HIT-SCIR

16TH CIPS SUMMER SCHOOL &
ADVANCED TECHNOLOGY TUTORIAL

（下）

自然语言处理：基于预训练模型的方法

NATURAL LANGUAGE PROCESSING: A PRE-TRAINED MODEL APPROACH

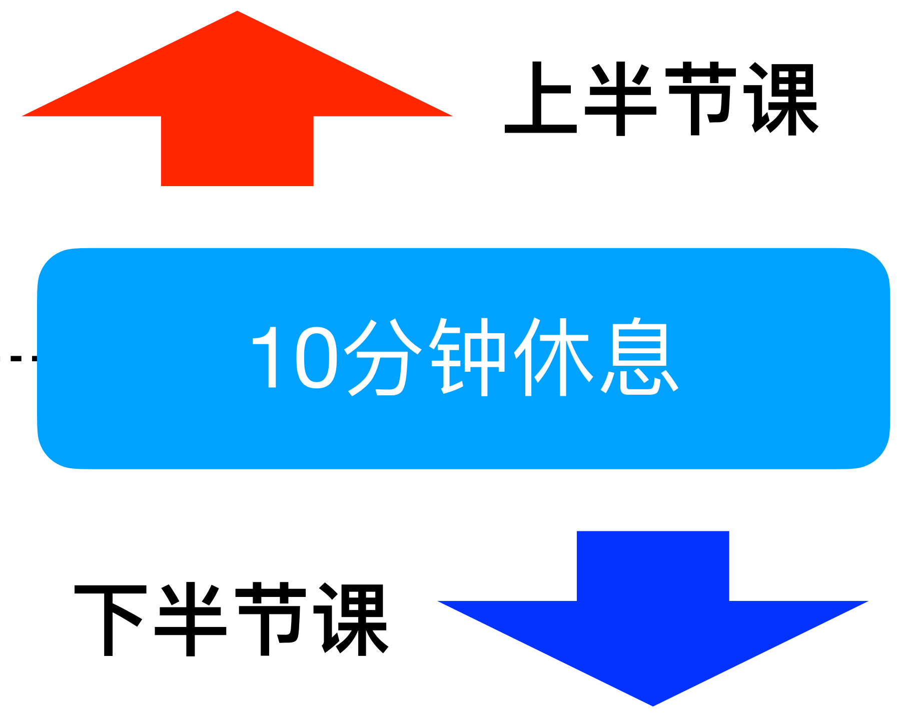
崔一鸣

哈工大讯飞联合实验室 (HFL)

2021年7月22日

目录

- 基于上下文的语言模型
  - CoVe, ELMo
- 预训练语言模型
  - GPT 系列 (GPT, GPT-2, GPT-3)
  - BERT 系列 (MLM, WWM, NM)
- 预训练语言模型进阶
  - XLNet, RoBERTa, ALBERT, ELECTRA, MacBERT, T5
- 面向预训练模型的知识蒸馏
  - DistilBERT, TinyBERT, MobileBERT, TextBrewer
- 总结



源代码

论文 / 资源

特别提示

开源项目

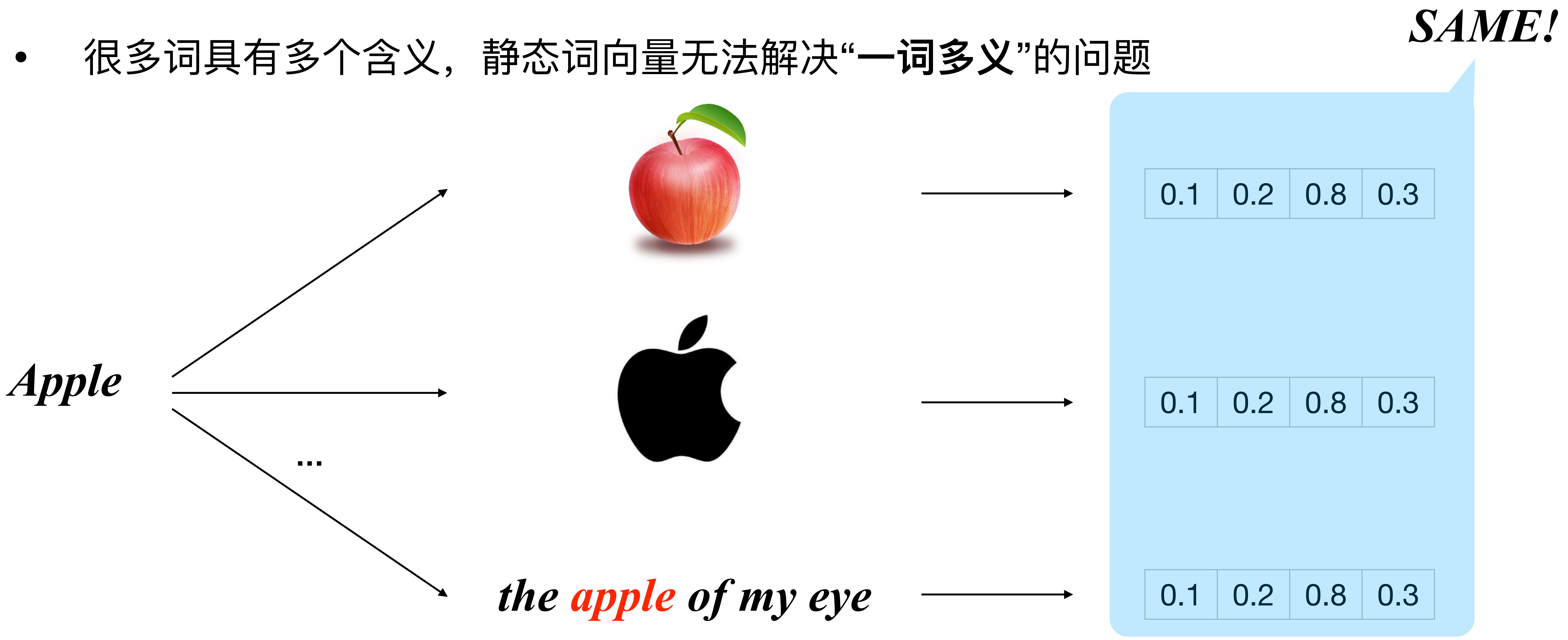
Colaboratory

基于上下文的语言模型

CONTEXTUALIZED LANGUAGE MODELS

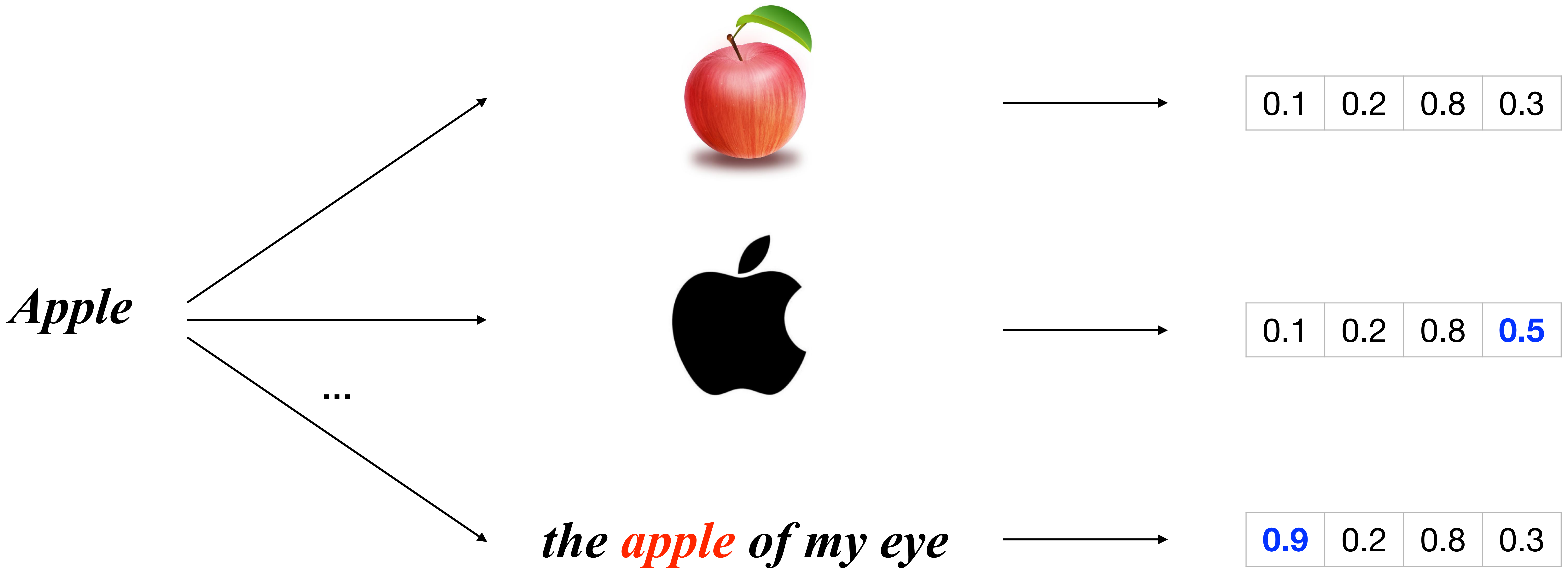
问题

- **静态词向量的问题**
  - 很多词具有多个含义，静态词向量无法解决“**一词多义**”的问题



问题

- **静态词向量的问题**
  - 词向量应根据其所处的**上下文的不同而发生改变**



0.1	0.2	0.8	0.3

0.1	0.2	0.8	**0.5**

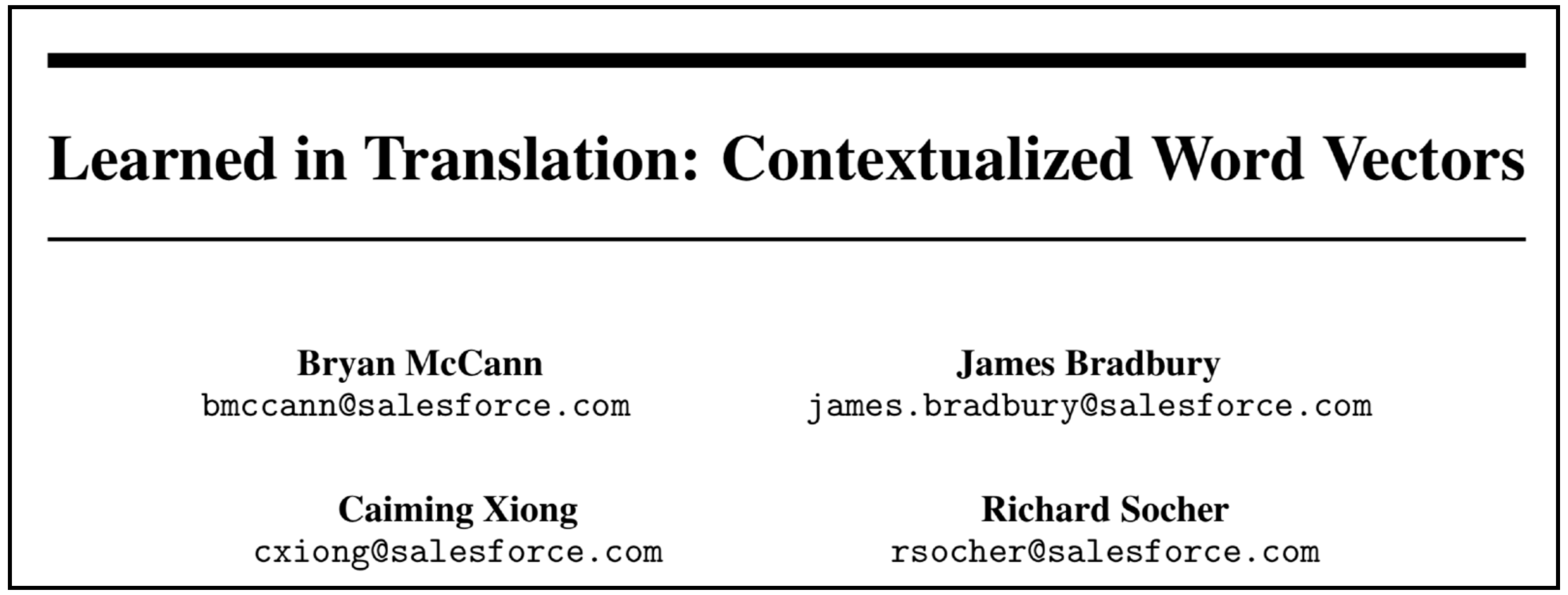
0.9	0.2	0.8	0.3

1 CoVe

2 ELMo

CoVe

- **CoVe: Co**ntextualized Word **Ve**ctors
  - 首次提出使用上下文相关的文本表示，即每个token的向量表示不唯一
  - 主要思想：将神经机器翻译（NMT）的表示迁移到通用NLP任务上



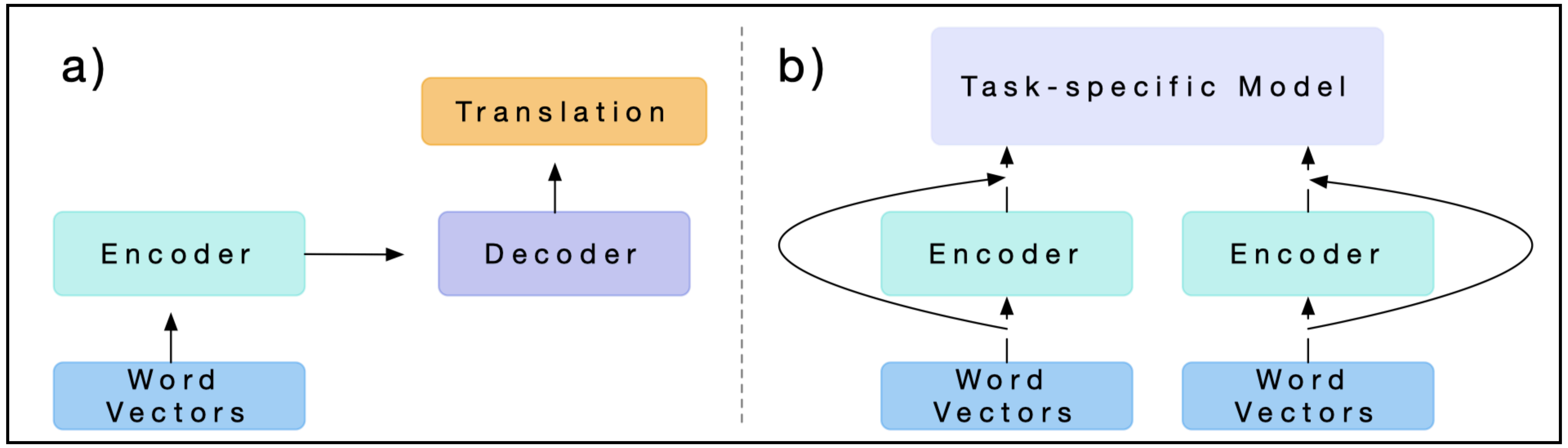
Learned in Translation: Contextualized Word Vectors

Bryan McCann
bmccann@salesforce.com

James Bradbury
james.bradbury@salesforce.com

Caiming Xiong
cxiong@salesforce.com

Richard Socher
rsocher@salesforce.com



***McCann et al., NeurIPS 2017.** Learned in Translation: Contextualized Word Vectors*

CoVe

- **训练阶段：训练一个基于神经网络的机器翻译模型（NMT）**
  - 给定一个源语言句子 w^x 和目标语言句子 w^z

两层 *bi-LSTM*

$$h = \text{MT-LSTM}(\text{GloVe}(w^x))$$

  - 基于注意力的解码器

$$\alpha_t = \text{softmax}\left(H(W_1 h_t^{\text{dec}} + b_1)\right)$$

$$\tilde{h}_t = \left[\tanh\left(W_2 H^\top \alpha_t + b_2; h_t^{\text{dec}}\right)\right]$$

$$h_t^{\text{dec}} = \text{LSTM}\left([z_{t-1}; \tilde{h}_{t-1}], h_{t-1}^{\text{dec}}\right)$$

  - 输出层

$$p(\hat{w}_t^z | X, w_1^z, \ldots, w_{t-1}^z) = \text{softmax}\left(W_{\text{out}} \tilde{h}_t + b_{\text{out}}\right)$$

McCann et al., NeurIPS 2017. *Learned in Translation: Contextualized Word Vectors*

CoVe

- **推理阶段**
  - 对于给定的一个源语言句子 w

$$\text{CoVe}(w) = \text{MT-LSTM}(\text{GloVe}(w))$$

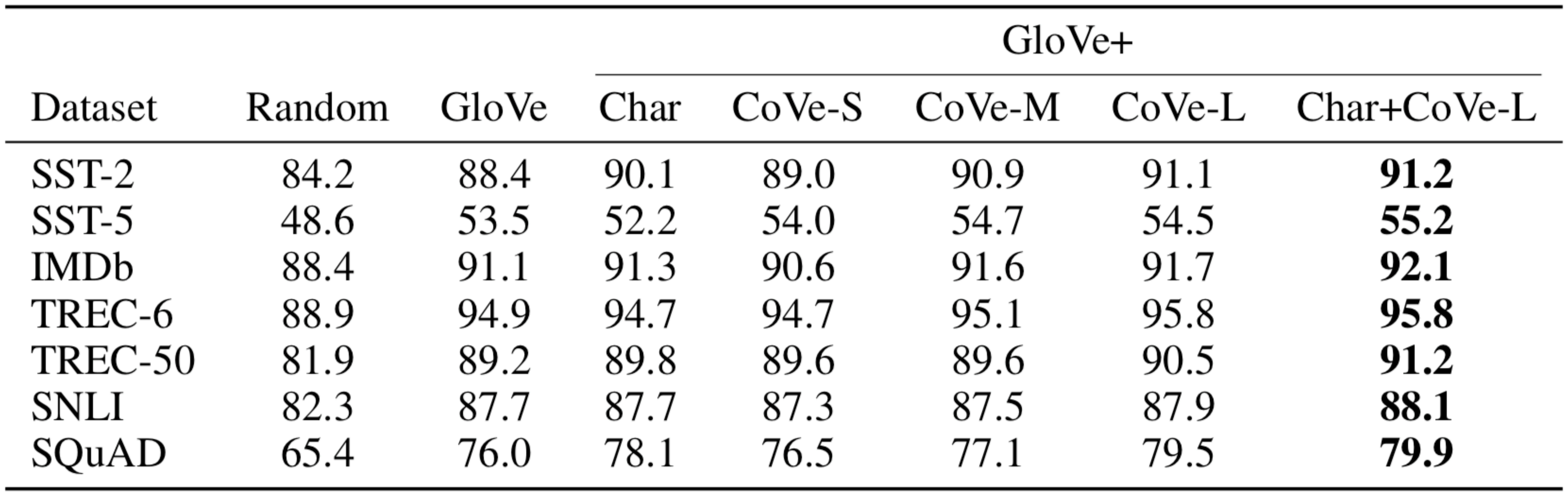
  - 如何在下游任务中使用CoVe?
    - 直接拼接GloVe和CoVe向量

$$\tilde{w} = [\text{GloVe}(w); \text{CoVe}(w)]$$

McCann et al., NeurIPS 2017. *Learned in Translation: Contextualized Word Vectors*

CoVe

- **实验结果**
  - 训练数据: En-De 30K (small), 209K (medium), 7M (large)
  - 相比传统词向量模型**提升有限**，适合作为词向量的一种信息补充



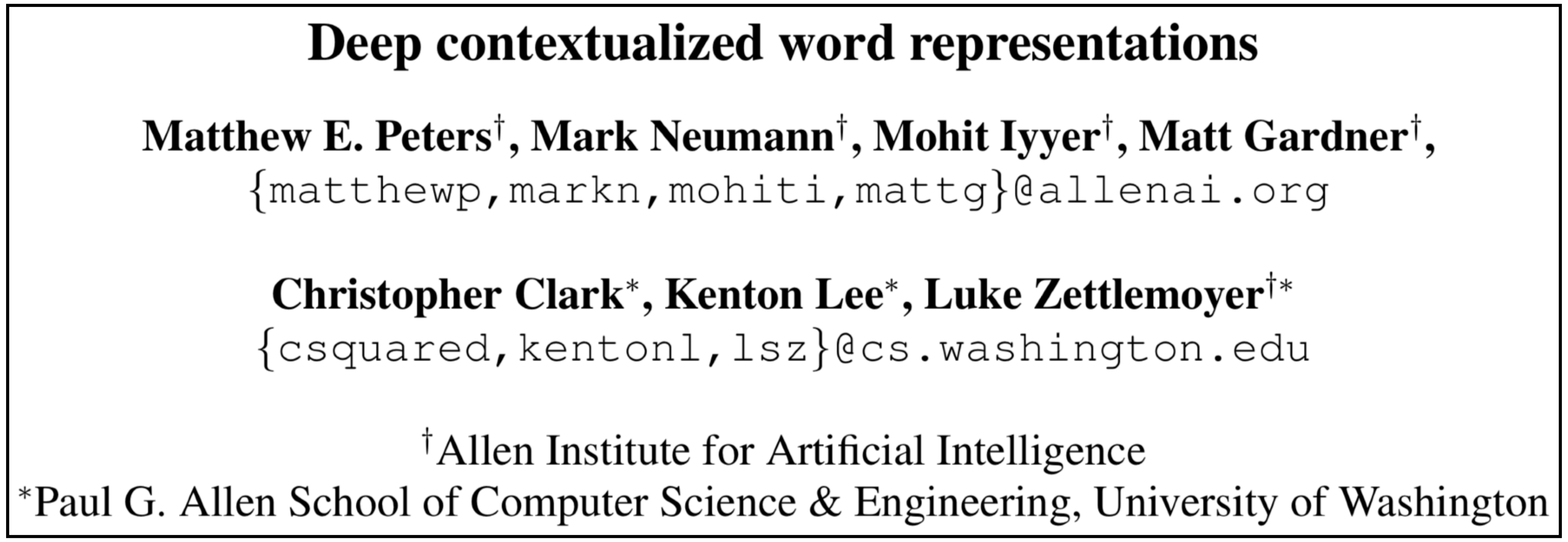
			GloVe+				
Dataset	Random	GloVe	Char	CoVe-S	CoVe-M	CoVe-L	Char+CoVe-L
SST-2	84.2	88.4	90.1	89.0	90.9	91.1	**91.2**
SST-5	48.6	53.5	52.2	54.0	54.7	54.5	**55.2**
IMDb	88.4	91.1	91.3	90.6	91.6	91.7	**92.1**
TREC-6	88.9	94.9	94.7	94.7	95.1	95.8	**95.8**
TREC-50	81.9	89.2	89.8	89.6	89.6	90.5	**91.2**
SNLI	82.3	87.7	87.7	87.3	87.5	87.9	**88.1**
SQuAD	65.4	76.0	78.1	76.5	77.1	79.5	**79.9**

*S=Small, M=Medium, L=Large

McCann et al., NeurIPS 2017. *Learned in Translation: Contextualized Word Vectors*

ELMo

- **ELMo: E**mbeddings from **L**anguage **Mo**dels (NAACL 2018 Best Paper)
  - 使用大规模语料训练一个深层双向的语言模型
  - ELMo可以很方便地添加在现有模型中
  - 特点：上下文相关、鲁棒性较强



Deep contextualized word representations

Matthew E. Peters[†], **Mark Neumann**[†], **Mohit Iyyer**[†], **Matt Gardner**[†],
{matthewp,markn,mohiti,mattg}@allenai.org

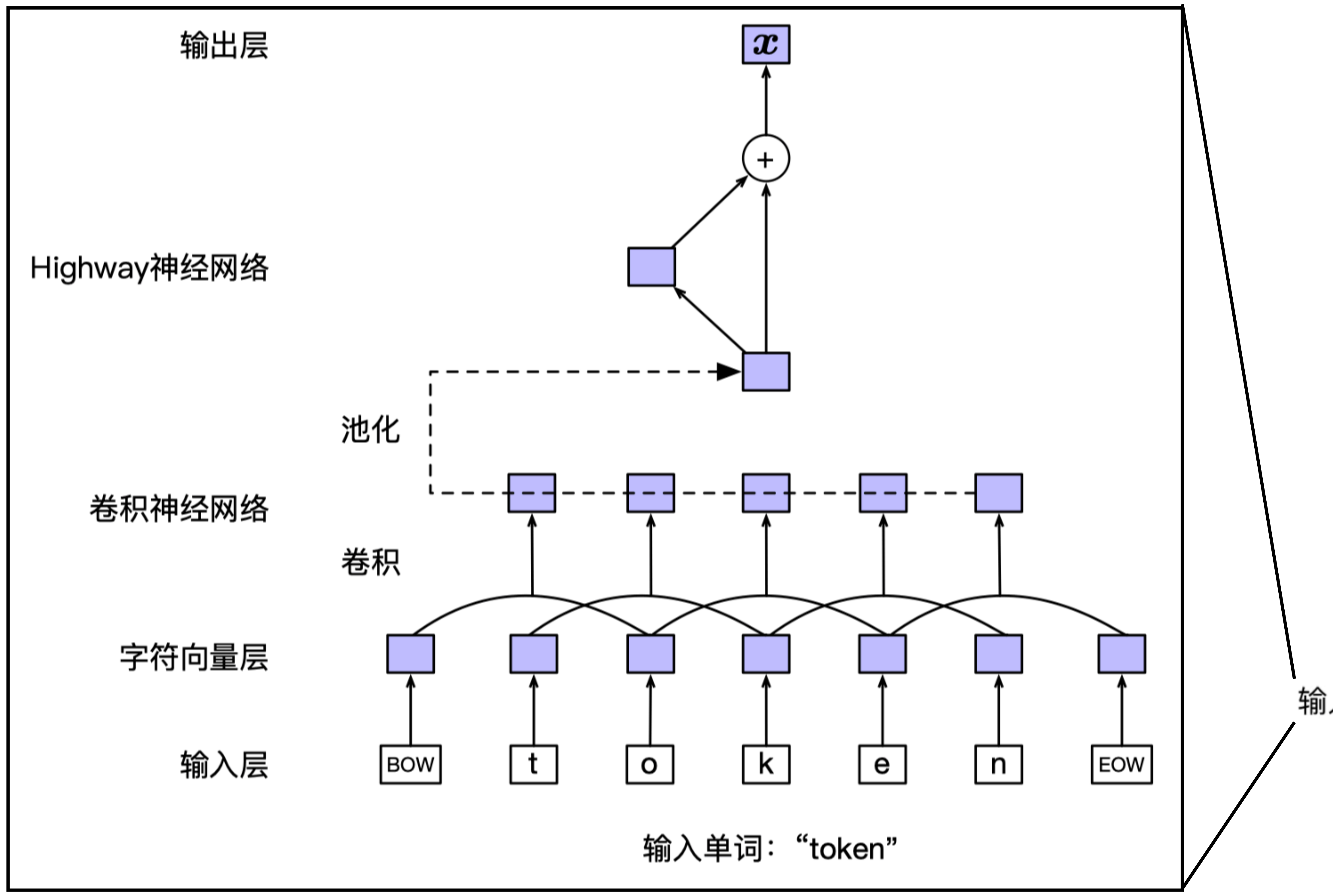
Christopher Clark[*], **Kenton Lee**[*], **Luke Zettlemoyer**[†*]
{csquared,kentonl,lsz}@cs.washington.edu

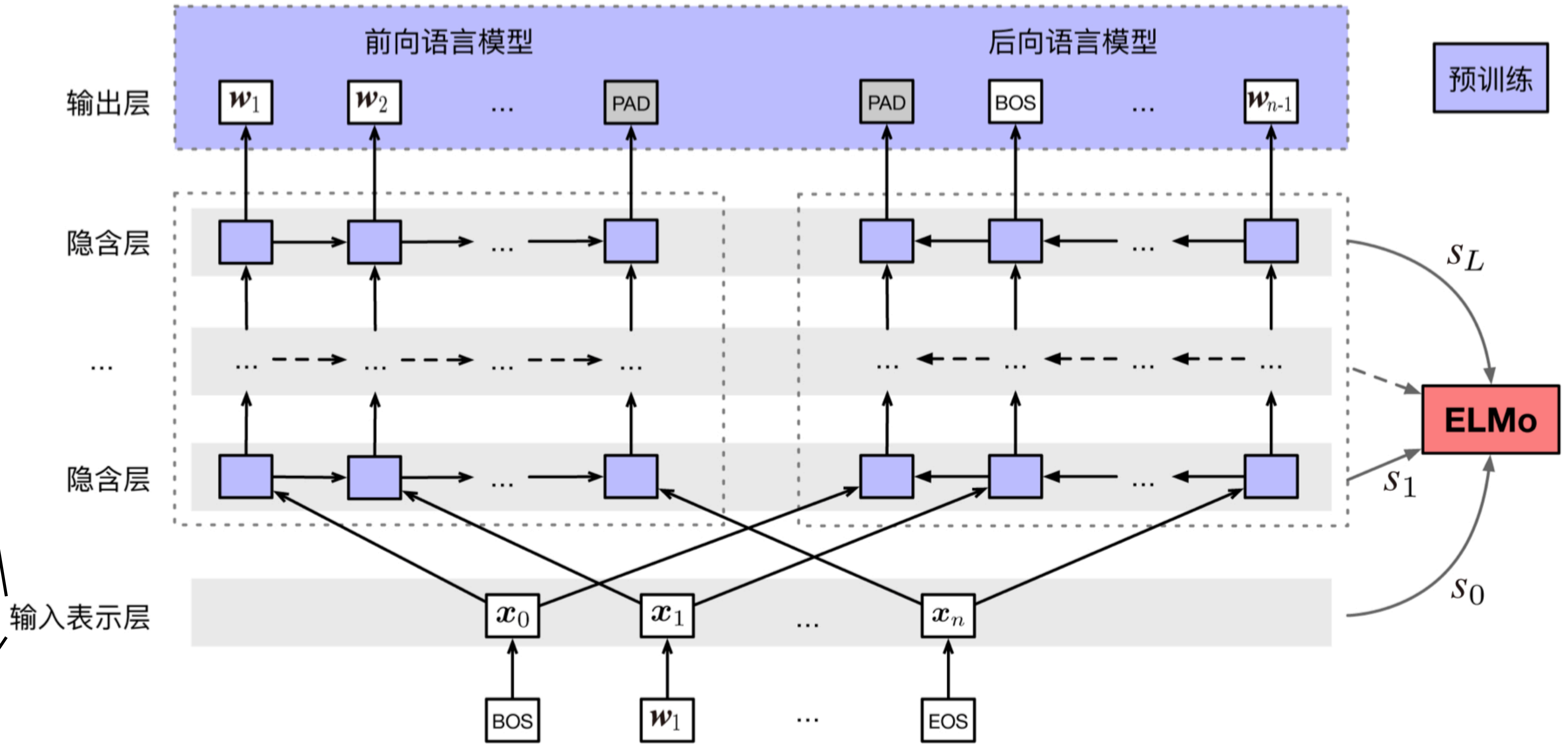
[†]Allen Institute for Artificial Intelligence
[*]Paul G. Allen School of Computer Science & Engineering, University of Washington

Peters et al., NAACL 2018. *Deep contextualized word representations*

ELMo

- **ELMo的整体结构**





Peters et al., NAACL 2018. *Deep contextualized word representations*

ELMo

- **训练阶段：双向语言模型 (BiLM)**
  - 给定一个包含 N 个token的序列 $(t_1, t_2, \ldots, t_N)$
  - 前向语言模型

$$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k \mid t_1, t_2, \ldots, t_{k-1}).$$

  - 后向语言模型

$$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k \mid t_{k+1}, t_{k+2}, \ldots, t_N).$$

  - 双向语言模型

$$\sum_{k=1}^{N} \Big(\log p(t_k \mid t_1, \ldots, t_{k-1}; \Theta_x, \overrightarrow{\Theta}_{LSTM}, \Theta_s) + \log p(t_k \mid t_{k+1}, \ldots, t_N; \Theta_x, \overleftarrow{\Theta}_{LSTM}, \Theta_s) \Big).$$

Peters et al., NAACL 2018. *Deep contextualized word representations*

ELMo

- **推理阶段**
  - 对于每个token t_k, 一个L 层的BiLM可以得到 $2L+1$个表示（含词向量表示）

词向量

BiLM 输出

$$\begin{aligned} R_k &= \{\mathbf{x}_k^{LM}, \boxed{\overrightarrow{\mathbf{h}}_{k,j}^{LM}, \overleftarrow{\mathbf{h}}_{k,j}^{LM}} \mid j = 1, \ldots, L\} \\ &= \{\mathbf{h}_{k,j}^{LM} \mid j = 0, \ldots, L\}, \end{aligned}$$

  - 将 R 中的所有向量加权变换为一个向量

$$\mathbf{ELMo}_k^{task} = E(R_k; \Theta^{task}) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} \mathbf{h}_{k,j}^{LM}$$

Peters et al., NAACL 2018. *Deep contextualized word representations*

ELMo

- **推理阶段**
  - 将ELMo表示直接与其他类型的表示进行拼接

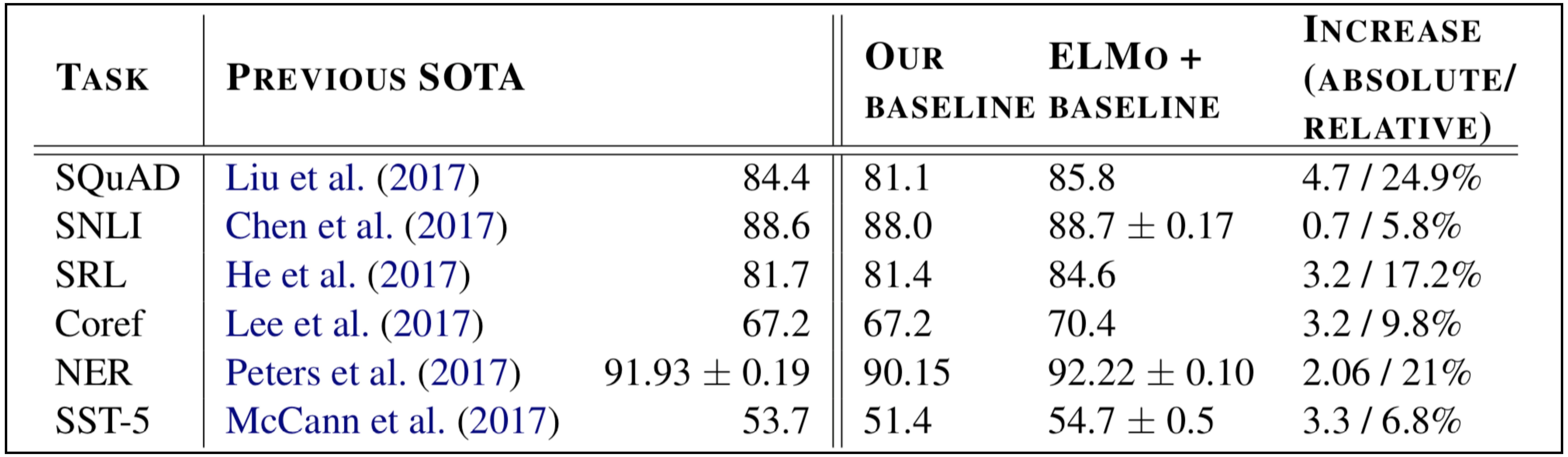
$$e_{final} = concat[e_{char}; e_{word}; e_{ELMo}]$$

- **使用技巧**
  - 对于某些任务来说（例如SQuAD），在RNN（此处指篇章与问题建模使用的RNN）输出后再次拼接ELMo表示可获得一定提升
  - 在ELMo输出上添加一定的dropout（默认的0.5是一个比较不错的选择）
  - **必要的时候可以让ELMo部分的权重参与下游任务训练**

Peters et al., NAACL 2018. *Deep contextualized word representations*

ELMo

- **实验结果**
  - 在多个NLP任务上获得显著性能提升
  - 如果ELMo也参与下游任务精调（权重可更新），并不是所有任务都会获得显著提升



TASK	PREVIOUS SOTA		OUR BASELINE	ELMO + BASELINE	INCREASE (ABSOLUTE/ RELATIVE)
SQuAD	Liu et al. (2017)	84.4	81.1	85.8	4.7 / 24.9%
SNLI	Chen et al. (2017)	88.6	88.0	88.7 ± 0.17	0.7 / 5.8%
SRL	He et al. (2017)	81.7	81.4	84.6	3.2 / 17.2%
Coref	Lee et al. (2017)	67.2	67.2	70.4	3.2 / 9.8%
NER	Peters et al. (2017)	91.93 ± 0.19	90.15	92.22 ± 0.10	2.06 / 21%
SST-5	McCann et al. (2017)	53.7	51.4	54.7 ± 0.5	3.3 / 6.8%

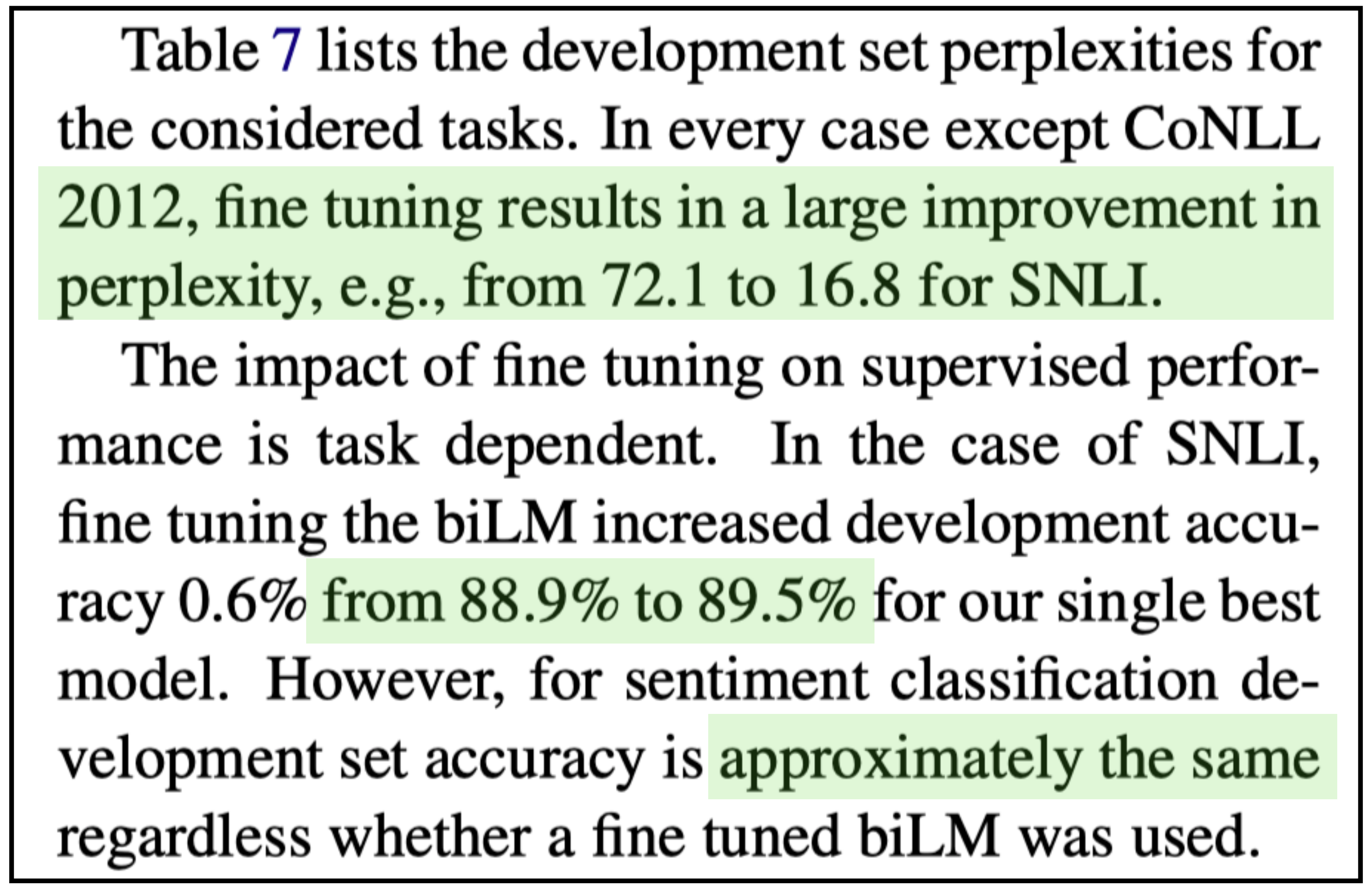


Table 7 lists the development set perplexities for the considered tasks. In every case except CoNLL 2012, fine tuning results in a large improvement in perplexity, e.g., from 72.1 to 16.8 for SNLI.

The impact of fine tuning on supervised performance is task dependent. In the case of SNLI, fine tuning the biLM increased development accuracy 0.6% from 88.9% to 89.5% for our single best model. However, for sentiment classification development set accuracy is approximately the same regardless whether a fine tuned biLM was used.

From Appendix A.1

Peters et al., NAACL 2018. *Deep contextualized word representations*

经典预训练语言模型

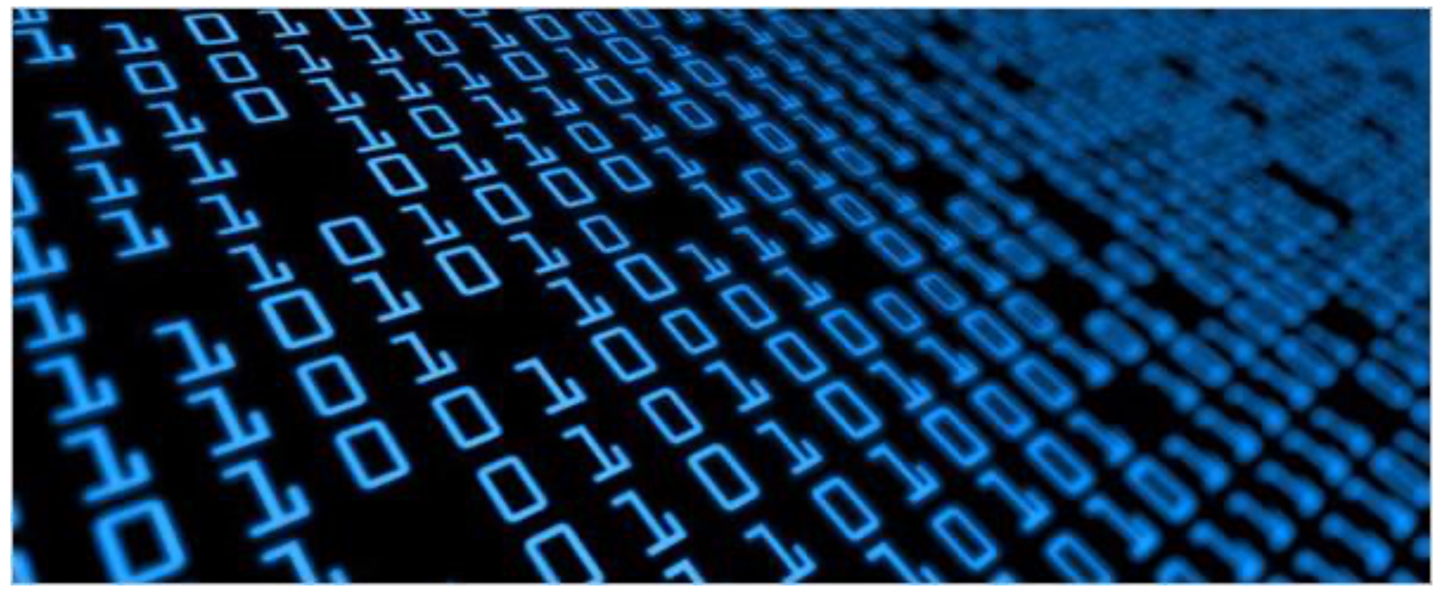
PRE-TRAINED LANGUAGE MODELS

问题

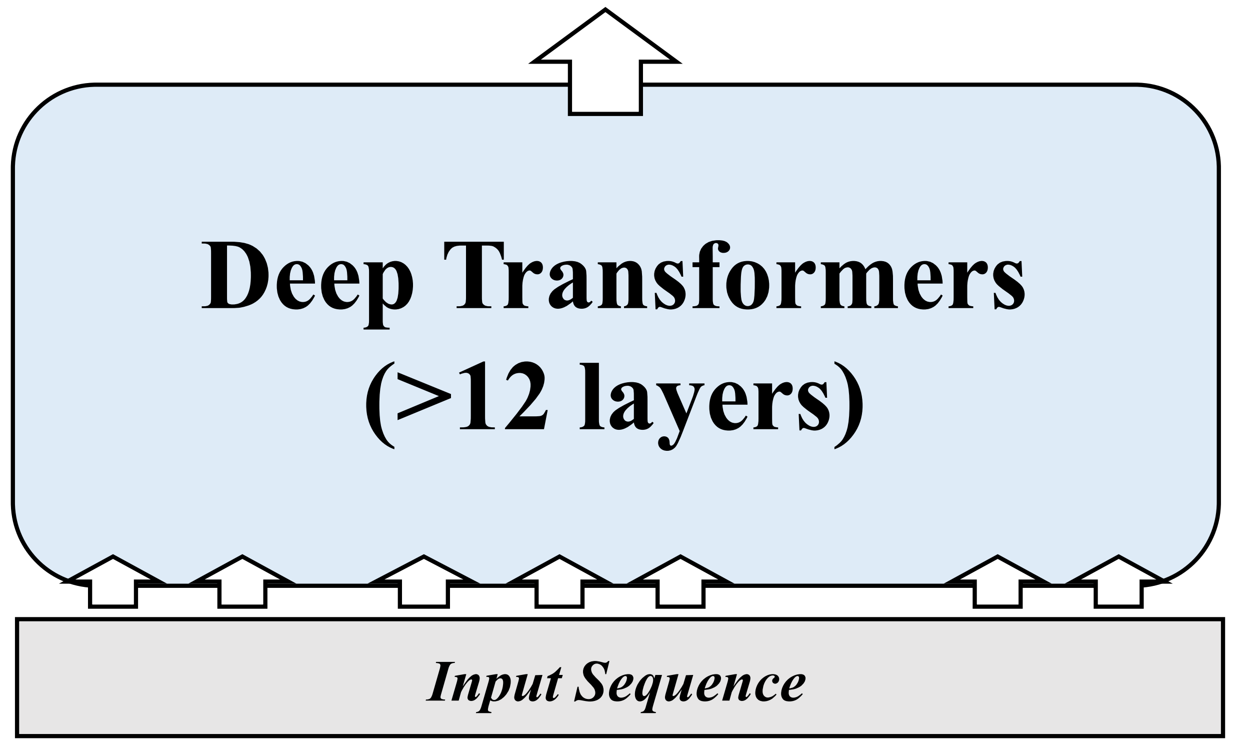
- **CoVe/ELMo模型的问题**
  - **数据**
    - 训练数据相对比较局限，例如CoVe要求使用双语平行句对
  - **模型**
    - 表示模型的参数量相对较小（相比预训练语言模型），模型深度不够
  - **用法**
    - 通常使用这类模型时，表示模型本身是不参与训练的（权重无更新）
    - 表示模型本身不参与训练，一定程度上限制了表示模型在下游任务上的泛化能力

PLMs

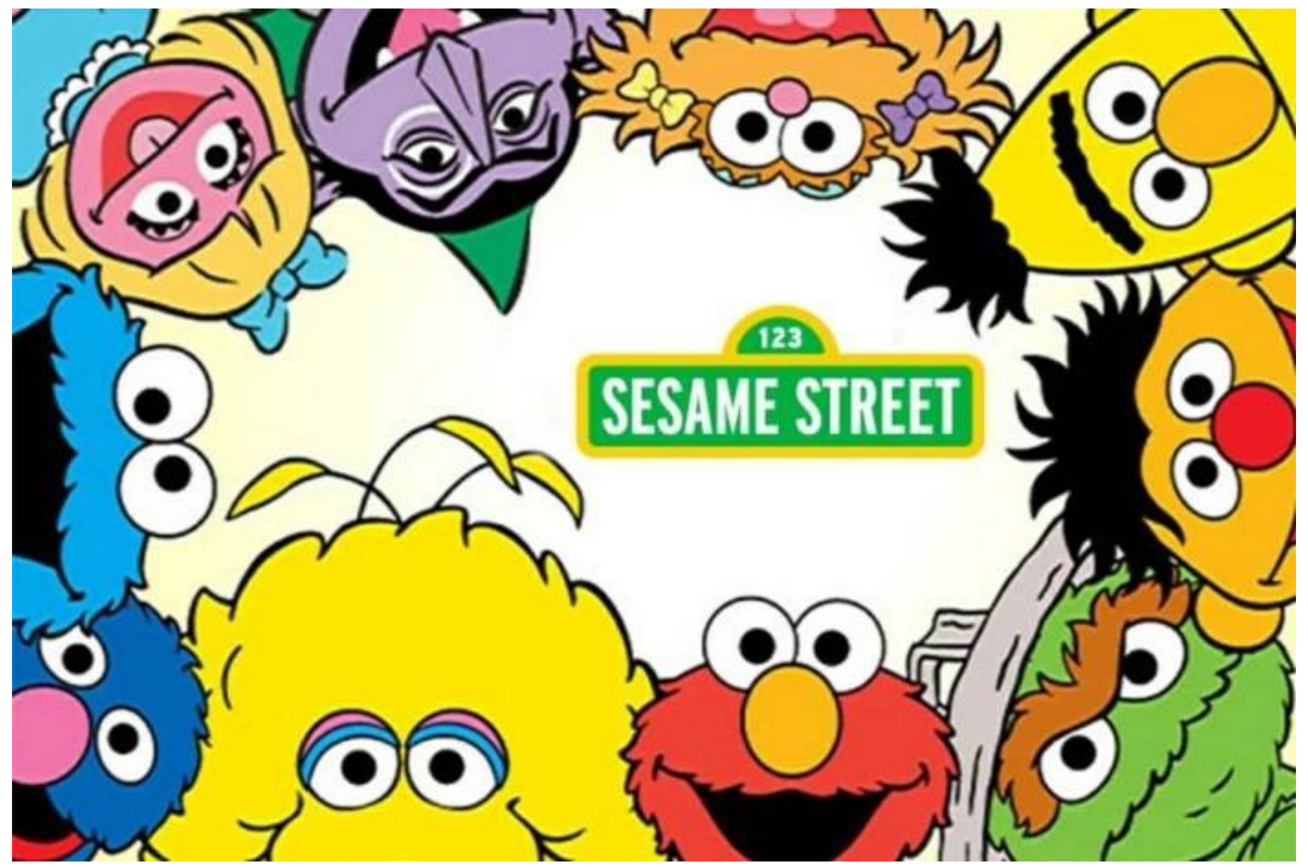
大数据
（无标注文本）



大模型
（深度神经网络）



大算力
（并行计算集群）



OpenAI

GPT

GPT-2

GPT-3

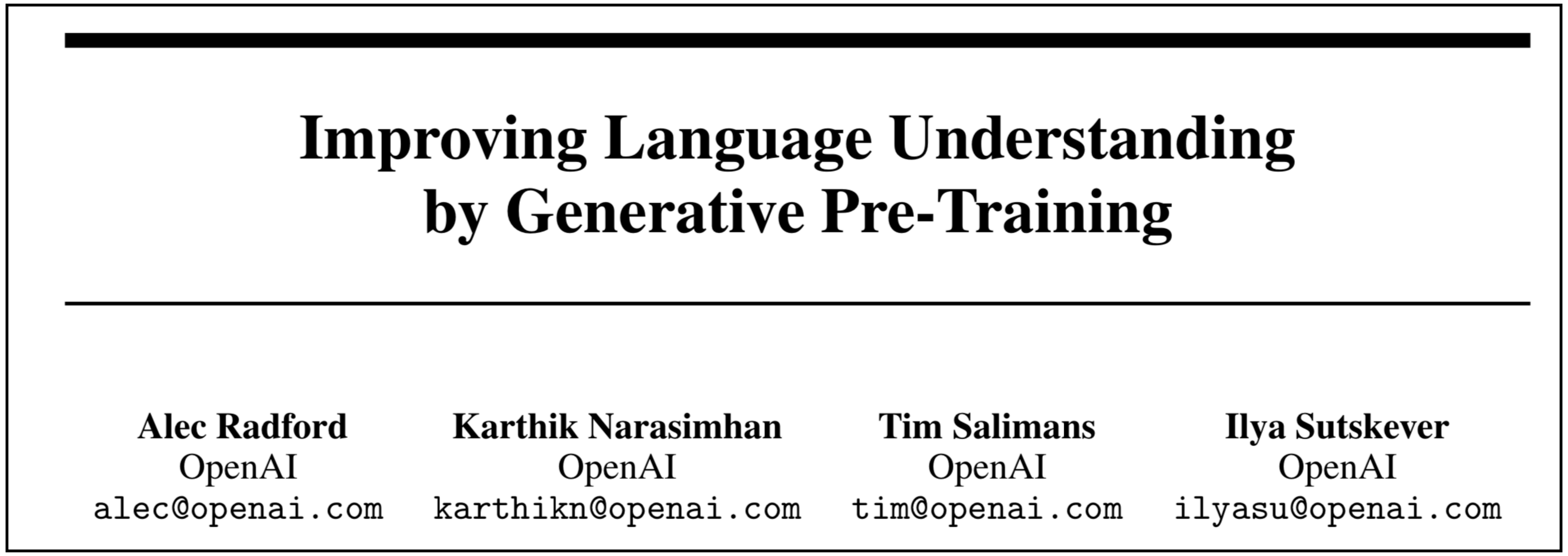
以及更多钱

1 GPT 系列

2 BERT 系列

GPT

- **GPT: G**enerative **P**re-**T**raining
  - OpenAI提出了“生成式预训练+判别式精调”框架
  - 正式开启了自然语言处理领域**“预训练+精调”**的新时代



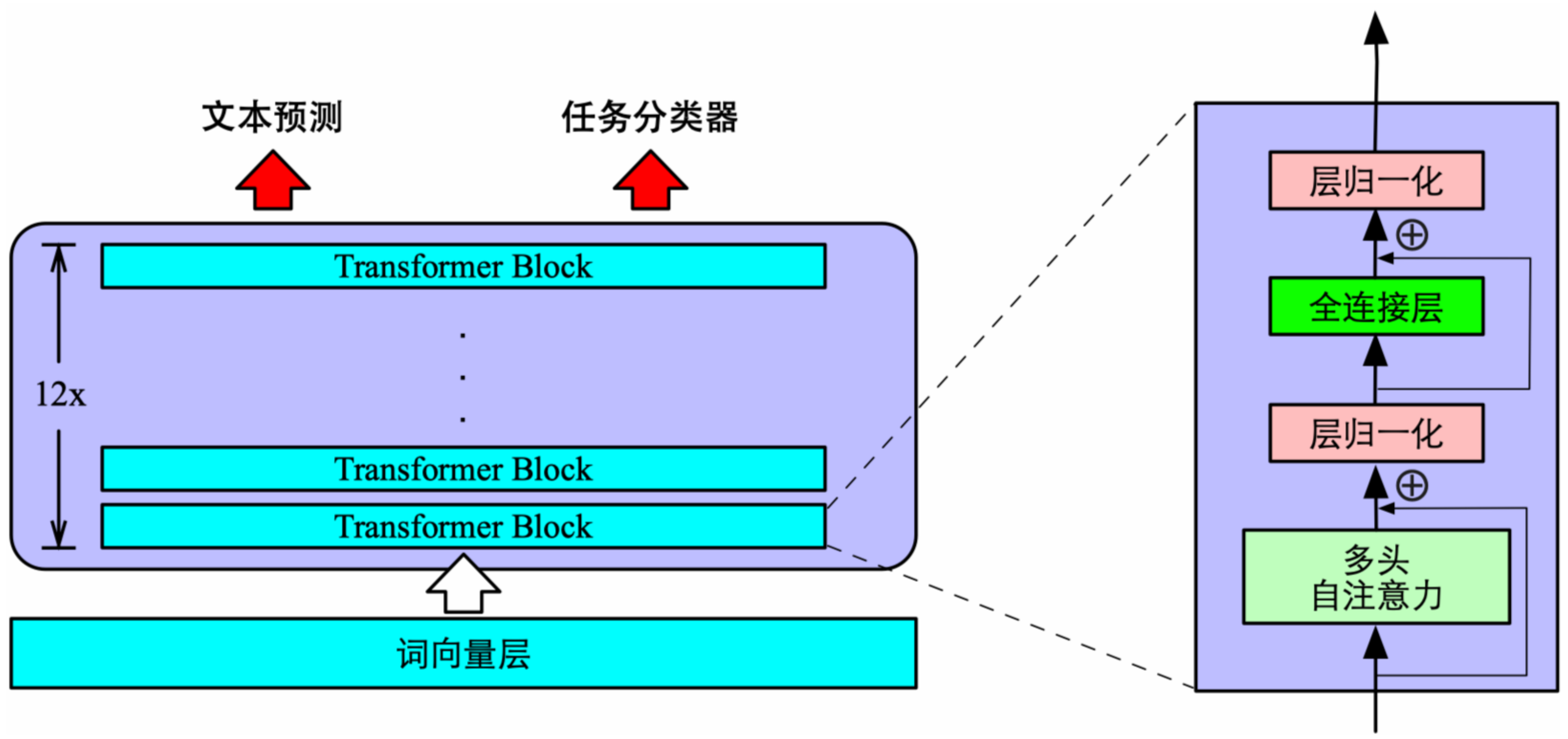
Improving Language Understanding by Generative Pre-Training

Alec Radford OpenAI alec@openai.com
Karthik Narasimhan OpenAI karthikn@openai.com
Tim Salimans OpenAI tim@openai.com
Ilya Sutskever OpenAI ilyasu@openai.com

Radford et al., 2018. *Improving Language Understanding by Generative Pre-Training*

GPT

- **模型结构**



Radford et al., 2018. Improving Language Understanding by Generative Pre-Training

GPT

- **训练阶段**
  - 利用自由文本学习一个基于Transformer的高容量**单向**神经语言模型
  - 即利用历史文本序列预测下一个单词是什么

$$h_0 = \overset{\text{上下文向量}}{U}\overset{\text{词向量矩阵}}{W_e} + W_p \longrightarrow \text{位置向量矩阵}$$

$$h_l = \texttt{transformer\_block}(h_{l-1}) \forall i \in [1, n]$$

$$P(u) = \texttt{softmax}(h_n W_e^T)$$

$$L_1(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \ldots, u_{i-1}; \Theta)$$

Radford et al., 2018. *Improving Language Understanding by Generative Pre-Training*

GPT

- **推理阶段**
  - 利用下游任务的有标注数据，对GPT模型进行**精调**
  - 对于给定的有标注数据集 C, 输入文本 $x^1, \ldots, x^m$, 标签 y

$$P(y|x^1, \ldots, x^m) = \texttt{softmax}(h_l^m W_y).$$

***Transformer* 隐层输出**

$$L_2(\mathcal{C}) = \sum_{(x,y)} \log P(y|x^1, \ldots, x^m).$$

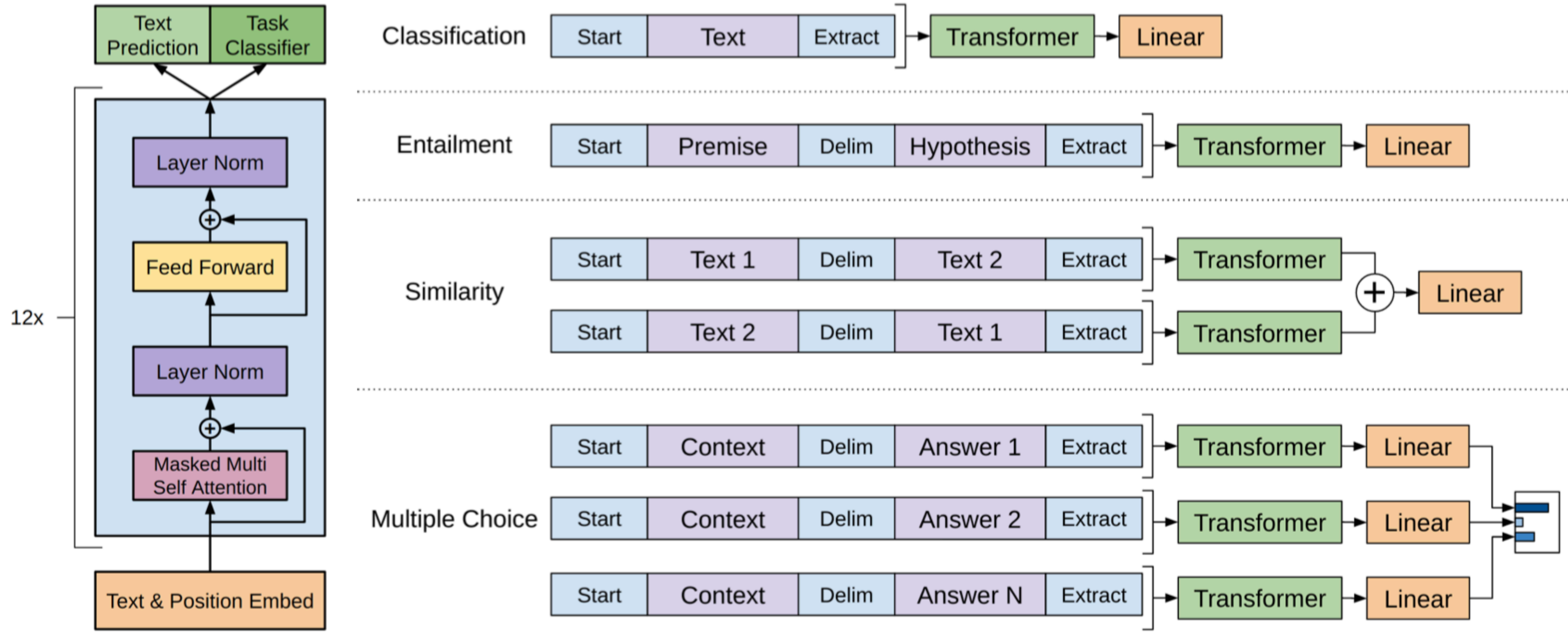
  - 某些情况下，添加额外的预训练损失可以进一步提升性能

$$L_3(\mathcal{C}) = L_2(\mathcal{C}) + \lambda * L_1(\mathcal{C})$$

Radford et al., 2018. *Improving Language Understanding by Generative Pre-Training*

GPT

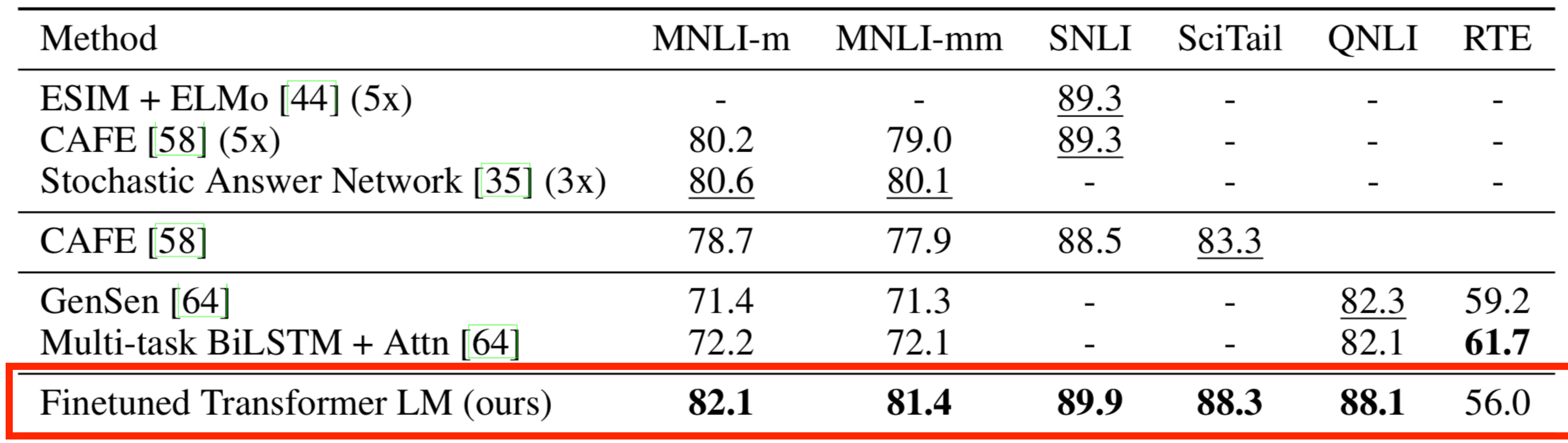
- 利用GPT精调各类下游任务



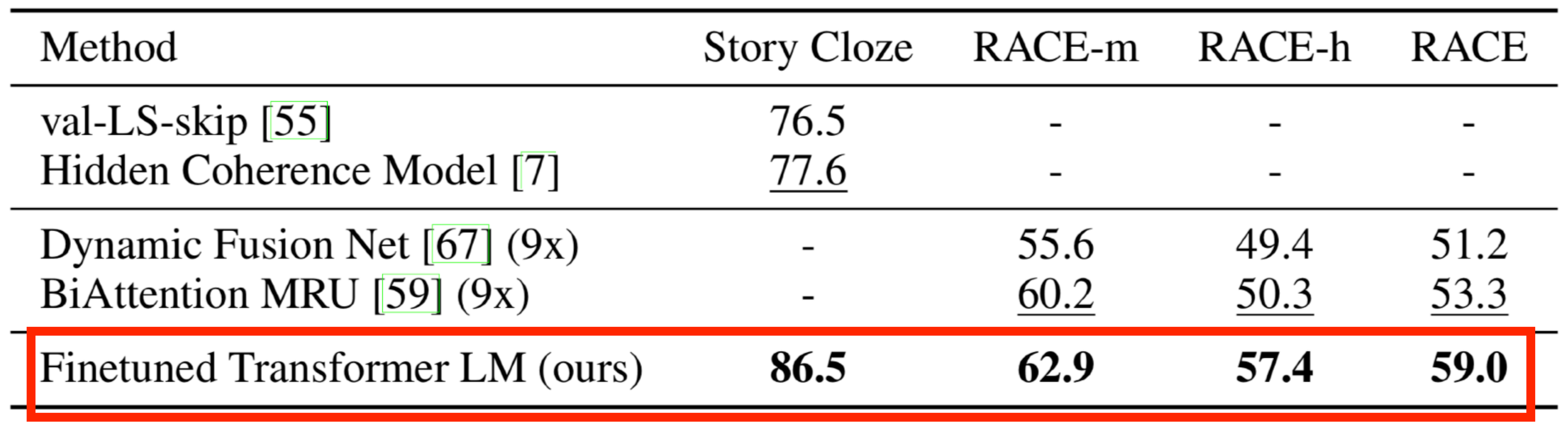
Radford et al., 2018. *Improving Language Understanding by Generative Pre-Training*

GPT

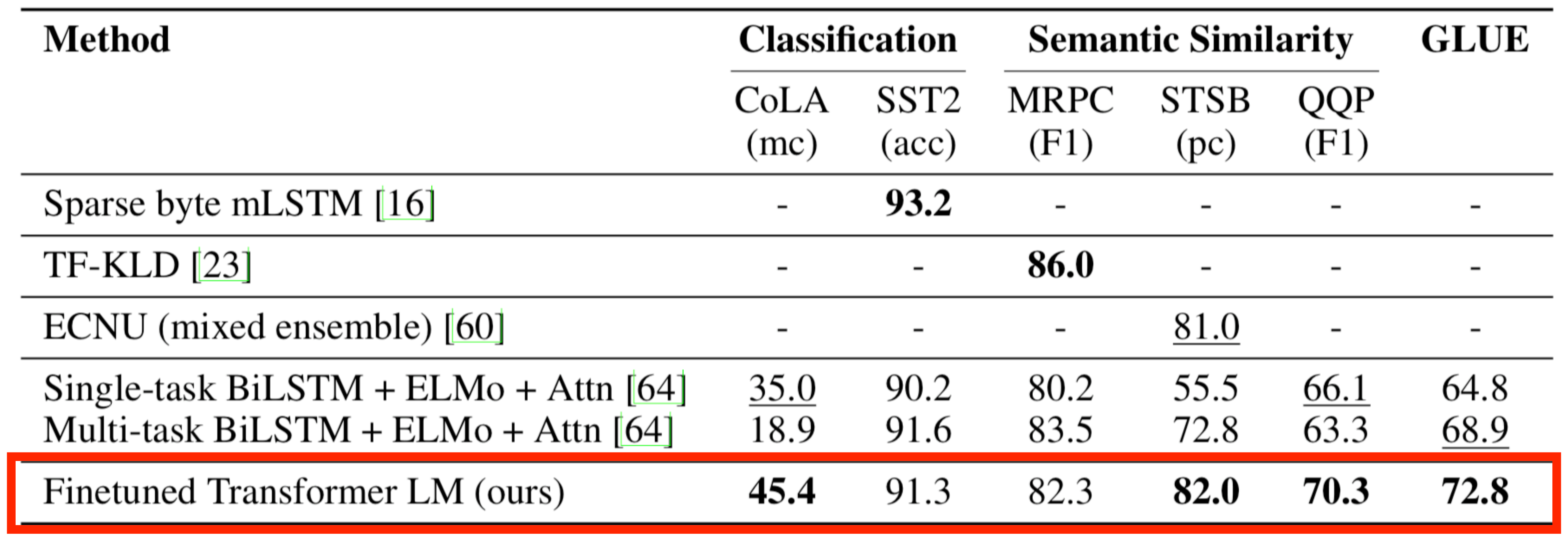
- **实验结果**



Method	MNLI-m	MNLI-mm	SNLI	SciTail	QNLI	RTE
ESIM + ELMo [44] (5x)	-	-	89.3	-	-	-
CAFE [58] (5x)	80.2	79.0	89.3	-	-	-
Stochastic Answer Network [35] (3x)	80.6	80.1	-	-	-	-
CAFE [58]	78.7	77.9	88.5	83.3		
GenSen [64]	71.4	71.3	-	-	82.3	59.2
Multi-task BiLSTM + Attn [64]	72.2	72.1	-	-	82.1	**61.7**
Finetuned Transformer LM (ours)	**82.1**	**81.4**	**89.9**	**88.3**	**88.1**	56.0



Method	Story Cloze	RACE-m	RACE-h	RACE
val-LS-skip [55]	76.5	-	-	-
Hidden Coherence Model [7]	77.6	-	-	-
Dynamic Fusion Net [67] (9x)	-	55.6	49.4	51.2
BiAttention MRU [59] (9x)	-	60.2	50.3	53.3
Finetuned Transformer LM (ours)	**86.5**	**62.9**	**57.4**	**59.0**

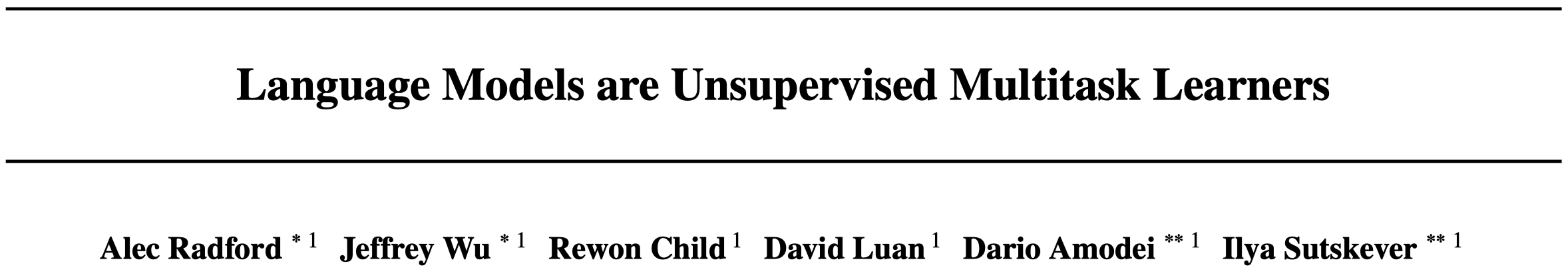


Method	**Classification**		**Semantic Similarity**			**GLUE**
	CoLA (mc)	SST2 (acc)	MRPC (F1)	STSB (pc)	QQP (F1)	
Sparse byte mLSTM [16]	-	**93.2**	-	-	-	-
TF-KLD [23]	-	-	**86.0**	-	-	-
ECNU (mixed ensemble) [60]	-	-	-	81.0	-	-
Single-task BiLSTM + ELMo + Attn [64]	35.0	90.2	80.2	55.5	66.1	64.8
Multi-task BiLSTM + ELMo + Attn [64]	18.9	91.6	83.5	72.8	63.3	68.9
Finetuned Transformer LM (ours)	**45.4**	91.3	82.3	**82.0**	**70.3**	**72.8**

Radford et al., 2018. *Improving Language Understanding by Generative Pre-Training*

GPT-2

- **GPT-2: Language Models are Unsupervised Multitask Learners**
  - 提出语言模型可以在zero-shot设置下完成一定的下游任务
  - 语言模型的容量是实现zero-shot任务迁移的关键所在



Language Models are Unsupervised Multitask Learners

Alec Radford [* 1] **Jeffrey Wu** [* 1] **Rewon Child** [1] **David Luan** [1] **Dario Amodei** [** 1] **Ilya Sutskever** [** 1]

Radford et al., 2019. *Language Models are Unsupervised Multitask Learners*

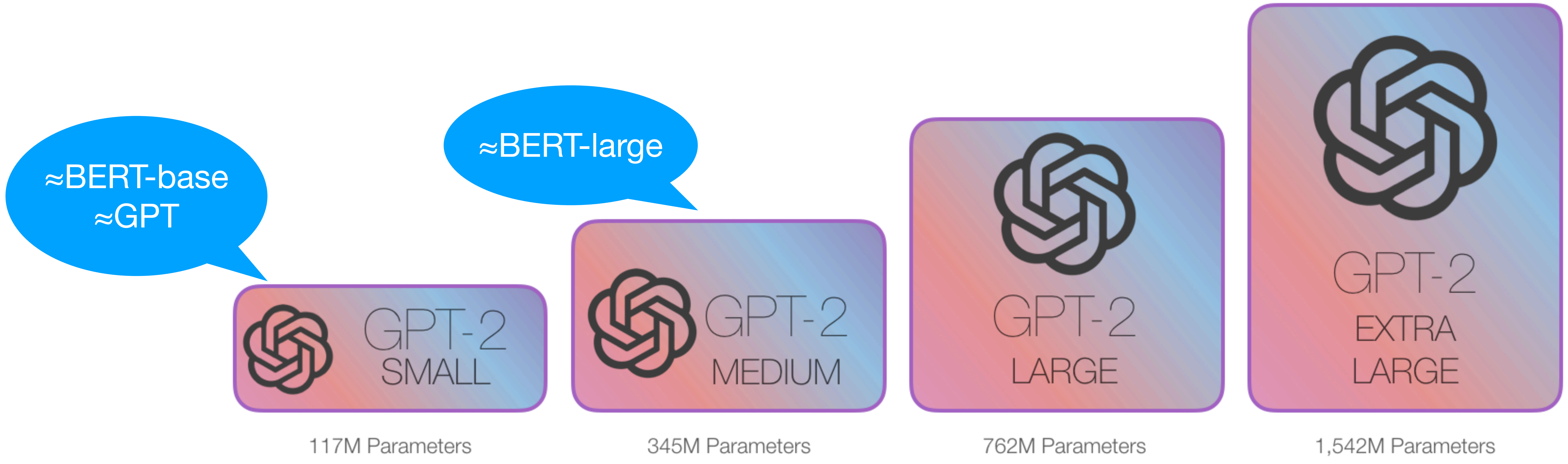
GPT-2

- **训练阶段**
  - 模型结构方面与GPT并无明显差别
  - 预训练数据：6GB → 40GB （未压缩自由文本）
  - 其他改进
    - 层归一化（Layer normalization）被放在每个block的输入端
    - 在最终自注意力block后添加了额外的层归一化
    - 词表大小扩展至 50,257 （GPT: 40,000）
    - 上下文长度扩展至 1024 （GPT: 512）

Radford et al., 2019. *Language Models are Unsupervised Multitask Learners*

GPT-2

- 模型大小



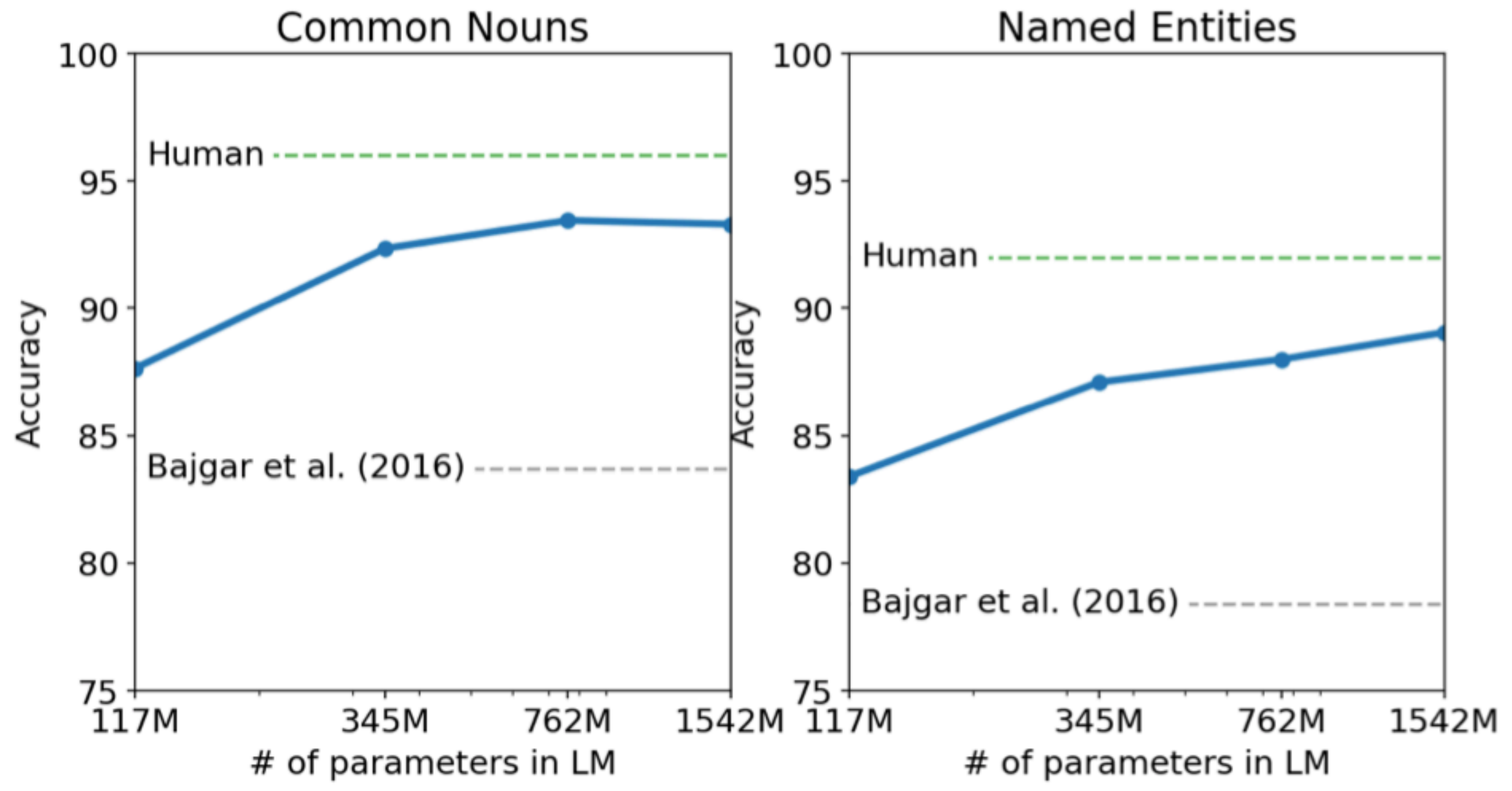
Radford et al., 2019. *Language Models are Unsupervised Multitask Learners*

GPT-2

- **实验结果**

	LAMBADA (PPL)	LAMBADA (ACC)	CBT-CN (ACC)	CBT-NE (ACC)	WikiText2 (PPL)	PTB (PPL)	enwik8 (BPB)	text8 (BPC)	WikiText103 (PPL)	1BW (PPL)
SOTA	99.8	59.23	85.7	82.3	39.14	46.54	0.99	1.08	18.3	**21.8**
117M	**35.13**	45.99	**87.65**	**83.4**	**29.41**	65.85	1.16	1.17	37.50	75.20
345M	**15.60**	55.48	**92.35**	**87.1**	**22.76**	47.33	1.01	**1.06**	26.37	55.72
762M	**10.87**	**60.12**	**93.45**	**88.0**	**19.93**	**40.31**	**0.97**	**1.02**	22.05	44.575
1542M	**8.63**	**63.24**	**93.30**	**89.05**	**18.34**	**35.76**	**0.93**	**0.98**	**17.48**	42.16

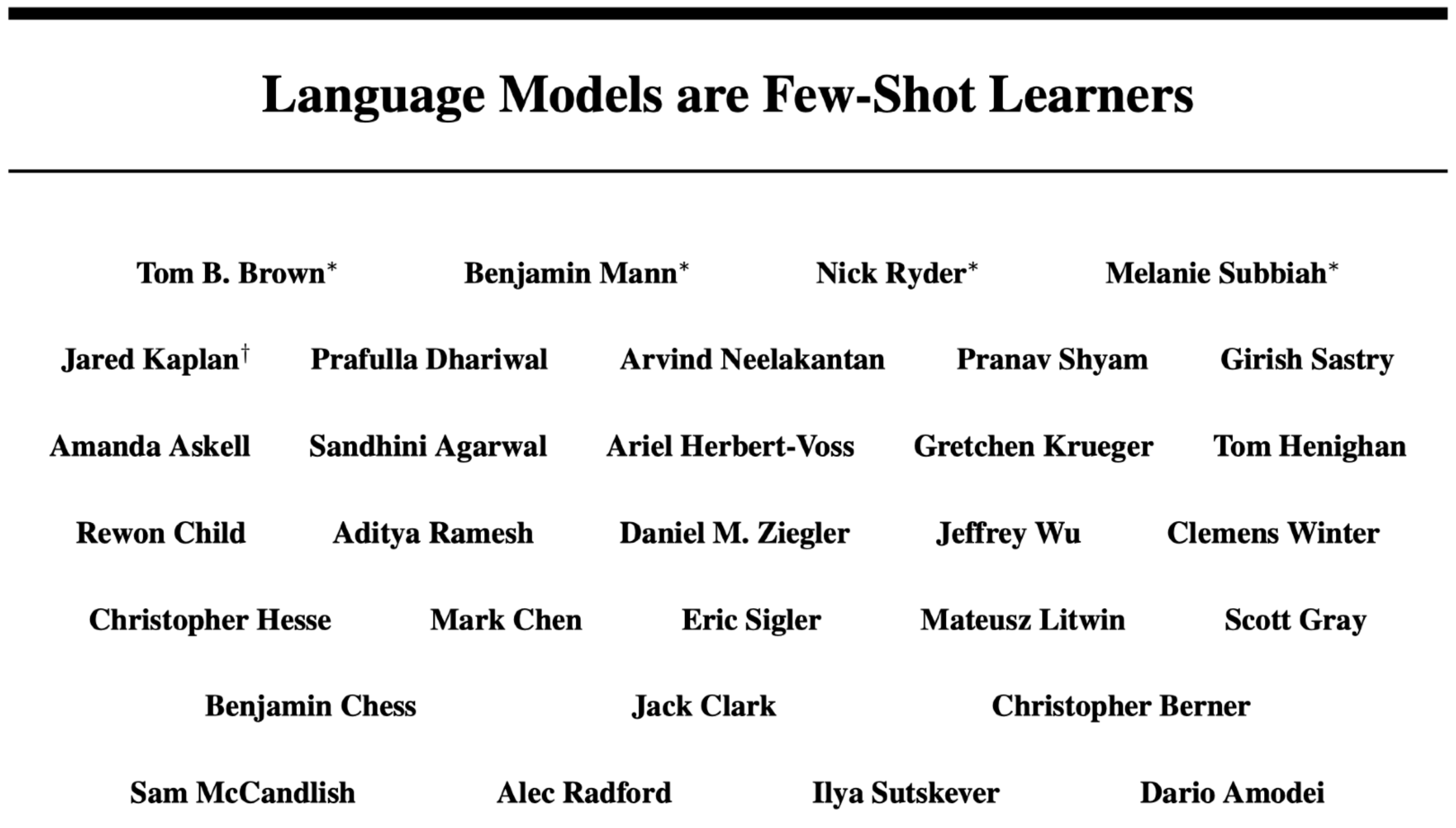
Cloze-style
Reading Comprehension



Radford et al., 2019. *Language Models are Unsupervised Multitask Learners*

GPT-3

- **GPT-3: Language Models are Few-Shot Learners**
  - 主要展示了超大规模语言模型在小样本学习（few-shot learning）上的能力
  - 模型参数量进一步扩展至175B，预示着预训练模型进入到**超大规模时代**



Language Models are Few-Shot Learners

Tom B. Brown* Benjamin Mann* Nick Ryder* Melanie Subbiah*

Jared Kaplan† Prafulla Dhariwal Arvind Neelakantan Pranav Shyam Girish Sastry

Amanda Askell Sandhini Agarwal Ariel Herbert-Voss Gretchen Krueger Tom Henighan

Rewon Child Aditya Ramesh Daniel M. Ziegler Jeffrey Wu Clemens Winter

Christopher Hesse Mark Chen Eric Sigler Mateusz Litwin Scott Gray

Benjamin Chess Jack Clark Christopher Berner

Sam McCandlish Alec Radford Ilya Sutskever Dario Amodei

Brown et al., 2020. *Language Models are Few-Shot Learners*

GPT-3

- **模型**
  - 与GPT-2并无明显差别，主要添加了稀疏注意力等技术以进一步降低超大模型的训练量

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}

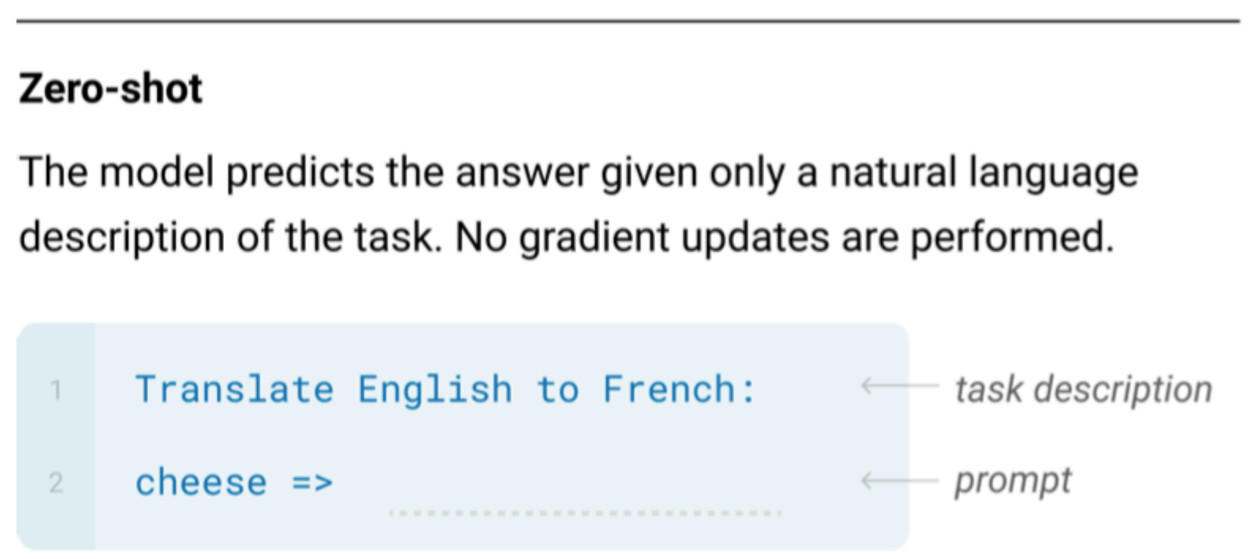
GPT-2 最大版本参数量是**1.5B**，需要占用**6G**磁盘空间

>700G 磁盘空间

Brown et al., 2020. *Language Models are Few-Shot Learners*

GPT-3

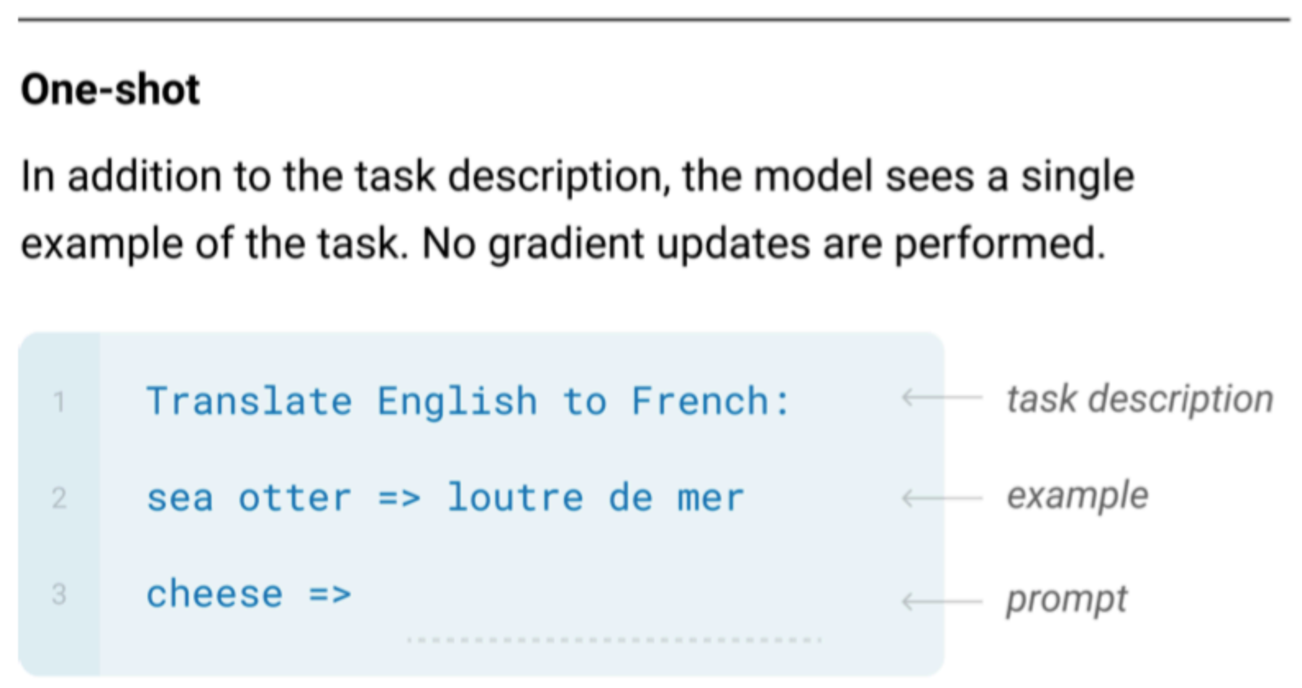
- **应用方式**
  - 传统预训练模型：直接在下游任务数据上精调
  - GPT-3类超大规模模型: zero-shot, one-shot, few-shot



Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

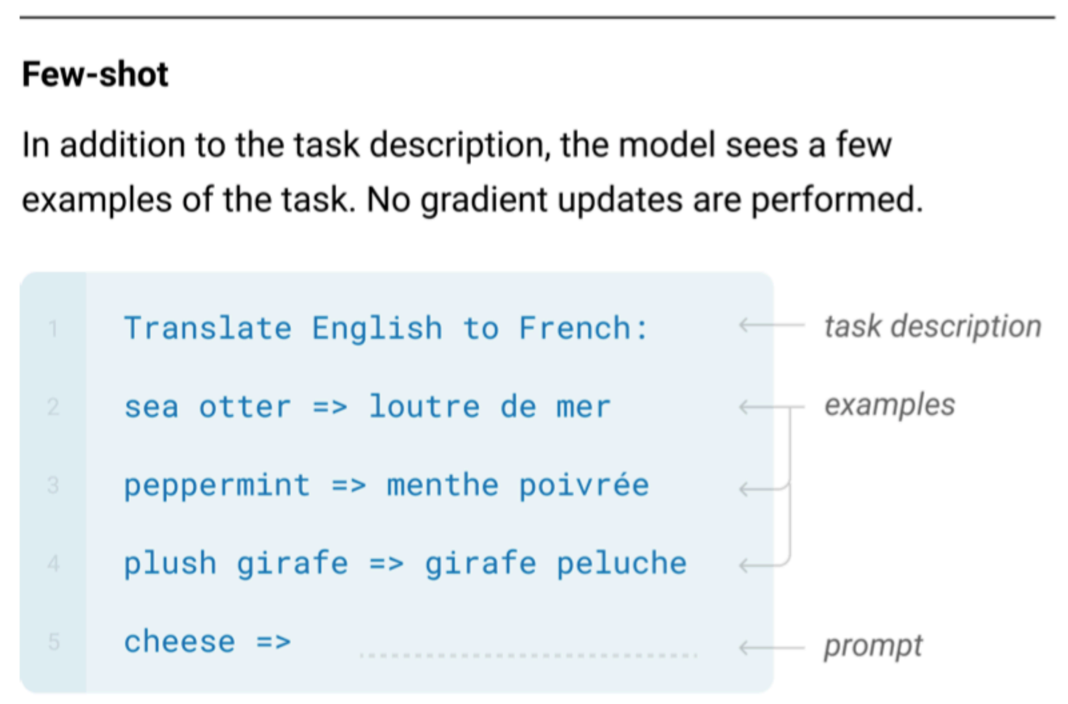
```
Translate English to French:     ← task description
cheese =>                        ← prompt
```



One-shot

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

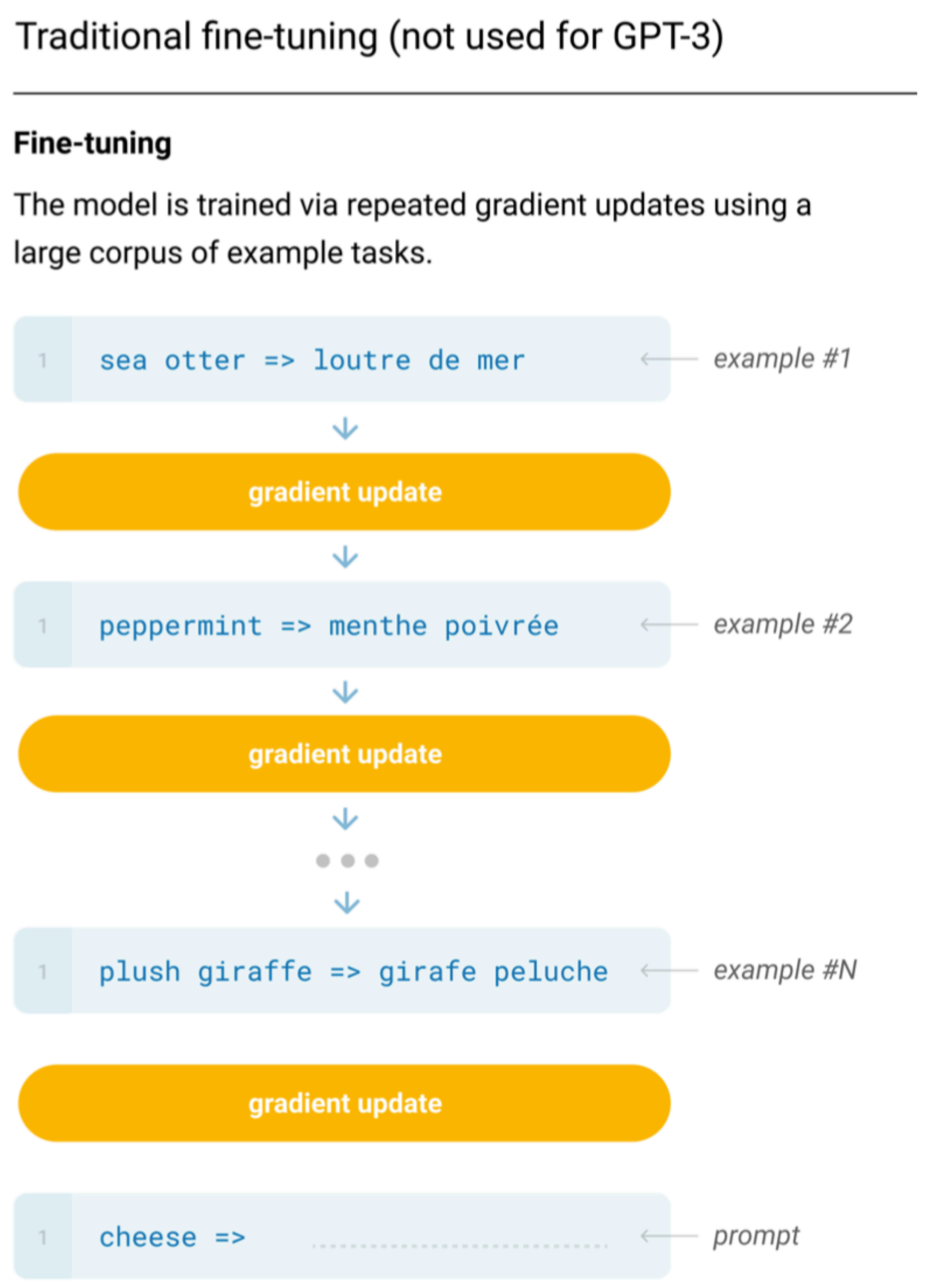
```
Translate English to French:     ← task description
sea otter => loutre de mer       ← example
cheese =>                        ← prompt
```



Few-shot

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

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



Traditional fine-tuning (not used for GPT-3)

Fine-tuning

The model is trained via repeated gradient updates using a large corpus of example tasks.

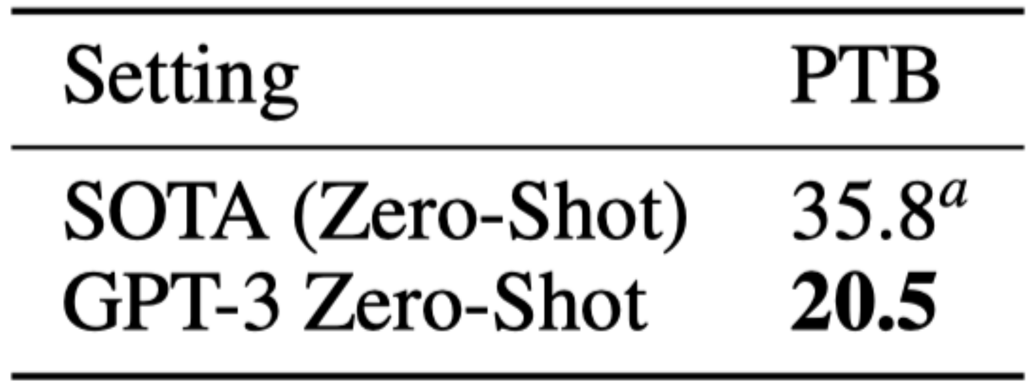
```
sea otter => loutre de mer            ← example #1
gradient update
peppermint => menthe poivrée          ← example #2
gradient update
• • •
plush giraffe => girafe peluche       ← example #N
gradient update
cheese =>                             ← prompt
```

Brown et al., 2020. *Language Models are Few-Shot Learners*

GPT-3

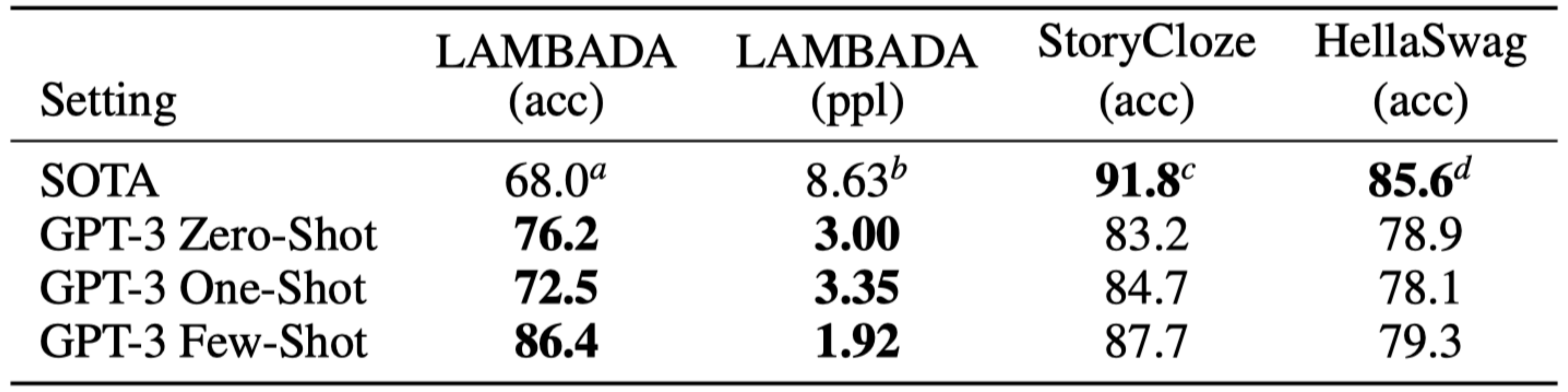
- **实验结果**
  - 在zero-shot, one-shot, few-shot实验设置上获得了非常显著的性能提升

LM



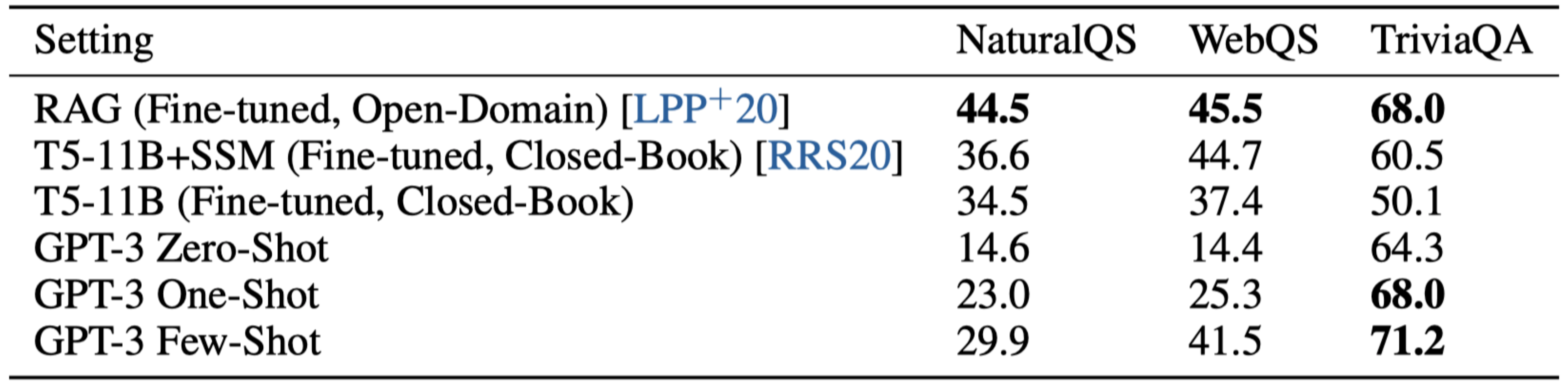
Setting	PTB
SOTA (Zero-Shot)	35.8[a]
GPT-3 Zero-Shot	**20.5**

CLOZE



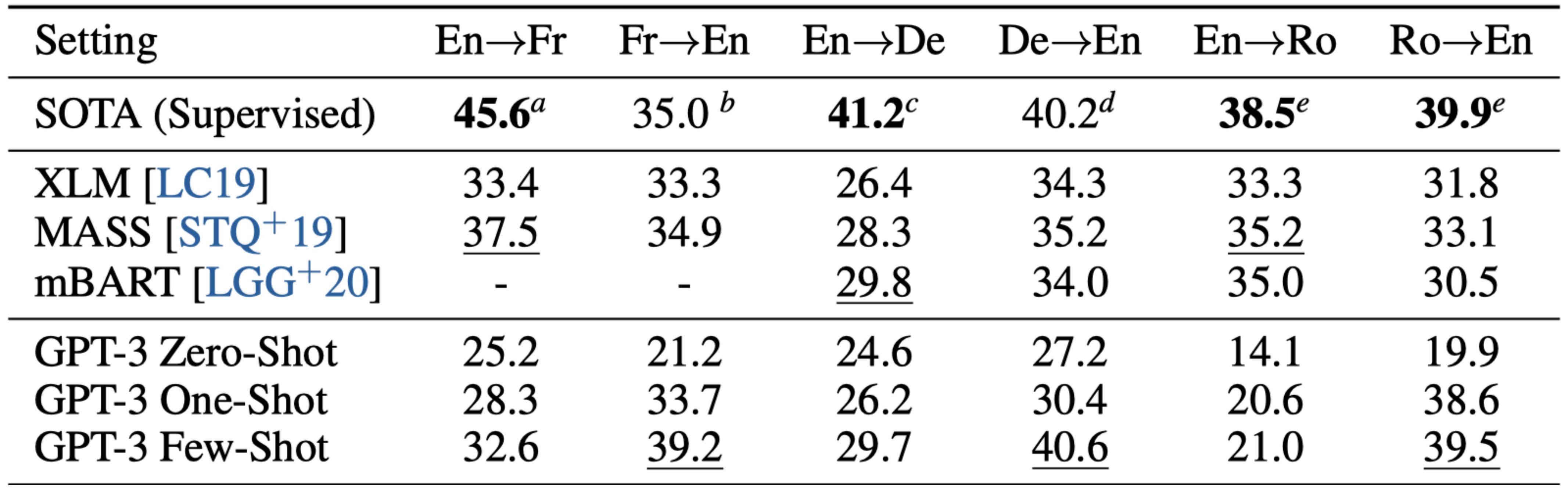
Setting	LAMBADA (acc)	LAMBADA (ppl)	StoryCloze (acc)	HellaSwag (acc)
SOTA	68.0[a]	8.63[b]	**91.8**[c]	**85.6**[d]
GPT-3 Zero-Shot	**76.2**	**3.00**	83.2	78.9
GPT-3 One-Shot	**72.5**	**3.35**	84.7	78.1
GPT-3 Few-Shot	**86.4**	**1.92**	87.7	79.3

QA



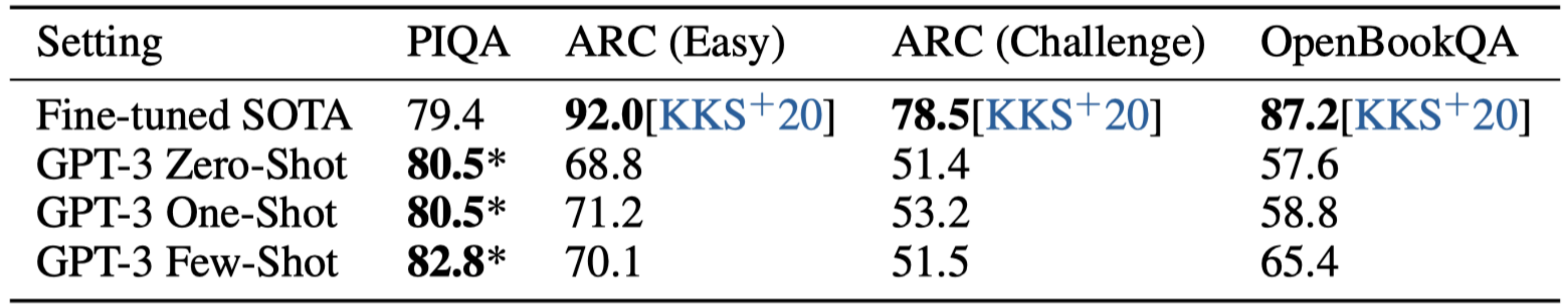
Setting	NaturalQS	WebQS	TriviaQA
RAG (Fine-tuned, Open-Domain) [LPP+20]	**44.5**	**45.5**	**68.0**
T5-11B+SSM (Fine-tuned, Closed-Book) [RRS20]	36.6	44.7	60.5
T5-11B (Fine-tuned, Closed-Book)	34.5	37.4	50.1
GPT-3 Zero-Shot	14.6	14.4	64.3
GPT-3 One-Shot	23.0	25.3	**68.0**
GPT-3 Few-Shot	29.9	41.5	**71.2**

MT



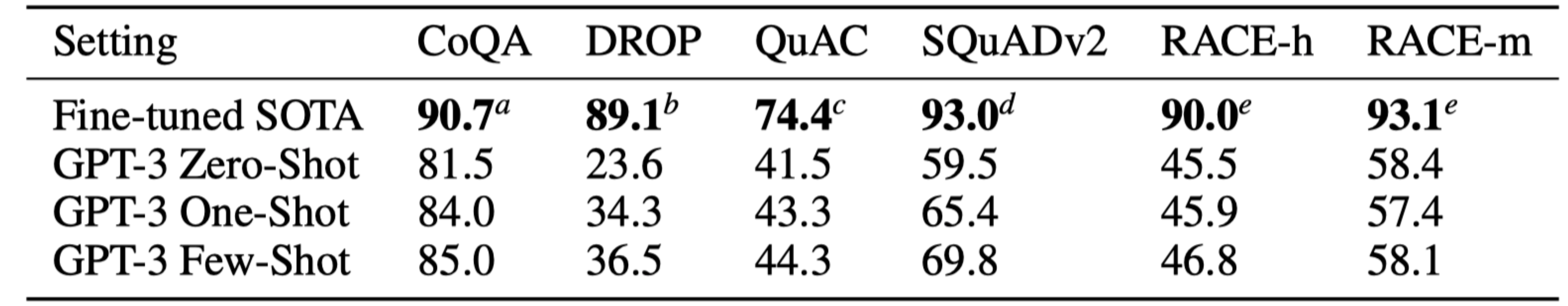
Setting	En→Fr	Fr→En	En→De	De→En	En→Ro	Ro→En
SOTA (Supervised)	**45.6**[a]	35.0 [b]	**41.2**[c]	40.2[d]	**38.5**[e]	**39.9**[e]
XLM [LC19]	33.4	33.3	26.4	34.3	33.3	31.8
MASS [STQ+19]	37.5	34.9	28.3	35.2	35.2	33.1
mBART [LGG+20]	-	-	29.8	34.0	35.0	30.5
GPT-3 Zero-Shot	25.2	21.2	24.6	27.2	14.1	19.9
GPT-3 One-Shot	28.3	33.7	26.2	30.4	20.6	38.6
GPT-3 Few-Shot	32.6	39.2	29.7	40.6	21.0	39.5

CQA



Setting	PIQA	ARC (Easy)	ARC (Challenge)	OpenBookQA
Fine-tuned SOTA	79.4	**92.0**[KKS+20]	**78.5**[KKS+20]	**87.2**[KKS+20]
GPT-3 Zero-Shot	**80.5***	68.8	51.4	57.6
GPT-3 One-Shot	**80.5***	71.2	53.2	58.8
GPT-3 Few-Shot	**82.8***	70.1	51.5	65.4

MRC



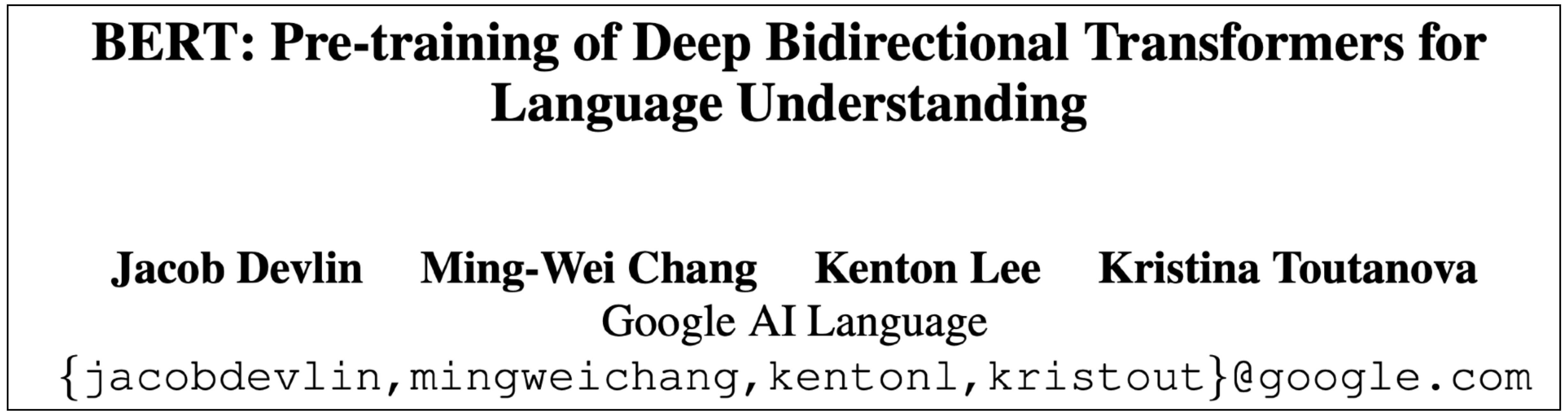
Setting	CoQA	DROP	QuAC	SQuADv2	RACE-h	RACE-m
Fine-tuned SOTA	**90.7**[a]	**89.1**[b]	**74.4**[c]	**93.0**[d]	**90.0**[e]	**93.1**[e]
GPT-3 Zero-Shot	81.5	23.6	41.5	59.5	45.5	58.4
GPT-3 One-Shot	84.0	34.3	43.3	65.4	45.9	57.4
GPT-3 Few-Shot	85.0	36.5	44.3	69.8	46.8	58.1

1 GPT 系列

➡ 2 BERT 系列

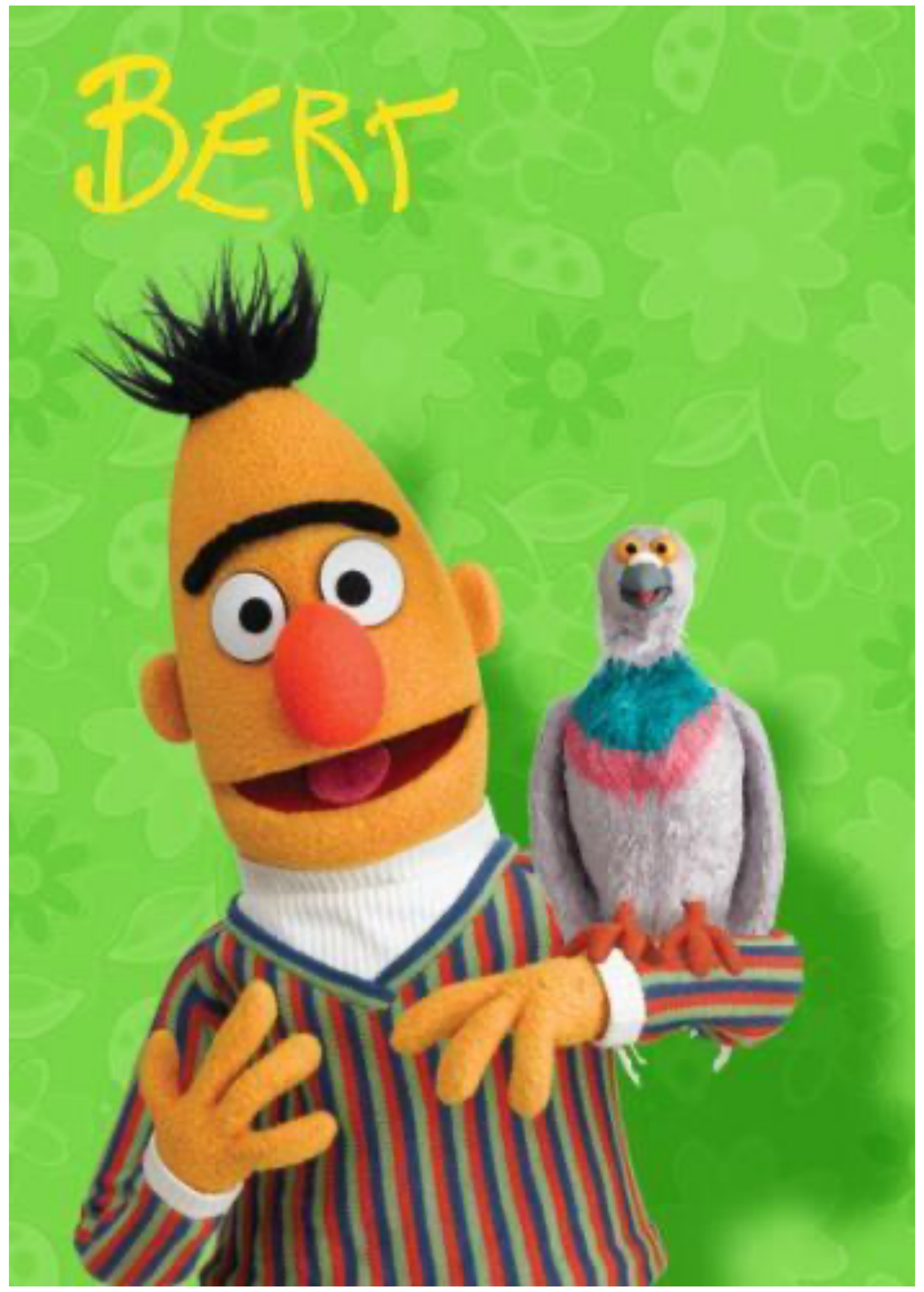
BERT

- **BERT**: **B**idirectional **E**ncoder **R**epresentations from **T**ransformers (NAACL 2019 Best Paper)
  - 提出了一种**双向**预训练语言模型方法，利用大规模自由文本训练两个无监督预训练任务
  - BERT在众多NLP任务中获得了显著性能提升
  - 进一步强调了使用通用预训练取代繁杂的任务特定的模型设计



BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

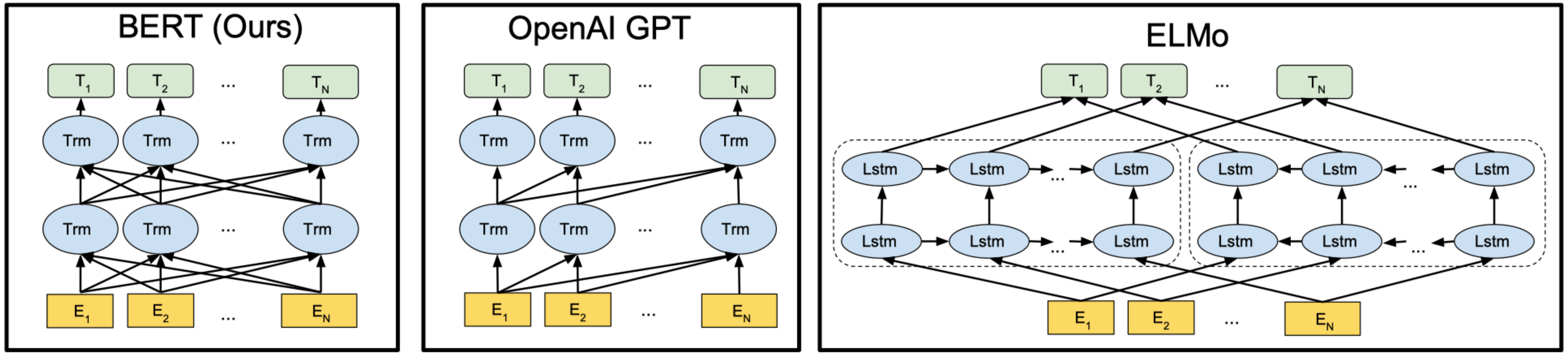
Jacob Devlin **Ming-Wei Chang** **Kenton Lee** **Kristina Toutanova**
Google AI Language
`{jacobdevlin,mingweichang,kentonl,kristout}@google.com`



Devlin et al., NAACL 2019. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*

BERT

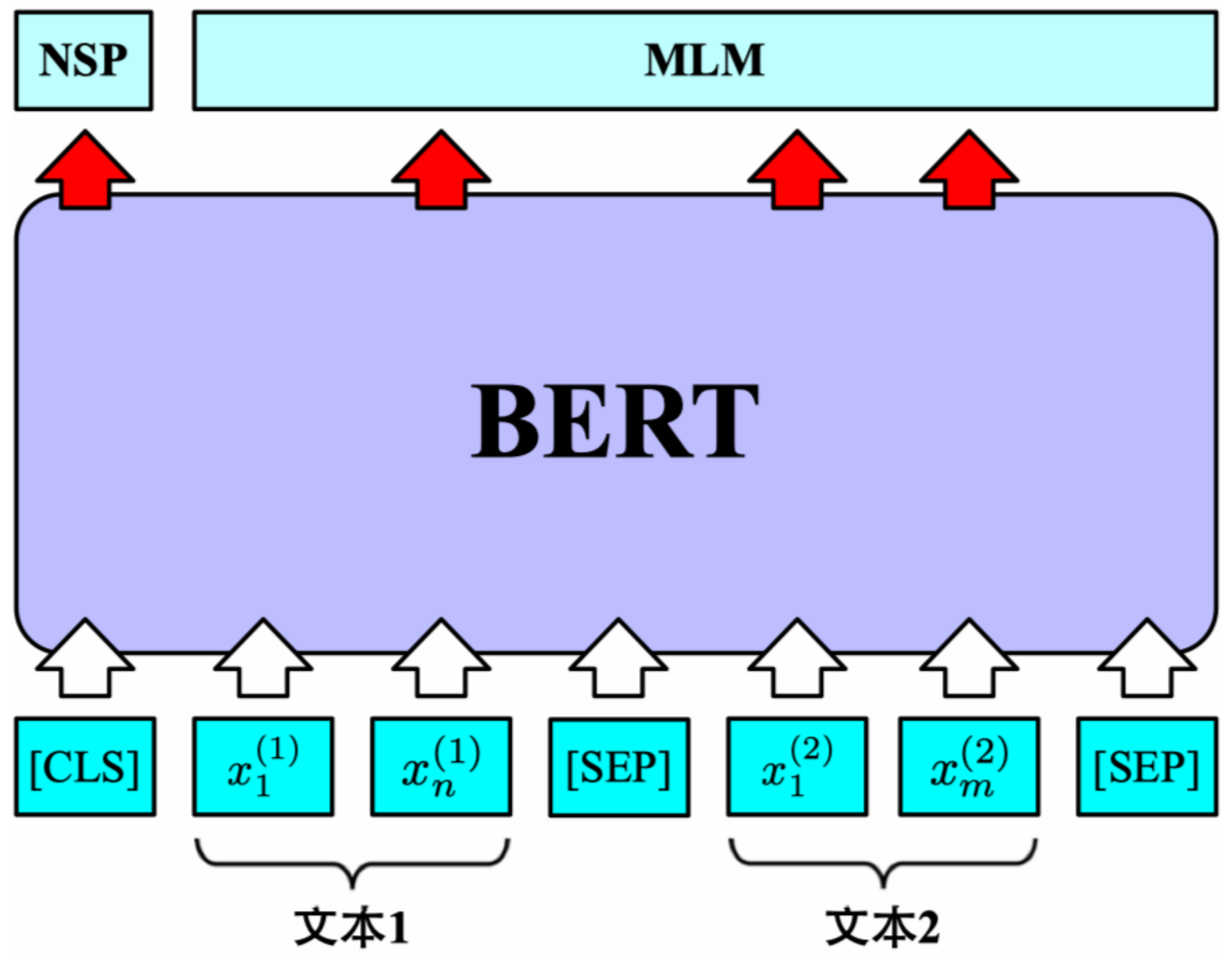
- **GPT/BERT/ELMo之间的对比**
  - GPT: **单向**从左至右的Transformer语言模型
  - ELMo: 将**独立的**前向和后向的LSTM语言模型拼接所得
  - BERT: **双向** Transformer语言模型



Devlin et al., NAACL 2019. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*

BERT

- **整体结构**
  - 由深层Transformer模型构成
    - base：12层，参数量110M
    - large：24层，参数量330M
- **预训练任务**
  - 掩码语言模型（Masked Language Model, MLM）
  - 下一个句子预测（Next Sentence Prediction, NSP）



BERT

- **预训练任务1：Masked Language Model (MLM)**
  - 将输入序列中的部分token进行掩码，并且要求模型将它们进行还原
  - 在BERT中，会将**15%**的输入文本进行mask
    - 如果掩码数量太少：模型很容易就能预测出原token是什么
    - 如果掩码数量太多：没有足够的上下文，模型的搜索空间会很大，进而很难进行预测

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

BERT

- 预训练任务1：**Masked Language Model (MLM)**
  - 问题：预训练阶段使用的表示掩码的`[MASK]`符号在下游任务中并不会出现
  - 解决方案：并不是将所有待掩码的token都替换成`[MASK]`
    - 80% of the time, replace with `[MASK]`

      `– went to the store → went to the [MASK]`
    - 10% of the time, replace random word

      `– went to the store → went to the apple`
    - 10% of the time, keep the same word

      `– went to the store → went to the store`

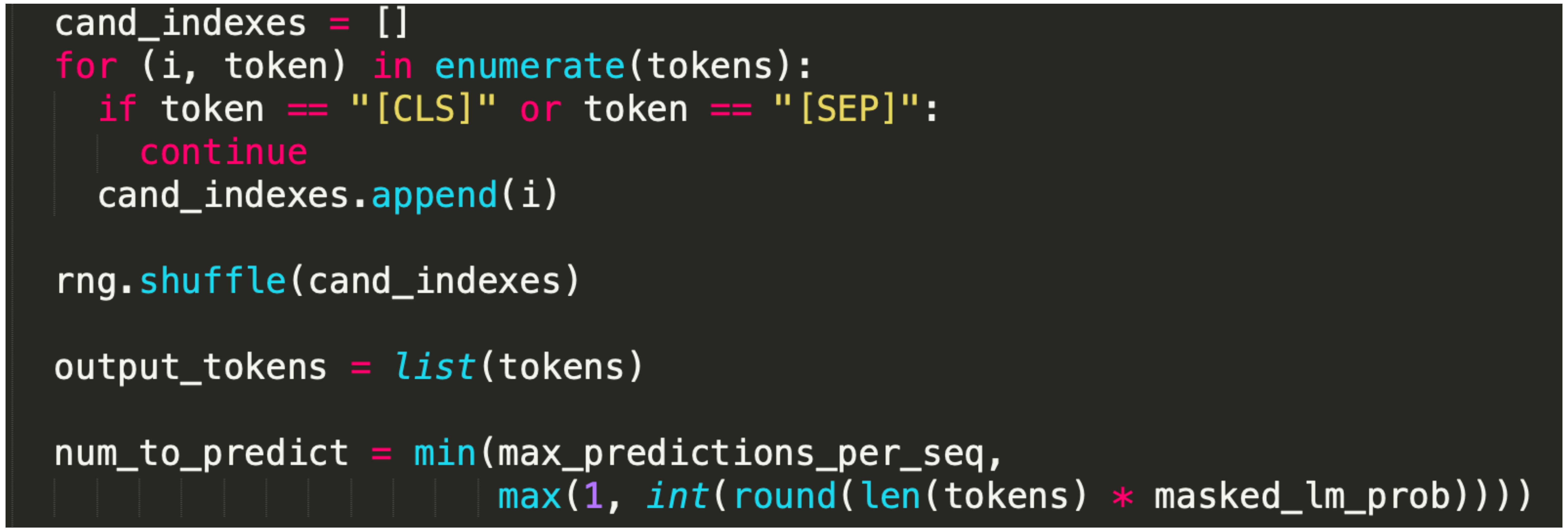
Devlin et al., NAACL 2019. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*

BERT

- **MLM的代码实现**
  - 文件: `create_pretraining_data.py`
  - 函数: `create_masked_lm_predictions()`
  - 参数列表
    - Tokens (list): tokenized sequence tokens
    - masked_lm_prob (float): how many words (proportion) should be masked
    - max_predictions_per_seq (int): maximum predictions per sequence
    - vocab_words (list): vocabulary
    - rng: random.Random(seed)

BERT

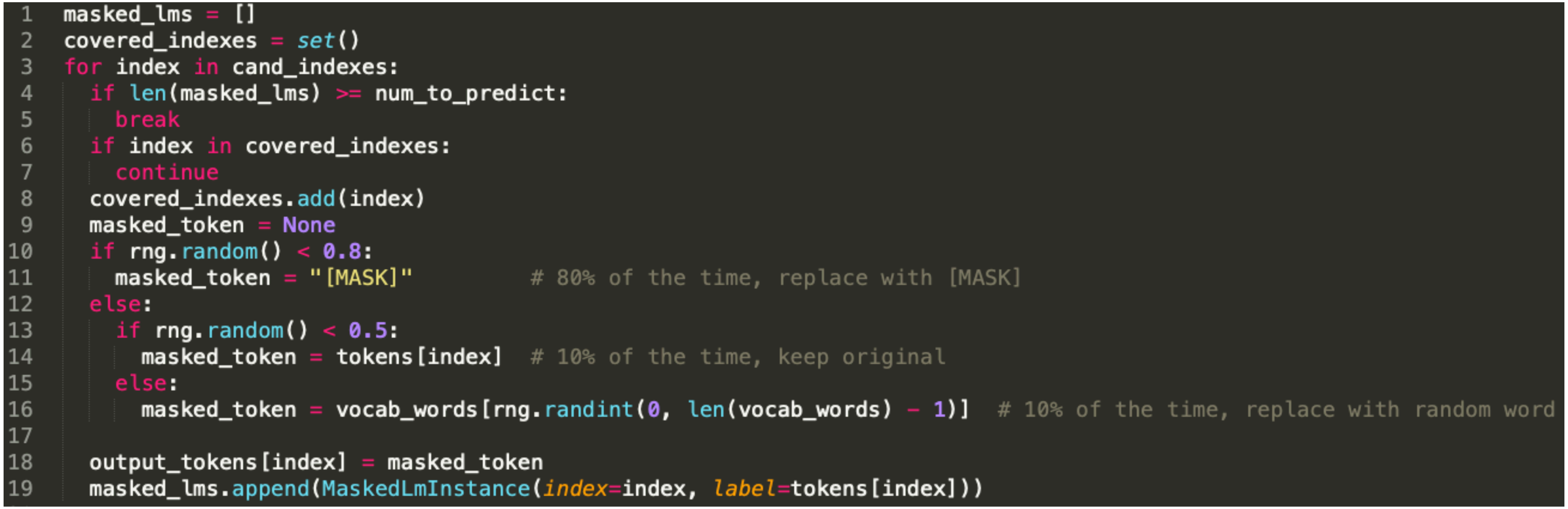
- **第一步: 生成候选下标集合**
  - 跳过 [CLS] 和 [SEP]
  - 将候选下标进行打乱
  - 确定要预测的token数量



```
cand_indexes = []
for (i, token) in enumerate(tokens):
  if token == "[CLS]" or token == "[SEP]":
    continue
  cand_indexes.append(i)

rng.shuffle(cand_indexes)

output_tokens = list(tokens)

num_to_predict = min(max_predictions_per_seq,
                     max(1, int(round(len(tokens) * masked_lm_prob))))
```

BERT

- **第二步：根据MLM算法对输入文本进行mask**
  - 生成随机数来决定用哪一种掩码方法



```python
masked_lms = []
covered_indexes = set()
for index in cand_indexes:
  if len(masked_lms) >= num_to_predict:
    break
  if index in covered_indexes:
    continue
  covered_indexes.add(index)
  masked_token = None
  if rng.random() < 0.8:
    masked_token = "[MASK]"          # 80% of the time, replace with [MASK]
  else:
    if rng.random() < 0.5:
      masked_token = tokens[index]  # 10% of the time, keep original
    else:
      masked_token = vocab_words[rng.randint(0, len(vocab_words) - 1)]  # 10% of the time, replace with random word

  output_tokens[index] = masked_token
  masked_lms.append(MaskedLmInstance(index=index, label=tokens[index]))
```

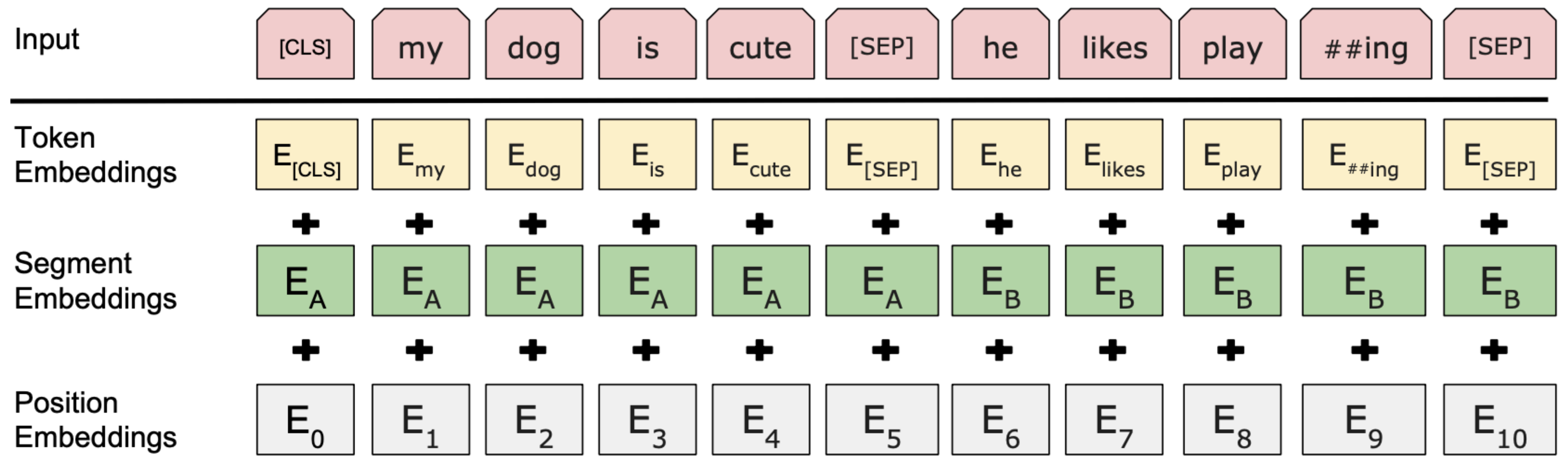
BERT

- **预训练任务2：Next Sentence Prediction (NSP)**
  - 学习两段文本之间的关系（上下文信息）
  - 预测*Sentence B*是否是*Sentence A*的下一个句子

Sentence A = `The man went to the store.` **Sentence B =** `He bough a gallon of milk.` **Label =** `IsNextSentence`	**Sentence A =** `The man went to the store.` **Sentence B =** `Penguins are flightless.` **Label =** `NotNextSentence`

BERT

- **输入表示**
  - 使用了一个包含 30,000 WordPiece的词表
  - 最终的输入表示包含三部分：词向量、块向量、位置向量
    - 块向量：表示当前token属于哪一个块（部分）
    - 位置向量：表示当前token的绝对位置

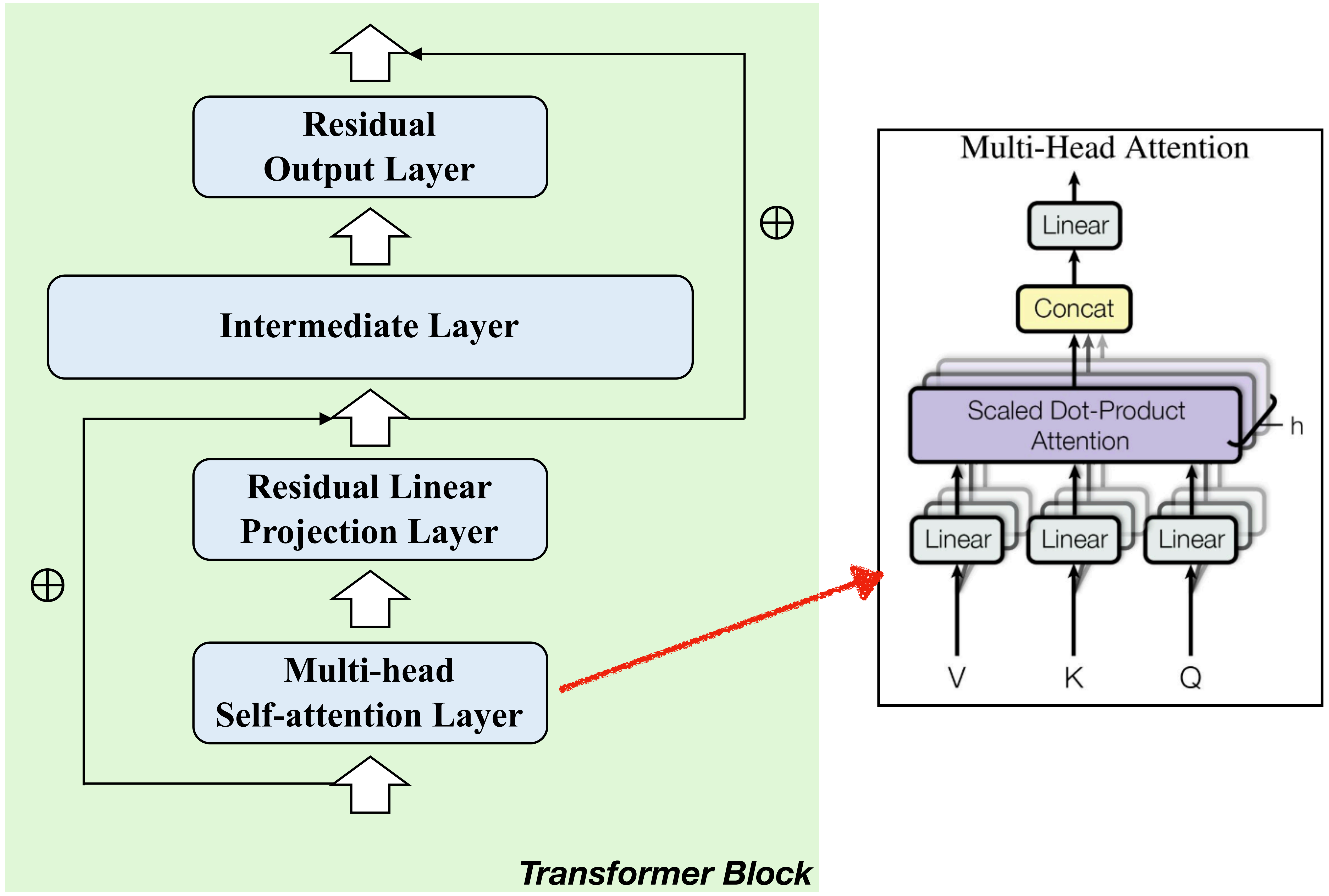


Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8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

Devlin et al., NAACL 2019. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*

BERT

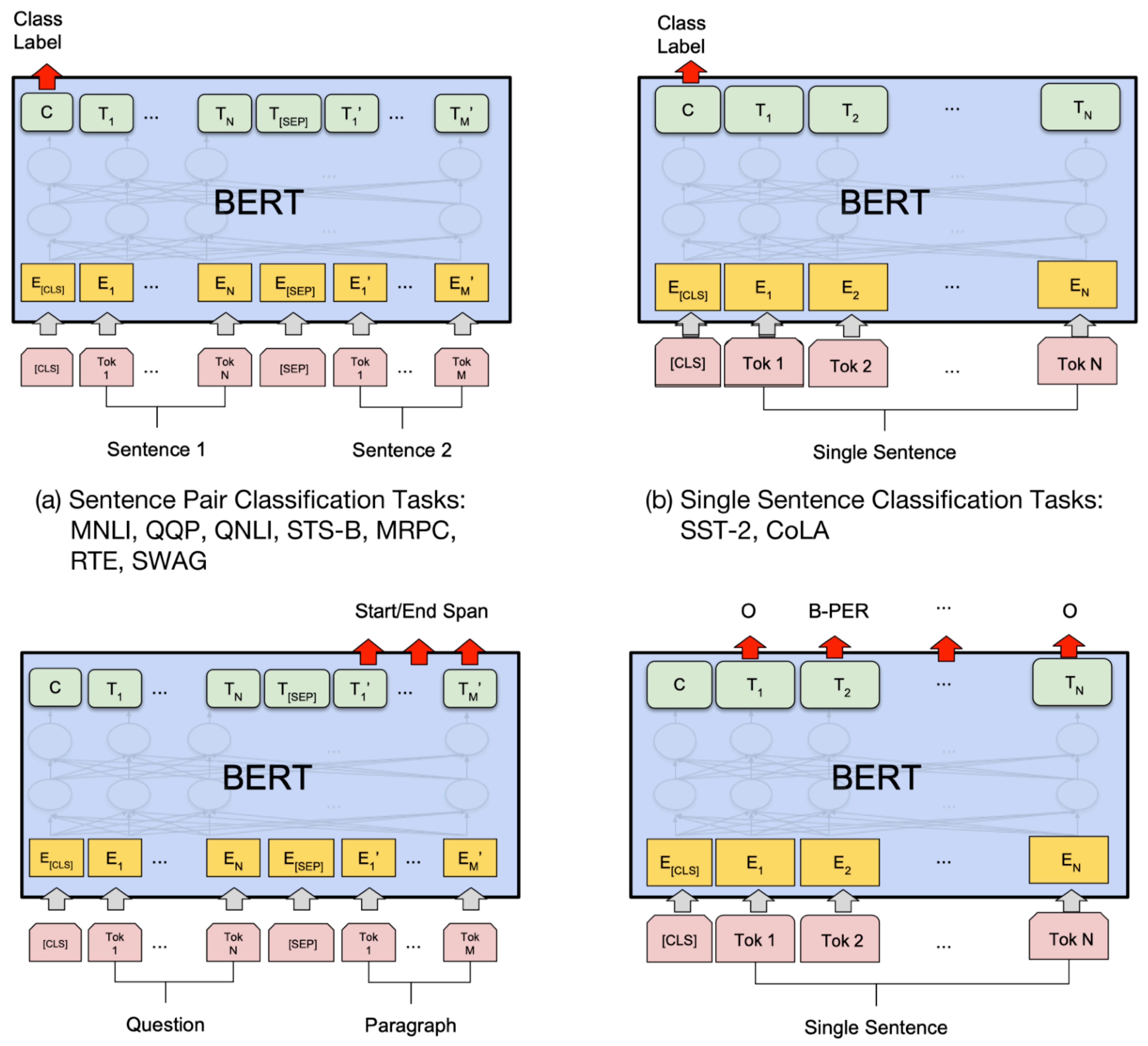
- **深层Transformer编码器**
  - BERT中使用了12或24层Transformers
  - 主循环
    - Multi-head Self-attention Layer
    - Residual Linear Projection Layer (+LayerNorm)
    - Intermediate Layer
    - Residual Output Layer (+LayerNorm)



Devlin et al., NAACL 2019. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*

BERT

- 在下游任务中精调BERT
  - 输入
    - 单句分类： [CLS] sent1
    - 句对分类： [CLS] sent1 [SEP] sent2
    - 阅读理解： [CLS] Q [SEP] P
  - 剩余的建模事情交给BERT 😄
  - 输出
    - 单句/句对分类： [CLS] → 标签
    - 阅读理解：输出答案开始和结束指针



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

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

(c) Question Answering Tasks: SQuAD v1.1

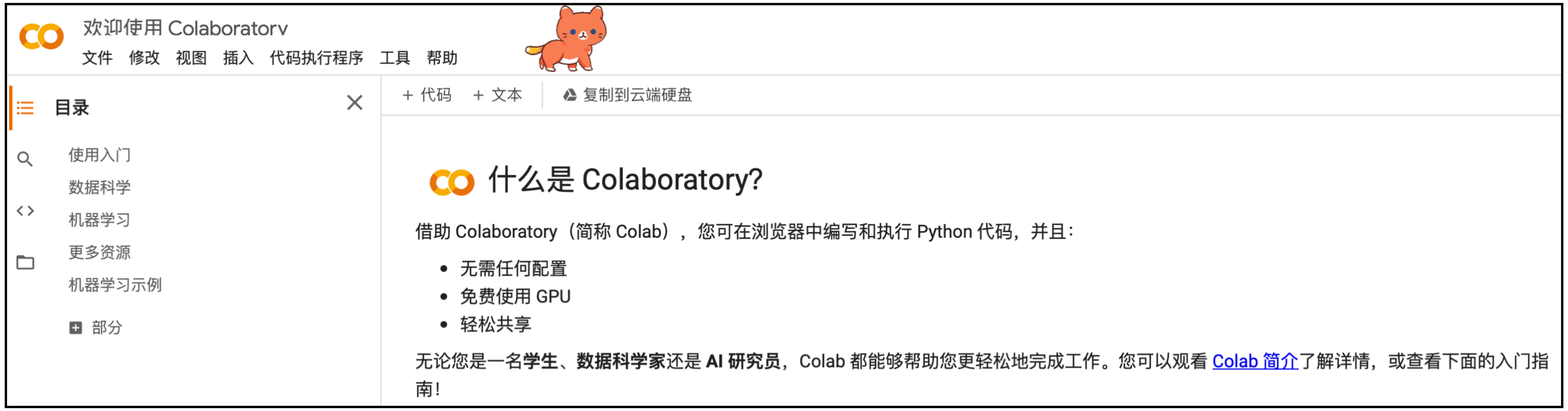
(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

***Devlin et al., NAACL 2019.** BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*

Let’s coding!

Colab

- **Google Colaboratory**（https://colab.research.google.com/）
  - 提供了一种基于Jupyter Notebook的交互式在线编程平台
  - 目前免费提供一块GPU（K80/T4/P4）或者TPU（v2）使用，适合运行轻量级的实验
  - Pro版本：每月$9.99，可支持更长运行时，更快GPU（T4/P100/V100），更多内存

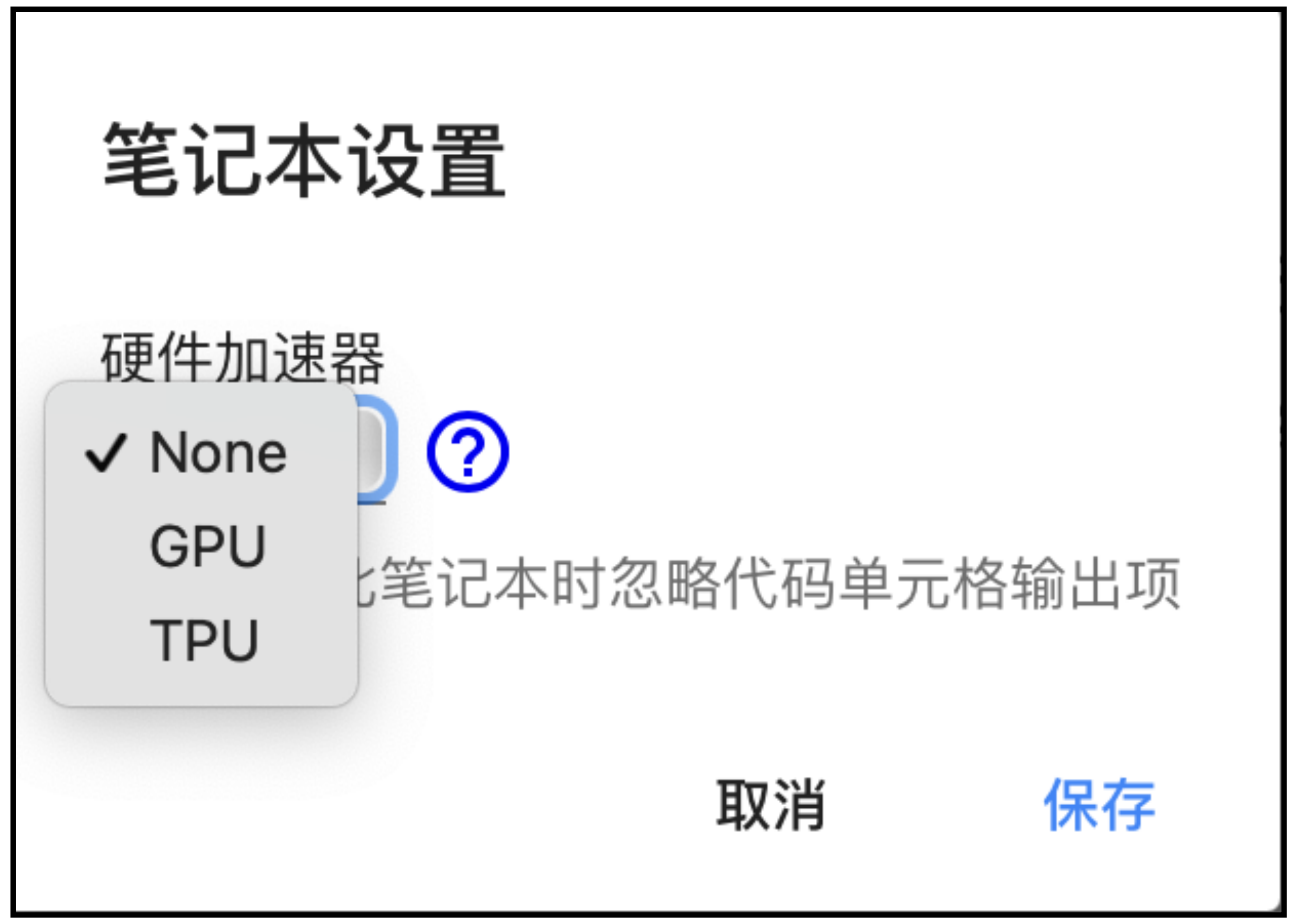


什么是 Colaboratory?

借助 Colaboratory（简称 Colab），您可在浏览器中编写和执行 Python 代码，并且：

- 无需任何配置
- 免费使用 GPU
- 轻松共享

无论您是一名**学生**、**数据科学家**还是 **AI 研究员**，Colab 都能够帮助您更轻松地完成工作。您可以观看 Colab 简介了解详情，或查看下面的入门指南！



笔记本设置

硬件加速器: [✓ None] GPU TPU

Colab

- **Google Colaboratory**（https://colab.research.google.com/）

优点😄

免费！
可以体验TPU
随时随地可以编程
下载速度极快

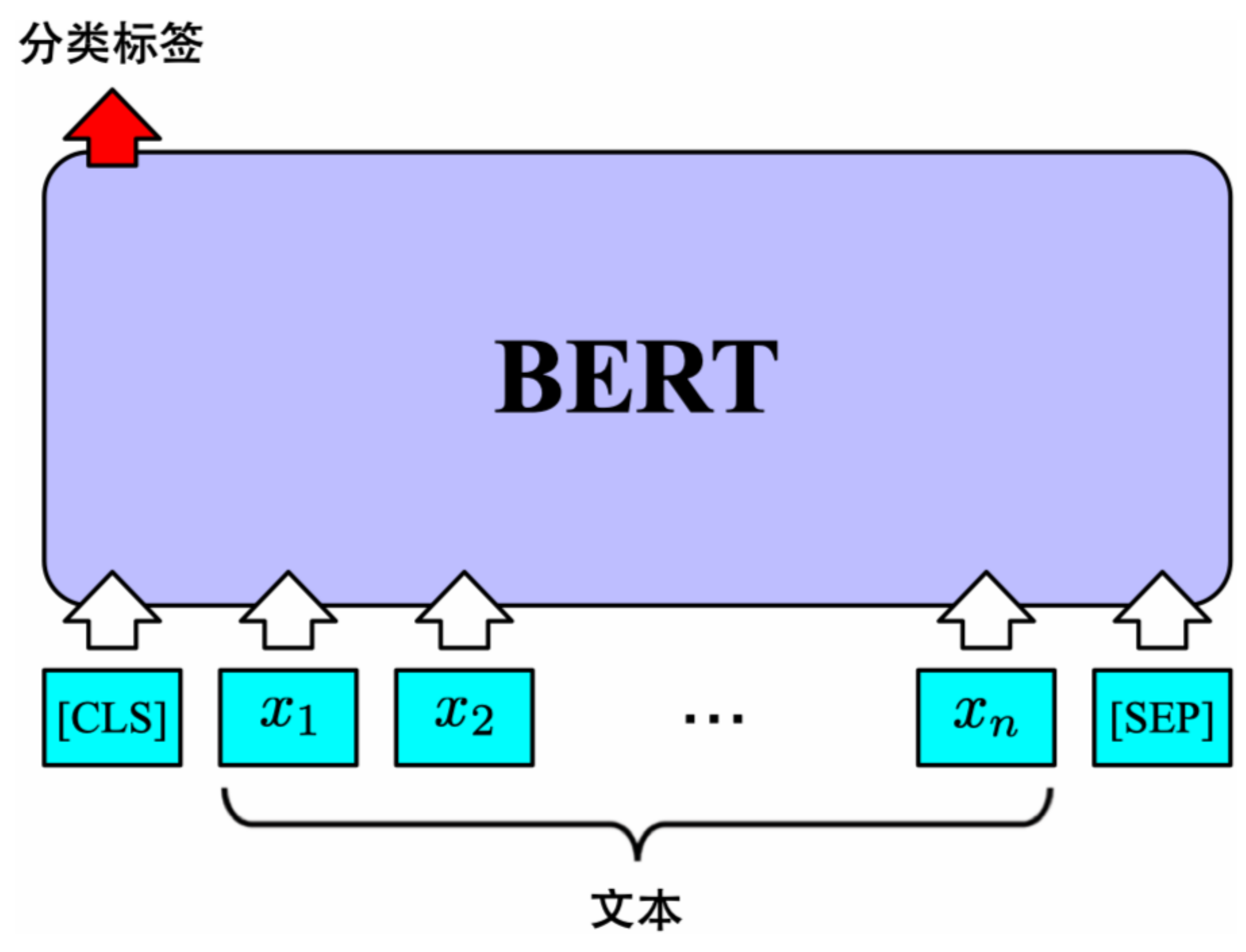
缺点😢

长时间不动会断开程序
需要科学访问
免费版的GPU稍慢

BERT

- 在单句文本分类中精调BERT

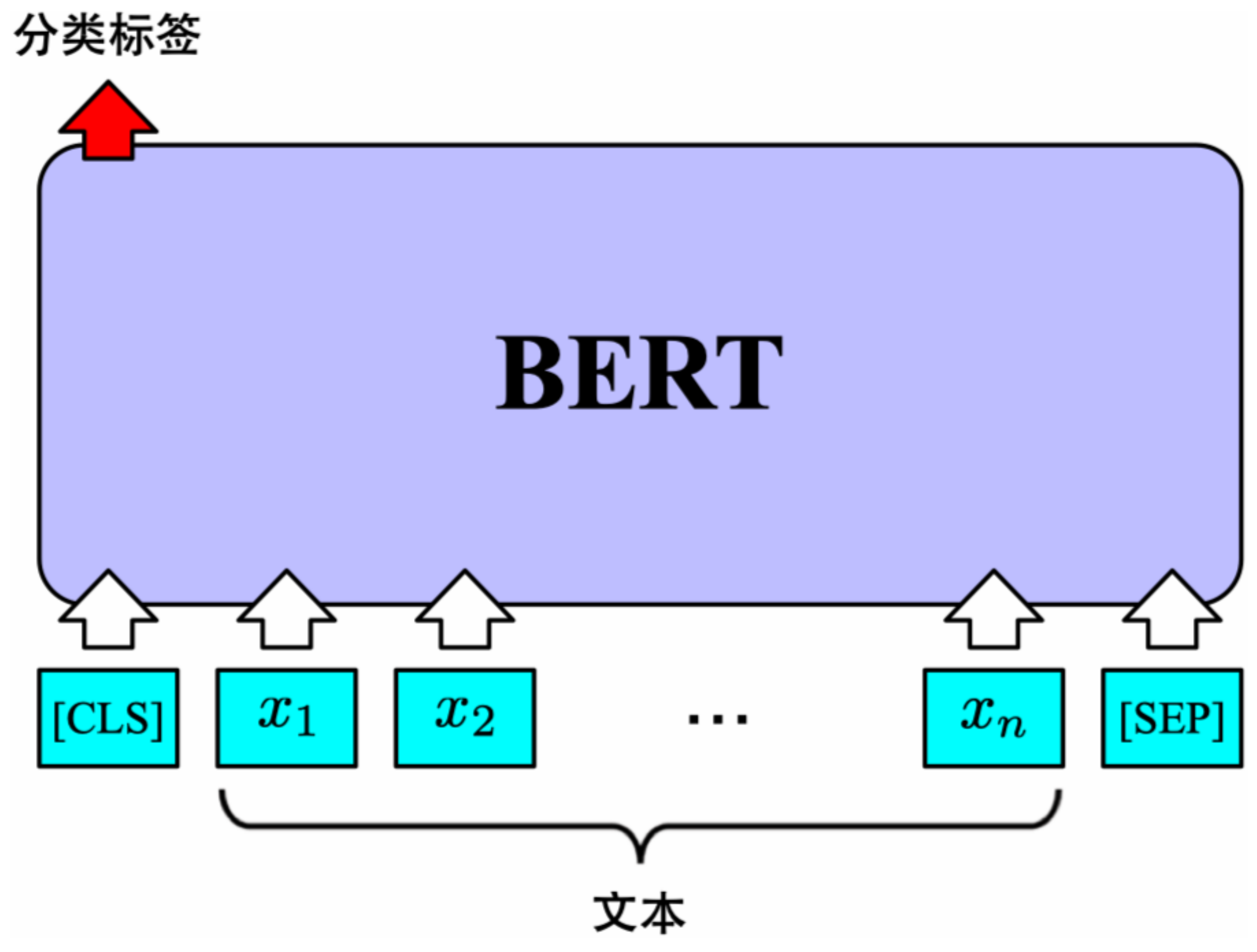
```
import numpy as np
from datasets import load_dataset, load_metric
from transformers import BertTokenizerFast, BertForSequenceClassification,
    TrainingArguments, Trainer

# 加载训练数据、分词器、预训练模型和评价方法
dataset = load_dataset('glue', 'sst2')
tokenizer = BertTokenizerFast.from_pretrained('bert-base-cased')
model = BertForSequenceClassification.from_pretrained('bert-base-cased',
    return_dict=True)
metric = load_metric('glue', 'sst2')

# 对训练集分词
def tokenize(examples):
    return tokenizer(examples['sentence'], truncation=True, padding='
    max_length')
dataset = dataset.map(tokenize, batched=True)
encoded_dataset = dataset.map(lambda examples: {'labels': examples['label']},
    batched=True)
```



BERT

- **在单句文本分类中精调BERT**

```python
# 将数据集格式化为torch.Tensor类型以训练PyTorch模型
columns = ['input_ids', 'token_type_ids', 'attention_mask', 'labels']
encoded_dataset.set_format(type='torch', columns=columns)

# 定义评价指标
def compute_metrics(eval_pred):
    predictions, labels = eval_pred
    return metric.compute(predictions=np.argmax(predictions, axis=1),
    references=labels)

# 定义训练参数TrainingArguments，默认使用AdamW优化器
args = TrainingArguments(
    "ft-sst2",                          # 输出路径，存放检查点和其他输出文件
    evaluation_strategy="epoch",        # 定义每轮结束后进行评价
    learning_rate=2e-5,                 # 定义初始学习率
    per_device_train_batch_size=16,     # 定义训练批次大小
    per_device_eval_batch_size=16,      # 定义测试批次大小
    num_train_epochs=2,                 # 定义训练轮数
)
```

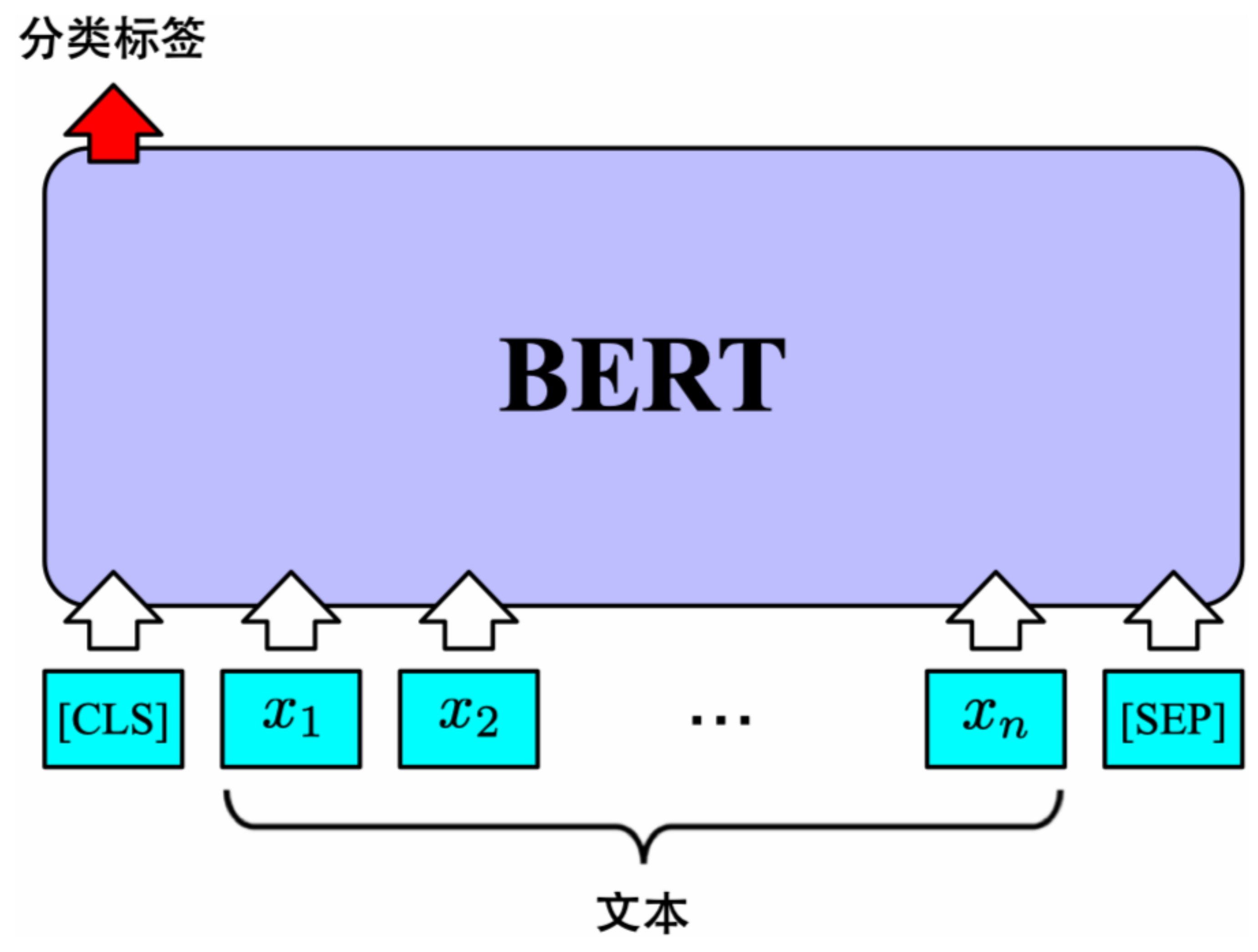


Devlin et al., NAACL 2019. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*

BERT

- **在单句文本分类中精调BERT**
  - 开始训练

```
# 定义Trainer，指定模型和训练参数，输入训练集、验证集、分词器和评价函数
trainer = Trainer(
    model,
    args,
    train_dataset=encoded_dataset["train"],
    eval_dataset=encoded_dataset["validation"],
    tokenizer=tokenizer,
    compute_metrics=compute_metrics
)

# 开始训练！（主流GPU上耗时约几小时）
trainer.train()
```



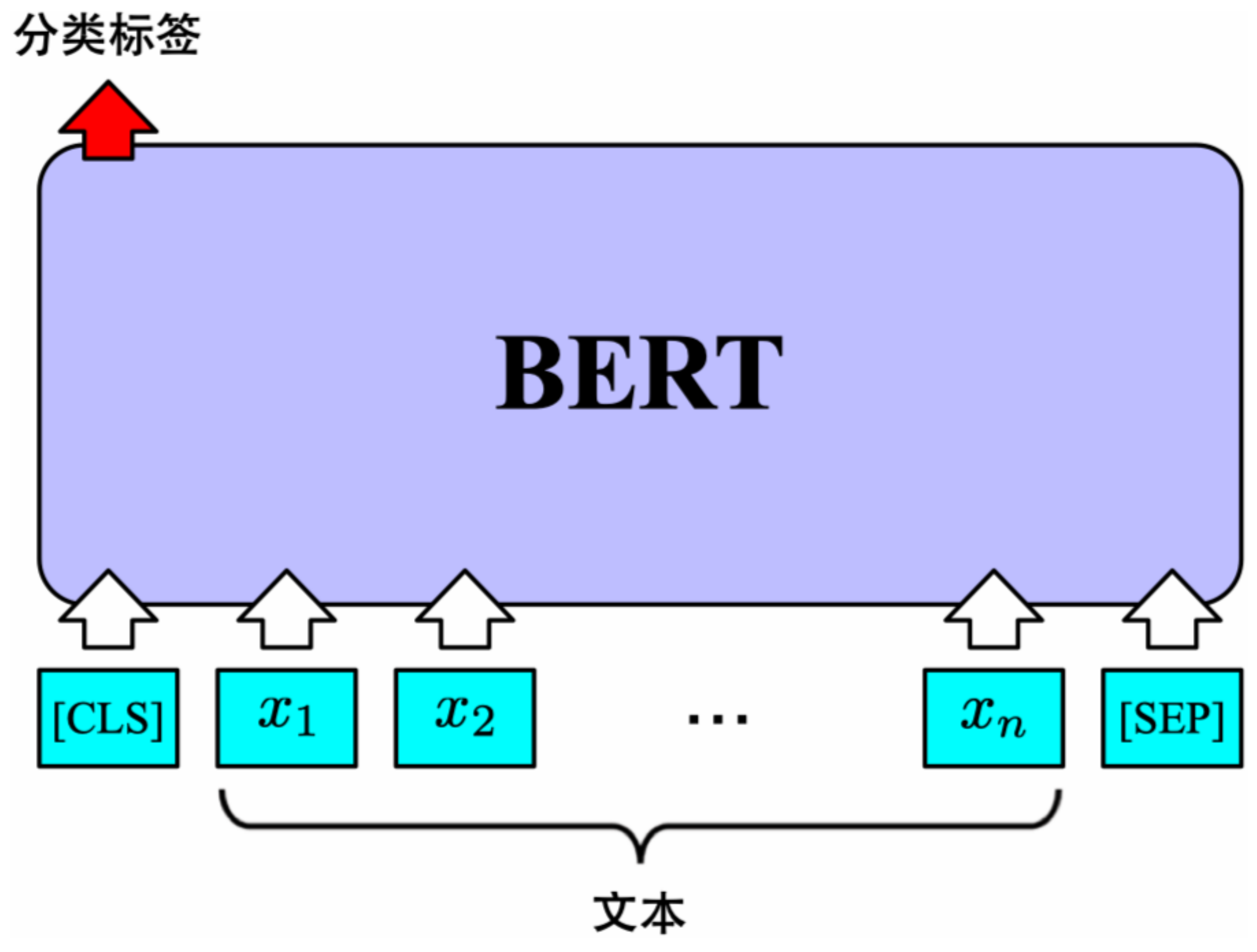
BERT

- 在单句文本分类中精调BERT
  - 预测

```
# 在训练完毕后，开始测试！
trainer.evaluate()
```

  - 结果

```
{'epoch': 2,
 'eval_accuracy': 0.7350917431192661,
 'eval_loss': 0.9351930022239685}
```

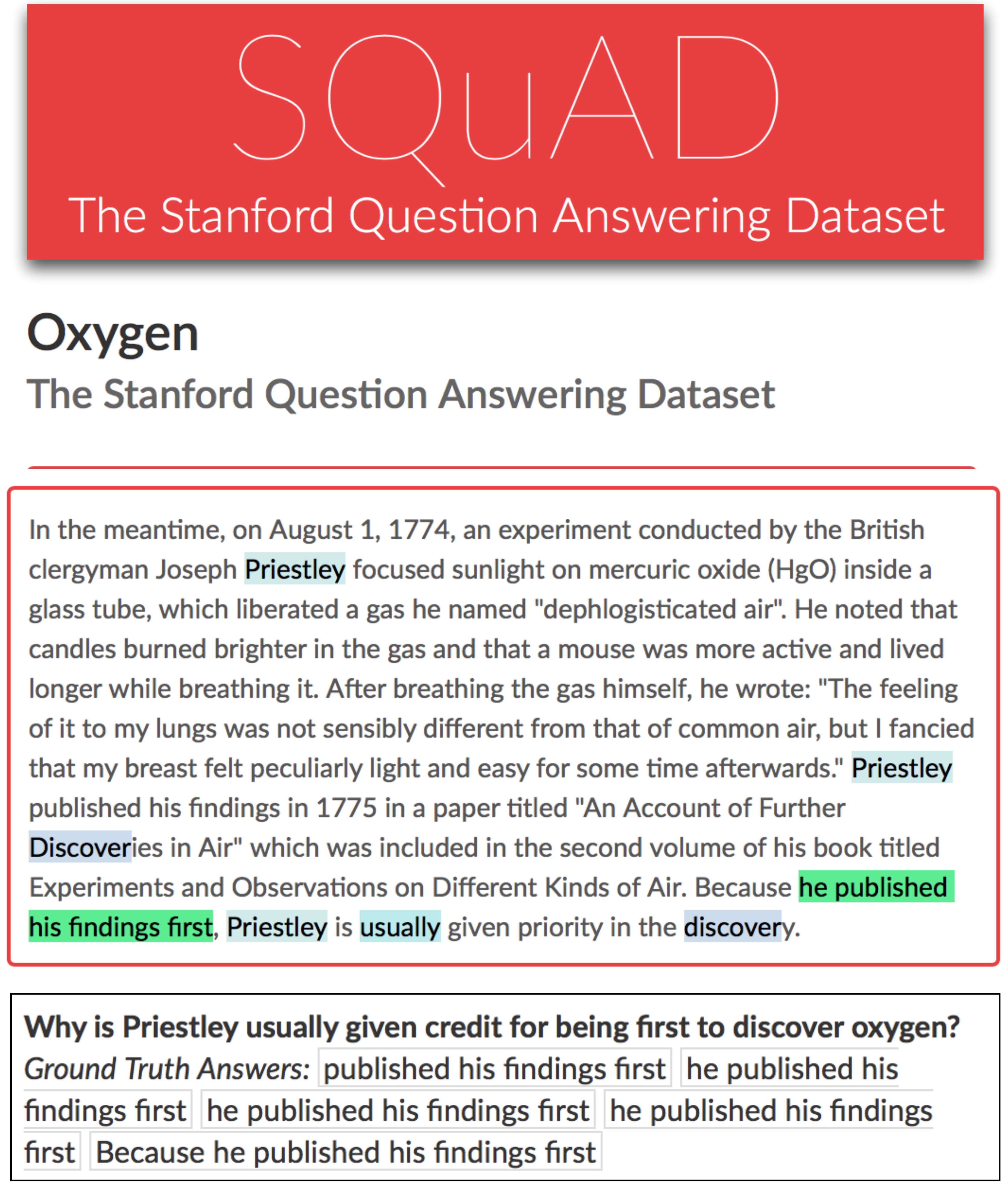


官方实现

Let's read the code!

BERT

- **SQuAD (Stanford Question Answering Dataset)**
  - 斯坦福大学发布，开启了阅读理解数据集的新篇章
    - 答案不再是单个词，扩展到短语甚至是句子，难度更大
    - 问题不再是自动生成，采用了人工标注，问题更加自然
    - 相比之前的人工标注集合，规模较大（10万问题）
  - **篇章**：维基百科文章，切分成若干段落组成“小篇章”
  - **问题**：由人工提问，问题类型较多，丰富了问题的多样
  - **答案**：篇章中某个连续片段，搜索空间更大，难度更高
  - SQuAD 2.0加入了“不可回答的问题”，进一步提升难度


SQuAD
The Stanford Question Answering Dataset

Oxygen
The Stanford Question Answering Dataset

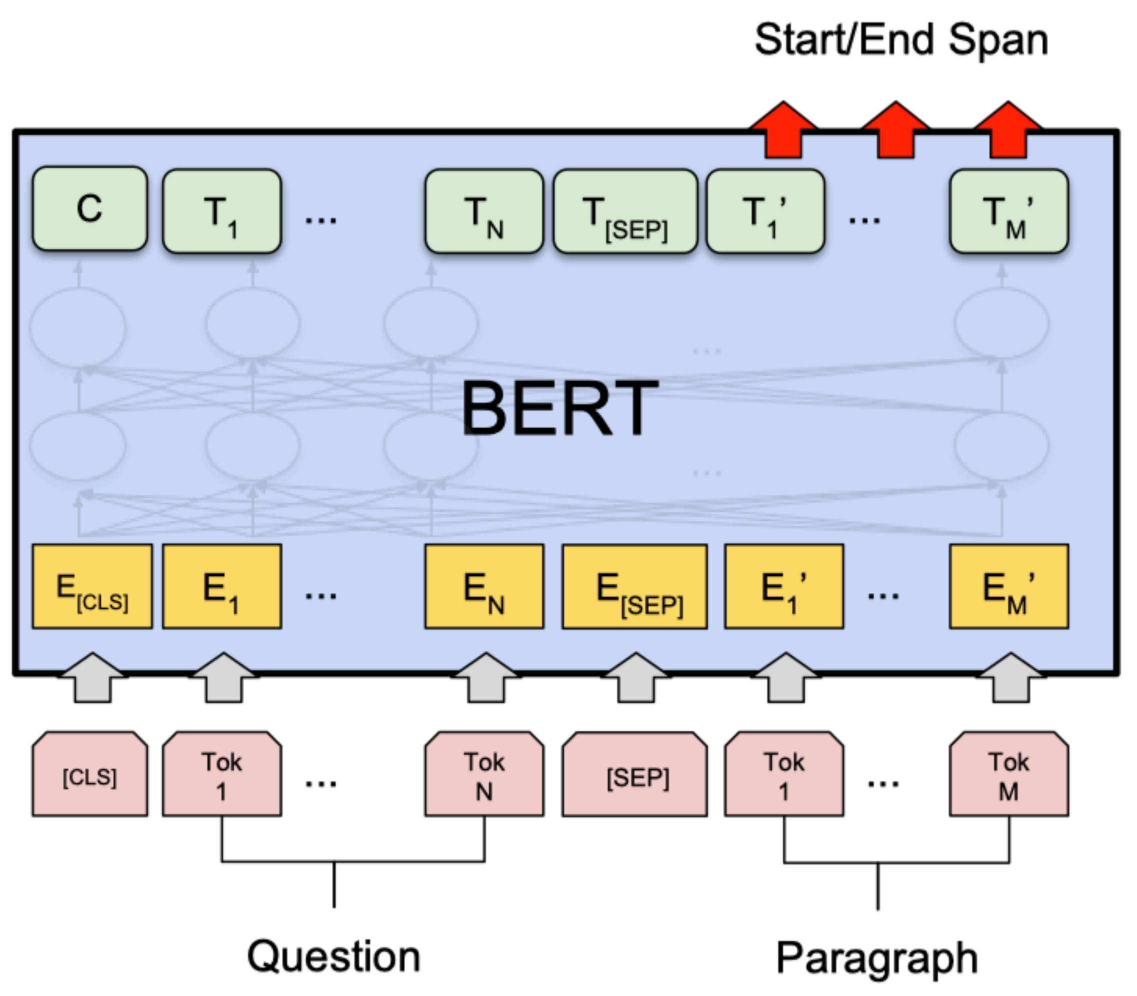
In the meantime, on August 1, 1774, an experiment conducted by the British clergyman Joseph Priestley focused sunlight on mercuric oxide (HgO) inside a glass tube, which liberated a gas he named "dephlogisticated air". He noted that candles burned brighter in the gas and that a mouse was more active and lived longer while breathing it. After breathing the gas himself, he wrote: "The feeling of it to my lungs was not sensibly different from that of common air, but I fancied that my breast felt peculiarly light and easy for some time afterwards." Priestley published his findings in 1775 in a paper titled "An Account of Further Discoveries in Air" which was included in the second volume of his book titled Experiments and Observations on Different Kinds of Air. Because he published his findings first, Priestley is usually given priority in the discovery.

Why is Priestley usually given credit for being first to discover oxygen?
Ground Truth Answers: published his findings first | he published his findings first | he published his findings first | he published his findings first | Because he published his findings first

Rajpurkar et al., EMNLP 2016. *SQuAD: 100,000+ Questions for Machine Comprehension of Text*

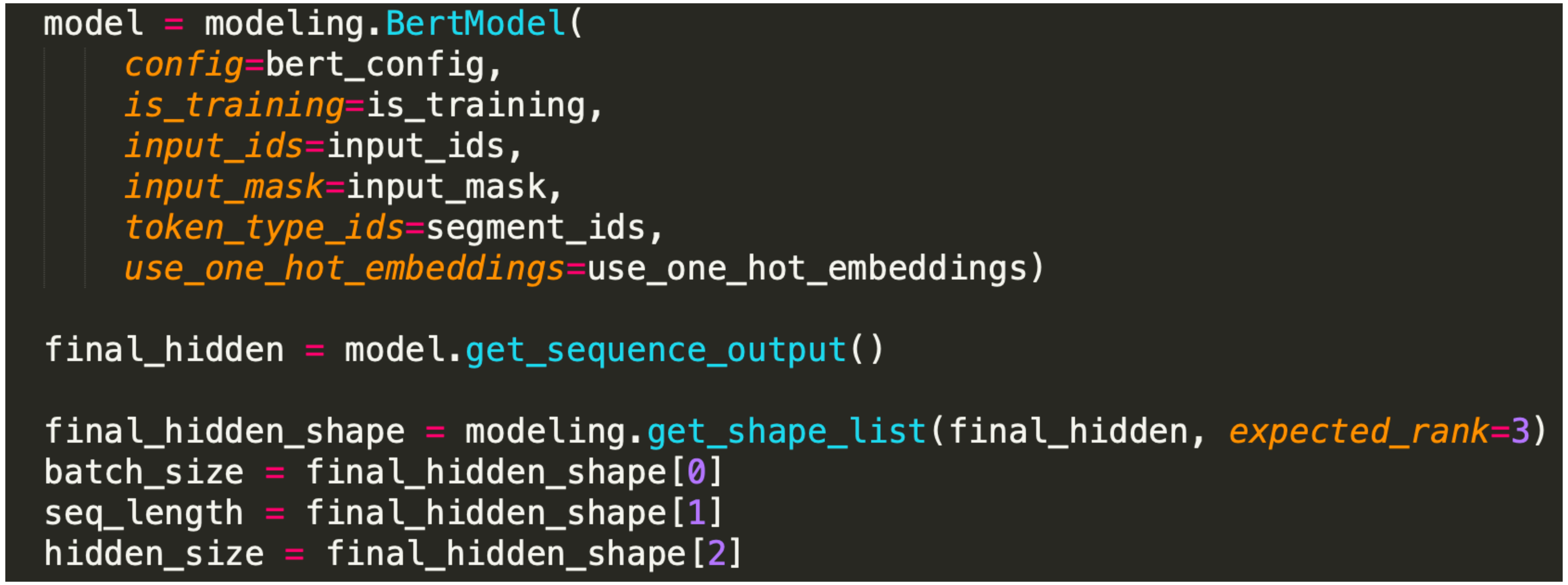
BERT

- **精调SQuAD（抽取式阅读理解任务）**
  - 文件: `run_squad.py`
  - 函数: `create_model()`
  - 参数列表
    - bert_config (json): BERT config file
    - is_training (bool): training mode option
    - input_ids (tensor): input ids for token embeddings
    - input_mask (tensor): input mask for indicating non-padding positions
    - segment_ids (tensor): segment_id tensor
    - use_one_hot_embeddings (bool)



BERT

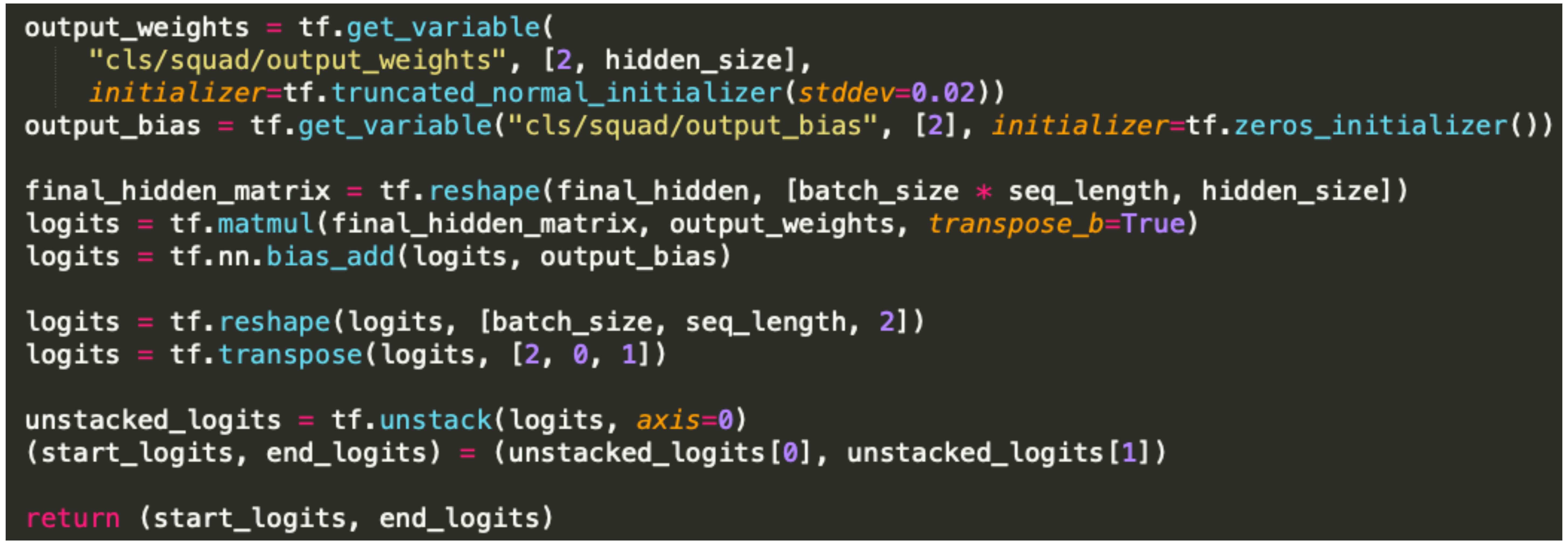
- **第一步：生成BERT的隐层输出表示**
  - 首先定义一个BERT模型，送入必要的输入
  - 输出一个3D-tensor，大小为(batch_size, seq_length, hidden_size)



```
model = modeling.BertModel(
        config=bert_config,
        is_training=is_training,
        input_ids=input_ids,
        input_mask=input_mask,
        token_type_ids=segment_ids,
        use_one_hot_embeddings=use_one_hot_embeddings)

final_hidden = model.get_sequence_output()

final_hidden_shape = modeling.get_shape_list(final_hidden, expected_rank=3)
batch_size = final_hidden_shape[0]
seq_length = final_hidden_shape[1]
hidden_size = final_hidden_shape[2]
```

BERT

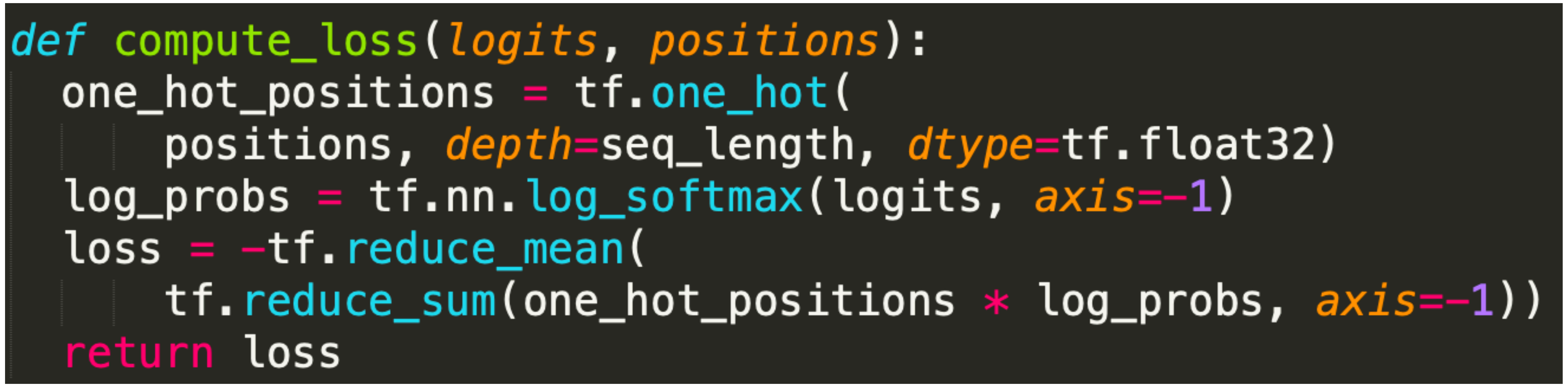
- **第二步：利用BERT表示进行答案预测**
  - 定义一个全连接层
  - 将BERT输出向量的hidden_size维度，利用全连接层变换至2（开始/结束位置各占1维）
  - 最终大小为(batch_size, seq_length, 2)



```
output_weights = tf.get_variable(
    "cls/squad/output_weights", [2, hidden_size],
    initializer=tf.truncated_normal_initializer(stddev=0.02))
output_bias = tf.get_variable("cls/squad/output_bias", [2], initializer=tf.zeros_initializer())

final_hidden_matrix = tf.reshape(final_hidden, [batch_size * seq_length, hidden_size])
logits = tf.matmul(final_hidden_matrix, output_weights, transpose_b=True)
logits = tf.nn.bias_add(logits, output_bias)

logits = tf.reshape(logits, [batch_size, seq_length, 2])
logits = tf.transpose(logits, [2, 0, 1])

unstacked_logits = tf.unstack(logits, axis=0)
(start_logits, end_logits) = (unstacked_logits[0], unstacked_logits[1])

return (start_logits, end_logits)
```

BERT

- **第三步：答案输出损失**
  - 函数：model_fn_builder() → compute_loss()
  - 篇章片段抽取可以抽象为输出两个指针，分别指向片段起始和终止位置
  - 对开始位置和终止位置计算交叉熵损失



```python
def compute_loss(logits, positions):
  one_hot_positions = tf.one_hot(
      positions, depth=seq_length, dtype=tf.float32)
  log_probs = tf.nn.log_softmax(logits, axis=-1)
  loss = -tf.reduce_mean(
      tf.reduce_sum(one_hot_positions * log_probs, axis=-1))
  return loss
```

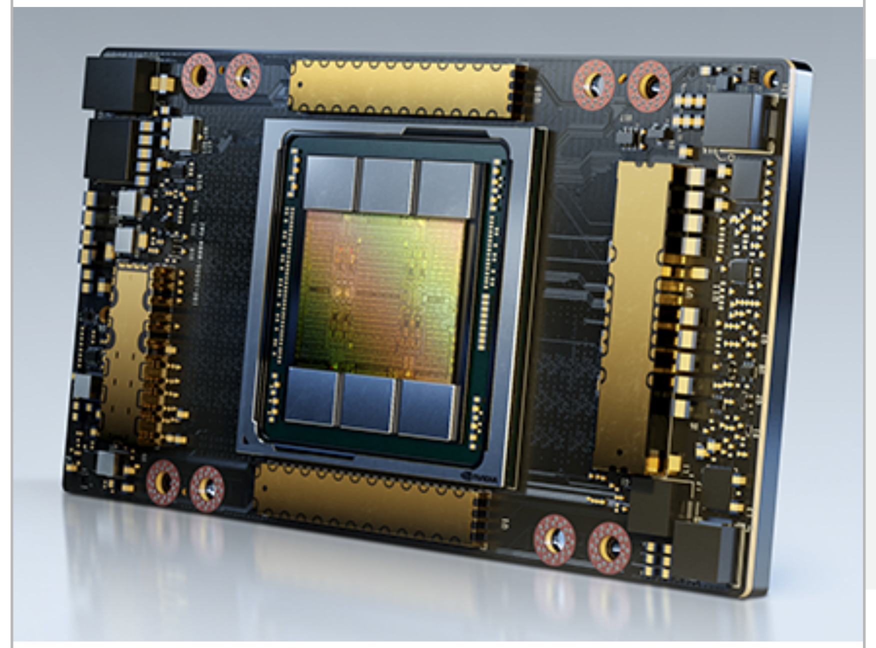
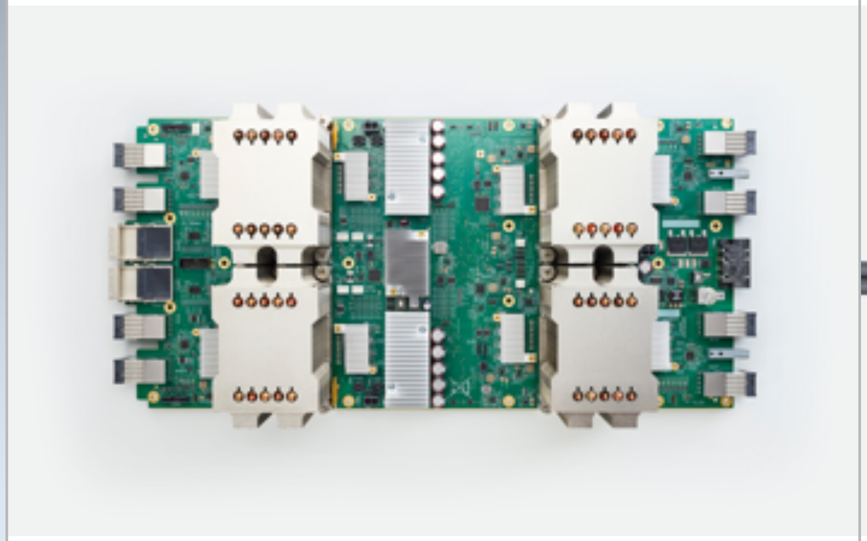
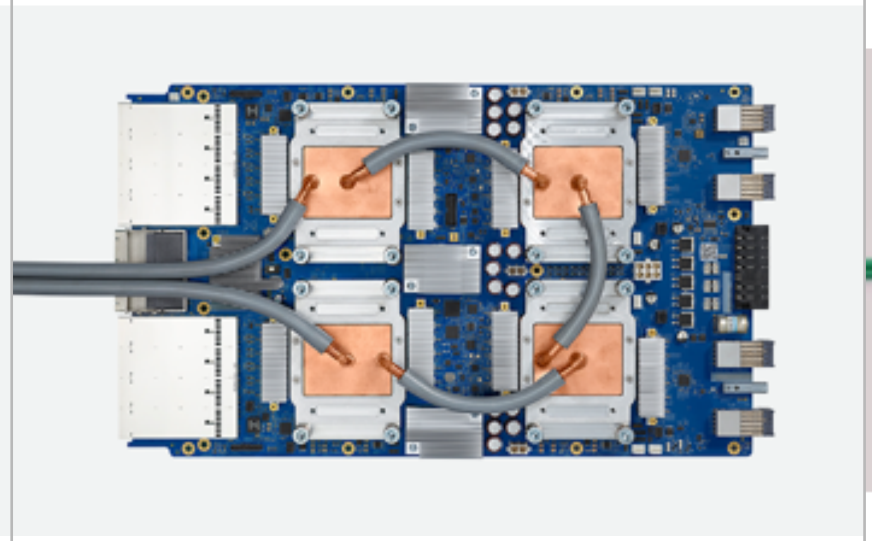
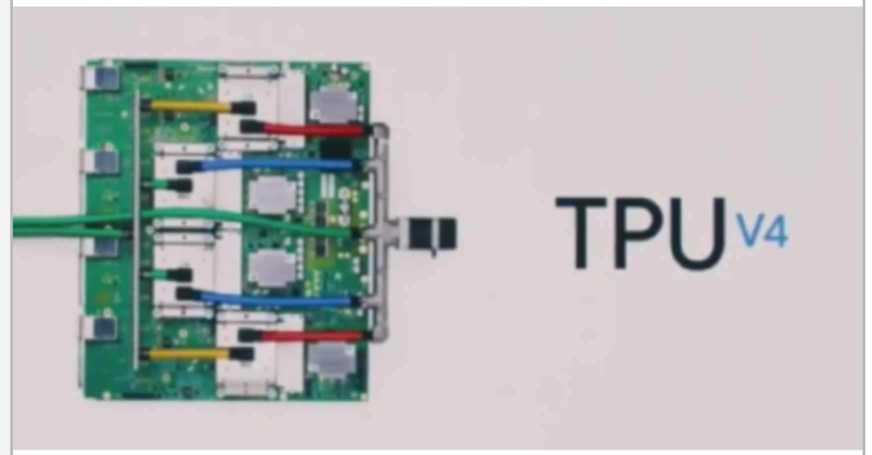
BERT

- **实验设置**
  - 数据：Wikipedia + BookCorpus (33B words in total)
  - 训练：256 batch_size * 512 max_token_length, 1M步
  - Warmup: 10K steps (总训练步数的1%)
  - 训练时长：4天
  - 计算设备
    - BERT-base: 4 Cloud TPUs in Pod config (16 chips)
    - BERT-large: 16 Cloud TPUs (64 chips)

1 TPU has 2 cores, and 4 chips each

BERT

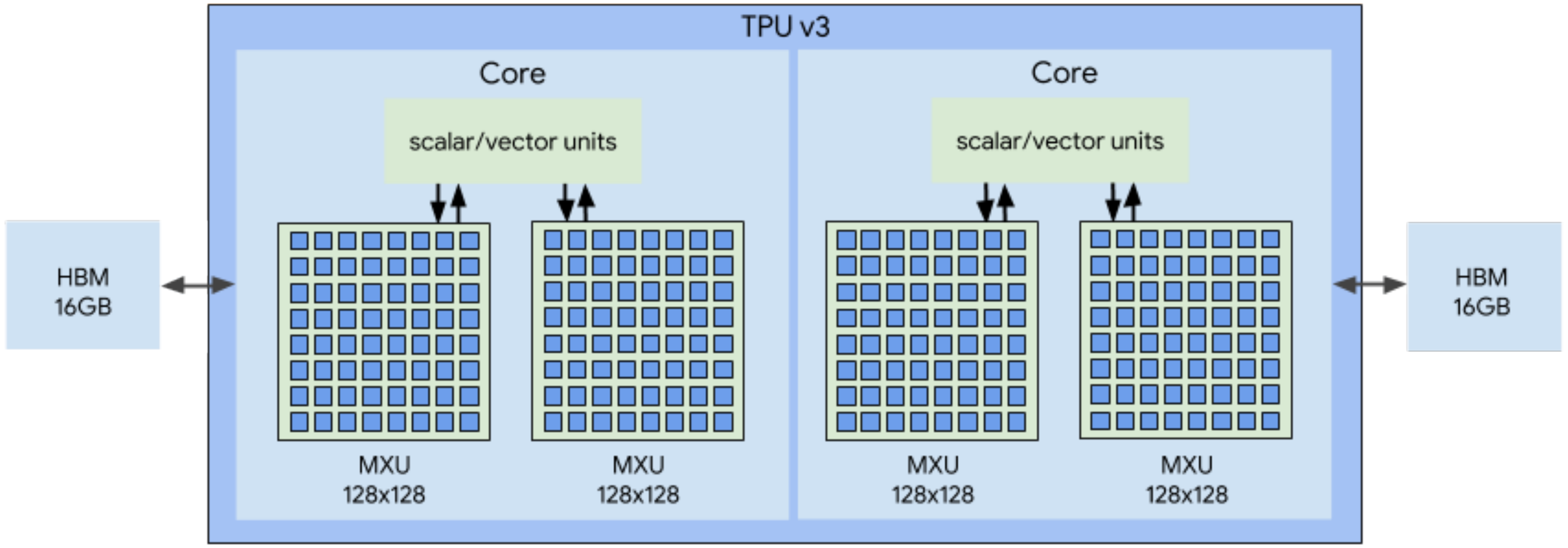
- 张量运算单元 (Tensor Processing Units, TPU)

	NVIDIA V100	NVIDIA A100	TPU v2	TPU v3	TPU v4
Hardware					
Arch	NVIDIA Volta	NVIDIA Ampere	Google Cloud TPU	Google Cloud TPU	Google Cloud TPU
Memory	16GB / 32GB	40GB / 80GB	64GB	128GB	?
FLOPS	Double: 7 TFLOPS Single: 14 TFLOPS DL: 112 TFLOPS	Double: 9.7 TFLOPS Single: 19.5 TFLOPS DL: 156 TFLOPS	180 TFLOPS	420 TFLOPS	2.7x over TPU v3

Google Cloud TPU. *https://cloud.google.com/tpu*

BERT

- **TPU v3**
    - 1 hardware = 4 chips = 8 cores = 128GB HBM
    - 每小时价格：8.0 USD（独占）or 2.4 USD（抢占）



Google Cloud TPU. https://cloud.google.com/tpu

BERT

- 训这样一个模型需要多少钱？（以BERT-large为例）

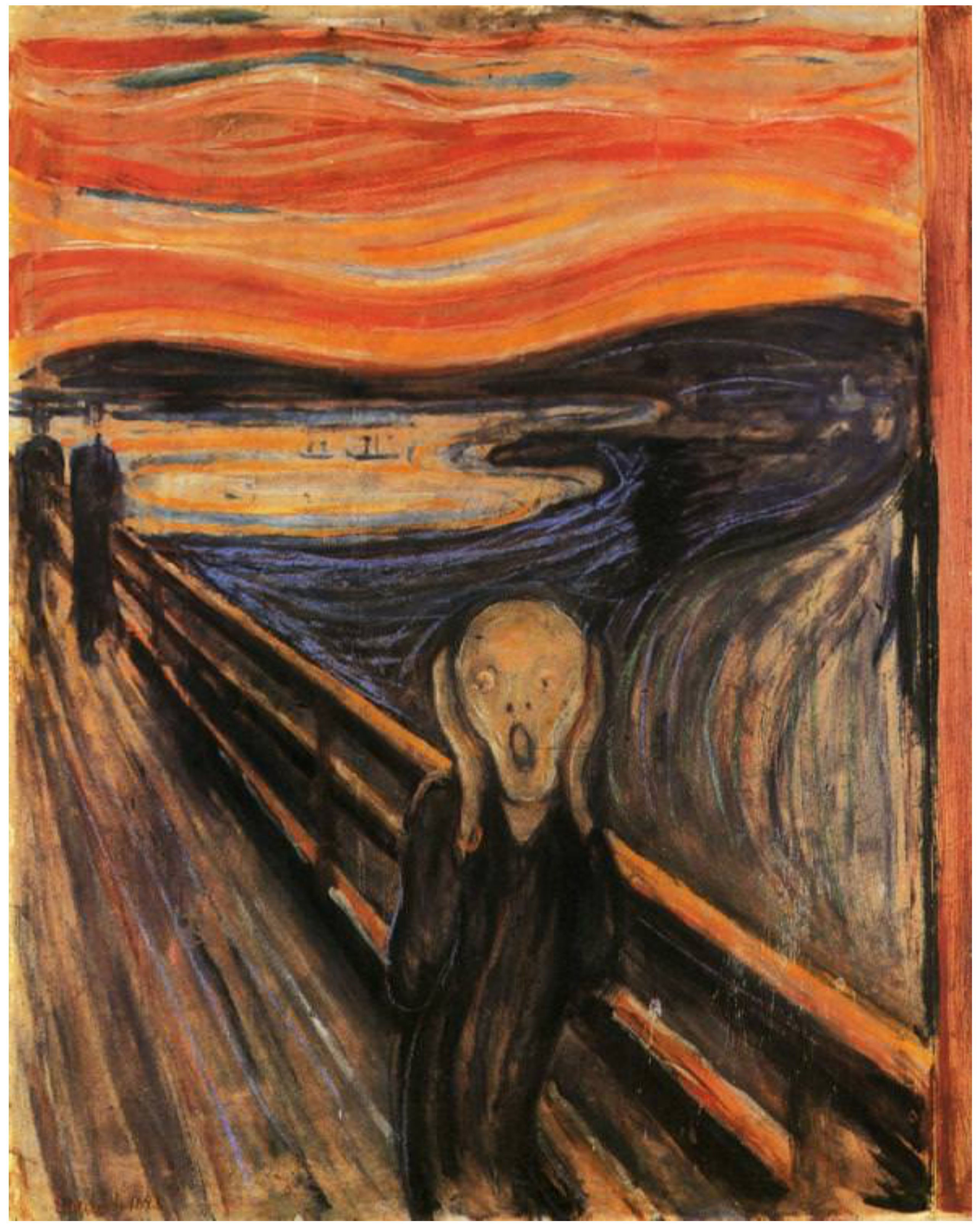
16 Cloud TPUs = 16 * 4.5 = 72 USD / hour

一天消耗 = 72 * 24 = 1,728 USD

四天消耗 = 1,728 USD * 4 = 6,912 USD

6,912 USD ≈ 44,238 CNY

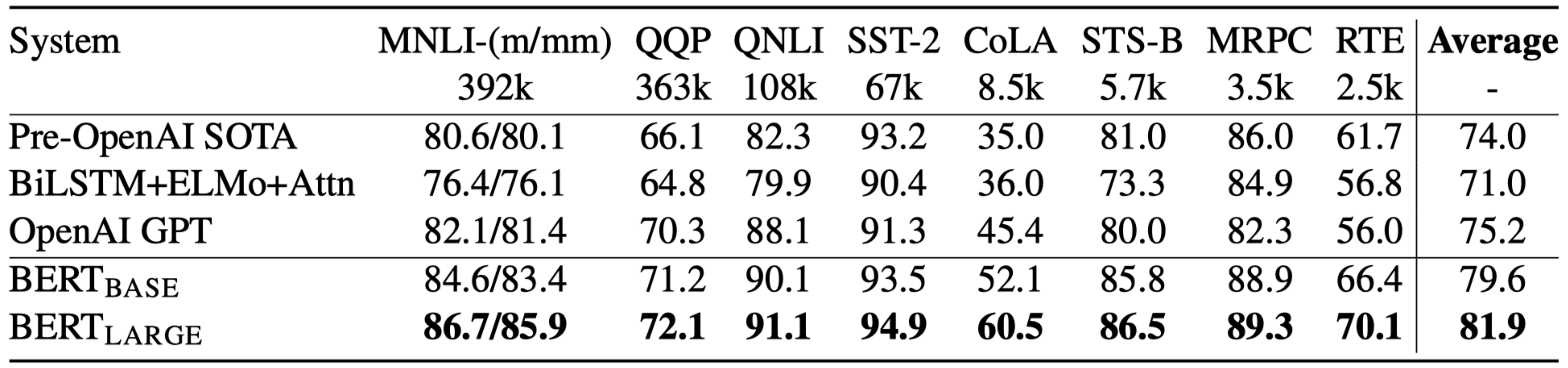
早期训练预训练模型的成本较高
但目前一些机构采用“二次预训练”的方法
大幅降低了训练成本



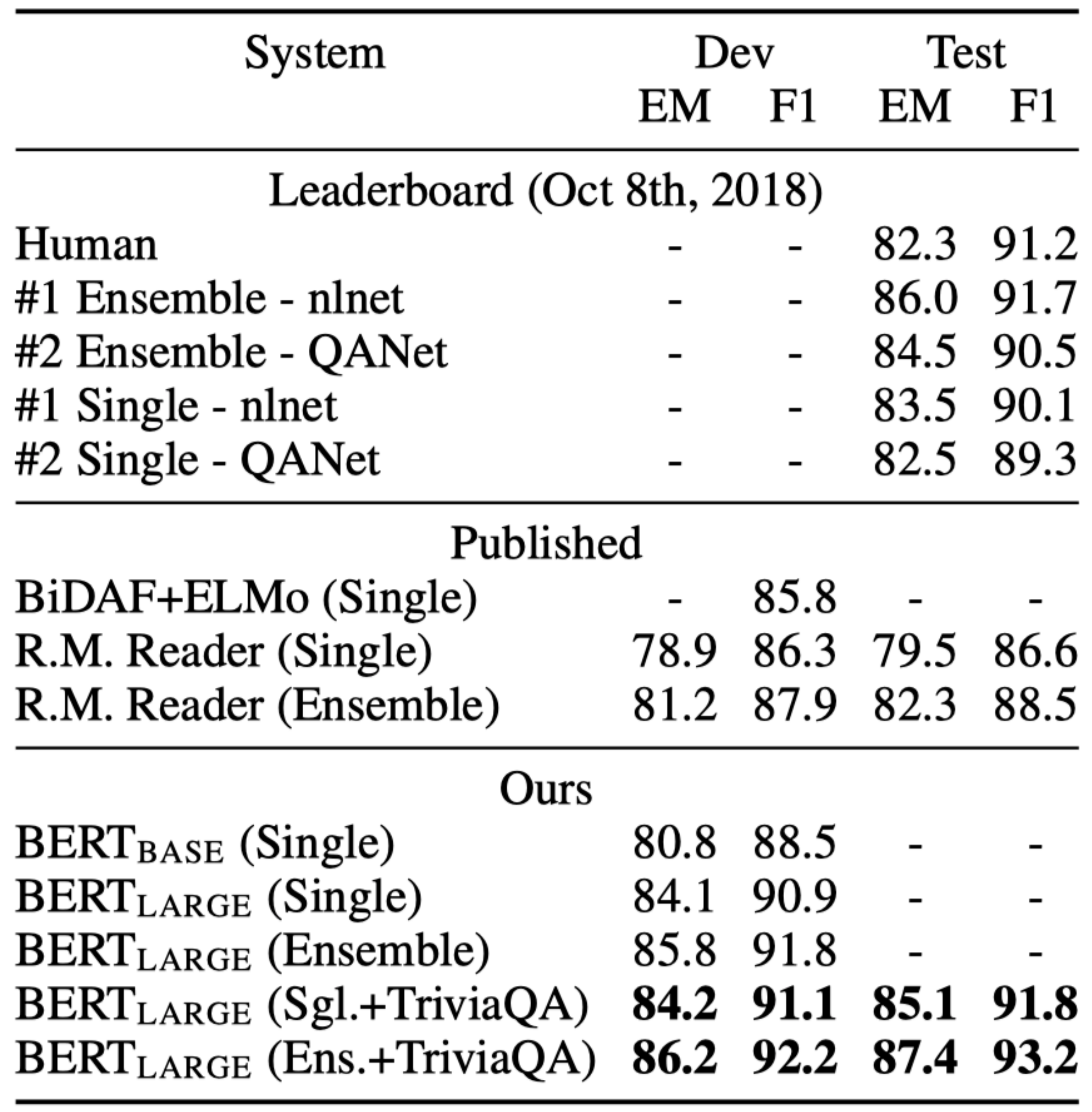
Devlin et al., NAACL 2019. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*

BERT

- **实验结果**
  - 在GLUE和SQuAD等任务上相比GPT获得更优的实验结果



System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
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	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
BERT$_{\text{BASE}}$	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
BERT$_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**91.1**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**81.9**



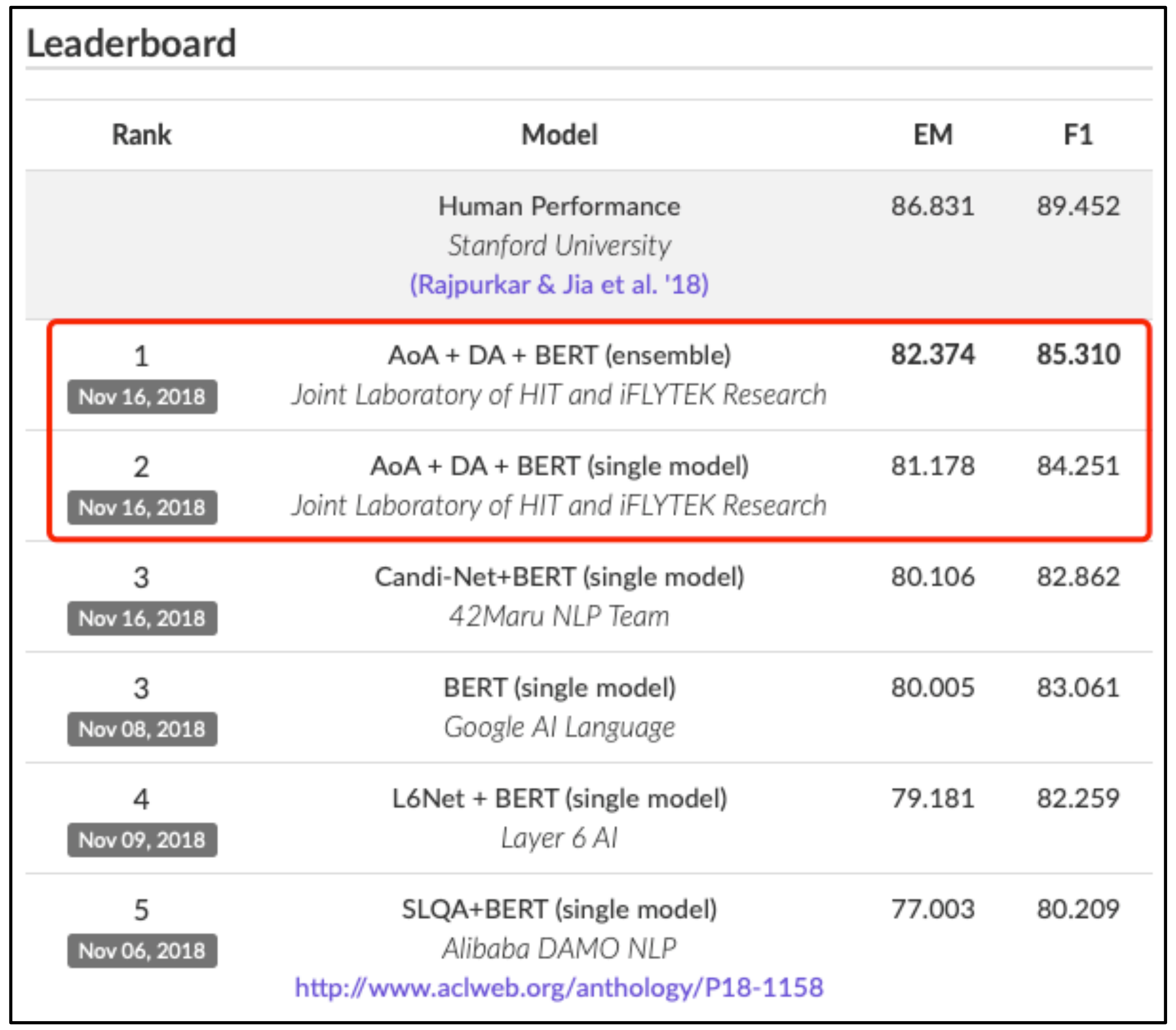
System	Dev		Test	
	EM	F1	EM	F1
Leaderboard (Oct 8th, 2018)				
Human	-	-	82.3	91.2
#1 Ensemble - nlnet	-	-	86.0	91.7
#2 Ensemble - QANet	-	-	84.5	90.5
#1 Single - nlnet	-	-	83.5	90.1
#2 Single - QANet	-	-	82.5	89.3
Published				
BiDAF+ELMo (Single)	-	85.8	-	-
R.M. Reader (Single)	78.9	86.3	79.5	86.6
R.M. Reader (Ensemble)	81.2	87.9	82.3	88.5
Ours				
BERT$_{\text{BASE}}$ (Single)	80.8	88.5	-	-
BERT$_{\text{LARGE}}$ (Single)	84.1	90.9	-	-
BERT$_{\text{LARGE}}$ (Ensemble)	85.8	91.8	-	-
BERT$_{\text{LARGE}}$ (Sgl.+TriviaQA)	**84.2**	**91.1**	**85.1**	**91.8**
BERT$_{\text{LARGE}}$ (Ens.+TriviaQA)	**86.2**	**92.2**	**87.4**	**93.2**

Devlin et al., NAACL 2019. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*

BERT

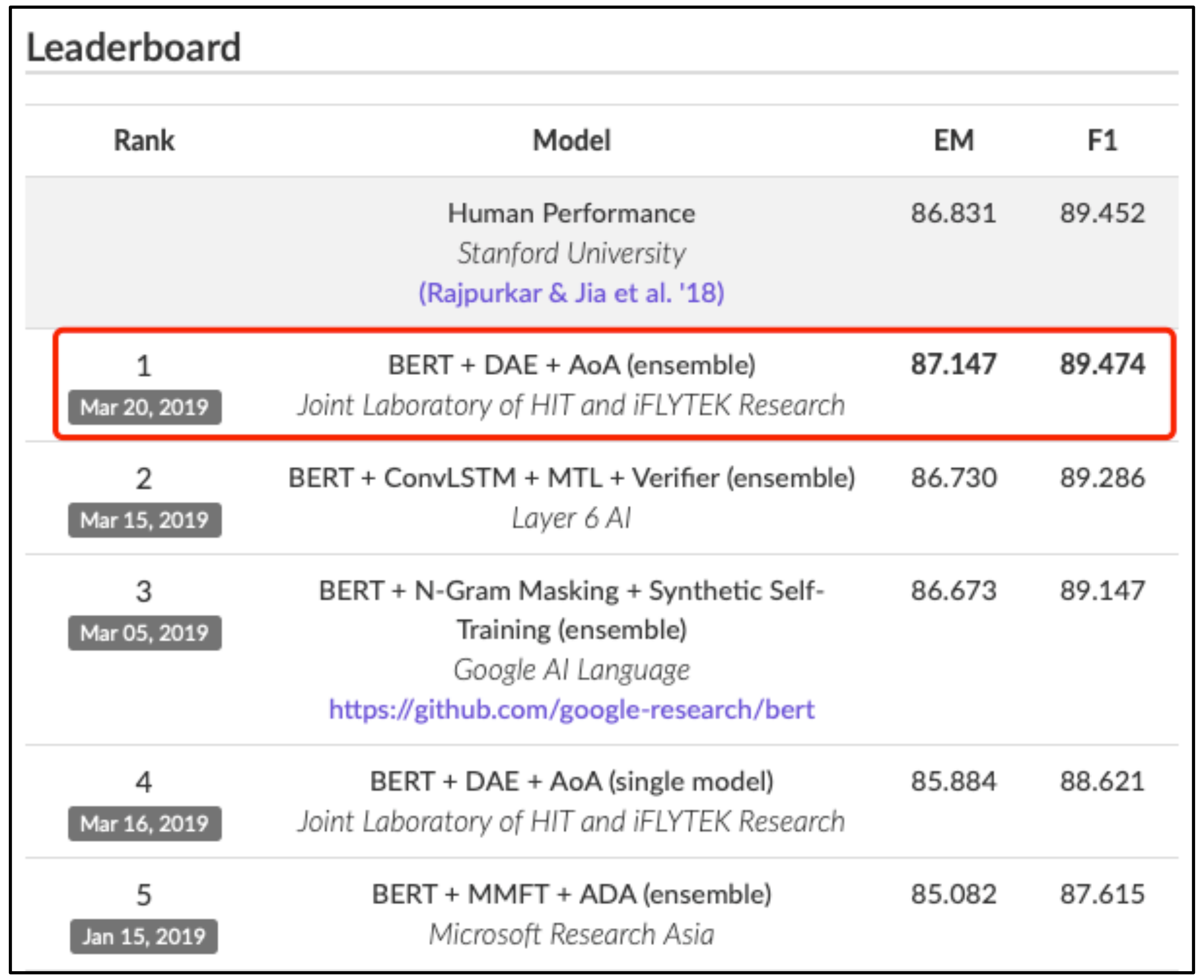
让世界聆听我们的声音

- **BERT + DAE + AoA** (by HFL)
  - 在国际权威机器阅读理解评测SQuAD 2.0中，全球首次超过人类平均水平



Leaderboard

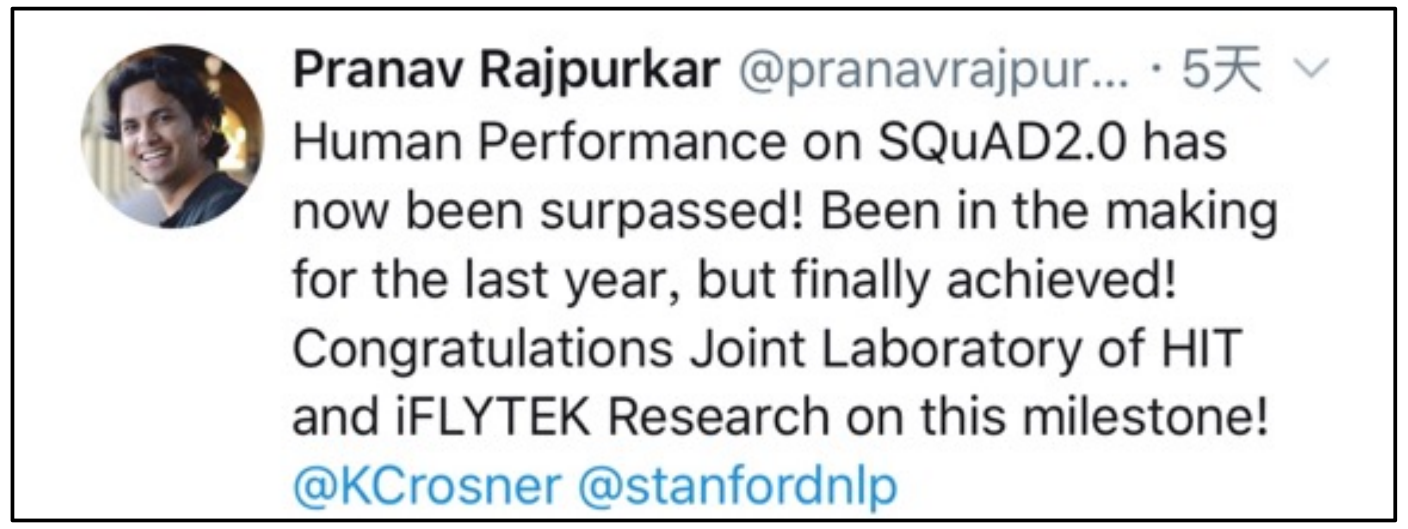
Rank	Model	EM	F1
	Human Performance *Stanford University* (Rajpurkar & Jia et al. '18)	86.831	89.452
1 Nov 16, 2018	AoA + DA + BERT (ensemble) *Joint Laboratory of HIT and iFLYTEK Research*	**82.374**	**85.310**
2 Nov 16, 2018	AoA + DA + BERT (single model) *Joint Laboratory of HIT and iFLYTEK Research*	81.178	84.251
3 Nov 16, 2018	Candi-Net+BERT (single model) *42Maru NLP Team*	80.106	82.862
3 Nov 08, 2018	BERT (single model) *Google AI Language*	80.005	83.061
4 Nov 09, 2018	L6Net + BERT (single model) *Layer 6 AI*	79.181	82.259
5 Nov 06, 2018	SLQA+BERT (single model) *Alibaba DAMO NLP* http://www.aclweb.org/anthology/P18-1158	77.003	80.209



Leaderboard

Rank	Model	EM	F1
	Human Performance *Stanford University* (Rajpurkar & Jia et al. '18)	86.831	89.452
1 Mar 20, 2019	BERT + DAE + AoA (ensemble) *Joint Laboratory of HIT and iFLYTEK Research*	**87.147**	**89.474**
2 Mar 15, 2019	BERT + ConvLSTM + MTL + Verifier (ensemble) *Layer 6 AI*	86.730	89.286
3 Mar 05, 2019	BERT + N-Gram Masking + Synthetic Self-Training (ensemble) *Google AI Language* https://github.com/google-research/bert	86.673	89.147
4 Mar 16, 2019	BERT + DAE + AoA (single model) *Joint Laboratory of HIT and iFLYTEK Research*	85.884	88.621
5 Jan 15, 2019	BERT + MMFT + ADA (ensemble) *Microsoft Research Asia*	85.082	87.615

SQuAD2.0
The Stanford Question Answering Dataset



Pranav Rajpurkar @pranavrajpur... · 5天
Human Performance on SQuAD2.0 has now been surpassed! Been in the making for the last year, but finally achieved! Congratulations Joint Laboratory of HIT and iFLYTEK Research on this milestone! @KCrosner @stanfordnlp

科大讯飞 iFLYTEK

BERT-wwm

- **基于整词掩码（Whole Word Masking）的BERT**
  - MLM：随机选取一定比例的WordPiece**子词**
  - WWM：随机选取一定比例的**整词**，属于同一个整词的WordPiece子词均被掩码
  - 总掩码数量不变，变动的是掩码位置的选取

原始句子	the man jumped up , put his basket on phil ##am ##mon ' s head
MLM掩码方法	[M] man [M] up , put his [M] on phil [M] ##mon ' s head
WWM掩码方法	the man [M] up , put his basket on [M] [M] [M] ' s head

BERT-wwm

- **关于Whole Word Masking的重要提示**
  - “掩码”是一个广义概念，并不只是代表“替换为 [MASK] 符号”
  - “掩码”包含三种随机操作：替换为 [MASK]、替换为随机词、保持原词

原句： there is an apple tree nearby.	
分词后： ["there", "is", "an", "ap", "##p", "##le", "tr", "##ee", "nearby", "."]	
MLM	there [MASK] an ap [MASK] ##le tr [RANDOM] nearby . [MASK] [MASK] an ap ##p [MASK] tr ##ee nearby . there is [MASK] ap ##p ##le [MASK] ##ee [MASK] . there is [MASK] ap [MASK] ##le tr ##ee nearby [MASK] . there is an! ap ##p ##le tr [MASK] nearby [MASK] . there is an [MASK] ##p [MASK] tr ##ee nearby [MASK] .
WWM	there is an [MASK] [MASK] [RANDOM] tr ##ee nearby . there is! [MASK] ap ##p ##le tr ##ee nearby [MASK] . there is [MASK] ap ##p ##le [MASK] [MASK] nearby . there [MASK] [MASK] ap ##p ##le tr ##ee [RANDOM] . there is an ap ##p ##le [MASK] [MASK] nearby [MASK] .

BERT-wwm

- **实验结果**
  - 相比传统MLM方法有显著性能提升

英文任务

Model	SQuAD 1.1 (F1/EM)	Multi NLI (Acc)
BERT-Large, Uncased (Original)	91.0 / 84.3	86.05
BERT-Large, Uncased (WWM)	92.8 / 86.7	87.07
BERT-Large, Cased (Original)	91.5 / 84.8	86.09
BERT-Large, Cased (WWM)	92.9 / 86.7	86.46

中文任务

Model	CMRC 2018 (F1/EM)	XNLI (Acc)
BERT-base	84.5 / 65.5	77.8
BERT-wwm (base)	85.6 / 66.3	79.0
BERT-wwm-ext (base)	85.7 / 67.1	79.4

N-gram Masking

- **基于N元文法掩码（N-gram Masking）的BERT**
  - 对一个连续的N-gram单元进行掩码，进一步增加MLM任务的难度
  - 难度：N-gram Masking > Whole Word Masking > vanilla MLM

We went to the store to buy some fruits.

→ We went to [M] store to [M] some [M].

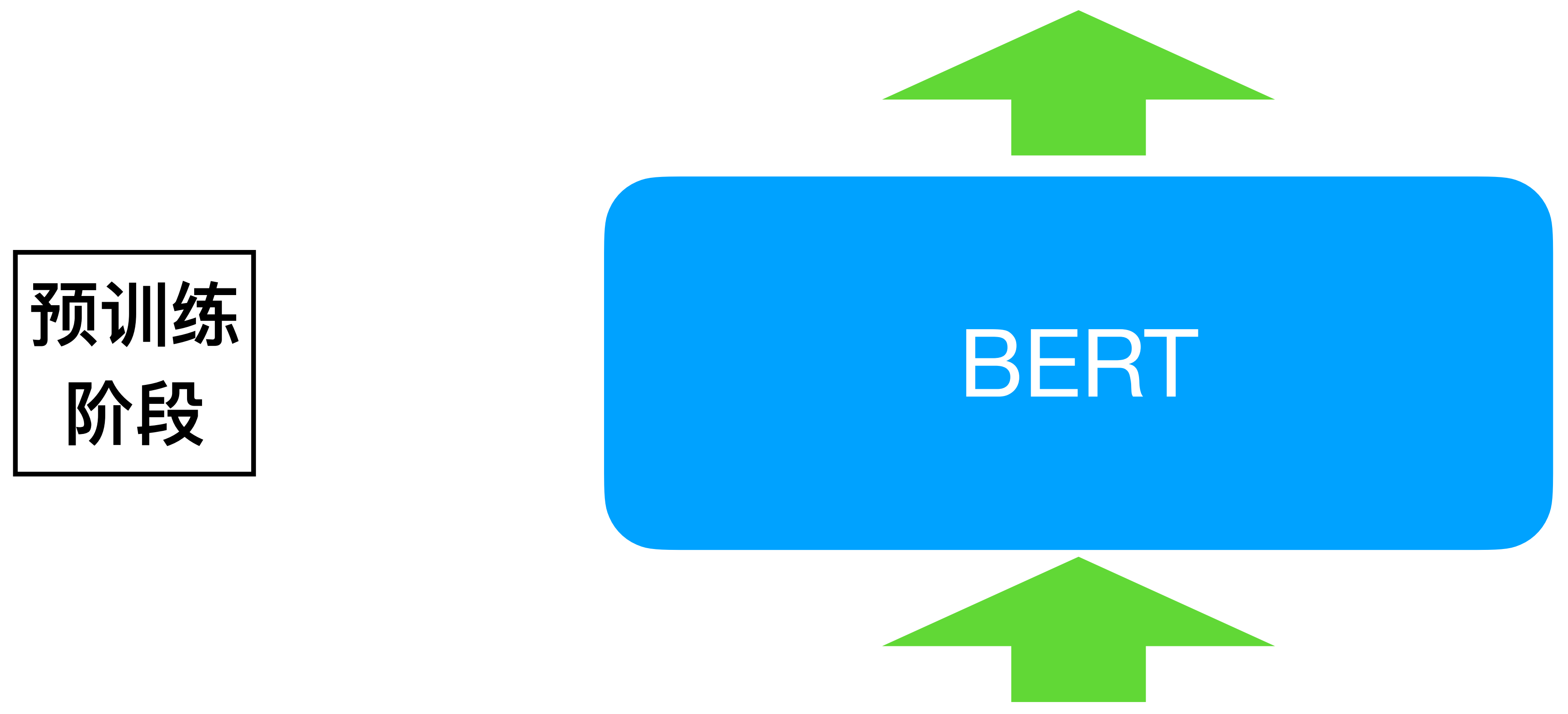
Original MLM

→ We went to [M] [M] [M] buy some fruits.

N-gram Masking

N-gram Masking

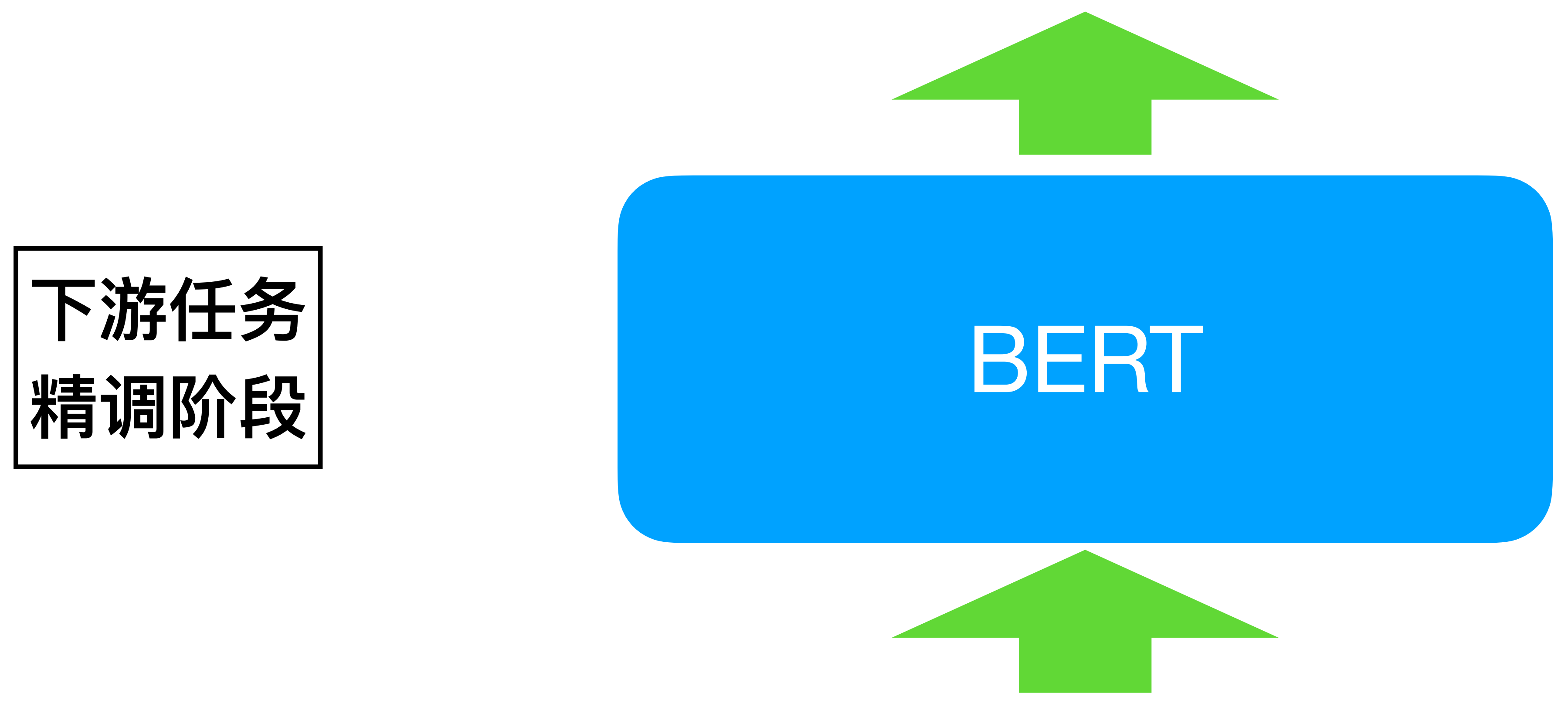
- **关于WWM、NM的重要提示**
  - MLM、WWM、NM只影响预训练阶段的输入形式
  - 不论使用任意一种MLM、WWM、NM训练出的BERT，**下游精调时的输入均相同**



NM: 我 [M] [M] [M] 天 安 门

WWM: [M] 爱 [M] [M] 天 安 门

MLM: 我 [M] [M] 京 天 [M] 门



✓ MLM/WWM/NM: 我 爱 北 京 天 安 门

❎ MLM/WWM/NM: 我 爱 北京 天安门

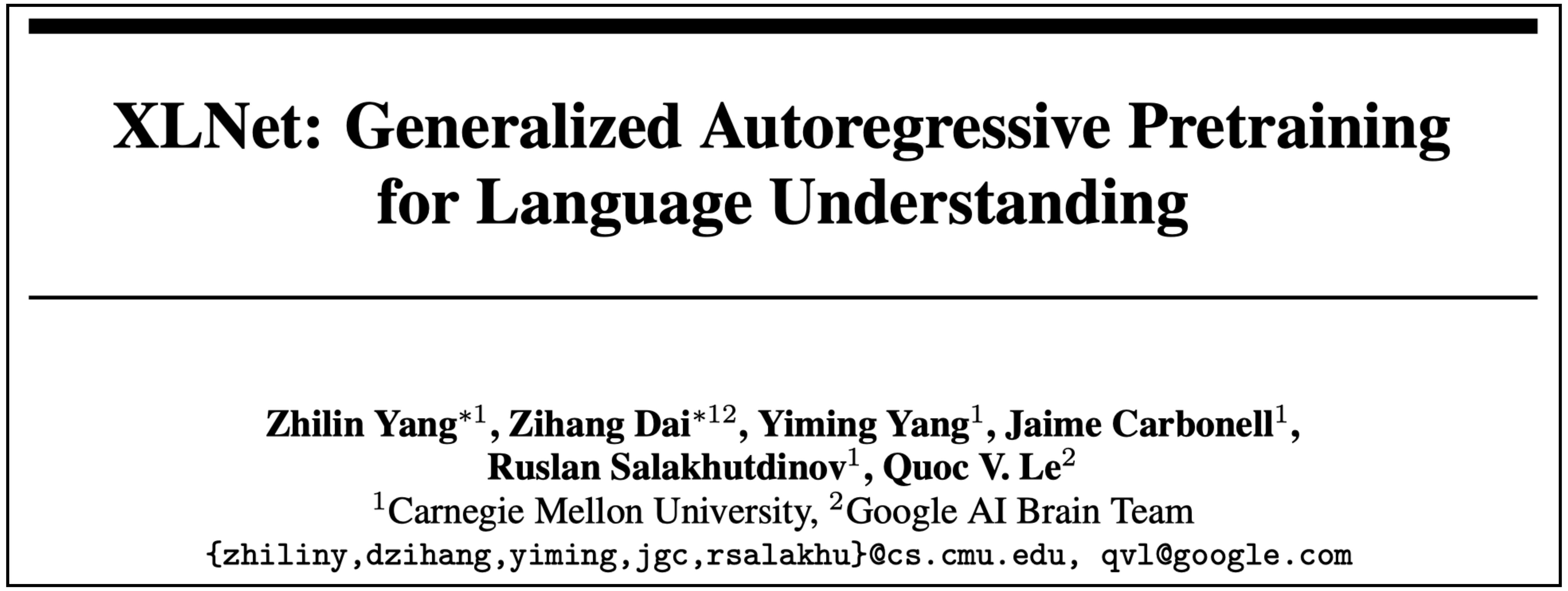
预训练语言模型进阶

ADVANCED PRE-TRAINED LANGUAGE MODELS

1. XLNet
2. RoBERTa
3. ALBERT
4. ELECTRA
5. MacBERT
6. T5

XLNet

- **XLNet:** Transformer-**XL Net**
  - 提出了一种可以捕获双向上下文的基于自回归的语言建模方法
  - 解决了BERT中存在的“**预训练-精调**”不一致的问题



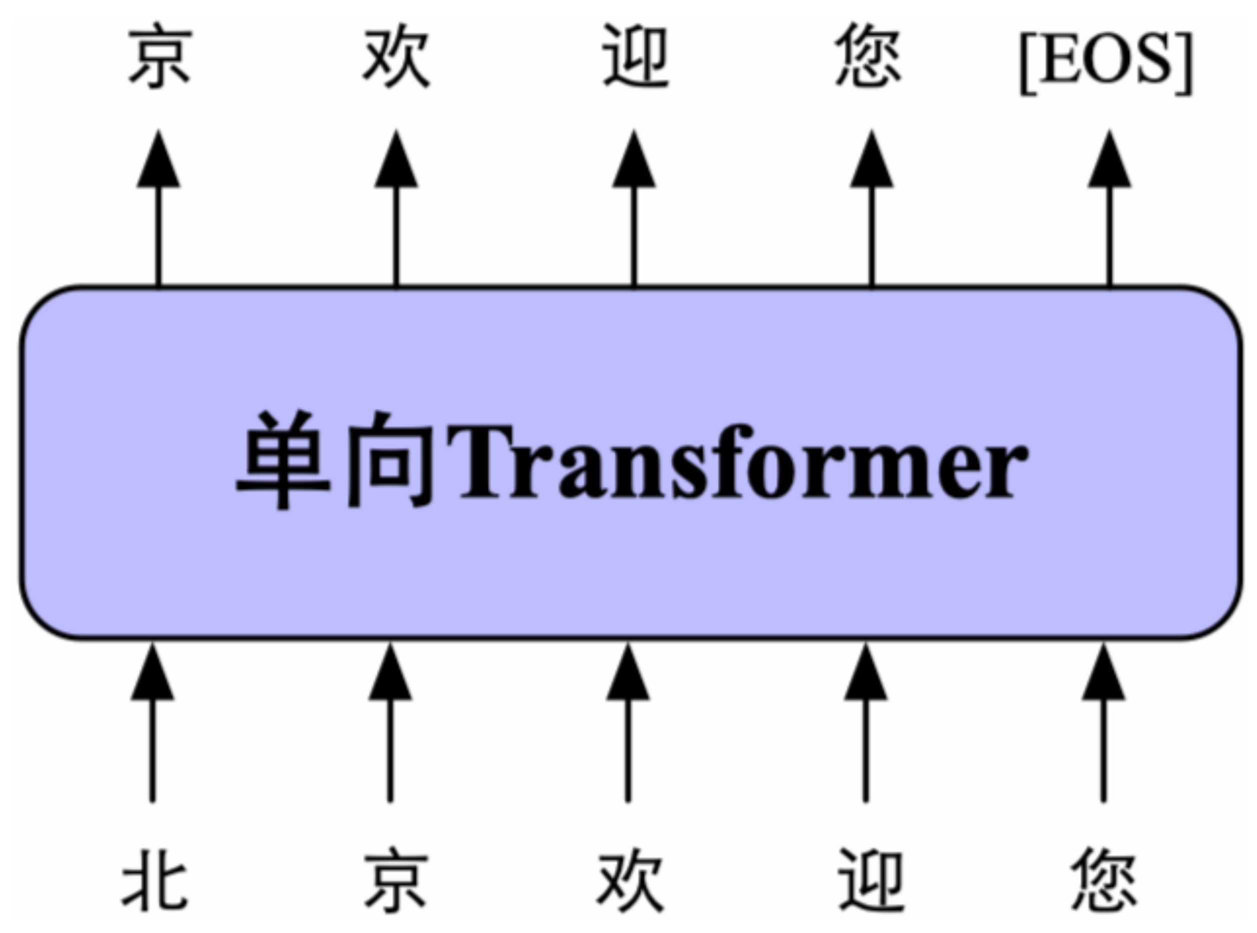
XLNet: Generalized Autoregressive Pretraining for Language Understanding

Zhilin Yang*[1], **Zihang Dai***[1,2], **Yiming Yang**[1], **Jaime Carbonell**[1], **Ruslan Salakhutdinov**[1], **Quoc V. Le**[2]
[1]Carnegie Mellon University, [2]Google AI Brain Team
{zhiliny,dzihang,yiming,jgc,rsalakhu}@cs.cmu.edu, qvl@google.com

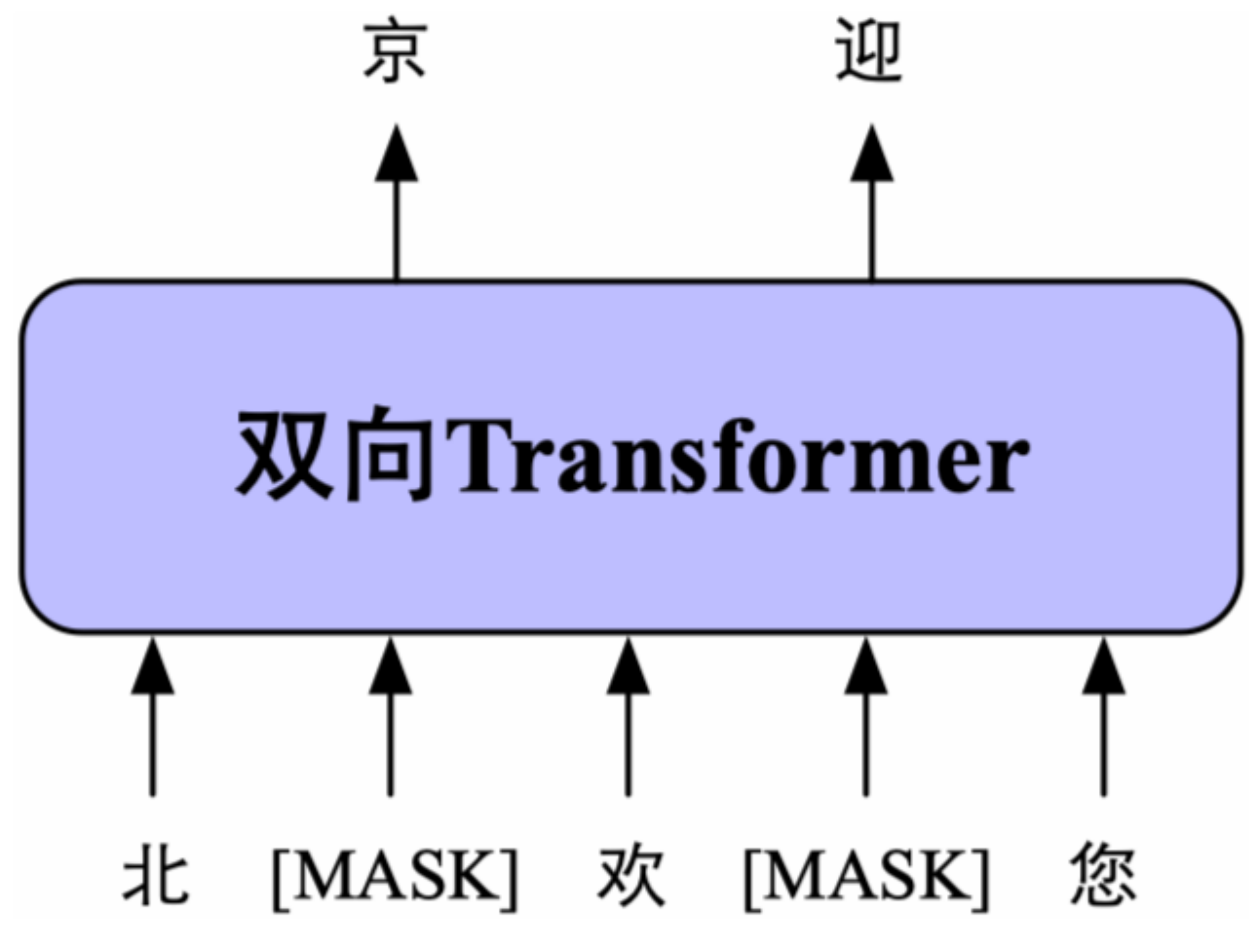
Yang et al., NeurIPS 2020. *XLNet: Generalized Autoregressive Pretraining for Language Understanding*

XLNet

- **两种典型的语言建模方法**
  - 自回归语言模型：利用输入文本的**单向**历史预测下一个单词
  - 自编码语言模型：利用输入文本的**双向**上下文预测缺失位置的单词



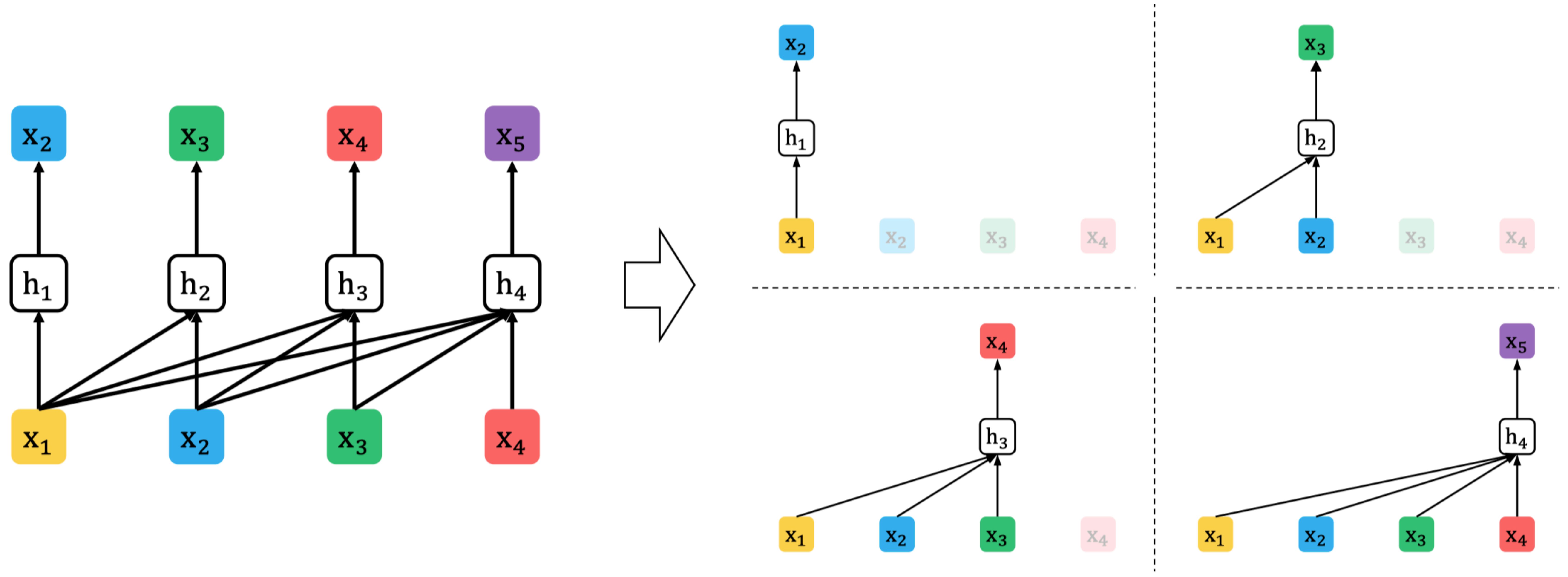
(a) 自回归语言模型



(b) 自编码语言模型

XLNet

- **基于自回归的语言模型**
  - 从左至右的建模顺序：1→2→3→4
  - $P(\mathbf{x}) = P(x_1)P(x_2 \mid \mathbf{x}_1)P(x_3 \mid \mathbf{x}_{1,2})P(x_4 \mid \mathbf{x}_{1,2,3})\ldots$



Yang et al., NeurIPS 2020. XLNet: Generalized Autoregressive Pretraining for Language Understanding

XLNet

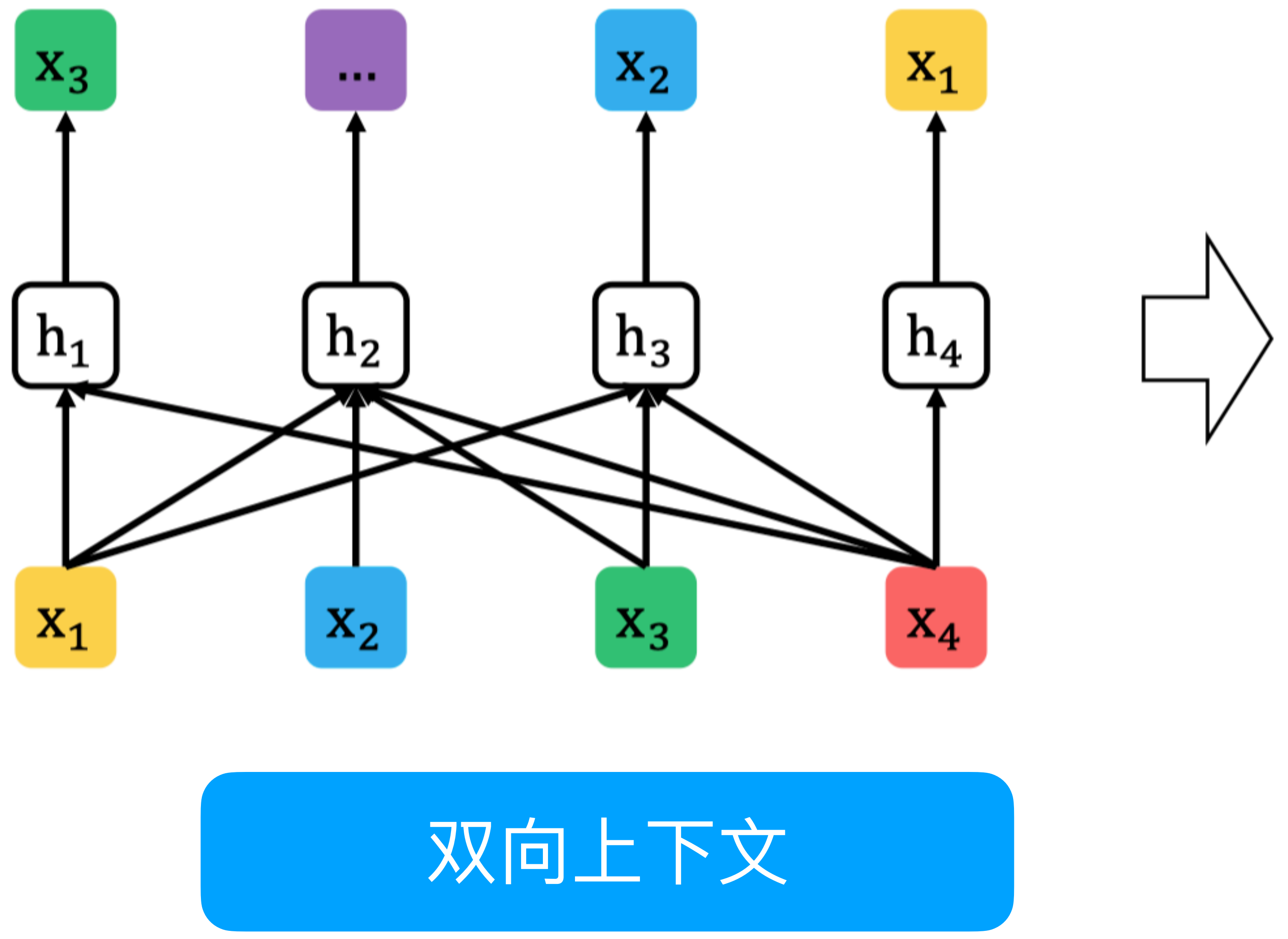
- **排列语言模型（Permutation Language Model）**
  - 给定一个长度为T 的序列 $\mathbf{x}$
  - 从所有可能的排列当中均匀采样一种排列顺序（factorization order）记为 $\mathbf{z}$
  - 最大化对数似然函数

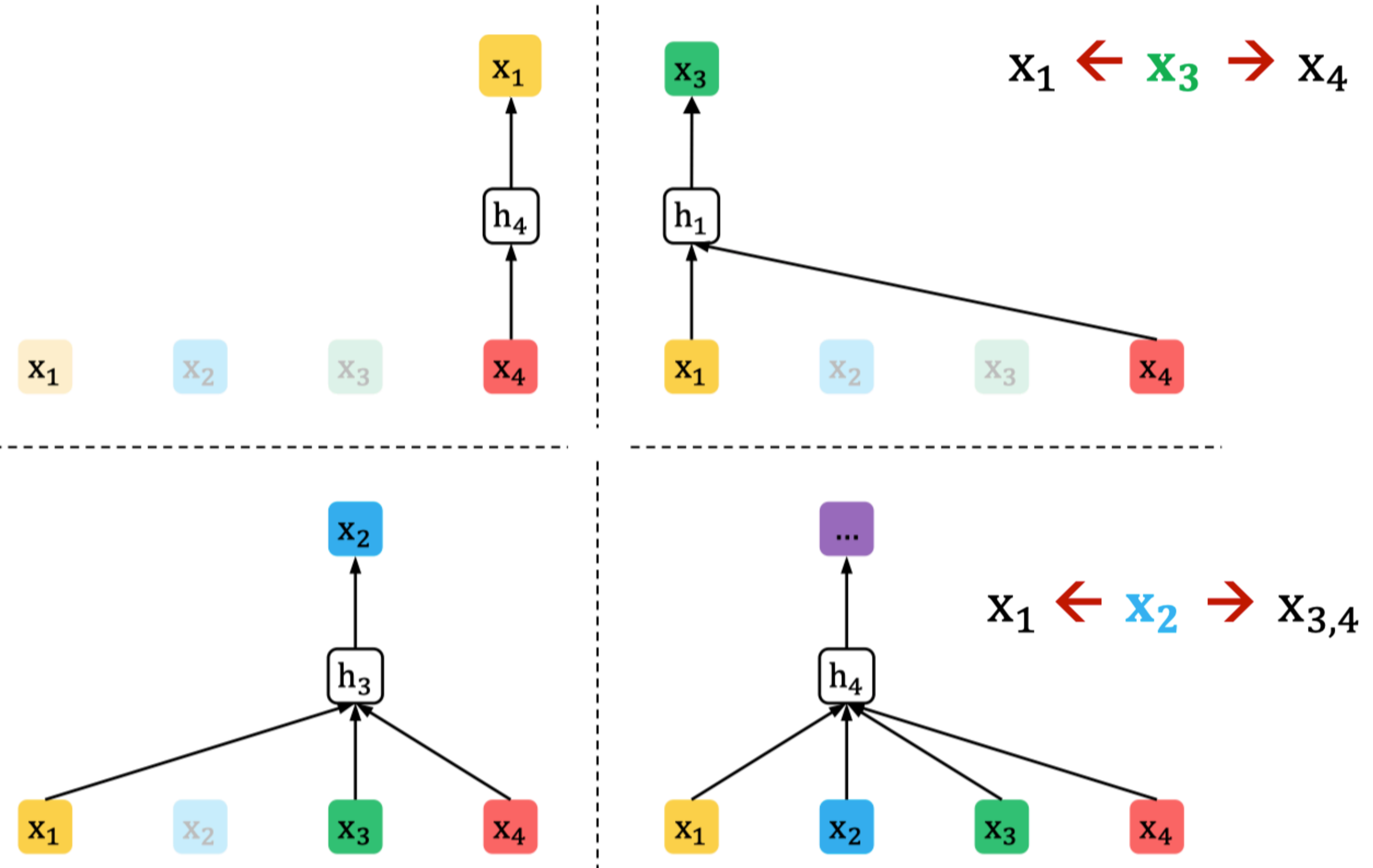
$$\mathbb{E}_{\mathbf{z}\sim\mathcal{Z}_T}\left[\log P(\mathbf{x} \mid \mathbf{z})\right] = \mathbb{E}_{\mathbf{z}\sim\mathcal{Z}_T}\left[\sum_{t=1}^{T} P(x_{z_t} \mid \mathbf{x}_{\mathbf{z}_{<t}}, z_t)\right]$$

Yang et al., NeurIPS 2020. *XLNet: Generalized Autoregressive Pretraining for Language Understanding*

XLNet

- **Permutation Language Model**
  - 改变建模顺序：4→1→3→2
  - $P(\mathbf{x}) = P(x_4)P(x_1 \mid \mathbf{x}_4)P(x_3 \mid \mathbf{x}_{1,4})P(x_2 \mid \mathbf{x}_{1,3,4})\ldots$





Yang et al., NeurIPS 2020. *XLNet: Generalized Autoregressive Pretraining for Language Understanding*

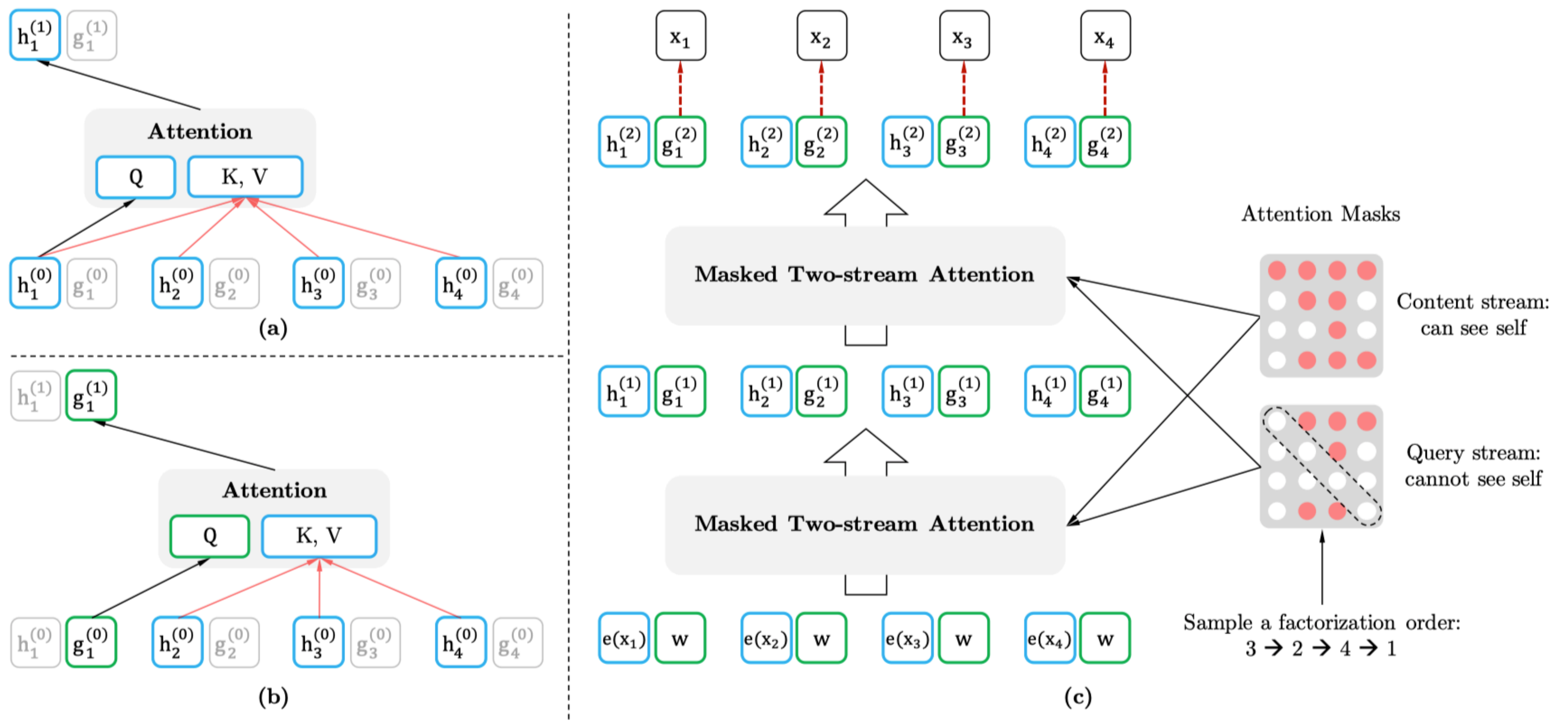
XLNet

- 双流注意力机制

当前预测	当前历史	内容流置一元素	查询流置一元素
$3 \rightarrow 2 \rightarrow 4 \rightarrow 1$	$\varnothing$	$M^{h}_{3,3}$	$\varnothing$
$3 \rightarrow 2 \rightarrow 4 \rightarrow 1$	x_3	$M^{h}_{2,3}, M^{h}_{2,2}$	$M^{g}_{2,3}$
$3 \rightarrow 2 \rightarrow 4 \rightarrow 1$	x_3, x_2	$M^{h}_{4,3}, M^{h}_{4,2}, M^{h}_{4,4}$	$M^{g}_{4,3}, M^{g}_{4,2}$
$3 \rightarrow 2 \rightarrow 4 \rightarrow 1$	x_3, x_2, x_4	$M^{h}_{1,3}, M^{h}_{1,2}, M^{h}_{1,4}, M^{h}_{1,1}$	$M^{g}_{1,3}, M^{g}_{1,2}, M^{g}_{1,4}$

内容流注意力
（Content Stream Attention）

查询流注意力
（Query Stream Attention）



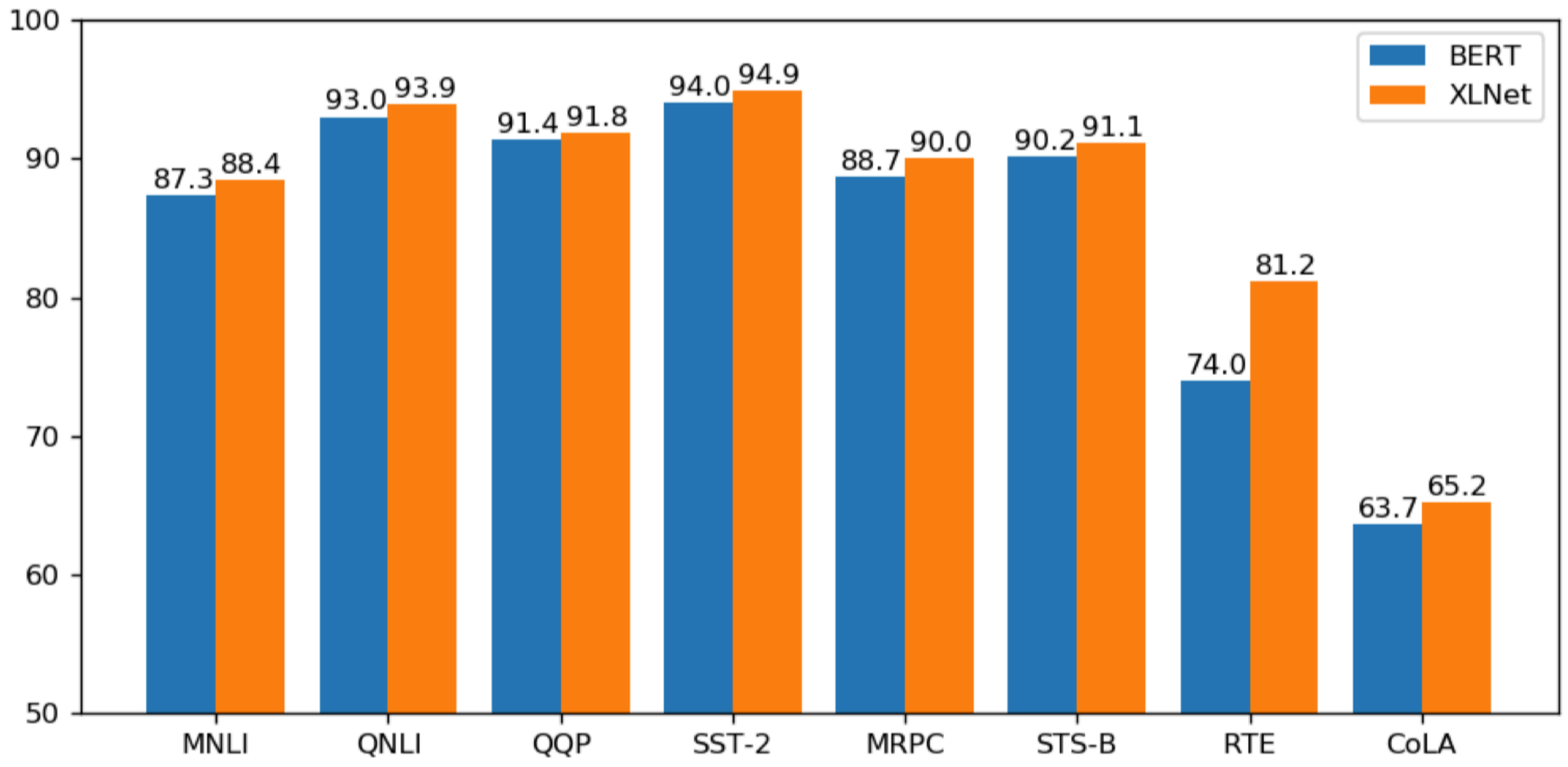
Attention
Q
K, V
(a)
(b)
(c)
Masked Two-stream Attention
Attention Masks
Content stream: can see self
Query stream: cannot see self
Sample a factorization order: 3 → 2 → 4 → 1

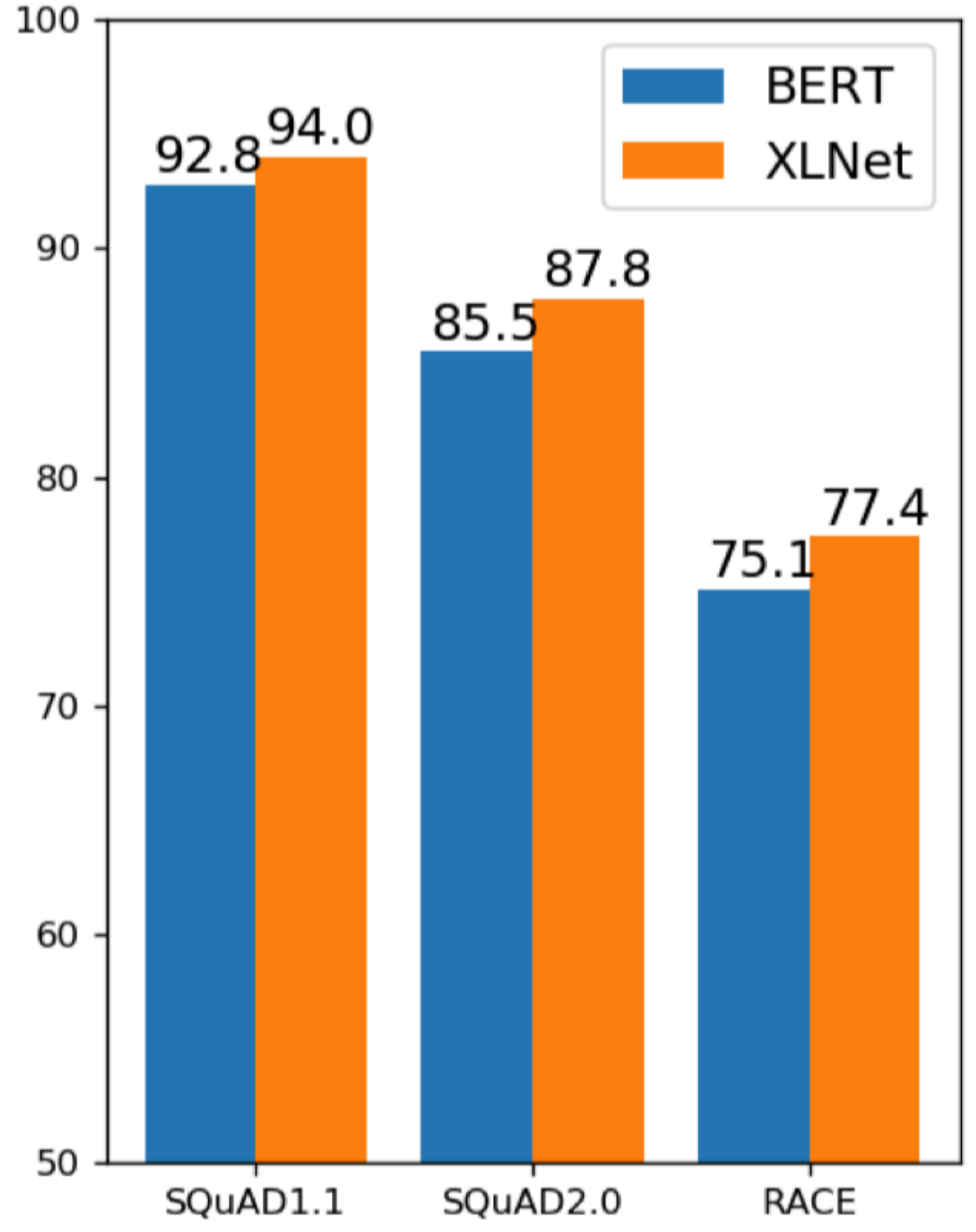
Yang et al., NeurIPS 2020. XLNet: Generalized Autoregressive Pretraining for Language Understanding

XLNet

- **实验结果**
  - 在可比的实验条件下，XLNet获得了优于BERT的实验效果

扩展阅读

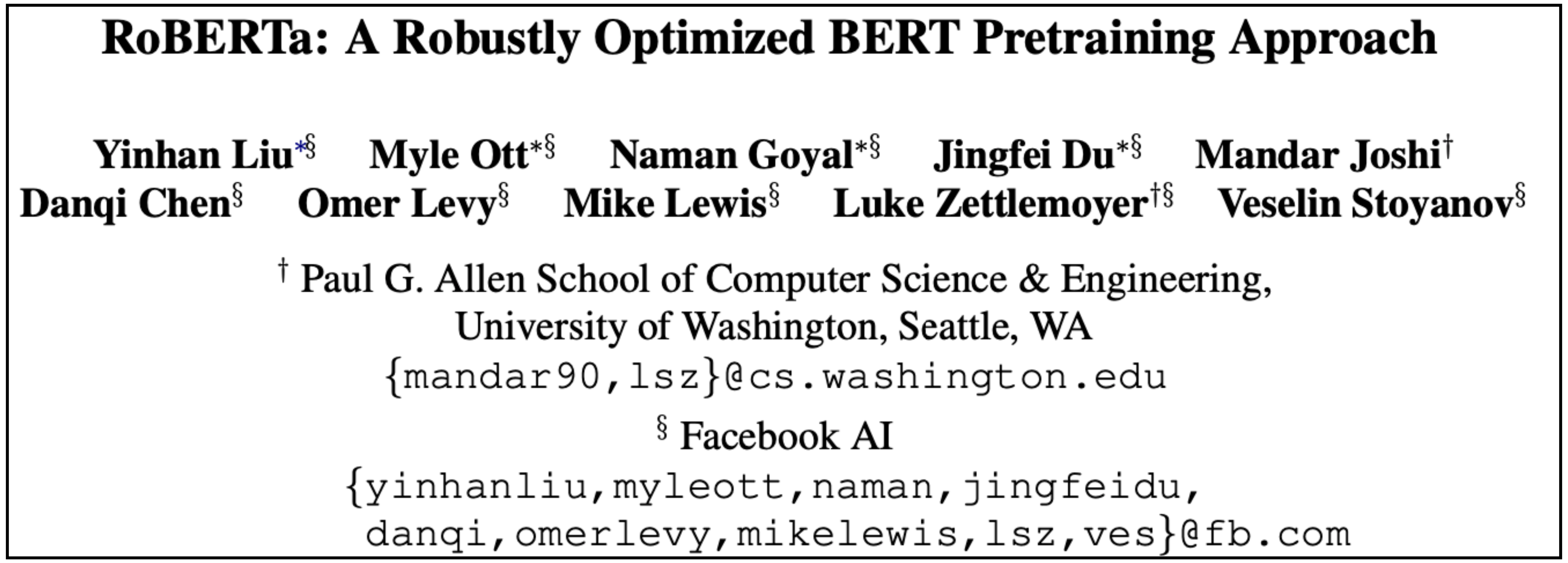




Yang et al., NeurIPS 2020. *XLNet: Generalized Autoregressive Pretraining for Language Understanding*

RoBERTa

- **RoBERTa: R**obustly **o**ptimized **BERT** pretraining **a**pproach
  - 探究了BERT的各个设计环节，包括掩码策略、NSP的有效性、训练步数等
  - 提出CC-News数据集，并且证明使用更多数据可以进一步提升预训练模型效果



RoBERTa: A Robustly Optimized BERT Pretraining Approach

Yinhan Liu[*§] **Myle Ott**[*§] **Naman Goyal**[*§] **Jingfei Du**[*§] **Mandar Joshi**[†]
Danqi Chen[§] **Omer Levy**[§] **Mike Lewis**[§] **Luke Zettlemoyer**[†§] **Veselin Stoyanov**[§]

[†] Paul G. Allen School of Computer Science & Engineering,
University of Washington, Seattle, WA
{mandar90,lsz}@cs.washington.edu

[§] Facebook AI
{yinhanliu,myleott,naman,jingfeidu,
danqi,omerlevy,mikelewis,lsz,ves}@fb.com

Liu et al., arXiv 2019. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*

RoBERTa

- **静态掩码 vs. 动态掩码**
  - 目的：增加掩码语言模型中的随机性，提高文本的利用度
  - 静态掩码：模型训练**之前**（数据预处理）阶段决定哪些词被mask，即BERT使用的方法
  - 动态掩码：模型训练的**过程中**决定哪些词被mask

轮数

```
went to the store → went to the [MASK]
went to the store → went to the [MASK]
went to the store → went to the [MASK]
went to the store → went to the [MASK]
went to the store → went to the [MASK]
```

静态掩码

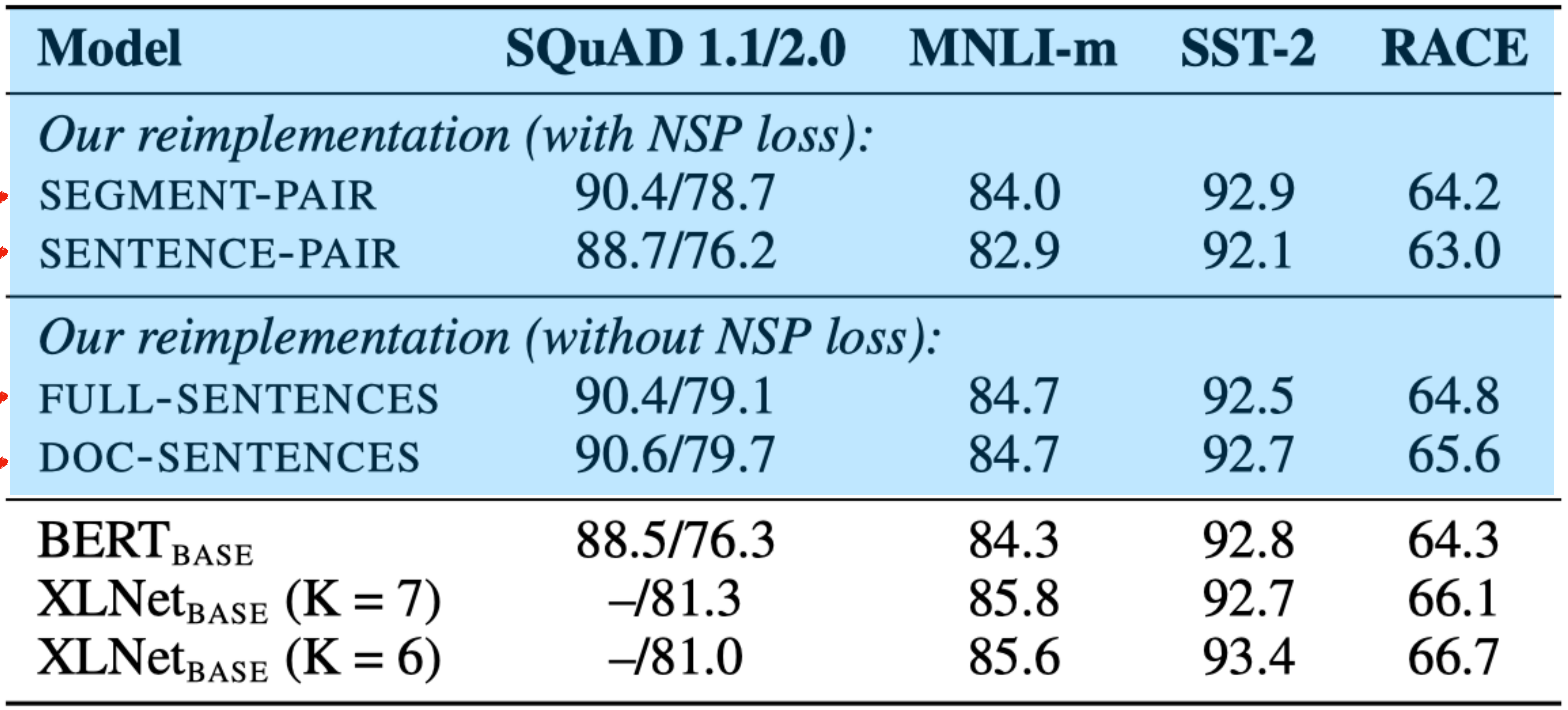
```
went to the store → went to the [MASK]
went to the store → went to [MASK] store
went to the store → go to the store
went to the store → went to the store
went to the store → went [MASK] the store
```

动态掩码

RoBERTa

- **NSP预训练任务是否是必要的?**
  - 实验结果表明，舍弃NSP任务可以获得微弱性能提升

Original BERT implementation → SEGMENT-PAIR
Natural sentences → SENTENCE-PAIR
Could cross the document boundary → FULL-SENTENCES
Could NOT cross the document boundary → DOC-SENTENCES



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

RoBERTa

- **使用更大的批次大小以及更多数据**
  - 适当增加预训练步数，可以进一步提升预训练效果
  - 目前广泛被认可的实证结论：训练预训练模型的关键之一是选用大的批次大小 💡

batch size	learning rate	epochs	steps	perplexity	MNLI-m	SST-2
256	1e-4	32	1M	3.99	84.7	92.5
2K	7e-4	32	125K	3.68	85.2	93.1
		64	250K	3.59	85.3	**94.1**
		128	500K	3.51	85.4	93.5
8K	1e-3	32	31K	3.77	84.4	93.2
		64	63K	3.60	85.3	93.5
		128	125K	**3.50**	**85.8**	**94.1**

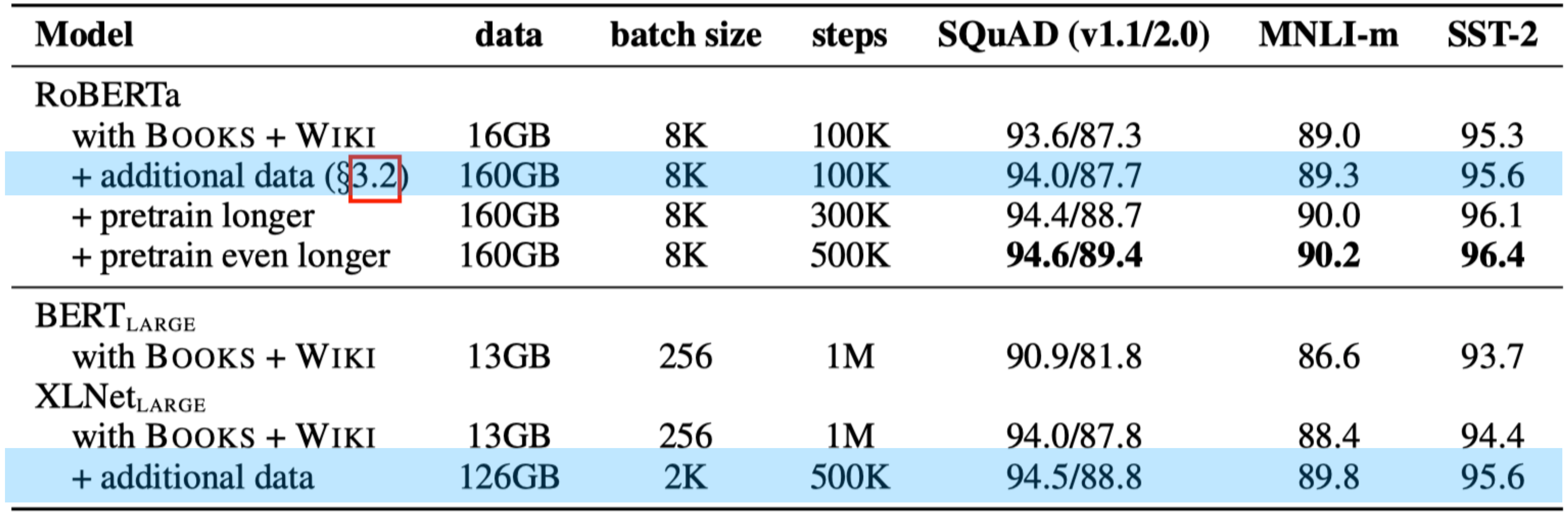
Liu et al., arXiv 2019. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*

RoBERTa

- **RoBERTa模型的最终结构：Sum Up All Good Things**
  - **预训练任务**
    - 动态掩码技术（Dynamic Masking）
    - 使用整句输入，舍弃NSP损失（Full-Sentences without NSP loss）
  - **预训练实验设置**
    - 使用更大的批次大小：256 → 8192
    - 更大的byte-level BPE词表（sentencepiece）：30k → 50K

RoBERTa

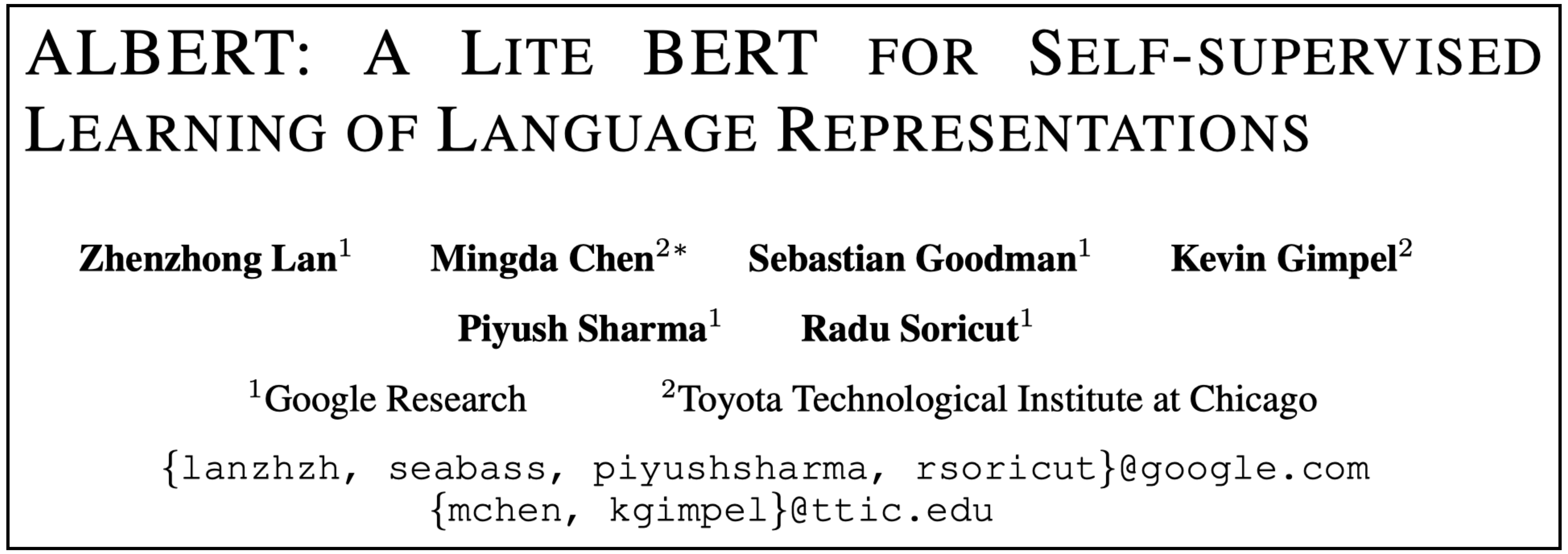
- **实验结果**
  - 在可比条件下：XLNet > RoBERTa > BERT
  - 经过更长时间的训练，RoBERTa可以获得进一步性能提升



Model	data	batch size	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

ALBERT

- **ALBERT: A L**ite **BERT** for Self-supervised Learning of Language Representations
  - 提出一种更加小巧（从参数量的角度）的预训练
  - 两种主要技术：词向量因式分解、跨层参数共享



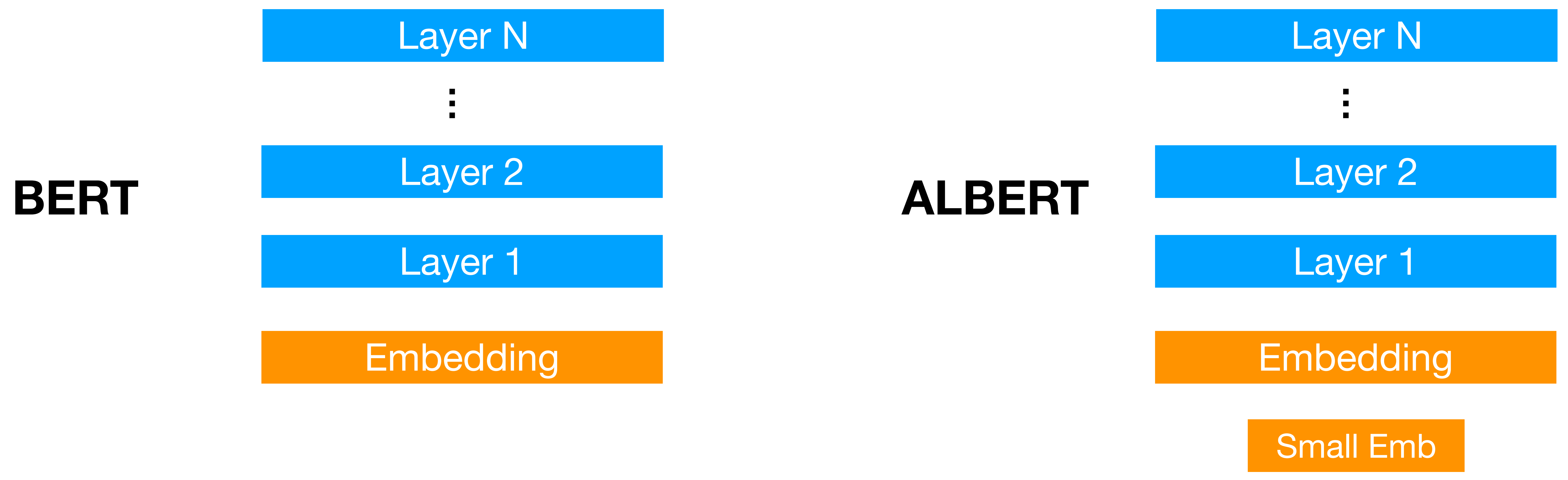
ALBERT: A LITE BERT FOR SELF-SUPERVISED LEARNING OF LANGUAGE REPRESENTATIONS

Zhenzhong Lan[1] **Mingda Chen**[2*] **Sebastian Goodman**[1] **Kevin Gimpel**[2]
Piyush Sharma[1] **Radu Soricut**[1]
[1]Google Research [2]Toyota Technological Institute at Chicago
{lanzhzh, seabass, piyushsharma, rsoricut}@google.com
{mchen, kgimpel}@ttic.edu

Lan et al., ICLR 2020. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations

ALBERT

- **词向量因式分解（Factorized Embedding Parameterization）**
  - 在BERT中，`embedding_size` == `hidden_size`
  - 在ALBERT中，`embedding_size` < `hidden_size`



ALBERT

- **词向量因式分解**
    - 利用词向量因式分解后

$$O(V \times H) \rightarrow O(V \times E + E \times H)$$

    - 举例： $V = 30{,}000,\ H = 1024,\ E = 128$

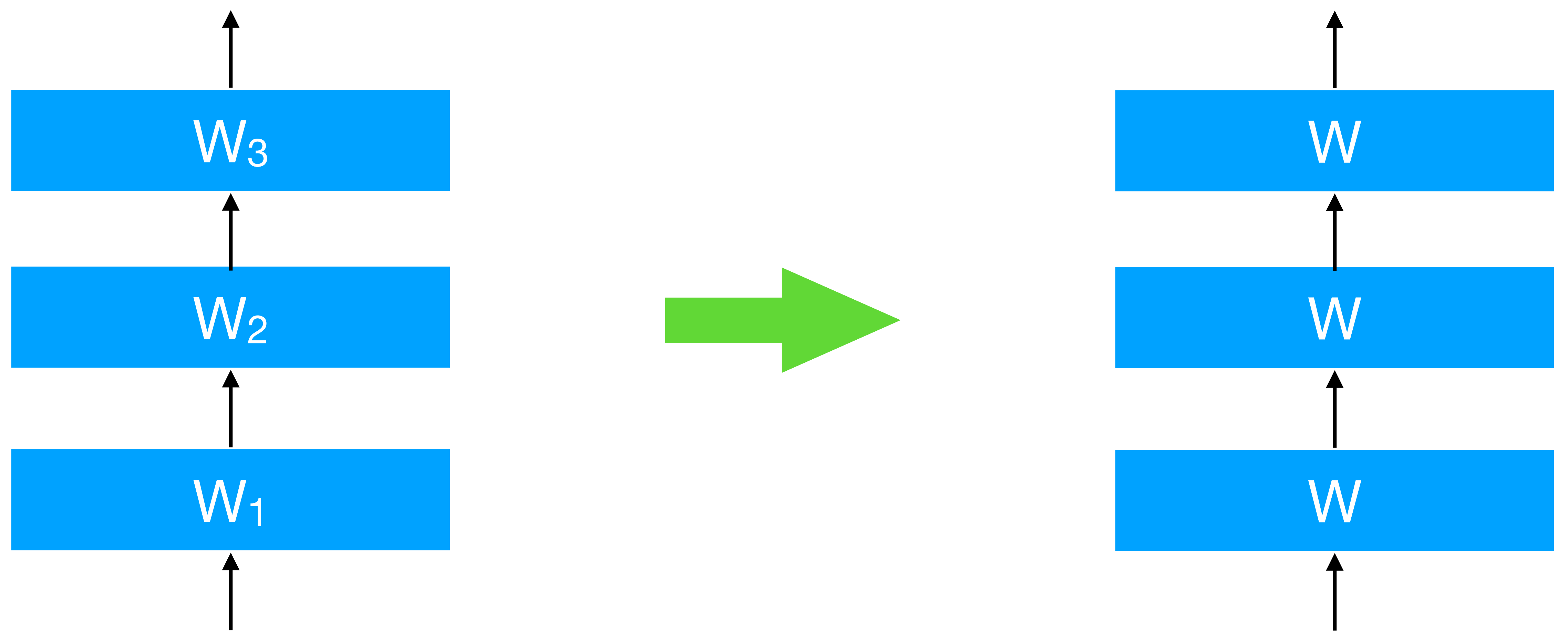
BERT > **ALBERT**

$V*H = 30000*1024=\mathbf{30{,}720{,}000}$

$V*E+E*H = 30000*128+128*1024=\mathbf{3{,}971{,}072}$

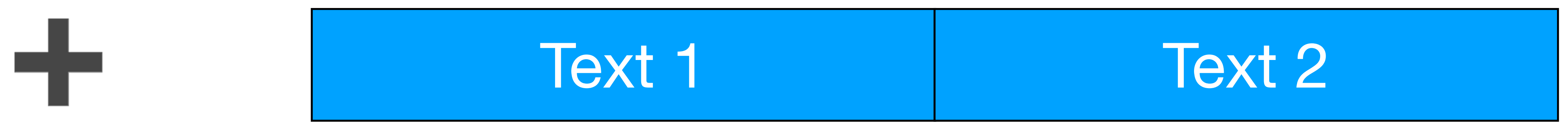
ALBERT

- **跨层参数共享**
  - Transformer每层的参数是**共享**的，即只需要存一份参数，与层数无关
  - 训练时：虽然参数共享，但每层的梯度是不同的，仍然需要额外空间存储
  - 推断时：前向计算过程仍然要一层层展开，**并不能节省推断时间**

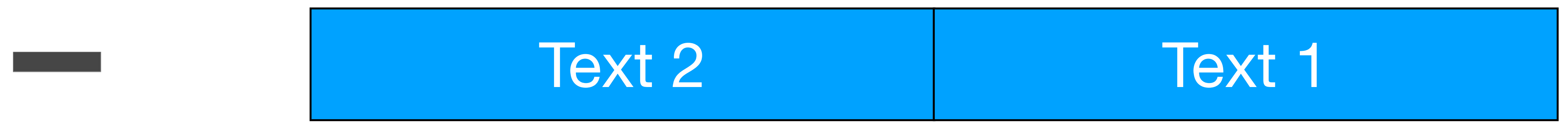


ALBERT

- **句子顺序预测 (Sentence Order Prediction, SOP)**
  - NSP任务实际隐含了“连贯性”和“主题”预测
  - 然而，判断两段文本的主题是否一致是比较容易的
  - ALBERT提出句子顺序预测任务
    - 正样本：与BERT相同，由两个连续的文本段组成

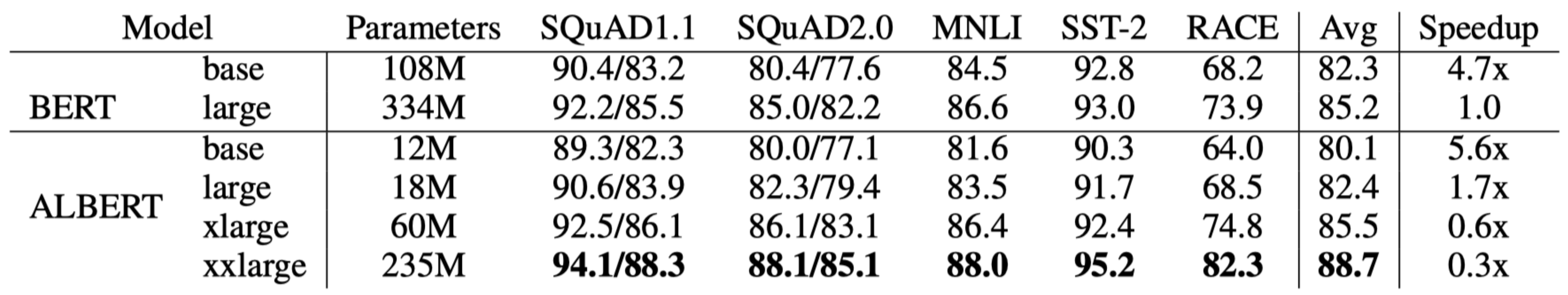


    - 负样本：交换两个连续文本段的顺序

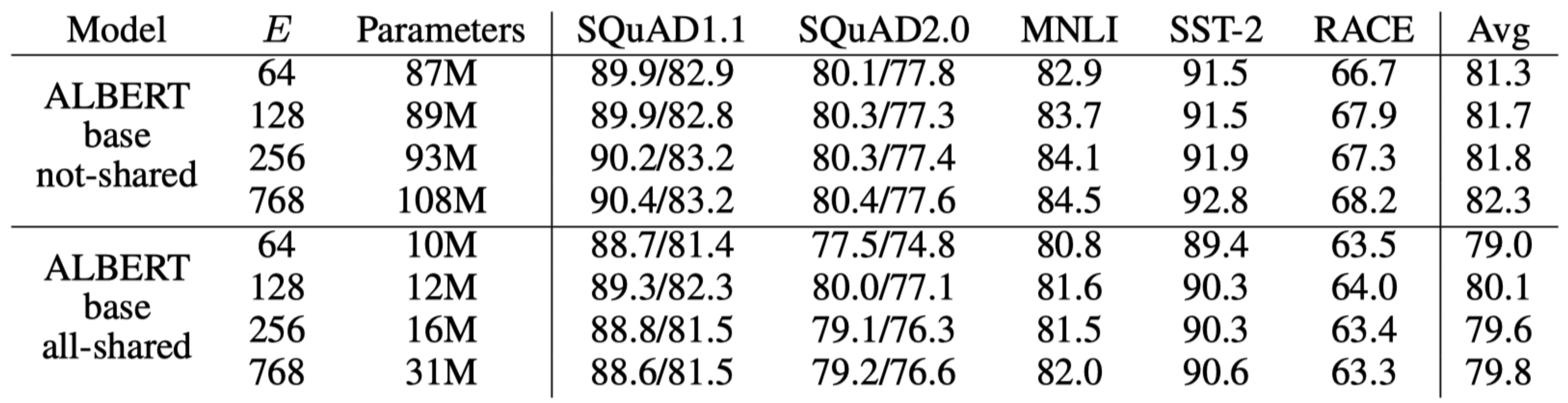


ALBERT

- **有效性分析**
  - ALBERT空间占用小，但并不意味着有很大加速
  - 实验效果上
    - ALBERT-large ≈ BERT-base
    - ALBERT-xlarge ≈ BERT-large
  - ALBERT-xxlarge达到最好效果
  - 参数共享以及词向量分解会导致**效果下降**
  - SOP预训练任务优于NSP以及不使用句对预训练任务

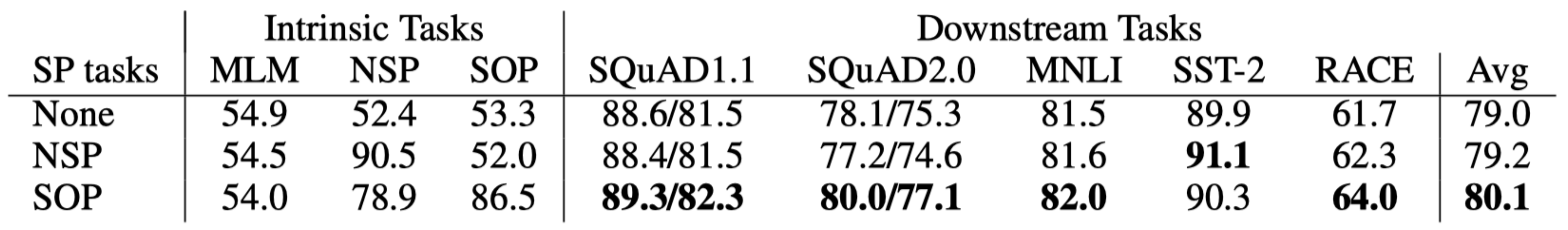


Model		Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg	Speedup
BERT	base	108M	90.4/83.2	80.4/77.6	84.5	92.8	68.2	82.3	4.7x
	large	334M	92.2/85.5	85.0/82.2	86.6	93.0	73.9	85.2	1.0
ALBERT	base	12M	89.3/82.3	80.0/77.1	81.6	90.3	64.0	80.1	5.6x
	large	18M	90.6/83.9	82.3/79.4	83.5	91.7	68.5	82.4	1.7x
	xlarge	60M	92.5/86.1	86.1/83.1	86.4	92.4	74.8	85.5	0.6x
	xxlarge	235M	**94.1/88.3**	**88.1/85.1**	**88.0**	**95.2**	**82.3**	**88.7**	0.3x



Model	E	Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
ALBERT base not-shared	64	87M	89.9/82.9	80.1/77.8	82.9	91.5	66.7	81.3
	128	89M	89.9/82.8	80.3/77.3	83.7	91.5	67.9	81.7
	256	93M	90.2/83.2	80.3/77.4	84.1	91.9	67.3	81.8
	768	108M	90.4/83.2	80.4/77.6	84.5	92.8	68.2	82.3
ALBERT base all-shared	64	10M	88.7/81.4	77.5/74.8	80.8	89.4	63.5	79.0
	128	12M	89.3/82.3	80.0/77.1	81.6	90.3	64.0	80.1
	256	16M	88.8/81.5	79.1/76.3	81.5	90.3	63.4	79.6
	768	31M	88.6/81.5	79.2/76.6	82.0	90.6	63.3	79.8

Table 3: The effect of vocabulary embedding size on the performance of ALBERT-base.

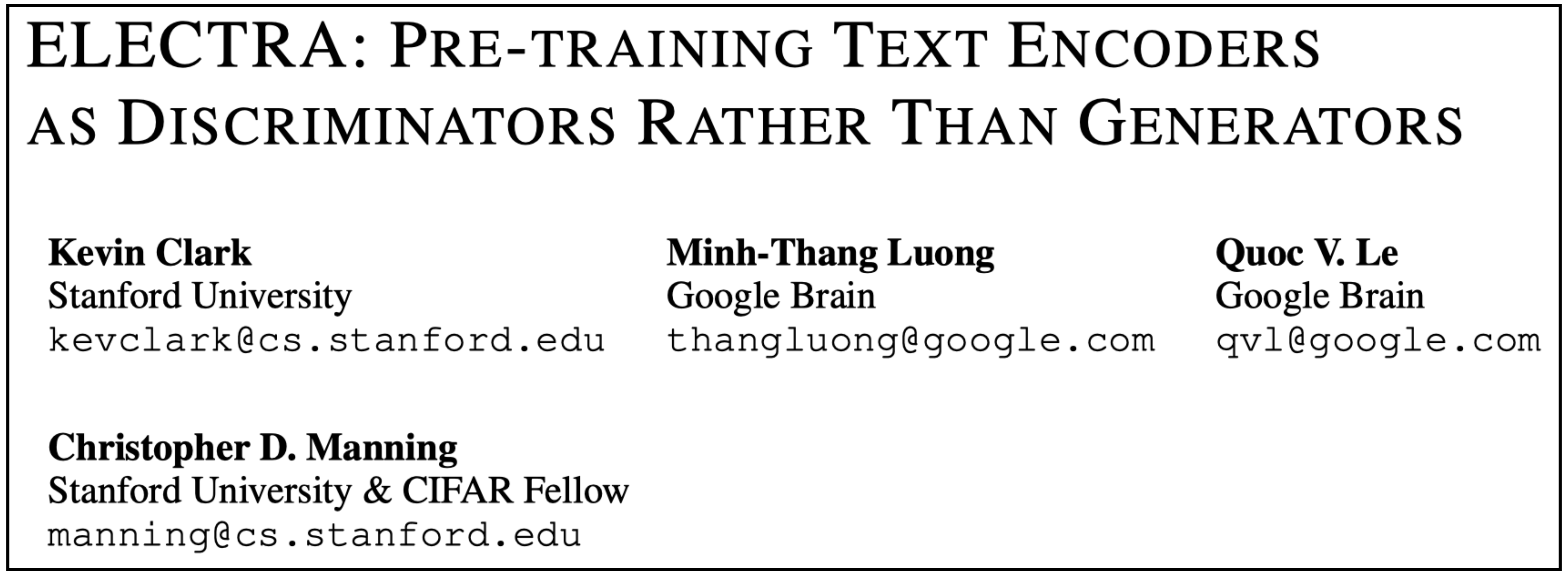


	Intrinsic Tasks			Downstream Tasks					
SP tasks	MLM	NSP	SOP	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg
None	54.9	52.4	53.3	88.6/81.5	78.1/75.3	81.5	89.9	61.7	79.0
NSP	54.5	90.5	52.0	88.4/81.5	77.2/74.6	81.6	**91.1**	62.3	79.2
SOP	54.0	78.9	86.5	**89.3/82.3**	**80.0/77.1**	**82.0**	90.3	**64.0**	**80.1**

Table 5: The effect of sentence-prediction loss, NSP vs. SOP, on intrinsic and downstream tasks.

ELECTRA

- **ELECTRA: E**fficiently **L**earning an **E**ncoder that **C**lassifies **T**oken **R**eplacements **A**ccurately
  - 提出了一种全新基于生成器-判别器框架的预训练语言模型
  - 相比传统预训练模型的训练效率更高



ELECTRA: PRE-TRAINING TEXT ENCODERS AS DISCRIMINATORS RATHER THAN GENERATORS

Kevin Clark
Stanford University
kevclark@cs.stanford.edu

Minh-Thang Luong
Google Brain
thangluong@google.com

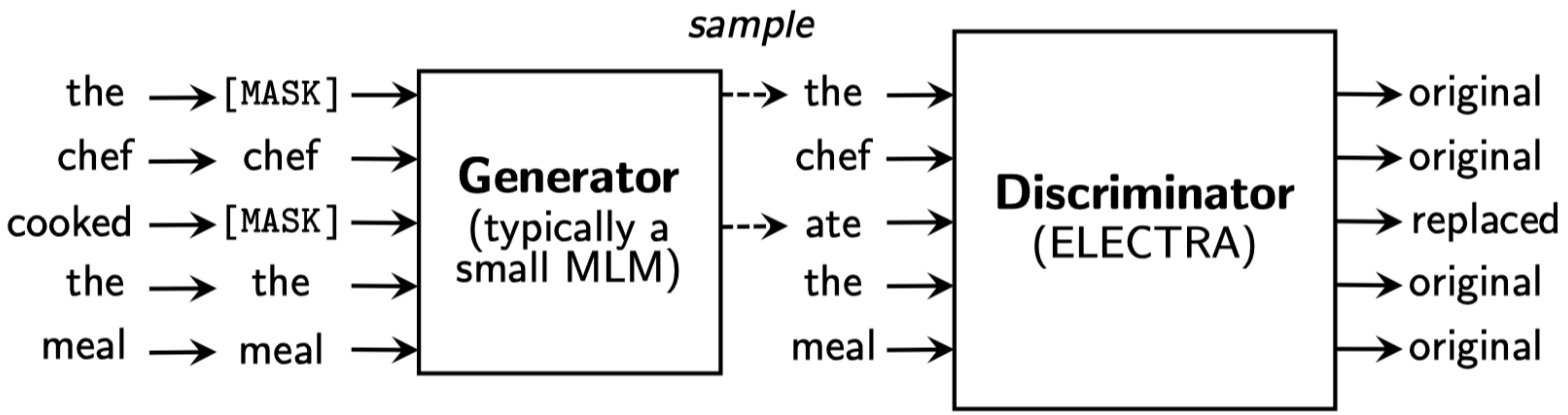
Quoc V. Le
Google Brain
qvl@google.com

Christopher D. Manning
Stanford University & CIFAR Fellow
manning@cs.stanford.edu

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

ELECTRA

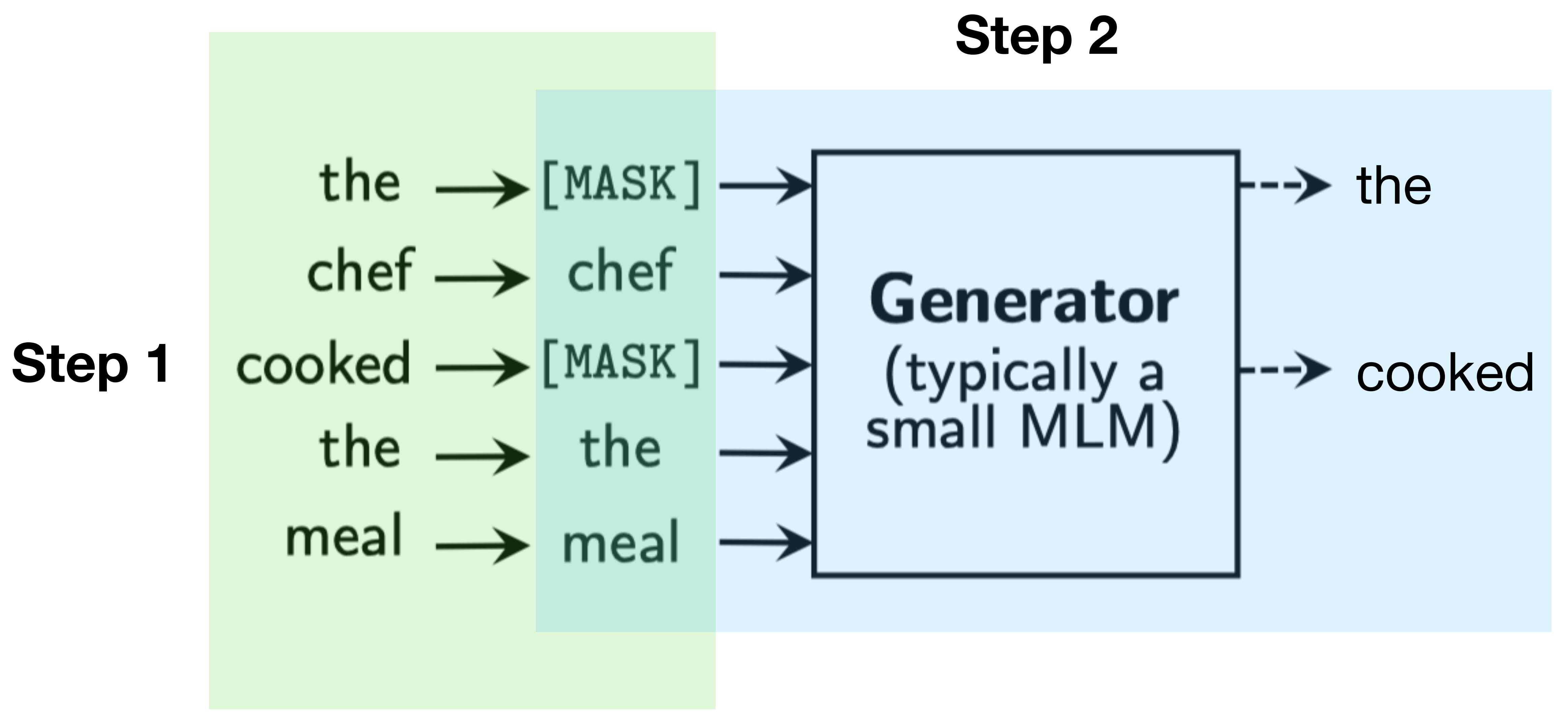
- **整体结构**
  - 生成器-判别器框架，与GAN (Goodfellow et al., 2014) 类似
  - **Generator**: 将输入中的缺失信息还原为原单词
  - **Discriminator**: 判断输入的单词是否被替换过



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

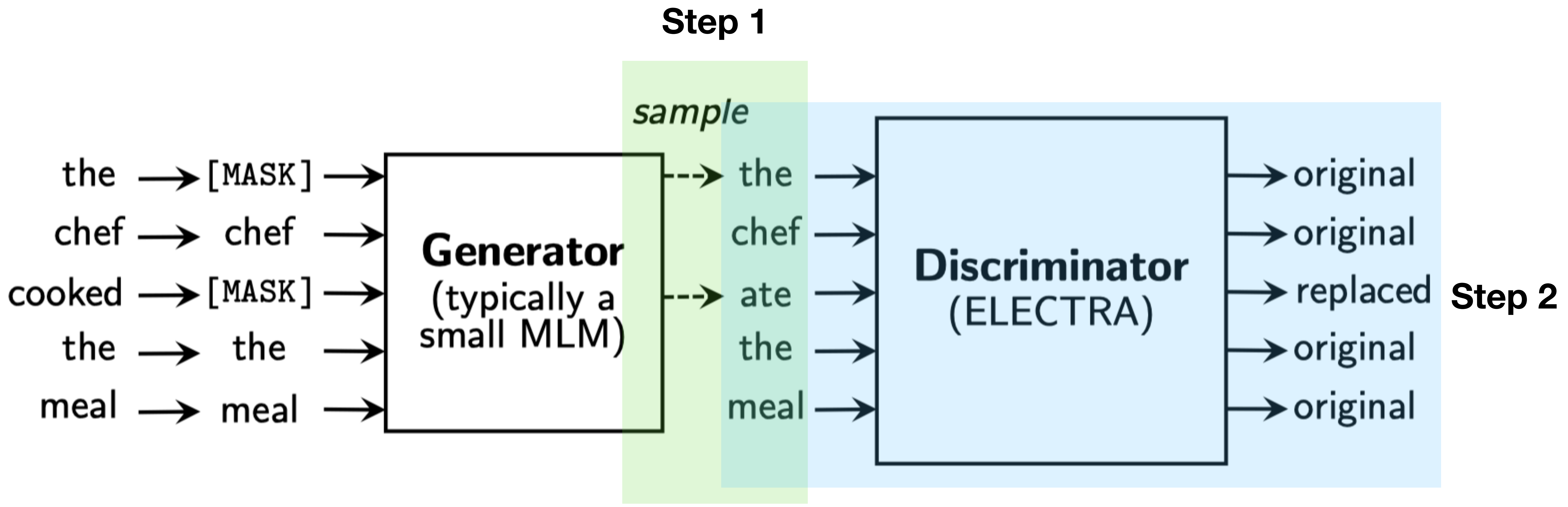
ELECTRA

- **生成器：一个小的MLM**
  - 第一步：随机选取输入序列中的一部分进行遮蔽，通常比例选取15%
  - 第二步：利用MLM任务，将缺失信息还原为原单词



ELECTRA

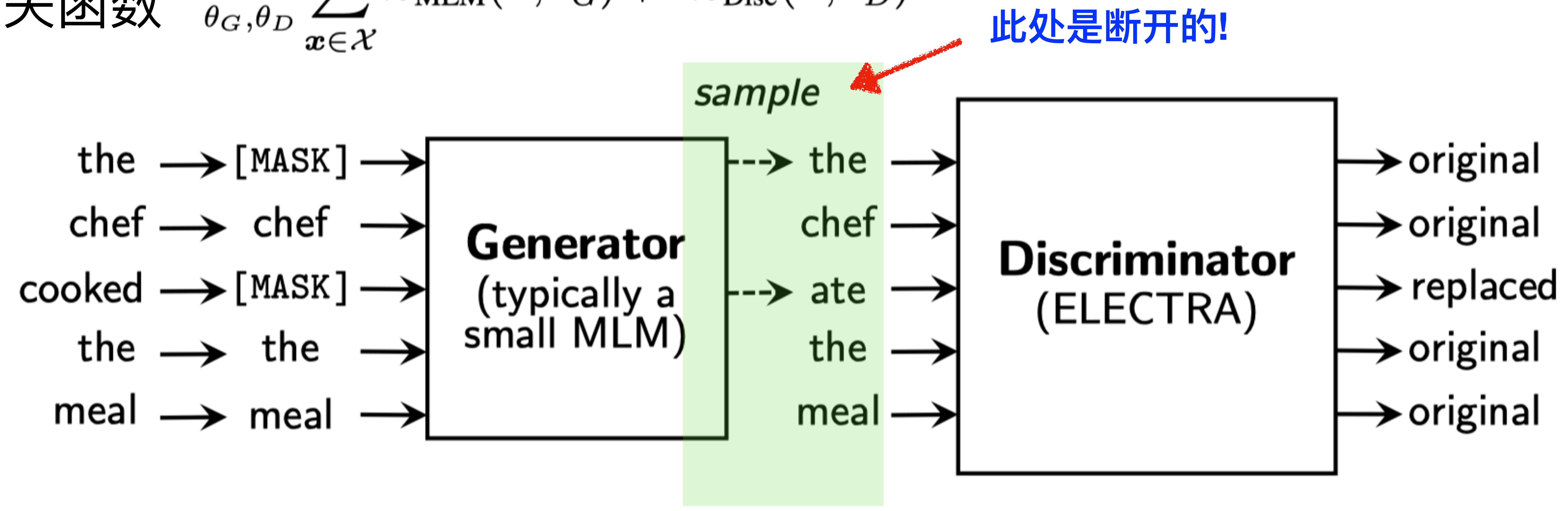
- **判别器：常规的BERT结构**
  - 第一步：将缺失信息替换为生成器预测出来的单词
  - 第二步：判别器学习判断输入句子中哪些单词被生成器替换过



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

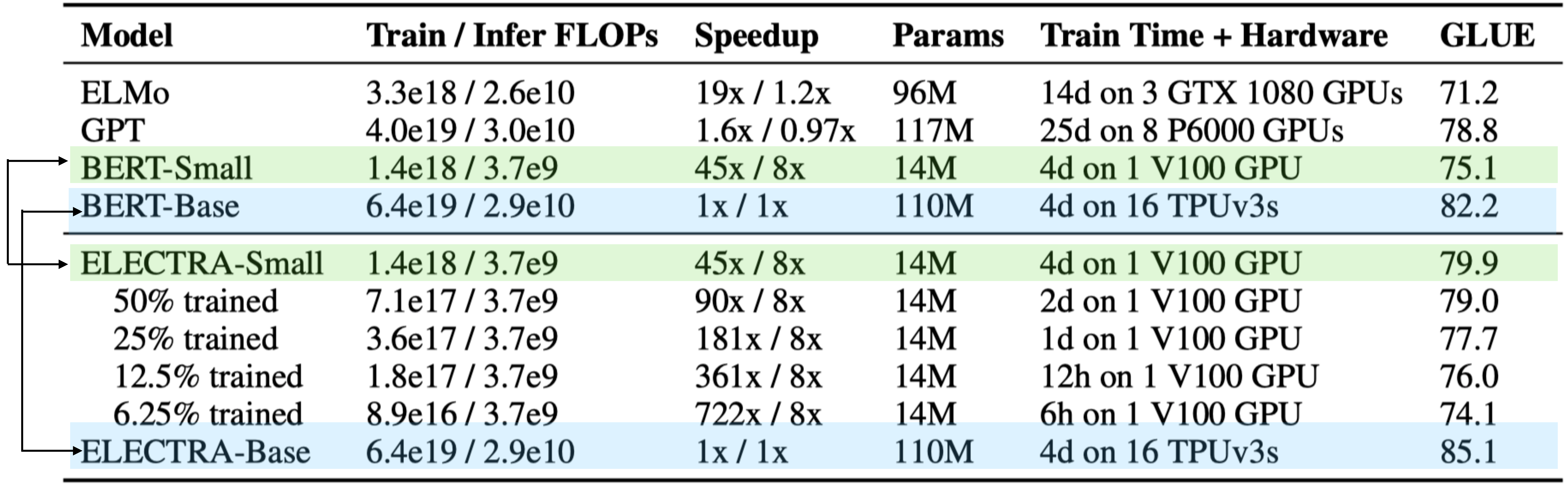
ELECTRA

- **ELECTRA的训练方式与GAN并不相同**
    - 生成器和判别器衔接部分是“采样”操作，此处不可导
    - 作者尝试使用强化学习，但效果比较差
    - 最终损失函数 $\min_{\theta_G,\theta_D} \sum_{\boldsymbol{x}\in\mathcal{X}} \mathcal{L}_{\text{MLM}}(\boldsymbol{x},\theta_G) + \lambda\mathcal{L}_{\text{Disc}}(\boldsymbol{x},\theta_D)$



ELECTRA

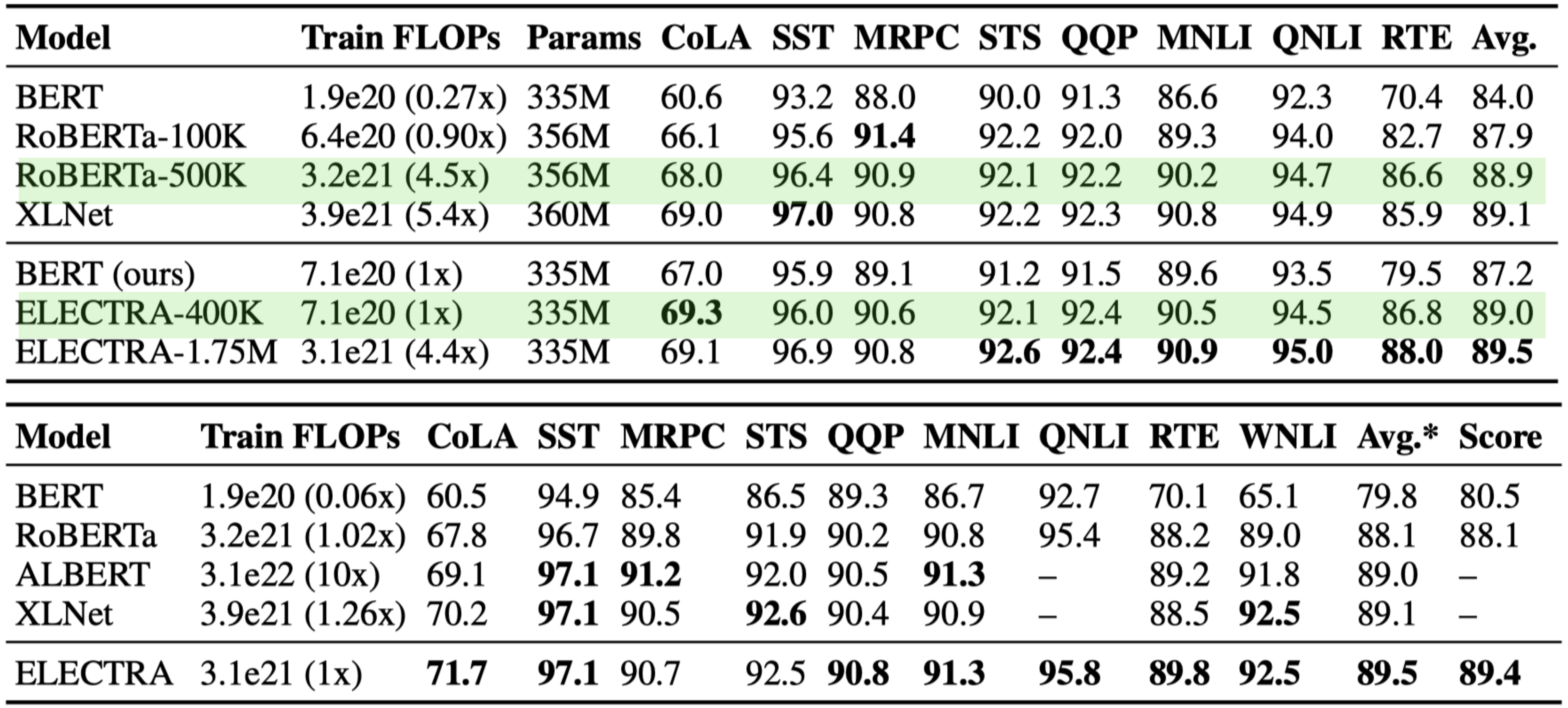
- **实验结果**
  - ELECTRA-small/base模型相比同等大小的BERT效果更优



Model	Train / Infer FLOPs	Speedup	Params	Train Time + Hardware	GLUE
ELMo	3.3e18 / 2.6e10	19x / 1.2x	96M	14d on 3 GTX 1080 GPUs	71.2
GPT	4.0e19 / 3.0e10	1.6x / 0.97x	117M	25d on 8 P6000 GPUs	78.8
BERT-Small	1.4e18 / 3.7e9	45x / 8x	14M	4d on 1 V100 GPU	75.1
BERT-Base	6.4e19 / 2.9e10	1x / 1x	110M	4d on 16 TPUv3s	82.2
ELECTRA-Small	1.4e18 / 3.7e9	45x / 8x	14M	4d on 1 V100 GPU	79.9
50% trained	7.1e17 / 3.7e9	90x / 8x	14M	2d on 1 V100 GPU	79.0
25% trained	3.6e17 / 3.7e9	181x / 8x	14M	1d on 1 V100 GPU	77.7
12.5% trained	1.8e17 / 3.7e9	361x / 8x	14M	12h on 1 V100 GPU	76.0
6.25% trained	8.9e16 / 3.7e9	722x / 8x	14M	6h on 1 V100 GPU	74.1
ELECTRA-Base	6.4e19 / 2.9e10	1x / 1x	110M	4d on 16 TPUv3s	85.1

ELECTRA

- **实验结果**
  - ELECTRA-large在GLUE-dev/test上获得了最优效果

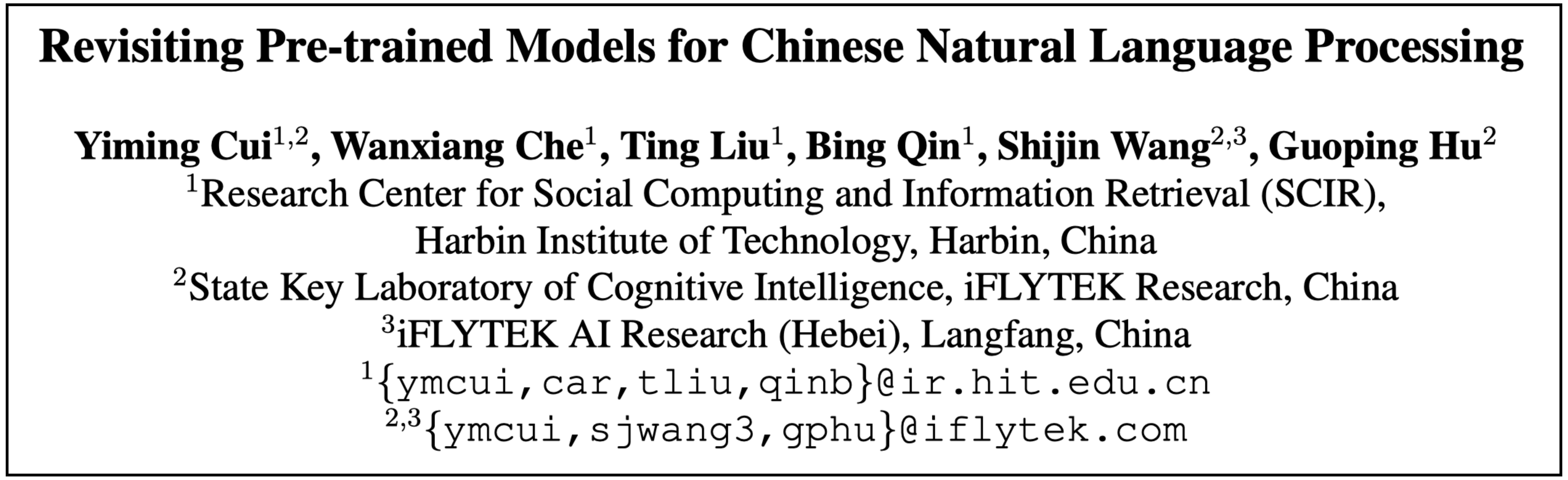


Model	Train FLOPs	Params	CoLA	SST	MRPC	STS	QQP	MNLI	QNLI	RTE	Avg.
BERT	1.9e20 (0.27x)	335M	60.6	93.2	88.0	90.0	91.3	86.6	92.3	70.4	84.0
RoBERTa-100K	6.4e20 (0.90x)	356M	66.1	95.6	**91.4**	92.2	92.0	89.3	94.0	82.7	87.9
RoBERTa-500K	3.2e21 (4.5x)	356M	68.0	96.4	90.9	92.1	92.2	90.2	94.7	86.6	88.9
XLNet	3.9e21 (5.4x)	360M	69.0	**97.0**	90.8	92.2	92.3	90.8	94.9	85.9	89.1
BERT (ours)	7.1e20 (1x)	335M	67.0	95.9	89.1	91.2	91.5	89.6	93.5	79.5	87.2
ELECTRA-400K	7.1e20 (1x)	335M	**69.3**	96.0	90.6	92.1	92.4	90.5	94.5	86.8	89.0
ELECTRA-1.75M	3.1e21 (4.4x)	335M	69.1	96.9	90.8	**92.6**	**92.4**	**90.9**	**95.0**	**88.0**	**89.5**

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**

MacBERT

- **MacBERT: M**LM **as** correction **BERT**
  - 在可比条件下，评测了主流预训练模型在中文下的表现
  - 提出一种新的预训练模型MacBERT，解决了“预训练-精调”不一致的问题
  - 文中涉及到的所有中文预训练模型已开源至整个研究社区


Revisiting Pre-trained Models for Chinese Natural Language Processing

Yiming Cui[1,2], **Wanxiang Che**[1], **Ting Liu**[1], **Bing Qin**[1], **Shijin Wang**[2,3], **Guoping Hu**[2]
[1]Research Center for Social Computing and Information Retrieval (SCIR),
Harbin Institute of Technology, Harbin, China
[2]State Key Laboratory of Cognitive Intelligence, iFLYTEK Research, China
[3]iFLYTEK AI Research (Hebei), Langfang, China
[1]{ymcui,car,tliu,qinb}@ir.hit.edu.cn
[2,3]{ymcui,sjwang3,gphu}@iflytek.com

MacBERT

- **提出MacBERT预训练语言模型**
  - 解决BERT中的“预训练-精调”不一致的问题
  - 提出了一种基于相似词替换的预训练任务（MLM as correction），相当于“文本纠错”

用 语 言 模 型 **预 测** 下 一 个 词

BERT

- 80% of the time, replace with [M]
  – 用 语 言 模 型 **[M] [M]** 下 一 个 词
- 10% of the time, replace random word
  – 用 语 言 模 型 **预 见** 下 一 个 词
- 10% of the time, keep the same word
  – 用 语 言 模 型 **预 测** 下 一 个 词

MACBERT

- 80% of the time, replace with [M]
  – 用 语 言 模 型 **预 见** 下 一 个 词
- 10% of the time, replace random word
  – 用 语 言 模 型 **好 是** 下 一 个 词
- 10% of the time, keep the same word
  – 用 语 言 模 型 **预 测** 下 一 个 词

Cui et al., Findings of EMNLP 2020. *Revisiting Pre-trained Models for Chinese Natural Language Processing*

MacBERT

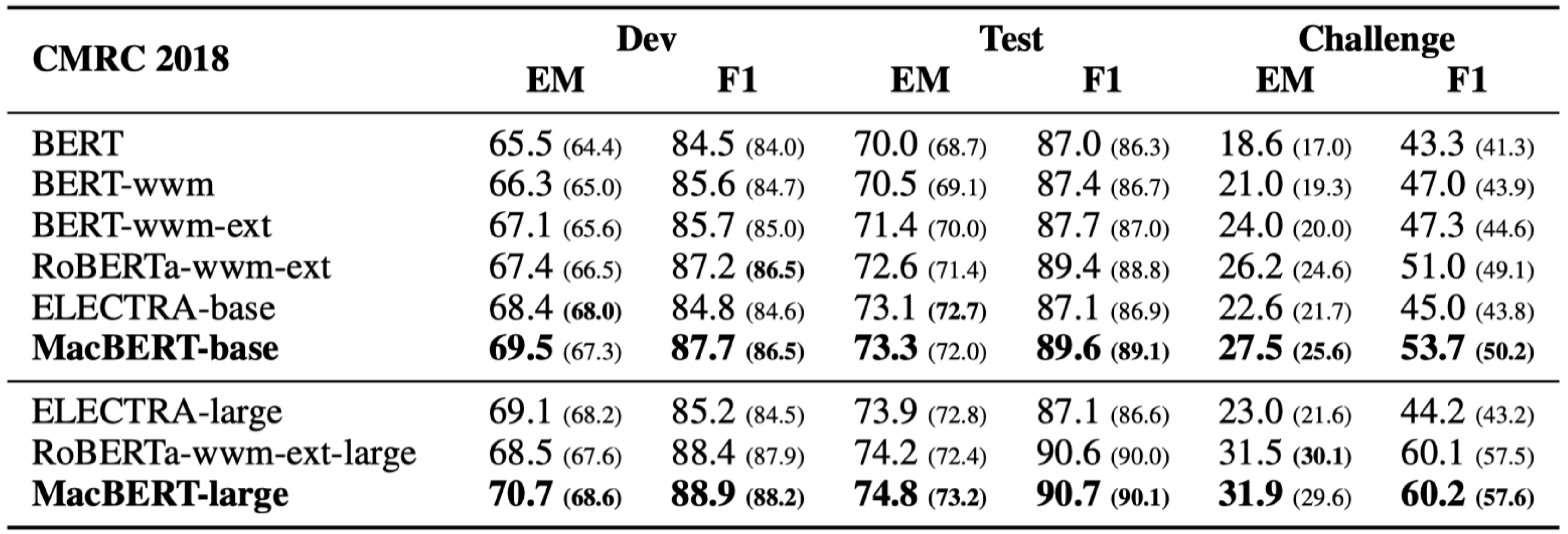
- 不同预训练模型之间的对比

	BERT	ERNIE	XLNet	RoBERTa	ALBERT	ELECTRA	MacBERT
Type	AE	AE	AR	AE	AE	AE	AE
Embeddings	T/S/P	T/S/P	T/S/P	T/S/P	T/S/P	T/S/P	T/S/P
LM Task	MLM	MLM	PLM	MLM	MLM	Gen-Dis	Mac
Masking	T	T/E/Ph	-	T	T	T	WWM/NM
Paired Task	NSP	NSP	-	-	SOP	-	SOP

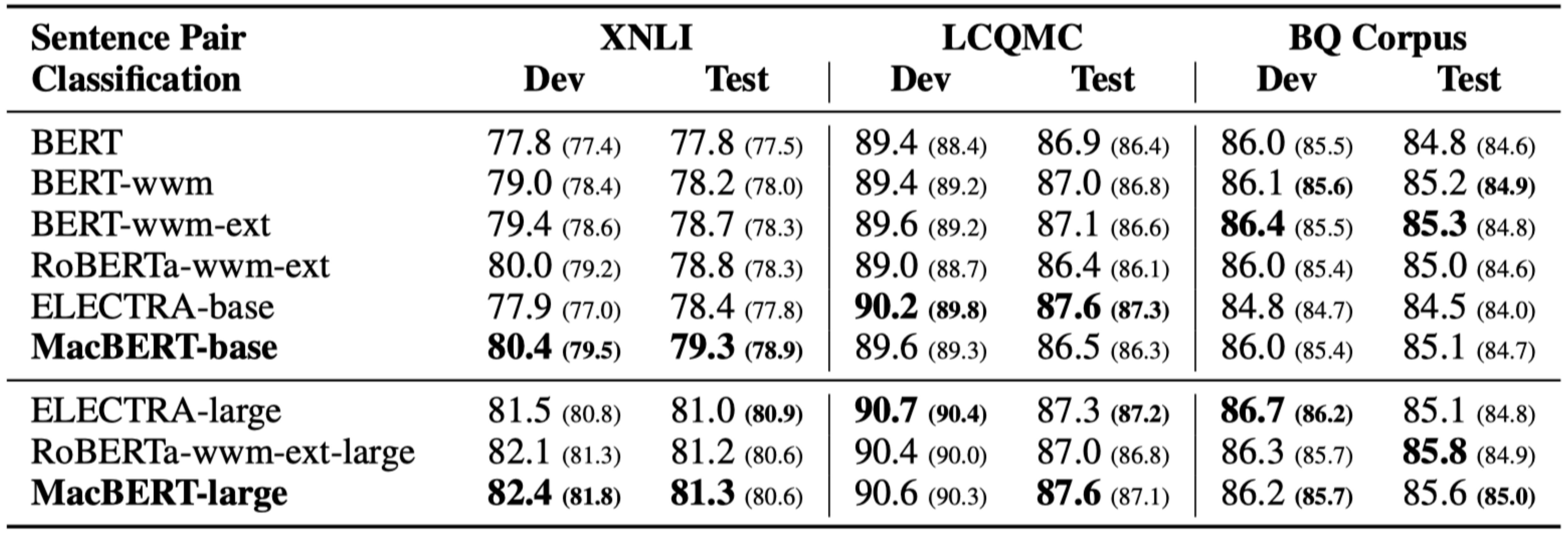
- AE=自编码，AR=自回归
- T=Token, S=Segment, P=Position
- E=Entity, Ph=Phrase

MacBERT

- **实验结果**
  - 在机器阅读理解任务上获得显著性能提升
  - 在文本分类任务上也获得了一定程度的性能提升



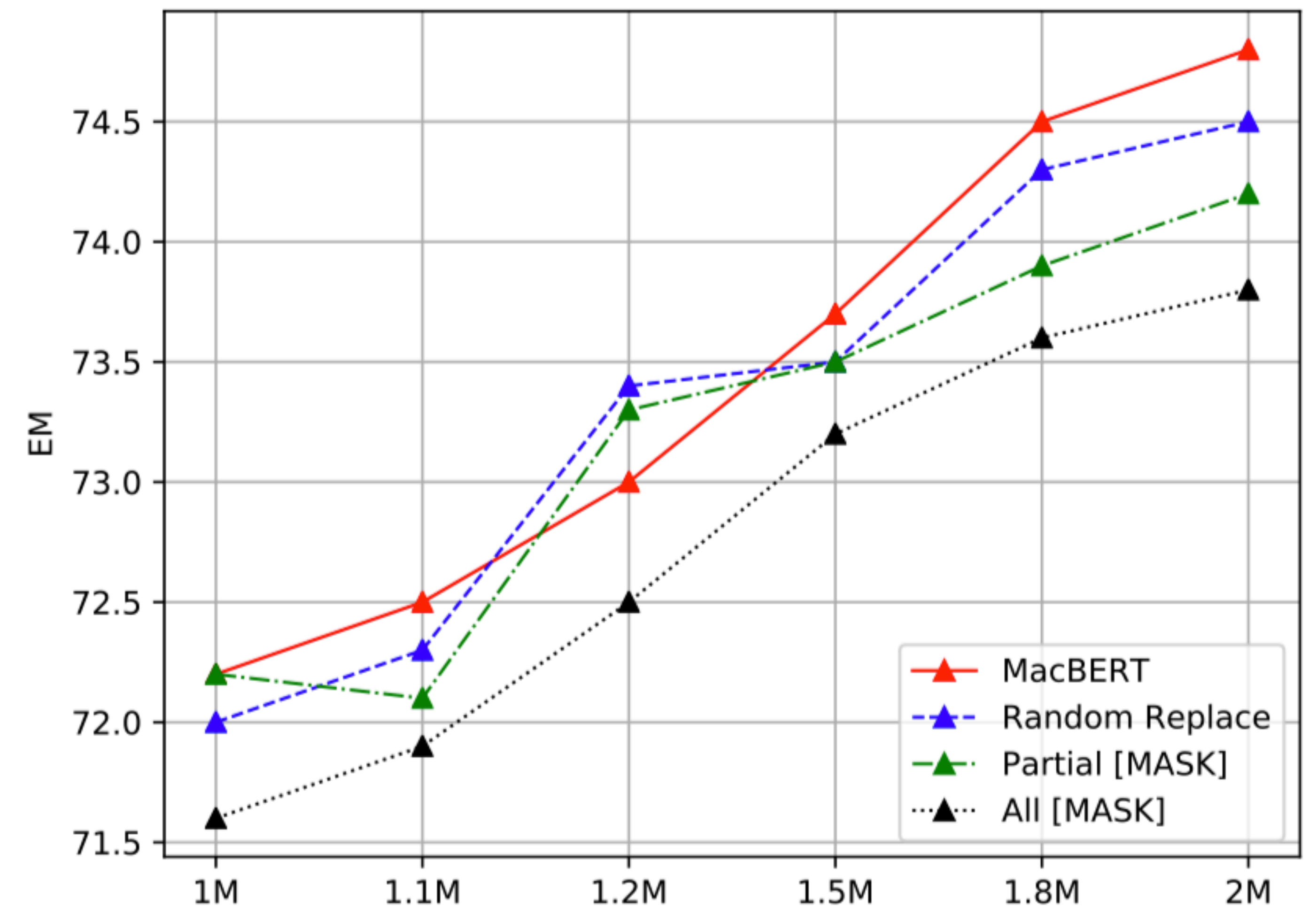
CMRC 2018	Dev EM	Dev F1	Test EM	Test F1	Challenge EM	Challenge F1
BERT	65.5 (64.4)	84.5 (84.0)	70.0 (68.7)	87.0 (86.3)	18.6 (17.0)	43.3 (41.3)
BERT-wwm	66.3 (65.0)	85.6 (84.7)	70.5 (69.1)	87.4 (86.7)	21.0 (19.3)	47.0 (43.9)
BERT-wwm-ext	67.1 (65.6)	85.7 (85.0)	71.4 (70.0)	87.7 (87.0)	24.0 (20.0)	47.3 (44.6)
RoBERTa-wwm-ext	67.4 (66.5)	87.2 **(86.5)**	72.6 (71.4)	89.4 (88.8)	26.2 (24.6)	51.0 (49.1)
ELECTRA-base	68.4 **(68.0)**	84.8 (84.6)	73.1 **(72.7)**	87.1 (86.9)	22.6 (21.7)	45.0 (43.8)
MacBERT-base	**69.5** (67.3)	**87.7 (86.5)**	**73.3** (72.0)	**89.6 (89.1)**	**27.5 (25.6)**	**53.7 (50.2)**
ELECTRA-large	69.1 (68.2)	85.2 (84.5)	73.9 (72.8)	87.1 (86.6)	23.0 (21.6)	44.2 (43.2)
RoBERTa-wwm-ext-large	68.5 (67.6)	88.4 (87.9)	74.2 (72.4)	90.6 (90.0)	31.5 **(30.1)**	60.1 (57.5)
MacBERT-large	**70.7 (68.6)**	**88.9 (88.2)**	**74.8 (73.2)**	**90.7 (90.1)**	**31.9** (29.6)	**60.2 (57.6)**



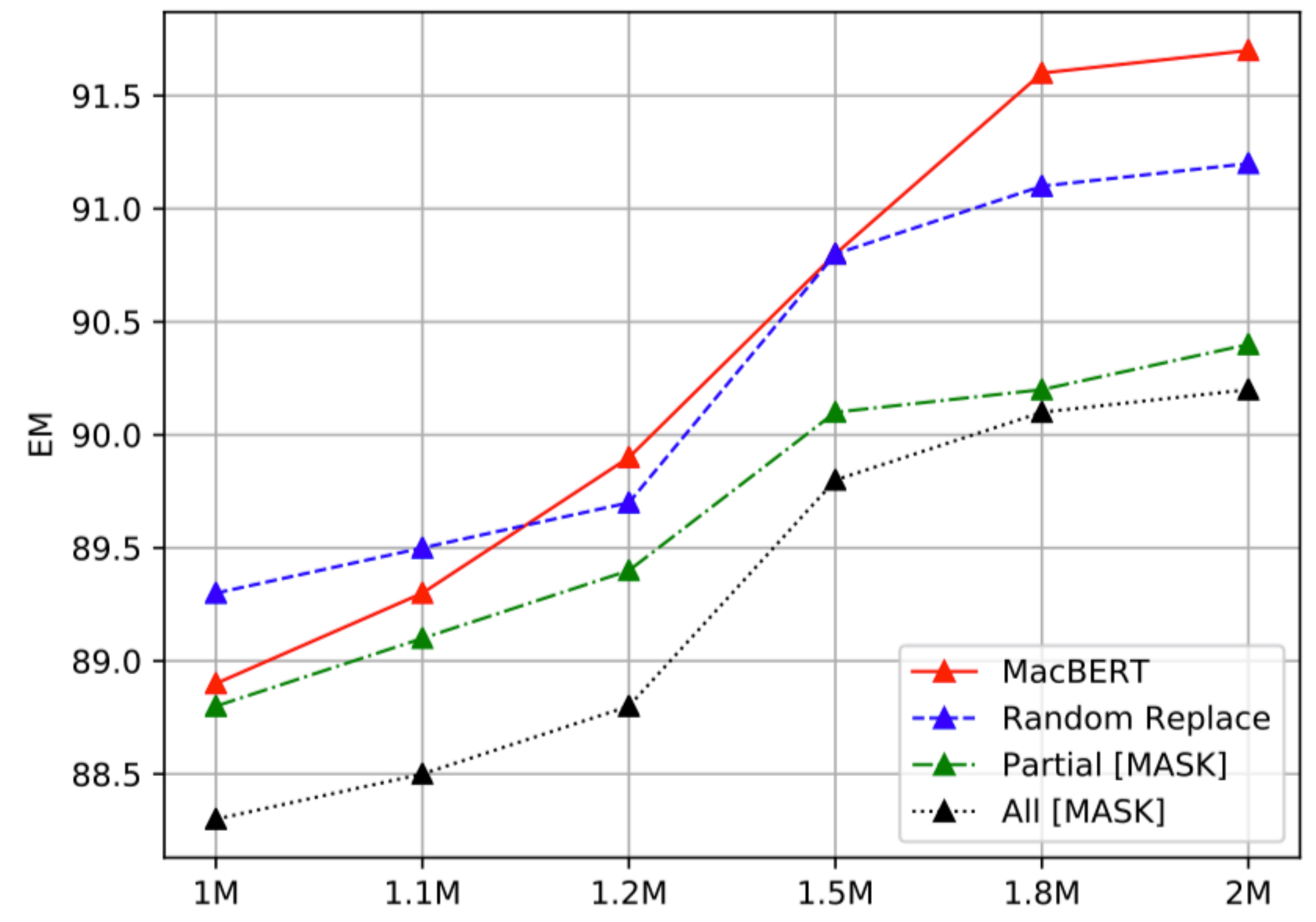
Sentence Pair Classification	XNLI Dev	XNLI Test	LCQMC Dev	LCQMC Test	BQ Corpus Dev	BQ Corpus Test
BERT	77.8 (77.4)	77.8 (77.5)	89.4 (88.4)	86.9 (86.4)	86.0 (85.5)	84.8 (84.6)
BERT-wwm	79.0 (78.4)	78.2 (78.0)	89.4 (89.2)	87.0 (86.8)	86.1 **(85.6)**	85.2 **(84.9)**
BERT-wwm-ext	79.4 (78.6)	78.7 (78.3)	89.6 (89.2)	87.1 (86.6)	**86.4** (85.5)	**85.3** (84.8)
RoBERTa-wwm-ext	80.0 (79.2)	78.8 (78.3)	89.0 (88.7)	86.4 (86.1)	86.0 (85.4)	85.0 (84.6)
ELECTRA-base	77.9 (77.0)	78.4 (77.8)	**90.2 (89.8)**	**87.6 (87.3)**	84.8 (84.7)	84.5 (84.0)
MacBERT-base	**80.4 (79.5)**	**79.3 (78.9)**	89.6 (89.3)	86.5 (86.3)	86.0 (85.4)	85.1 (84.7)
ELECTRA-large	81.5 (80.8)	81.0 **(80.9)**	**90.7 (90.4)**	87.3 **(87.2)**	**86.7 (86.2)**	85.1 (84.8)
RoBERTa-wwm-ext-large	82.1 (81.3)	81.2 (80.6)	90.4 (90.0)	87.0 (86.8)	86.3 (85.7)	**85.8** (84.9)
MacBERT-large	**82.4 (81.8)**	**81.3** (80.6)	90.6 (90.3)	**87.6** (87.1)	86.2 **(85.7)**	85.6 **(85.0)**

MacBERT

- **对MLM任务的分析**
  - 输入句子中15%的token被处理
    - MacBERT：80%的token被替换为相似词，10%被替换为随机词
    - Random Replace：90%的token被替换为随机词
    - Partial Mask：原始BERT实现，即80%的token被替换为[MASK]符号，10%被替换为随机词
    - All Mask：90%的token被替换为[MASK]
    - 剩余的10%不做任何替换（负样本）



CMRC 2018

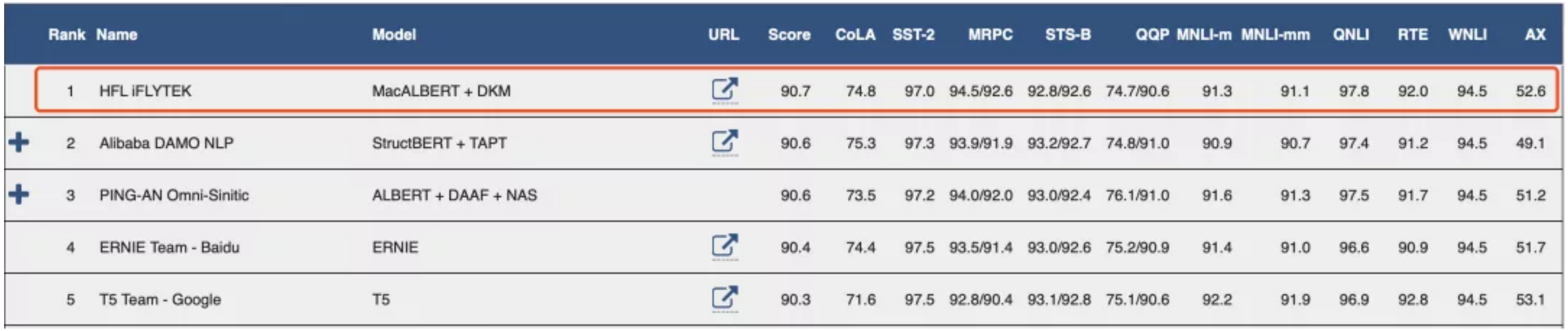


DRCD

Cui et al., Findings of EMNLP 2020. Revisiting Pre-trained Models for Chinese Natural Language Processing

MacBERT

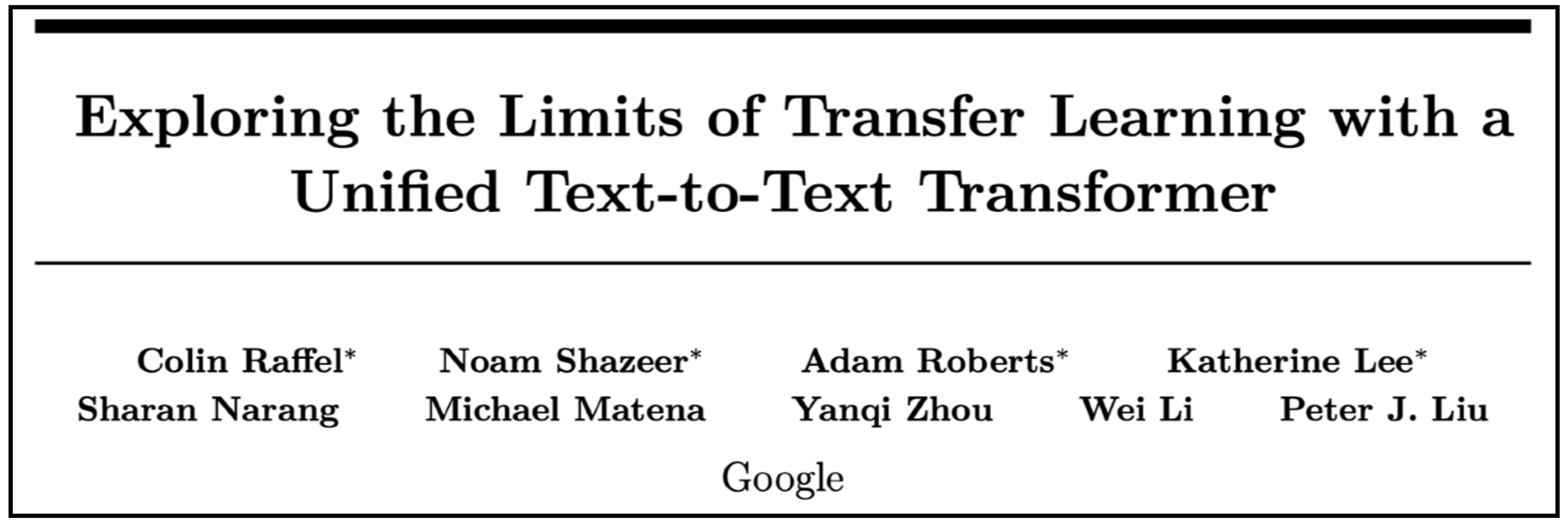
- **2020年8月，获得权威自然语言理解评测GLUE冠军**
  - GLUE评测由纽约大学、华盛顿大学和谷歌DeepMind共同举办
  - 以总平均分90.7荣登该评测榜首，且多项子任务超过榜单最好水平
  - 将相关思想迁移到ALBERT模型，结合了数据增广方法以及动态关键词匹配技术



	Rank	Name	Model	URL	Score	CoLA	SST-2	MRPC	STS-B	QQP	MNLI-m	MNLI-mm	QNLI	RTE	WNLI	AX
	1	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	92.0	94.5	52.6
+	2	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	91.2	94.5	49.1
+	3	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	91.7	94.5	51.2
	4	ERNIE Team - Baidu	ERNIE		90.4	74.4	97.5	93.5/91.4	93.0/92.6	75.2/90.9	91.4	91.0	96.6	90.9	94.5	51.7
	5	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	92.8	94.5	53.1

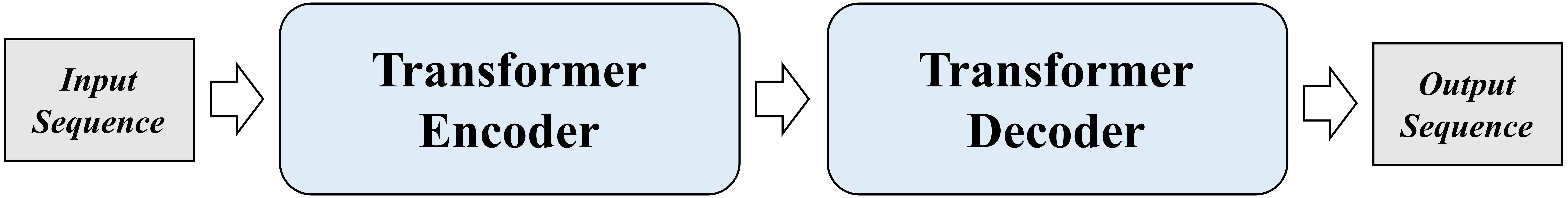
T5

- **T5: Text-to-Text Transfer Transformer**
  - 提出一种适用于各种类型NLP任务的Encoder-Decoder框架
  - 提供了非常细致的模型设计决策过程
  - 提出了C4数据集，包含750G高质量英文数据



Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer

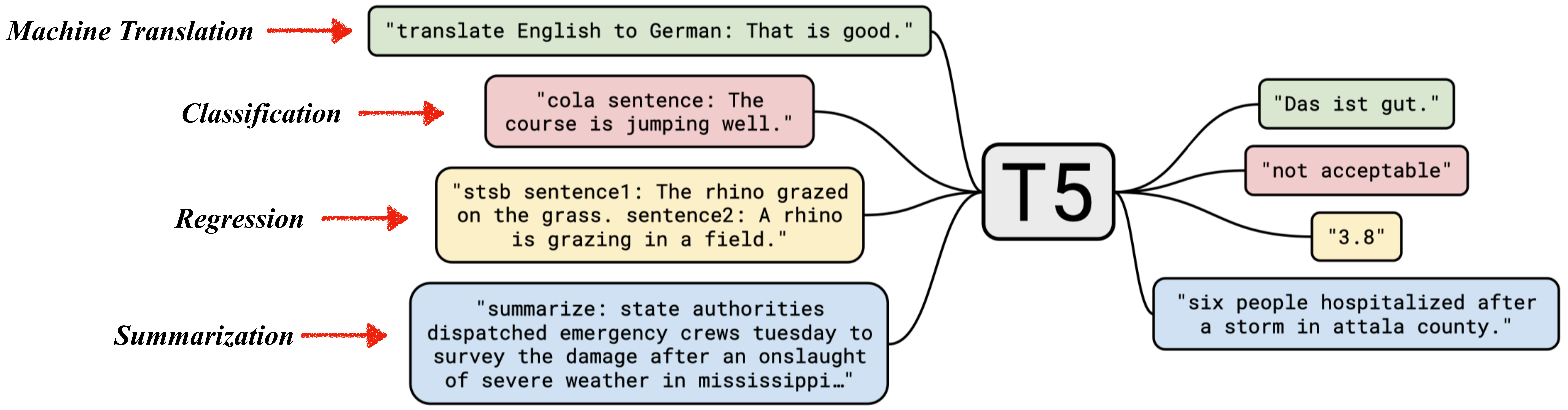
Colin Raffel* Noam Shazeer* Adam Roberts* Katherine Lee*
Sharan Narang Michael Matena Yanqi Zhou Wei Li Peter J. Liu
Google



Raffel et al., arXiv 2019. *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*

T5

- **Text-to-Text 任务设计（下游任务精调阶段）**
  - 将所有NLP任务看做是“text-to-text”问题

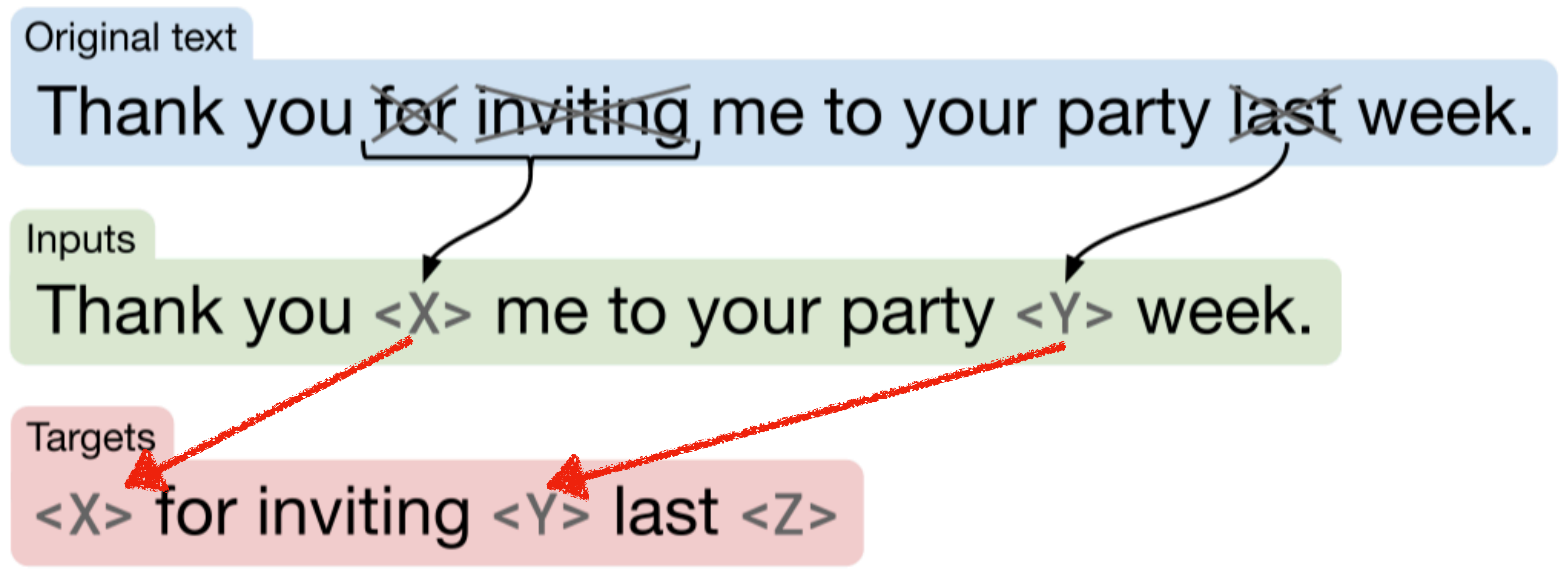


Machine Translation
Classification
Regression
Summarization
"translate English to German: That is good."
"cola sentence: The course is jumping well."
"stsb sentence1: The rhino grazed on the grass. sentence2: A rhino is grazing in a field."
"summarize: state authorities dispatched emergency crews tuesday to survey the damage after an onslaught of severe weather in mississippi…"
T5
"Das ist gut."
"not acceptable"
"3.8"
"six people hospitalized after a storm in attala county."

Raffel et al., arXiv 2019. *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*

T5

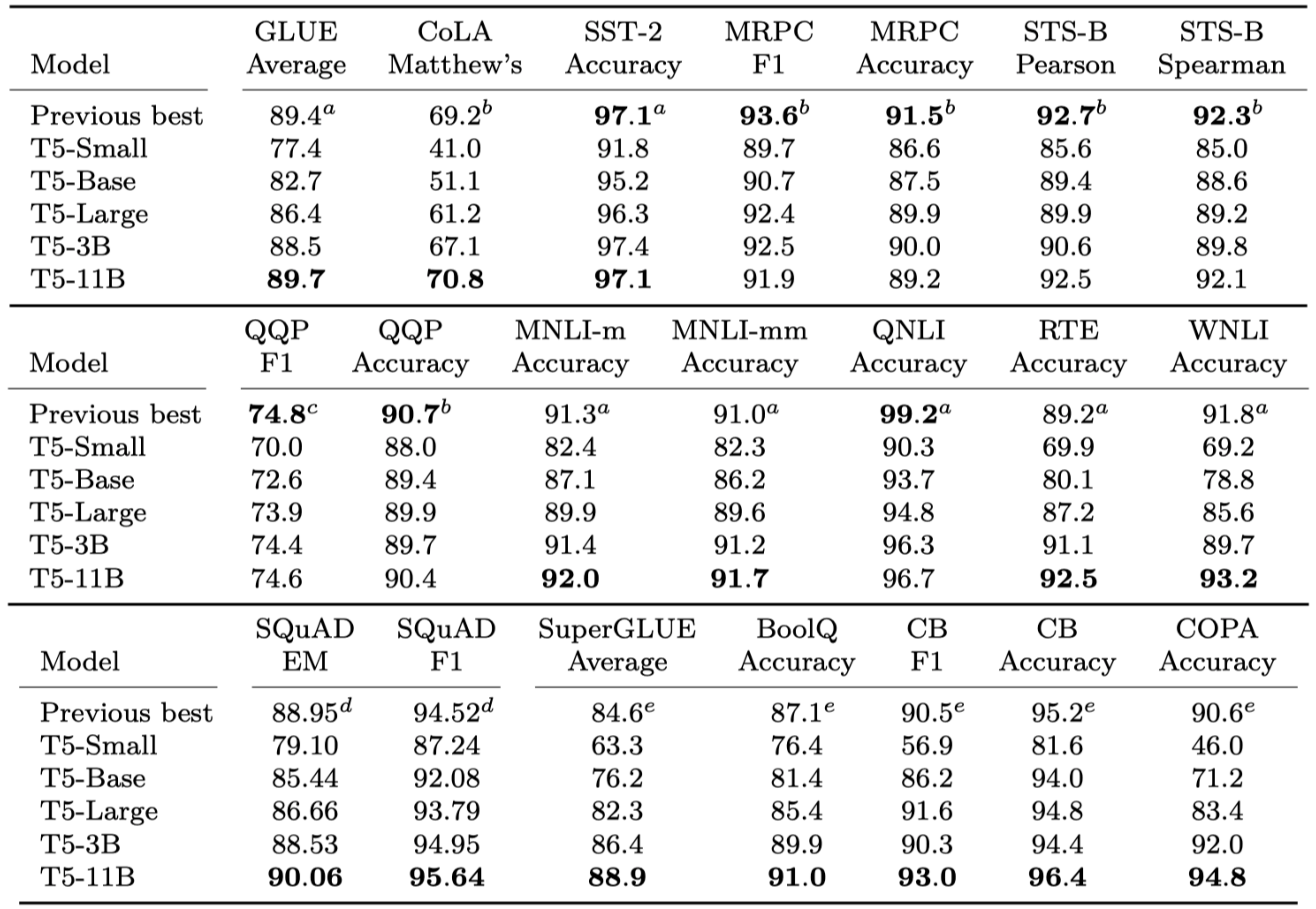
- **训练阶段**
  - 提出了一种基于span-corruption的无监督训练任务
  - 输入：对句子中的若干文本片段进行mask
  - 输出：预测被mask的文本片段


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>

Raffel et al., arXiv 2019. *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*

T5

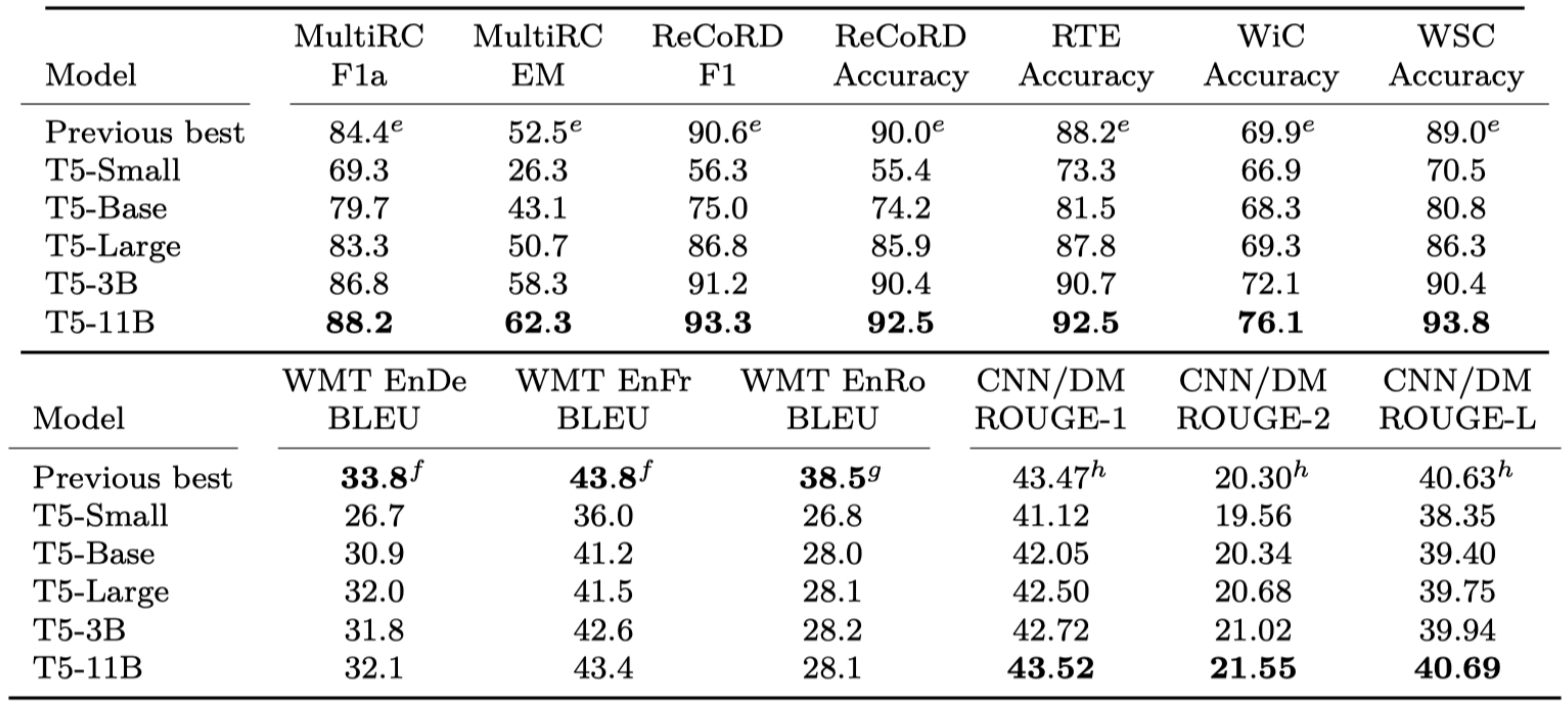
- **实验结果**
  - Small: 60M, base: 220M, large: 770M, 3B, 11B



Model	GLUE Average	CoLA Matthew's	SST-2 Accuracy	MRPC F1	MRPC Accuracy	STS-B Pearson	STS-B Spearman
Previous best	89.4[a]	69.2[b]	**97.1**[a]	**93.6**[b]	**91.5**[b]	**92.7**[b]	**92.3**[b]
T5-Small	77.4	41.0	91.8	89.7	86.6	85.6	85.0
T5-Base	82.7	51.1	95.2	90.7	87.5	89.4	88.6
T5-Large	86.4	61.2	96.3	92.4	89.9	89.9	89.2
T5-3B	88.5	67.1	97.4	92.5	90.0	90.6	89.8
T5-11B	**89.7**	**70.8**	**97.1**	91.9	89.2	92.5	92.1

Model	QQP F1	QQP Accuracy	MNLI-m Accuracy	MNLI-mm Accuracy	QNLI Accuracy	RTE Accuracy	WNLI Accuracy
Previous best	**74.8**[c]	**90.7**[b]	91.3[a]	91.0[a]	**99.2**[a]	89.2[a]	91.8[a]
T5-Small	70.0	88.0	82.4	82.3	90.3	69.9	69.2
T5-Base	72.6	89.4	87.1	86.2	93.7	80.1	78.8
T5-Large	73.9	89.9	89.9	89.6	94.8	87.2	85.6
T5-3B	74.4	89.7	91.4	91.2	96.3	91.1	89.7
T5-11B	74.6	90.4	**92.0**	**91.7**	96.7	**92.5**	**93.2**

Model	SQuAD EM	SQuAD F1	SuperGLUE Average	BoolQ Accuracy	CB F1	CB Accuracy	COPA Accuracy
Previous best	88.95[d]	94.52[d]	84.6[e]	87.1[e]	90.5[e]	95.2[e]	90.6[e]
T5-Small	79.10	87.24	63.3	76.4	56.9	81.6	46.0
T5-Base	85.44	92.08	76.2	81.4	86.2	94.0	71.2
T5-Large	86.66	93.79	82.3	85.4	91.6	94.8	83.4
T5-3B	88.53	94.95	86.4	89.9	90.3	94.4	92.0
T5-11B	**90.06**	**95.64**	**88.9**	**91.0**	**93.0**	**96.4**	**94.8**



Model	MultiRC F1a	MultiRC EM	ReCoRD F1	ReCoRD Accuracy	RTE Accuracy	WiC Accuracy	WSC Accuracy
Previous best	84.4[e]	52.5[e]	90.6[e]	90.0[e]	88.2[e]	69.9[e]	89.0[e]
T5-Small	69.3	26.3	56.3	55.4	73.3	66.9	70.5
T5-Base	79.7	43.1	75.0	74.2	81.5	68.3	80.8
T5-Large	83.3	50.7	86.8	85.9	87.8	69.3	86.3
T5-3B	86.8	58.3	91.2	90.4	90.7	72.1	90.4
T5-11B	**88.2**	**62.3**	**93.3**	**92.5**	**92.5**	**76.1**	**93.8**

Model	WMT EnDe BLEU	WMT EnFr BLEU	WMT EnRo BLEU	CNN/DM ROUGE-1	CNN/DM ROUGE-2	CNN/DM ROUGE-L
Previous best	**33.8**[f]	**43.8**[f]	**38.5**[g]	43.47[h]	20.30[h]	40.63[h]
T5-Small	26.7	36.0	26.8	41.12	19.56	38.35
T5-Base	30.9	41.2	28.0	42.05	20.34	39.40
T5-Large	32.0	41.5	28.1	42.50	20.68	39.75
T5-3B	31.8	42.6	28.2	42.72	21.02	39.94
T5-11B	32.1	43.4	28.1	**43.52**	**21.55**	**40.69**

Raffel et al., arXiv 2019. *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*

小结

	GPT	BERT	XLNet	RoBERTa	ALBERT	ELECTRA	MacBERT	T5
Type	AR	AE	AR	AE	AE	AE	AE	AE+AR
Embedding	T	T/S/P	T/S/P	T/S/P	T/S/P	T/S/P	T/S/P	T/S/P
Masking	/	T	/	T	T	T	WWM+NM	T
LM Task	LM	MLM	PLM	MLM	MLM	Gen-Dis	Mac	Span-Corruption
Paired Task	/	NSP	/	/	SOP	/	SOP	/
Data Source	BC	BC+Wiki	BC+Wiki+Giga5 +CW+CC	BC+Wiki+CCNews +OWT+Stories	BC+Wiki	BC+Wiki+Giga5 +CW+CC	Wiki, News, Baike	C4
Data Size	/	/	110G	160G	16G	~110G	~20G	750G
Tokenization	BPE	WordPiece	SentencePiece	BPE	SentencePiece	WordPiece	WordPiece	SentencePiece
Tokens	800M	3300M	32.89B	/	/	~33B	/	/
Vocabulary	40,000	30,522	32,000	50,000	30,000	30,522	21,128	32,000
MaxSeqLen	512	512	512	512	512	512	512	512
Layers	12	12/24	12/24	12/24	12/24/24/12	12/12/24	12/24	6/12/24

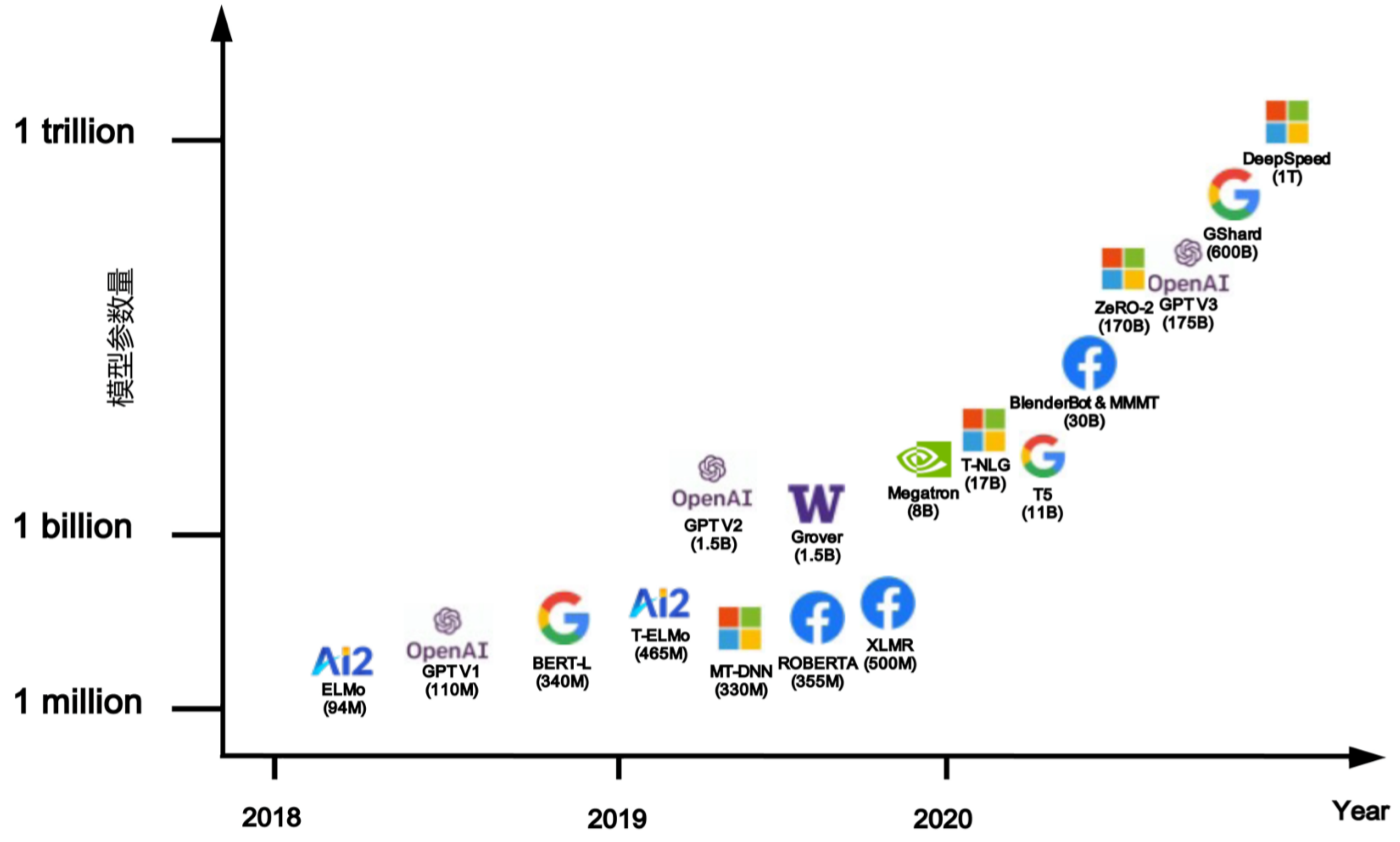
面向预训练语言模型的知识蒸馏

KNOWLEDGE DISTILLATION FOR PRE-TRAINED LANGUAGE MODELS

1. DistilBERT
2. TinyBERT
3. MobileBERT
4. HFL 哈工大讯飞联合实验室 TextBrewer

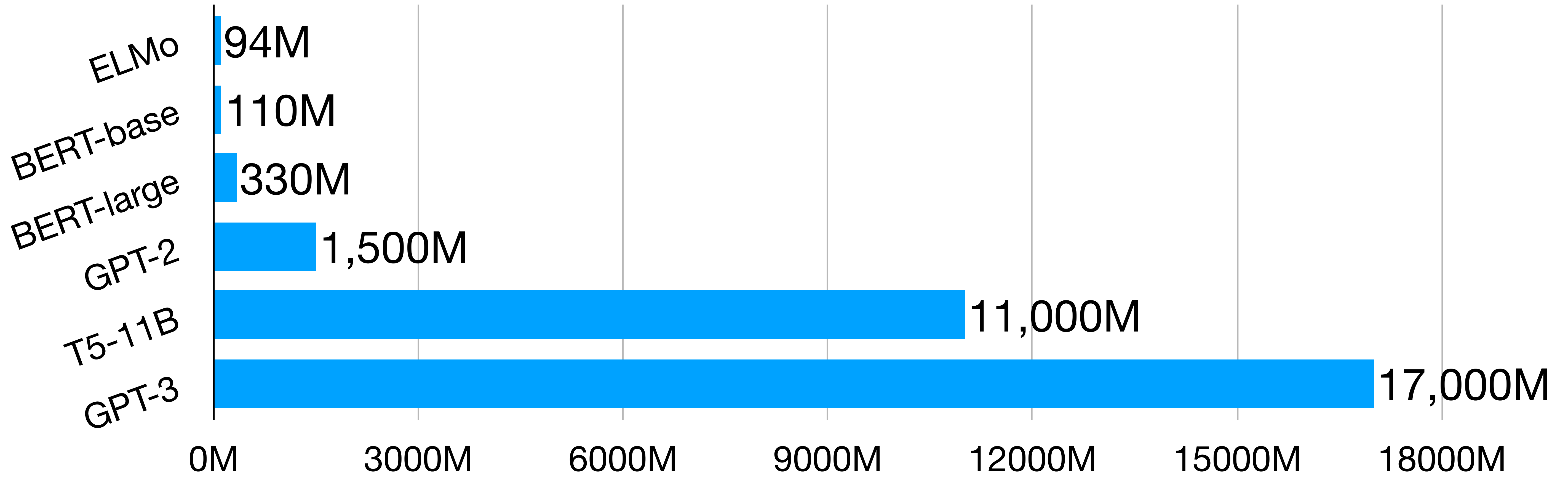
知识蒸馏

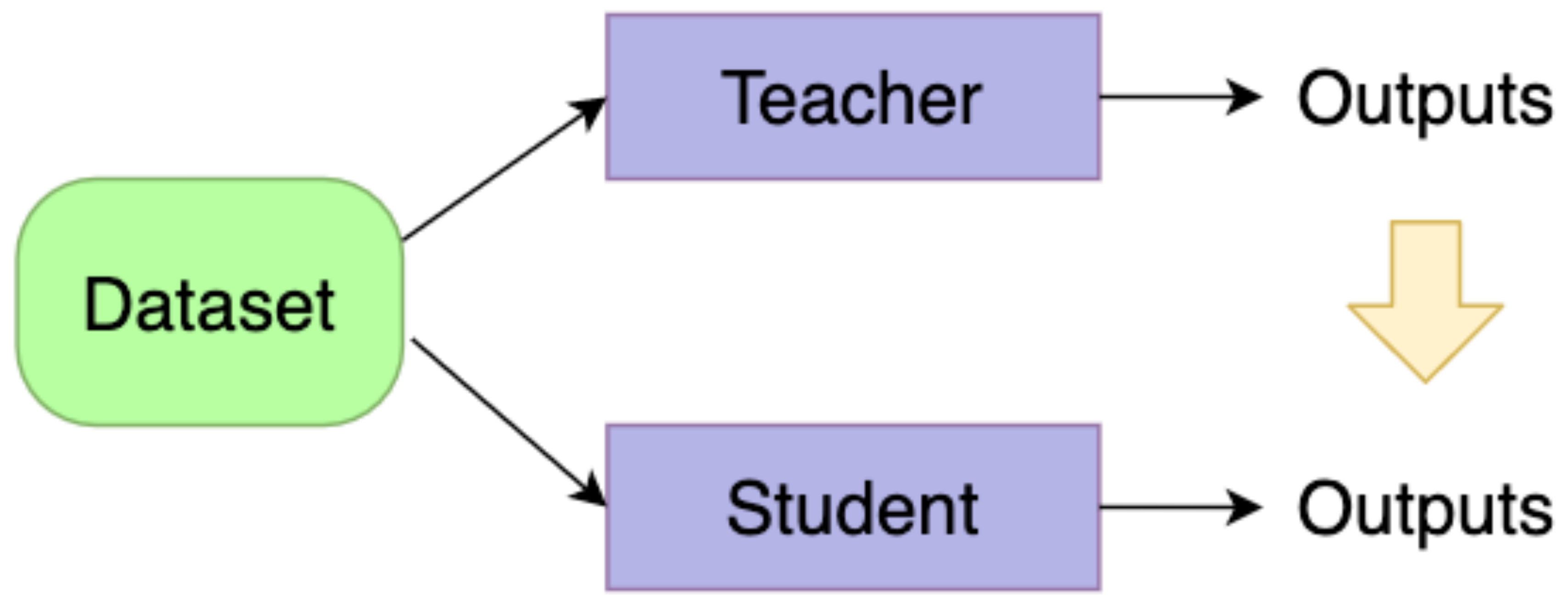
- 预训练模型参数量趋势



知识蒸馏

- **为什么需要模型压缩与知识蒸馏技术?**
  - 预训练模型通常需要占用很大的空间，并且训练和推断时间也很慢
  - 直接在实际产品或应用中使用预训练模型难以满足时间和空间需求
  - **知识蒸馏技术**可以在不损失或少量损失性能的情况下，将大模型的知识迁移到小模型，从而提升推断速度

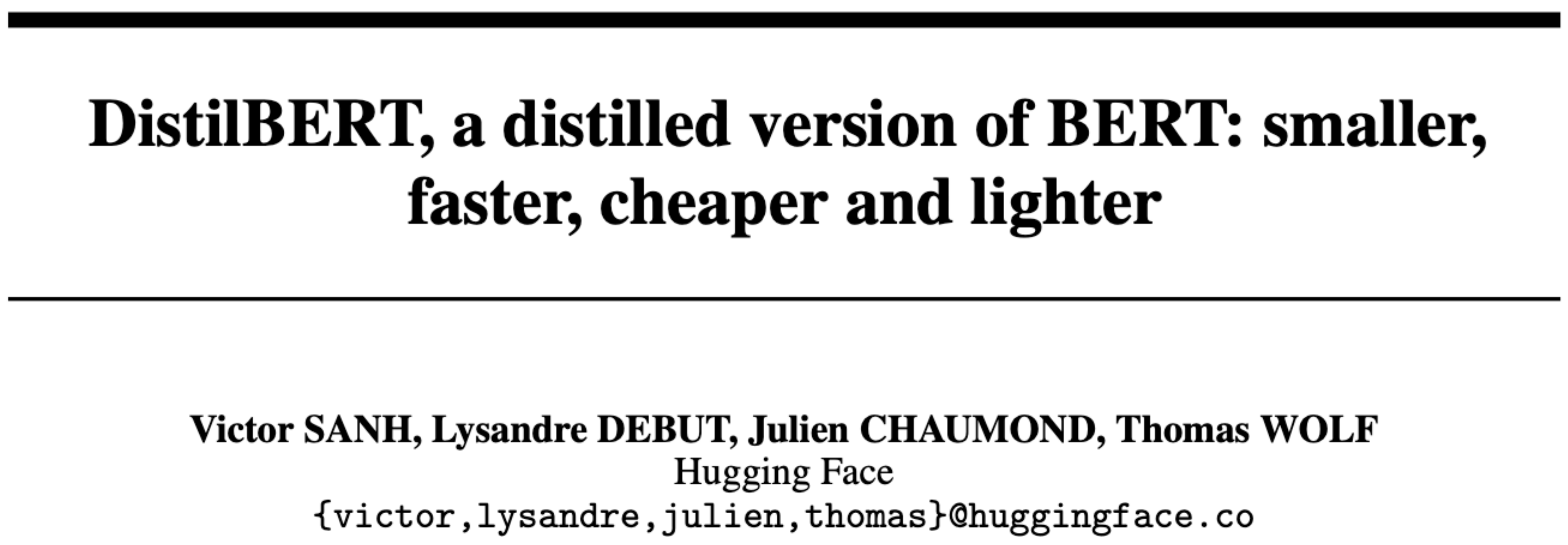




知识蒸馏过程：“老师教学生”

知识蒸馏

- **DistilBERT**
  - 一个经过知识蒸馏的BERT模型（6层）
  - 特点：通用、任务无关
  - 相比原始BERT-base，小40%、快60%、在自然语言理解任务上可达到原模型的97%

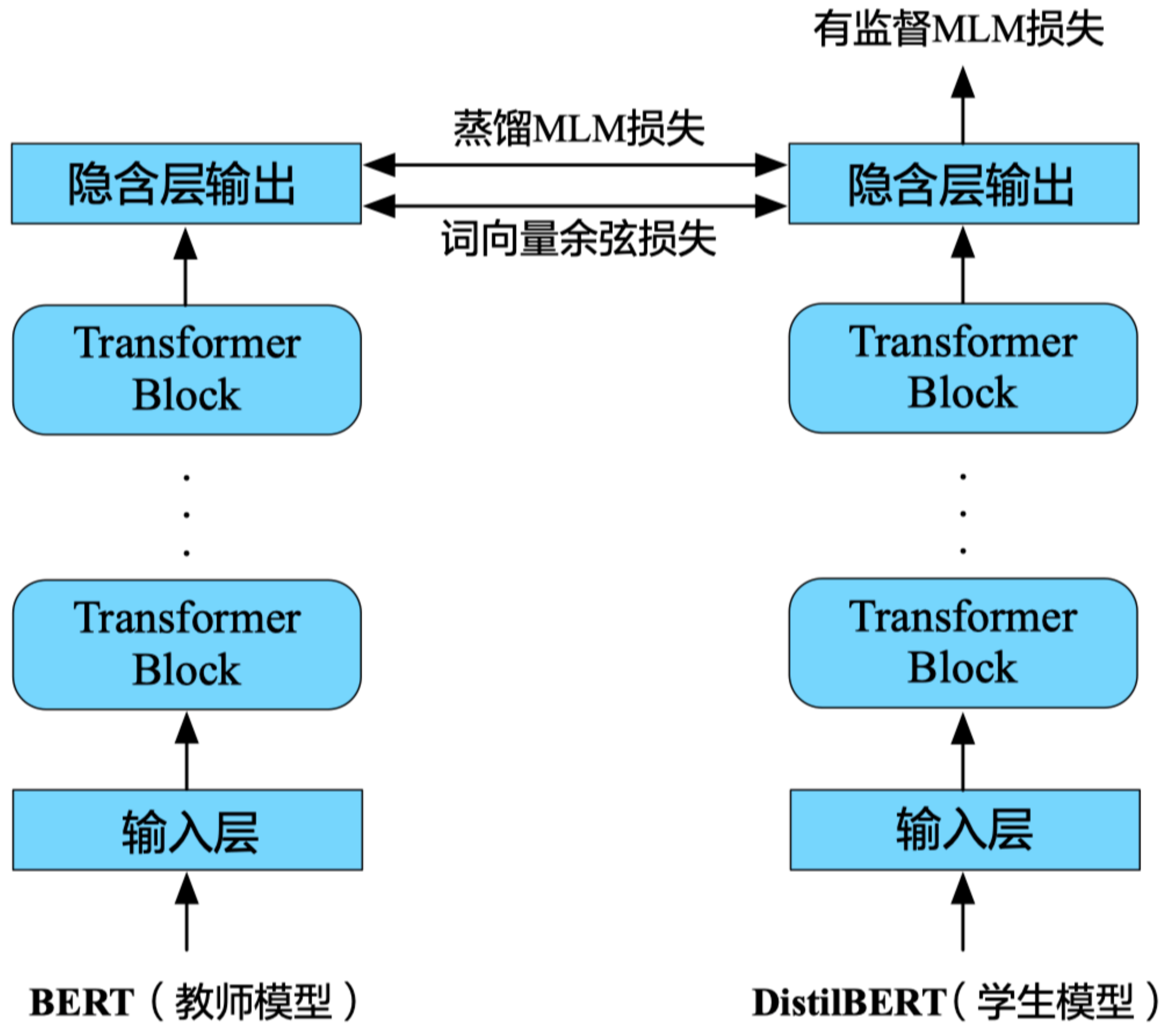


DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter

Victor SANH, Lysandre DEBUT, Julien CHAUMOND, Thomas WOLF
Hugging Face
`{victor,lysandre,julien,thomas}@huggingface.co`

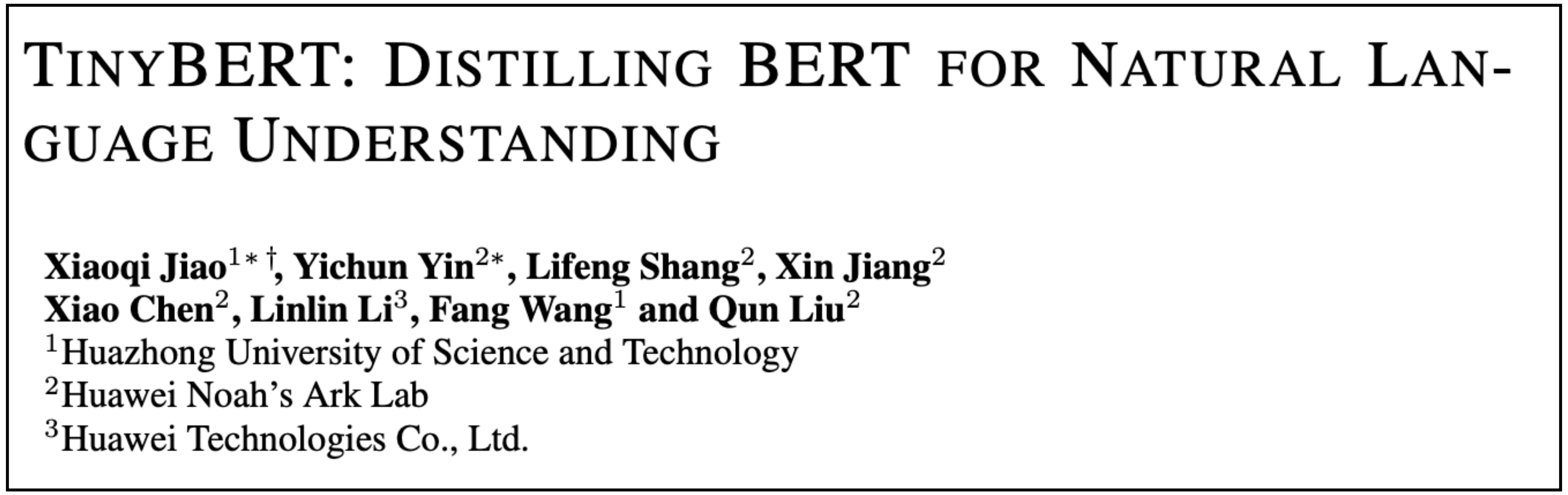
知识蒸馏

- **DistilBERT**
  - 学生模型（即DistilBERT）是一个通用的6层Transformer结构
  - 使用MLM预训练任务进行知识蒸馏（无NSP）
  - 训练目标由以下三部分组成
    - 由数据集自带的硬标签（hard-labels）计算的有监督MLM损失
    - 由教师模型提供的软标签（soft-labels）计算的蒸馏MLM损失
    - 教师模型和学生模型隐层输出之间的余弦相似度损失



知识蒸馏

- **TinyBERT**
  - 提出了一种对BERT不同层进行匹配的蒸馏策略
  - 提出了两阶段蒸馏策略，在预训练和精调阶段均进行知识蒸馏
  - TinyBERT能达到教师模型BERT-base的96%的效果（GLUE），大小只有教师的13.3%

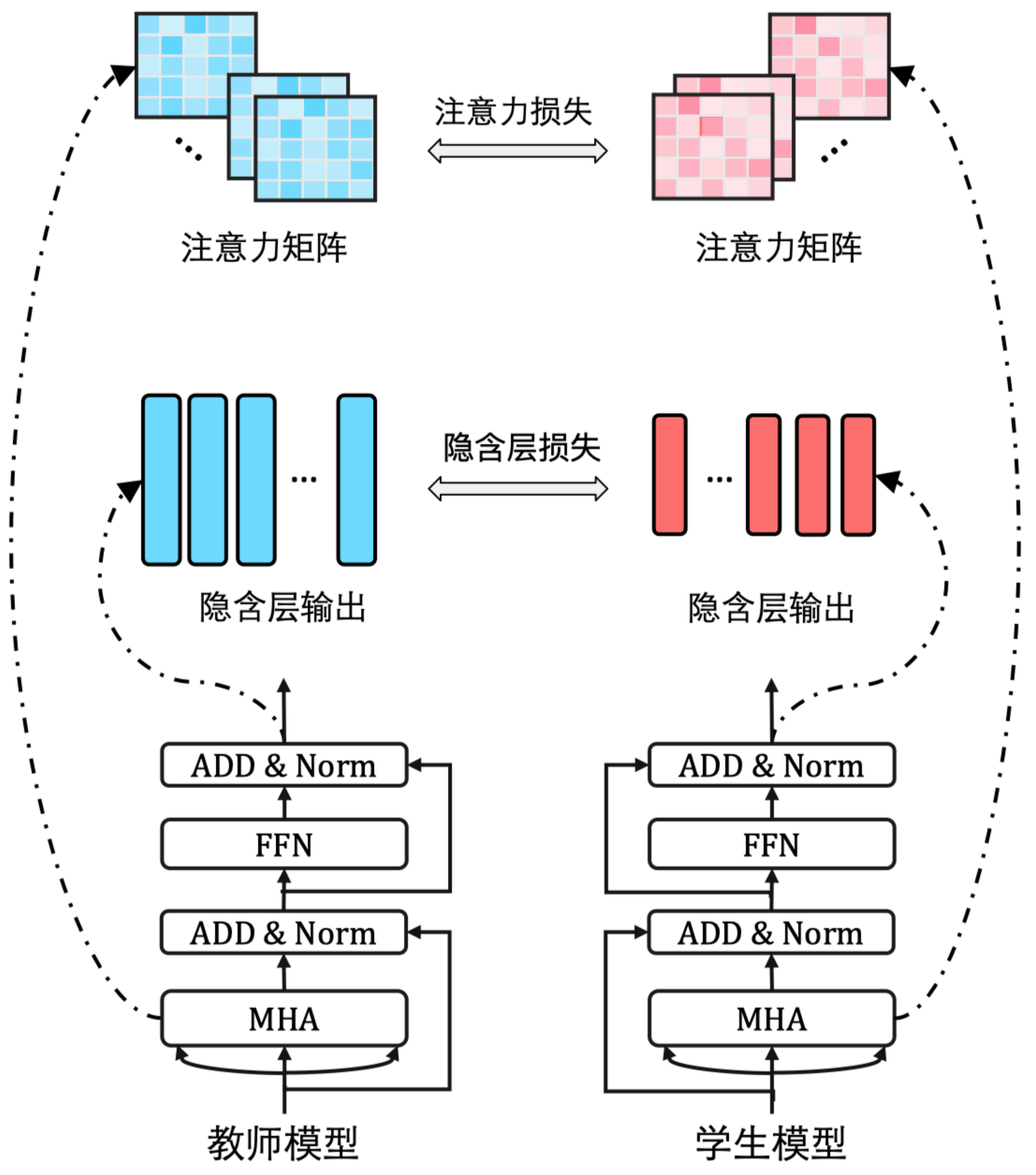


TINYBERT: DISTILLING BERT FOR NATURAL LANGUAGE UNDERSTANDING

Xiaoqi Jiao[1*†], **Yichun Yin**[2*], **Lifeng Shang**[2], **Xin Jiang**[2]
Xiao Chen[2], **Linlin Li**[3], **Fang Wang**[1] **and Qun Liu**[2]
[1]Huazhong University of Science and Technology
[2]Huawei Noah's Ark Lab
[3]Huawei Technologies Co., Ltd.

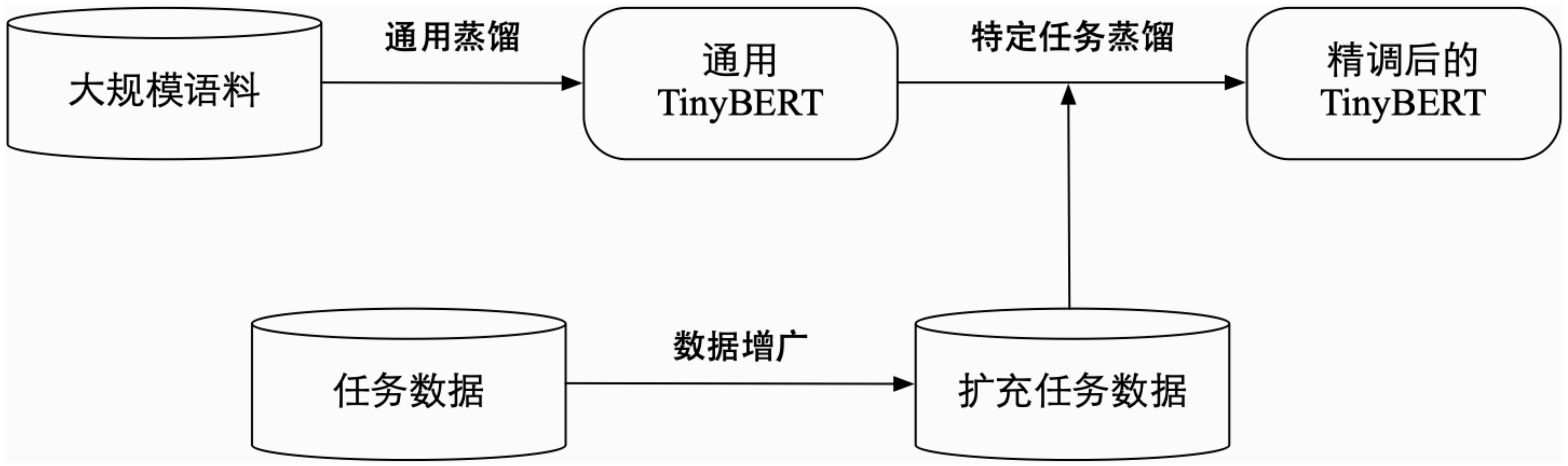
知识蒸馏

- **TinyBERT**
  - 蒸馏损失由3部分组成：词向量损失、中间层损失、预测层损失
  - 词向量层损失：计算教师和学生模型的词向量均方误差
  - 中间层损失
    - 计算教师模型第 i 层与学生模型第 j 层隐层输出（或注意力矩阵）的均方误差
    - 文中使用匹配函数为$g(i) = 3i$
    - 例如：教师模型第3层与学生模型第1层
  - 输出层损失：计算软标签损失



知识蒸馏

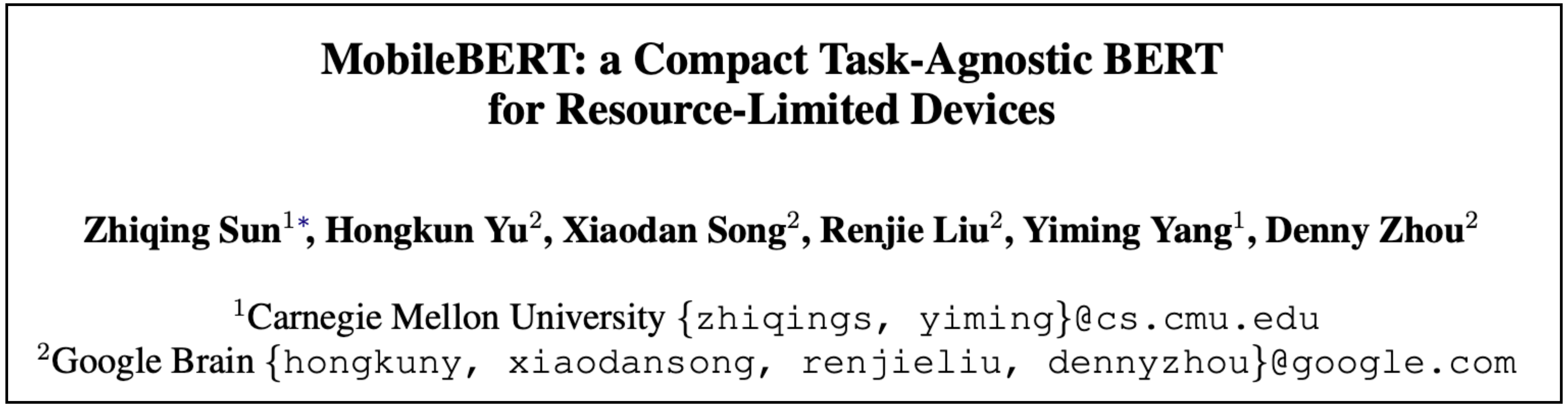
- **TinyBERT: 两段式蒸馏**
  - 通用蒸馏：利用原始BERT作为教师，并使用大规模文本训练MLM任务
  - 特定任务蒸馏：使用精调过的BERT作为教师，使用数据增广后的任务数据进行蒸馏



*Jiao et al., **Findings of EMNLP 2020.** TinyBERT: Distilling BERT for Natural Language Understanding*

知识蒸馏

- **MobileBERT**
  - MobileBERT与BERT-large深度相同但更“苗条”
  - 在自注意力和前馈神经网络的设计上也有一定的改进
  - 能够达到教师模型（BERT-base）99.2%的性能效果（以GLUE为测试基准），推理速度快5.5倍，参数量降低至23.2%



MobileBERT: a Compact Task-Agnostic BERT for Resource-Limited Devices

Zhiqing Sun[1*], Hongkun Yu[2], Xiaodan Song[2], Renjie Liu[2], Yiming Yang[1], Denny Zhou[2]

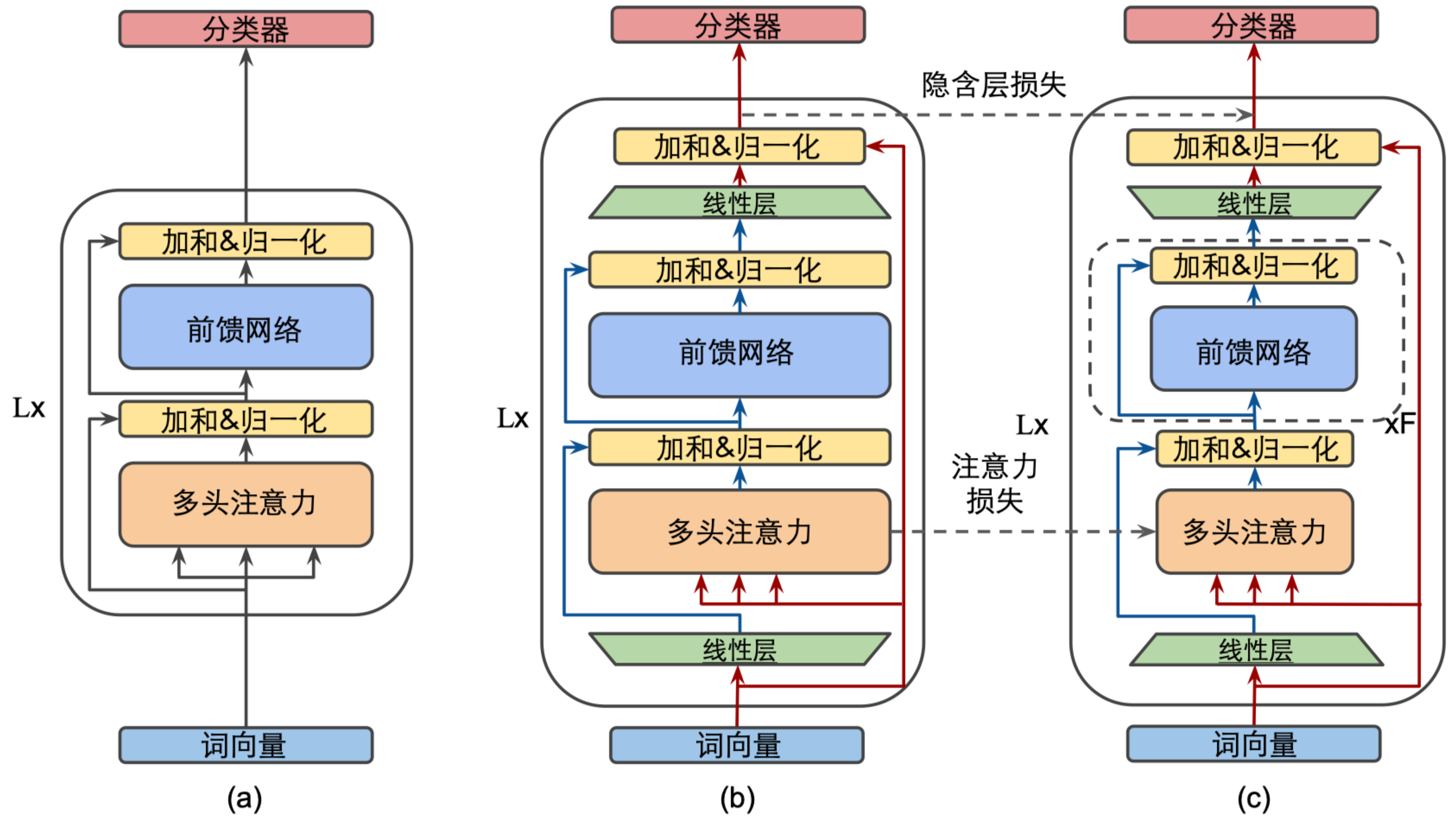
[1]Carnegie Mellon University {zhiqings, yiming}@cs.cmu.edu
[2]Google Brain {hongkuny, xiaodansong, renjieliu, dennyzhou}@google.com

Sun et al., ACL 2020. *MobileBERT: a Compact Task-Agnostic BERT for Resource-Limited Devices*

知识蒸馏

- **MobileBERT**
  - 教师模型：训练一个24层BERT（称之为IB-BERT）
  - 训练目标
    - 普通的MLM蒸馏损失
    - 隐层匹配损失
    - 注意力匹配损失与TinyBERT类似，但使用了KL散度进行度量

$$\mathcal{L}^{\text{att}}(i,j) = \frac{1}{K}\sum_{k=1}^{K}\text{KL}(\boldsymbol{A}^{\text{t}^{[i]}} \| \boldsymbol{A}^{\text{s}^{[j]}})$$

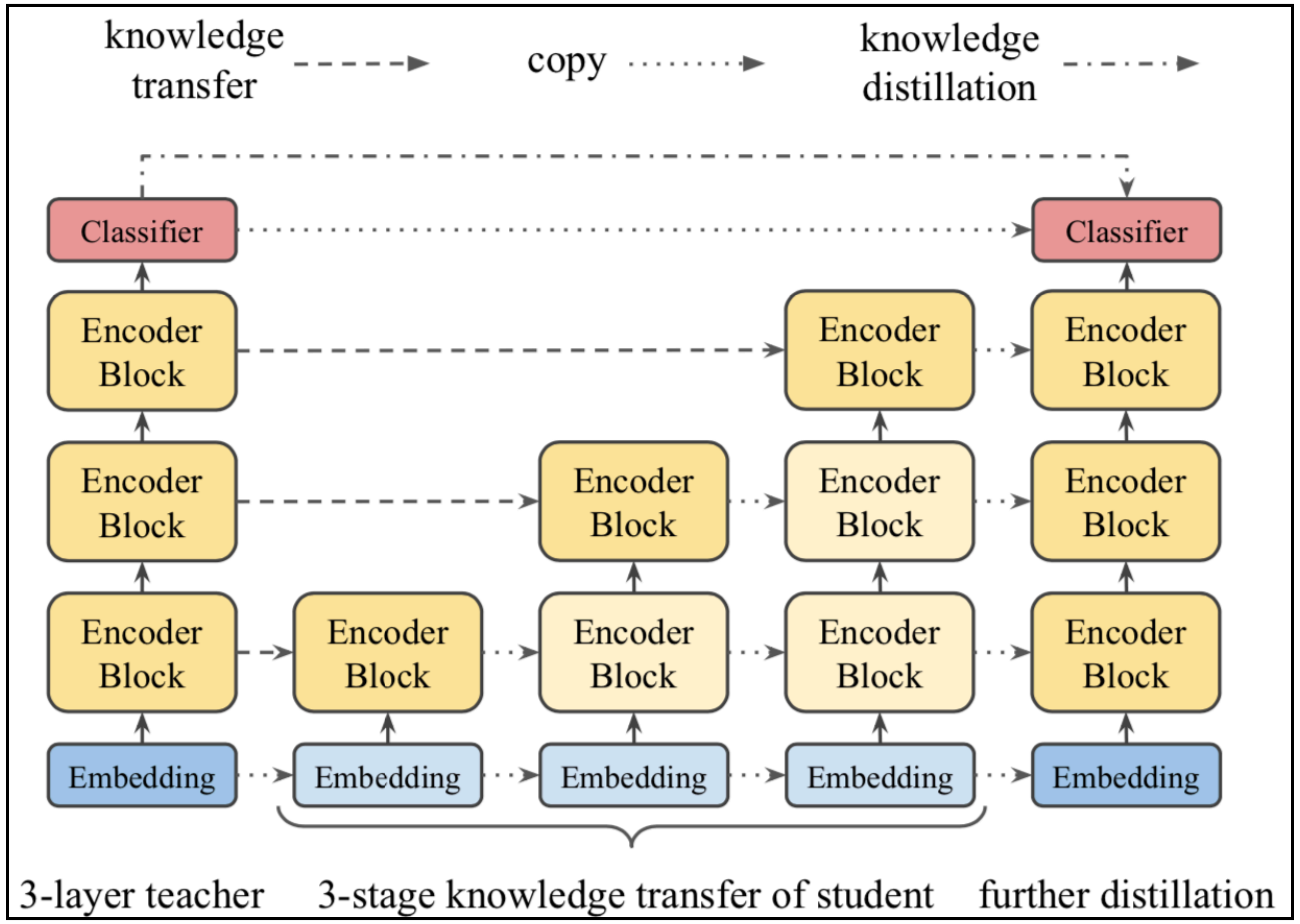


BERT　　IB-BERT　　MobileBERT

***Sun et al., ACL 2020.** MobileBERT: a Compact Task-Agnostic BERT for Resource-Limited Devices*

知识蒸馏

- **MobileBERT**
  - 渐进式知识迁移（Progressive Knowledge Transfer）
    - 词向量层和最终分类输出层的权重是直接从教师模型拷贝至学生模型的，始终不参与参数更新
    - 对于中间的 Transformer 层，采用了渐进的方式逐步训练
    - 当学生模型学习教师模型的第 i 层时，学生模型中所有小于 i 层的权重均不参与更新

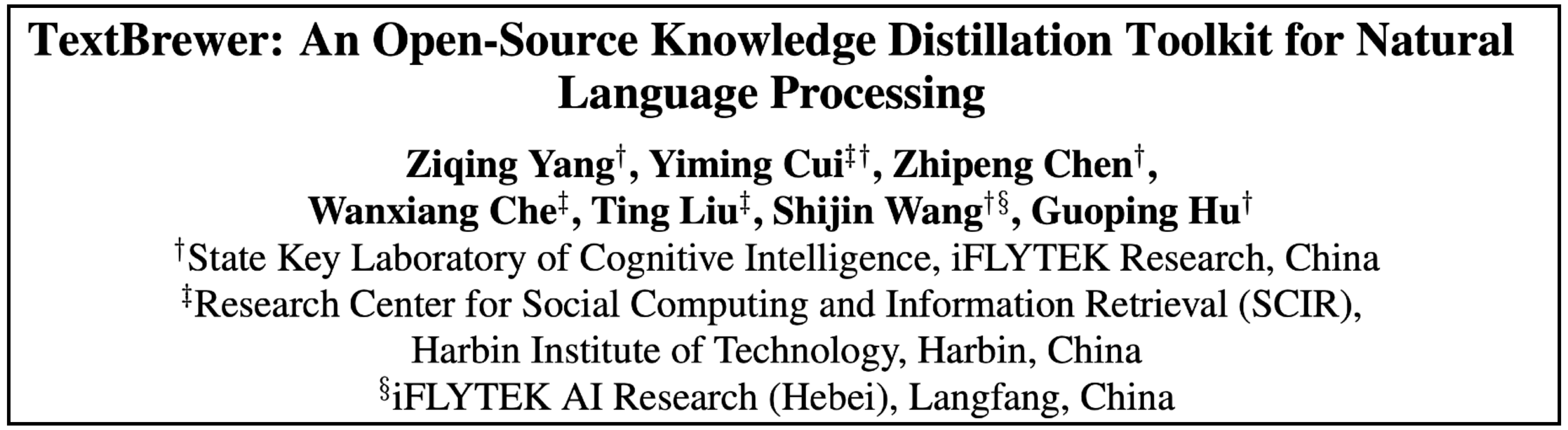


Sun et al., ACL 2020. *MobileBERT: a Compact Task-Agnostic BERT for Resource-Limited Devices*

TextBrewer

- **TextBrewer: An Open-Source Knowledge Distillation Toolkit**
  - 推出了首个面向自然语言处理领域的基于Pytorch的知识蒸馏工具包
  - 提供了方便、快捷、易用的知识蒸馏框架，少量性能损失换取大幅速度提升
  - http://textbrewer.hfl-rc.com/ 或通过 `pip install textbrewer` 安装

- **模型无关**：适用于多种模型结构（主要面向Transfomer结构）
- **方便灵活**：可自由组合多种蒸馏方法，支持增加自定义损失等模块
- **非侵入式**：无需对教师与学生模型本身结构进行修改
- **适用面广**：支持典型NLP任务，如文本分类、阅读理解、序列标注等



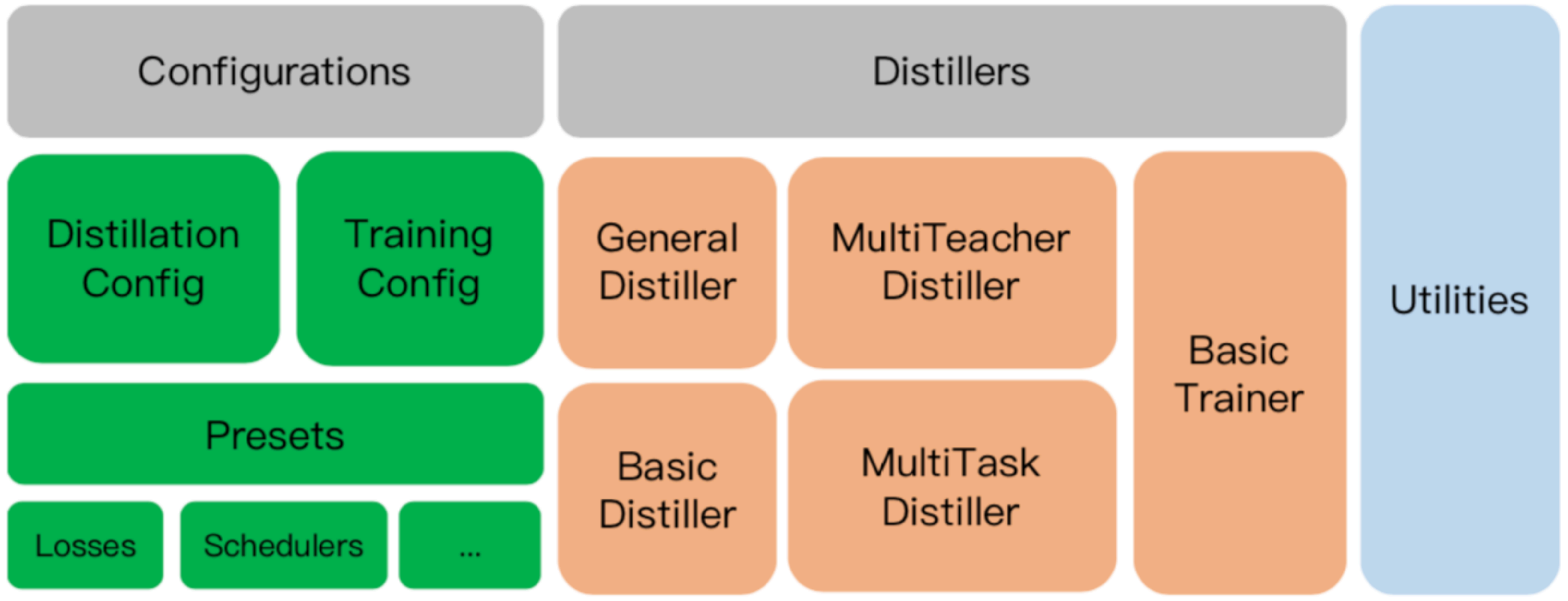
TextBrewer: An Open-Source Knowledge Distillation Toolkit for Natural Language Processing

Ziqing Yang[†], **Yiming Cui**[‡†], **Zhipeng Chen**[†],
Wanxiang Che[‡], **Ting Liu**[‡], **Shijin Wang**[†§], **Guoping Hu**[†]
[†]State Key Laboratory of Cognitive Intelligence, iFLYTEK Research, China
[‡]Research Center for Social Computing and Information Retrieval (SCIR),
Harbin Institute of Technology, Harbin, China
[§]iFLYTEK AI Research (Hebei), Langfang, China

__Yang et al., ACL 2020.__ TextBrewer: An Open-Source Knowledge Distillation Toolkit for Natural Language Processing

TextBrewer

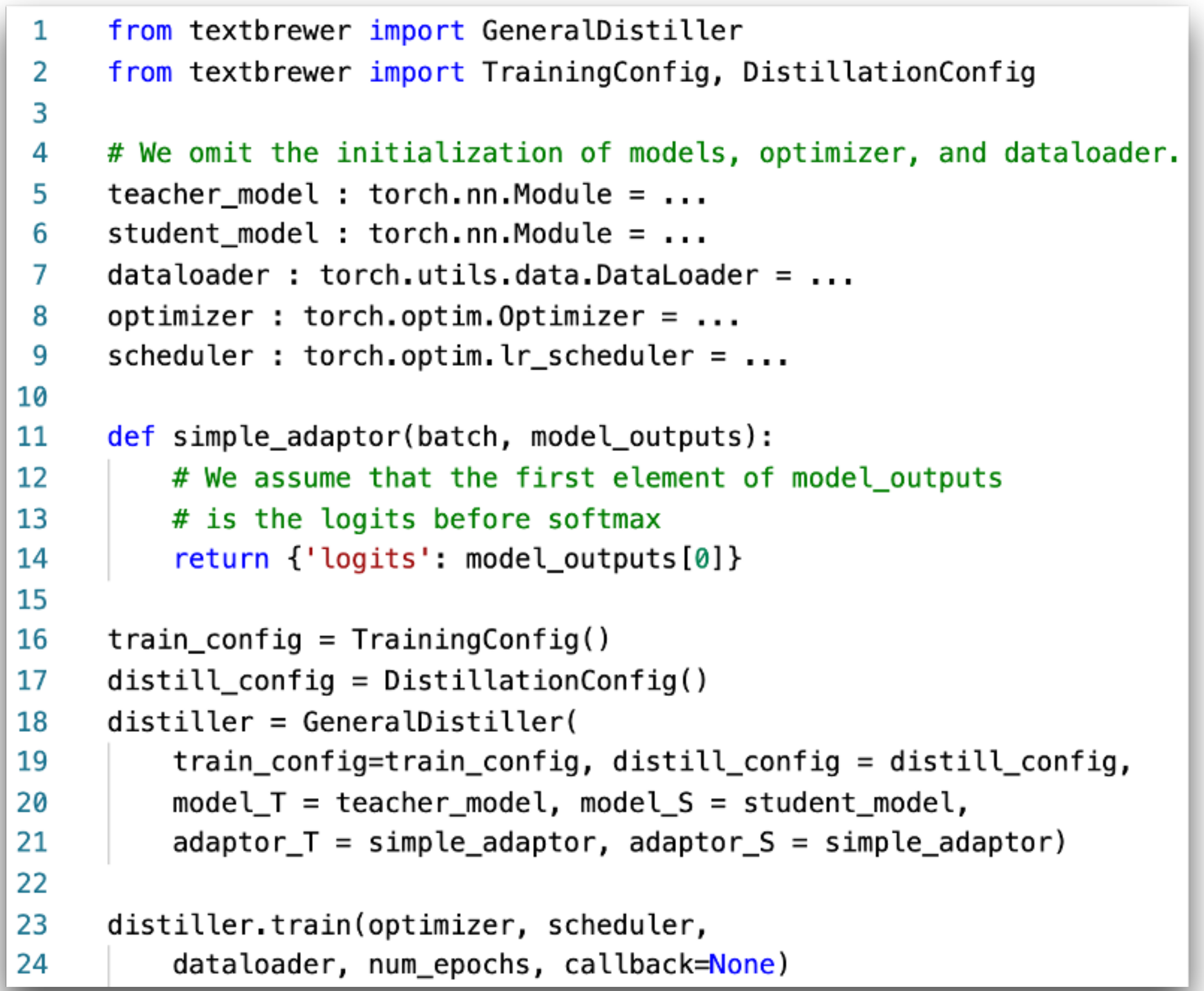
- **整体架构**
  - Distillers：用于执行实际的知识蒸馏工作
  - Configurations：为Distillers提供必要的配置
  - Utilities：包含一些辅助的功能，如模型参数统计等



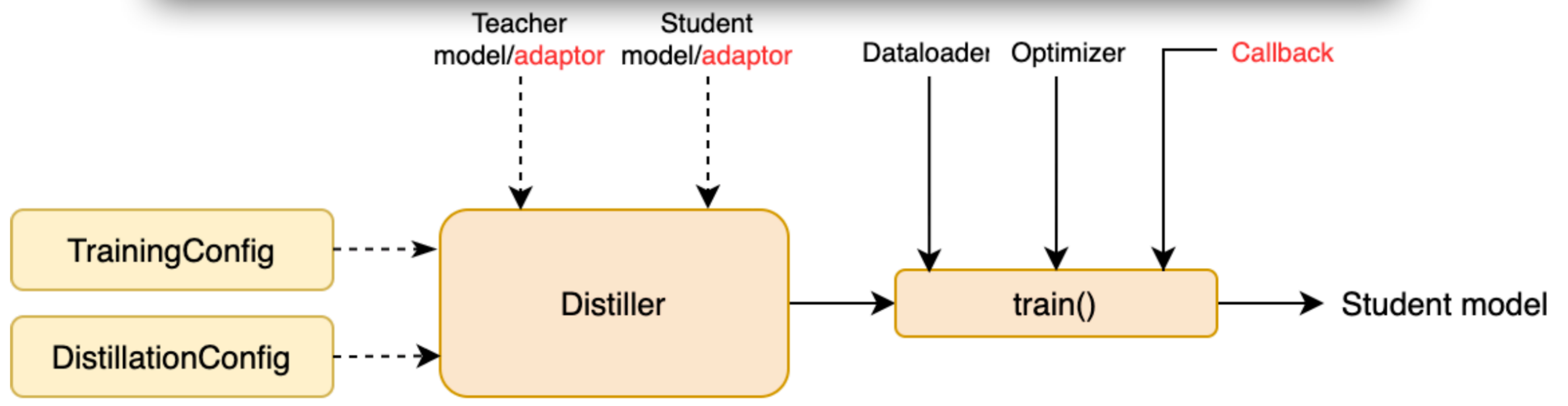
***Yang et al., ACL 2020.** TextBrewer: An Open-Source Knowledge Distillation Toolkit for Natural Language Processing*

TextBrewer

- **工作流程**
  - **第一步：开始蒸馏之前的准备工作**
    - 训练教师模型，定义并初始化学生模型
    - 构造蒸馏用数据集的DataLoader
  - **第二步：知识蒸馏**
    - 初始化Distiller，构造训练配置和蒸馏配置
    - 定义adaptors和callback，分别用于适配模型输入输出和训练过程中的回调
    - 调用Distiller的train方法开始蒸馏
  - 一共只需20行左右的代码！



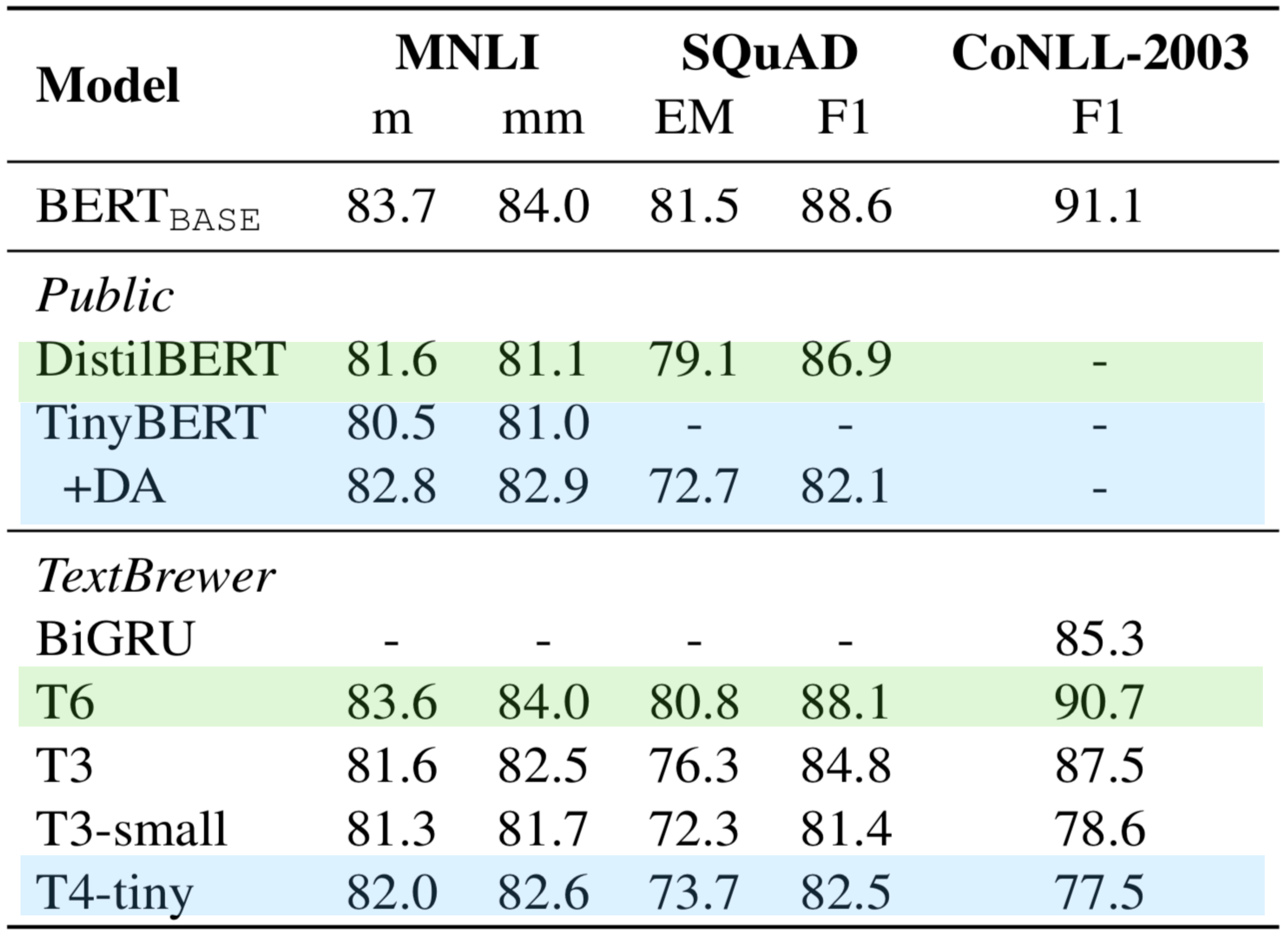
```python
from textbrewer import GeneralDistiller
from textbrewer import TrainingConfig, DistillationConfig

# We omit the initialization of models, optimizer, and dataloader.
teacher_model : torch.nn.Module = ...
student_model : torch.nn.Module = ...
dataloader : torch.utils.data.DataLoader = ...
optimizer : torch.optim.Optimizer = ...
scheduler : torch.optim.lr_scheduler = ...

def simple_adaptor(batch, model_outputs):
    # We assume that the first element of model_outputs
    # is the logits before softmax
    return {'logits': model_outputs[0]}

train_config = TrainingConfig()
distill_config = DistillationConfig()
distiller = GeneralDistiller(
    train_config=train_config, distill_config = distill_config,
    model_T = teacher_model, model_S = student_model,
    adaptor_T = simple_adaptor, adaptor_S = simple_adaptor)

distiller.train(optimizer, scheduler,
    dataloader, num_epochs, callback=None)
```



Yang et al., ACL 2020. *TextBrewer: An Open-Source Knowledge Distillation Toolkit for Natural Language Processing*

TextBrewer

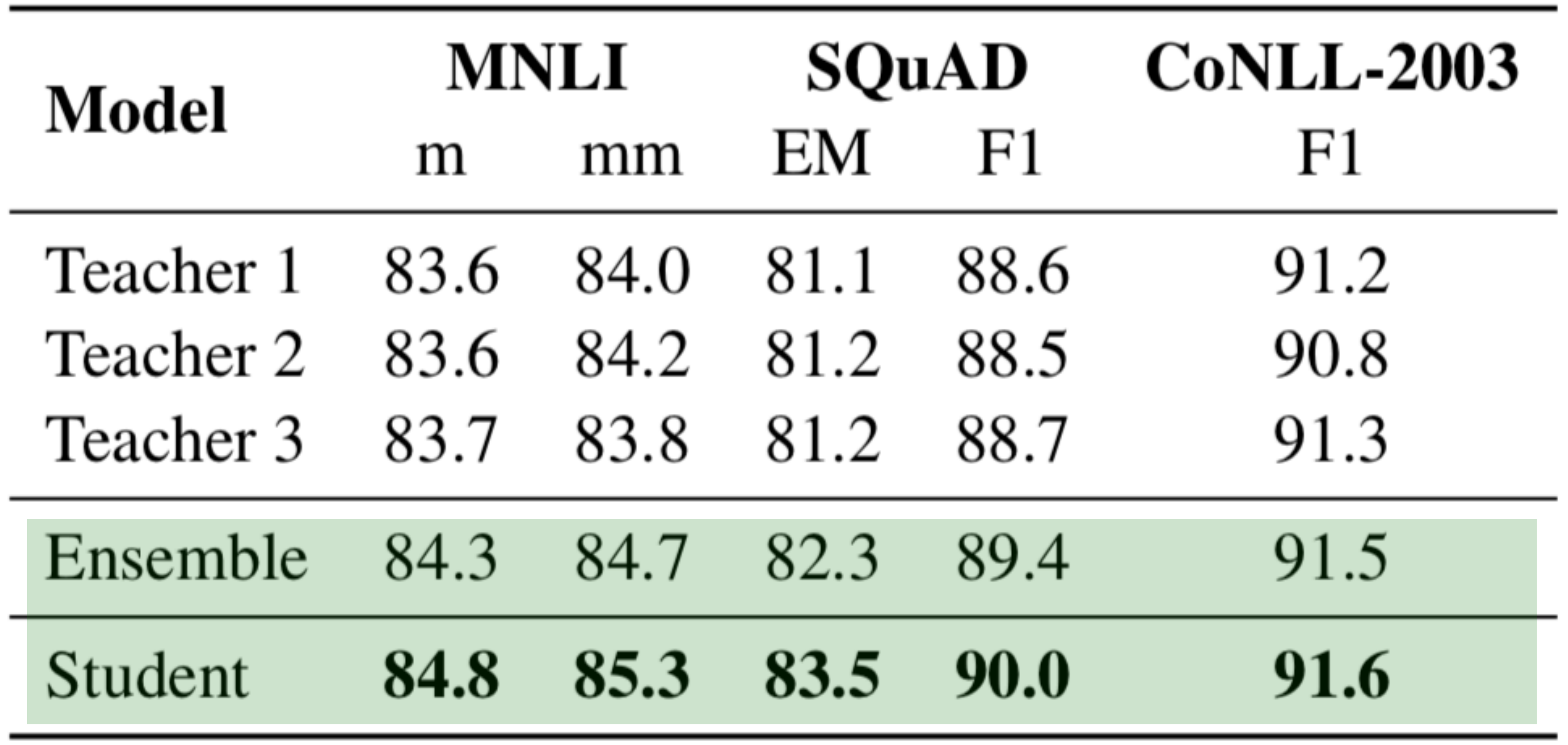
- **实验设置**
  - 教师模型: BERT-base（110M）
  - 学生模型: T6 (60%), T3 (41%), T3-small (16%), T4-tiny (same as TinyBERT, 13%)
- **单教师知识蒸馏**
  - T6结构可以达到教师模型效果的99%，且模型体积缩小至60%
  - T4-tiny知识蒸馏结果优于TinyBERT



Model	MNLI m	MNLI mm	SQuAD EM	SQuAD F1	CoNLL-2003 F1
BERT$_{\text{BASE}}$	83.7	84.0	81.5	88.6	91.1
Public					
DistilBERT	81.6	81.1	79.1	86.9	-
TinyBERT	80.5	81.0	-	-	-
+DA	82.8	82.9	72.7	82.1	-
TextBrewer					
BiGRU	-	-	-	-	85.3
T6	83.6	84.0	80.8	88.1	90.7
T3	81.6	82.5	76.3	84.8	87.5
T3-small	81.3	81.7	72.3	81.4	78.6
T4-tiny	82.0	82.6	73.7	82.5	77.5

TextBrewer

- **多教师知识蒸馏**
  - 所有模型使用同样的结构，即BERT-base
  - 蒸馏后的学生模型获得最优效果，且超过简单的模型融合（ensemble）



Model	MNLI		SQuAD		CoNLL-2003
	m	mm	EM	F1	F1
Teacher 1	83.6	84.0	81.1	88.6	91.2
Teacher 2	83.6	84.2	81.2	88.5	90.8
Teacher 3	83.7	83.8	81.2	88.7	91.3
Ensemble	84.3	84.7	82.3	89.4	91.5
Student	**84.8**	**85.3**	**83.5**	**90.0**	**91.6**

总结

SUMMARY

总结

- 基于上下文的语言模型：过渡产物，有时有用
  - CoVe, ELMo
- 预训练语言模型：经典系列，常读常新
  - GPT series (GPT, GPT-2, GPT-3)
  - BERT series (MLM, WWM, NM)
- 预训练语言模型进阶：效果更佳，推荐使用
  - XLNet, RoBERTa, ALBERT, ELECTRA, MacBERT, T5
- 面向预训练模型的知识蒸馏：高效利用，助推应用
  - DistilBERT, TinyBERT, MobileBERT, TextBrewer

资源推荐

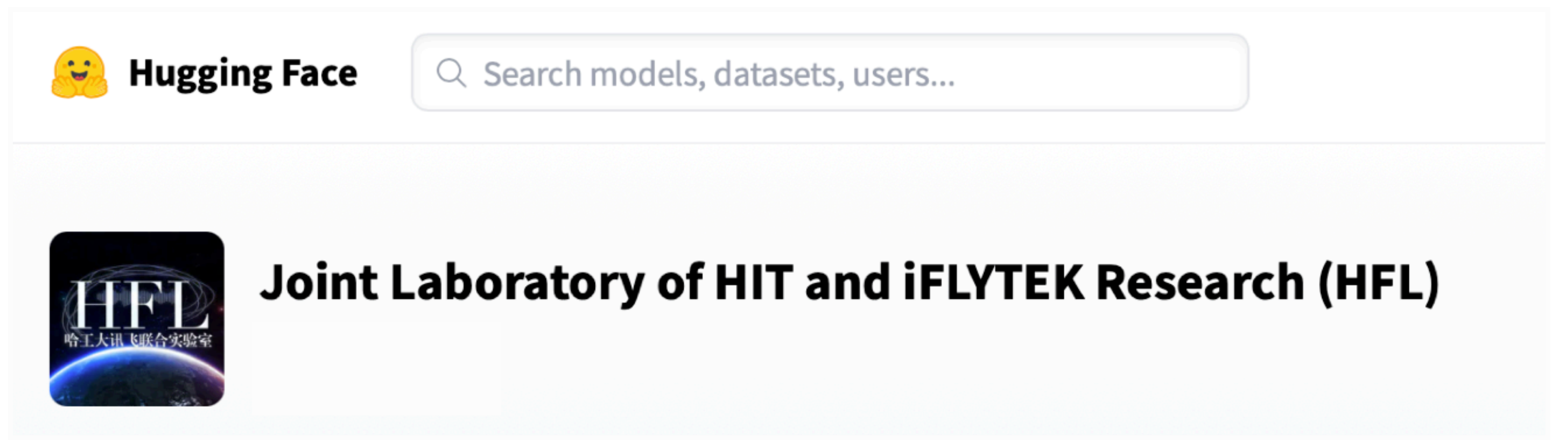
- **HFL @ GitHub: http://bert.hfl-rc.com**
  - BERT: BERT-wwm, BERT-wwm-ext
  - XLNet: XLNet-base, XLNet-mid
  - RoBERTa: RoBERTa-wwm-ext, RoBERTa-wwm-ext-large
  - RBT: RBT3, RBT4, RBT6, RBTL3
  - ELECTRA: ELECTRA-small, ELECTRA-small-ex, ELECTRA-base, ELECTRA-large, ELECTRA-legal
  - MacBERT: MacBERT-base, MacBERT-large

中文预训练系列模型获得 ***8,000+*** ⭐ *!!!*

资源推荐

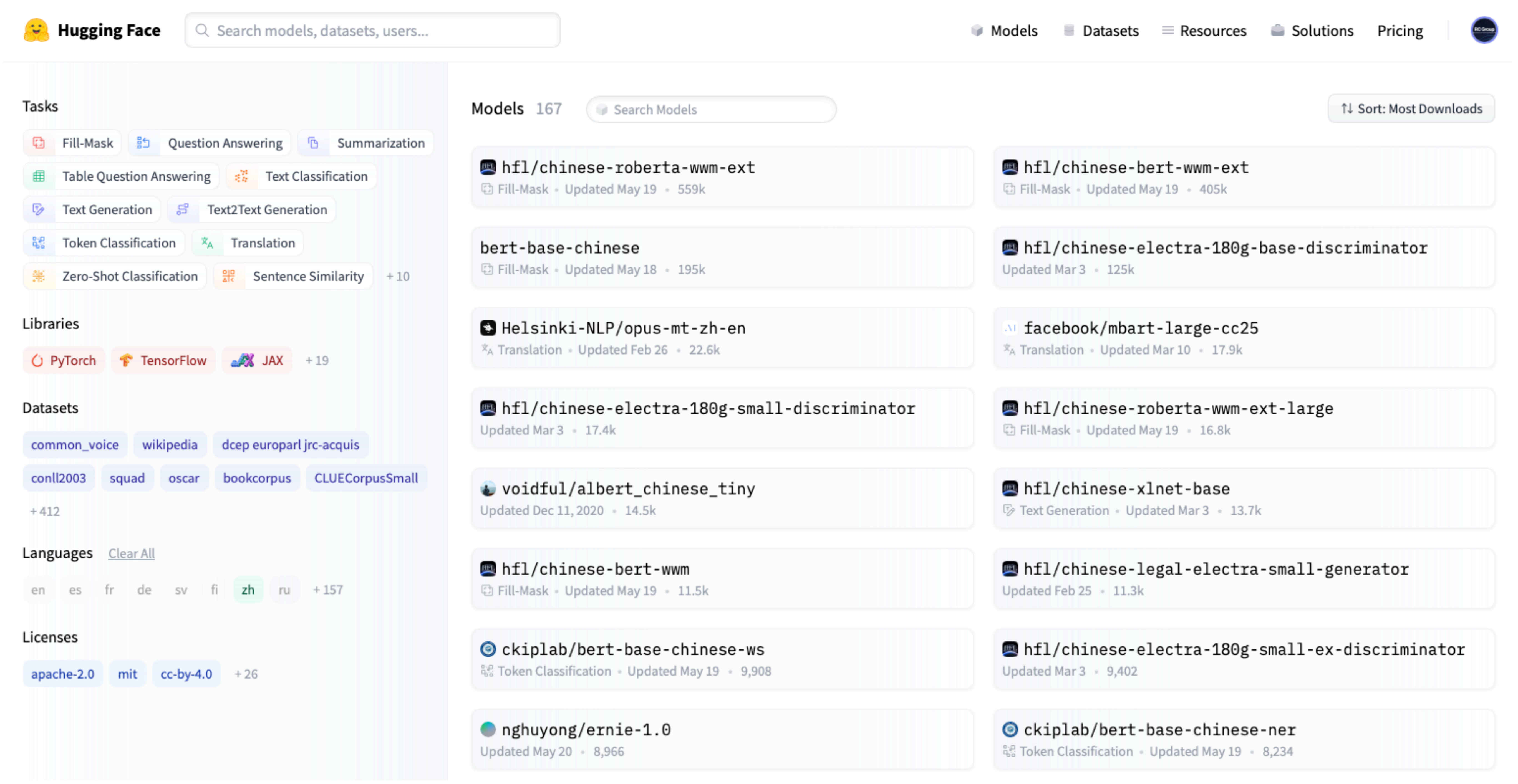
- **HFL @ Huggingface Model Hub: https://huggingface.co/hfl**
  - 快速加载各类中文预训练语言模型的首选方法
  - 只需一行代码即可加载预训练模型或者对应的分词器
  - 哈工大讯飞联合实验室已开源34个中文预训练模型，欢迎老师和同学使用

```
from transformers import BertTokenizer, BertModel

tokenizer = BertTokenizer.from_pretrained("hfl/chinese-bert-wwm")
model = BertModel.from_pretrained("hfl/chinese-bert-wwm")
```



Hugging Face　Search models, datasets, users...

Joint Laboratory of HIT and iFLYTEK Research (HFL)



Hugging Face　Search models, datasets, users...　Models　Datasets　Resources　Solutions　Pricing

Tasks
Fill-Mask　Question Answering　Summarization
Table Question Answering　Text Classification
Text Generation　Text2Text Generation
Token Classification　Translation
Zero-Shot Classification　Sentence Similarity　+ 10

Libraries
PyTorch　TensorFlow　JAX　+ 19

Datasets
common_voice　wikipedia　dcep europarl jrc-acquis
conll2003　squad　oscar　bookcorpus　CLUECorpusSmall
+ 412

Languages　Clear All
en　es　fr　de　sv　fi　zh　ru　+ 157

Licenses
apache-2.0　mit　cc-by-4.0　+ 26

Models 167　Search Models　Sort: Most Downloads

hfl/chinese-roberta-wwm-ext
Fill-Mask · Updated May 19 · 559k

hfl/chinese-bert-wwm-ext
Fill-Mask · Updated May 19 · 405k

bert-base-chinese
Fill-Mask · Updated May 18 · 195k

hfl/chinese-electra-180g-base-discriminator
Updated Mar 3 · 125k

Helsinki-NLP/opus-mt-zh-en
Translation · Updated Feb 26 · 22.6k

facebook/mbart-large-cc25
Translation · Updated Mar 10 · 17.9k

hfl/chinese-electra-180g-small-discriminator
Updated Mar 3 · 17.4k

hfl/chinese-roberta-wwm-ext-large
Fill-Mask · Updated May 19 · 16.8k

voidful/albert_chinese_tiny
Updated Dec 11, 2020 · 14.5k

hfl/chinese-xlnet-base
Text Generation · Updated Mar 3 · 13.7k

hfl/chinese-bert-wwm
Fill-Mask · Updated May 19 · 11.5k

hfl/chinese-legal-electra-small-generator
Updated Feb 25 · 11.3k

ckiplab/bert-base-chinese-ws
Token Classification · Updated May 19 · 9,908

hfl/chinese-electra-180g-small-ex-discriminator
Updated Mar 3 · 9,402

nghuyong/ernie-1.0
Updated May 20 · 8,966

ckiplab/bert-base-chinese-ner
Token Classification · Updated May 19 · 8,234

讲义下载暗号：
cipsatt2021

哈工大讯飞联合实验室
微信公众号

Thank You!

ymcui@ir.hit.edu.cn

me@ymcui.com