500 - 95s - loss: 5.3673 - dense_12_loss: 5.3673

i have a seen the first time , i have seen . the first , i 'm a lot better than a good thing [UNK] , i 'm sure , this movie was a good film

...

Epoch 30/30

500/500 - 93s - loss: 3.3402 - dense_12_loss: 3.3402

i have a lot of movies that have seen . i was very disappointed with this movie . it was a shame , because the plot was not the movie , but it was so bad , so i was surprised that it was

Inference

```
▶ class inference():
```

```
    start_prompt = 'i have a'  
    model_inference = inference(model, max_tokens=15, index_to_word=vocab, word_to_index=word_to_index, top_k=10)  
    model_inference.generate_sentence(start_prompt)
```

몇 단어 생성



후보 몇 개중 선택



```
[i', 'have', 'a'] -> hard  
[i', 'have', 'a', 'hard'] -> time  
[i', 'have', 'a', 'hard', 'time'] -> sitting  
[i', 'have', 'a', 'hard', 'time', 'sitting'] -> on  
[i', 'have', 'a', 'hard', 'time', 'sitting', 'on'] -> this  
[i', 'have', 'a', 'hard', 'time', 'sitting', 'on', 'this'] -> show  
[i', 'have', 'a', 'hard', 'time', 'sitting', 'on', 'this', 'show'] -> for  
[i', 'have', 'a', 'hard', 'time', 'sitting', 'on', 'this', 'show', 'for'] -> me  
[i', 'have', 'a', 'hard', 'time', 'sitting', 'on', 'this', 'show', 'for', 'me'] -> ,  
[i', 'have', 'a', 'hard', 'time', 'sitting', 'on', 'this', 'show', 'for', 'me', ','] -> and  
[i', 'have', 'a', 'hard', 'time', 'sitting', 'on', 'this', 'show', 'for', 'me', ', ', 'and'] -> i  
[i', 'have', 'a', 'hard', 'time', 'sitting', 'on', 'this', 'show', 'for', 'me', ', ', 'and', 'i'] -> am  
[i', 'have', 'a', 'hard', 'time', 'sitting', 'on', 'this', 'show', 'for', 'me', ', ', 'and', 'i', 'am'] -> very  
[i', 'have', 'a', 'hard', 'time', 'sitting', 'on', 'this', 'show', 'for', 'me', ', ', 'and', 'i', 'am', 'very'] -> happy  
[i', 'have', 'a', 'hard', 'time', 'sitting', 'on', 'this', 'show', 'for', 'me', ', ', 'and', 'i', 'am', 'very', 'happy'] -> to  
[i', 'have', 'a', 'hard', 'time', 'sitting', 'on', 'this', 'show', 'for', 'me', ', ', 'and', 'i', 'am', 'very', 'happy', 'to'] -> see
```

i have a hard time sitting on this show for me , and i am very happy to see

Inference

```
model_inference.generate_sentence('i need your')
```

['i', 'need', 'your'] -> own
['i', 'need', 'your', 'own'] -> this
['i', 'need', 'your', 'own', 'this'] -> show
['i', 'need', 'your', 'own', 'this', 'show'] -> because
['i', 'need', 'your', 'own', 'this', 'show', 'because'] -> i
['i', 'need', 'your', 'own', 'this', 'show', 'because', 'i'] -> thought
['i', 'need', 'your', 'own', 'this', 'show', 'because', 'i', 'thought'] -> this
['i', 'need', 'your', 'own', 'this', 'show', 'because', 'i', 'thought', 'this'] -> show
['i', 'need', 'your', 'own', 'this', 'show', 'because', 'i', 'thought', 'this', 'show'] -> was
['i', 'need', 'your', 'own', 'this', 'show', 'because', 'i', 'thought', 'this', 'show', 'was'] -> so
['i', 'need', 'your', 'own', 'this', 'show', 'because', 'i', 'thought', 'this', 'show', 'was', 'so'] -> awful
['i', 'need', 'your', 'own', 'this', 'show', 'because', 'i', 'thought', 'this', 'show', 'was', 'so', 'awful'] -> .
['i', 'need', 'your', 'own', 'this', 'show', 'because', 'i', 'thought', 'this', 'show', 'was', 'so', 'awful', '.'] -> this
['i', 'need', 'your', 'own', 'this', 'show', 'because', 'i', 'thought', 'this', 'show', 'was', 'so', 'awful', '.', 'this'] -> was
['i', 'need', 'your', 'own', 'this', 'show', 'because', 'i', 'thought', 'this', 'show', 'was', 'so', 'awful', '.', 'this', 'was'] -> a
['i', 'need', 'your', 'own', 'this', 'show', 'because', 'i', 'thought', 'this', 'show', 'was', 'so', 'awful', '.', 'this', 'was', 'a'] -> very

i need your own this show because i thought this show was so awful . this was a very

Inference

```
model_inference.generate_sentence('show me the money')
```

```
['show', 'me', 'the', 'money'] -> is
['show', 'me', 'the', 'money', 'is'] -> on
['show', 'me', 'the', 'money', 'is', 'on'] -> a
['show', 'me', 'the', 'money', 'is', 'on', 'a'] -> tv
['show', 'me', 'the', 'money', 'is', 'on', 'a', 'tv'] -> channel
['show', 'me', 'the', 'money', 'is', 'on', 'a', 'tv', 'channel'] -> .
['show', 'me', 'the', 'money', 'is', 'on', 'a', 'tv', 'channel', ':'] -> i
['show', 'me', 'the', 'money', 'is', 'on', 'a', 'tv', 'channel', ':', 'i'] -> think
['show', 'me', 'the', 'money', 'is', 'on', 'a', 'tv', 'channel', ':', 'i', 'think'] -> this
['show', 'me', 'the', 'money', 'is', 'on', 'a', 'tv', 'channel', ':', 'i', 'think', 'this'] -> show
['show', 'me', 'the', 'money', 'is', 'on', 'a', 'tv', 'channel', ':', 'i', 'think', 'this', 'show'] -> was
['show', 'me', 'the', 'money', 'is', 'on', 'a', 'tv', 'channel', ':', 'i', 'think', 'this', 'show', 'was'] -> the
['show', 'me', 'the', 'money', 'is', 'on', 'a', 'tv', 'channel', ':', 'i', 'think', 'this', 'show', 'was', 'the'] -> worst
['show', 'me', 'the', 'money', 'is', 'on', 'a', 'tv', 'channel', ':', 'i', 'think', 'this', 'show', 'was', 'the', 'worst'] -> show
['show', 'me', 'the', 'money', 'is', 'on', 'a', 'tv', 'channel', ':', 'i', 'think', 'this', 'show', 'was', 'the', 'worst', 'show'] -> ever
['show', 'me', 'the', 'money', 'is', 'on', 'a', 'tv', 'channel', ':', 'i', 'think', 'this', 'show', 'was', 'the', 'worst', 'show', 'ever'] -> .
```

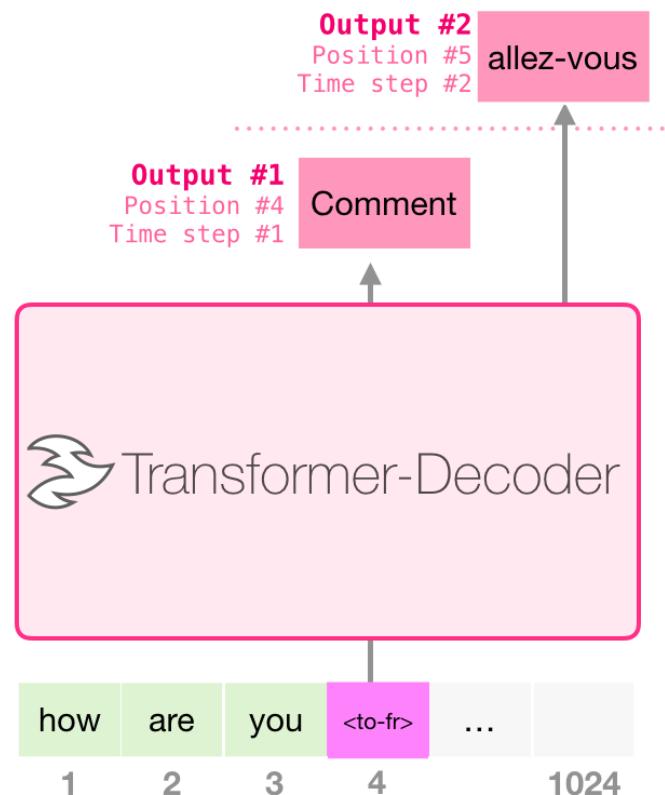
show me the money is on a tv channel . i think this show was the worst show ever .

Beyond LM

■ NMT

Training Dataset

I	am	a	student	<to-fr>	je	suis	étudiant
let	them	eat	cake	<to-fr>	Qu'ils	mangent	de
good	morning	<to-fr>	Bonjour				



Beyond LM

■ Summarization

 WIKIPEDIA
The Free Encyclopedia

Not logged in | Talk | Contributions | Create account | Log in

Article | Talk | Read | Edit | View history | Search Wikipedia |

Positronic brain

From Wikipedia, the free encyclopedia
(Redirected from Positronic robot)

This article is about a fictional technological device. For the manufacturing company based in Springfield, Missouri, see [Positronic \(company\)](#).

This article needs additional citations for verification. Please help improve this article by adding citations to reliable sources. Unsourced material may be challenged and removed.
Find sources: "Positronic brain" – news · newspapers · books · scholar · JSTOR (July 2008) [Learn how and when to remove this template message](#)

A **positronic brain** is a fictional technological device, originally conceived by science fiction writer Isaac Asimov.^[1] It functions as a central processing unit (CPU) for robots, and, in some unspecified way, provides them with a form of consciousness recognizable to humans. When Asimov wrote his first robot stories in 1939 and 1940, the positron was a newly discovered particle, and so the buzz word positron added a contemporary gloss of popular science to the concept. The short story "Runaround", by Asimov, elaborates on the concept, in the context of his fictional Three Laws of Robotics.

Contents [edit]

- 1 Conceptual overview
- 2 In Allen's Trilogy
- 3 References in other fiction and films
 - 3.1 *Attack of the Cosmic Go To Mars*
 - 3.2 *The Avengers*
 - 3.3 *Doctor Who*
 - 3.4 *Star Trek*
 - 3.5 *Perry Rhodan*
 - 3.6 *I, Robot* 2004 film
 - 3.7 *Star Wars*
 - 3.8 *Rock Rogers in the 25th Century*
 - 3.9 *Mystery Science Theater 3000*
 - 3.10 *Specimen*
 - 3.11 *Starless*
- 4 References
- 5 External links

Conceptual overview [edit]

Asimov remained vague about the technical details of positronic brains except to assert that their substructure was formed from an alloy of platinum and indium. They were said to be vulnerable to radiation and apparently involve a type of volatile memory (since robots in storage required a power source keeping their brains "alive"). The focus of Asimov's stories was directed more towards the software of robots—such as the Three Laws of Robotics—than the hardware in which it was implemented, although it is stated in his stories that to create a positronic brain without the Three Laws, it would have been necessary to spend years redesigning the fundamental approach towards the brain itself.

While his stories of robots on Earth and their development by U.S. Robots, Asimov's positronic brain is less of a plot device and more of a technological item worthy of study.

A positronic brain cannot ordinarily be built without incorporating the Three Laws, any modification thereof would drastically modify robot behavior. Behavioral dilemmas resulting from conflicting potentials set by inexperienced and/or malicious users of the robot for the Three Laws make up the bulk of Asimov's stories concerning robots. They are resolved by applying the science of logic and psychology together with mathematics, the supreme solution finder being Dr. Susan Calvin, Chief Robopsychologist of U.S. Robots.

The Three Laws are also a *boomerang* in brain sophistication. Very complex brains designed to handle world economy interpret the First Law in expanded sense to include humanity as opposed to a single human; in Asimov's later works like *Robots and Empire* this is referred to as the "Zenith Law". At least one brain constructed as a calculating machine, as opposed to a robot control circuit, was designed to have a flexible, childlike personality so that it was able to pursue difficult problems without the Three Laws inhibiting it completely. Specialized brains created for overseeing world economies were stated to have no personality at all.

Under specific conditions, the Three Laws can be circumvented, with the modification of the actual robot design.

- Robots that are of low enough value can have the *Third Law* deleted; they do not have to protect themselves from harm, and the brain size can be reduced by half.
- Robots that do not require orders from a human being may have the *Second Law* deleted, and therefore require smaller brains again, providing they do not require the *Third Law*.
- Robots that are disposable, cannot receive orders from a human being and are not able to harm a human, will not require even the *First Law*. The sophistication of positronic circuitry renders a brain so small that it could comfortably fit within the skull of an insect.

Robots of the latter type directly parallel contemporary industrial robotics practices, through real-life robots to contain safety sensors and systems, in a concern for human safety (a weak form of the First Law, the robot is a safe tool to use, but has no "judgment", which is implicit in Asimov's own stories).

In Allen's trilogy [edit]

Several robot stories have been written by other authors following Asimov's death. For example, in Roger Macdonald Allen's *Callahan Trilogy*, a Space robot called Gubber Anshaw invents the *gravitronic brain*. It offers speed and capacity improvements over traditional positronic designs, but the strong influence of tradition make robotics labs reject Anshaw's work. Only one robot, Freddi Leving, chooses to adopt gravitronics, because it offers her a blank slate on which she could explore alternatives to the Three Laws. Because they are not dependent upon centuries of earlier research, gravitronic brains can be programmed with the standard Laws, variations of the Laws, or even empty pathways which specify no Laws at all.

Not logged in | Talk | Contributions | Create account | Log in

Article | Talk | Read | Edit | View history | Search Wikipedia |

Positronic brain

From Wikipedia, the free encyclopedia
(Redirected from Positronic robot)

This article is about a fictional technological device. For the manufacturing company based in Springfield, Missouri, see [Positronic \(company\)](#).

This article needs additional citations for verification. Please help improve this article by adding citations to reliable sources. Unsourced material may be challenged and removed.
Find sources: "Positronic brain" – news · newspapers · books · scholar · JSTOR (July 2008) [Learn how and when to remove this template message](#)

A **positronic brain** is a fictional technological device, originally conceived by science fiction writer Isaac Asimov.^[1] It functions as a central processing unit (CPU) for robots, and, in some unspecified way, provides them with a form of consciousness recognizable to humans. When Asimov wrote his first robot stories in 1939 and 1940, the positron was a newly discovered particle, and so the buzz word positron added a contemporary gloss of popular science to the concept. The short story "Runaround", by Asimov, elaborates on the concept, in the context of his fictional Three Laws of Robotics.

SUMMARY

Contents [edit]

- 1 Conceptual overview
- 2 In Allen's Trilogy
- 3 References in other fiction and films
 - 3.1 *Attack of the Cosmic Go To Mars*
 - 3.2 *The Avengers*
 - 3.3 *Doctor Who*
 - 3.4 *Star Trek*
 - 3.5 *Perry Rhodan*
 - 3.6 *I, Robot* 2004 film
 - 3.7 *Star Wars*
 - 3.8 *Rock Rogers in the 25th Century*
 - 3.9 *Mystery Science Theater 3000*
 - 3.10 *Specimen*
 - 3.11 *Starless*
- 4 References
- 5 External links

Conceptual overview [edit]

Asimov remained vague about the technical details of positronic brains except to assert that their substructure was formed from an alloy of platinum and indium. They were said to be vulnerable to radiation and apparently involve a type of volatile memory (since robots in storage required a power source keeping their brains "alive"). The focus of Asimov's stories was directed more towards the software of robots—such as the Three Laws of Robotics—than the hardware in which it was implemented, although it is stated in his stories that to create a positronic brain without the Three Laws, it would have been necessary to spend years redesigning the fundamental approach towards the brain itself.

While his stories of robotics on Earth and their development by U.S. Robots, Asimov's positronic brain is less of a plot device and more of a technological item worthy of study.

A positronic brain cannot ordinarily be built without incorporating the Three Laws, any modification thereof would drastically modify robot behavior. Behavioral dilemmas resulting from conflicting potentials set by inexperienced and/or malicious users of the robot for the Three Laws make up the bulk of Asimov's stories concerning robots. They are resolved by applying the science of logic and psychology together with mathematics, the supreme solution finder being Dr. Susan Calvin, Chief Robopsychologist of U.S. Robots.

The Three Laws are also a *boomerang* in brain sophistication. Very complex brains designed to handle world economy interpret the First Law in expanded sense to include humanity as opposed to a single human; in Asimov's later works like *Robots and Empire* this is referred to as the "Zenith Law". At least one brain constructed as a calculating machine, as opposed to a robot control circuit, was designed to have a flexible, childlike personality so that it was able to pursue difficult problems without the Three Laws inhibiting it completely.

ARTICLE

Under specific conditions, the Three Laws can be circumvented, with the modification of the actual robot design.

- Robots that are of low enough value can have the *Third Law* deleted; they do not have to protect themselves from harm, and the brain size can be reduced by half.
- Robots that do not require orders from a human being may have the *Second Law* deleted, and therefore require smaller brains again, providing they do not require the *Third Law*.
- Robots that are disposable, cannot receive orders from a human being and are not able to harm a human, will not require even the *First Law*. The sophistication of positronic circuitry renders a brain as small that it could comfortably fit within the skull of an insect.

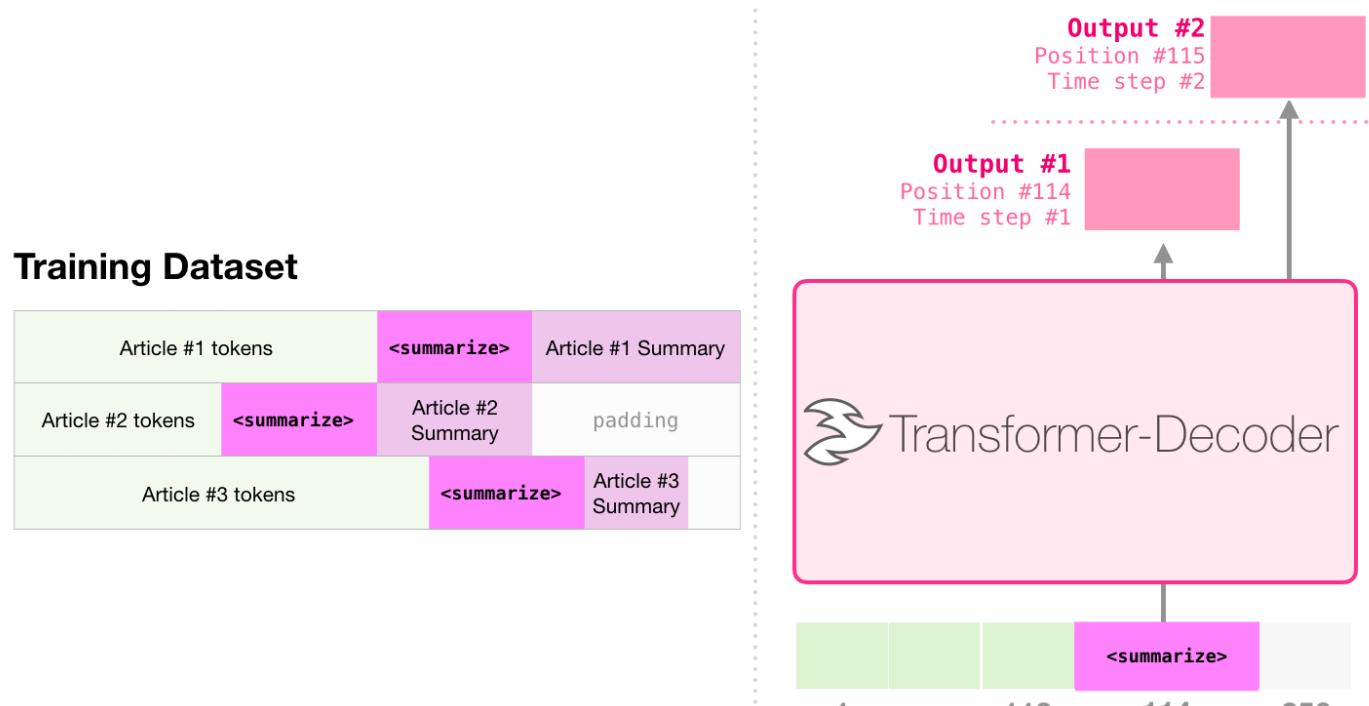
Robots of the latter type directly parallel contemporary industrial robotics practices, through real-life robots to contain safety sensors and systems, in a concern for human safety (a weak form of the First Law, the robot is a safe tool to use, but has no "judgment", which is implicit in Asimov's own stories).

In Allen's trilogy [edit]

Several robot stories have been written by other authors following Asimov's death. For example, in Roger Macdonald Allen's *Callahan Trilogy*, a Space robot called Gubber Anshaw invents the *gravitronic brain*. It offers speed and capacity improvements over traditional positronic designs, but the strong influence of tradition make robotics labs reject Anshaw's work. Only one robot, Freddi Leving, chooses to adopt gravitronics, because it offers her a blank slate on which she could explore alternatives to the Three Laws. Because they are not dependent upon centuries of earlier research, gravitronic brains can be programmed with the standard Laws, variations of the Laws, or even empty pathways which specify no Laws at all.

Beyond LM

▪ Summarization



감사합니다