UST UNIVERSITY OF SCIENCE & TECHNOLOGY

KISTI www.kisti.re.kr Korea Institute of Science and Technology Information

Spring 2021 Deep Learning: Technology and Applications

Language Model and Advances

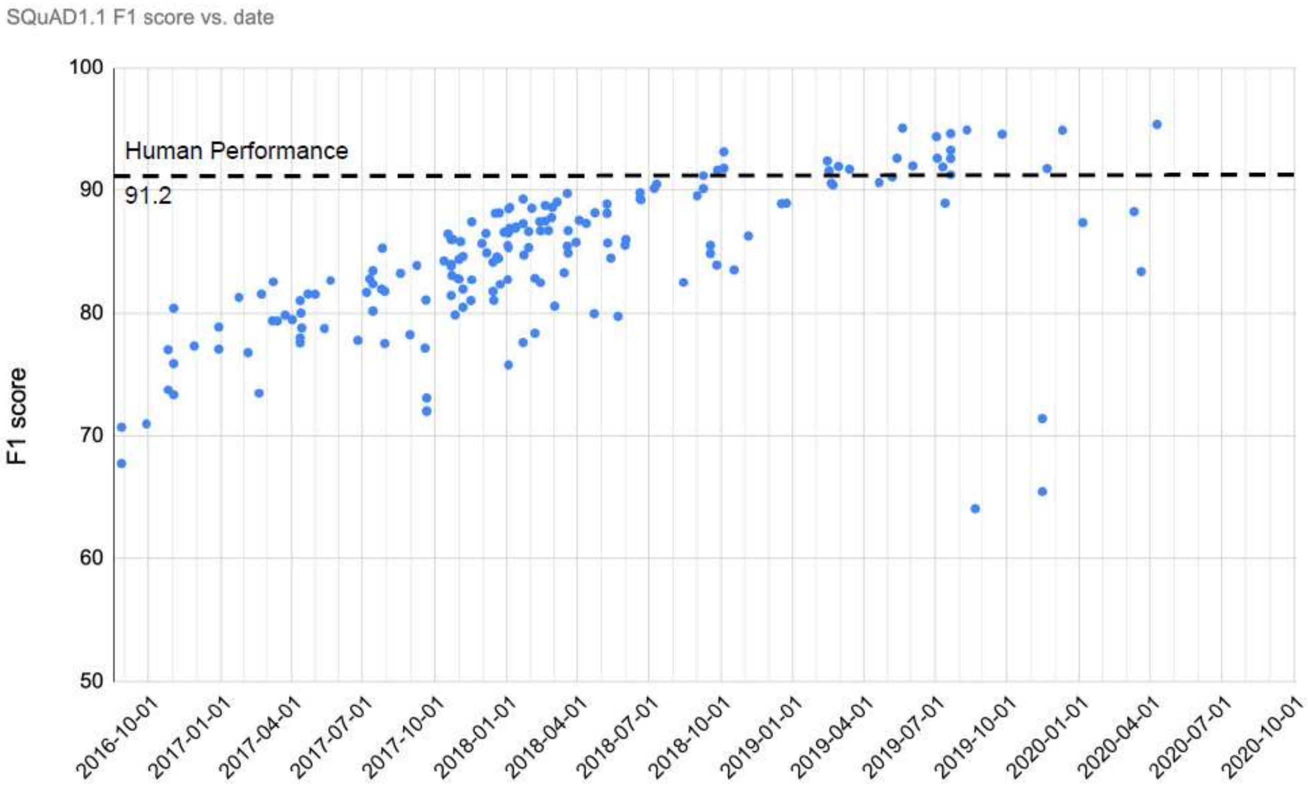
KISTI-UST
Kyong-Ha Lee

May 28, 2021

Contents

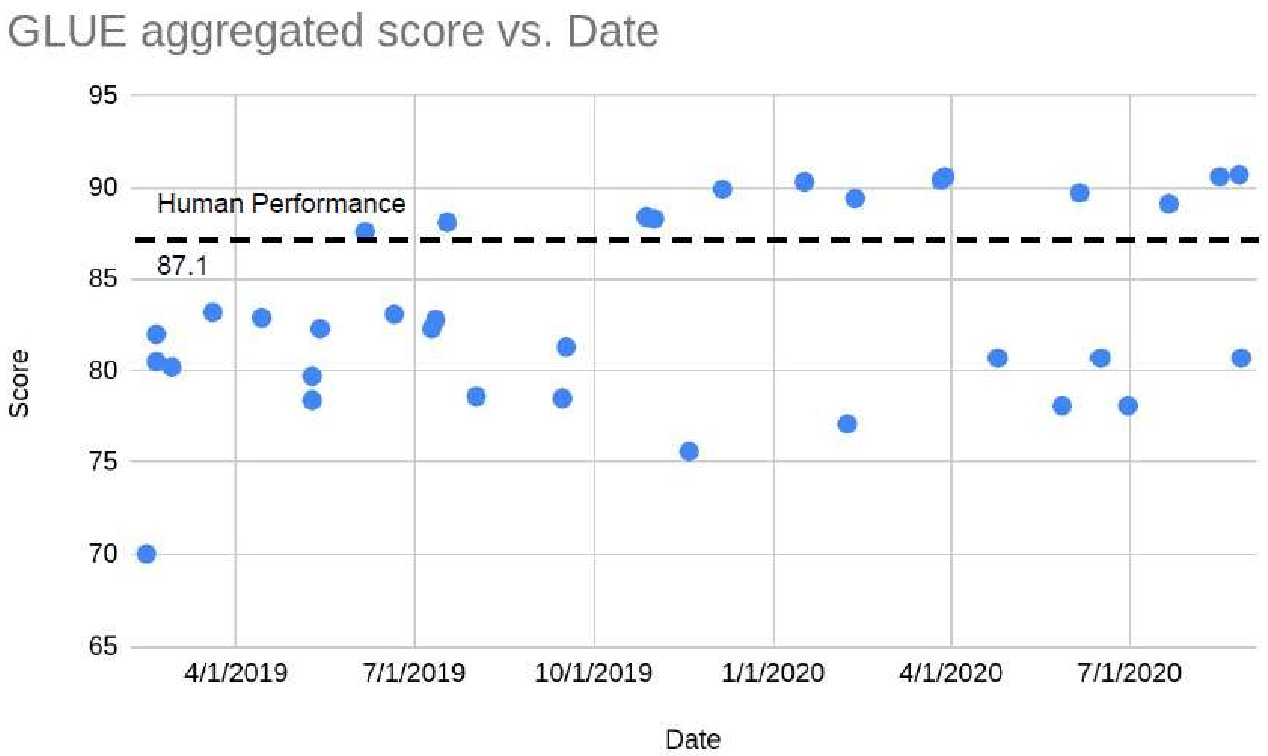
Recent years in NLP

- Benchmarks through the years - SQuAD 1.1



The Stanford Question Answering Dataset, https://rajpurkar.github.io/SQuAD-explorer/

Benchmarks through the years -GLUE



The GLUE Benchmark (Wang et al., 2018)

Brief History

January 16, 2013

WORD2VEC

Word2Vec Paper by Tomas Mikolov et al

January 2, 2014

GLOVE

GloVe: Global Vectors for Word Representation by Jeffrey Pennington et al.

July 15, 2016

FASTTEXT

Enriching Word Vectors with Subword Information by Piotr Bojanowski et al

June 12, 2017

TRANSFORMER

Attention Is All You Need by Ashish Vaswani et al

February 15, 2018

ELMO

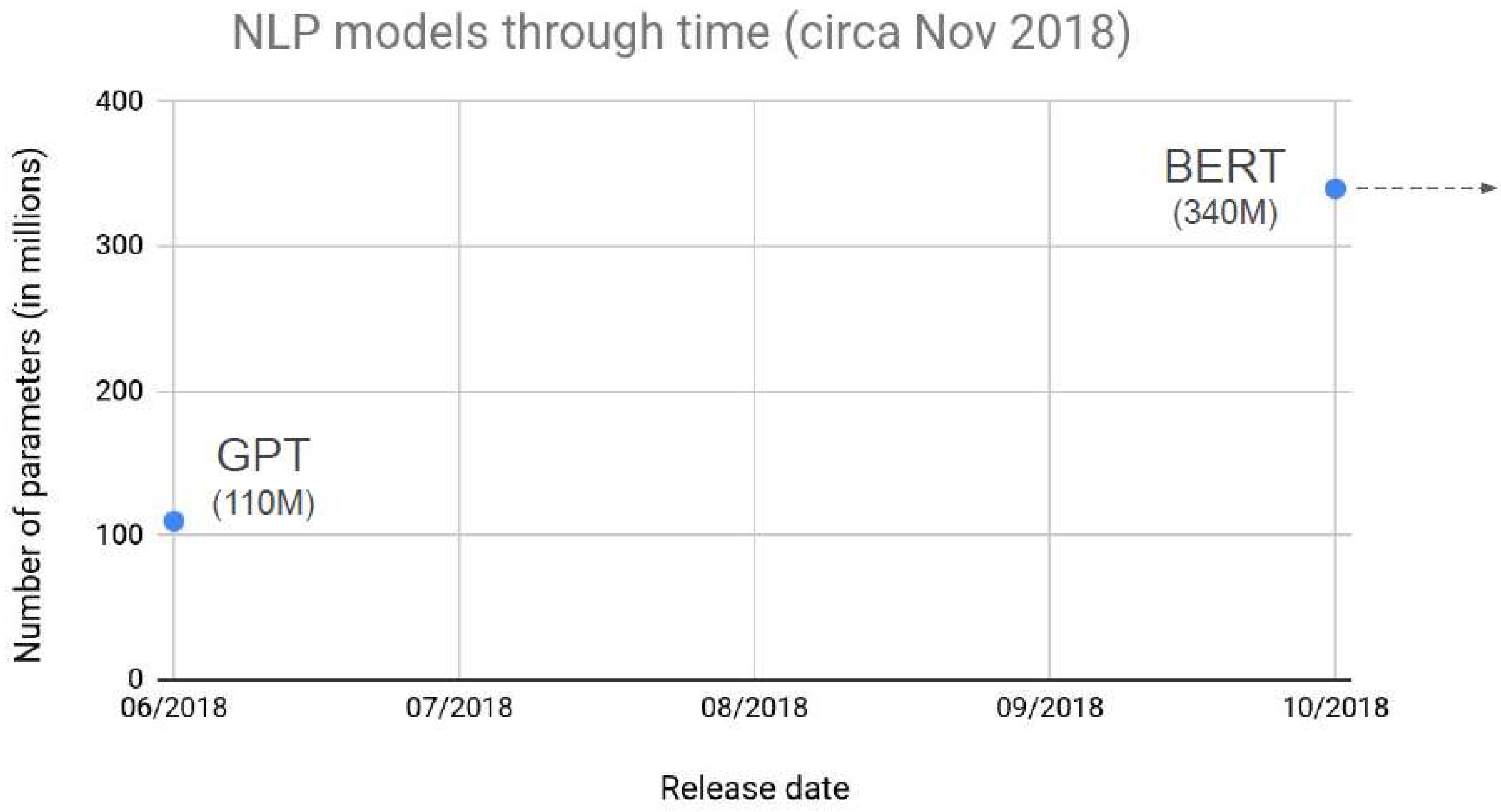
Deep contextualized word representations by Matthew E. Peters et al

October 11, 2018

BERT

BERT: Pre-training of Deep Bidirectional Transformers for...

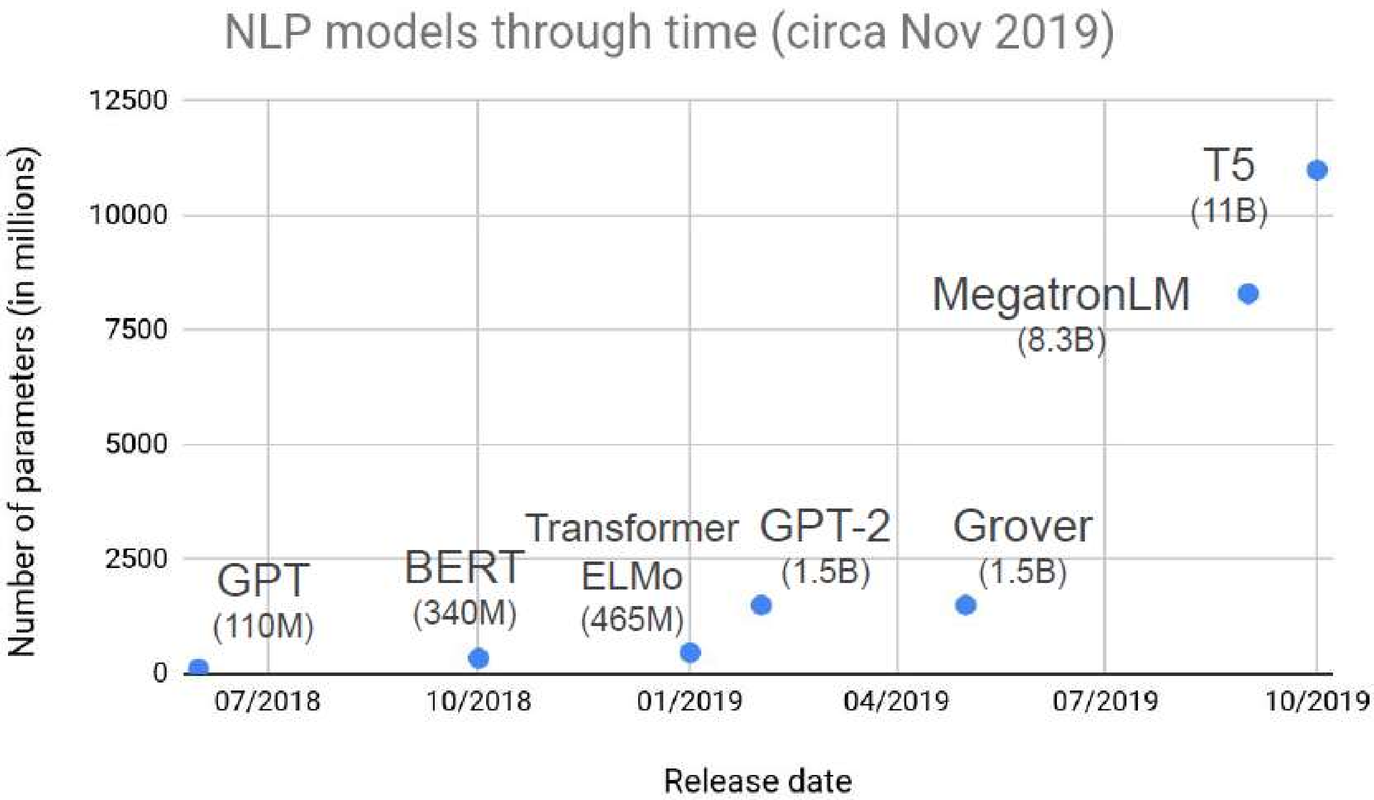
A brief recent history of scale in NLP



"[...] scaling to **extreme** model sizes also leads to large improvements [...]"

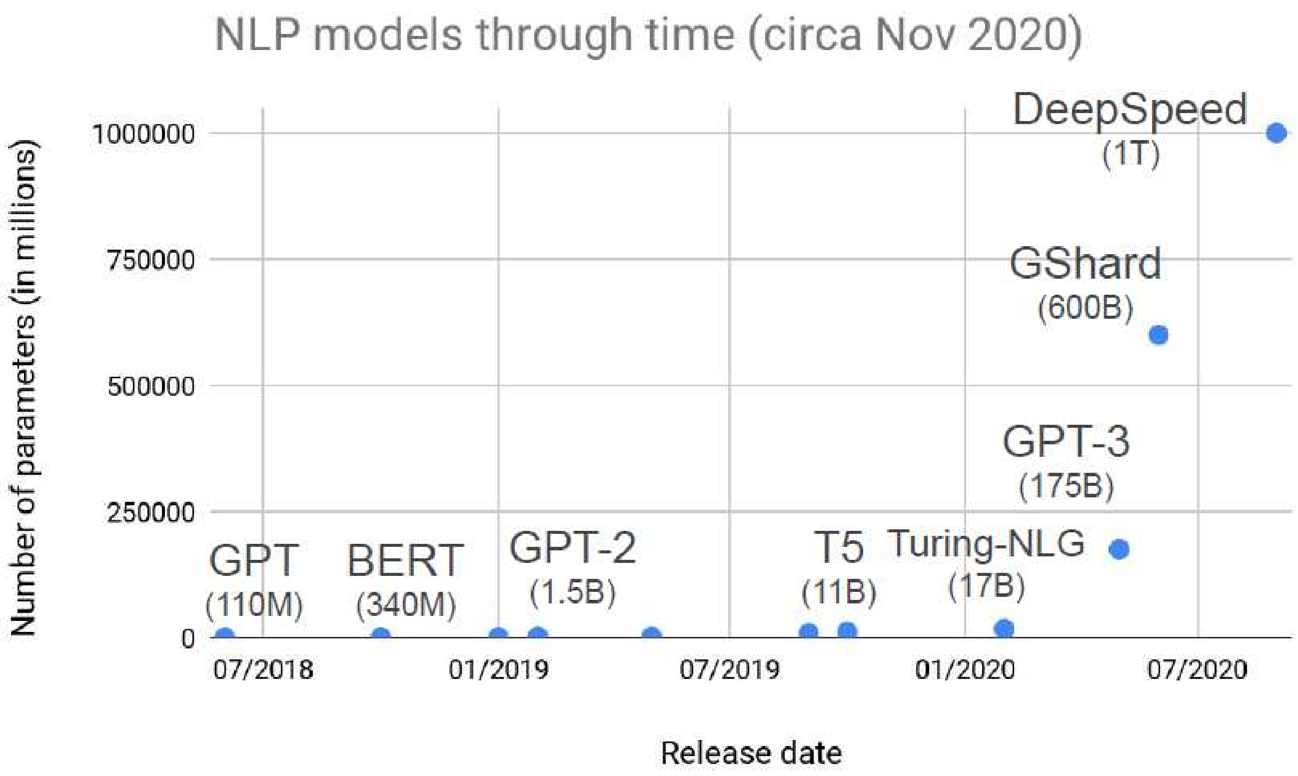
(Devlin et al., 2018)

A brief recent history of scale in NLP

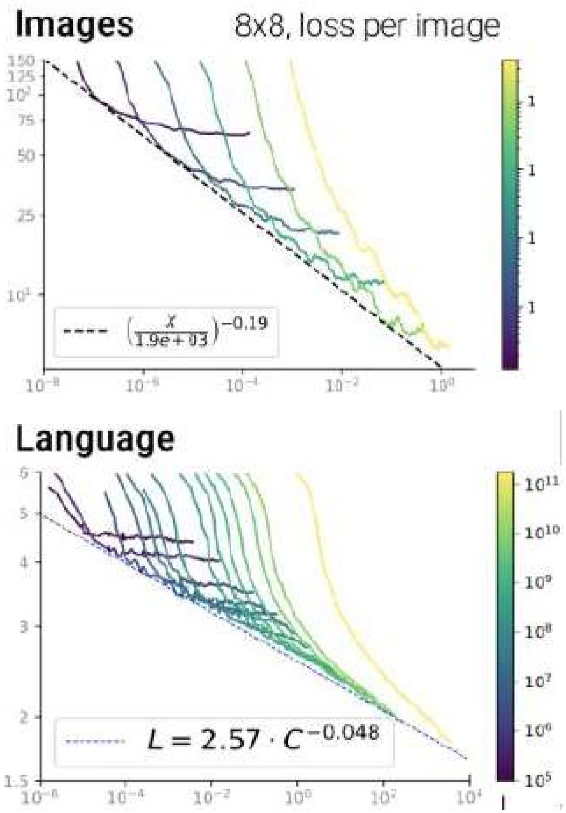


"[...] scaling the model size to 11 billion parameters was the most important ingredient for achieving our best performance."
(Raffel et al, 2019)

A brief recent history of scale in NLP

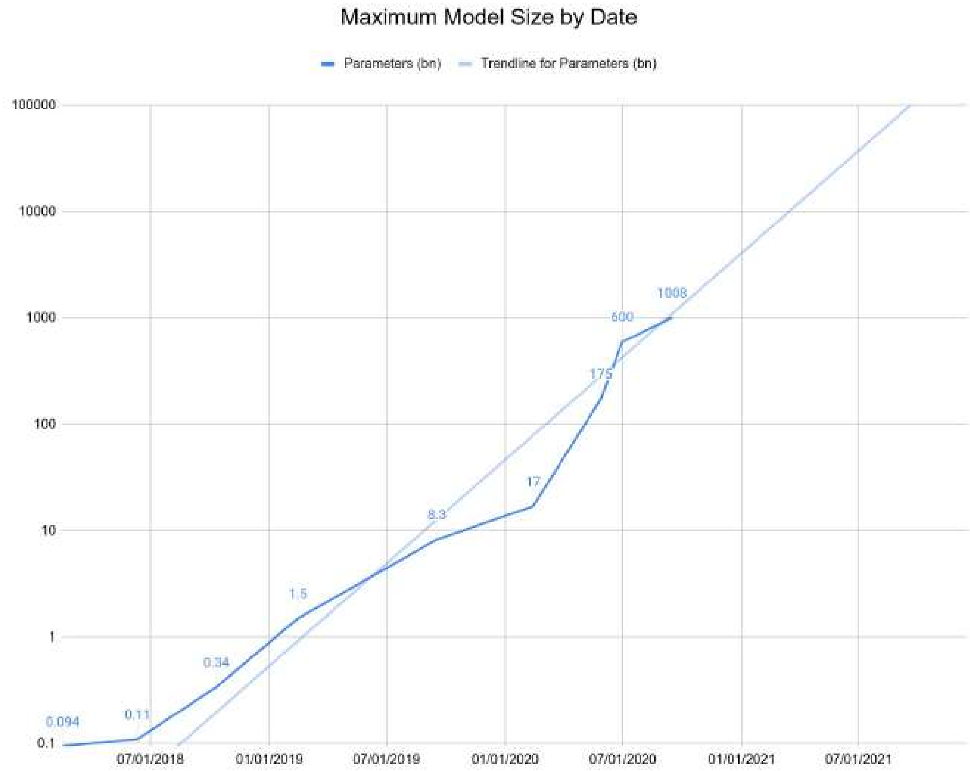


Scaling laws



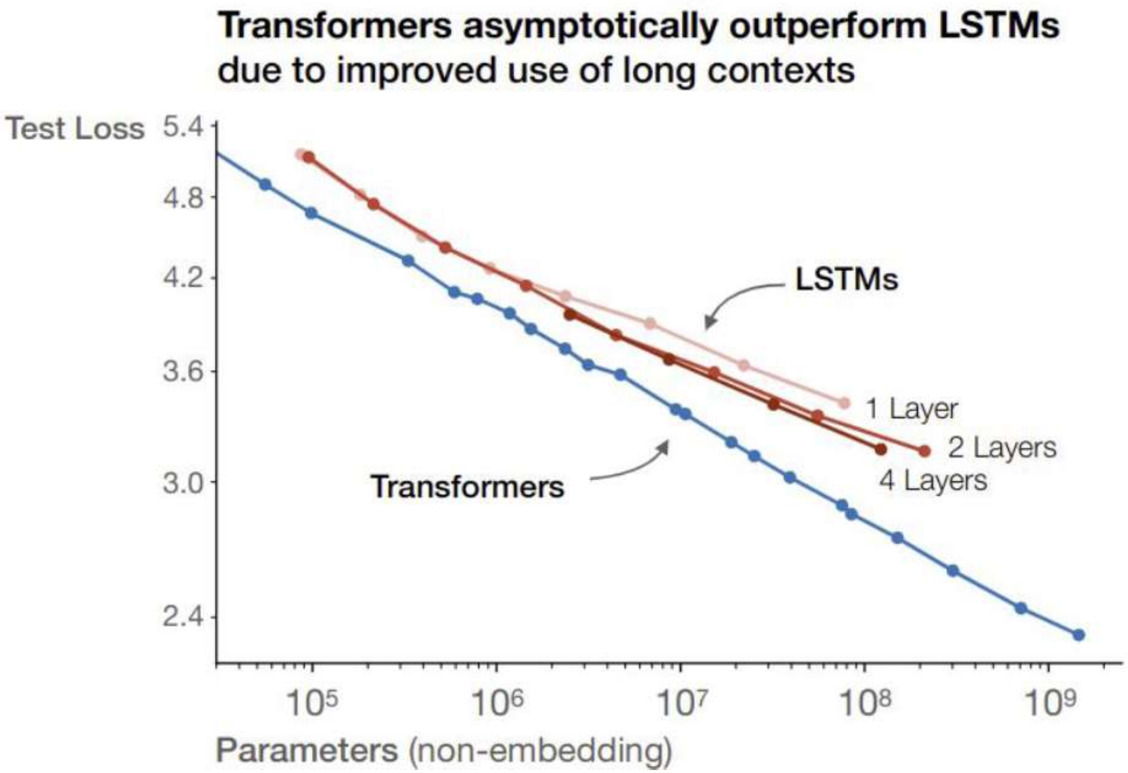
Line colors denote model sizes

* Henighan et al., 2020



Why do we need scale?

- Transformers are ubiquitous in NLP



Kaplan et al., 2020: arXiv

UNIVERSITY OF SCIENCE & TECHNOLOGY

Korea Institute of Science and Technology Information

ELMo : Embeddings for Language Models

- Pre-trained word representations
  - A key component in many neural language understanding models
- High quality representations should ideally model the followings
  - **Complex characteristics of word use** (e.g., syntax and semantics)
  - How these uses vary across **linguistic contexts** (i.e., to model polysemy)

* Deep contextualized word representations, M.E. Peters et al, 2018

ELMo : Embeddings for Language Models

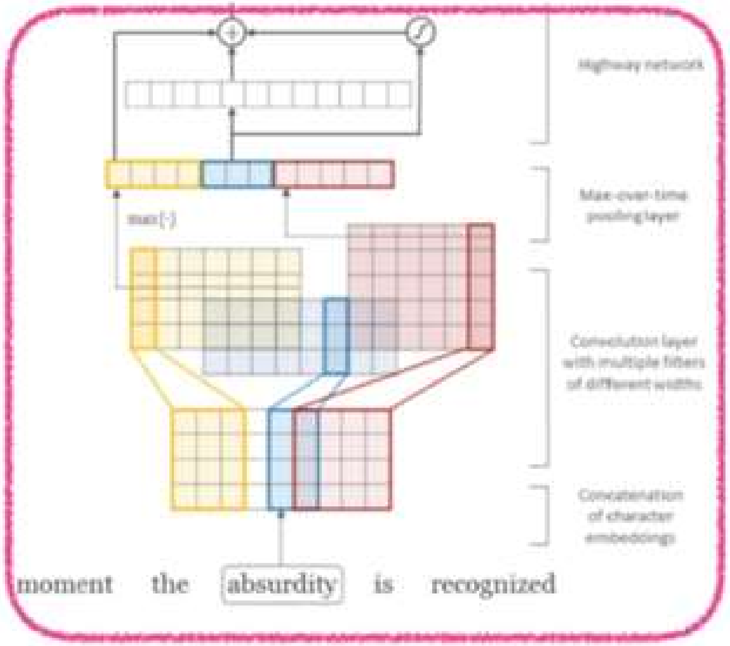
- Overview
  - Each token is assigned a representation that is a **function of the entire input sentence**
  - Use vectors derived from a **bidirectional LSTM** that is trained with **a coupled language model(LM) objective** on a large text corpus
- Features
  - ELMo representations are deep in the sense that they **are a function of all of the internal layers** of the biLM
    - A linear combination of the vectors stacked above each input word for each end task is learned, which markedly improves performance over just using the top LSTM layer
    - This allows for very rich word representations
      - Higher-level LSTM states captures context-dependent aspects of word meaning
      - Lower-level state model aspects of syntax

Character-based convolution layer

- No **OOV** problem
- Each character is fed to a CNN that consists of filters with various sizes
- Max-pooling to each feature map and then concatenate max-pooled results to generate a vector
- Feed the vector to Highway network layer

$$\mathbf{y} = H(\mathbf{x}, \mathbf{W_H}) \cdot T(\mathbf{x}, \mathbf{W_T}) + \mathbf{x} \cdot C(\mathbf{x}, \mathbf{W_C}). \quad (2)$$

$$\mathbf{y} = H(\mathbf{x}, \mathbf{W_H}) \cdot T(\mathbf{x}, \mathbf{W_T}) + \mathbf{x} \cdot (1 - T(\mathbf{x}, \mathbf{W_T})). \quad (3)$$



biLM(bidirectional Language Model)

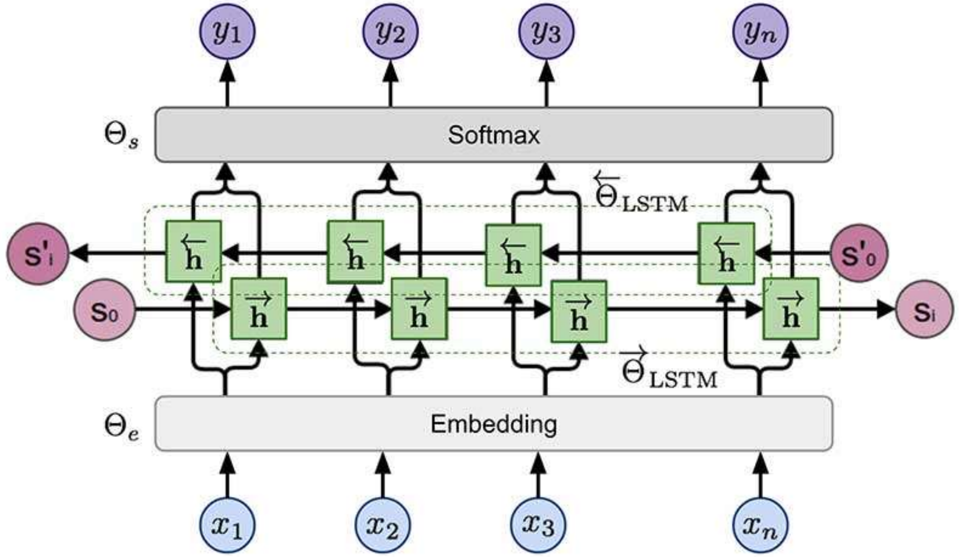
- Input : a sequence of n tokens (x_1,…, x_n)
- Learn to predict the prob. of a token given the token history
  - In forward pass, predict the next token after the given tokens

$$p(x_1, \ldots, x_n) = \prod_{i=1}^{n} p(x_i \mid x_1, \ldots, x_{i-1})$$

  - In backward pass, predict the previous token before the given tokens

$$p(x_1, \ldots, x_n) = \prod_{i=1}^{n} p(x_i \mid x_{i+1}, \ldots, x_n)$$

- Final layer's hidden state $\mathbf{h}_{i,L} = [\overrightarrow{\mathbf{h}}_{i,L}; \overleftarrow{\mathbf{h}}_{i,}$



$$\mathcal{L} = -\sum_{i=1}^{n} \Big(\log p(x_i \mid x_1, \ldots, x_{i-1}; \Theta_e, \overrightarrow{\Theta}_{\text{LSTM}}, \Theta_s) + \log p(x_i \mid x_{i+1}, \ldots, x_n; \Theta_e, \overleftarrow{\Theta}_{\text{LSTM}}, \Theta_s) \Big)$$

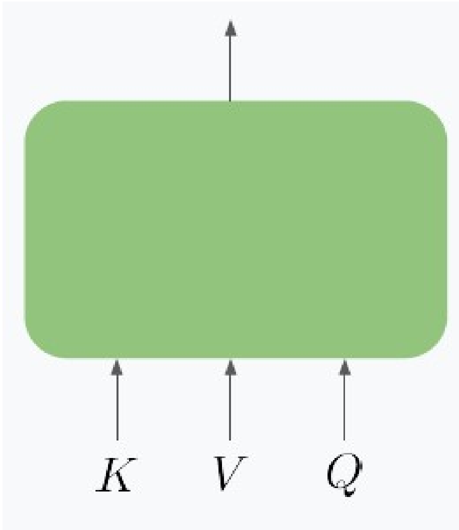
Transformers: scaled dot-product attention

Queries, keys and values

A *summary* of values, based on how similar their corresponding keys are with the query

$$O_i = \sum_{j=0}^{l} a_{ij} V_j$$

For some similarity function ϕ



$$a_{ij} = \frac{\phi(Q_i, K_j)}{\sum_{p=0}^{l} \phi(Q_i, K_p)}$$

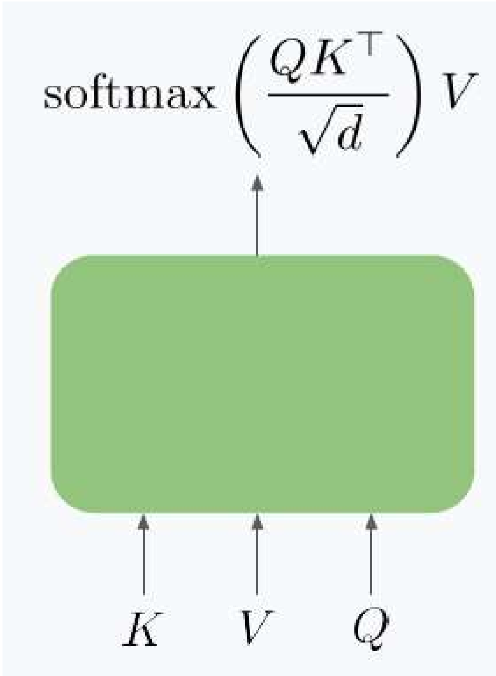
Scaled dot-product attention

Using dot-product similarity,
We can vectorize nicely

$$\emptyset(Q_i, K_j) = \exp\left(\frac{Q_i K_j^T}{\sqrt{d}}\right)$$

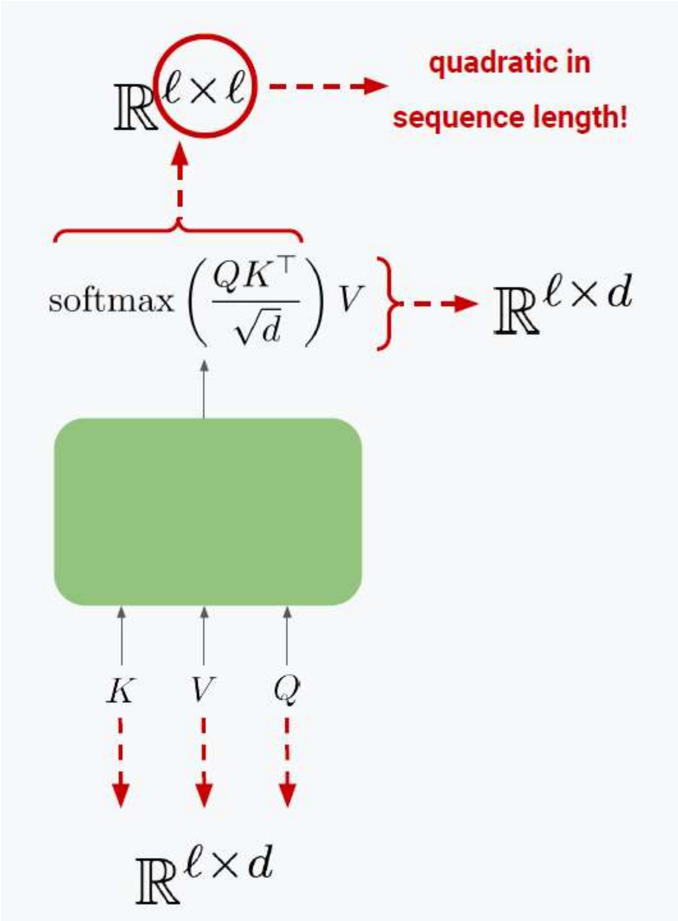
d = feature dimension
(Normalization factor for
numerical stability)

$$\text{softmax}(x)_i = \frac{\exp x_i}{\sum_j \exp x_j}$$



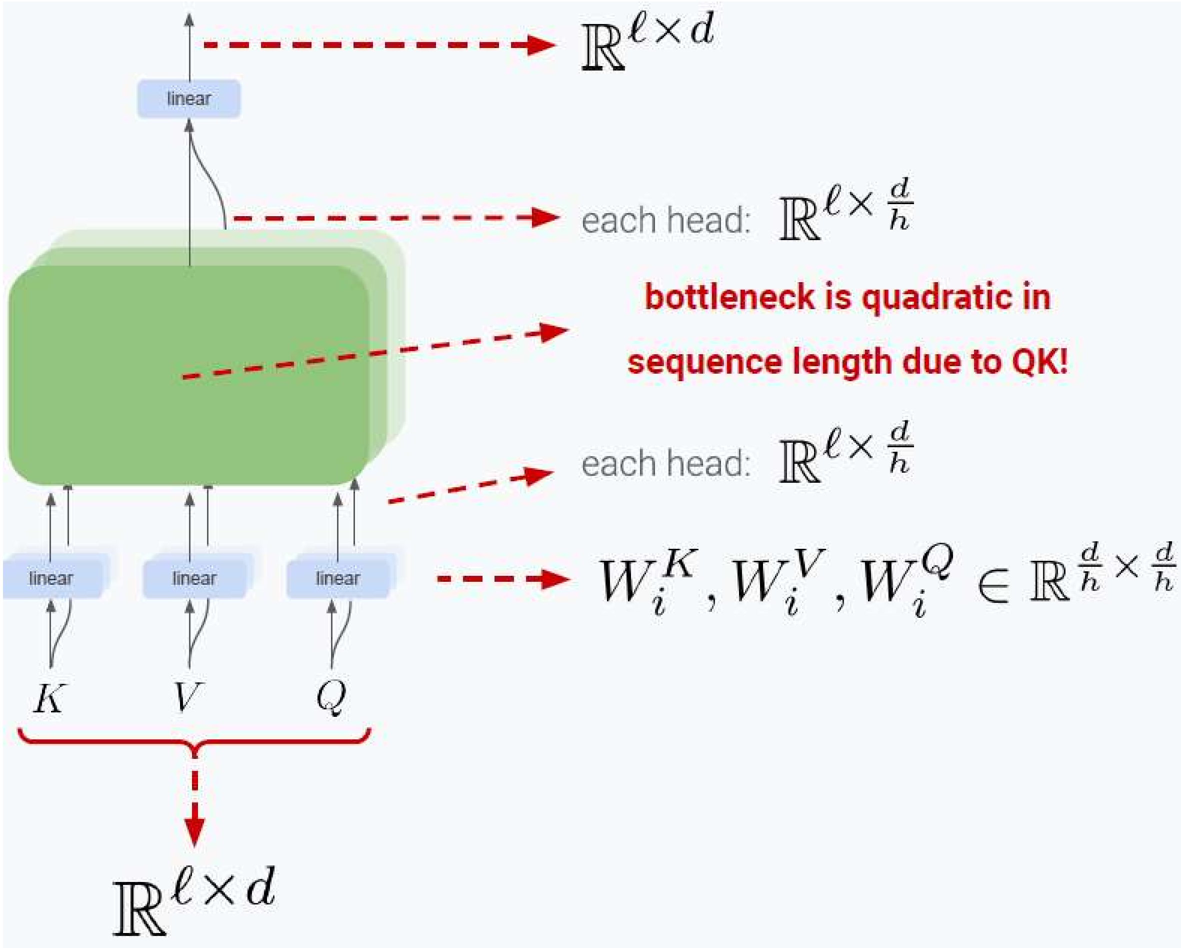
Scaled Dot-Product Attention

- l = sequence length
- d = feature dimension

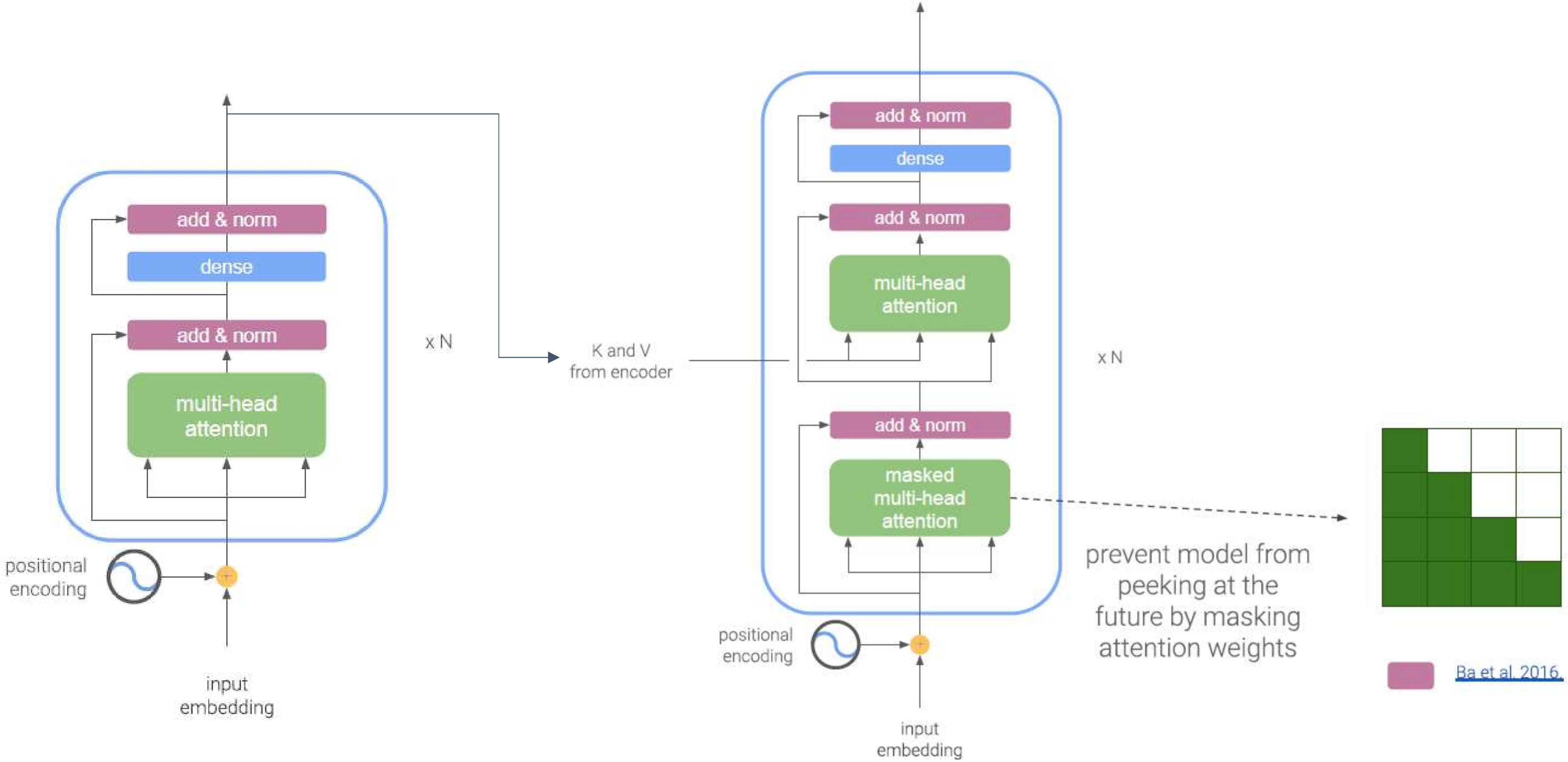


Multi-head attention

- l = sequence length
- d = feature dimension
- h = # of attention heads



Transformer

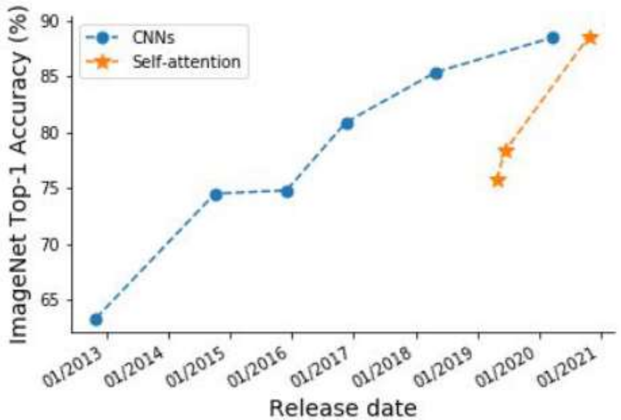


Transformers in recent literature

- Transformers have become successful in a wide range of domains and applications including:
  - Mathematics and theorem proving (e.g., Lample et al., 2019, Clark et al., 2020)
  - Music generation (e.g., Anna Huang et la., 2019)
  - Biology (Madani et al., 2020)
  - Vision Language (e.g., Tan et al., 2019, Chen et al., 2020)
  - Computer vision(e.g., Ramachandran et al., 2019, Dosovitskiy et al., 2020)

How many slices of pizza are there?
Is this a vegetarian pizza?

Visual Question Answering
(Agrawal et al. 2015)



Transformers in NLP

- Transformers are ubiquitous in NLP
- Large-scale pre-training has been enormously successful (e.g., BERT, ALBERT, T5, GPT-3)

Rank	Name	Model	URL	Score	CoLA	SST-2	MRPC	STS-B	QQP	MNLI-m	MNLI-mm	QNLI
1	ERNIE Team - Baidu	ERNIE		90.9	74.4	97.8	93.9/91.8	93.0/92.6	75.2/90.9	91.9	91.4	97.3
2	DeBERTa Team - Microsoft	DeBERTa / TuringNLRv4		90.8	71.5	97.5	94.0/92.0	92.9/92.6	76.2/90.8	91.9	91.6	99.2
3	HFL iFLYTEK	MacALBERT + DKM		90.7	74.8	97.0	94.5/92.6	92.8/92.6	74.7/90.6	91.3	91.1	97.8
4	Alibaba DAMO NLP	StructBERT + TAPT		90.6	75.3	97.3	93.9/91.9	93.2/92.7	74.8/91.0	90.9	90.7	97.4
5	PING-AN Omni-Sinitic	ALBERT + DAAF + NAS		90.6	73.5	97.2	94.0/92.0	93.0/92.4	76.1/91.0	91.6	91.3	97.5
6	T5 Team - Google	T5		90.3	71.6	97.5	92.8/90.4	93.1/92.8	75.1/90.6	92.2	91.9	96.9
7	Microsoft D365 AI & MSR AI & GATECH	MT-DNN-SMART		89.9	69.5	97.5	93.7/91.6	92.9/92.5	73.9/90.2	91.0	90.8	99.2
8	Huawei Noah's Ark Lab	NEZHA-Large		89.8	71.7	97.3	93.3/91.0	92.4/91.9	75.2/90.7	91.5	91.3	96.2
9	Zihang Dai	Funnel-Transformer (Ensemble B10-10-10H1024)		[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]/90.7	91.4	91.1	95.8
10	ELECTRA Team	ELECTRA-Large + Standard Tricks		89.4	71.7	97.1	93.1/90.7	92.9/92.5	75.6/90.8	91.3	90.8	95.8
11	liangzhu ge	deberta-xxlarge + standard tricks		89.4	71.9	96.6	92.0/89.4	93.0/92.6	74.9/90.4	91.3	91.1	95.9
12	Microsoft D365 AI & UMD	FreeLB-RoBERTa (ensemble)		88.4	68.0	96.8	93.1/90.8	92.3/92.1	74.8/90.3	91.1	90.7	95.6
13	Junjie Yang	HIRE-RoBERTa		88.3	68.6	97.1	93.0/90.7	92.4/92.0	74.3/90.2	90.7	90.4	95.5
14	Facebook AI	RoBERTa		88.1	67.8	96.7	92.3/89.8	92.2/91.9	74.3/90.2	90.8	90.2	95.4
15	Microsoft D365 AI & MSR AI	MT-DNN-ensemble		87.6	68.4	96.5	92.7/90.3	91.1/90.7	73.7/89.9	87.9	87.4	96.0
16	GLUE Human Baselines	GLUE Human Baselines		87.1	66.4	97.8	86.3/80.8	92.7/92.6	59.5/80.4	92.0	92.8	91.2

Funnel-Transformer (Ensemble B10-10-10H1024)

Click on a submission to see more information

Transformers in NLP

- Transformers are ubiquitous in NLP
- Large-scale pre-training has been enormously successful (e.g., BERT, ALBERT, T5, GPT-3)
- Models are typically used in 3 scenarios

Pre-training

- Large corpus (e.g. web crawled data)
- Typically unsupervised (e.g. masked language modeling)
- Usually runs in GPUs or TPUs

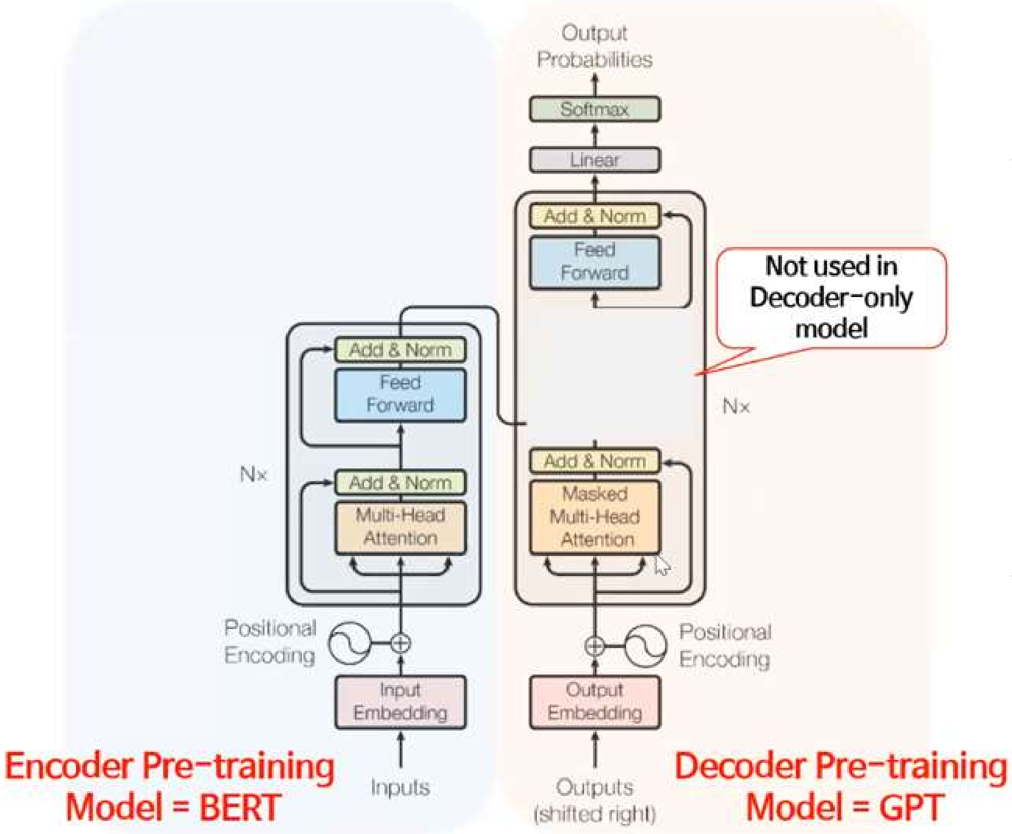
Fine-tuning

- Smaller corpus
- Typically supervised (e.g. question answering, natural language inference)
- Usually runs in GPUs or TPUs

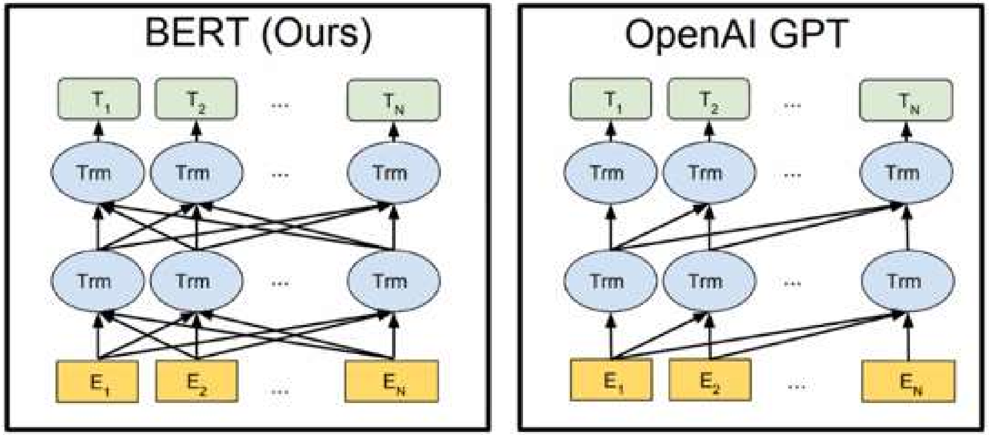
Production

- Inference
- Usually runs in CPUs, sometimes in mobile devices

BERT & GPT



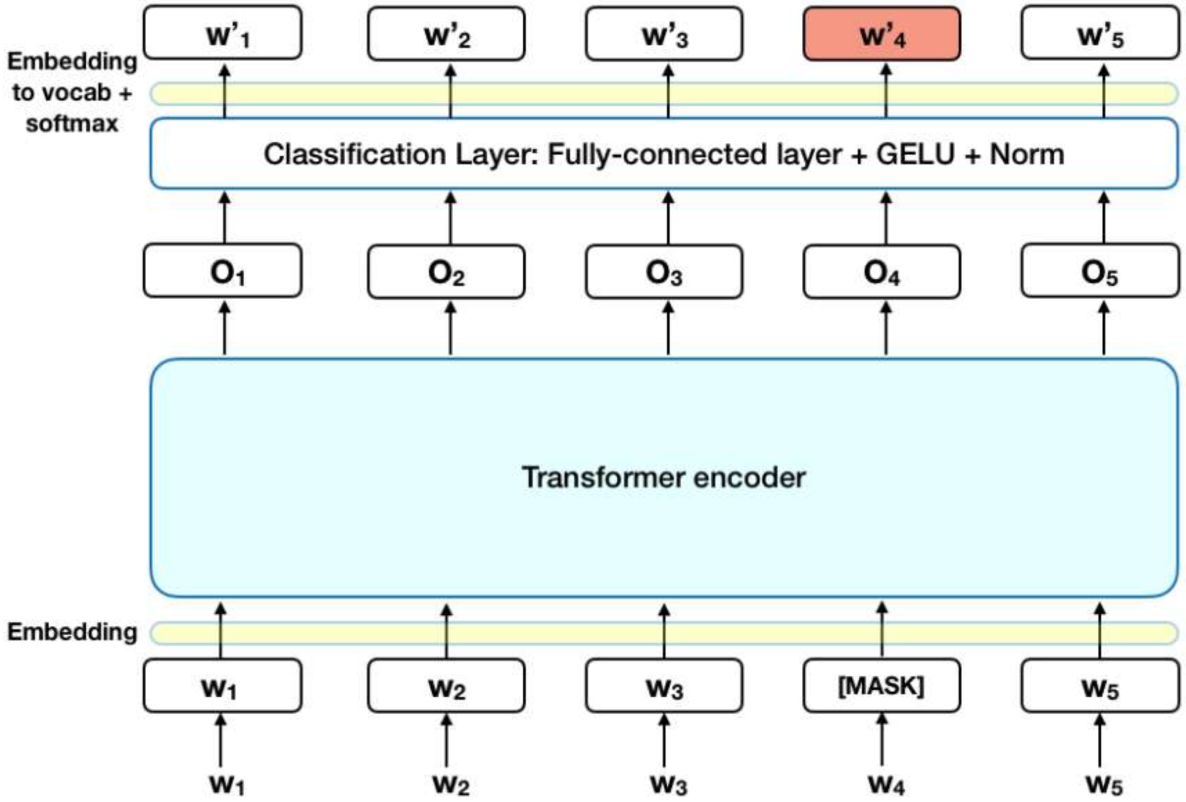
Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)
Not used in Decoder-only model
Encoder Pre-training Model = BERT
Decoder Pre-training Model = GPT



BERT (Ours)
OpenAI GPT
T_1 T_2 ... T_N
Trm Trm ... Trm
E_1 E_2 ... E_N

BERT

- **Masked language modeling** instead of predicting every next token
  - 15% tokens are chosen at random
    - 80% are actually replaced with the token [MASK]
    - 10% are replaced with a random token
    - 10% are left unchanged
- **NSP**(Next Sentence Prediction)



- Predicting masked token

```
                      store             gallon
                        ↑                  ↑
the man went to the [MASK] to buy a [MASK] of milk
```

- Next Sentence Prediction
  - Binary classification task if the 2nd sentence is the actual next sentence of the first one

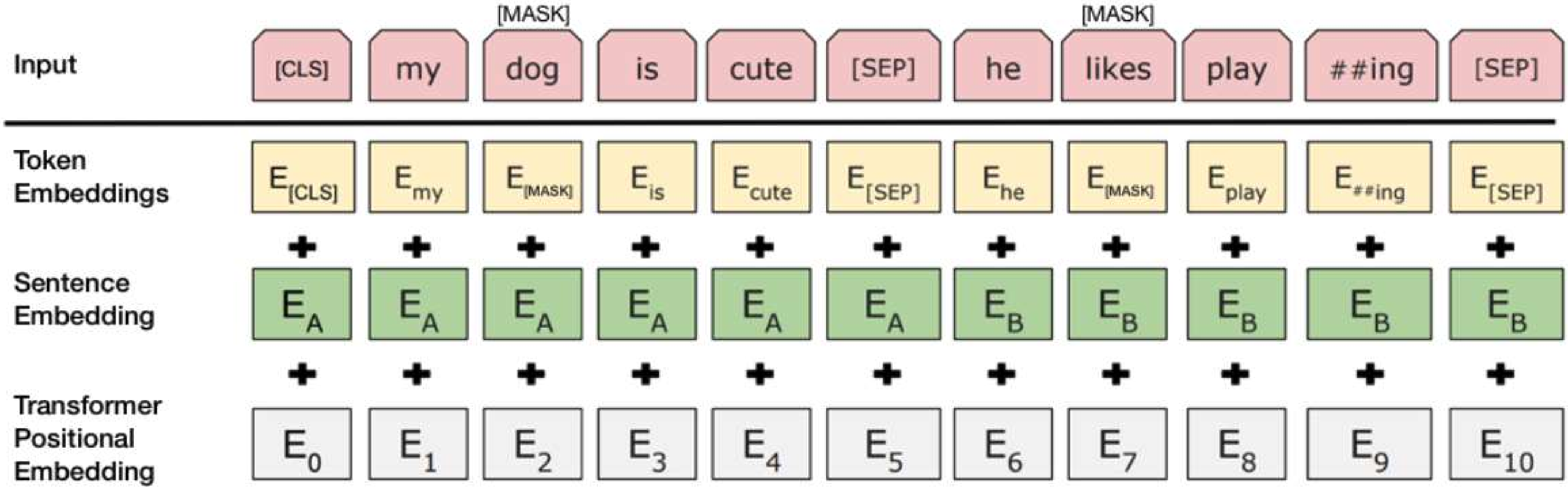
Input = [CLS] the man went to [MASK] store [SEP] he bought a gallon [MASK] milk [SEP]

Label = IsNext

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

Label = NotNext

- These two tasks are self-unsupervised


Input
[MASK]
[MASK]
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} $E_{[MASK]}$ E_{is} E_{cute} $E_{[SEP]}$ E_{he} $E_{[MASK]}$ E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Sentence Embedding
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Transformer Positional Embedding
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

Subword-based encoding: Byte Pair Encoding

- Originally a compression algorithm
  - Most frequent byte pair -> a new byte

Replace bytes with character ngrams

(though, actually, some people have done interesting things with bytes)

Rico Sennrich, Barry Haddow, and Alexandra Birch. ***Neural Machine Translation of Rare Words with Subword Units****. ACL 2016.*
https://arxiv.org/abs/1508.07909
https://github.com/rsennrich/subword-nmt
https://github.com/EdinburghNLP/nematus

Byte Pair Encoding

- A word segmentation algorithm:
    - Though done as bottom-up clustering
  - Start with a unigram vocabulary of all (Unicode) **characters** in data
  - Most frequent **ngram pairs** → a new **ngram**

Dictionary

5	l o w
2	l o w e r
6	n e w e s t
3	w i d e s t

Vocabulary

l, o, w, e, r, n, w, s, t, i, d

Start with all characters in vocab

Byte Pair Encoding

- A word segmentation algorithm:
    - Though done as bottom-up clustering
  - Start with a unigram vocabulary of all (Unicode) **characters** in data
  - Most frequent **ngram pairs** → a new **ngram**

Dictionary

5 l o w
2 l o w e r
6 n e w **es** t
3 w i d **es** t

Vocabulary

l, o, w, e, r, n, w, s, t, i, d, **es**

Add a pair (e, s) with freq 9

Byte Pair Encoding

- A word segmentation algorithm:
    - Though done as bottom-up clustering
  - Start with a unigram vocabulary of all (Unicode) **characters** in data
  - Most frequent **ngram pairs** → a new **ngram**

Dictionary

5	l o w
2	l o w e r
6	n e w **est**
3	w i d **est**

Vocabulary

l, o, w, e, r, n, w, s, t, i, d, es, **est**

Add a pair (es, t) with freq 9

Byte Pair Encoding

- A word segmentation algorithm:
    - Though done as bottom-up clustering
  - Start with a unigram vocabulary of all (Unicode) **characters** in data
  - Most frequent **ngram pairs** → a new **ngram**

Dictionary

5 **lo** w
2 **lo** w e r
6 n e w est
3 w i d est

Vocabulary

l, o, w, e, r, n, w, s, t, i, d, es, est, **lo**

Add a pair (l, o) with freq 7

Byte Pair Encoding

- Have a target vocabulary size and stop when you reach it
- Do deterministic longest piece segmentation of words
- Segmentation is only within words identified by some prior tokenizer

- Automatically decides vocabulary for systems
  - No longer strongly “word” based in conventional way

WordPiece/SentencePiece model

- Google NMT (GNMT) uses a variant of BPE
  - V1: WordPiece model
  - V2: SentencePiece model
- Rather than char n-gram count, uses a greedy approximation to maximizing LM log likelihood to choose the piece
  - Add n-gram that maximally reduces perplexity
- WordPiece model tokenizes inside words
- SentencePiece model works from raw text
  - Whitespace is retained as special token (_) and grouped normaly
  - You can reverse things at end by joining pieces and recoding them to space

WordPiece/SentencePiece model

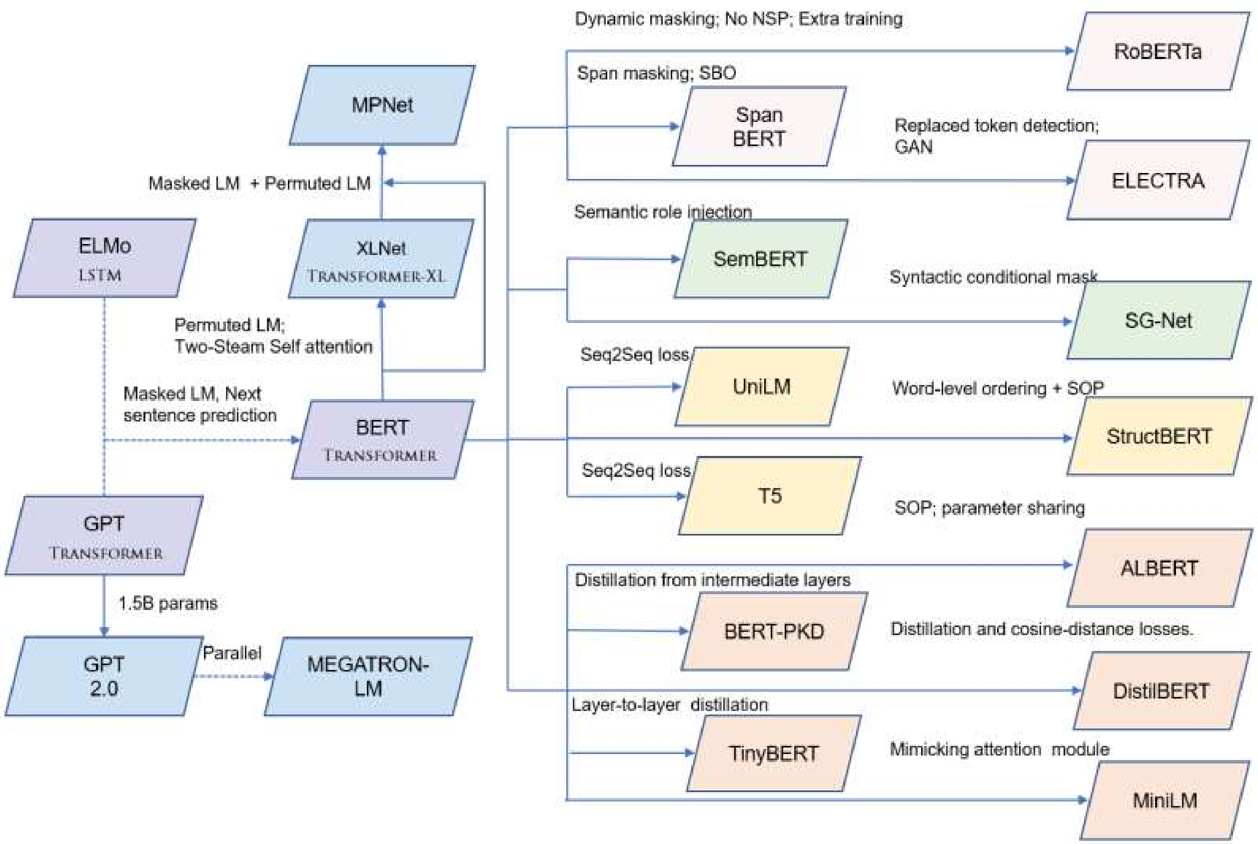
- BERT uses a variant of the wordpiece model
  - (relatively) common words are in the vocab:
    - At firafax, 1910s
  - Other words are built from wordpieces:
    - Hypatia = h ##yp ##ati ##a

GLUE benchmark

- General Language Understanding Evaluation
  - A collection of sentence- or sentence-pair lang. understanding tasks
  - Built on established existing datasets and selected to cover a diverse range of dataset sizes, text genres, and degrees of difficulty

Dataset	Description	Data example	Metric
CoLA	Is the sentence grammatical or ungrammatical?	"This building is than that one." = **Ungrammatical**	Matthews
SST-2	Is the movie review positive, negative, or neutral?	"The movie is funny , smart , visually inventive , and most of all , alive ." = **.93056 (Very Positive)**	Accuracy
MRPC	Is the sentence B a paraphrase of sentence A?	A) "Yesterday , Taiwan reported 35 new infections , bringing the total number of cases to 418 ." B) "The island reported another 35 probable cases yesterday , taking its total to 418 ." = **A Paraphrase**	Accuracy / F1
STS-B	How similar are sentences A and B?	A) "Elephants are walking down a trail." B) "A herd of elephants are walking along a trail." = **4.6 (Very Similar)**	Pearson / Spearman
QQP	Are the two questions similar?	A) "How can I increase the speed of my internet connection while using a VPN?" B) "How can Internet speed be increased by hacking through DNS?" = **Not Similar**	Accuracy / F1
MNLI-mm	Does sentence A entail or contradict sentence B?	A) "Tourist Information offices can be very helpful." B) "Tourist Information offices are never of any help." = **Contradiction**	Accuracy
QNLI	Does sentence B contain the answer to the question in sentence A?	A) "What is essential for the mating of the elements that create radio waves?" B) "Antennas are required by any radio receiver or transmitter to couple its electrical connection to the electromagnetic field." = **Answerable**	Accuracy
RTE	Does sentence A entail sentence B?	A) "In 2003, Yunus brought the microcredit revolution to the streets of Bangladesh to support more than 50,000 beggars, whom the Grameen Bank respectfully calls Struggling Members." B) "Yunus supported more than 50,000 Struggling Members." = **Entailed**	Accuracy
WNLI	Sentence B replaces sentence A's ambiguous pronoun with one of the nouns - is this the correct noun?	A) "Lily spoke to Donna, breaking her concentration." B) "Lily spoke to Donna, breaking Lily's concentration." = **Incorrect Referent**	Accuracy

Encoder-based LM (BERT family)



* The Role of Contextualized Language Models and Beyond, Zhang et al., 2020

RoBERTa

- A Robustly Optimized BERT Pretraining Approach by Facebook
  - Key points #1: Static vs. Dynamic Masking
    - Generate the masking pattern every time we feed a sequence to the model
    - Much crucial when pre-training for more steps or with larger datasets
  - Key points #2: Model Input Format and NSP
    - SEGMENT-PAIR+ NSP
    - SENTENCE-PAIR+NSP
    - Full-sentences
    - Doc-sentences

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

Table 1: Comparison between static and dynamic masking for $\text{BERT}_{\text{BASE}}$. We report F1 for SQuAD and accuracy for MNLI-m and SST-2. Reported results are medians over 5 random initializations (seeds). Reference results are from Yang et al. (2019).

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

Table 2: Development set results for base models pretrained over BOOKCORPUS and WIKIPEDIA. All models are trained for 1M steps with a batch size of 256 sequences. We report F1 for SQuAD and accuracy for MNLI-m, SST-2 and RACE. Reported results are medians over five random initializations (seeds). Results for $\text{BERT}_{\text{BASE}}$ and $\text{XLNet}_{\text{BASE}}$ are from Yang et al. (2019).

RoBERTa

- Key point #3 : training with large batches

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

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

Perplexity(ppl) is the inverse prob. of the test set, normalized by the # of words

$$PPL(W) = P(w_1, w_2, w_3, \ldots, w_N)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P(w_1, w_2, w_3, \ldots, w_N)}}$$

$$PPL(W) = \sqrt[N]{\frac{1}{P(w_1, w_2, w_3, \ldots, w_N)}} = \sqrt[N]{\frac{1}{\prod_{i=1}^{N} P(w_i|w_1, w_2, \ldots, w_{i-1})}}$$

$$PPL(W) = \sqrt[N]{\frac{1}{\prod_{i=1}^{N} P(w_i|w_{i-1})}}$$, For bigram

RoBERTa

Model	data	bsz	steps	SQuAD (v1.1/2.0)	MNLI-m	SST-2
RoBERTa						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	**94.6/89.4**	**90.2**	**96.4**
$BERT_{LARGE}$						
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7
$XLNet_{LARGE}$						
with BOOKS + WIKI	13GB	256	1M	94.0/87.8	88.4	94.4
+ additional data	126GB	2K	500K	94.5/88.8	89.8	95.6

Table 4: Development set results for RoBERTa as we pretrain over more data (16GB → 160GB of text) and pretrain for longer (100K → 300K → 500K steps). Each row accumulates improvements from the rows above. RoBERTa matches the architecture and training objective of $BERT_{LARGE}$. Results for $BERT_{LARGE}$ and $XLNet_{LARGE}$ are from Devlin et al. (2019) and Yang et al. (2019), respectively. Complete results on all GLUE tasks can be found in the

Model	SQuAD 1.1 EM	SQuAD 1.1 F1	SQuAD 2.0 EM	SQuAD 2.0 F1
Single models on dev, w/o data augmentation				
$BERT_{LARGE}$	84.1	90.9	79.0	81.8
$XLNet_{LARGE}$	**89.0**	94.5	86.1	88.8
RoBERTa	88.9	**94.6**	**86.5**	**89.4**
Single models on test (as of July 25, 2019)				
$XLNet_{LARGE}$			86.3†	89.1†
RoBERTa			86.8	89.8
XLNet + SG-Net Verifier			**87.0**†	**89.9**†

Table 6: Results on SQuAD. † indicates results that depend on additional external training data. RoBERTa uses only the provided SQuAD data in both dev and test settings. $BERT_{LARGE}$ and $XLNet_{LARGE}$ results are from Devlin et al. (2019) and Yang et al. (2019), respectively.

	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STS	WNLI	Avg
Single-task single models on dev										
$BERT_{LARGE}$	86.6/-	92.3	91.3	70.4	93.2	88.0	60.6	90.0	-	-
$XLNet_{LARGE}$	89.8/-	93.9	91.8	83.8	95.6	89.2	63.6	91.8	-	-
RoBERTa	**90.2/90.2**	**94.7**	**92.2**	**86.6**	**96.4**	**90.9**	**68.0**	**92.4**	**91.3**	-
Ensembles on test (from leaderboard as of July 25, 2019)										
ALICE	88.2/87.9	95.7	**90.7**	83.5	95.2	92.6	**68.6**	91.1	80.8	86.3
MT-DNN	87.9/87.4	96.0	89.9	86.3	96.5	92.7	68.4	91.1	89.0	87.6
XLNet	90.2/89.8	98.6	90.3	86.3	**96.8**	**93.0**	67.8	91.6	**90.4**	88.4
RoBERTa	**90.8/90.2**	**98.9**	90.2	**88.2**	96.7	92.3	67.8	**92.2**	89.0	**88.5**

Table 5: Results on GLUE. All results are based on a 24-layer architecture. $BERT_{LARGE}$ and $XLNet_{LARGE}$ results are from Devlin et al. (2019) and Yang et al. (2019), respectively. RoBERTa results on the development set are a median over five runs. RoBERTa results on the test set are ensembles of *single-task* models. For RTE, STS and MRPC we finetune starting from the MNLI model instead of the baseline pretrained model. Averages are obtained from the GLUE leaderboard.

ALBERT: A lite BERT for Self-supervised Learning of Language Representation (Google)

- Observation
  - Simply growing the hidden size of a model such as BERT-large can lead to worse performance

Model	Hidden Size	Parameters	RACE (Accuracy)
BERT-large (Devlin et al., 2019)	1024	334M	72.0%
BERT-large (ours)	1024	334M	73.9%
BERT-xlarge (ours)	2048	1270M	54.3%

Table 1: Increasing hidden size of BERT-large leads to worse performance on RACE.

* RACE: Large-scale ReAding Comprehension Dataset From Examinations, ACL 2017

ALBERT

- Key points #1: Factorized embedding parameterization
  - WordPiece embeddings are meant to learn context-independent representations, whereas hidden-layer embeddings are meant to learn context-dependent representations

  - Project the one-hot vectors into a lower dimensional embedding space of size E, and then project it to the hidden space.
    - Reduce the embedding parameters: $O(V \times H) \rightarrow O(V \times E + E \times H)$.
    - This parameter reduction is significant when H 〉〉 E

    - (실험 파라미터) E 128 / H 768, 1024, 2048, 4096

UNIVERSITY OF SCIENCE & TECHNOLOGY

Korea Institute of Science and Technology Information

ALBERT

- Key points #2: Cross-layer parameter sharing
  - Share all parameters across layers

  - The transitions from layer to layer are much smoother for ALBERT than for BERT
    - weight-sharing has an effect on stabilizing network parameters

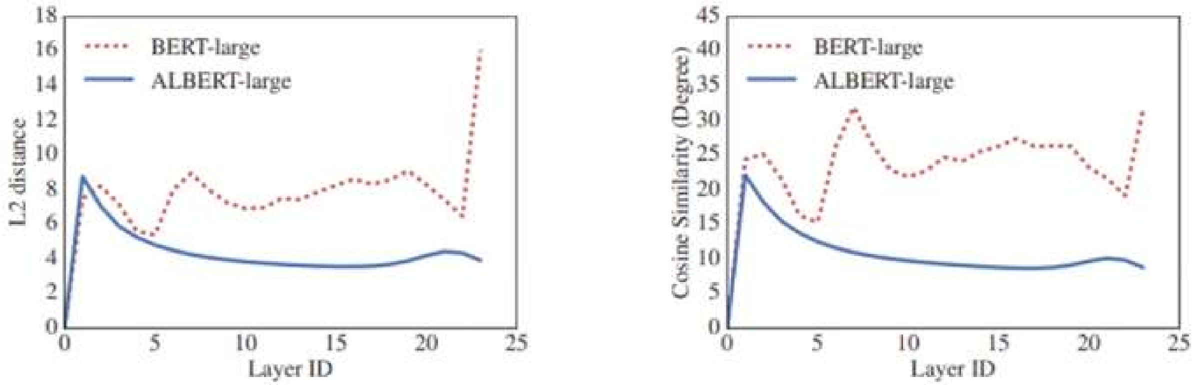


Figure 2: The L2 distances and cosine similarity (in terms of degree) of the input and output embedding of each layer for BERT-large and ALBERT-large.

ALBERT

- Key points #3: Inter-sentence coherence loss
  - Conjecture that the main reason behind NSP's ineffectiveness
    - lack of difficulty as a task
    - NSP conflates topic prediction and coherence prediction in a single task
    - topic prediction is easier to learn, and also overlaps more with what is learned using the MLM loss
  - Sentence-order prediction (SOP) loss
    - Negative examples the same two consecutive segments but with their order swapped.
    - This forces the model to learn finer-grained distinctions about discourse-level coherence properties

ALBERT

Models	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STS	WNLI	Avg
Single-task single models on dev										
BERT-large	86.6	92.3	91.3	70.4	93.2	88.0	60.6	90.0	-	-
XLNet-large	89.8	93.9	91.8	83.8	95.6	89.2	63.6	91.8	-	-
RoBERTa-large	90.2	94.7	**92.2**	86.6	96.4	**90.9**	68.0	92.4	-	-
ALBERT (1M)	90.4	95.2	92.0	88.1	96.8	90.2	68.7	92.7	-	-
ALBERT (1.5M)	**90.8**	**95.3**	**92.2**	**89.2**	**96.9**	**90.9**	**71.4**	**93.0**	-	-
Ensembles on test (from leaderboard as of Sept. 16, 2019)										
ALICE	88.2	95.7	**90.7**	83.5	95.2	92.6	**69.2**	91.1	80.8	87.0
MT-DNN	87.9	96.0	89.9	86.3	96.5	92.7	68.4	91.1	89.0	87.6
XLNet	90.2	98.6	90.3	86.3	96.8	93.0	67.8	91.6	90.4	88.4
RoBERTa	90.8	98.9	90.2	88.2	96.7	92.3	67.8	92.2	89.0	88.5
Adv-RoBERTa	91.1	98.8	90.3	88.7	96.8	93.1	68.0	92.4	89.0	88.8
ALBERT	**91.3**	**99.2**	90.5	**89.2**	**97.1**	**93.4**	69.1	**92.5**	**91.8**	**89.4**

Table 13: State-of-the-art results on the GLUE benchmark. For single-task single-model results, we report ALBERT at 1M steps (comparable to RoBERTa) and at 1.5M steps. The ALBERT ensemble uses models trained with 1M, 1.5M, and other numbers of steps.

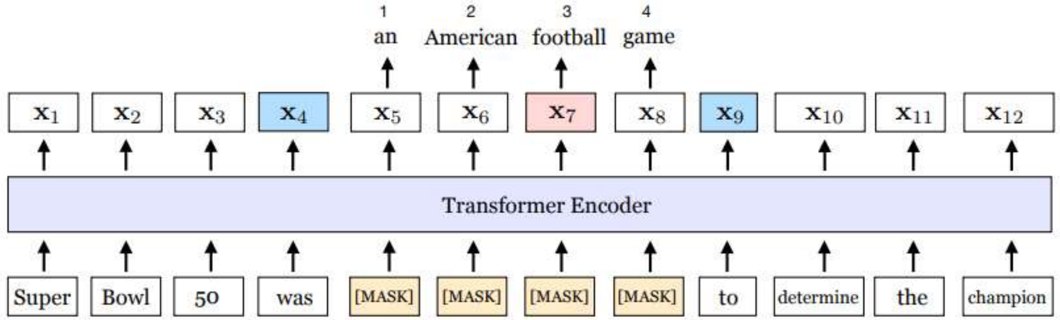
SpanBERT: Improving Pre-training by Representing and Predicting Spans

- Span-level pretraining
  - Masking contiguous random span rather than random tokens
  - Learning span boundary representation to predict whole masked tokens
- No use of NSP

$$\mathcal{L}(\text{football}) = \mathcal{L}_{\text{MLM}}(\text{football}) + \mathcal{L}_{\text{SBO}}(\text{football})$$
$$= -\log P(\text{football} \mid \mathbf{x}_7) - \log P(\text{football} \mid \mathbf{x}_4, \mathbf{x}_9, \mathbf{p}_3)$$

	SQuAD 1.1		SQuAD 2.0	
	EM	F1	EM	F1
Human Perf.	82.3	91.2	86.8	89.4
Google BERT	84.3	91.3	80.0	83.3
Our BERT	86.5	92.6	82.8	85.9
Our BERT-1seq	87.5	93.3	83.8	86.6
SpanBERT	**88.8**	**94.6**	**85.7**	**88.7**

Table 1: Test results on SQuAD 1.1 and SQuAD 2.0.



gure 1: An illustration of SpanBERT training. The span *an American football game* is masked. The span ›undary objective (SBO) uses the output representations of the boundary tokens, $\mathbf{x}_4$ and $\mathbf{x}_9$ (in blue), to predict ›ch token in the masked span. The equation shows the MLM and SBO loss terms for predicting the token, *football* ..ı pink), which as marked by the position embedding $\mathbf{p}_3$, is the *third* token from x_4.

K-BERT: Enabling Language Representation with Knowledge Graph

- Motivation
  - LM successfully capture a general language representation from large-scale corpora
  - But lack domain-specific knowledge

- K-BERT : a knowledge-enabled language representation model with knowledge graphs
  - Triples are injected into the sentences as dom knoweldge

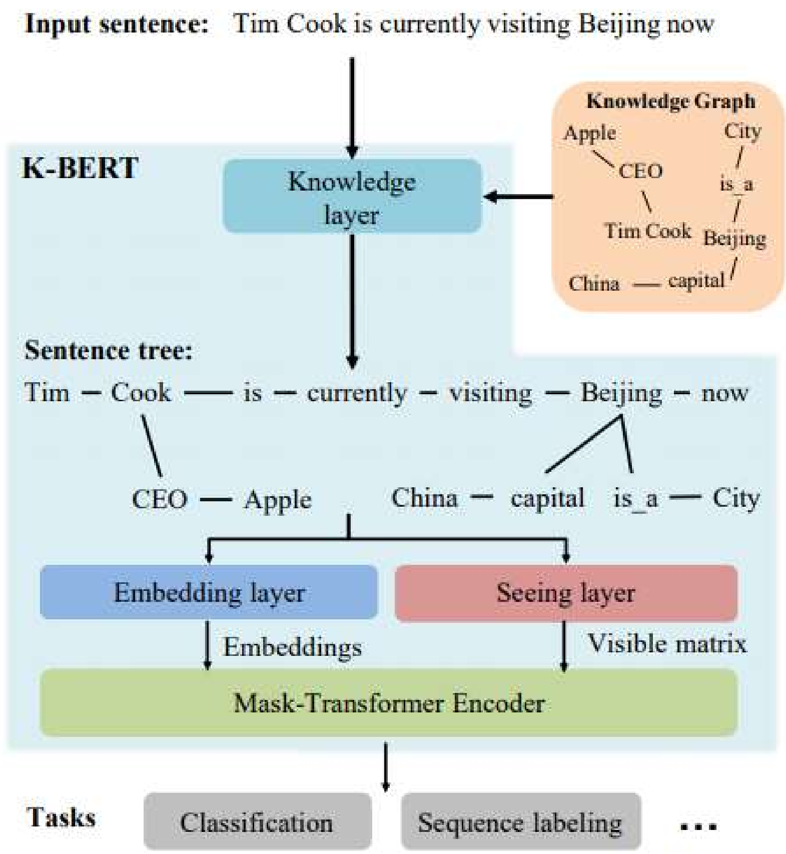


Figure 1: The model structure of K-BERT: Compared other RL models, the K-BERT is equipped with an editab KG, which can be adapted to its application domain. For e ample, for electronic medical record analysis, we can use medical KG to grant the K-BERT with medical knowledg

K-BERT

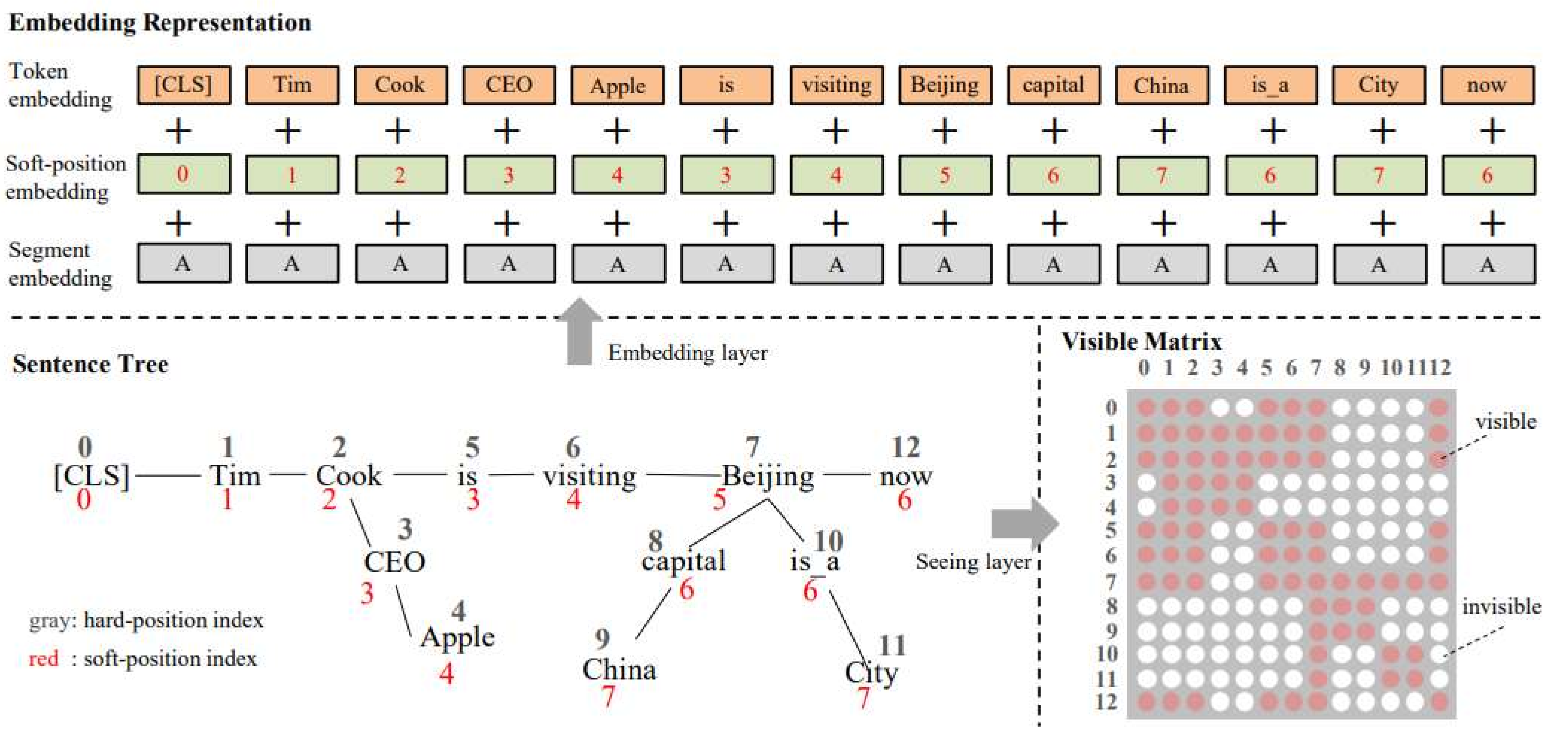


Figure 2: The process of converting a sentence tree into an embedding representation and a visible matrix. In the sentence tree, the red number is the soft-position index, and the gray is the hard-position index. **(1)** For token embedding, the tokens in the sentence tree are flattened into a sequence of token embedding by their hard-position index; **(2)** The soft-position index is used as position embedding along with the token embedding; **(3)** In segment embedding, all the tokens in the fist sentence are tagged as "A"; **(4)** In the visible matrix, red means visible, and white means invisible. For example, the cell at row 4, column 9 is white means that the "Apple(4)" cannot see "China(9)".

Table 1: Results of various models on sentence classification tasks on open-domain tasks (*Acc.* %)

Models\Datasets	**Book_review**		**Chnsenticorp**		**Shopping**		**Weibo**		**XNLI**		**LCQMC**	
	Dev	*Test*	*Dev*	*Test*	*Dev*	*Test*	*Dev*	*Test*	*Dev*	*Test*	*Dev*	*Test*
Pre-trainied on WikiZh by Google.												
Google BERT	88.3	**87.5**	93.3	94.3	96.7	96.3	98.2	98.3	76.0	75.4	88.4	86.2
K-BERT (HowNet)	88.6	87.2	**94.6**	**95.6**	**97.1**	**97.0**	98.3	98.3	**76.8**	**76.1**	**88.9**	86.9
K-BERT (CN-DBpedia)	**88.6**	87.3	93.9	95.3	96.6	96.5	98.3	98.3	76.5	76.0	88.6	**87.0**
Pre-trained on WikiZh and WebtextZh by us.												
Our BERT	**88.6**	87.9	94.8	95.7	96.9	**97.1**	98.2	98.2	77.0	76.3	89.0	86.7
K-BERT (HowNet)	88.5	87.4	**95.4**	95.6	96.9	96.9	98.3	**98.4**	**77.2**	**77.0**	**89.2**	**87.1**
K-BERT (CN-DBpedia)	88.8	87.9	95.0	**95.8**	**97.1**	97.0	98.3	98.3	76.2	75.9	89.0	86.9

Table 3: Results of various models on specific-domain tasks (%).

Models\Datasets	**Finance_Q&A**			**Law_Q&A**			**Finance_NER**			**Medicine_NER**		
	P.	*R.*	*F1*	*P.*	*R.*	*F1*	*P.*	*R.*	*F1*	*P.*	*R.*	*F1*
Pre-trained on WikiZh by Google.												
Google BERT	81.9	86.0	83.9	83.1	90.1	86.4	84.8	87.4	86.1	91.9	93.1	92.5
K-BERT (HowNet)	83.3	84.4	83.9	83.7	91.2	87.3	86.3	89.0	**87.6**	93.2	93.3	93.3
K-BERT (CN-DBpedia)	81.5	88.6	**84.9**	82.1	93.8	**87.5**	86.1	88.7	87.4	93.9	93.8	93.8
K-BERT (MedicalKG)	-	-	-	-	-	-	-	-	-	94.0	94.4	**94.2**
Pre-trained on WikiZh and WebtextZh by us.												
Our BERT	82.1	86.5	84.2	83.2	91.7	87.2	84.9	87.4	86.1	91.8	93.5	92.7
K-BERT (HowNet)	82.8	85.8	84.3	83.0	92.4	87.5	86.3	88.5	87.3	93.5	93.8	93.7
K-BERT (CN-DBpedia)	81.9	87.1	**84.4**	83.1	92.6	**87.6**	86.3	88.6	**87.4**	93.9	94.3	94.1
K-BERT (MedicalKG)	-	-	-	-	-	-	-	-	-	94.1	94.3	**94.2**

ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators

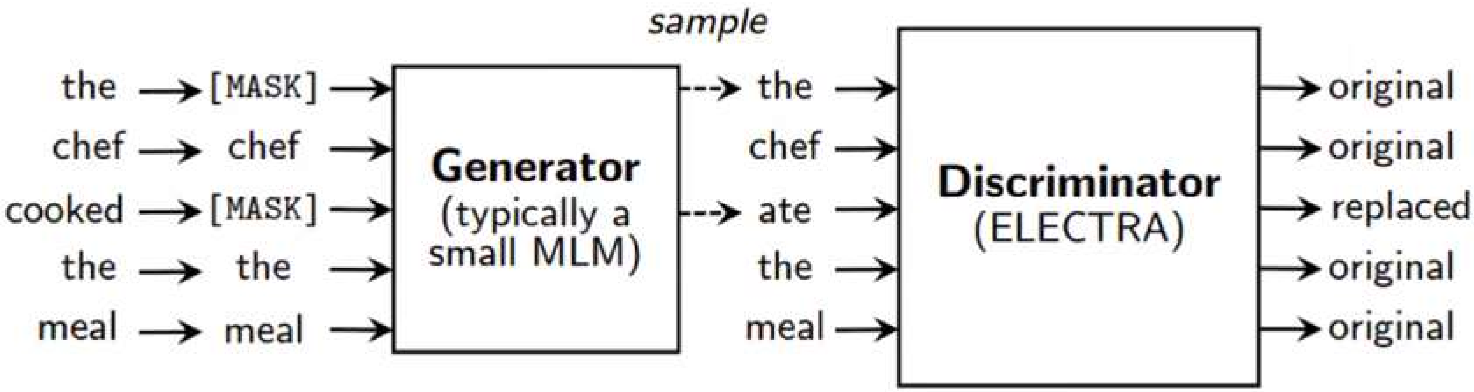


Figure 2: An overview of replaced token detection. The generator can be any model that produces an output distribution over tokens, but we usually use a small masked language model that is trained jointly with the discriminator. Although the models are structured like in a GAN, we train the generator with maximum likelihood rather than adversarially due to the difficulty of applying GANs to text. After pre-training, we throw out the generator and only fine-tune the discriminator (the ELECTRA model) on downstream tasks.

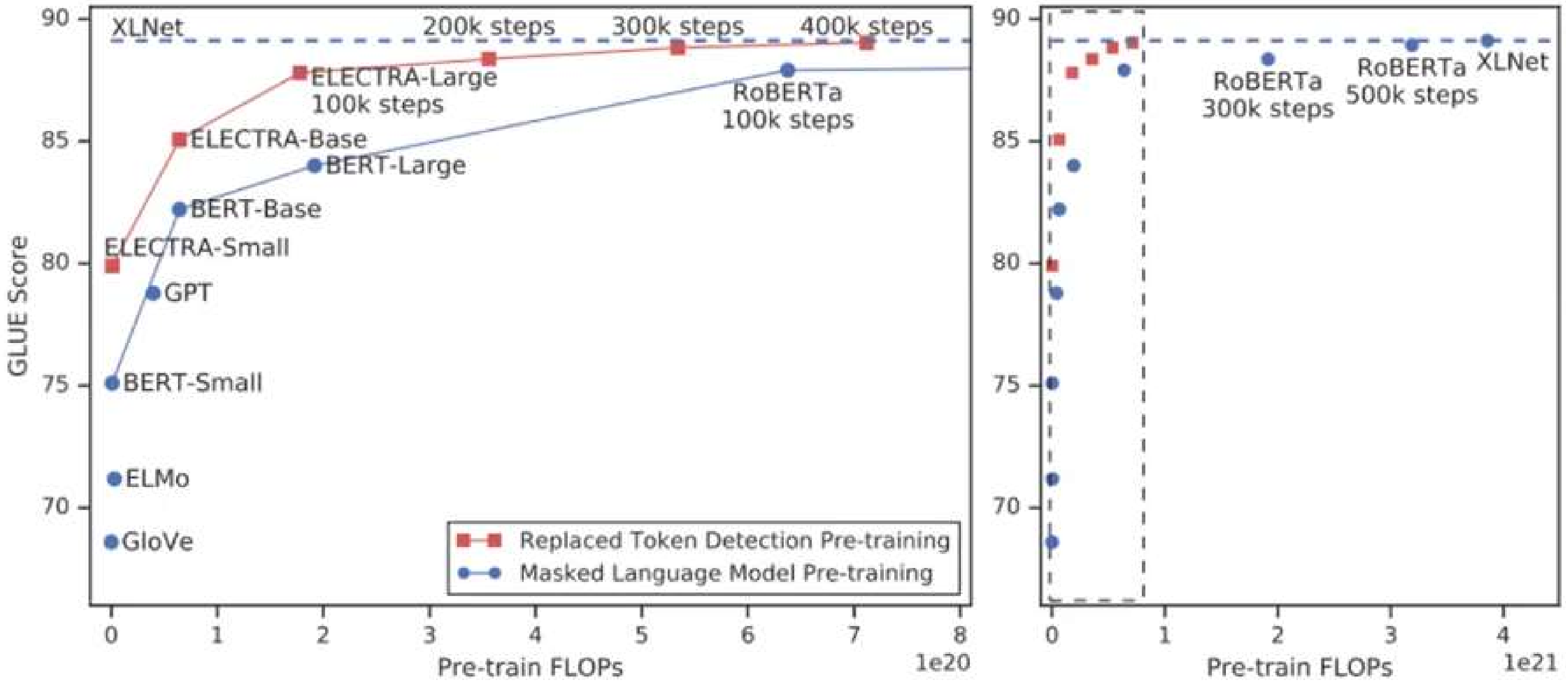


Figure 1: Replaced token detection pre-training consistently outperforms masked language model pre-training given the same compute budget. The left figure is a zoomed-in view of the dashed box.