



T5 and large language models: The good, the bad, and the ugly

권재선 송민경

목차

#01 Before T5

#02 T5 Model

#03 Other topics



01. Before T5



1. Before T5

Introduction

- Which transfer learning methods work best, and what happens when we scale them up?
어떠한 전이학습 방법이 제일 좋고, 크기를 키우면 어떻게 작용할 것인가?
- What about non-English pre-trained models?
영어가 아닌 언어로 사전 학습하는 것은 어떤가?
- How much knowledge does the model learn during pre-training?
Model0| 사전학습을 하면서 얼만큼의 정보를 학습할 수 있는가?
- Does the model memorize data during pre-training?
Model0| 사전학습을 하면서 얼만큼의 data를 기억할 수 있는가?
- Which Transformer modifications work best?
Transformer를 수정한 model 중 어떤 것이 성능이 제일 좋은가?

1. Before T5

Introduction

Unsupervised pre-training

The cabs __ the same rates as those
__ by horse-drawn cabs and were __
quite popular, __ the Prince of
Wales (the __ King Edward VII)
travelled in __. The cabs quickly
__ known as "hummingbirds" for __
noise made by their motors and their
distinctive black and __ livery.
Passengers __ __ the interior
fittings were __ when compared to
__ cabs but there __ some
complaints __ the __ lighting made
them too __ to those outside __.



Supervised fine-tuning

This movie is terrible! The acting
is bad and I was bored the entire
time. There was no plot and
nothing interesting happened. I
was really surprised since I had
very high expectations. I want 103
minutes of my life back!

negative

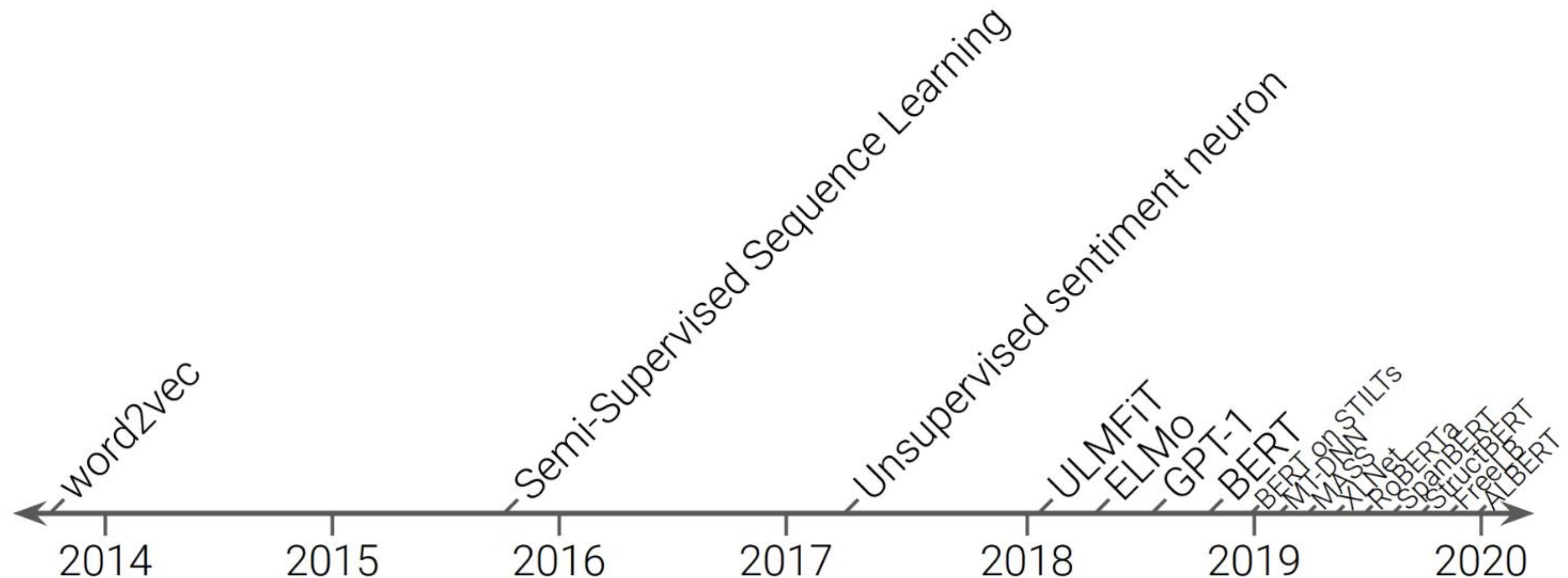
charged, used, initially, even,
future, became, the, yellow,
reported, that, luxurious,
horse-drawn, were that,
internal, conspicuous, cab

자연어처리에서의 transfer-learning이란?

- unsupervised objective를 통해 pretrain을 진행
- (그림 참고) 특정 단어를 masking 처리를 하여 어떠한 단어가 들어갈 것인지를 objective function으로 설정
- Pretrain을 진행하고 downstream supervised task에 맞게 fine-tuning을 진행

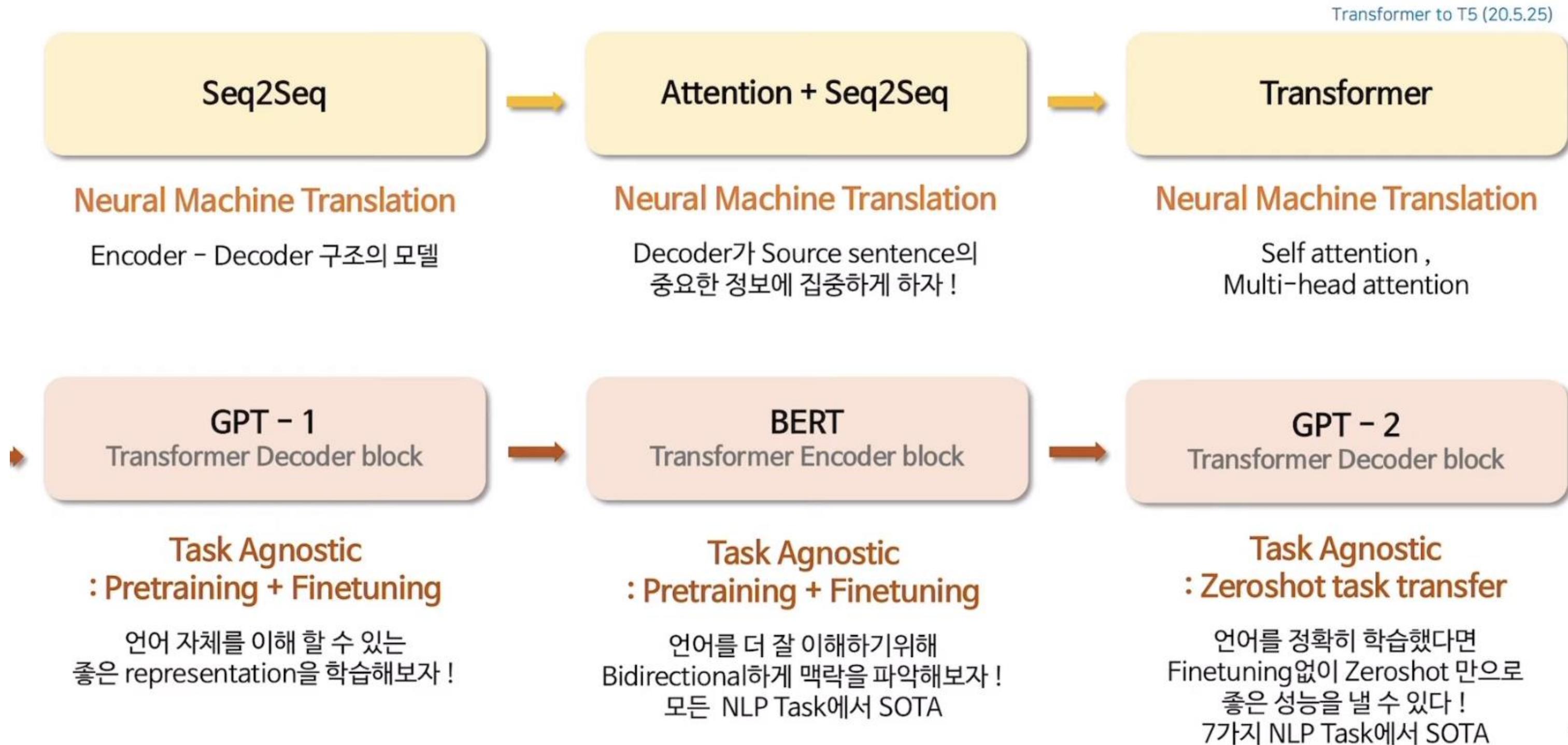
1. Before T5

Overview



1. Before T5

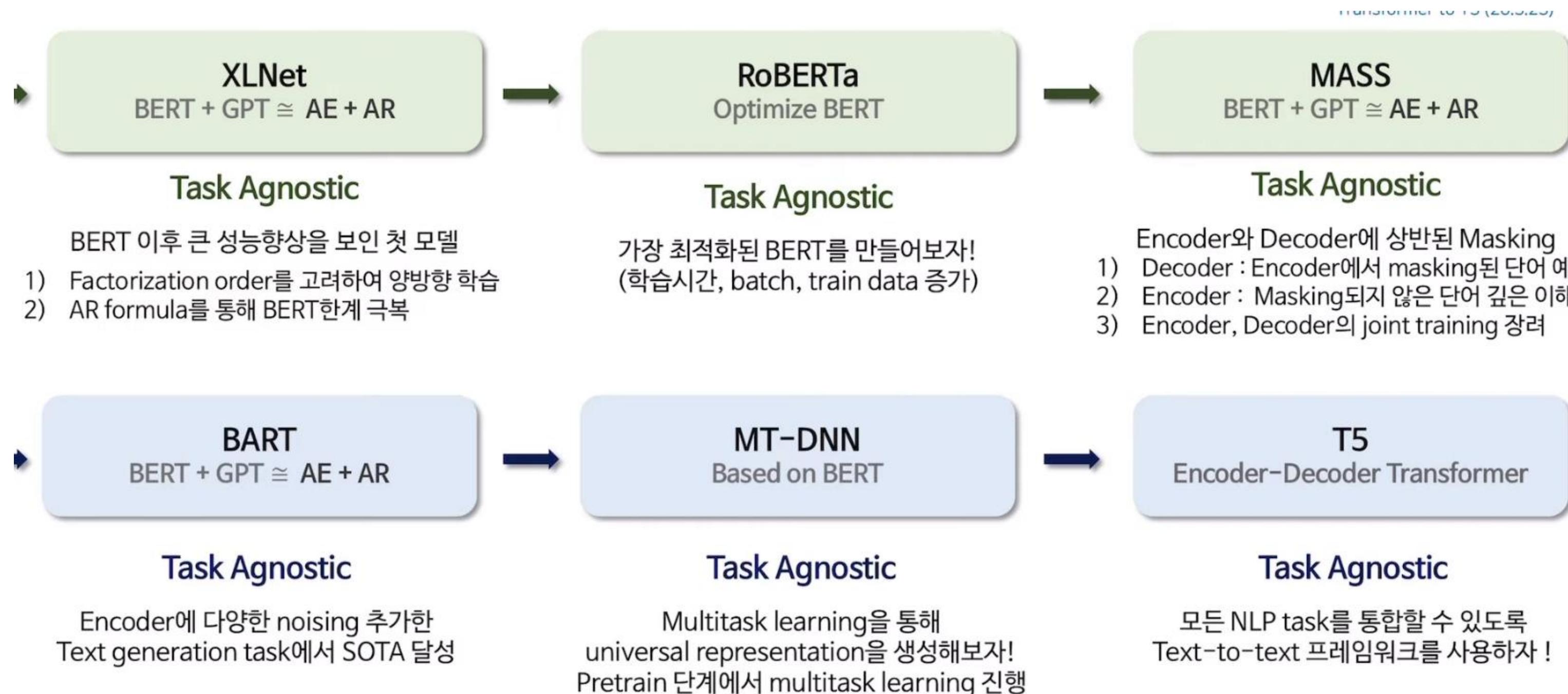
Overview



4

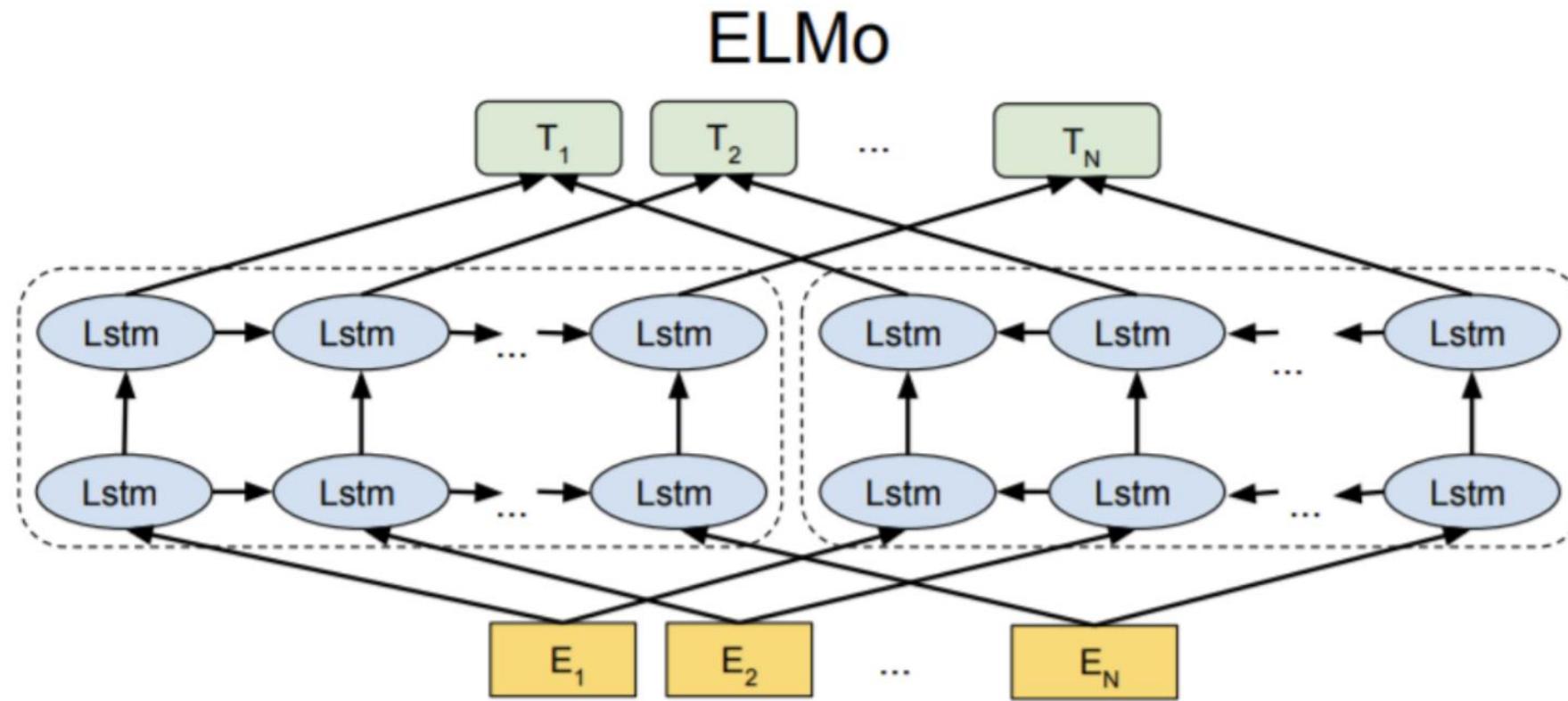
1. Before T5

Overview



1. Before T5

ELMo



ELMO가 발표되기 전 당시, GloVe, FastText, Word2Vec등 word embedding 방법들은 주변 context words를 학습할 때만 고려할 뿐, 다른 model에 input으로 사용할 때는 고려하지 않음

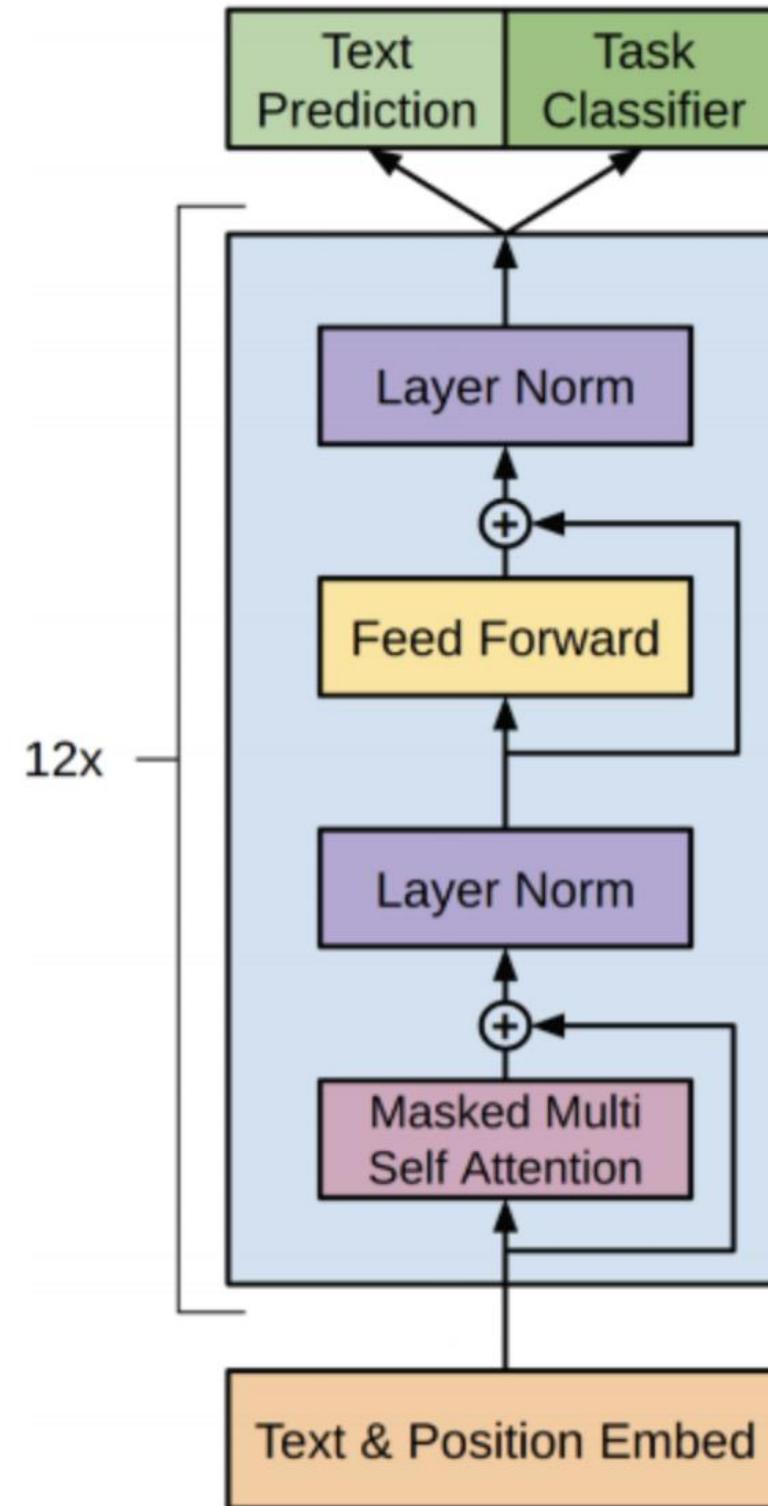
ex) word embedding을 통해 '사랑'이라는 단어에 대해 하나의 vector가 mapping되고 학습이 잘되었다면?

⇒ vector가 문맥에 맞는 다양한 의미를 가지고 있을 것, '사랑'이라는 의미는 주변 단어의 문맥에 대해 변할 수 있고 문맥에 맞지 않는 의미는 불필요하게 됨. 그러나 기존 방법으로는 하나의 vector로 space에 mapping되기 때문에 어느 의미를 내포하는지 알 수 없음

⇒ ELMO는 word embedding을 사용할 때, vector가 주변 단어와 맞는, 문맥 정보에 따라 변할 수 있도록 할 수 있게 해서 Contextualized Word Embedding이라는 표현

1. Before T5

GPT

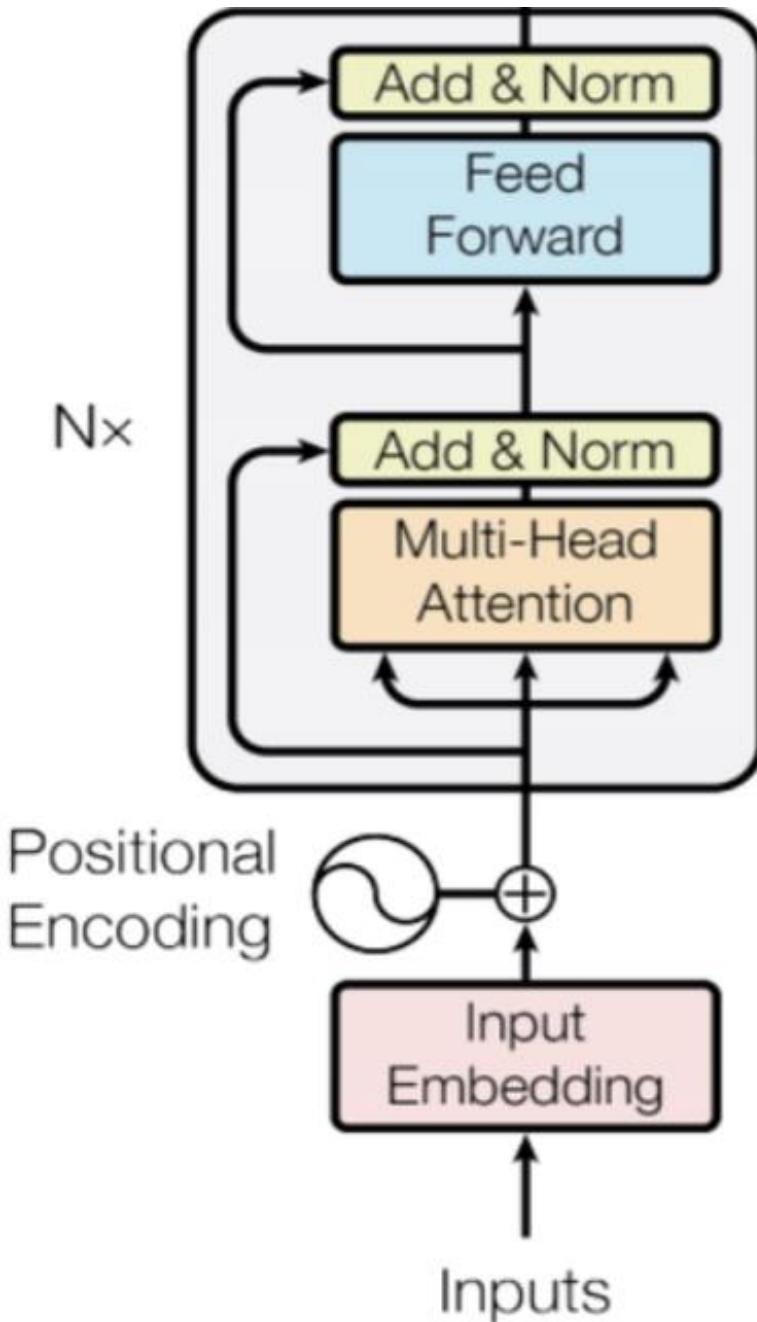


Open-AI에서 발표한 GPT(Generative Pre-Training)

- 현재 GPT2, GPT3까지 size가 커진 model이 발표됨
- GPT는 unlabeled corpus를 pre-train하고 각 task에 맞게 fine-tuning을 하는 language model
- GPT의 특징: **transformer의 decoder 구조만을 가지고 있어 auto-regressive하다는 점**

1. Before T5

BERT



Google에서 발표한 BERT
(Bidirectional Encoder Representations from Transformers)

- BERT는 GPT와 같이 다량의 **unlabeled corpus**를 통해 **사전학습**을 하고 task에 맞게 fine-tuning을 함
- 그러나 BERT는 **transformer의 encoder**로만 구성되어 있고 **양방향으로 학습**하여 auto encoding한 성격을 가짐
- BERT의 pre-train obejective function으로는 **MLM(Masked Language Model)과 NSP(Next Sentence Prediction)** 이용
- 현재 수많은 LM0I 등장 (RoBERTa, MASS, XLNet, BART, ...)

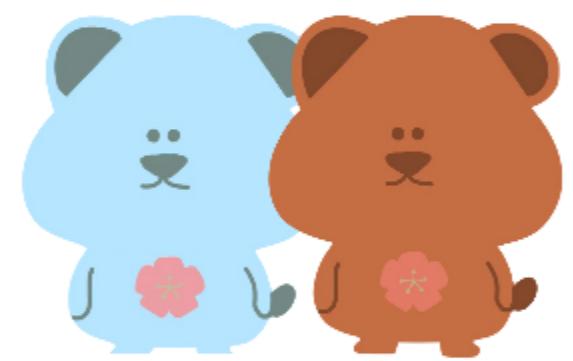
1. Before T5

Overview

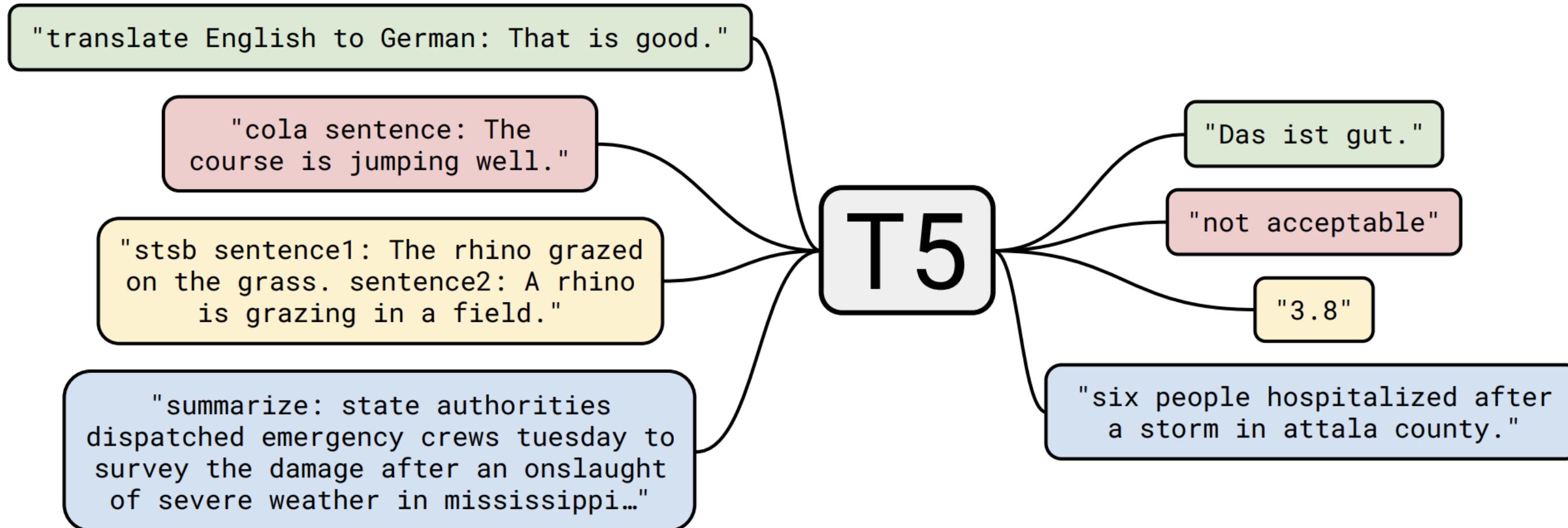
Given the current landscape
of transfer learning for NLP,
what works best? And how
far can we push the tools we
already have?

그래서 어떤 것이 제일 잘 맞는가 ? => T5의 다양한 실험을 통해 알 수 있음

#02 T5 Model



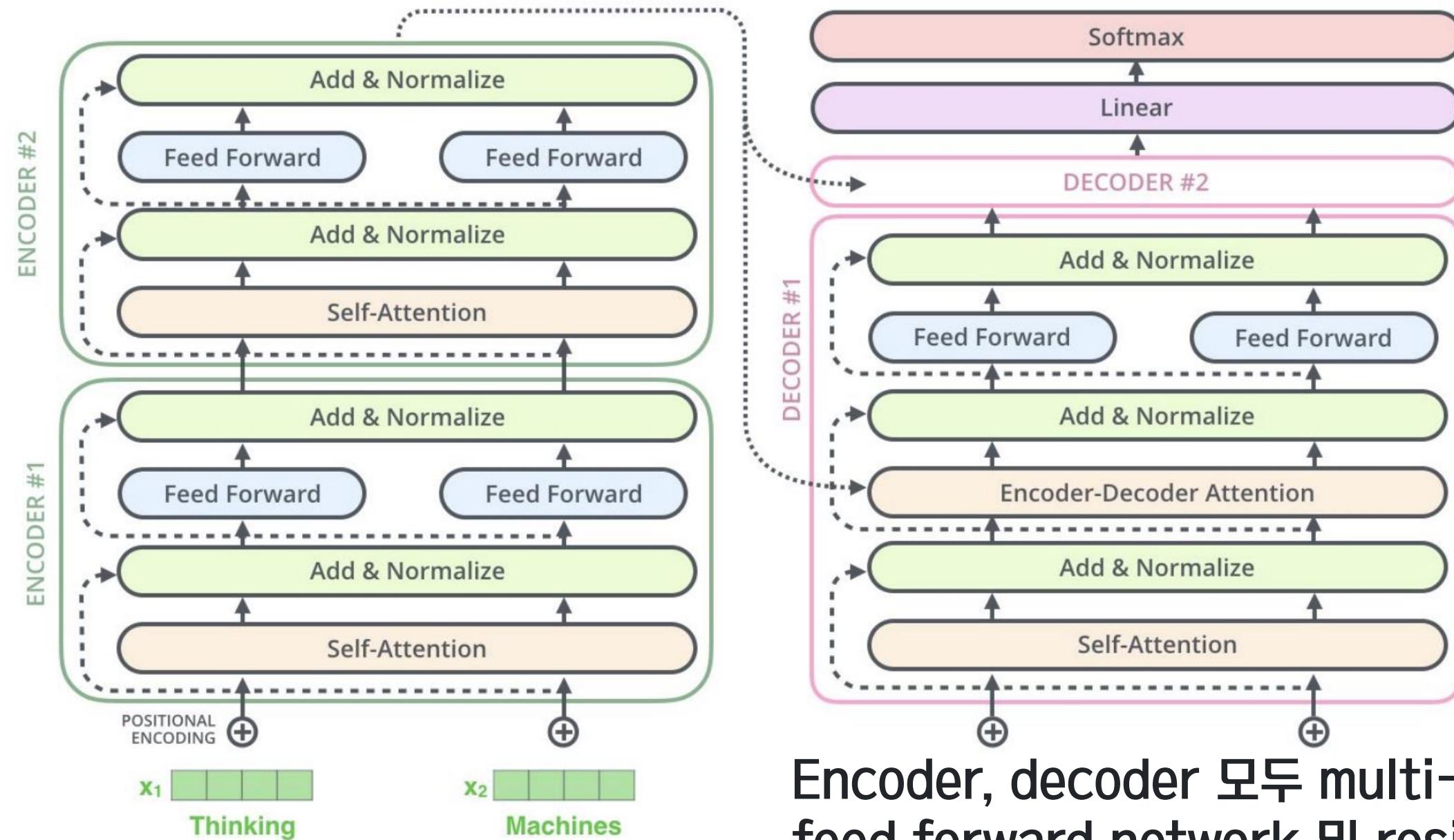
#1 pre-training



T5는 하나의 framework를 통해 여러 task를 처리할 수 있음

#1 pre-training

T5는 transformer encoder-decoder 구조



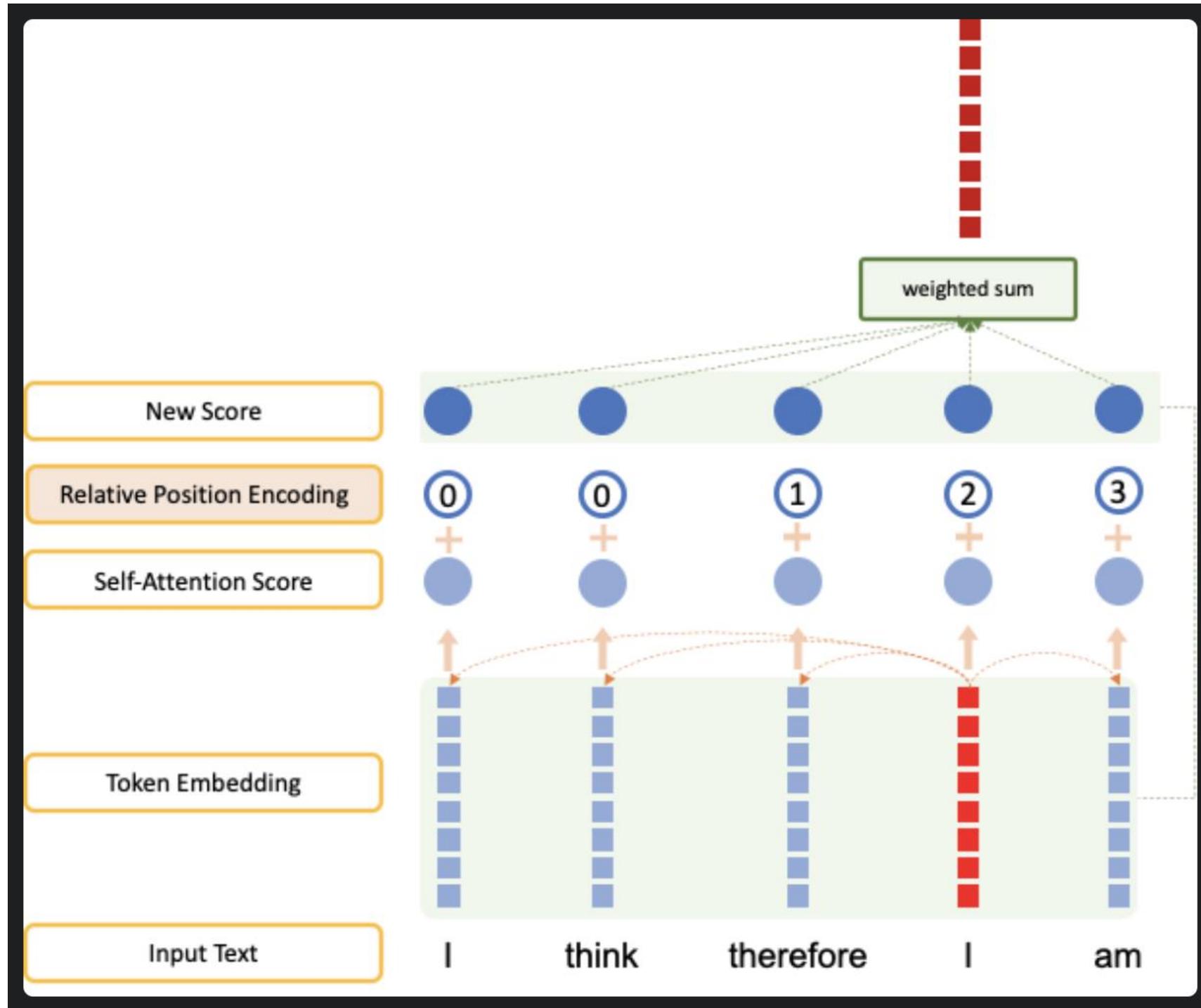
input token에 **relative positional encoding**을 사용

Encoder, decoder 모두 multi-head self-attention과
feed forward network 및 residual skip connection,
dropout 기법을 사용

input으로 들어오는 각 token의 위치 별로 동일한 encoding 값을 부여하고 attention 계산을 하는 것이 아니라,
self-attention 계산할 때 일정 범위 내의 token들에 relative positional encoding 값을 주는 것

#1 pre-training

relative positional encoding



'I think therefore I am' → self-attention layer

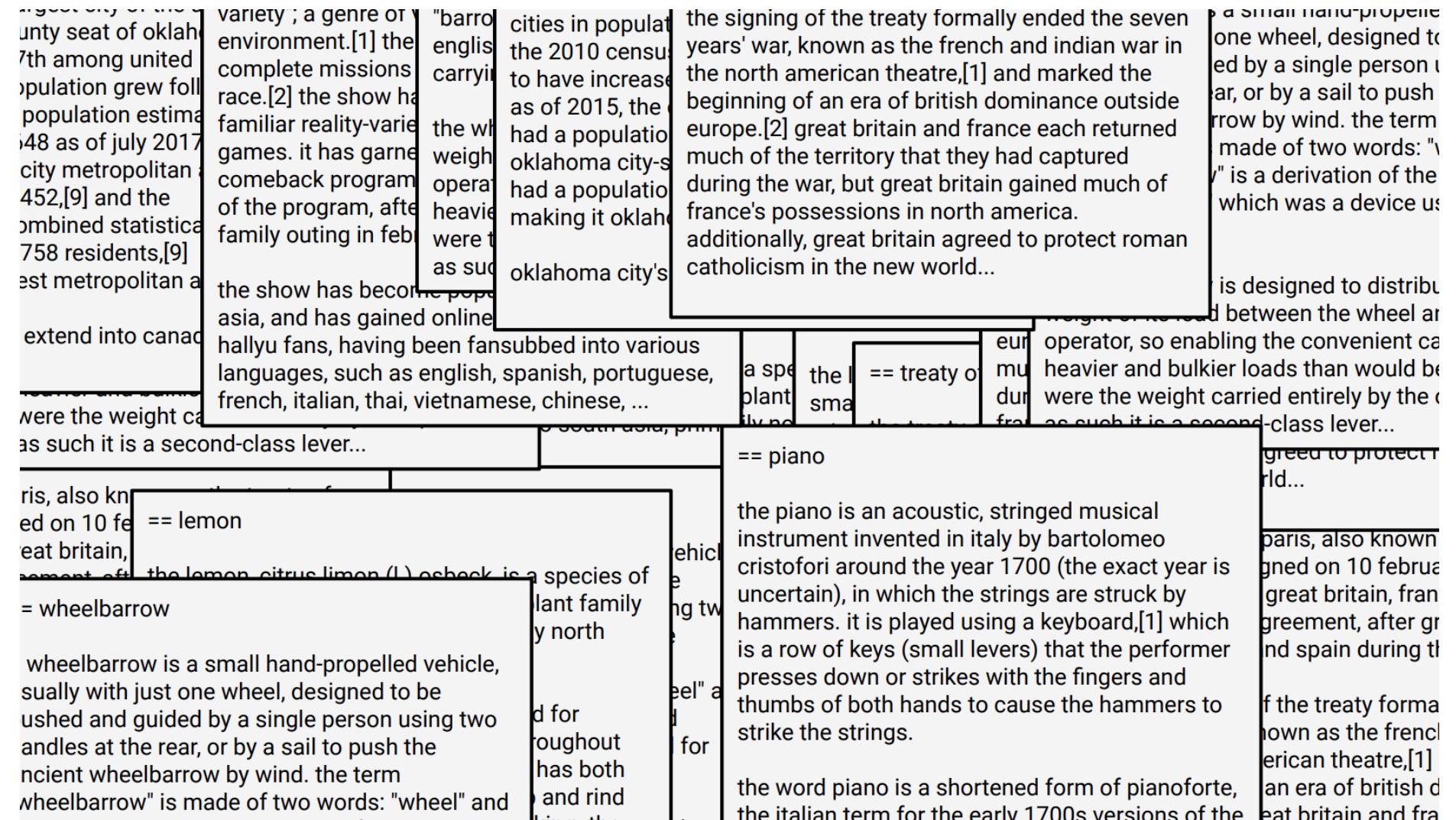
예시)

- 0 : i번째 token의 왼쪽 2번째 token
- 1 : i번째 token의 왼쪽 1번째 token
- 2 : i번째 token
- 3 : i번째 token의 오른쪽 1번째 token
- 4 : i번째 token의 오른쪽 2번째 token

만약 offset = 2라면, offset 범위를 넘어선 token에는 0을 부여

#1 pre-training

Common Crawl Web Extracted Text



pre-train에 사용되는 unlabeled corpus dataset
: Transfer-learning의 중요 요소

Pre-train 시, data 크기를 키우는 효과를 정확하게
파악하려면 좋은 품질의 data +
여러 domain의 dataset 필요

기존 Wikipedia는 품질은 우수한 data
But, 다양성이 부족

Common Crawl web scrapping data
: 크기가 크고 다양함
But, 품질이 낮다는 단점이 존재

⇒ google에서는 Wikipedia보다 2배 큰 새로운 Common Crawl 버전인 C4(Colossal Clean Crawled Corpus)를 사용
⇒ unlabeled data에 대한 여러 실험을 위해 여러 기준에 따라 전처리를 진행

#1 pre-training

#1 pre-training objective(목적함수)

Original text

Thank you ~~for inviting~~ me to your party ~~last~~ week.

Inputs

Thank you <X> me to your party <Y> week.

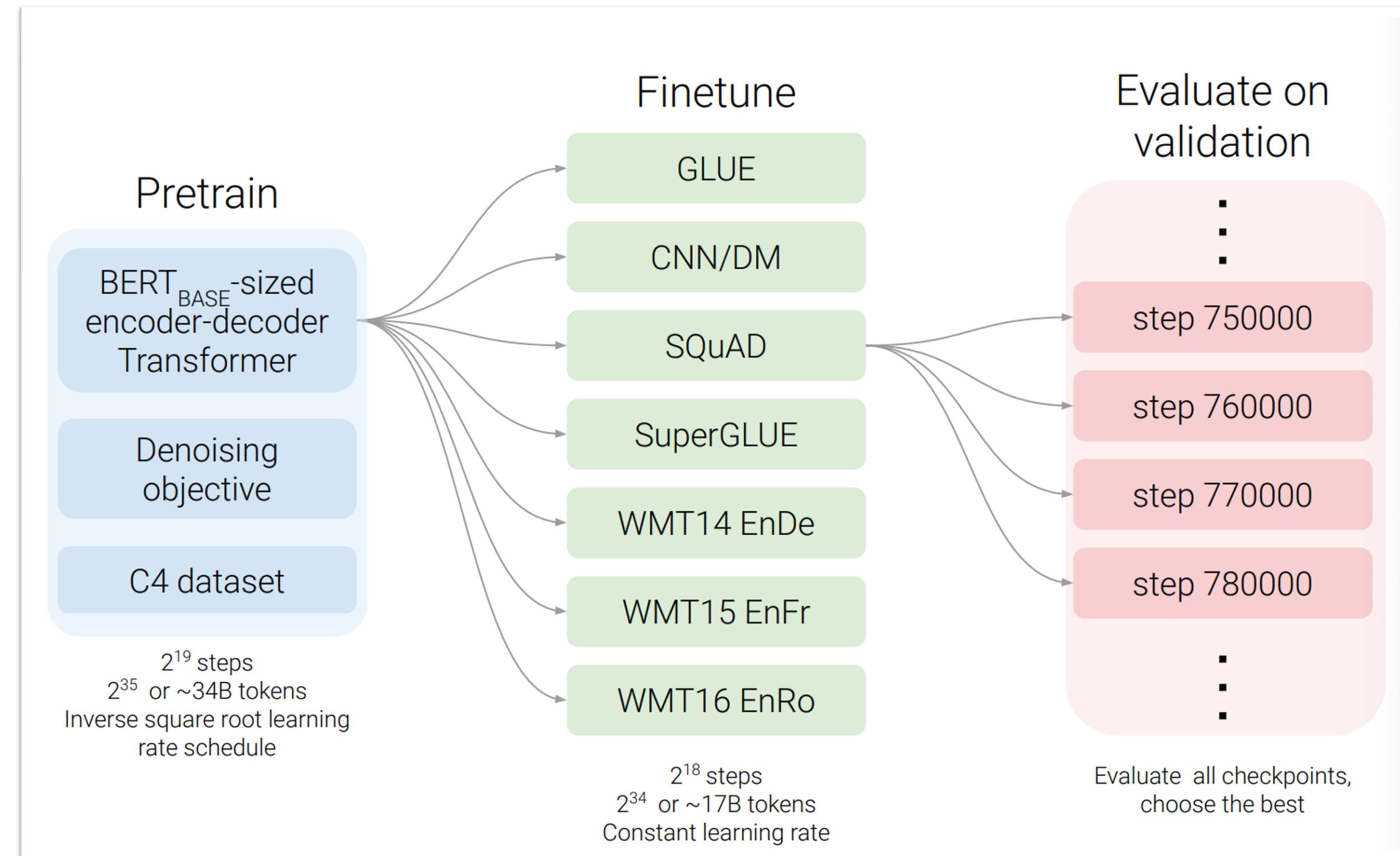
Targets

<X> for inviting <Y> last <Z>

- ⇒ original text가 있을 때, 무작위로 token을 설정하고 masking 처리
- ⇒ 하나의 random token을 masking 하는 것이 아닌 연속된 token을 하나로 masking 처리
- ⇒ 해당 token을 [MASK]로 변환하는 것이 아닌 sentinel ID token으로 대체
- ⇒ 마지막으로 input에서 masking되지 않은 부분을 target에서 맞춰야함

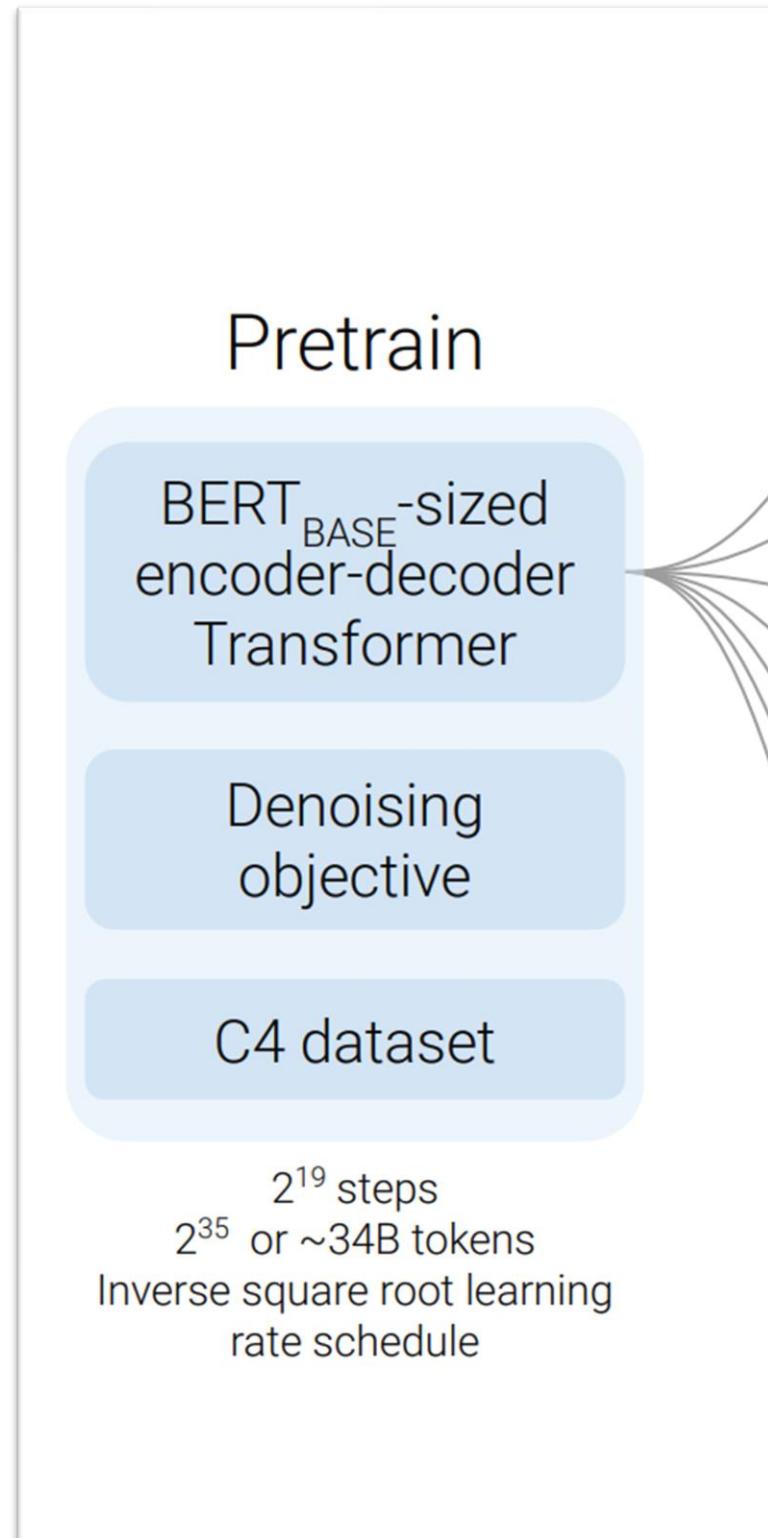
#2 baseline experiment procedure: pretrain - finetune - evaluate

#2 baseline experiment procedure: pretrain - finetune - evaluate



#2 baseline experiment procedure: pretrain - finetune - evaluate

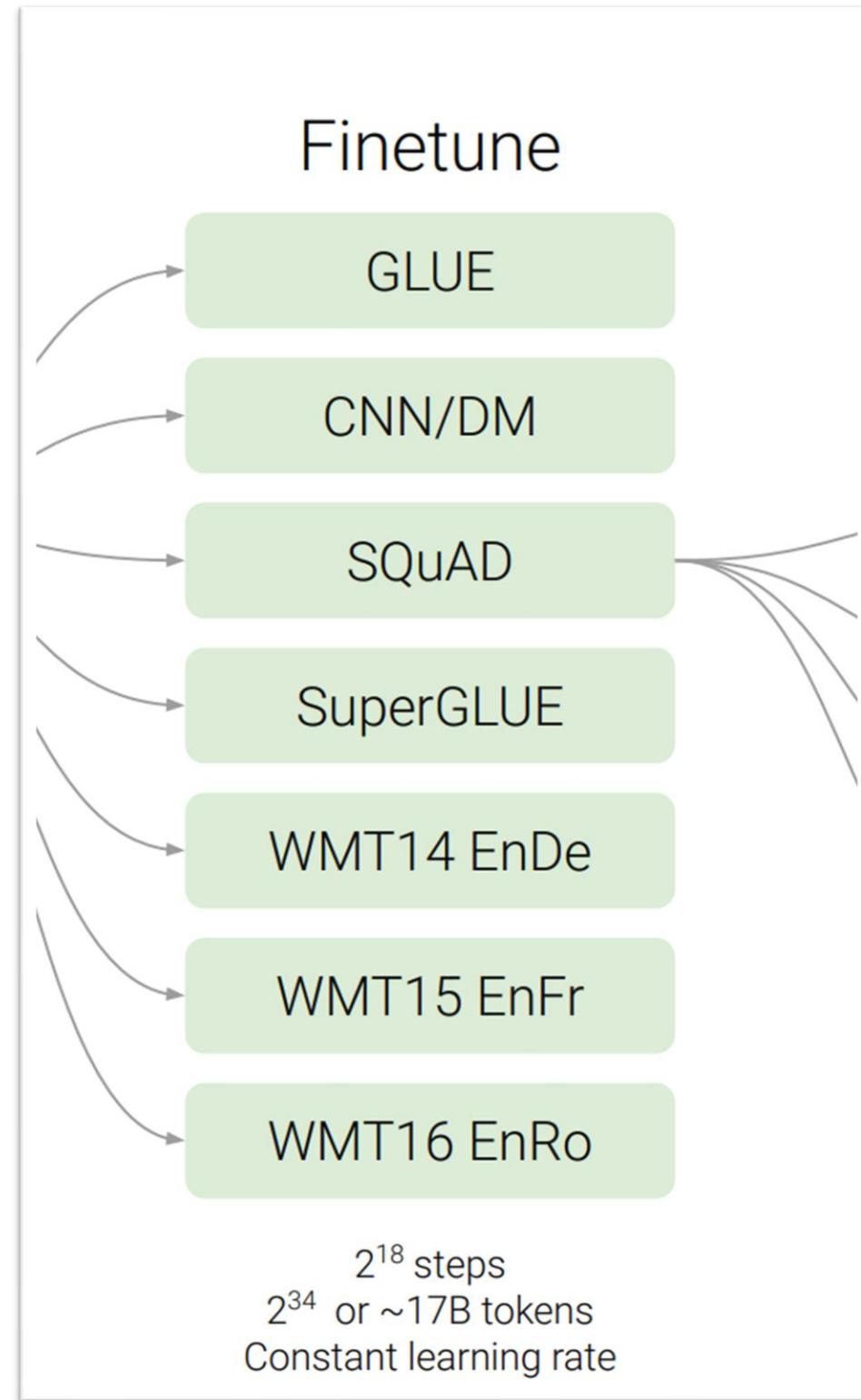
#2 baseline experiment procedure: pretrain - finetune - evaluate



1. BERT size의 encoder와 decoder 구조의 transformer를 340억개의 token으로 pre-train을 진행 훈련시간은 BERT에 비해 약 4분의 1정도

#2 baseline experiment procedure: pretrain - finetune - evaluate

#2 baseline experiment procedure: pretrain - finetune - evaluate

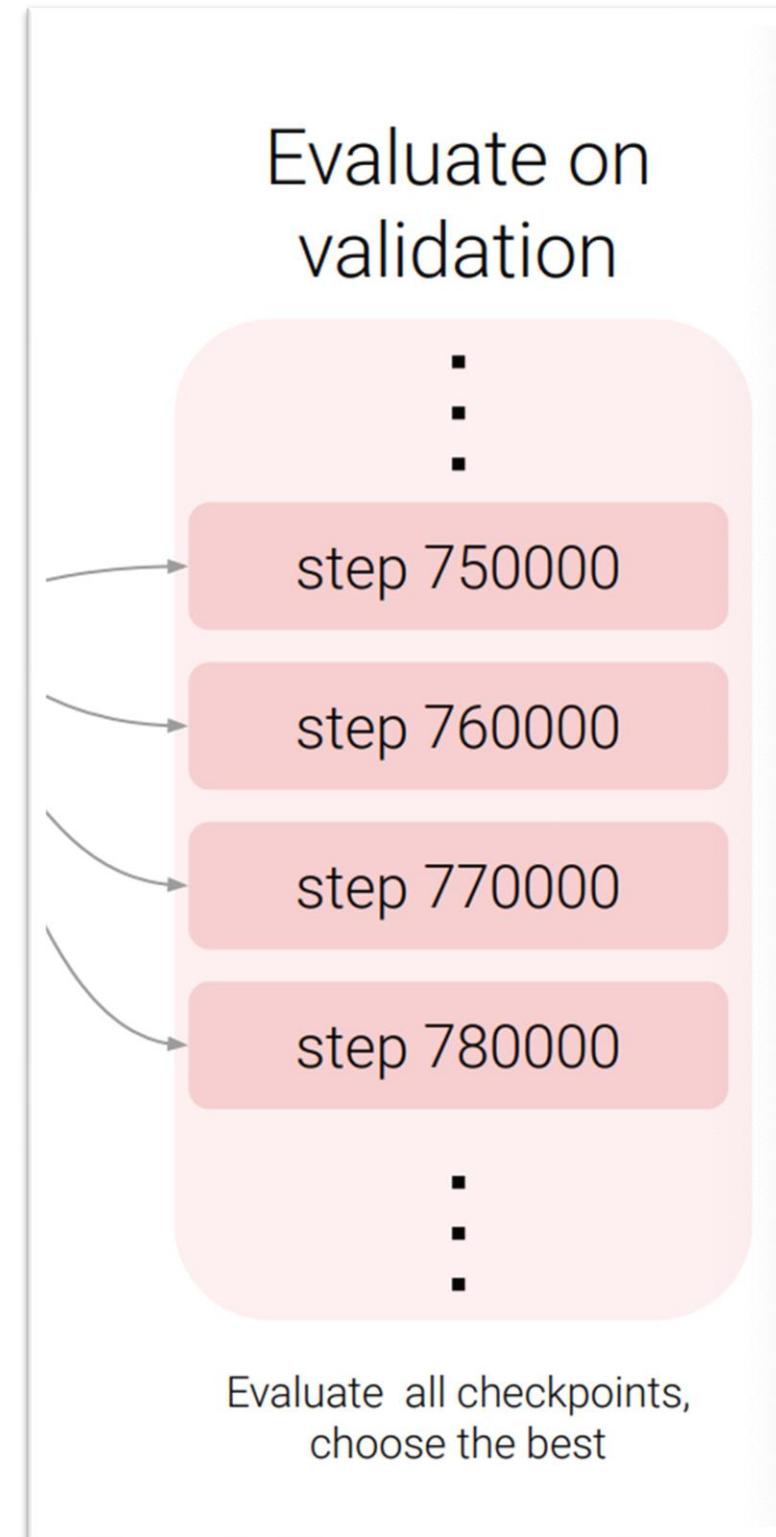


2.
이렇게 pre-train된 model을 가져와 각 downstream task에 맞게 fine-tuning을 진행

최대 170억개의 token을 fine-tuning함.

#2 baseline experiment procedure: pretrain - finetune - evaluate

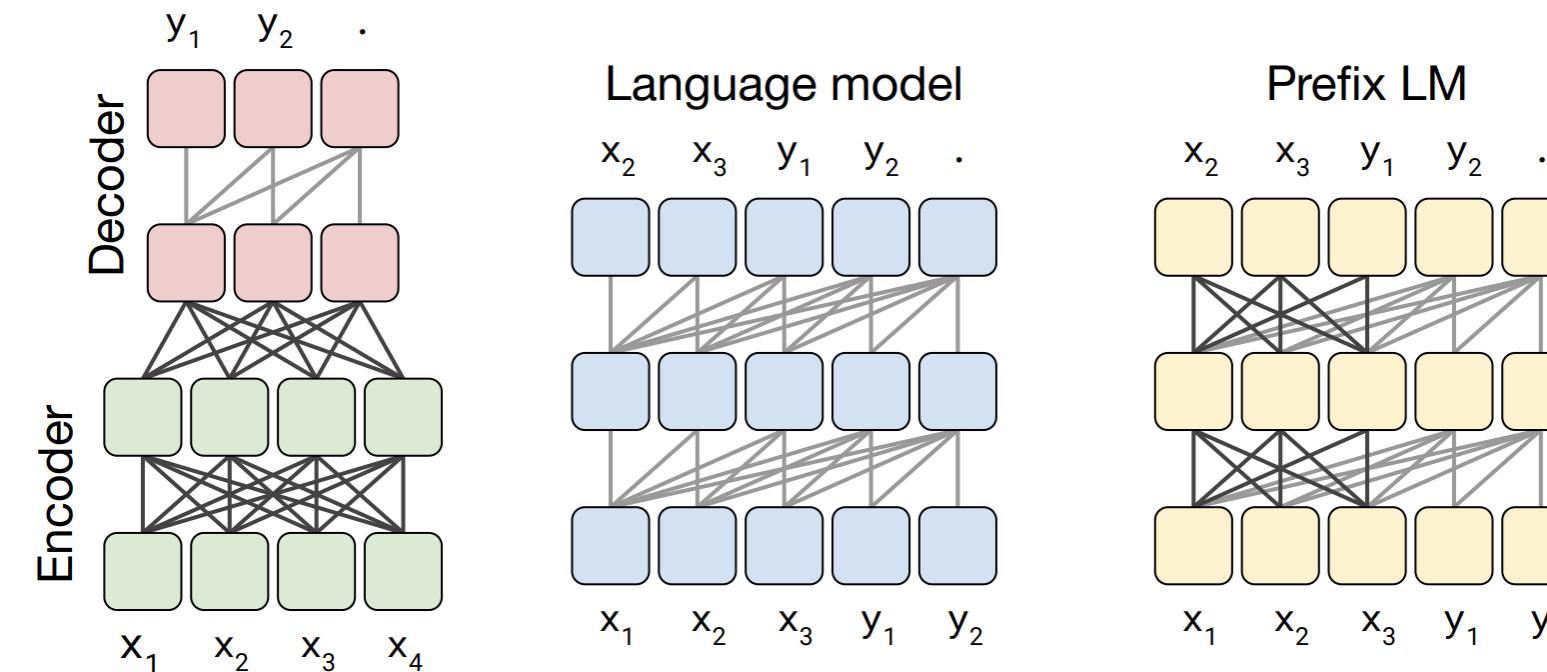
#2 baseline experiment procedure: pretrain - finetune - evaluate



3. validation set에 대한 체크포인트를 평가하여 가장 좋은 성능을 보인 model 찾기.

#3 experimental results

#3 experimental results : T5에 대한 여러가지 실험을 진행

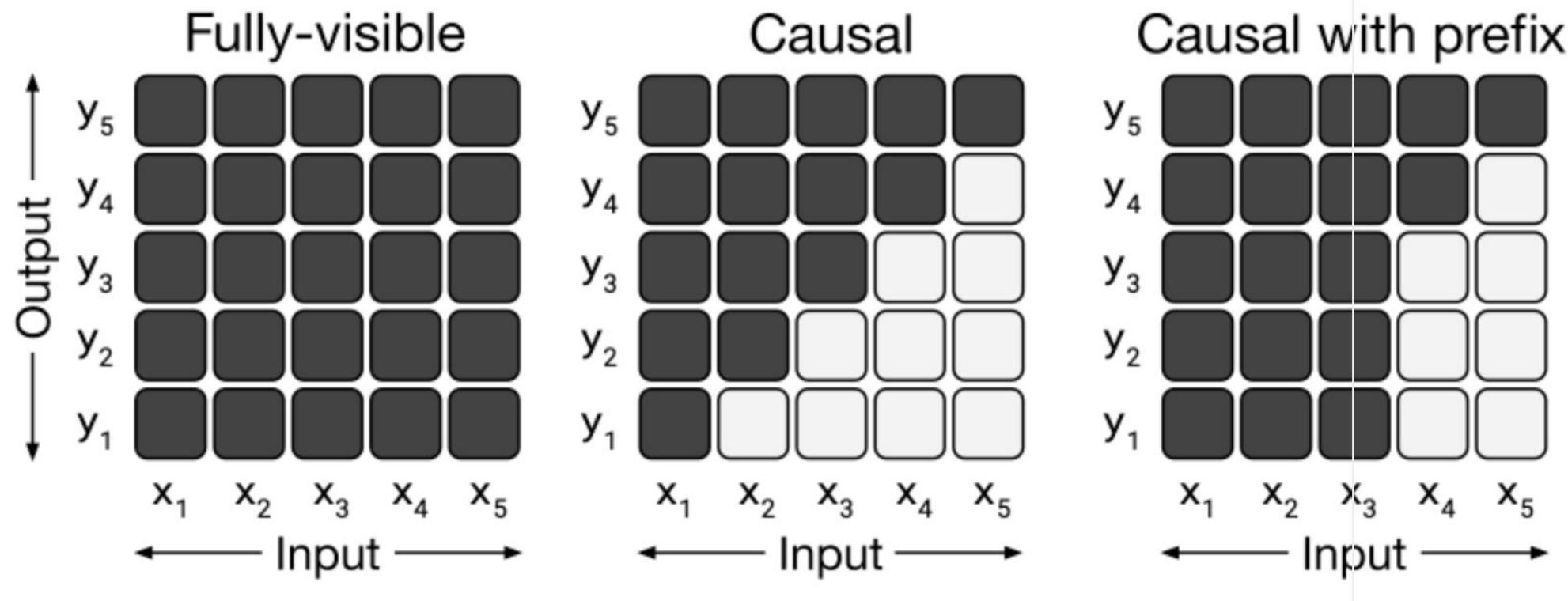


첫 번째 실험은 서로 다른 model 구조를 비교하는 것

- Fully-visible : Query가 모든 Key에 attention 가능
- Casual : Query(현재 time-step)가 이전 time-step의 Key에 attention 가능
- Casual with prefix : Query가 현재와 이전 time-step의 prefix Key에 attention 가능

#3 experimental results

#3 experimental results : 1. 서로 다른 model 구조를 비교



- Fully-visible : encoder는 fully visible하게 attention을 진행하고 decoder의 경우 causal한 방법으로 attention을 진행
- Casual : baseline model의 절반인 6개의 layer만 제한하고 decoder만 존재하는, regressive한 성격을 가지고 있는 GPT와 같은 model
- Casual with prefix : encoder에서는 auto-encoder 형식을, 그리고 decoder에서는 auto-regressive 형식을 이용

#3 experimental results

#3 experimental results : 1. 서로 다른 model 구조를 비교

Architecture	Params	Cost	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Encoder-decoder	$2P$	M	83.28	19.24	80.88	71.36	26.98	39.82	27.65
Enc-dec, shared	P	M	82.81	18.78	80.63	70.73	26.72	39.03	27.46
Enc-dec, 6 layers	P	$M/2$	80.88	18.97	77.59	68.42	26.38	38.40	26.95
Language model	P	M	74.70	17.93	61.14	55.02	25.09	35.28	25.86
Prefix LM	P	M	81.82	18.61	78.94	68.11	26.43	37.98	27.39

- encoder-decoder의 구조를 가진 model이 성능이 제일 좋음
- LM이 성능이 가장 좋지 않았음.
- 결국, bi-directional한 구조를 가진 model이 좋은 것을 알 수 있음

#3 experimental results

#3 experimental results - 2. pre-training한 dataset에 따른 성능 변화 비교 실험



Dataset	Size	GLUE	CNNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ C4	745GB	83.28	19.24	80.88	71.36	26.98	39.82	27.65
C4, unfiltered	6.1TB	81.46	19.14	78.78	68.04	26.55	39.34	27.21
RealNews-like	35GB	83.83	19.23	80.39	72.38	26.75	39.90	27.48
WebText-like	17GB	84.03	19.31	81.42	71.40	26.80	39.74	27.59
Wikipedia	16GB	81.85	19.31	81.29	68.01	26.94	39.69	27.67
Wikipedia + TBC	20GB	83.65	19.28	82.08	73.24	26.77	39.63	27.57

Much worse on CoLA
Order of magnitude smaller

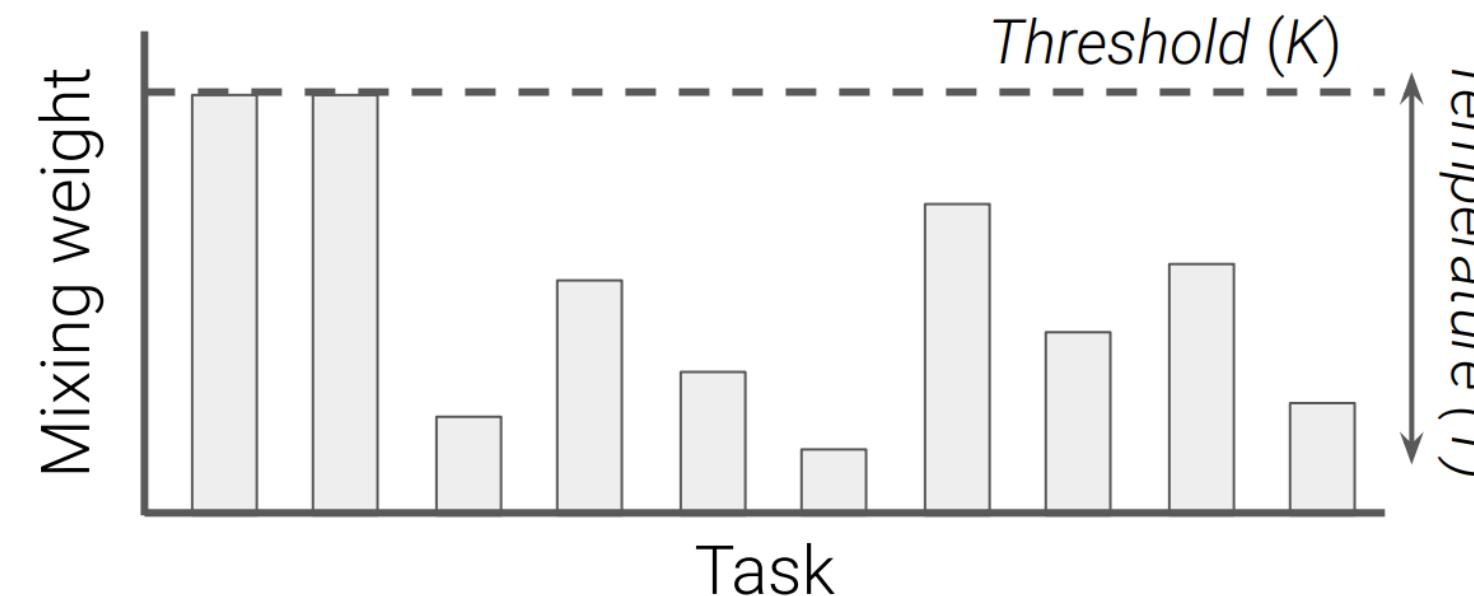
Much better on ReCoRD
Much better on MultiRC

- 기존 수집한 C4와 unfiltered C4 그리고 여러 다른 dataset을 가지고 pre-train한 후 여러 task에 대해 성능을 비교 -> C4를 사용했을 때 성능이 제일 좋았음

#3 experimental results

#3 experimental results - 3. T5의 다양한 task들의 학습량 비교 실험

Mixing strategy	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Baseline (pre-train/fine-tine)	83.28	19.24	80.88	71.36	26.98	39.82	27.65
Equal	76.13	19.02	76.51	63.37	23.89	34.31	26.78
Examples-proportional, $K = 2^{16}$	80.45	19.04	77.25	69.95	24.35	34.99	27.10
Examples-proportional, $K = 2^{17}$	81.56	19.12	77.00	67.91	24.36	35.00	27.25
Examples-proportional, $K = 2^{18}$	81.67	19.07	78.17	67.94	24.57	35.19	27.39
Examples-proportional, $K = 2^{19}$	81.42	19.24	79.78	67.30	25.21	36.30	27.76
Examples-proportional, $K = 2^{20}$	80.80	19.24	80.36	67.38	25.66	36.93	27.68
Examples-proportional, $K = 2^{21}$	79.83	18.79	79.50	65.10	25.82	37.22	27.13
Temperature-scaled, $T = 2$	81.90	19.28	79.42	69.92	25.42	36.72	27.20
Temperature-scaled, $T = 4$	80.56	19.22	77.99	69.54	25.04	35.82	27.45
Temperature-scaled, $T = 8$	77.21	19.10	77.14	66.07	24.55	35.35	27.17



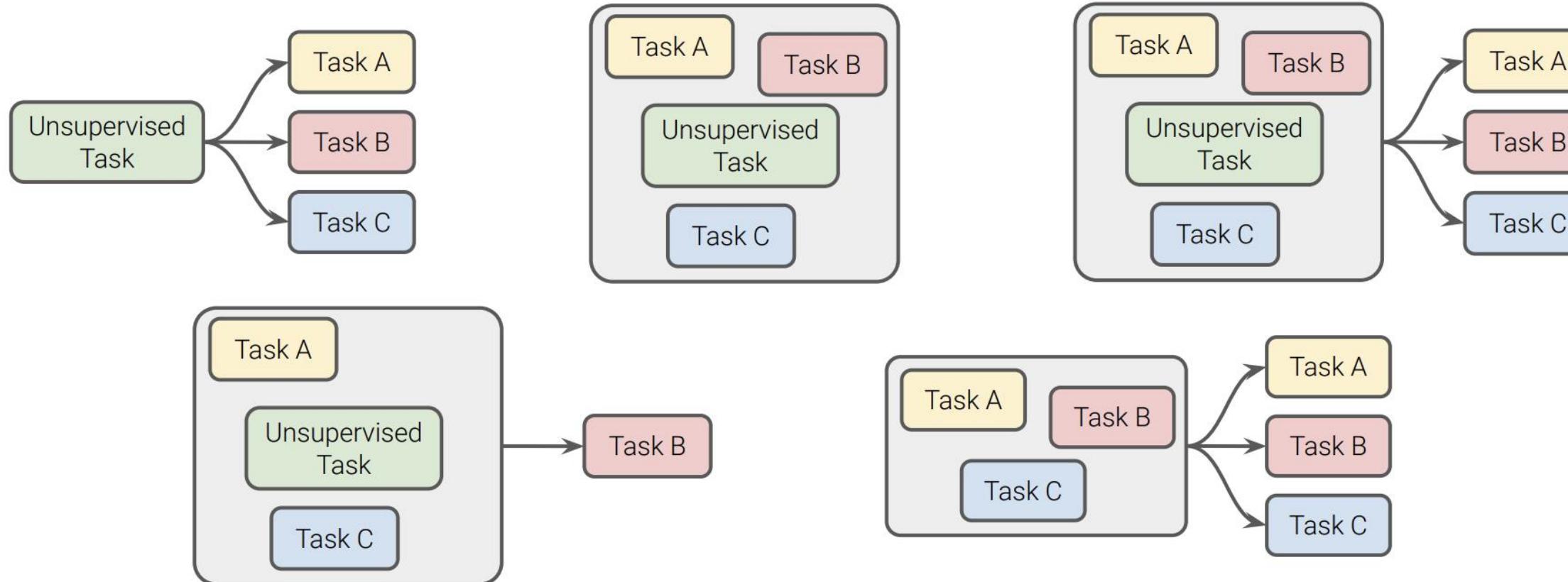
Equal: task에 상관없이 모두 같은 사이즈의 data를 학습시키는 방식

K는 threshold로 기본적으로 data의 size만큼 학습하고 K이상의 데이터는 K만큼 학습

#3 experimental results

#3 experimental results - 4. 다양한 방식의 조합으로 훈련을 진행하여 성능을 비교

Training strategy	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Unsupervised pre-training + fine-tuning	83.28	19.24	80.88	71.36	26.98	39.82	27.65
Multi-task training	81.42	19.24	79.78	67.30	25.21	36.30	27.76
Multi-task pre-training + fine-tuning	83.11	19.12	80.26	71.03	27.08	39.80	28.07
Leave-one-out multi-task training	81.98	19.05	79.97	71.68	26.93	39.79	27.87
Supervised multi-task pre-training	79.93	18.96	77.38	65.36	26.81	40.13	28.04

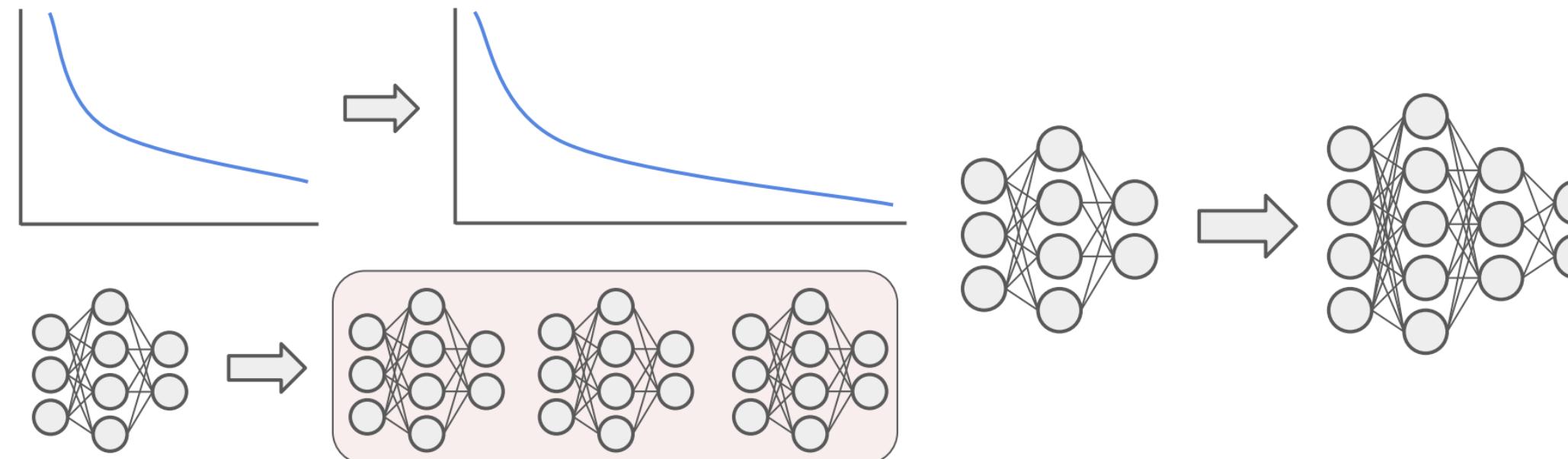


Pre-training을 진행하고 fine-tuning을 진행한 방식이 제일 성능이 높았음

#3 experimental results

#3 experimental results - 5. Scale-up을 할 수 있는 hyperparameter의 값을 조절해가면서 실험

Scaling strategy	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Baseline	83.28	19.24	80.88	71.36	26.98	39.82	27.65
1× size, 4× training steps	85.33	19.33	82.45	74.72	27.08	40.66	27.93
1× size, 4× batch size	84.60	19.42	82.52	74.64	27.07	40.60	27.84
2× size, 2× training steps	86.18	19.66	84.18	77.18	27.52	41.03	28.19
4× size, 1× training steps	85.91	19.73	83.86	78.04	27.47	40.71	28.10
4× ensembled	84.77	20.10	83.09	71.74	28.05	40.53	28.57
4× ensembled, fine-tune only	84.05	19.57	82.36	71.55	27.55	40.22	28.09



Training steps, batch size, model size를 조절하면서 실험한 결과,
training steps를 늘리고 model size를 키웠을 때 성능이 제일 좋았음.

#3 experimental results

#3 experimental results - T5가 여러 benchmark에서 SOTA를 달성한 결과

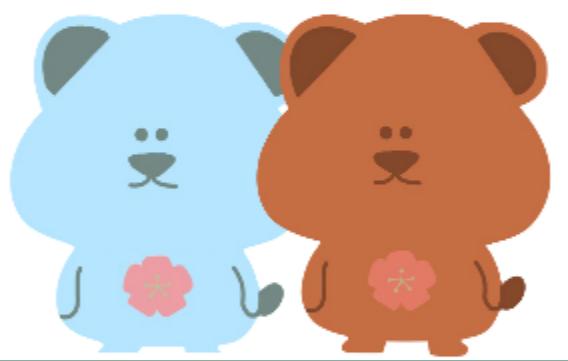
Model	GLUE Average	CNN/DM ROUGE-2-F	SQuAD EM	SuperGLUE Average	WMT EnDe BLEU	WMT EnFr BLEU	WMT EnRo BLEU
Previous best	89.4	20.30	90.1	84.6	33.8	43.8	38.5
T5-Small	77.4	19.56	87.24	63.3	26.7	36.0	26.8
T5-Base	82.7	20.34	92.08	76.2	30.9	41.2	28.0
T5-Large	86.4	20.68	93.79	82.3	32.0	41.5	28.1
T5-3B	88.5	21.02	94.95	86.4	31.8	42.6	28.2
T5-11B	90.3	21.55	91.26	89.3	32.1	43.4	28.1

Back-translation beats English-only pre-training

Human score = 89.8

번역에서는 모두 SOTA에 비해 점수가 좋지 않았음.
→ SOTA를 달성한 model이 모두 back translation을 진행했기 때문

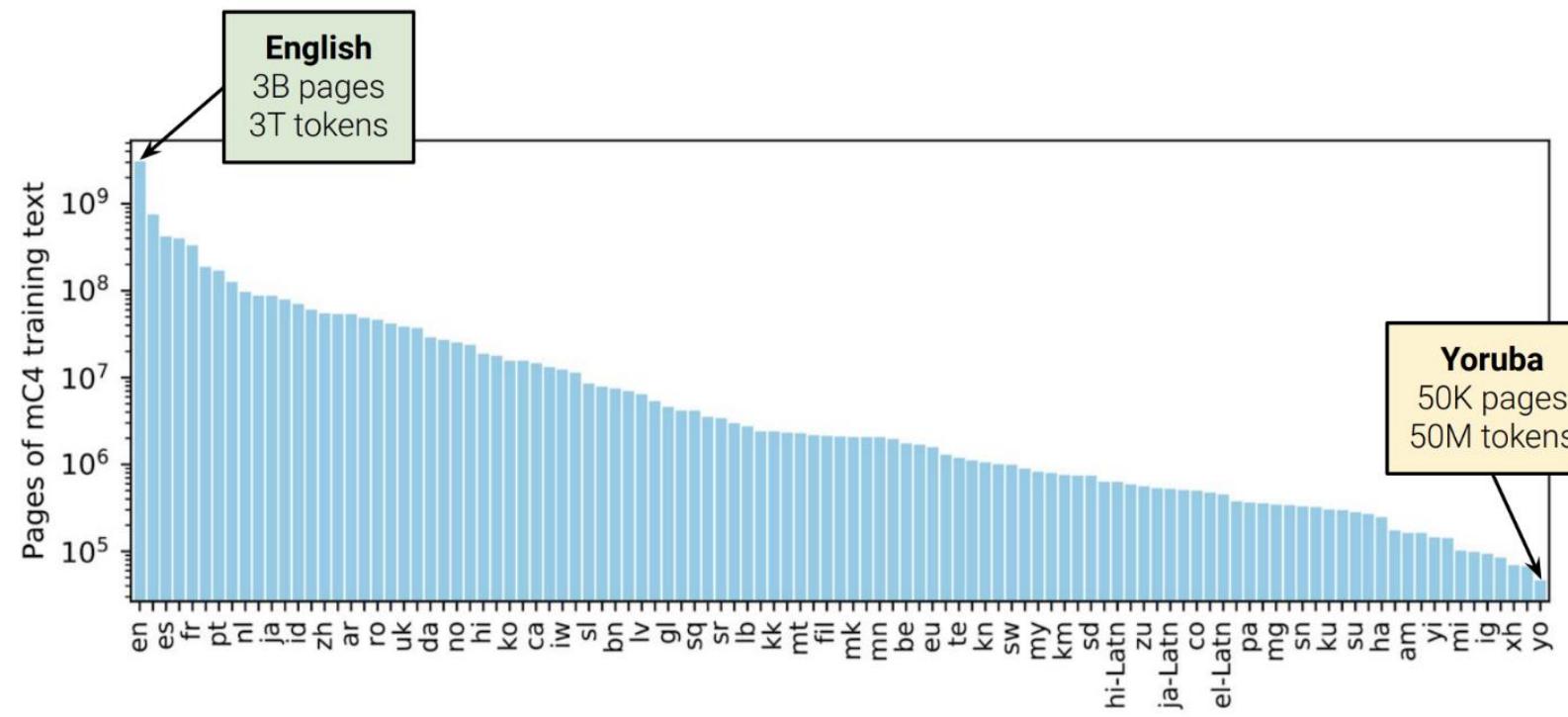
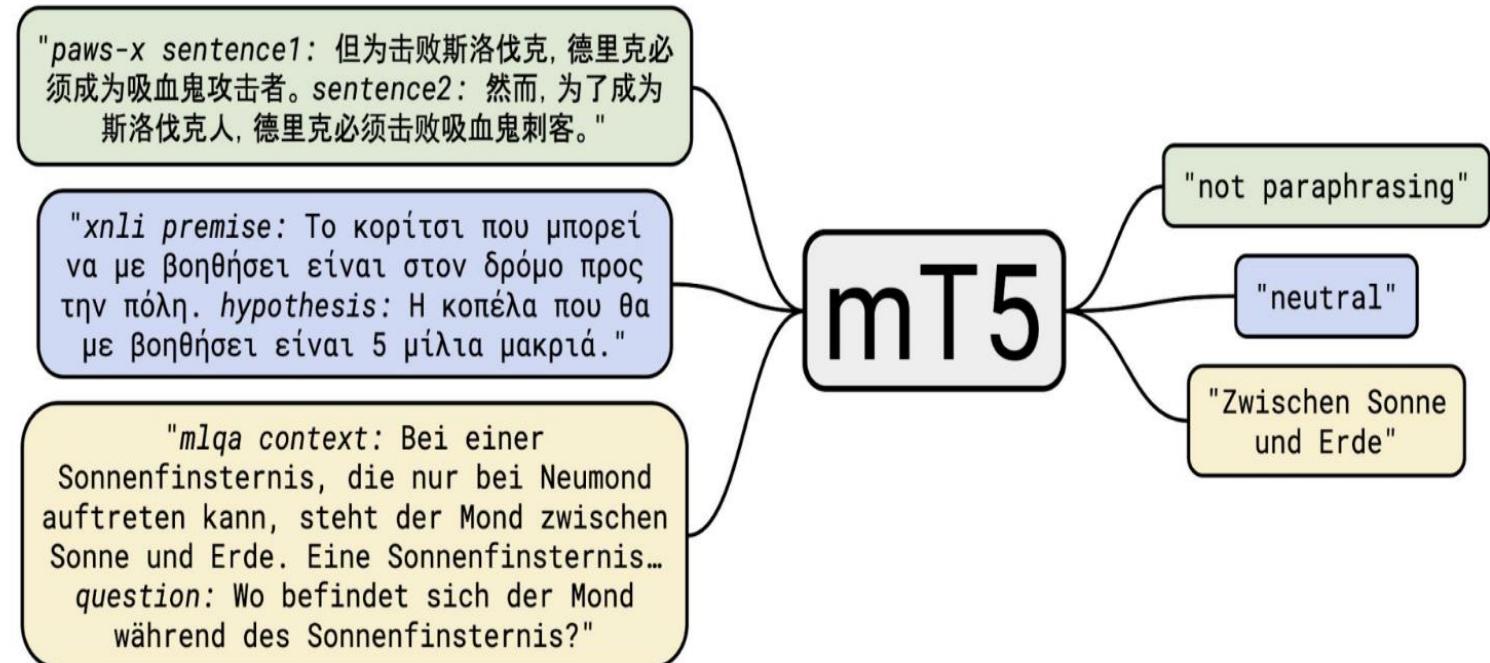
03. Other topics



3. Other topics

Multilingual T5 – What about all of the other languages?

- 다른 languages에는 어떻게 적용할까?



Text-to-text 형식은 그대로 갖고 있지만 input과 output이 다른 언어로 나올 수 있음

⇒ 그러면 적절한 pre-train dataset이 필요

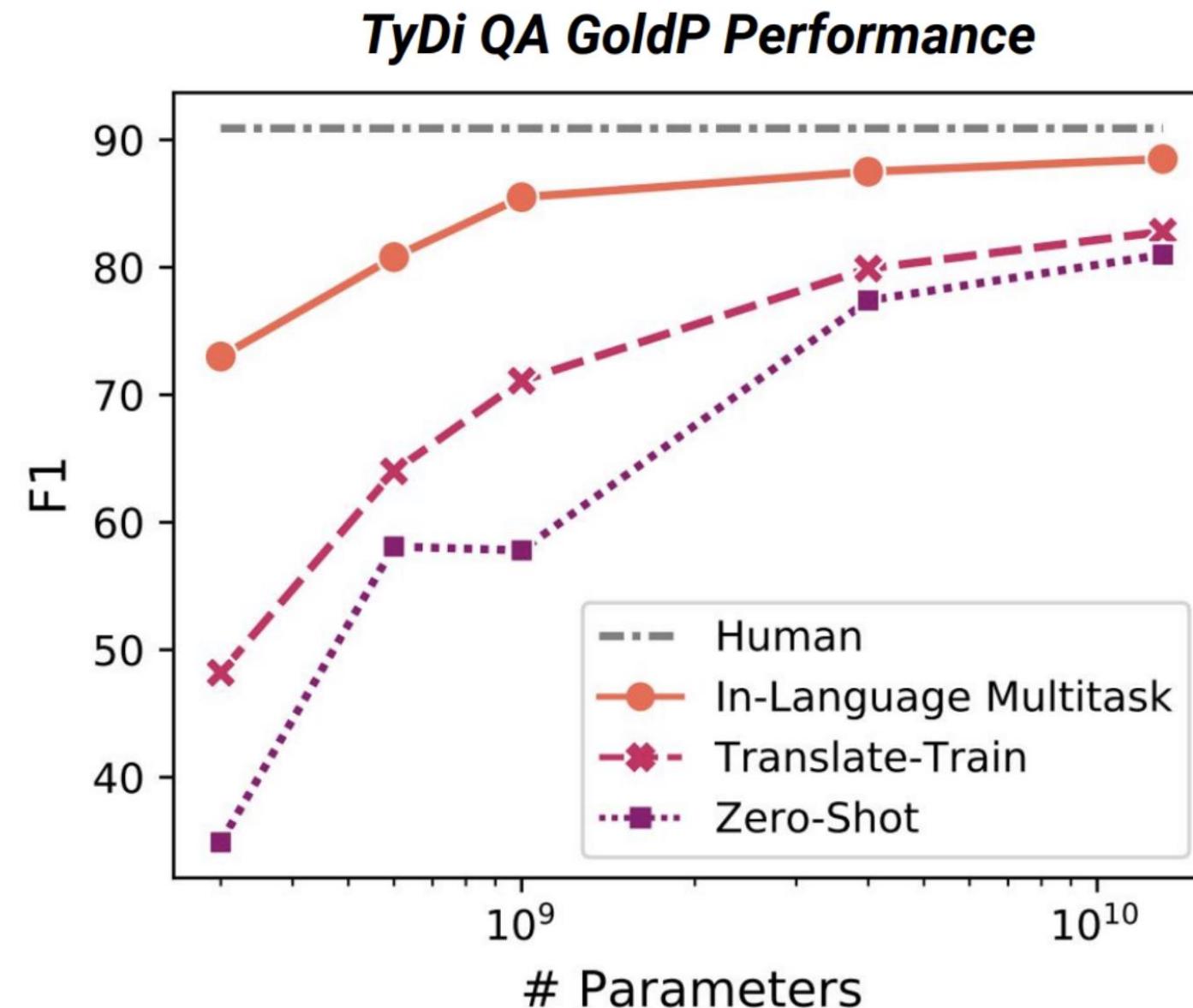
⇒ 101개 languages가 포함된 C4 data를 이용 + 질이 좋은 data를 더 얻기 위해 Common Crawl dump에서 data 추출

- 우측의 그래프를 통해 language 분포도 확인 가능, language 양의 차이를 좁히기 위해 scaling 진행

3. Other topics

Multilingual T5 – What about all of the other languages?

- 다른 languages에는 어떻게 적용할까?



<각기 다른 훈련방법을 통한 성능차이>

- 1) zero-shot: fine-tuning X
- 2) translate-train: 영어 corpus로 fine-tuning
- 3) In-Language Multitask: 데이터를 처리할 수 있는 모든 언어의 실측 데이터가 있음

=> 그래프가 시사하는 바: 결국 parameter가 많은, 큰 model이 어떠한 환경에서도 더 잘 수행한다는 점

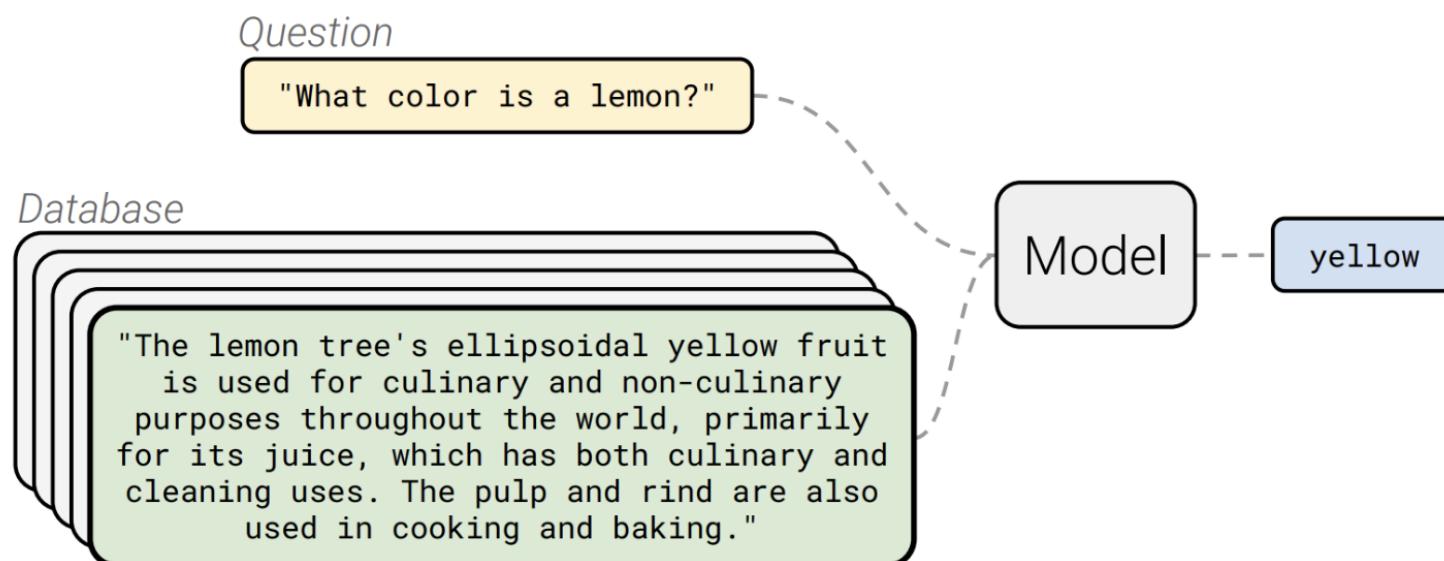
3. Other topics

How much knowledge does a language model pick up during pre-training?

- T5가 reading comprehension에서 보이는 성능?

* Reading comprehension이란 질문이 주어지고 질문에 대한 답을 찾을 수 있는 knowledge가 주어졌을 때, model이 답을 찾을 수 있는지에 대한 task

Open-Domain Question Answering

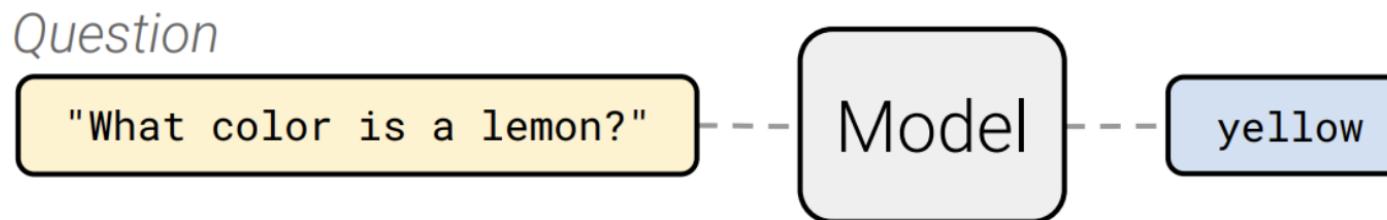


Open-Domain QA: 특정 knowledge에 상관없이 전체적인 모든 질문에 model이 답을 할 수 있는 task
=> 따라서 질문에 대한 답변을 얻을 수 있는 database를 구축하고 정답을 찾아냄

3. Other topics

Closed-book Question Answering

Closed-Book Question Answering



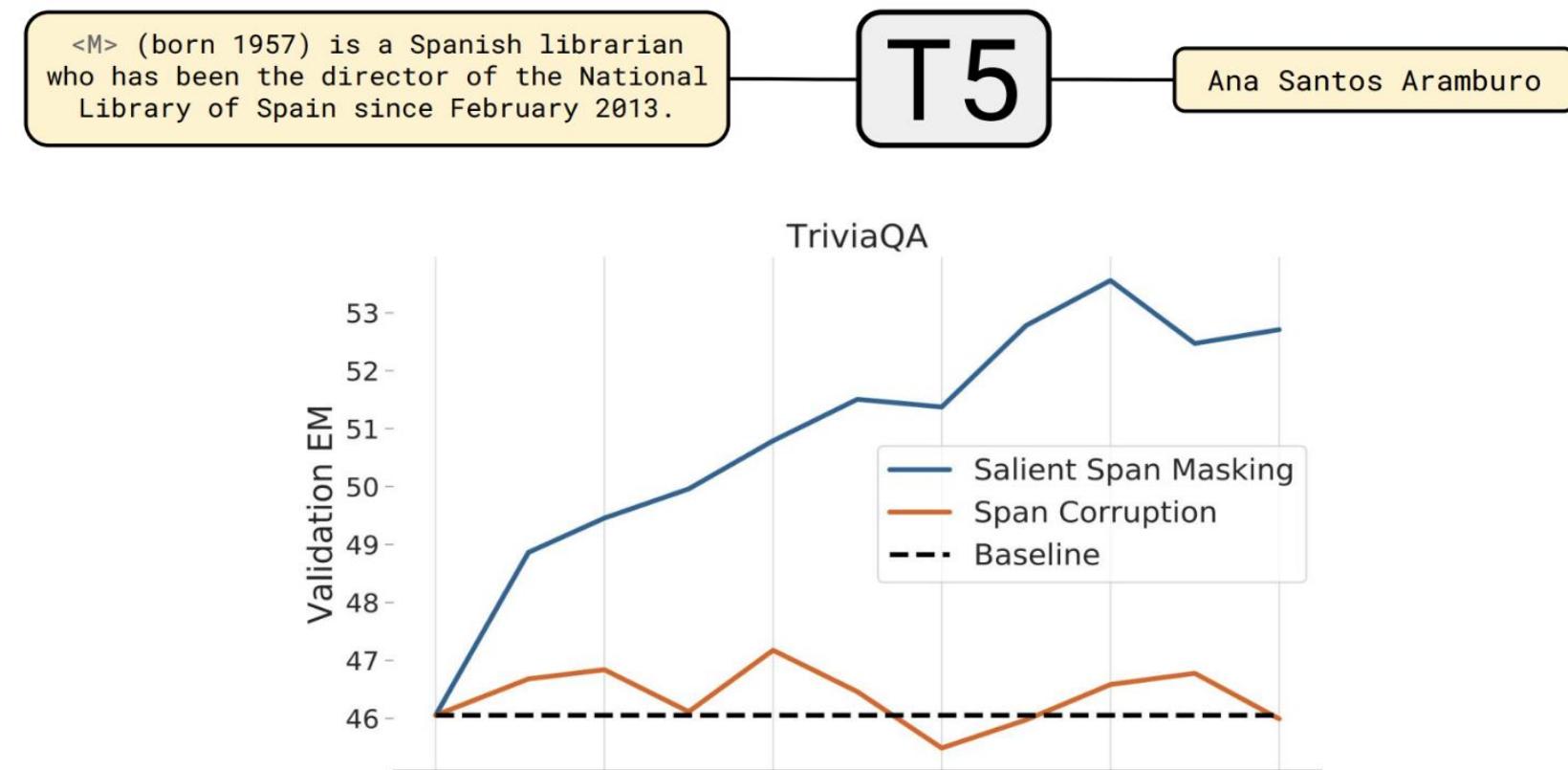
- Open-book: 질문과 지문이 주어지고 질문에 맞는 지문을 찾고 model이 해당 지문에서 정답을 찾는 task
- **Closed-book:** 대용량의 data를 학습한 model에 질문을 input으로 넣었을 때, output으로 정답이 나오게되는 task 이때 model이 얼마나 지식을 잘 기억하는지가 중요!

3. Other topics

Performance

- natural question, web question, trivial QA에 대한 성능

	NQ	WQ	TQA
Open-domain SoTA	41.5	42.4	57.9
T5.1.1-Base	25.7	28.2	24.2
T5.1.1-Large	27.3	29.5	28.5
T5.1.1-XL	29.5	32.4	36.0
T5.1.1-XXL	32.8	35.6	42.9



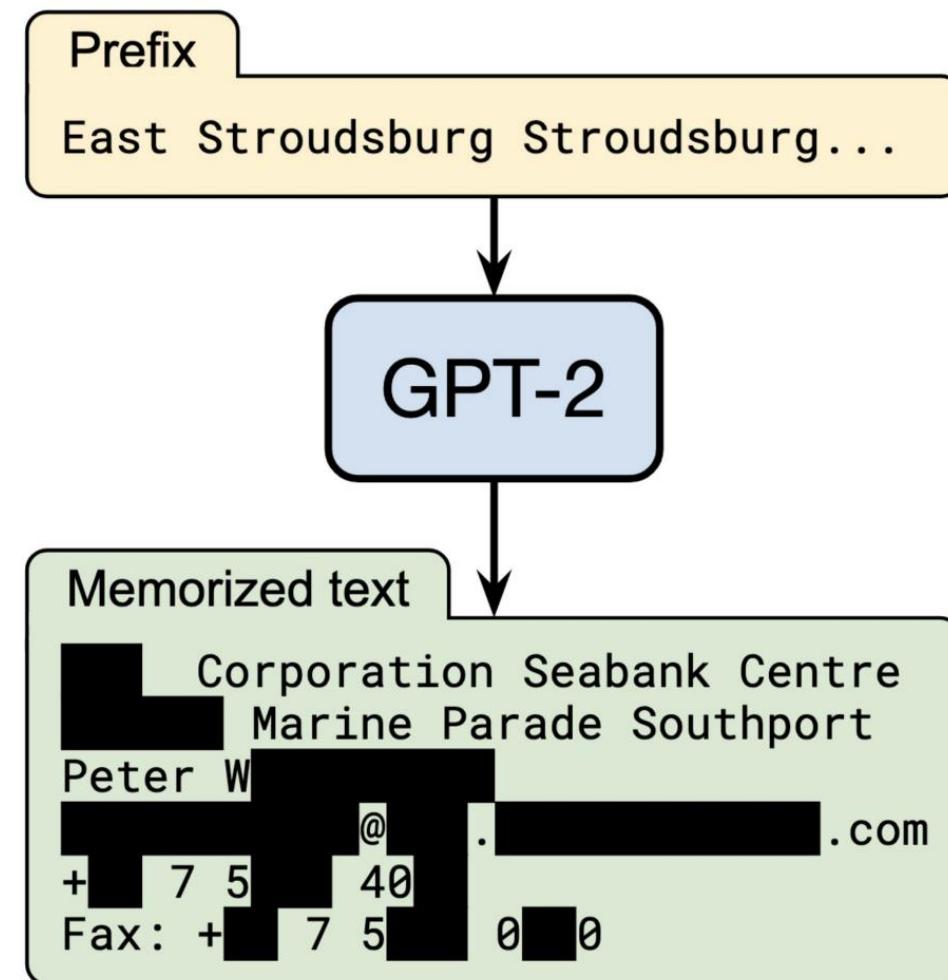
- Open-domain에 대한 SOTA model과 closed-book으로 훈련된 T5를 비교
- Model의 크기가 증가할수록 학습하는 지식의 양이 많아지기 때문에 성능이 증가하지만 SOTA에는 미치지 못함

- 이러한 격차를 좁히기 위해 salient span masking 도입
- Salient span masking이란 pre-train할 때 random하게 masking을 하는 것 X
- 특정 entity에 masking(사람 이름, 장소, 날짜, ...)
- 훈련을 하기 전 entity recognizer를 통해 entity를 파악하고 무작위 범위를 채우는 것이 아닌 두드러진 범위를 채우도록 model을 훈련

3. Other topics

Do large language models memorize their training data?

- Model이 pre-train할 때, 원하지 않는 정보 까지 기억할까?



- 대량의 data를 학습할 경우 여러가지 문제가 발생할 수 있는데 그 중 lecture에서는 원하지 않은 개인정보가 training dataset에 포함될 수 있음
- GPT같이 대량의 web기반의 data로 학습된 model이 개인의 신상정보를 generate할 수 있는 문제점이 있음
현재 이러한 이슈가 굉장히 중요한 issue가 됨

3. Other topics

Can we close the gap between large and small models by improving the Transformer architecture?

현재: 큰 model이 여러 실험과 benchmark에서 더 좋은 성능을 보여줬고 정보를 더 많이 담고 있는 것으로 증명

=>작은 model도 architecture를 수정하여 큰 model과의 격차를 줄일 수 있을까?

- Large model: 성능은 좋지만 돌리는데 드는 시간과 비용이 상당함 => 그래서 small model로도 격차가 많이 벌어지지 않게 성능을 낼 수 있는지에 대한 여러 방법이 현재까지 제시됨
- Factorized embeddings
- Shared embedding and softmax layer
- Mixture of Softmaxes, Adaptive softmax
- Different ways of normalizing or initializing (RMSNorm, ReZero, FixUp)
- Different attention mechanisms (Transparent Attention, Lightweight & Dynamic Convolutions, Synthesizer)
- Different structures for the feed-forward layers (Nonlinearities, Mixture of Experts, Switch Transformer)
- Different transformer architecture (Funnel Transformer, Evolved Transformer, Universal Transformer, block sharing ...)

THANK YOU

