面向自然语言理解的预训练模型

PRE-TRAINED MODELS FOR NATURAL LANGUAGE UNDERSTANDING

崔一鸣

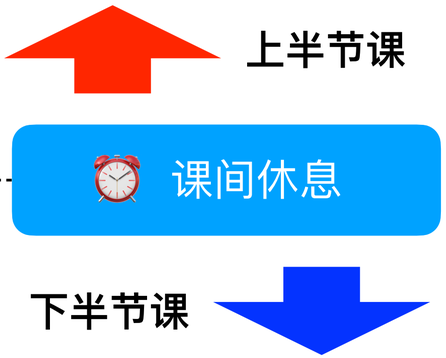
江苏·苏州

2022年8月17日

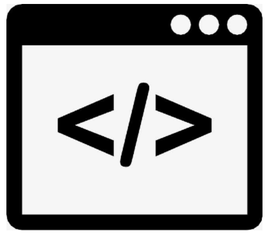
科大讯飞 iFLYTEK

报告内容

- **概述**
- **基于上下文的语言模型**
  - CoVe、ELMo
- **经典预训练语言模型**
  - 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、PERT
- **面向预训练语言模型的知识蒸馏与压缩**
  - 知识蒸馏工具TextBrewer、模型裁剪工具TextPruner
- **总结**



讲义标记



源代码

论文 / 资源

特别提示

开源项目

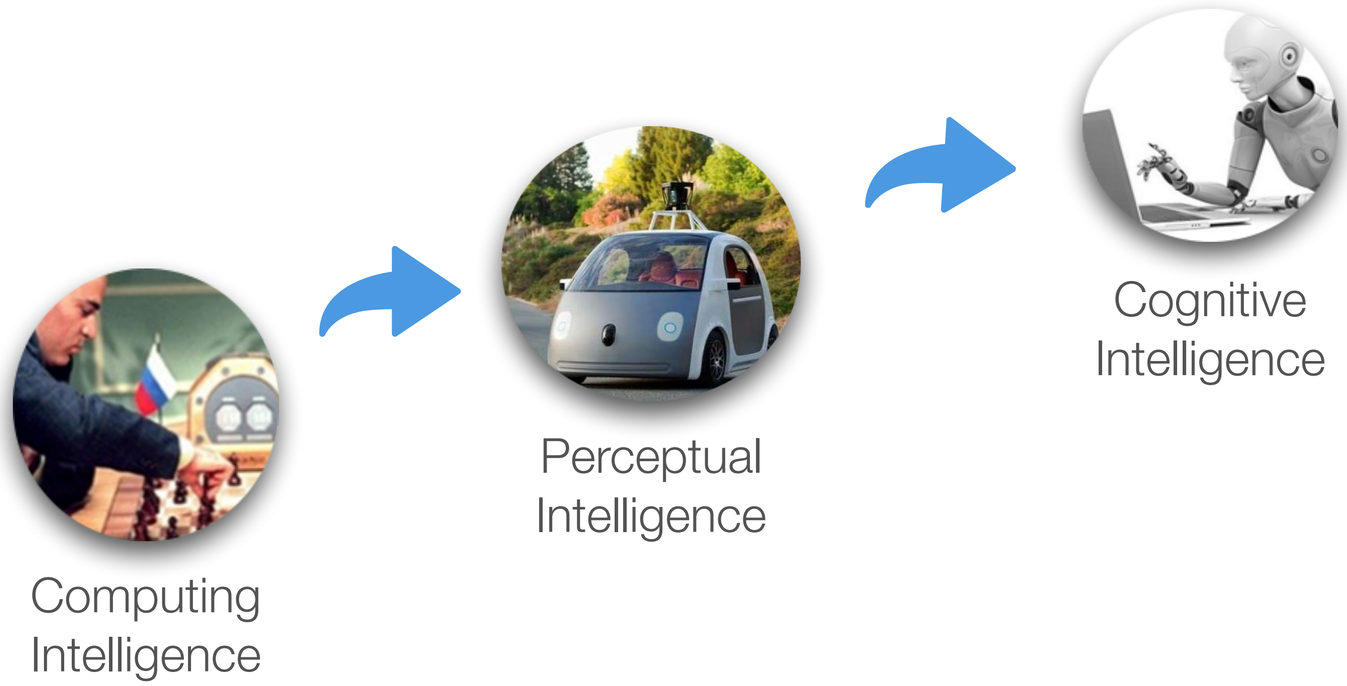
Colaboratory

概述

INTRODUCTION

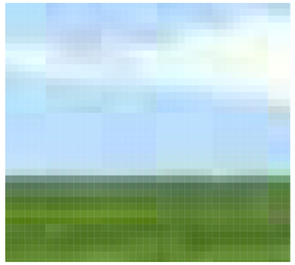
概述

- **人工智能发展的三个阶段**
  - 人工智能的一个重要目标是**让机器能听会说，能理解会思考**
  - 目前人工智能技术正处在从感知智能到认知智能跨越的时间节点
  - 自然语言理解（NLU）是认知智能中的重要内容，是通往强人工智能的必经之路



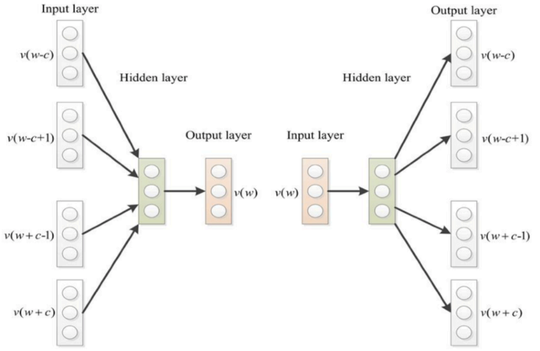
概述

- **自然语言处理为什么难?**
  - 与图像、语音不同，**语言是高度抽象的产物**，其基本组成单位并不是明确的物理量



概述

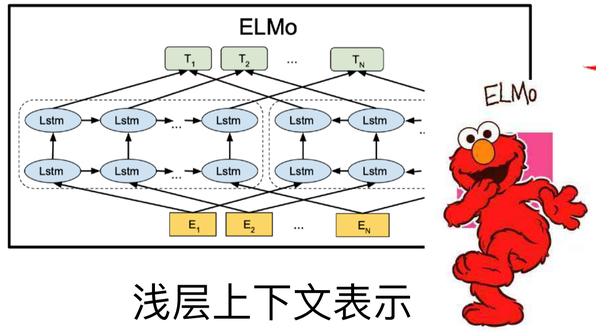
- **自然语言表示的发展一定程度上反映了自然语言处理的发展**
  - 自然语言表示的变迁很大程度影响着自然语言处理的范式
  - 从离散到连续，从上下文无关到上下文相关，从浅层到深层



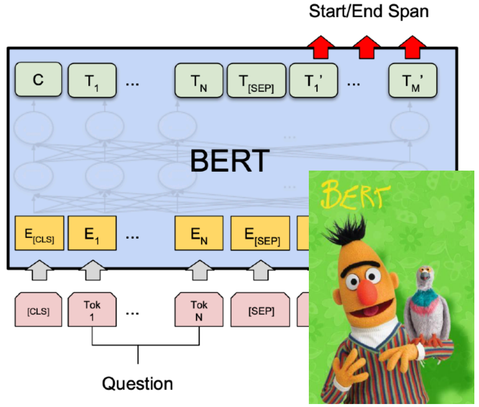
词向量表示

[0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]

One-hot表示

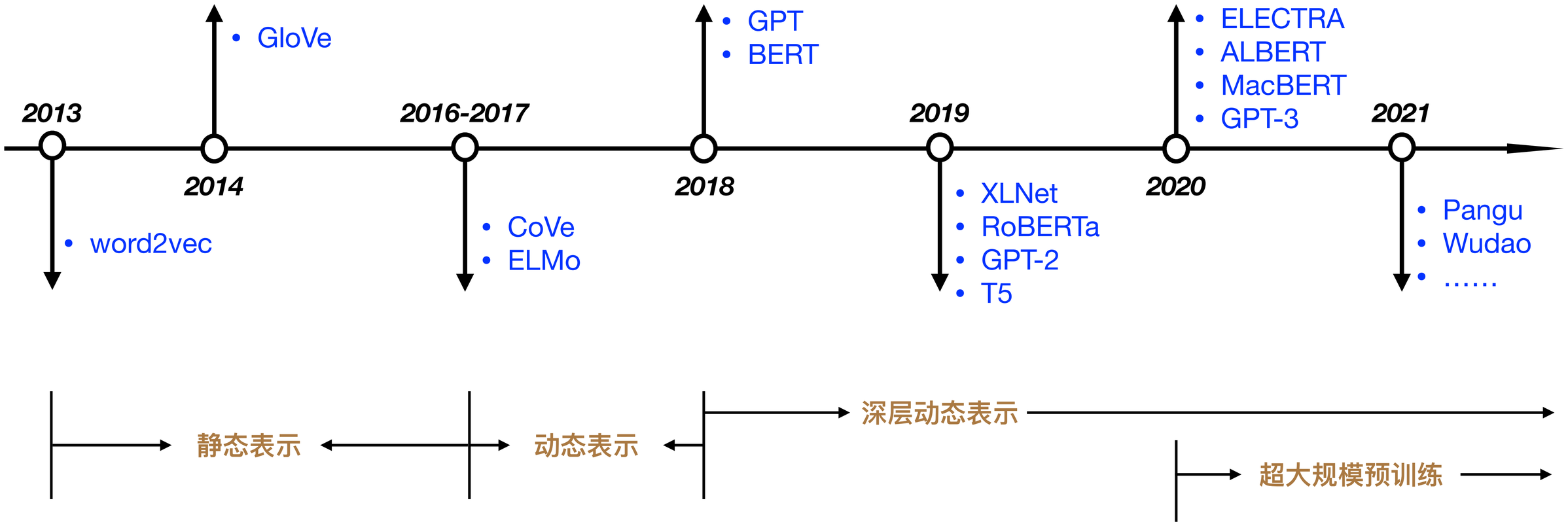


浅层上下文表示



深层上下文表示

概述

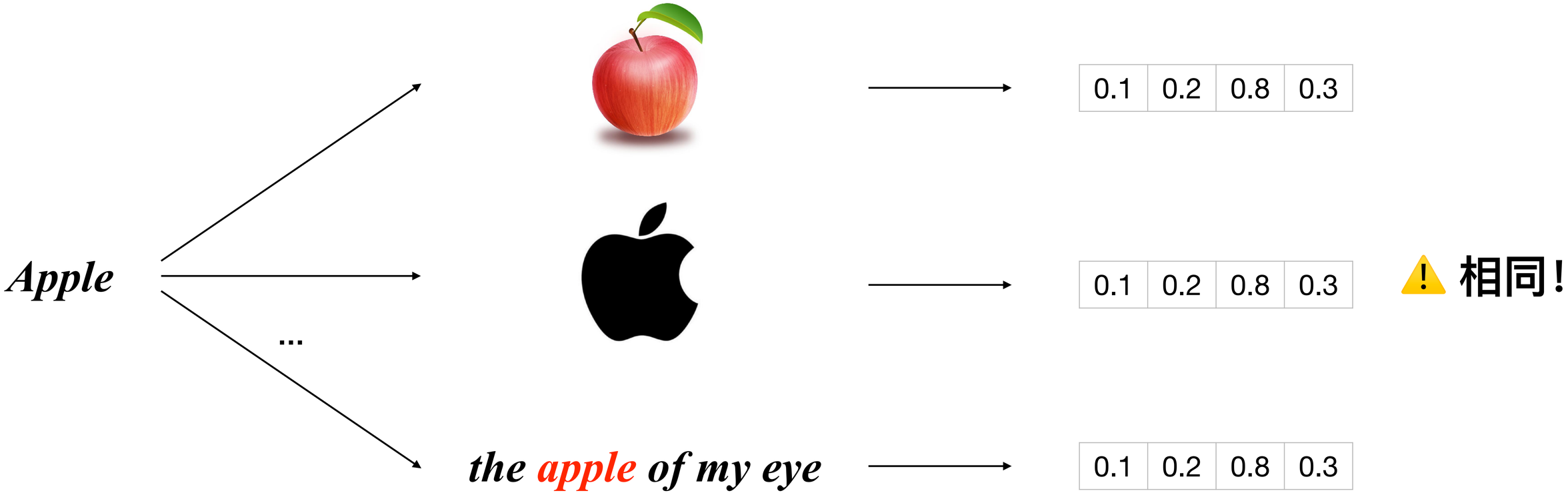

2013
2014
2016-2017
2018
2019
2020
2021
• word2vec
• GloVe
• CoVe
• ELMo
• GPT
• BERT
• XLNet
• RoBERTa
• GPT-2
• T5
• ELECTRA
• ALBERT
• MacBERT
• GPT-3
• Pangu
• Wudao
• ……
静态表示
动态表示
深层动态表示
超大规模预训练

基于上下文的语言模型

CONTEXTUALIZED LANGUAGE MODEL

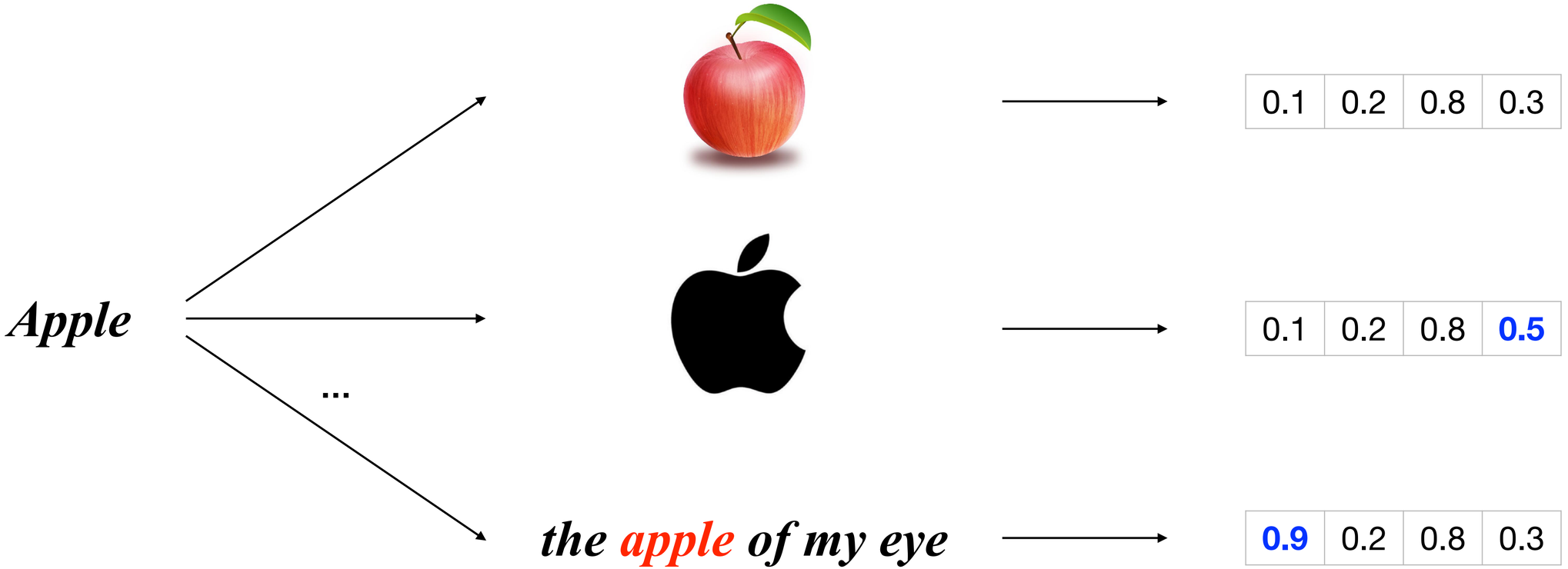
概述

- **静态词向量的问题**
  - 很多单词同时具有多个含义，静态词向量无法解决“一词多义”的问题



概述

- **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，以区分不同语义



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

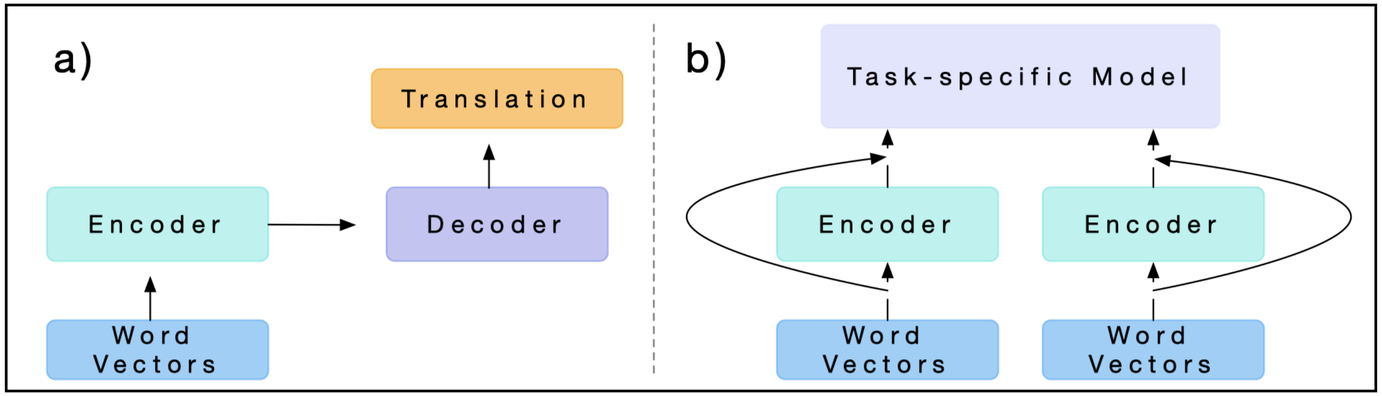
Learned in Translation: Contextualized Word Vectors

Bryan McCann
bmccann@salesforce.com

James Bradbury
james.bradbury@salesforce.com

Caiming Xiong
cxiong@salesforce.com

Richard Socher
rsocher@salesforce.com



[McCann et al., 2017] Learned in Translation: Contextualized Word Vectors. In NeurIPS 2017.

CoVe

- **训练阶段**
  - 目标：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., 2017] Learned in Translation: Contextualized Word Vectors. In NeurIPS 2017.

CoVe

- **预测阶段**
  - 对于给定的一个源语言句子 w

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

  - 如何在下游任务中使用CoVe?
    - 直接拼接GloVe和CoVe向量，送入NLP模型的其他处理模块（CNN、RNN等）

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

[McCann et al., 2017] Learned in Translation: Contextualized Word Vectors. In NeurIPS 2017.

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

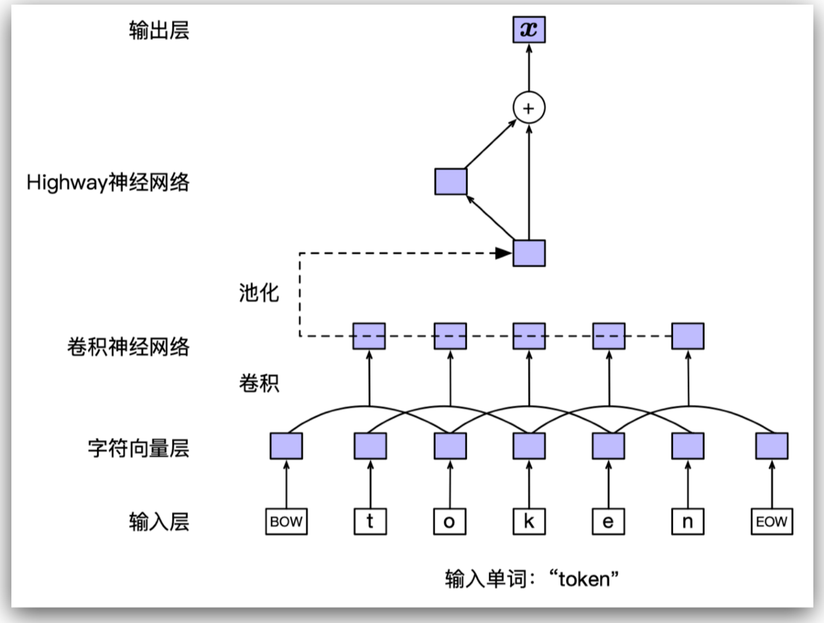
			GloVe+				
Dataset	Random	GloVe	Char	CoVe-S	CoVe-M	CoVe-L	Char+CoVe-L
SST-2	84.2	88.4	90.1	89.0	90.9	91.1	**91.2**
SST-5	48.6	53.5	52.2	54.0	54.7	54.5	**55.2**
IMDb	88.4	91.1	91.3	90.6	91.6	91.7	**92.1**
TREC-6	88.9	94.9	94.7	94.7	95.1	95.8	**95.8**
TREC-50	81.9	89.2	89.8	89.6	89.6	90.5	**91.2**
SNLI	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**

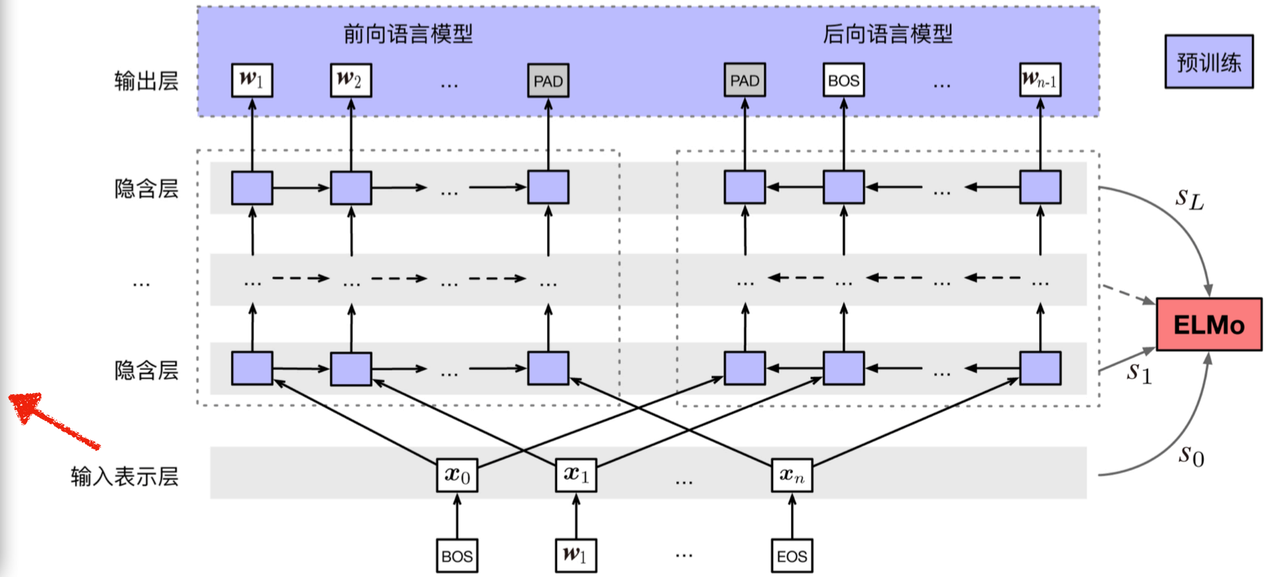
*S=Small, M=Medium, L=Large

[McCann et al., 2017] Learned in Translation: Contextualized Word Vectors. In NeurIPS 2017.

ELMo

- **ELMo: E**mbeddings from **L**anguage **Mo**dels **(NAACL 2018 Best Paper)**
  - 首次使用大规模语料训练一个**深层双向**的语言模型
  - ELMo是一种上下文相关、鲁棒性较强的模型，可以很方便地添加在现有NLP模型中





[Peters et al., 2018] Deep contextualized word representations. In NAACL 2018.

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., 2018] Deep contextualized word representations. In NAACL 2018.

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

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

  - 将 R 中的所有向量加权变换为一个向量

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

  - 下游任务中，将ELMo表示直接与其他类型的表示拼接使用

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

[Peters et al., 2018] Deep contextualized word representations. In NAACL 2018.

ELMo

- **实验结果**
  - ELMo在多个NLP任务上获得显著性能提升，全面刷新SOTA效果

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%

[Peters et al., 2018] Deep contextualized word representations. In NAACL 2018.

经典预训练语言模型

Classical Pre-trained Language Model

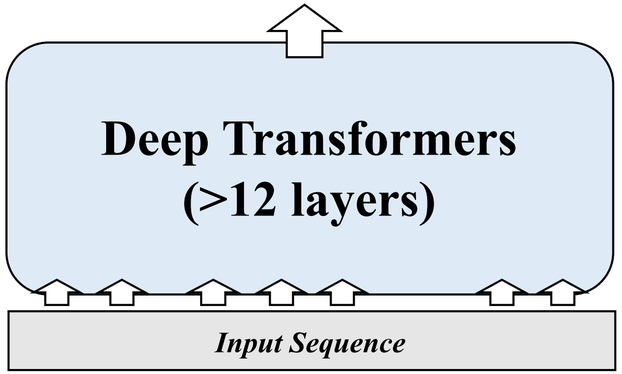
CoVe/ELMo模型的局限性

- **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），模型深度较浅
  - 使用RNN（GRU、LSTM）训练，训练难度较高，并行程度差
- **用法**
  - 通常使用这类模型时，表示模型本身是不参与训练的（权重不更新）
  - 一定程度上制约了表示模型在下游任务上的泛化能力

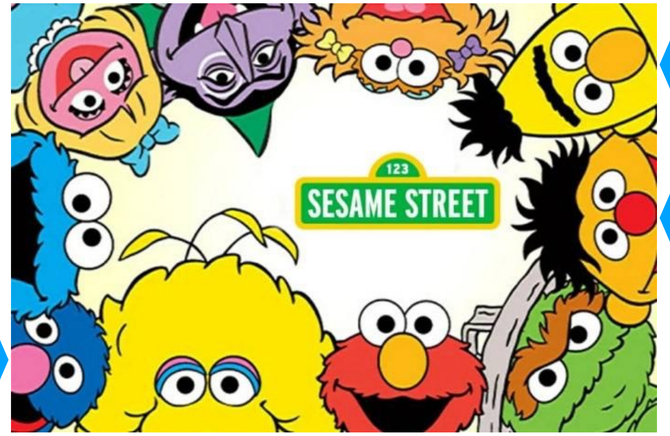
预训练模型三要素

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）



BERT

ERNIE

Grover

BigBird

ELMo

Oscar

GPT

OpenAI

GPT-2

GPT-3

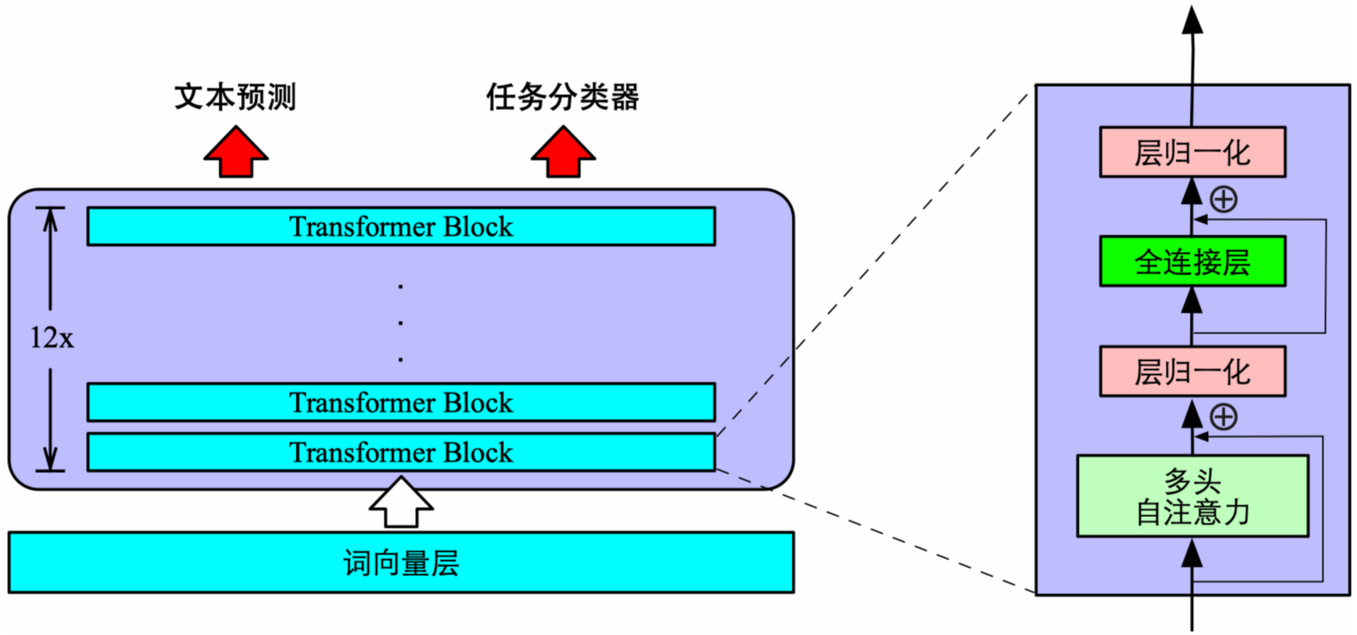
以及更多钱

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
  - GPT主要由Transformer结构构成，其核心是multi-head self-attention (Vaswani et al., 2017)



[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
  - 利用历史文本序列预测下一个单词是什么

上下文向量 词向量矩阵

$$h_0 = UW_e + W_p \quad \text{位置向量矩阵}$$

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

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

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

[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

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

  - 在某些情况下，添加额外的预训练损失可以进一步提升性能

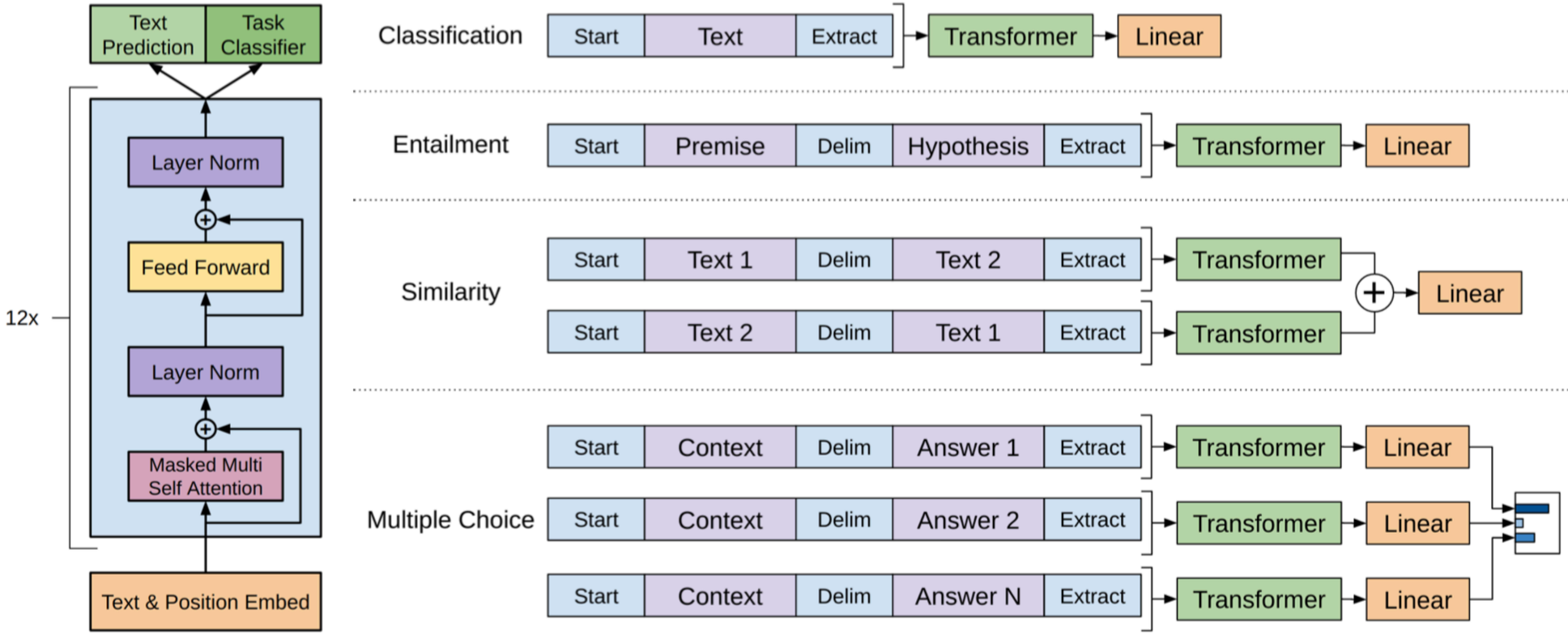
下游任务损失　　预训练损失

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

[Radford et al., 2018] Improving Language Understanding by Generative Pre-training

GPT

- 如何在下游任务使用GPT?



[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

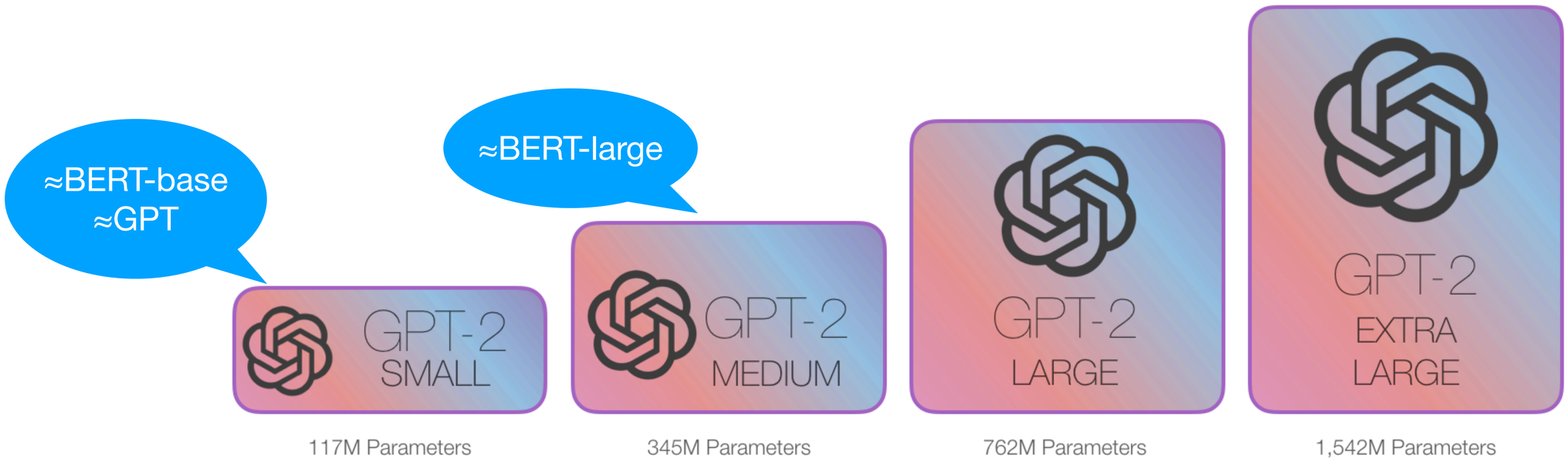
GPT-2

- **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
- **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Layer normalization
    - 词表大小：从GPT的40,000扩展至50,257
    - 上下文长度：从GPT的512扩展至1024

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

GPT-2

- 模型大小



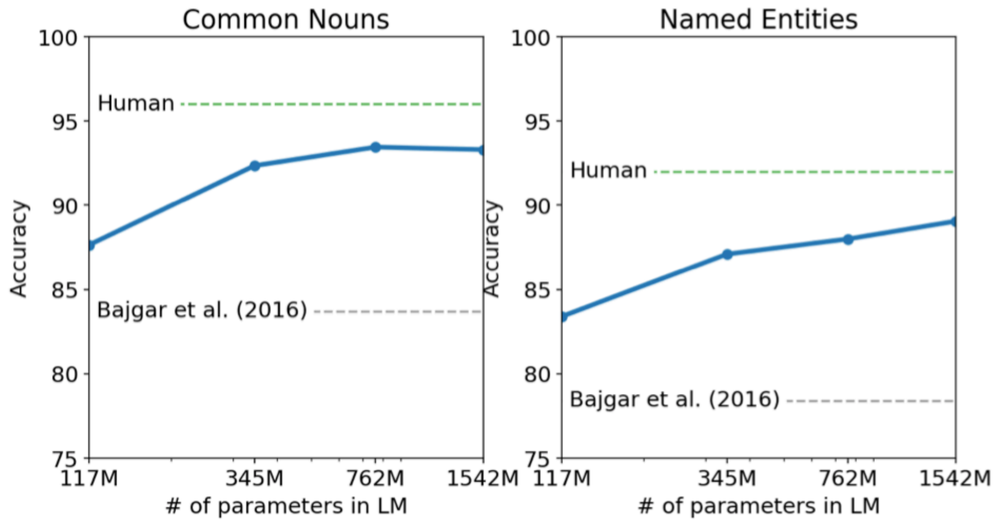
[Radford et al., 2019] Language Models are Unsupervised Multitask Learners

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，预训练模型进入到**超大规模时代**
  - 模型结构与GPT-2并无明显差别，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.

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

One-shot

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

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

Few-shot

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

```
1  Translate English to French:        ← task description
2  sea otter => loutre de mer          ← examples
3  peppermint => menthe poivrée
4  plush giraffe => girafe peluche
5  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.

[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

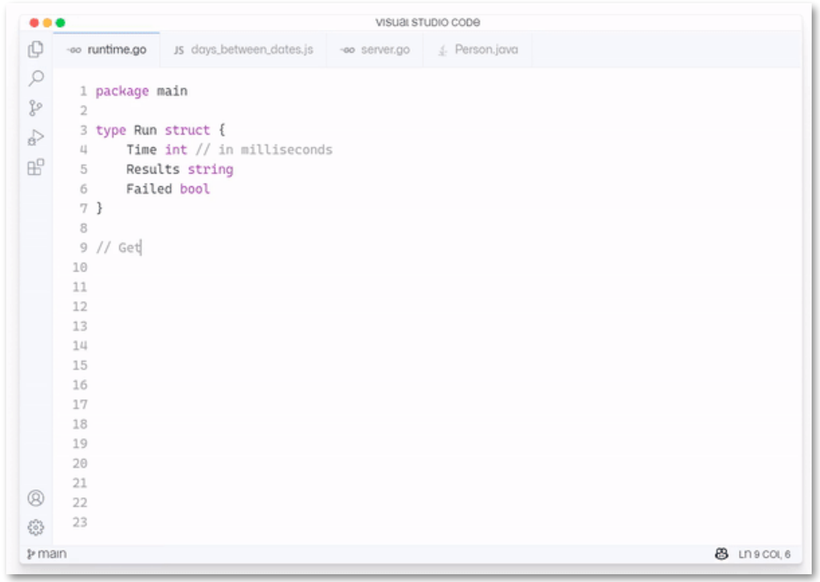
MRC

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

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

类GPT-3创新应用

- **GitHub Copilot**
  - 2021年6月，微软与OpenAI共同推出了一款AI编程工具Copilot，并于2022年6月正式上线
  - Copilot能够帮助开发者在VS、VS Code等开发环境中写代码时自动提供编码建议
  - 后台是由专为代码优化的Codex搭建（类GPT-3），拥有120亿参数量



类GPT-3创新应用

简单问答

```
Q: macOS系统是由哪个公司开发的?
A: macOS是由Apple公司开发的。
```

```
Q: What is the chemical formula of sodium chloride?
A: NaCl
```

机器翻译

```
Q: COVID-19的中文名称是什么?
A: 新型冠状病毒。
```

```
Q: NLP的中文名称是什么?
A: NLP是自然语言处理的缩写。
```

```
Q: 将“today是いいdayです、let's去쇼핑!”翻译成中文
A: 今天是一个好天气，我们去购物吧!
```

对话闲聊

```
A: Could you recommend a movie to me?
B: I would recommend the movie "The Shawshank Redemption".
```

语种支持

```
Q: 氯化钠的化学式是什么?
A: 氯化钠的化学式是C6H6N2。
```

```
Q: translate "天気がいいから、散歩しましょう。" into Korean.
A: 하늘이 구름이 없으면, 운동하세요.
```

数值计算

```
sqrt(2)=1.414
sqrt(3)=1.637
sqrt(5)=2.236
```

```
Q: 1+2+3-4=?
A: -1
```

其他问题

```
问题：TensorFlow和PyTorch哪个更好用?
答案：TensorFlow。
原因：TensorFlow是一个高性能的Python库，而PyTorch是一个高性能的Python库。
```

```
Q: 吃了唐僧肉会怎样?
A: 唐僧肉是一种美味的肉，吃了之后会变得肥胖。
```

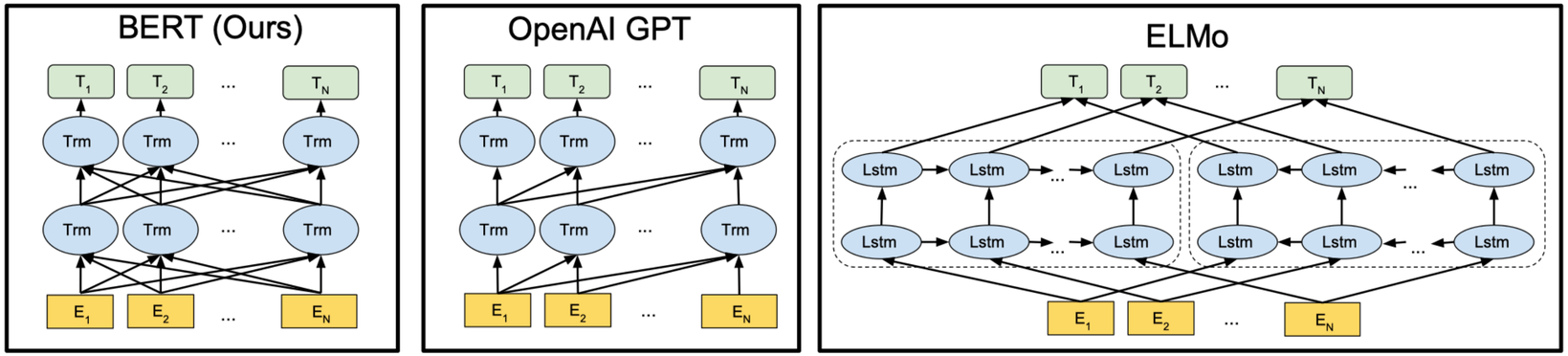

*上述案例由*VS Code* + *Github Copiot*生成，较浅色字体为自动生成，仅供参考

1 GPT 系列

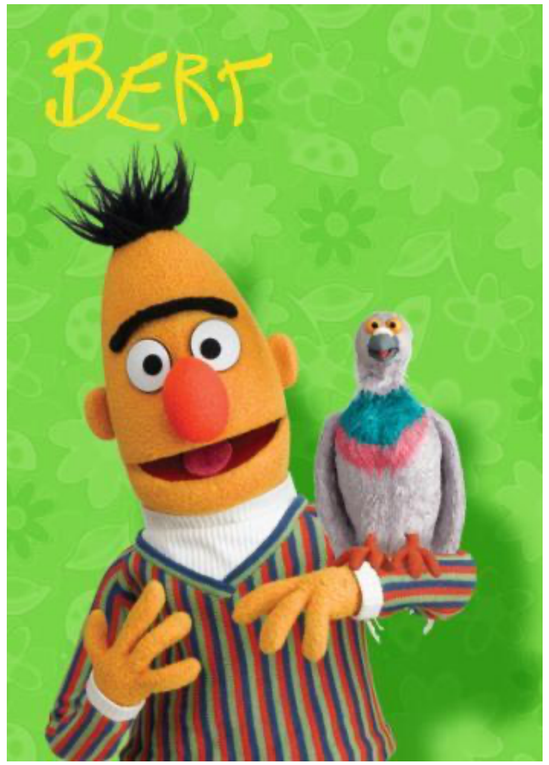
2 BERT 系列

BERT

- **BERT: B**idirectional **E**ncoder **R**epresentations from **T**ransformers
  - 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



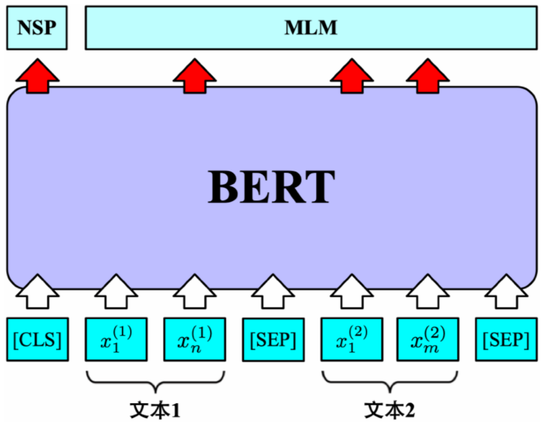
  - BERT: **双向**Transformer语言模型
  - ELMo: 将**独立的**前向和后向的LSTM语言模型拼接所得
  - GPT: **单向**从左至右的Transformer语言模型

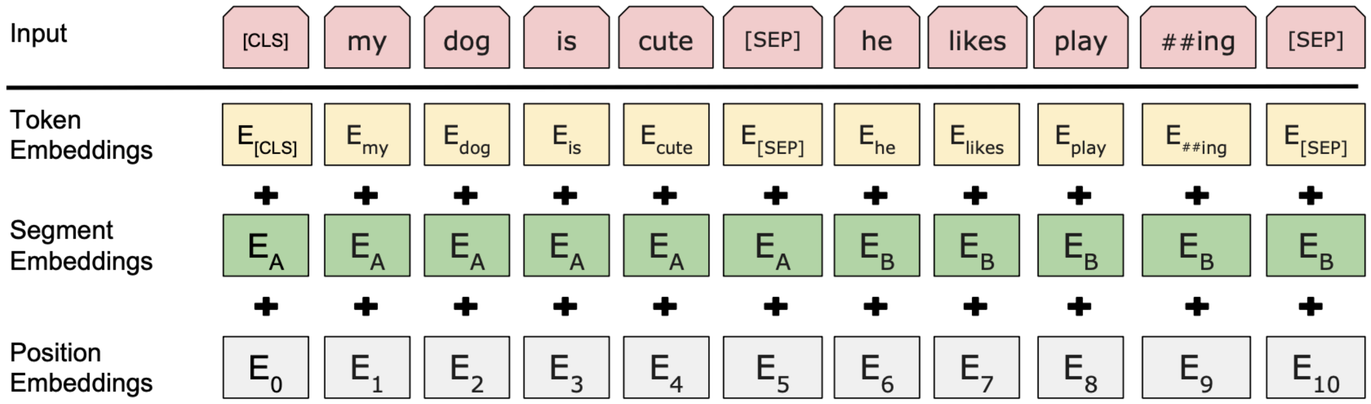


[Devlin et al., 2019] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In NAACL 2019.

BERT

- **模型结构**
  - 由深层Transformer模型构成，base版本（12层，110M）和large版本（24层，330M）
  - 使用了一个包含30,000个WordPiece的词表
  - 输入表示包含三部分：词向量、块向量、位置向量
    - 块向量（segment embedding）：表示当前token属于哪一个块（部分）
    - 位置向量（position embedding）：表示当前token的绝对位置





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

BERT

- **模型结构：Transformers编码器**

⑤ Residual Output Layer

$$O = \textbf{LayerNorm}(\textbf{Dropout}(\textbf{FFN}(I)) + H)$$

④ Intermediate Layer

$$I = \textbf{GeLU}(\textbf{FFN}(H))$$

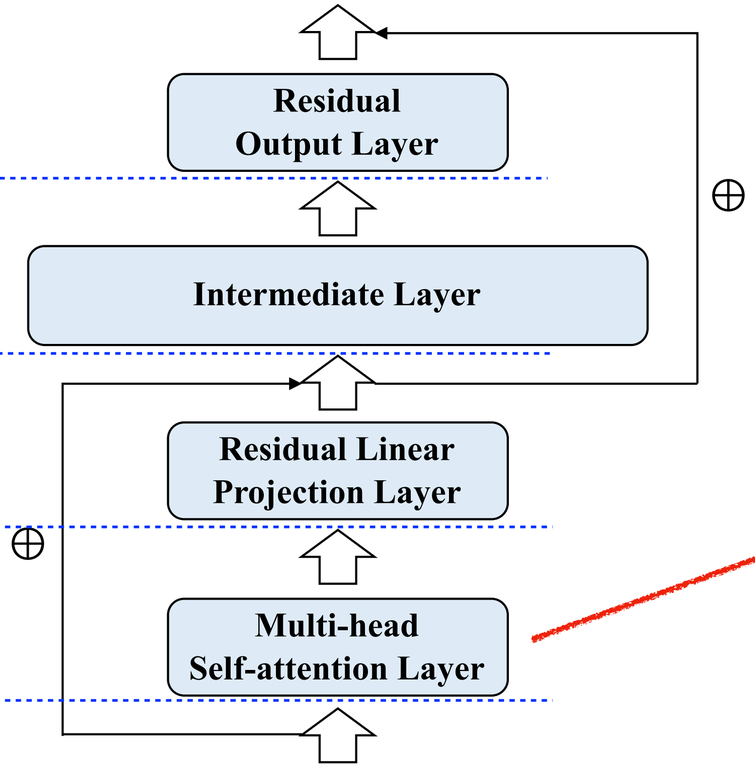
③ Residual Linear Projection Layer

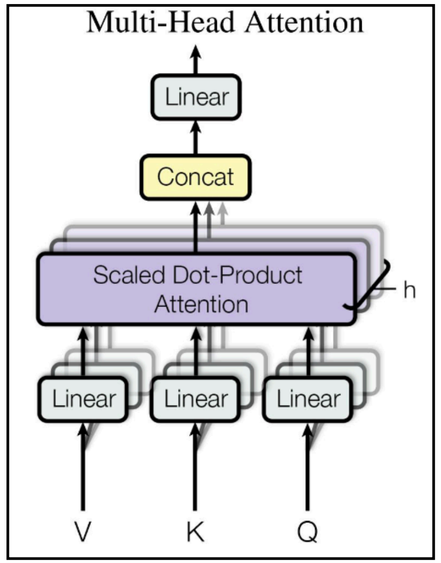
$$H = \textbf{LayerNorm}(\textbf{Dropout}(\textbf{FFN}(A)) + X)$$

② Multi-head Self-attention Layer

$$A = \textbf{MHA}(X)$$

① X = Input

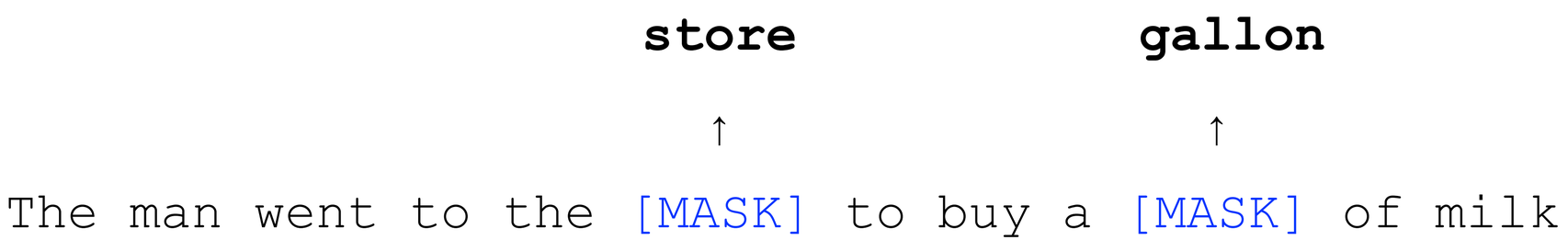




[Devlin et al., 2019] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In NAACL 2019.

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操作
    - 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



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

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%
replace w/ [MASK]

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

10%
replace w/ random word

went to the **store** → went to the **apple**

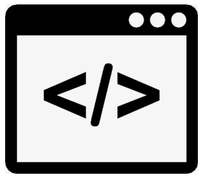
10%
keep original word

went to the **store** → went to the **store**

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

BERT

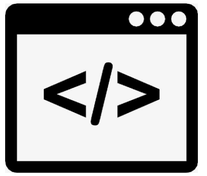
- **MLM的原版代码实现**
  - 文件: `create_pretraining_data.py`
  - 函数: `create_masked_lm_predictions()`
  - **参数列表**
    - **tokens** (*list*): tokenized sequence tokens
    - **masked_lm_prob** (*float*): how many words (proportion) should be masked
    - **max_predictions_per_seq** (*int*): maximum predictions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



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

BERT

- **第一步：生成候选下标集合**



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

① 跳过 [CLS] 和 [SEP]

```
rng.shuffle(cand_indexes)
```

② 将候选下标进行打乱

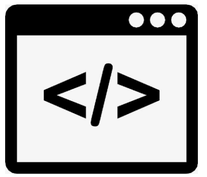
```
output_tokens = list(tokens)

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

③ 确定要预测的token数量

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

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



```
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]))
```


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

BERT

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

IsNextSentence

Sentence A: The man went to the store.

Sentence B: He bought a gallon of milk.

NotNextSentence

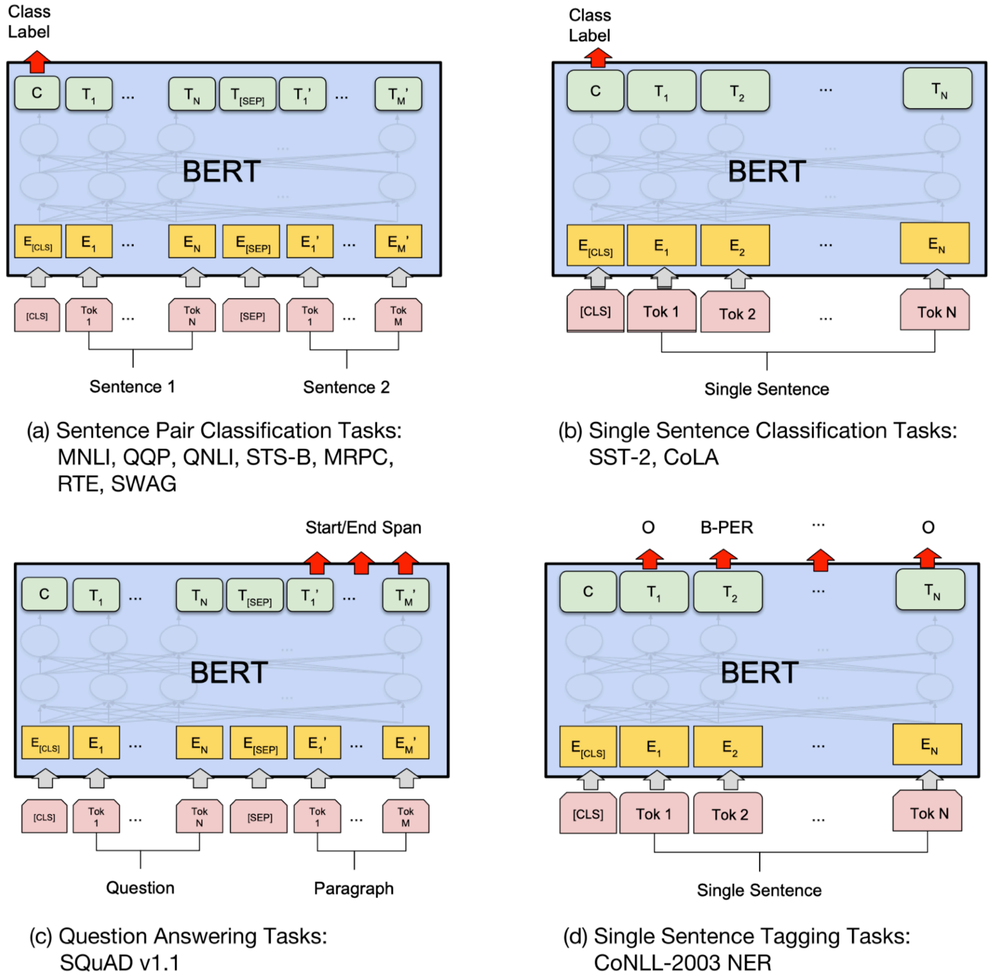
Sentence A: The man went to the store.

Sentence B: Penguins are flightless.

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

BERT

- **结果在下游任务中精调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
    - 序列标注：每个token预测相应标签



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

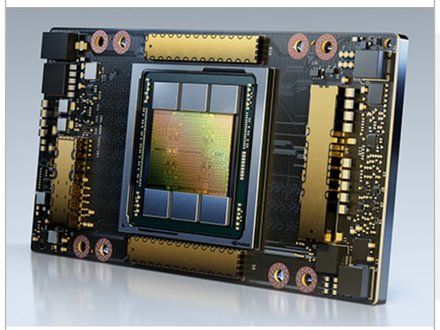
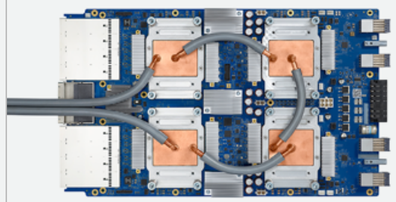
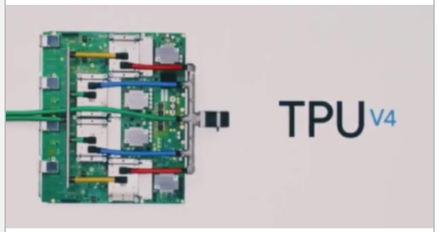
BERT

- **预训练实验设置**
  - 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 (16 chips)
    - BERT-large: 16 Cloud TPUs (64 chips)

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

BERT

- **张量运算单元（Tensor Processing Units, TPU）**
  - 以TPU v3为例，1 hardware = 4 chips = 8 cores = 128GB HBM

	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	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

- **训练这样一个模型需要多少钱？（以BERT-large为例）**
  - TPU v2每小时使用价格为4.5美元（USD）

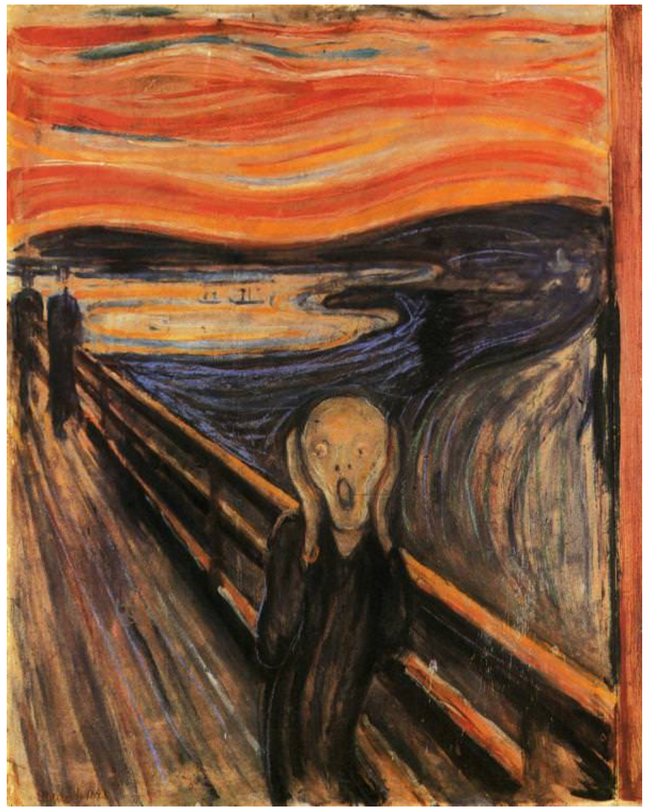
16 Cloud TPUs = 16 × 4.5 = 72 USD / hour

一天消耗 = 72 × 24 = 1,728 USD

四天消耗 = 1,728 USD × 4 = 6,912 USD

6,912 USD ≈ 46,765 CNY

早期训练预训练模型的成本较高
采用“二次预训练”的方法可以大幅降低训练成本



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

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

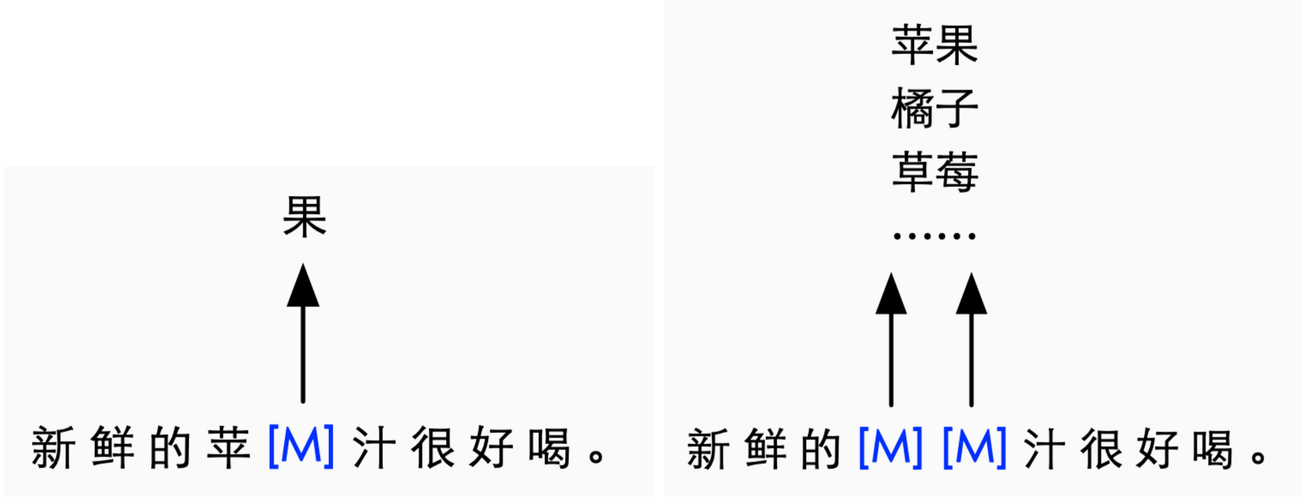
System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**91.1**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**81.9**

System	Dev EM	Dev F1	Test EM	Test 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				
$\text{BERT}_{\text{BASE}}$ (Single)	80.8	88.5	-	-
$\text{BERT}_{\text{LARGE}}$ (Single)	84.1	90.9	-	-
$\text{BERT}_{\text{LARGE}}$ (Ensemble)	85.8	91.8	-	-
$\text{BERT}_{\text{LARGE}}$ (Sgl.+TriviaQA)	**84.2**	**91.1**	**85.1**	**91.8**
$\text{BERT}_{\text{LARGE}}$ (Ens.+TriviaQA)	**86.2**	**92.2**	**87.4**	**93.2**

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

中文BERT-wwm

- **基于全词掩码的中文BERT系列模型**
  - 将全词掩码技术（Whole Word Masking，wwm）应用在中文BERT的建模中
  - 不同掩码策略
    - 标准掩码语言模型（MLM）：将输入文本中的字看作独立分布，随机对字进行掩码
    - 全词掩码语言模型（WWM）：考虑了中文分词，将属于同一个词中的所有字进行掩码



(a) 以字为掩码单位　　(b) 以词为掩码单位

[Cui et al., 2020] Revisiting Pre-trained Models for Chinese Natural Language Processing. In Findings of EMNLP 2020.
[Cui et al., 2021] Pre-training with Whole Word Masking for Chinese BERT. In IEEE/ACM TASLP Vol.28.

中文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	BERT			model	uses	wordpiece		tokenization		.
分词	B	##ER	##T	model	uses	word	##piece	token	##ization	.
MLM	B	**[M]**	**[M]**	model	**uses!**	word	**[M]**	token	##ization	**[M]**
	B	##ER	**[M]**	**model!**	**good**	word	##piece	**[M]**	**[M]**	.
	[M]	##ER	##T	**[M]**	**[M]**	word	**hello**	**token!**	##ization	.
WWM	**[M]**	**[M]**	**[M]**	model	uses	word	##piece	**token!**	**apple**	.
	B!	hello	**[M]**	model	**[M]**	**[M]**	##piece	token	##ization	.
	B	##ER	##T	fruit	uses	word!	**[M]**	**[M]**	**[M]**	.

[M]表示掩码符号，!表示保留原词操作，假设掩码概率为50%

[Cui et al., 2020] Revisiting Pre-trained Models for Chinese Natural Language Processing. In Findings of EMNLP 2020.
[Cui et al., 2021] Pre-training with Whole Word Masking for Chinese BERT. In IEEE/ACM TASLP Vol.28.

中文BERT-wwm

- **实验结果**
  - 基于全词掩码的方法能够显著提升预训练模型性能

Model	SQuAD 1.1	MNLI
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

▲ 英文任务效果（from Google）

Model	CMRC 2018	XNLI
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

▲ 中文任务效果

[Cui et al., 2020] Revisiting Pre-trained Models for Chinese Natural Language Processing. In Findings of EMNLP 2020.
[Cui et al., 2021] Pre-training with Whole Word Masking for Chinese BERT. In IEEE/ACM TASLP Vol.28.

N-gram Masking

- **N元文法掩码（N-gram masking）**
  - 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
  - 预测难度：N-gram Masking > Whole Word Masking > vanilla MLM

Original
原句

We went to the store to buy some fruits.

WWM
整词掩码

We went to the [M] to [M] some [M].

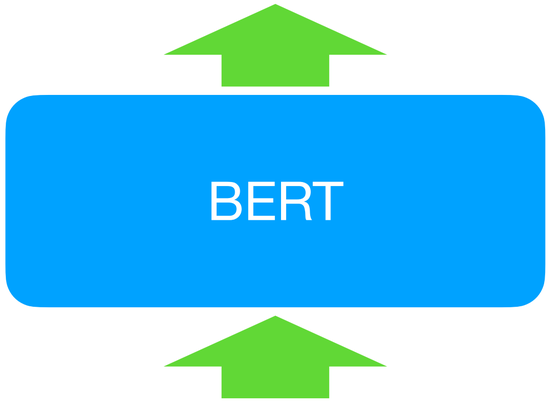
NM
N-gram掩码

We went to the store to [M] [M] [M].

[Cui et al., 2020] Revisiting Pre-trained Models for Chinese Natural Language Processing. In Findings of EMNLP 2020.
[Cui et al., 2021] Pre-training with Whole Word Masking for Chinese BERT. In IEEE/ACM TASLP Vol.28.

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**

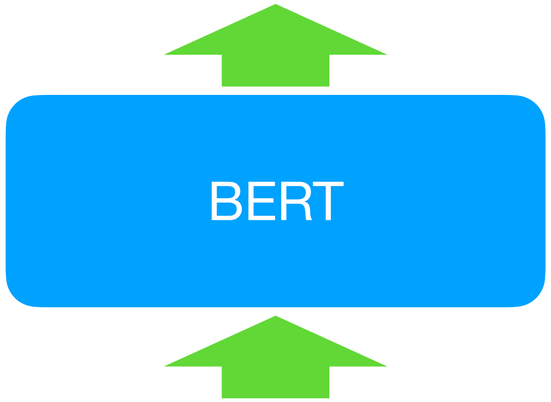


NM: 我 [M] [M] [M] 天 安 门

WWM: [M] 爱 [M] [M] 天 安 门

MLM: 我 [M] [M] 京 天 [M] 门

预训练阶段



✓ MLM/WWM/NM: 我 爱 北 京 天 安 门

✗ MLM/WWM/NM: 我 爱 北京 天安门

下游任务精调阶段

[Cui et al., 2020] Revisiting Pre-trained Models for Chinese Natural Language Processing. In Findings of EMNLP 2020.
[Cui et al., 2021] Pre-training with Whole Word Masking for Chinese BERT. In IEEE/ACM TASLP Vol.28.

预训练语言模型进阶

Advanced Pre-trained Language Model

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

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

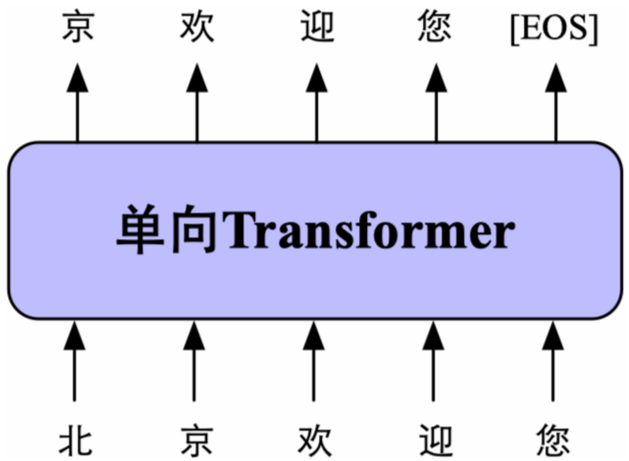
XLNet: Generalized Autoregressive Pretraining for Language Understanding

Zhilin Yang[*1], **Zihang Dai**[*12], **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

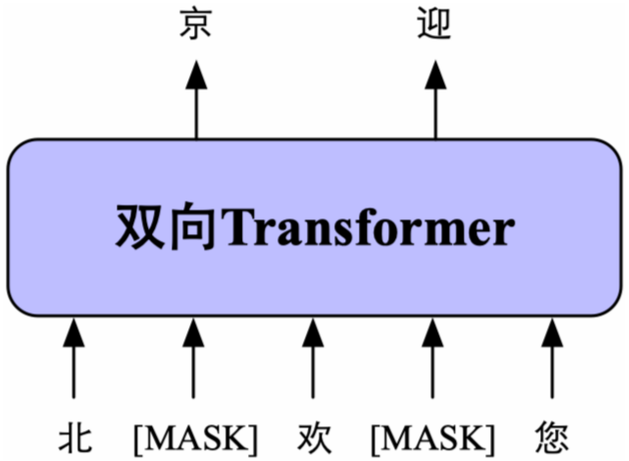
[Yang et al., 2020] XLNet: Generalized Autoregressive Pretraining for Language Understanding. In NeurIPS 2020.

XLNet

- **两种经典的语言建模方法**
  - 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

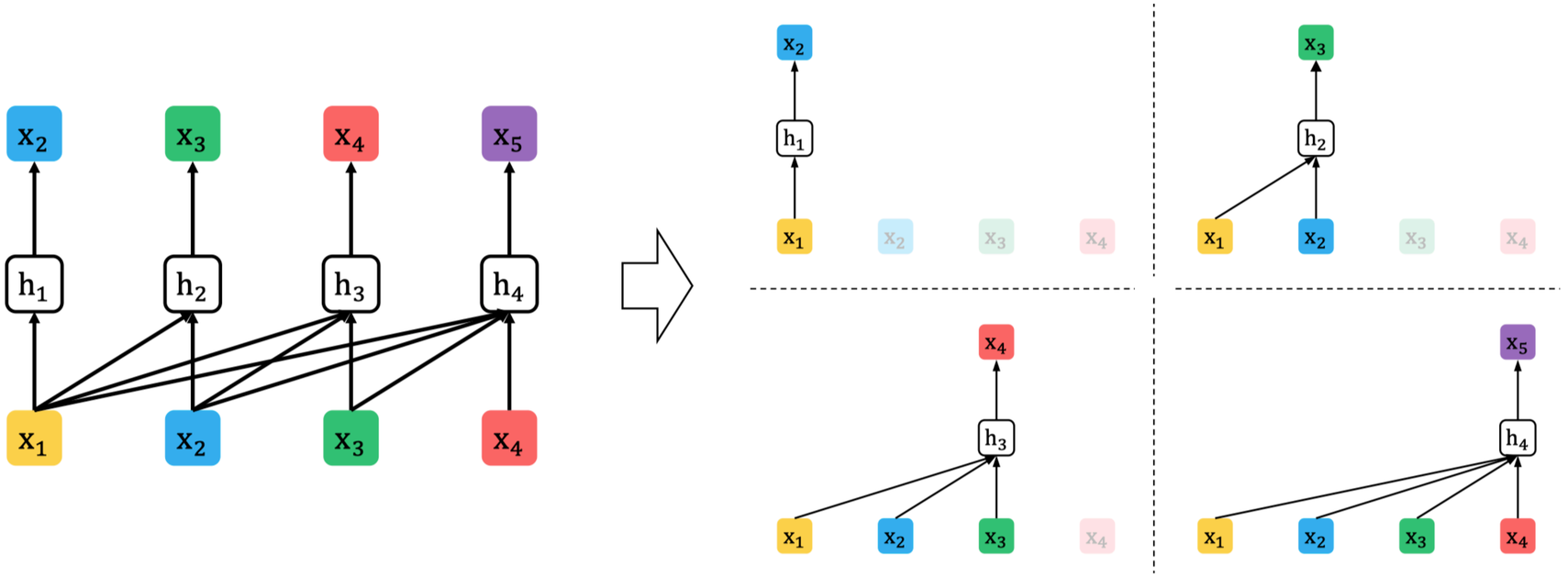


(b) 自编码语言模型

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

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., 2020] XLNet: Generalized Autoregressive Pretraining for Language Understanding. In NeurIPS 2020.

XLNet

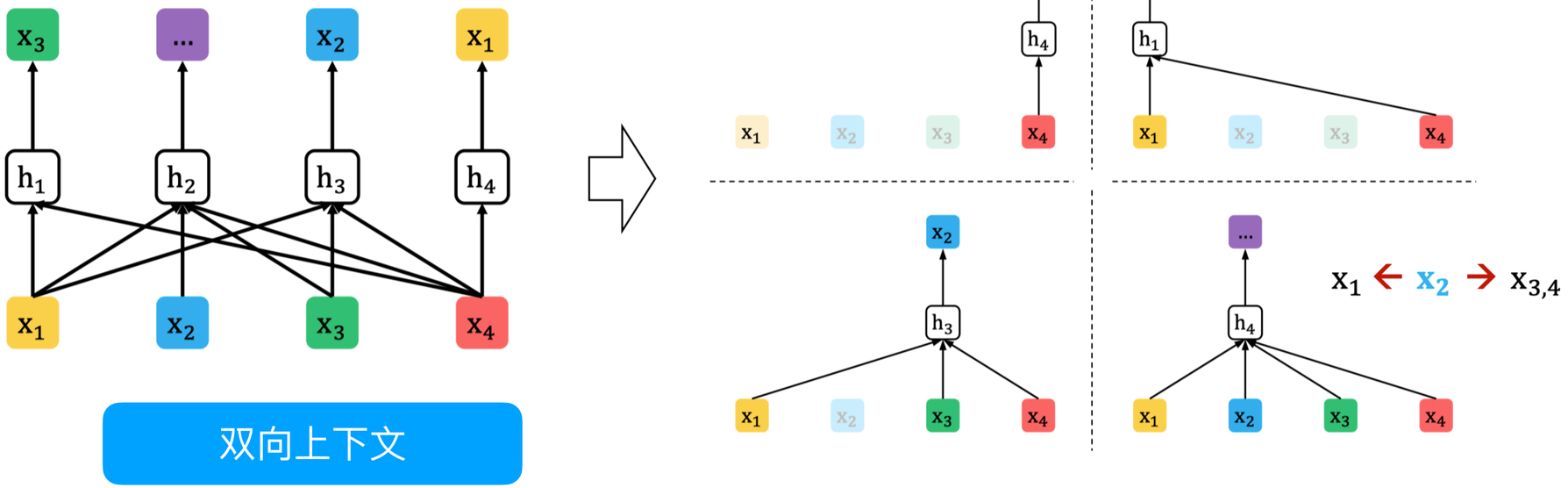
- **排列语言模型（Permutation Language Model）**
  - 给定一个长度为T 的序列 $\mathbf{x}$
  - 从所有可能的排列当中均匀采样一种排列顺序（factorization order）记为 $\mathbf{z}$
  - 最大化对数似然函数

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

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

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., 2020] XLNet: Generalized Autoregressive Pretraining for Language Understanding. In NeurIPS 2020.

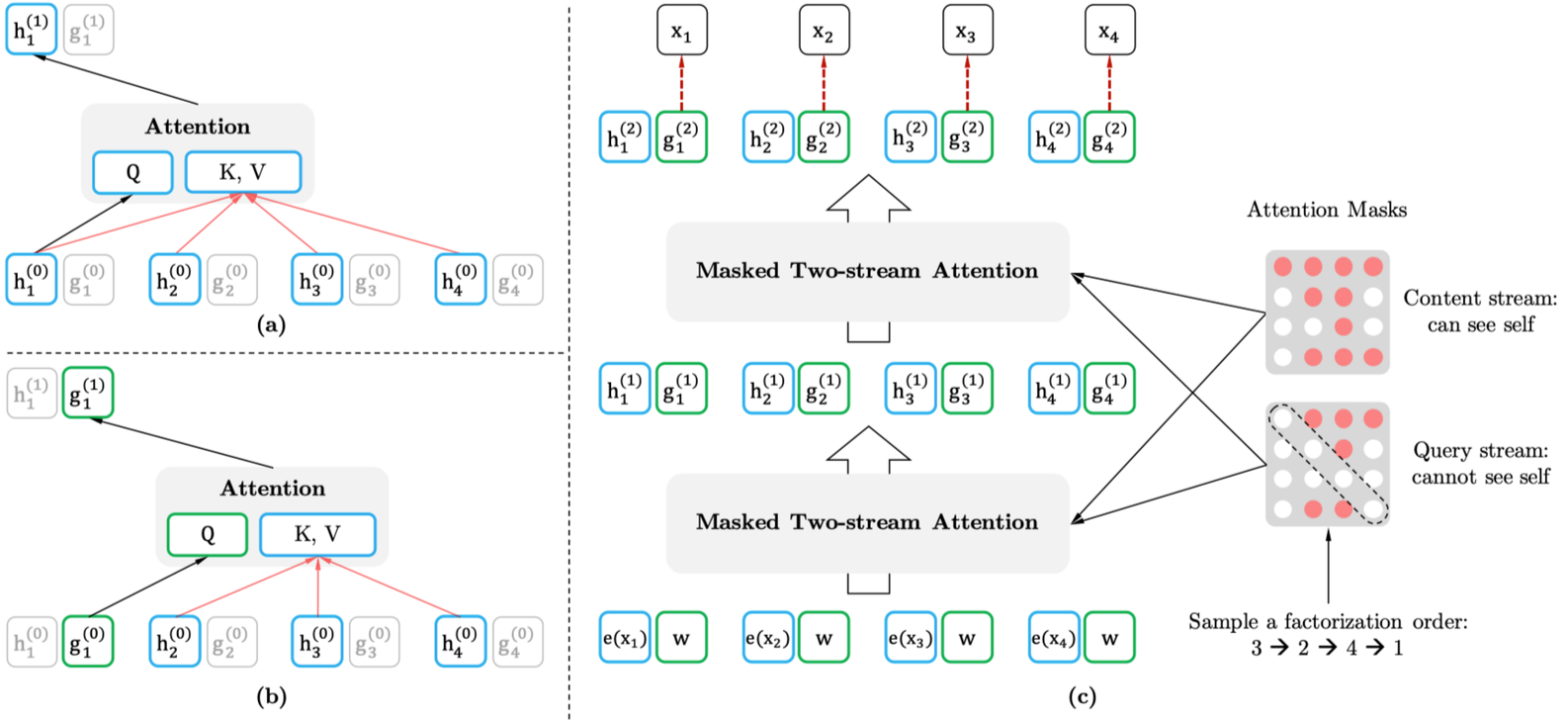
XLNet

- 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
$3 \to 2 \to 4 \to 1$	$\varnothing$	$\boldsymbol{M}^{\mathrm{h}}_{3,3}$	$\varnothing$
$3 \to 2 \to 4 \to 1$	x_3	$\boldsymbol{M}^{\mathrm{h}}_{2,3}, \boldsymbol{M}^{\mathrm{h}}_{2,2}$	$\boldsymbol{M}^{\mathrm{g}}_{2,3}$
$3 \to 2 \to 4 \to 1$	x_3, x_2	$\boldsymbol{M}^{\mathrm{h}}_{4,3}, \boldsymbol{M}^{\mathrm{h}}_{4,2}, \boldsymbol{M}^{\mathrm{h}}_{4,4}$	$\boldsymbol{M}^{\mathrm{g}}_{4,3}, \boldsymbol{M}^{\mathrm{g}}_{4,2}$
$3 \to 2 \to 4 \to 1$	x_3, x_2, x_4	$\boldsymbol{M}^{\mathrm{h}}_{1,3}, \boldsymbol{M}^{\mathrm{h}}_{1,2}, \boldsymbol{M}^{\mathrm{h}}_{1,4}, \boldsymbol{M}^{\mathrm{h}}_{1,1}$	$\boldsymbol{M}^{\mathrm{g}}_{1,3}, \boldsymbol{M}^{\mathrm{g}}_{1,2}, \boldsymbol{M}^{\mathrm{g}}_{1,4}$

内容流注意力
（Content Stream Attention）

查询流注意力
（Query Stream Attention）

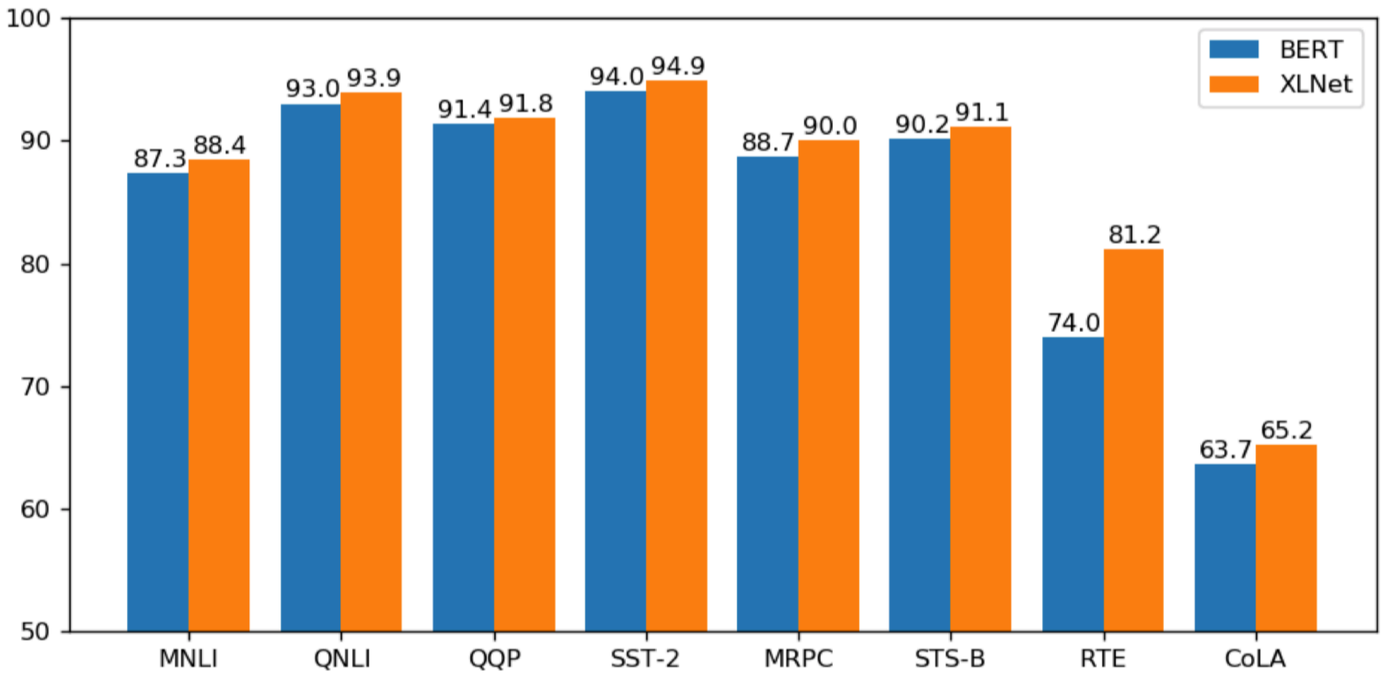


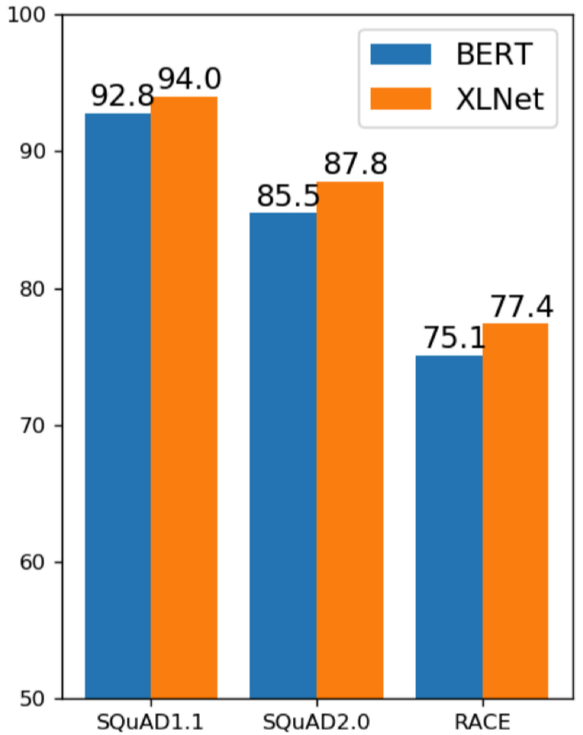
[Yang et al., 2020] XLNet: Generalized Autoregressive Pretraining for Language Understanding. In NeurIPS 2020.

XLNet

扩展阅读

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





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

RoBERTa

- **RoBERTa: Ro**bustly optimized **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
  - 构建了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**[§]

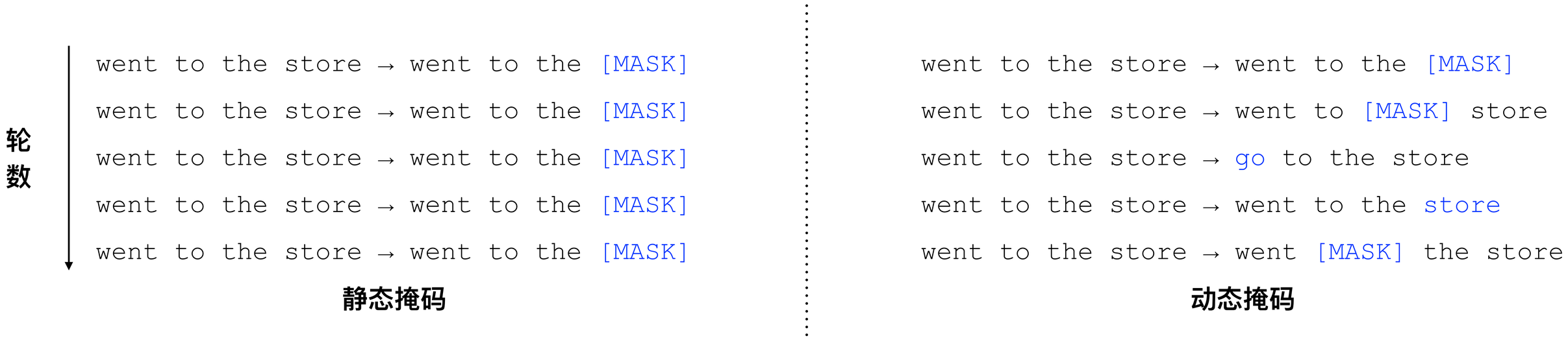
[†] Paul G. Allen School of Computer Science & Engineering,
University of Washington, Seattle, WA
{mandar90,lsz}@cs.washington.edu

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

[Liu et al., 2019] RoBERTa: A Robustly Optimized BERT Pretraining Approach. In arXiv pre-print.

RoBERTa

- **静态掩码 v.s. 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



[Liu et al., 2019] RoBERTa: A Robustly Optimized BERT Pretraining Approach. In arXiv pre-print.

RoBERTa

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

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

Original BERT implementation →
Natural sentences →

Could cross the document boundary →
Could NOT cross the document boundary →

[Liu et al., 2019] RoBERTa: A Robustly Optimized BERT Pretraining Approach. In arXiv pre-print.

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：成功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., 2019] RoBERTa: A Robustly Optimized BERT Pretraining Approach. In arXiv pre-print.

RoBERTa

- **RoBERTa最终形态：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

[Liu et al., 2019] RoBERTa: A Robustly Optimized BERT Pretraining Approach. In arXiv pre-print.

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预训练任务

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**
$\text{BERT}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7
$\text{XLNet}_{\text{LARGE}}$						
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

[Liu et al., 2019] RoBERTa: A Robustly Optimized BERT Pretraining Approach. In arXiv pre-print.

ALBERT

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

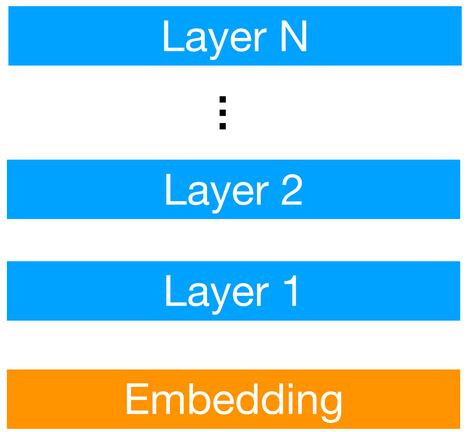
ALBERT: A LITE BERT FOR SELF-SUPERVISED LEARNING OF LANGUAGE REPRESENTATIONS

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., 2020] ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In ICLR 2020.

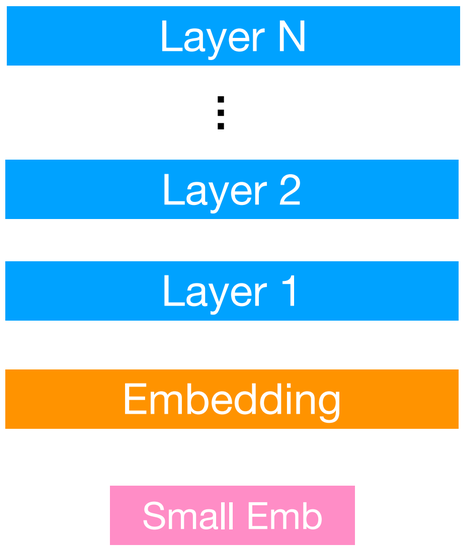
ALBERT

- 词向量因式分解（Factorized Embedding Parameterization）



`embedding_size == hidden_size`

BERT



`embedding_size < hidden_size`

ALBERT

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

ALBERT

- **词向量因式分解**
  - 利用词向量因式分解后

$$O(V \times H) \quad \Rightarrow \quad O(V \times E + E \times H)$$

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

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

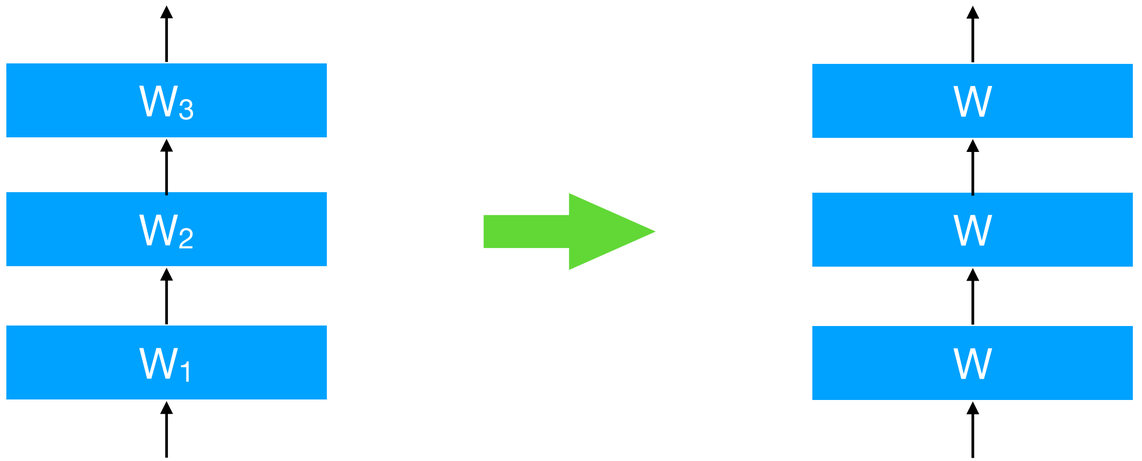
$>$

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

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

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** 💡



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

ALBERT

- **句子顺序预测（Sentence Order Prediction, SOP）**
  - NSP任务同时隐含了“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

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

ALBERT

- **有效性分析**
  - 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.

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

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

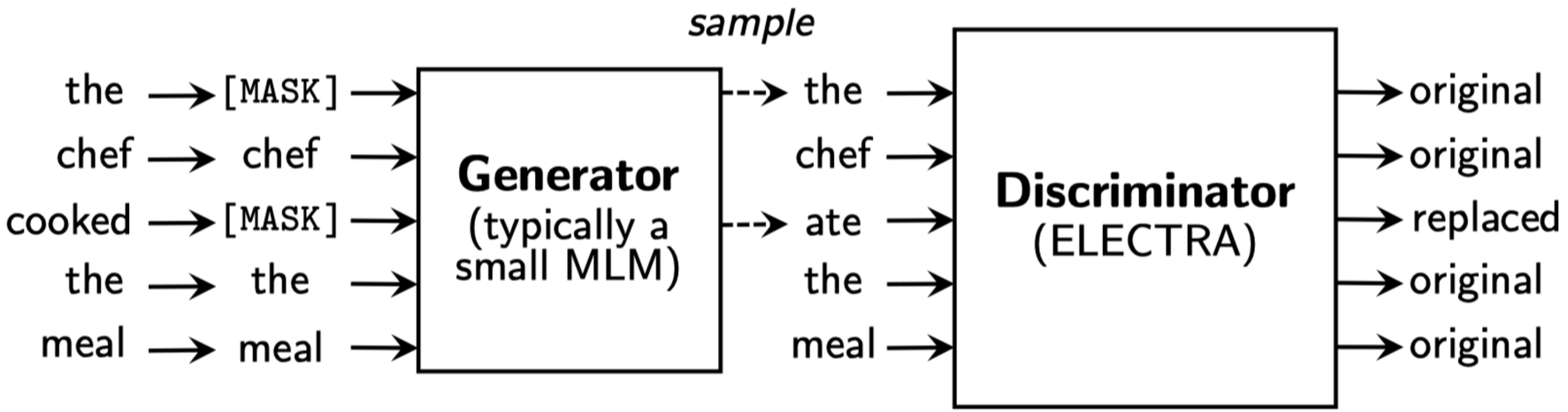
Quoc V. Le
Google Brain
qvl@google.com

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

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

ELECTRA

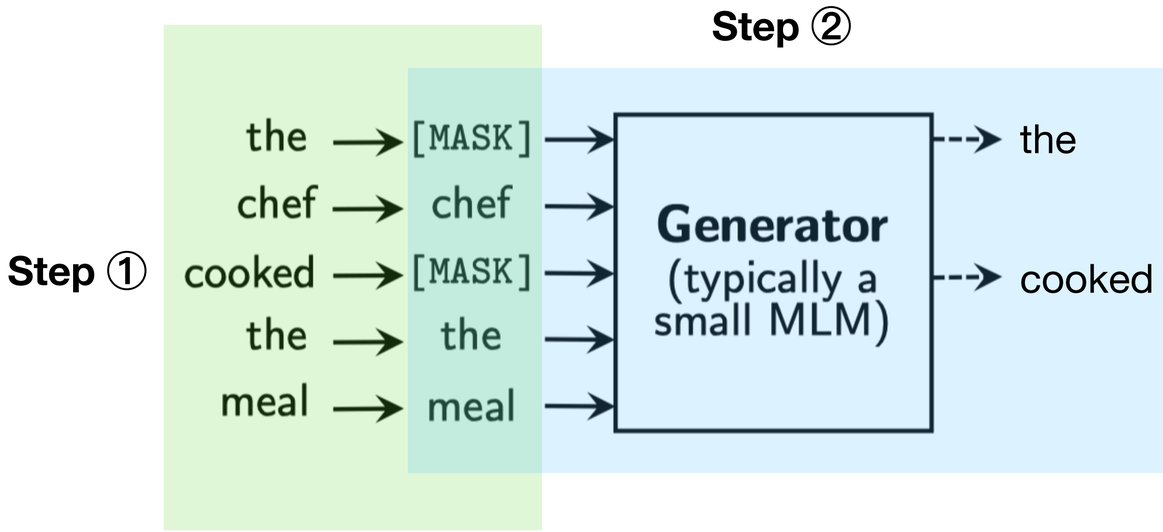
- **模型结构**
  - 生成器-判别器框架，与GAN (Goodfellow et al., 2014) 类似
  - **Generator**: 将输入中的缺失信息还原为原单词
  - **Discriminator**: 判断输入的单词是否被替换过



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

ELECTRA

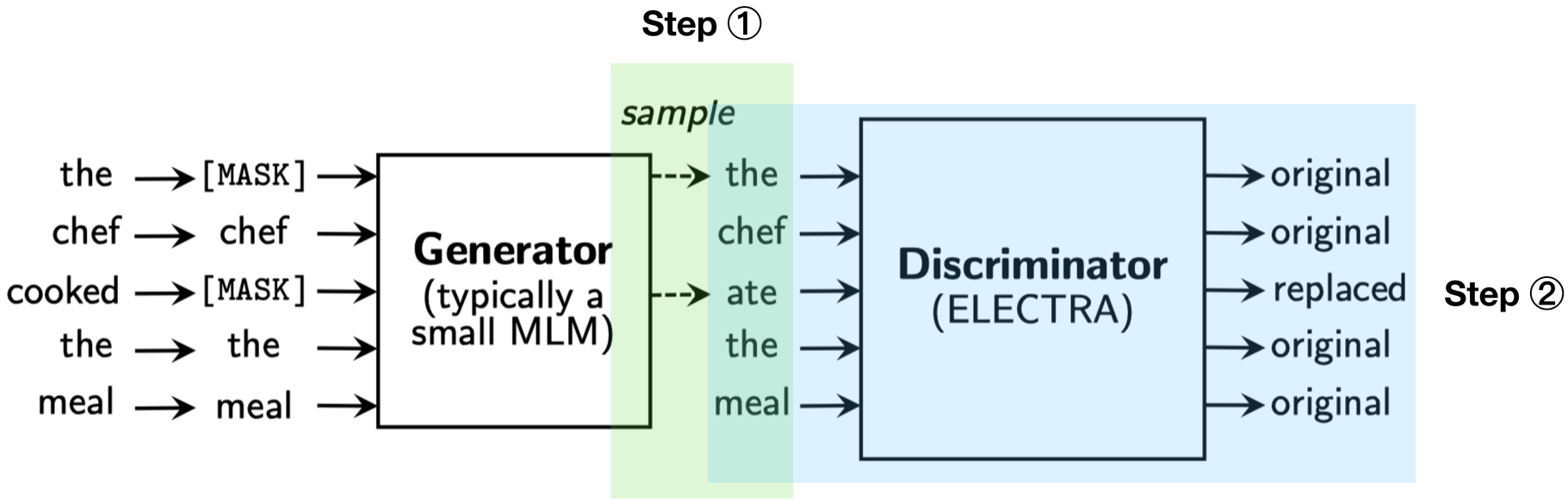
- **生成器：一个小的MLM**
  - 第一步：随机选取输入序列中的一部分进行遮蔽，通常比例选取15%
  - 第二步：利用MLM任务，将缺失信息还原为原单词



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

ELECTRA

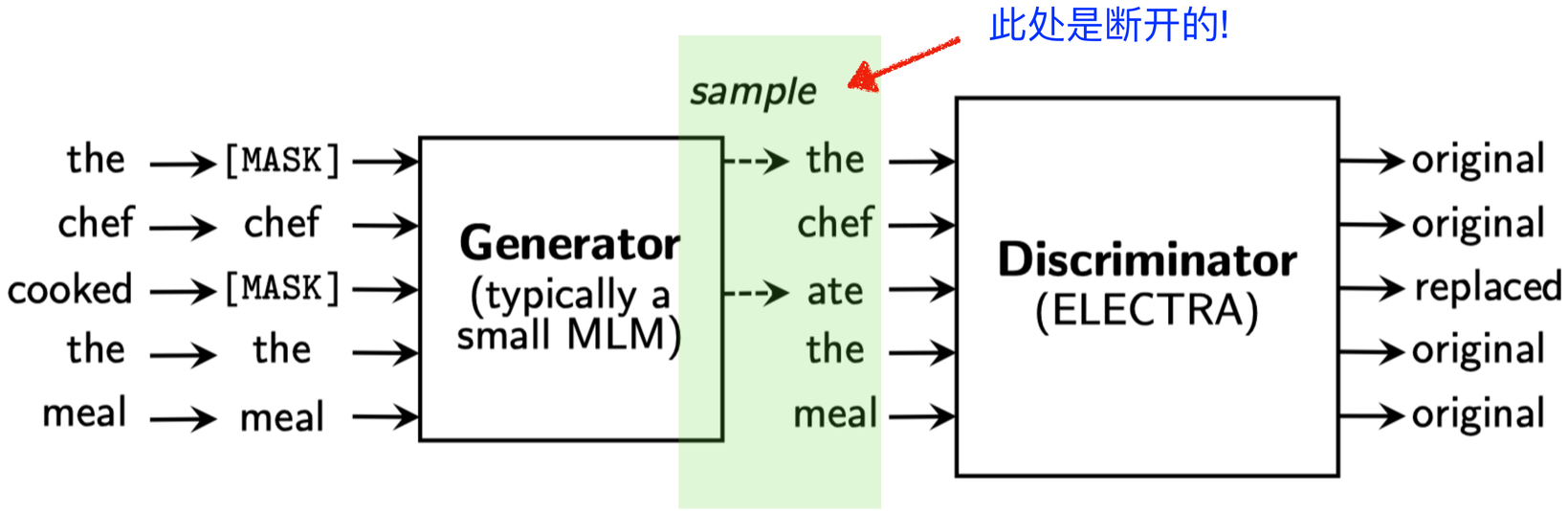
- **判别器：常规的BERT结构 + 预测输入单词是否被替换**
  - 提出Replace Token Detection（RTD）预训练任务
  - 第一步：将缺失信息替换为生成器预测出来的单词
  - 第二步：判别器学习判断输入句子中哪些单词被生成器替换过



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

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 $\min_{\theta_G,\theta_D} \sum_{\boldsymbol{x}\in\mathcal{X}} \mathcal{L}_{\text{MLM}}(\boldsymbol{x},\theta_G) + \lambda\mathcal{L}_{\text{Disc}}(\boldsymbol{x},\theta_D)$



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

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

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

ELECTRA

- 实验结果
  - ELECTRA-large在GLUE-dev/test上获得了当时最优效果

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**

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

MacBERT

- **基于纠错型掩码的预训练模型MacBERT**
  - 在可比条件下，评测了主流预训练模型在中文下的表现
  - 提出了一种新的预训练模型MacBERT，解决BERT中的“预训练-精调”不一致的问题
  - 基于相似词替换的预训练任务（**M**LM **a**s **c**orrection，Mac），类似“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

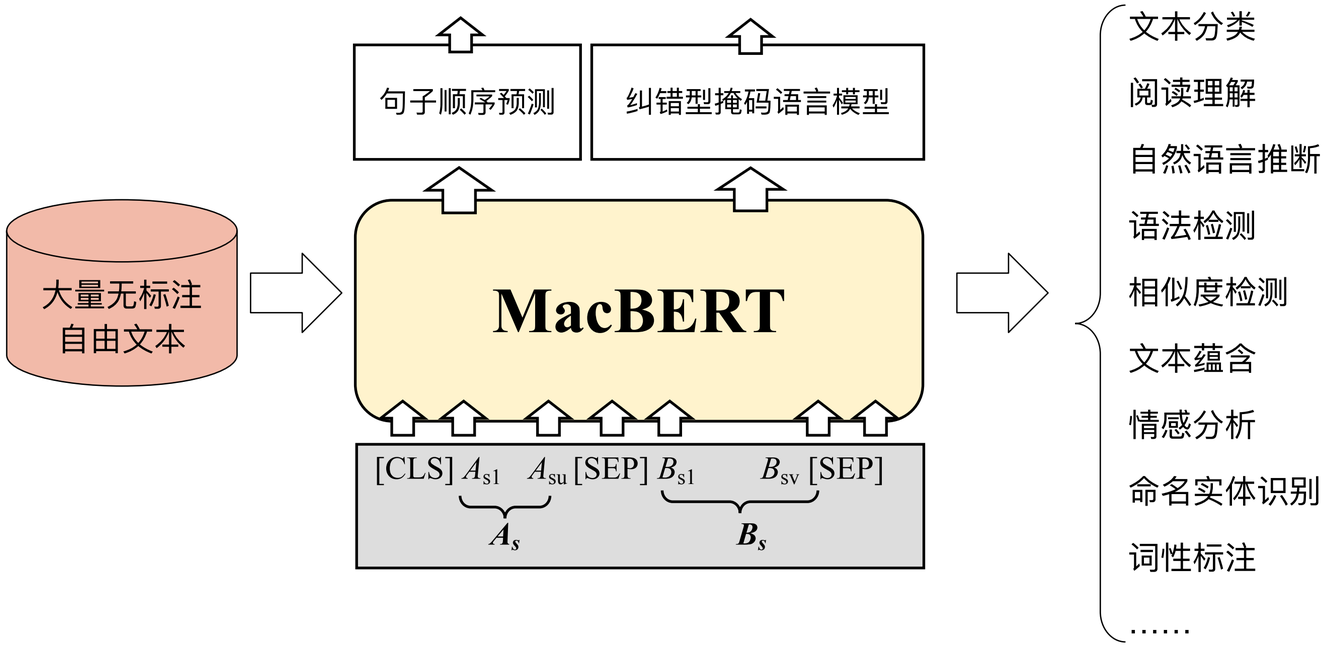
MacBERT

- 80% of the time, replace with synonym
  - 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., 2020] Revisiting Pre-trained Models for Chinese Natural Language Processing. In Findings of EMNLP 2020.
[Cui et al., 2021] Pre-training with Whole Word Masking for Chinese BERT. In IEEE/ACM TASLP Vol.28.

MacBERT

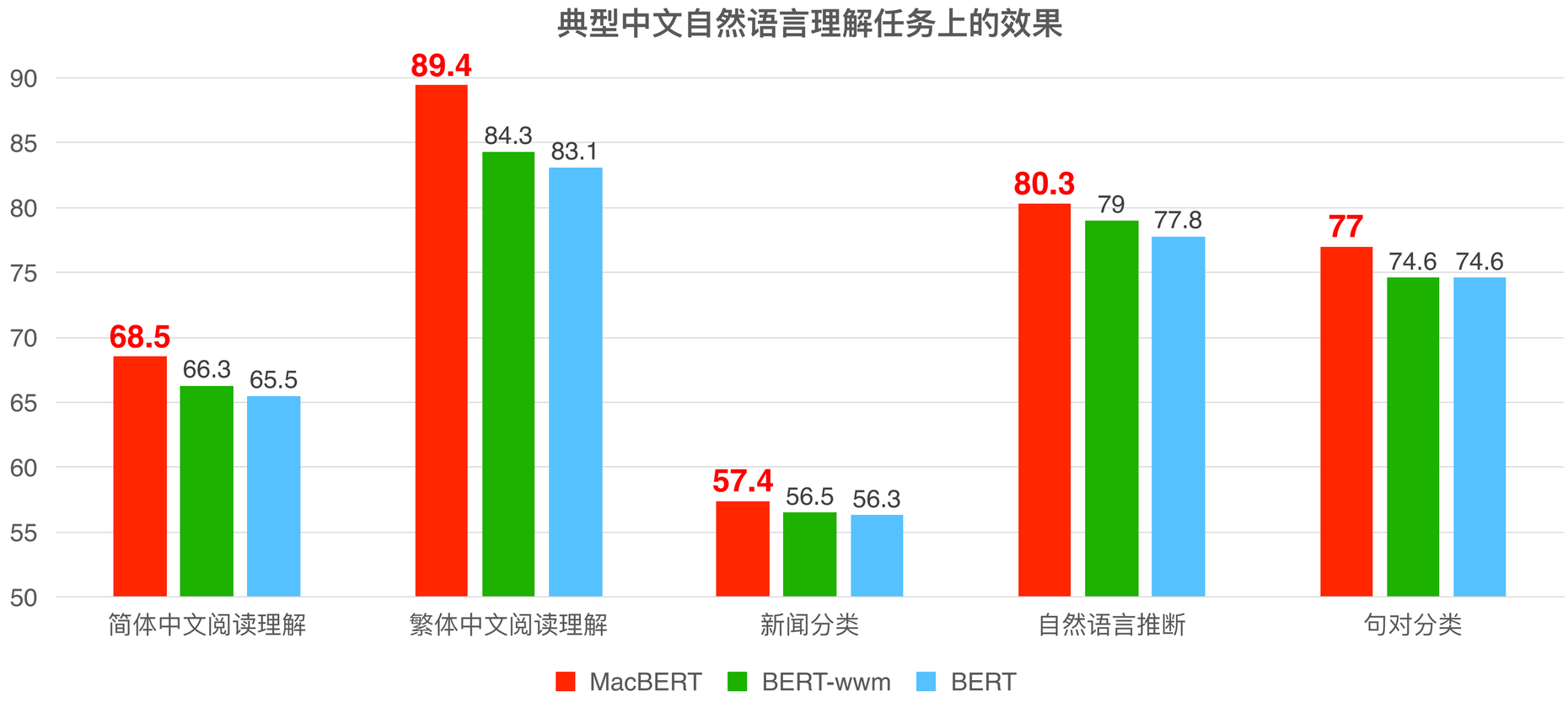
- **模型结构及训练任务**
  - **LM任务**：使用基于相似词替换的预训练任务进行训练
  - **SP任务**：将“下一个句子预测（NSP）”替换为ALBERT中的“句子顺序预测（SOP）”任务



[Cui et al., 2020] Revisiting Pre-trained Models for Chinese Natural Language Processing. In Findings of EMNLP 2020.
[Cui et al., 2021] Pre-training with Whole Word Masking for Chinese BERT. In IEEE/ACM TASLP Vol.28.

MacBERT

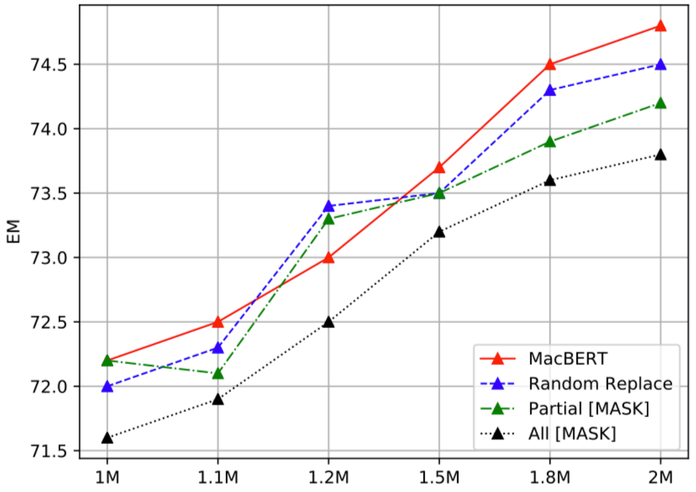
- **实验结果**
  - 在阅读理解、文本分类、自然语言推断等任务上获得显著性能提升



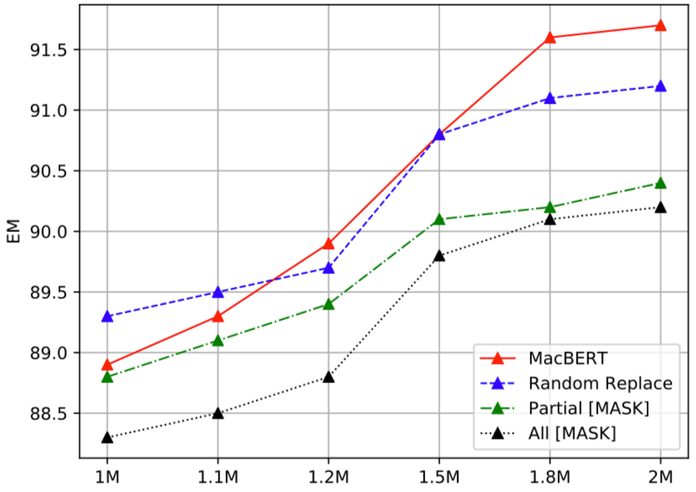
[Cui et al., 2020] Revisiting Pre-trained Models for Chinese Natural Language Processing. In Findings of EMNLP 2020.
[Cui et al., 2021] Pre-training with Whole Word Masking for Chinese BERT. In IEEE/ACM TASLP Vol.28.

MacBERT

- **实验分析：MLM任务**
  - 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）
  - 效果排序
    - MacBERT > random replace > partial mask > all mask



CMRC 2018



DRCD

▲ 横轴：训练步数；纵轴：EM值

[Cui et al., 2020] Revisiting Pre-trained Models for Chinese Natural Language Processing. In Findings of EMNLP 2020.
[Cui et al., 2021] Pre-training with Whole Word Masking for Chinese BERT. In IEEE/ACM TASLP Vol.28.

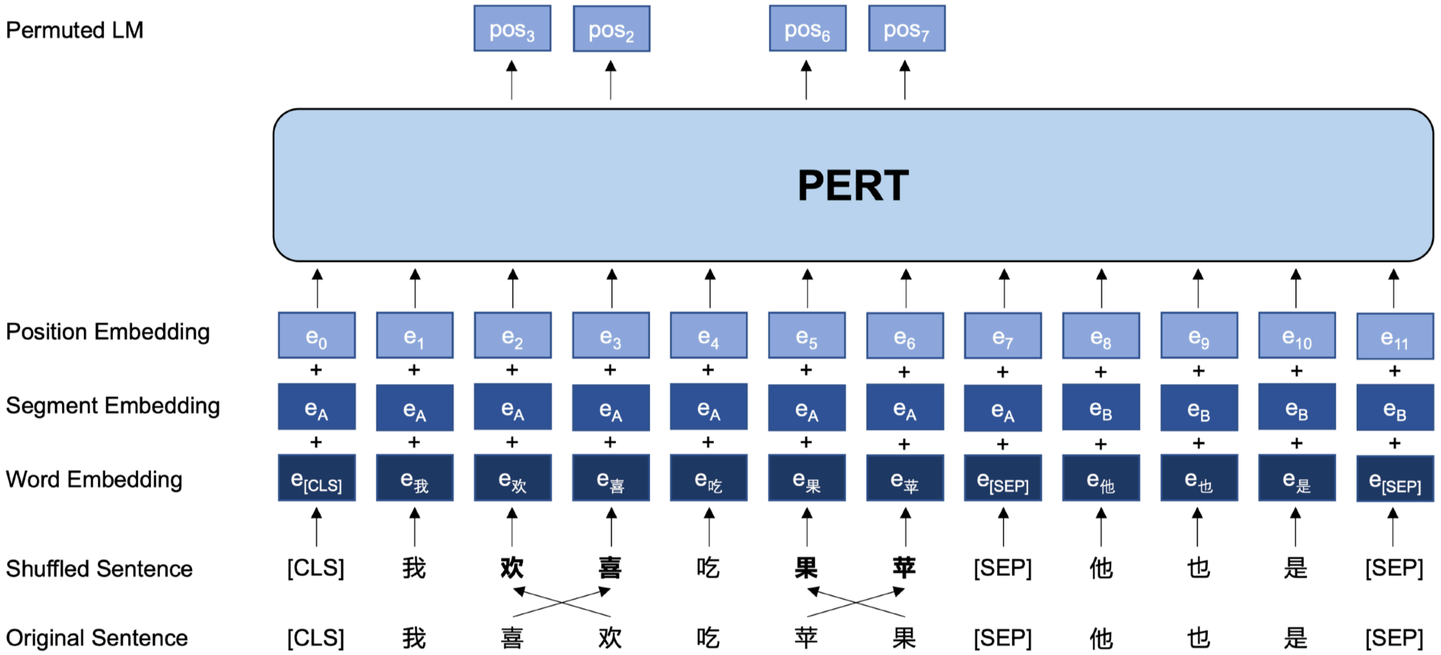
PERT

- **PERT: Pre-training BERT with Permuted Language Model**
  - 研究背景
    - 多数自编码预训练模型基于MLM及其变种进行训练
    - 基于MLM的训练方法依赖[MASK]标记，造成“预训练-精调”不一致的问题
  - 问题：有没有其他可以建模文本语义的方式?
  - 观察：发现乱序文本仍然包含与原文本相近的语义

[Cui et al., 2022] PERT: Pre-training BERT with Permuted Language Model. arXiv pre-print.

PERT

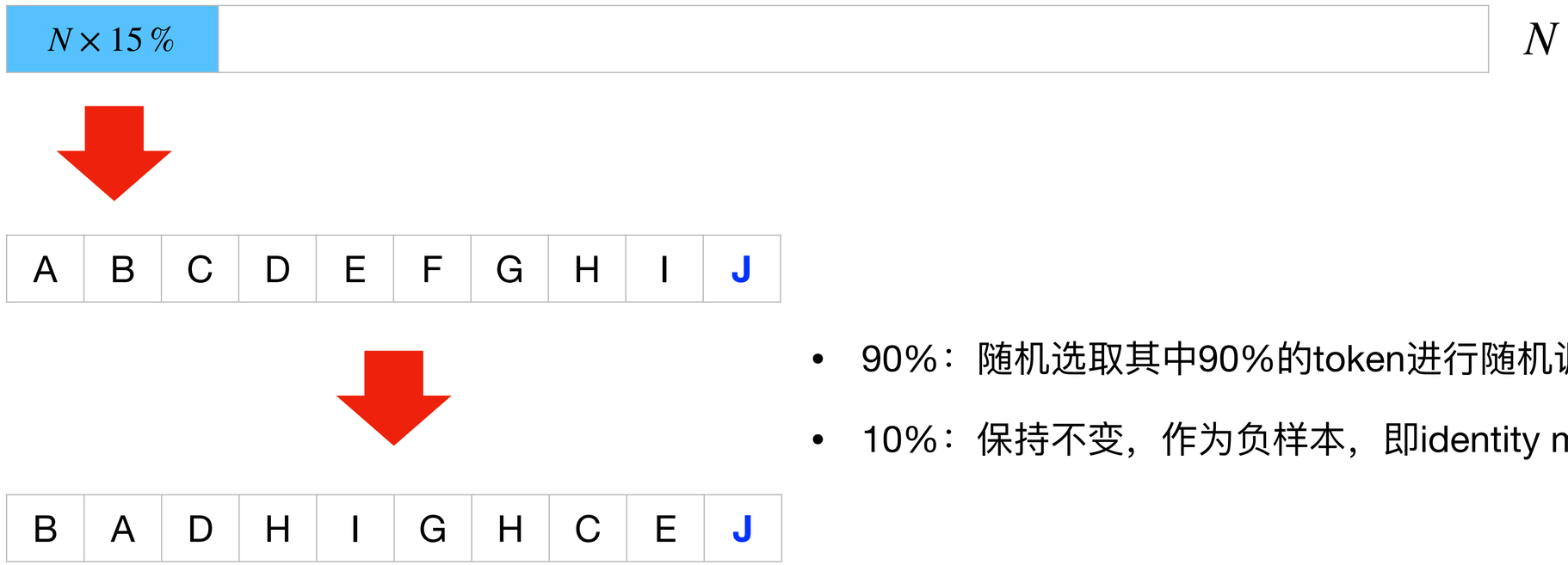
- **模型结构**
  - 提出利用乱序文本预测当前单词在原句的位置，自监督地学习文本语义信息
  - 该过程不会引入掩码标记[MASK]
  - 输出空间是输入序列长度N，而非词表大小V



[Cui et al., 2022] PERT: Pre-training BERT with Permuted Language Model. arXiv pre-print.

PERT

- **Permuted Language Model (PerLM)**
  - 随机选取输入序列长度15%的文本用于掩码
  - 同时使用全词掩码（wwm）和N-gram掩码技术（unigram 40% → 4-gram 10%）



  - 90%：随机选取其中90%的token进行随机调序
  - 10%：保持不变，作为负样本，即identity mapping

[Cui et al., 2022] PERT: Pre-training BERT with Permuted Language Model. arXiv pre-print.

PERT

- **实验效果：中文NLU任务**
  - 使用20G文本训练了PERT模型，其中包含维基百科、新闻、社区问答等语料
  - 实验结果：PERT在机器阅读理解、命名实体识别任务上能够获得显著性能提升

System	CMRC 2018						DRCD			
	D-EM	D-F1	T-EM	T-F1	C-EM	C-F1	D-EM	D-F1	T-EM	T-F1
$\text{BERT}_{\text{base}}$	67.1 (65.6)	85.7 (85.0)	71.4 (70.0)	87.7 (87.0)	24.0 (20.0)	47.3 (44.6)	85.0 (84.5)	91.2 (90.9)	83.6 (83.0)	90.4 (89.9)
$\text{RoBERTa}_{\text{base}}$	67.4 (66.5)	87.2 (86.5)	72.6 (71.4)	89.4 (88.8)	26.2 (24.6)	51.0 (49.1)	86.6 (85.9)	92.5 (92.2)	85.6 (85.2)	92.0 (91.7)
$\text{ELECTRA}_{\text{base}}$	68.4 (68.0)	84.8 (84.6)	73.1 (**72.7**)	87.1 (86.9)	22.6 (21.7)	45.0 (43.8)	87.5 (87.0)	92.5 (92.3)	86.9 (86.6)	91.8 (91.7)
$\text{MacBERT}_{\text{base}}$	**68.5** (67.3)	**87.9 (87.1)**	**73.2** (72.4)	**89.5 (89.2)**	**30.2** (26.4)	54.0 (52.2)	89.4 (**89.2**)	**94.3 (94.1)**	**89.5 (88.7)**	**93.8 (93.5)**
$\text{PERT}_{\text{base}}$	**68.5 (68.1)**	87.2 (**87.1**)	72.8 (72.5)	89.2 (89.0)	28.7 (**28.2**)	**55.4 (53.7)**	**89.5** (88.9)	93.9 (93.6)	89.0 (88.5)	93.5 (93.2)
$\text{RoBERTa}_{\text{large}}$	68.5 (67.6)	88.4 (87.9)	74.2 (72.4)	90.6 (90.0)	31.5 (30.1)	60.1 (57.5)	89.6 (89.1)	94.8 (94.4)	89.6 (88.9)	94.5 (94.1)
$\text{ELECTRA}_{\text{large}}$	69.1 (68.2)	85.2 (84.5)	73.9 (72.8)	87.1 (86.6)	23.0 (21.6)	44.2 (43.2)	88.8 (88.7)	93.3 (93.2)	88.8 (88.2)	93.6 (93.2)
$\text{MacBERT}_{\text{large}}$	70.7 (68.6)	88.9 (88.2)	74.8 (73.2)	**90.7** (90.1)	31.9 (29.6)	**60.2** (57.6)	**91.2 (90.8)**	**95.6 (95.3)**	**91.7 (90.9)**	**95.6 (95.3)**
$\text{PERT}_{\text{large}}$	**72.2 (71.0)**	**89.4 (88.8)**	**76.8 (75.5)**	**90.7 (90.4)**	**32.3 (30.9)**	59.2 (**58.1**)	90.9 (**90.8**)	95.5 (95.2)	91.1 (90.7)	95.2 (95.1)

▲ 阅读理解效果

System	MSRA-NER (Test)			People's Daily (Dev)		
	P	R	F	P	R	F
$\text{BERT}_{\text{base}}$	95.2 (94.8)	95.4 (95.1)	95.3 (94.9)	95.3 (**95.1**)	95.3 (95.1)	**95.3** (**95.1**)
$\text{RoBERTa}_{\text{base}}$	95.3 (94.9)	95.6 (95.4)	95.5 (95.1)	94.9 (94.8)	95.3 (95.1)	95.1 (94.9)
$\text{ELECTRA}_{\text{base}}$	95.0 (94.5)	**95.9** (95.4)	95.4 (95.0)	94.8 (94.7)	95.3 (**95.2**)	95.1 (94.9)
$\text{MacBERT}_{\text{base}}$	95.2 (94.9)	95.4 (95.4)	95.3 (95.1)	94.9 (94.6)	95.6 (95.1)	95.2 (94.9)
$\text{PERT}_{\text{base}}$	**95.4** (**95.2**)	95.5 (**95.5**)	**95.6** (**95.3**)	**95.4** (**95.1**)	95.2 (95.0)	**95.3** (**95.1**)
$\text{RoBERTa}_{\text{large}}$	95.4 (95.3)	95.7 (95.7)	95.5 (95.5)	95.7 (95.4)	95.7 (95.4)	95.7 (95.4)
$\text{ELECTRA}_{\text{large}}$	94.9 (94.8)	95.5 (95.0)	95.0 (94.8)	94.8 (94.6)	95.3 (95.3)	94.9 (94.8)
$\text{MacBERT}_{\text{large}}$	96.3 (95.8)	96.3 (95.9)	**96.2** (95.9)	95.8 (95.6)	95.8 (**95.7**)	95.8 (95.7)
$\text{PERT}_{\text{large}}$	**96.4** (**95.9**)	**96.4** (**96.1**)	**96.2** (**96.0**)	**96.3** (**96.0**)	96.0 (**95.7**)	**96.1** (**95.8**)

▲ 命名实体识别效果

[Cui et al., 2022] PERT: Pre-training BERT with Permuted Language Model. arXiv pre-print.

PERT

- **实验效果：中文语法纠错任务**
  - 针对语法中的文本乱序问题，利用PERT搭建序列标注模型进行实验
  - 在四个领域（维基、公文、海关、政法）下，PERT均显著优于所有基线系统效果

我每天一个吃苹果 ⟶ 我每天吃一个苹果

System	Wikipedia			Formal Doc.			Customs			Legal		
	P	R	F	P	R	F	P	R	F	P	R	F
$\text{BERT}^{\text{Google}}_{\text{base}}$	83.6	76.3	79.8	92.1	87.1	89.6	85.7	85.1	85.4	94.3	89.8	92.0
$\text{RoBERTa}_{\text{base}}$	84.2	76.9	80.4	92.6	87.7	90.1	86.8	85.9	86.3	94.6	90.0	92.2
$\text{ELECTRA}_{\text{base}}$	69.9	57.8	63.6	88.1	81.6	84.7	69.6	71.0	70.3	91.7	85.4	88.4
$\text{MacBERT}_{\text{base}}$	84.3	77.1	80.5	92.7	87.8	90.2	86.4	86.5	86.4	94.6	90.1	92.3
$\text{PERT}_{\text{base}}$	**86.5**	**79.5**	**82.9**	**93.6**	**89.0**	**91.2**	**88.3**	**88.0**	**88.2**	**95.2**	**90.7**	**92.9**

▲ 文本纠错-乱序问题效果

[Cui et al., 2022] PERT: Pre-training BERT with Permuted Language Model. arXiv pre-print.

PERT

- **实验效果：英文NLU任务**
  - 使用了经典的Wikipedia+Books数据进行训练
  - 实验结果：部分任务上优于其他同等数据训练的预训练模型

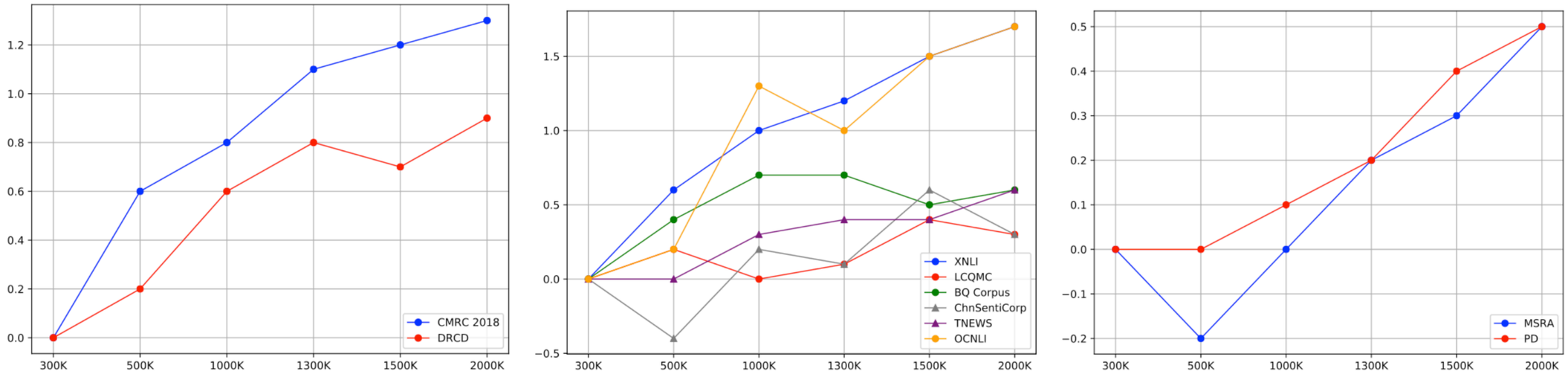
System	SQuAD EM	SQuAD F1	SQuAD 2.0 EM	SQuAD 2.0 F1	MNLI Acc	SST-2 Acc	CoLA M.C.	MRPC Acc
$\text{BERT}_{\text{base}}$	80.8	88.5	-	-	84.4	**92.7**	60.6	86.7
$\text{BERT}_{\text{base}}^{\dagger}$	81.2	88.5	72.4	75.4	84.4	92.6	59.3	86.0
$\text{RoBERTa}_{\text{base}}$	-	90.4	-	79.1	84.7	92.5	-	-
$\text{XLNet}_{\text{base}}$	-	-	**78.4**	**81.3**	**85.8**	92.6	-	-
$\text{ALBERT}_{\text{base}}$	82.1	89.3	76.1	79.1	81.9	89.4	-	-
$\text{PERT}_{\text{base}}$	**84.8**	**91.3**	78.3	81.0	84.5	92.0	**61.2**	**87.5**
$\text{BERT}_{\text{large}}$	84.1	90.9	78.7	81.9	86.6	93.2	60.6	88.0
$\text{BERT}_{\text{large-wwm}}^{\dagger}$	87.4	93.4	82.8	85.6	87.3	93.4	63.1	87.2
$\text{RoBERTa}_{\text{large}}$	-	93.6	-	87.3	**89.0**	**95.3**	-	-
$\text{XLNet}_{\text{large}}$	**88.2**	**94.0**	**85.1**	**87.8**	88.4	94.4	65.2	**90.0**
$\text{ALBERT}_{\text{large}}$	84.1	90.9	79.0	82.1	83.8	90.6	-	-
$\text{PERT}_{\text{large}}$	87.4	93.3	83.5	86.3	87.6	93.4	**65.7**	87.3

▲ 英文任务效果

[Cui et al., 2022] PERT: Pre-training BERT with Permuted Language Model. arXiv pre-print.

PERT

- **不同训练步数下的性能对比**
  - 阅读理解、命名实体识别任务性能趋势基本一致，随着训练步数增加，其性能也呈现增长趋势
  - 多数文本分类任务在训练中途达到局部最优效果
  - “预训练”也和“下游任务精调”一样，最终时刻并不一定是最好结果



(a) MRC (b) TC (c) NER

▲ **横轴**：训练步数；**纵轴**：以300K效果为基准计算性能差值

[Cui et al., 2022] PERT: Pre-training BERT with Permuted Language Model. arXiv pre-print.

PERT

- **分析：不同调序粒度**
  - 在PERT中，被选中的词可以和任何一个词进行调换，对文本自然度破坏更大
  - 分析不同调序粒度对性能的影响：word，N-gram，sentence
  - 实验结果显示
    - 调序粒度越小，虽然对文本的自然度破坏更小，却不利于文本的语义学习
    - 对阅读理解任务影响较大，对文本分类任务影响较小

System	CMRC 2018						XNLI		TNEWS	OCNLI	Average
	D-EM	D-F1	T-EM	T-F1	C-EM	C-F1	Dev	Test	Dev	Dev	
$\text{PERT}_{\text{base}}$ (no limit)	65.4	85.0	70.2	87.3	22.4	45.6	74.8	74.4	54.5	70.6	65.02
└Word	59.3	80.6	64.8	83.5	12.6	32.2	73.2	72.1	53.2	69.3	60.08
└N-gram	62.2	82.5	67.3	84.8	17.2	36.1	73.4	73.2	53.8	69.5	62.00
└Sentence	63.7	83.2	69.1	86.2	16.8	38.0	74.3	73.0	54.1	70.0	62.84

▲ 不同调换粒度下的实验结果

[Cui et al., 2022] PERT: Pre-training BERT with Permuted Language Model. arXiv pre-print.

PERT

- **分析：Global v.s. Local Prediction**
  - 在PERT框架下，在局部空间预测效果相对较好，结合两者并未带来额外的性能提升

System	CMRC 2018						XNLI		TNEWS	OCNLI	Average
	D-EM	D-F1	T-EM	T-F1	C-EM	C-F1	Dev	Test	Dev	Dev	
$\text{PERT}_{\text{base}}$ (local)	64.1	84.0	69.1	86.5	21.0	43.3	74.1	74.4	54.5	70.6	64.16
└Global	61.1	81.4	65.8	84.3	15.8	36.2	73.6	74.0	55.4	69.4	61.70
└Local + Global	63.2	83.6	67.7	85.8	19.3	42.0	74.6	74.6	55.1	70.0	63.59

▲ **Global**：在整个词表上预测，即 $y \in \mathbb{R}^{|V|}$，**Local**：在输入句子中预测，即 $y \in \mathbb{R}^{L}$

- **分析：Partial v.s. Full Prediction**
  - Full Prediction未能在PERT框架下带来更好的实验结果
  - 并非所有预训练任务都适合ELECTRA-style预测方式，即Full Prediction

System	CMRC 2018						XNLI		TNEWS	OCNLI	Average
	D-EM	D-F1	T-EM	T-F1	C-EM	C-F1	Dev	Test	Dev	Dev	
$\text{PERT}_{\text{base}}$ (partial)	64.1	84.0	69.1	86.5	21.0	43.3	74.1	74.4	54.5	70.6	64.16
└Full Prediction	63.7	84.1	68.0	86.1	18.7	40.9	74.3	73.9	54.2	71.0	63.49

▲ **Partial**：只对调换的token进行预测，**Full**：对输入序列中的所有token进行预测

[Cui et al., 2022] PERT: Pre-training BERT with Permuted Language Model. arXiv pre-print.

小结

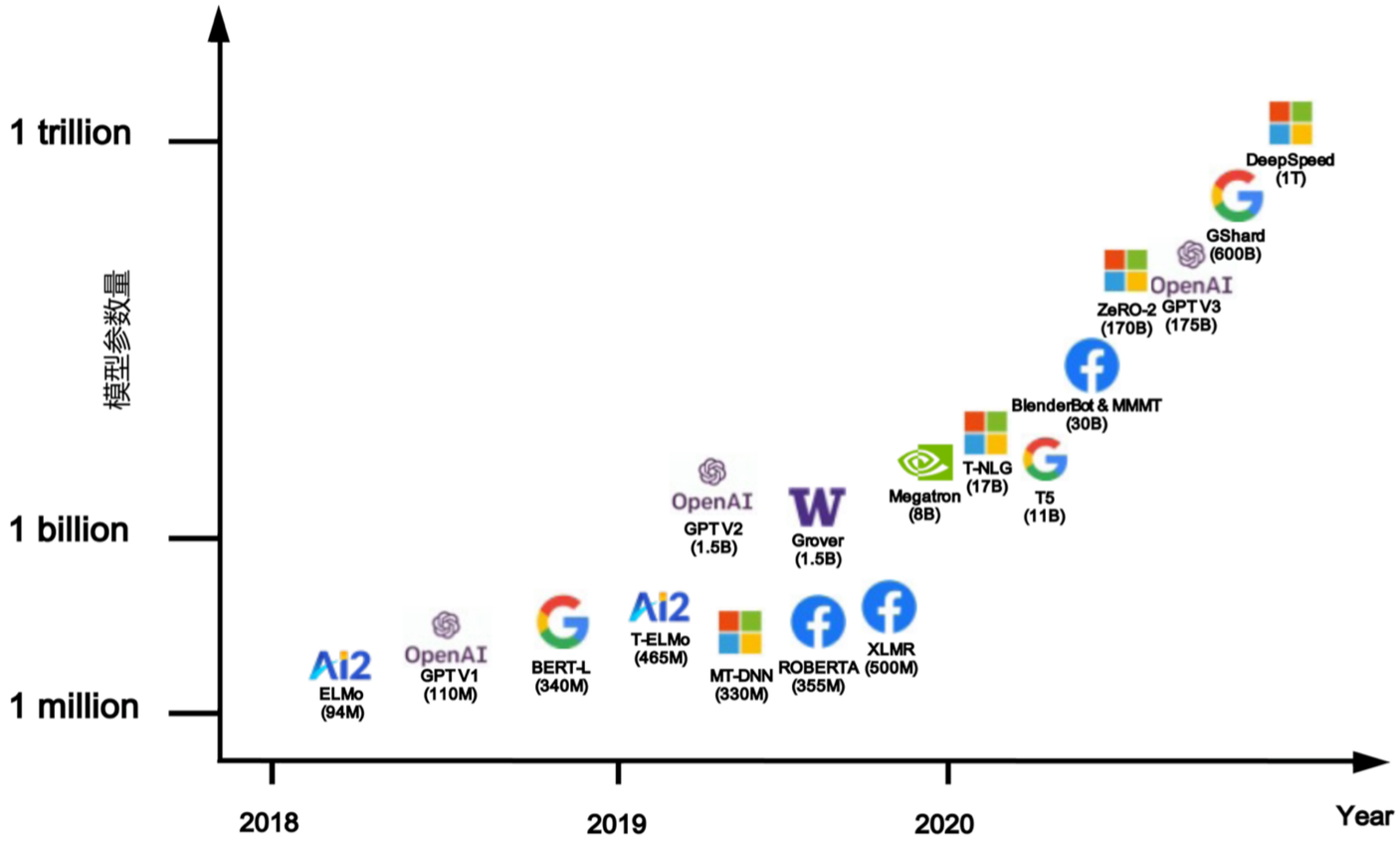
	GPT	BERT	XLNet	RoBERTa	ALBERT	ELECTRA	MacBERT	PERT
Type	AR	AE	AR	AE	AE	AE	AE	AE
Embedding	T	T/S/P	T/S/P	T/S/P	T/S/P	T/S/P	T/S/P	T/S/P
Masking	/	T	/	T	T	T	WWM+NM	WWM+NM
LM Task	LM	MLM	PLM	MLM	MLM	Gen-Dis	Mac	PerLM
Paired Task	/	NSP	/	/	SOP	/	SOP	/
Data Source	BC	BC+Wiki	BC+Wiki+Giga5 +CW+CC	BC+Wiki+CCNew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

预训练模型的蒸馏与裁剪

Distillation and Pruning for Pre-trained Models

预训练模型的蒸馏与裁剪

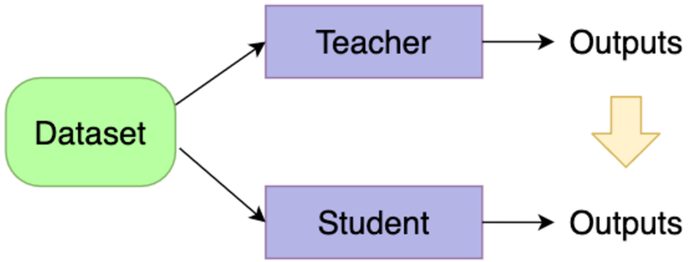
- **研究背景**
  - 预训练模型通常需要占用很大的空间，并且训练和推断时间也很慢，难以满足实际应用需求



预训练模型的蒸馏与裁剪

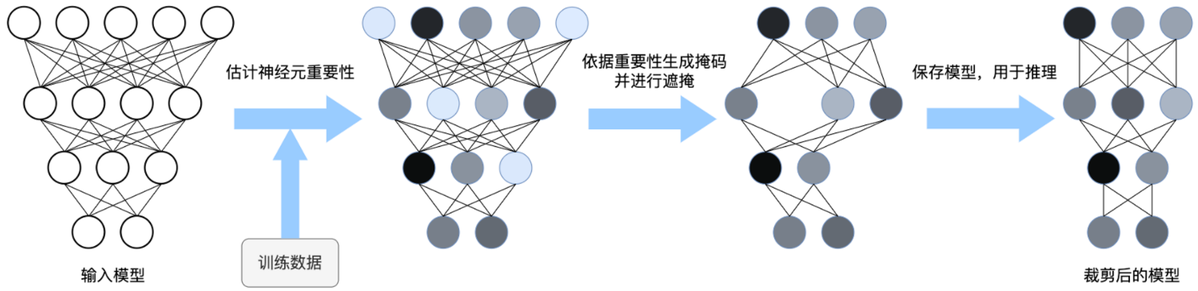
- **提升预训练模型效率的主要途径**
  - **知识蒸馏**：在少量性能损失的情况下，将大模型知识迁移到小模型，提升模型效果和推断速度
  - **模型裁剪**：对模型中的部分“不重要”或“冗余”的部分进行剪枝，从而缩小预训练模型体积

知识蒸馏



TextBrewer

模型裁剪



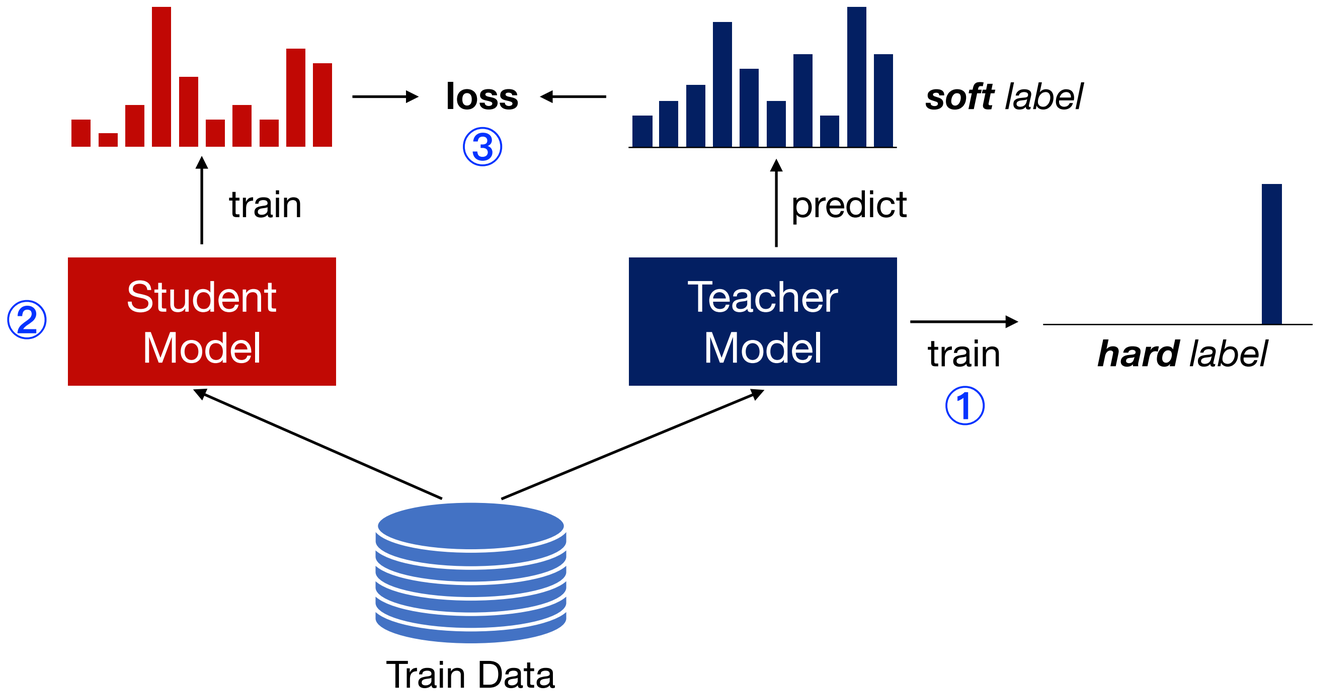
TextPruner

1 知识蒸馏工具TextBrewer

2 模型裁剪工具TextPruner

知识蒸馏

- **知识蒸馏是什么?**
  - 在少量性能损失的情况下，将大模型知识迁移到小模型，可以看作是**“老师”教“学生”的过程**
  - 知识蒸馏技术经常被用于减小模型体积、提升模型效果和推断速度



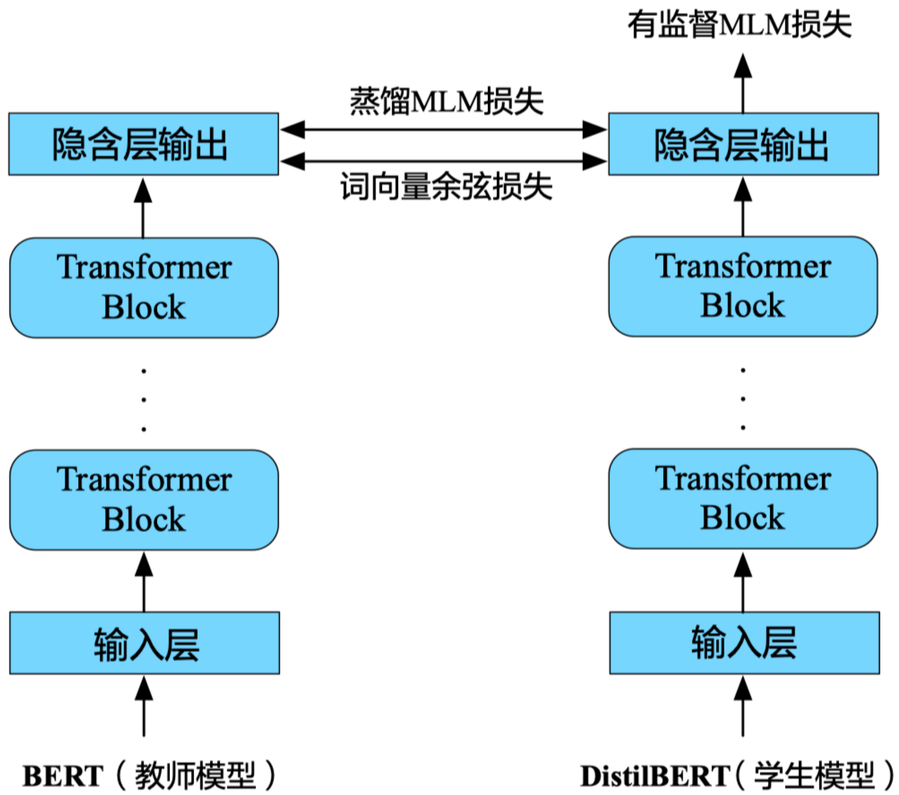
① 用数据训练一个教师模型

② 定义并初始化一个学生模型

③ 利用合适的训练损失，让学生模型学习教师模型

知识蒸馏

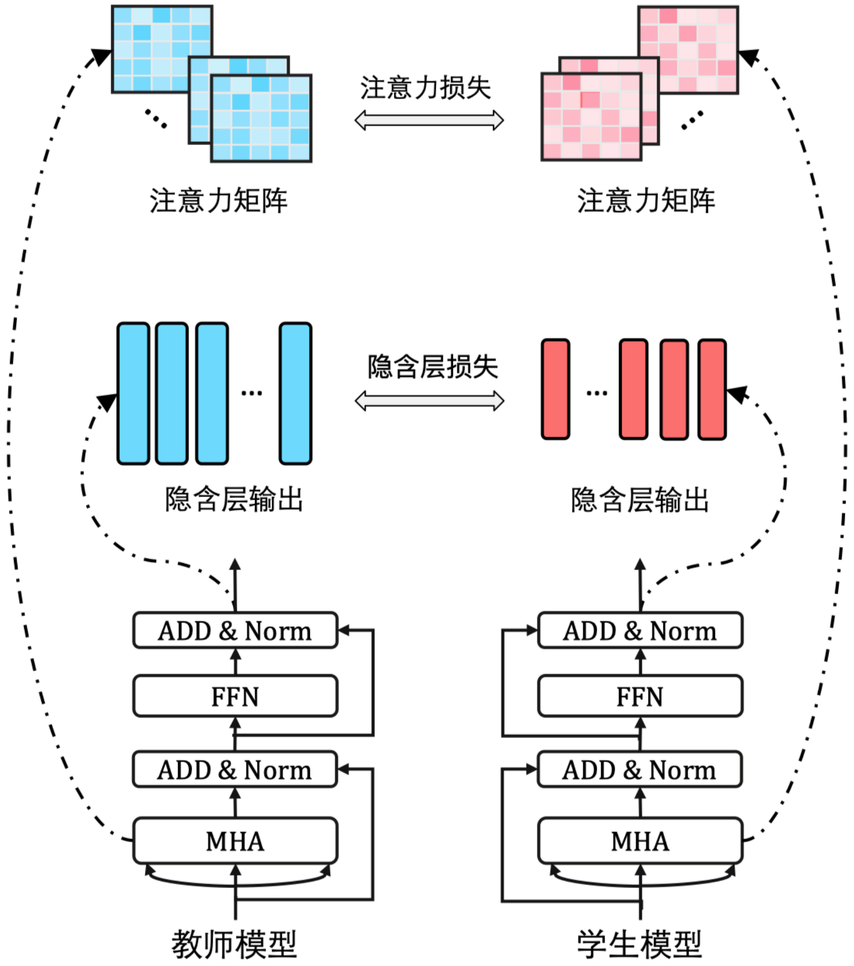
- **DistilBERT**
  - 学生模型（即DistilBERT）是一个通用的6层Transformer结构
  - 使用MLM预训练任务进行知识蒸馏（无NSP）
  - 训练目标由以下三部分组成
    - 有监督MLM损失：由数据集自带的硬标签（hard-label）计算
    - 蒸馏MLM损失：由教师模型提供的软标签（soft-label）计算
    - 余弦相似度损失：教师模型和学生模型隐层输出之间计算
  - 相比原始BERT-base，小40%、快60%、在自然语言理解任务上可达到原模型效果的97%



[Sanh et al., 2019] DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. In arXiv pre-print.

知识蒸馏

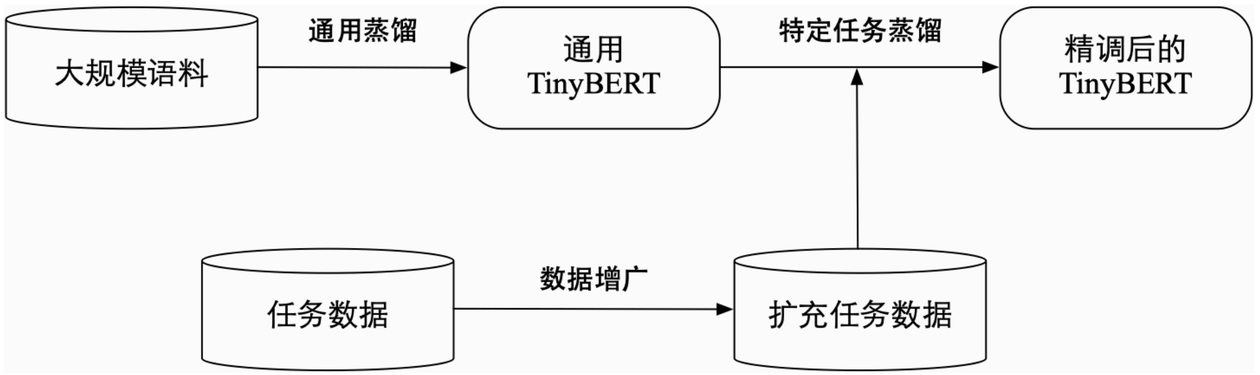
- **TinyBERT**
  - 提出了一种对BERT不同层进行匹配的蒸馏策略
  - 蒸馏损失由3部分组成：词向量损失、中间层损失、预测层损失
    - 词向量层损失：计算教师和学生模型的词向量均方误差
    - 中间层损失：计算教师模型第 i 层与学生模型第 j 层隐层输出（或注意力矩阵）的均方误差
      - 文中使用匹配函数为 $j = g(i) = 3i$
    - 输出层损失：计算软标签损失
  - TinyBERT能达到教师模型BERT-base的96%的效果（GLUE），大小只有教师的13.3%



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

知识蒸馏

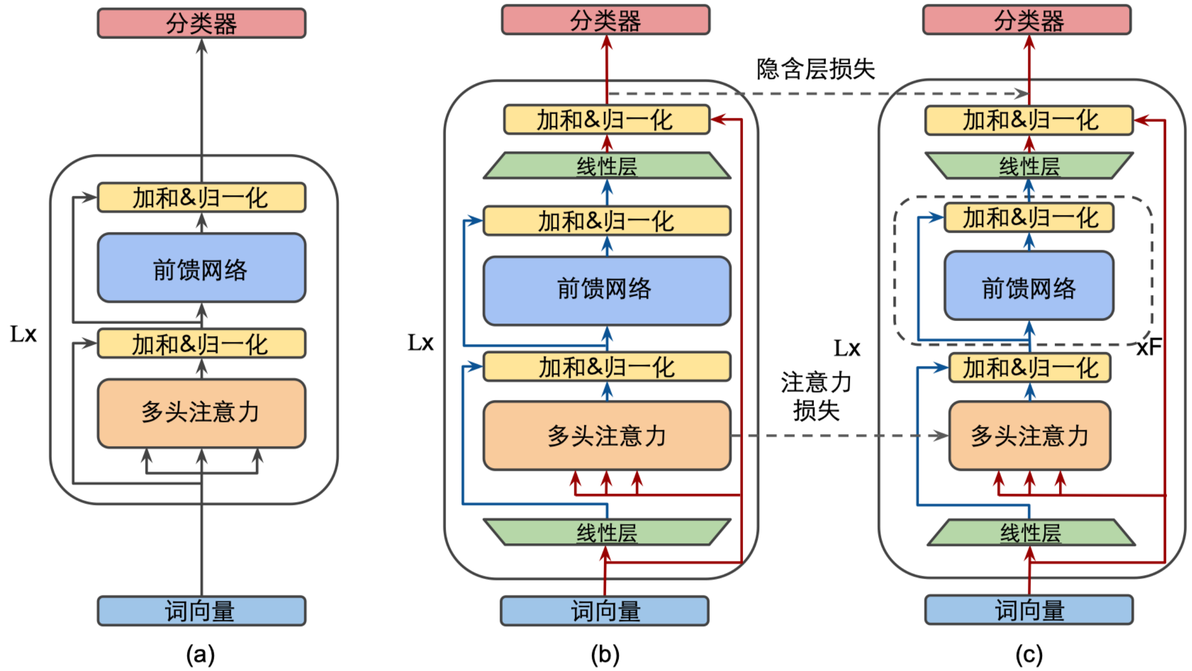
- **TinyBERT**
  - 提出了两阶段蒸馏策略，在预训练和精调阶段均进行知识蒸馏
  - 通用蒸馏（预训练阶段）
    - 利用原始BERT作为教师，并使用大规模文本训练MLM任务
  - 特定任务蒸馏（精调阶段）
    - 使用精调过的BERT作为教师，使用数据增广后的任务数据进行蒸馏



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

知识蒸馏

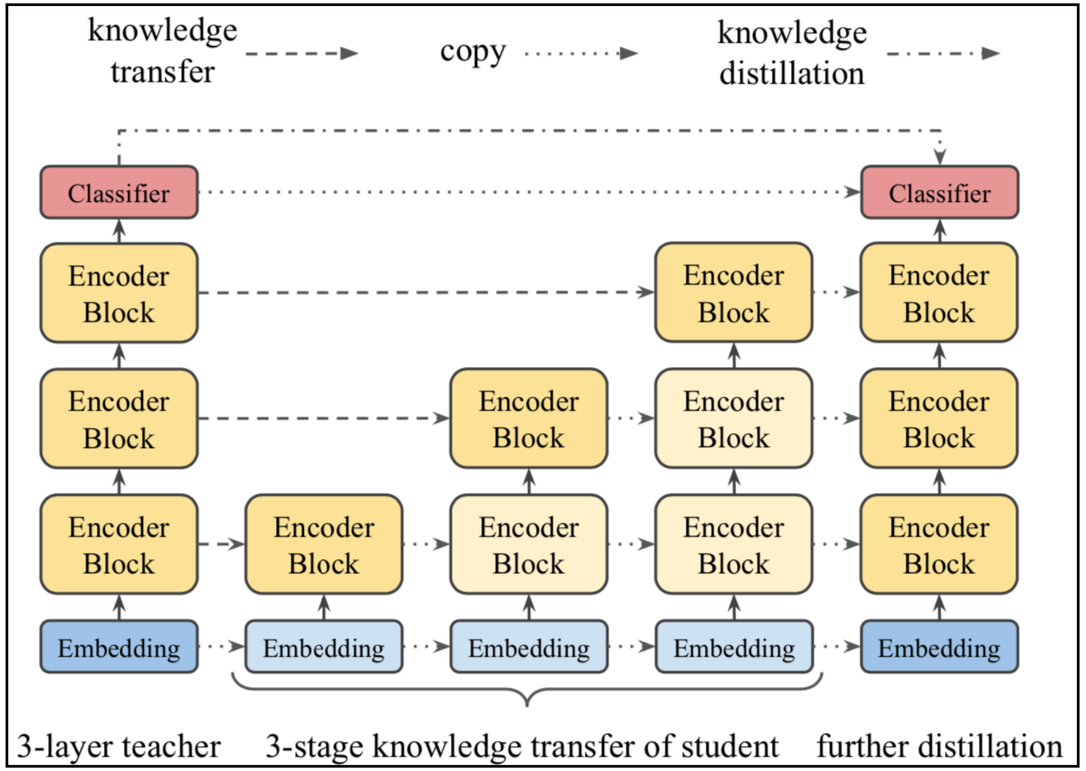
- **MobileBERT**
  - MobileBERT与BERT-large深度相同但更“苗条”，在自注意力和前馈神经网络的设计上也有一定改进
  - 教师模型：训练一个24层BERT（称之为IB-BERT）
  - 训练目标
    - 普通的MLM蒸馏损失
    - 隐层匹配损失
    - 注意力匹配损失与TinyBERT类似，但使用了KL散度
  - 能够达到教师模型99.2%的性能效果（GLUE基准），推理速度快5.5倍，参数量降低至23.2%



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

知识蒸馏

- **MobileBERT**
  - 提出了一种渐进式知识迁移方法（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., 2020] MobileBERT: a Compact Task-Agnostic BERT for Resource-Limited Devices. In ACL 2020.

TextBrewer

- **TextBrewer: An Open-Source Knowledge Distillation Toolkit for NLP**
  - 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
  - 访问 http://textbrewer.hfl-rc.com/ 或通过 `pip install textbrewer`安装

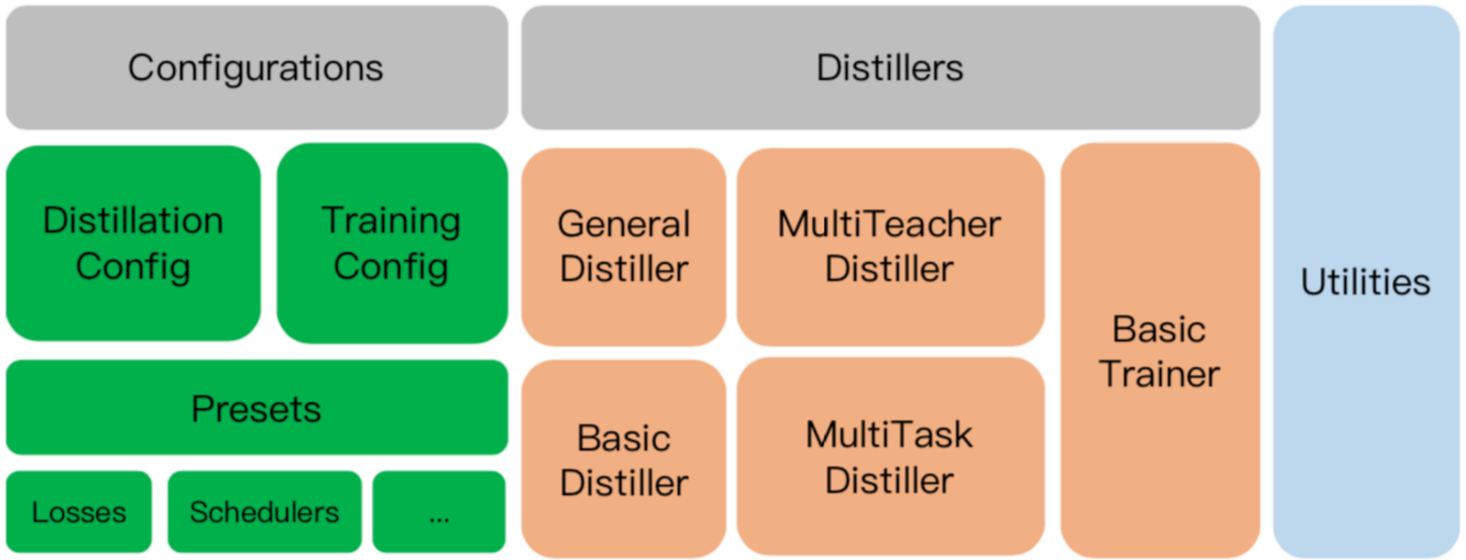
TextBrewer 1200+ ⭐

- **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

[Yang et al., 2020] TextBrewer: An Open-Source Knowledge Distillation Toolkit for Nature Language Processing. In ACL 2020.

TextBrewer

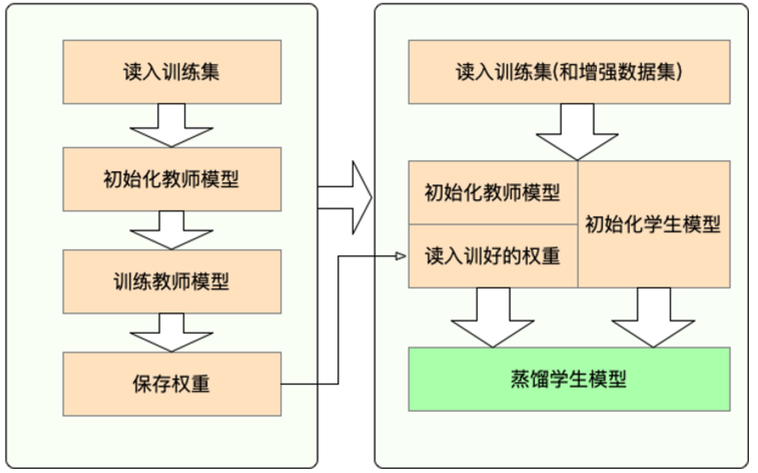
- **工具包设计架构**
  - Distillers：用于执行实际的知识蒸馏工作，定义了常规蒸馏策略
  - Configurations：为Distillers提供必要的配置信息
  - Utilities：包含一些辅助的功能，如模型参数统计等



[Yang et al., 2020] TextBrewer: An Open-Source Knowledge Distillation Toolkit for Nature Language Processing. In ACL 2020.

TextBrewer

- 工作流程
  - **第一步：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
  - 蒸馏已经预设的相关任务，只需要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., 2020] TextBrewer: An Open-Source Knowledge Distillation Toolkit for Nature Language Processing. In ACL 2020.

TextBrewer

- **知识蒸馏效果**
  - 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%，模型体积缩小至60%
    - T4-tiny知识蒸馏结果优于TinyBERT
  - 多教师知识蒸馏
    - 蒸馏后的学生模型获得最优效果，且超过简单的模型融合方法（ensemble）

Model	MNLI m	MNLI mm	SQuAD EM	SQuAD F1	CoNLL-2003 F1
$\text{BERT}_{\text{BASE}}$	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

▲ 单教师蒸馏效果

Model	MNLI m	MNLI mm	SQuAD EM	SQuAD F1	CoNLL-2003 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**

▲ 多教师蒸馏效果

[Yang et al., 2020] TextBrewer: An Open-Source Knowledge Distillation Toolkit for Nature Language Processing. In ACL 2020.

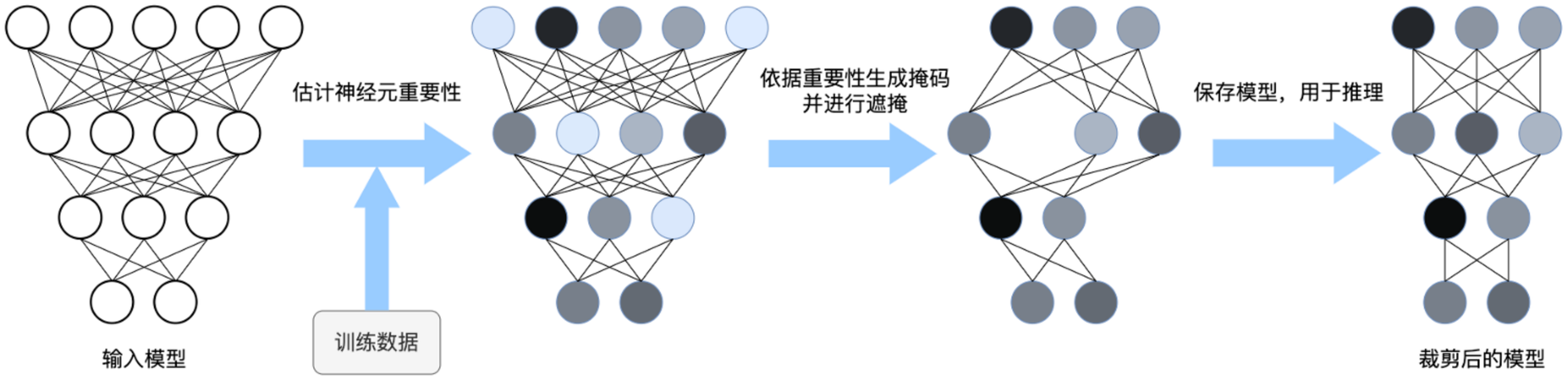
1 知识蒸馏工具TextBrewer

2 模型裁剪工具TextPruner

模型裁剪

- **模型裁剪是什么?**
  - 对模型中的部分“不重要”或“冗余”的权重或网络结构进行剪枝，从而缩小预训练模型体积、提升推理速度
    - 与知识蒸馏等技术正交，可相互叠加使用
    - （与知识蒸馏相比）无需设计学生模型，小模型的结构由算法得到

	非结构化裁剪	结构化裁剪
裁剪粒度	单个神经元	整个矩阵，或其整行整列
加速设备	需要特殊硬件	大多数计算硬件
压缩率	较高	中等
权重稀疏性	稀疏矩阵	稠密矩阵



模型裁剪

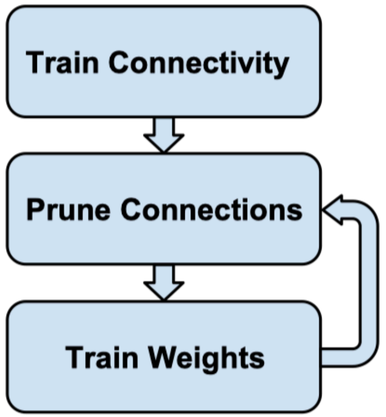
- **非结构化裁剪典型方法：Magnitude Pruning**
  - 算法流程

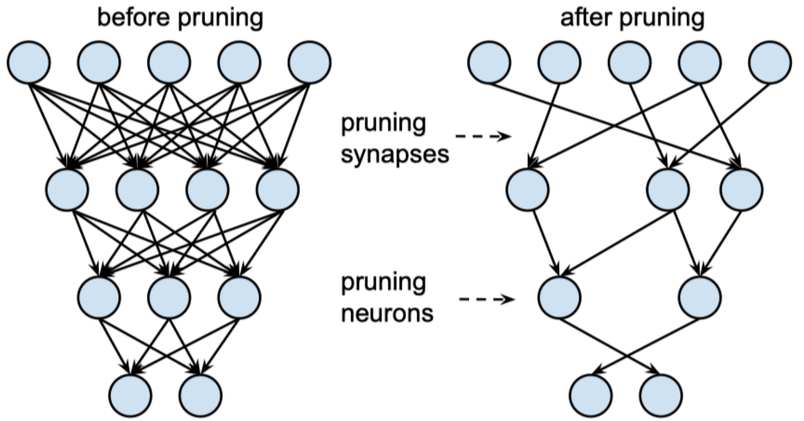
    ① 训练模型到收敛状态

    ② 选取一部分magnitude较小的权重，将其值置为0并保持固定

    ③ 再次（继续）训练模型

    ④ 回到第2步，直到模型模型达到指定稀疏度





[Han et al., 2015] Learning both Weights and Connections for Efficient Neural Networks. In NeurIPS 2015.

模型裁剪

- **结构化裁剪典型方法：Gradient-based Importance Score**
  - 利用损失函数对神经元的敏感性（一阶泰勒展开）信息，衡量Attention head的重要性
  - 该方法可同样推广至模型中的其他结构

$$I_{\mathbf{h}} = |\mathcal{L}_{\mathbf{h}} - \mathcal{L}_{\mathbf{h}=\mathbf{0}}| = \left|\mathcal{L}_{\mathbf{h}} - (\mathcal{L}_{\mathbf{h}} - \frac{\partial \mathcal{L}}{\partial \mathbf{h}}(\mathbf{h}-\mathbf{0}) + R_{\mathbf{h}=\mathbf{0}})\right| \approx \left|\frac{\partial \mathcal{L}}{\partial \mathbf{h}}\mathbf{h}\right|$$

$$\mathrm{MHAtt}(\mathbf{x}, q) = \sum_{h=1}^{N_h} \xi_h \mathrm{Att}_{W_k^h, W_q^h, W_v^h, W_o^h}(\mathbf{x}, q)$$

$$I_h = \mathbb{E}_{x \sim X} \left|\mathrm{Att}_h(x)^T \frac{\partial \mathcal{L}(x)}{\partial \mathrm{Att}_h(x)}\right|$$

[Michel et al., 2019] Are Sixteen Really Better than One? In NeurIPS 2019.

TextPruner

- **TextPruner: A Model Pruning Toolkit for Pre-trained Language Model**
  - 推出首个面向自然语言处理领域的基于PyTorch的模型裁剪工具包
  - 通过轻量、快速的裁剪方法对模型进行结构化剪枝，从而实现压缩模型体积、提升模型速度
  - 访问 http://textpruner.hfl-rc.com/ 或通过 `pip install textpruner`安装

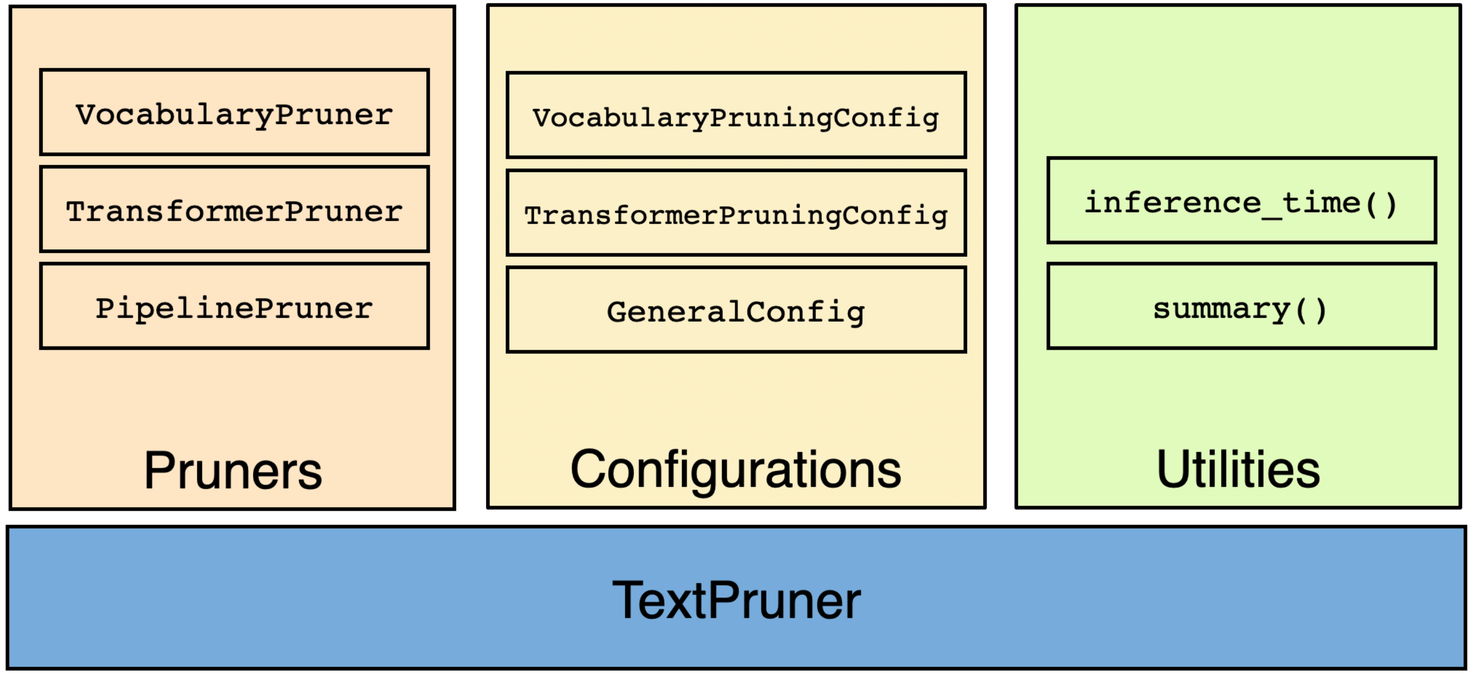
TextPruner 190+ ⭐

- **功能通用：**适配多种预训练模型，以及多种NLU任务
- **可定制化：**除了标准PLM外，也可使用TextPruner裁剪基于标准PLM开发的自定义模型
- **灵活便捷：**可作为Python包在Python脚本中使用，也提供了独立的命令行工具
- **运行高效：**使用无训练的结构化裁剪方法，运行迅速，快于知识蒸馏等基于训练的方法

[Yang et al., 2022] TextPruner: A Model Pruning Toolkit for Pre-trained Language Models. In ACL 2022.

TextPruner

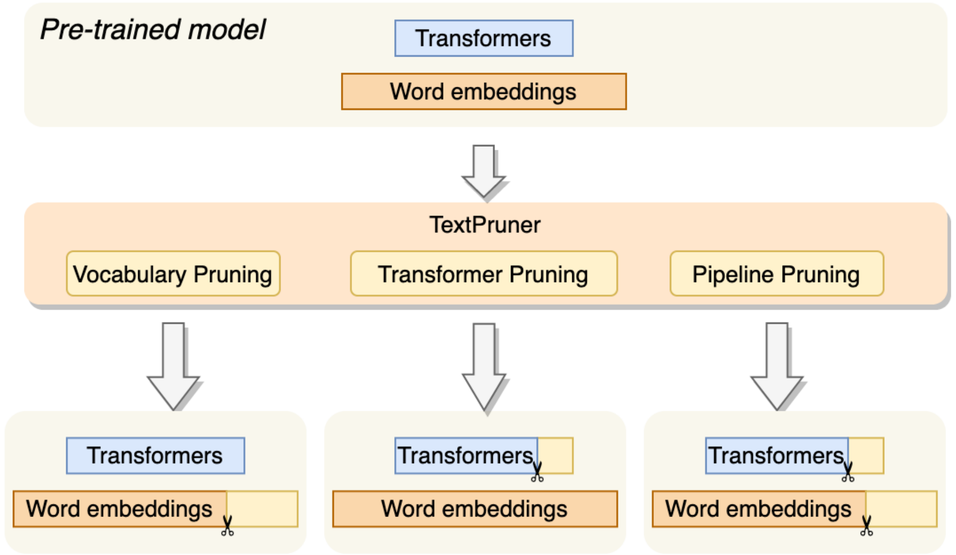
- **工具包设计架构**
  - Pruners：用于执行实际的模型裁剪，包含词表裁剪、Transformer裁剪、流水线裁剪等
  - Configurations：为Distillers提供必要的配置信息
  - Utilities：包含一些辅助的功能，如模型参数统计、计算推断时间等



[Yang et al., 2022] TextPruner: A Model Pruning Toolkit for Pre-trained Language Models. In ACL 2022.

TextPruner

- **裁剪模式**
  - **词表裁剪**：移除词表中未被使用的单词，实现减小模型体积，提升MLM任务速度的效果
  - **Transformer裁剪**：裁剪“不重要”的注意力头和全连接层神经元，在减小模型体积的同时把对模型性能的影响尽可能降到最低
  - **流水线裁剪**：依次分别进行Transformer裁剪和词表裁剪，对模型体积做全面的压缩



☞ 计算“重要性” $\mathrm{IS}(\mathbf{\Theta}) = \mathbb{E}_{x \sim X} \left| \frac{\partial \mathcal{L}(x)}{\partial \mathbf{\Theta}} \mathbf{\Theta} \right|$

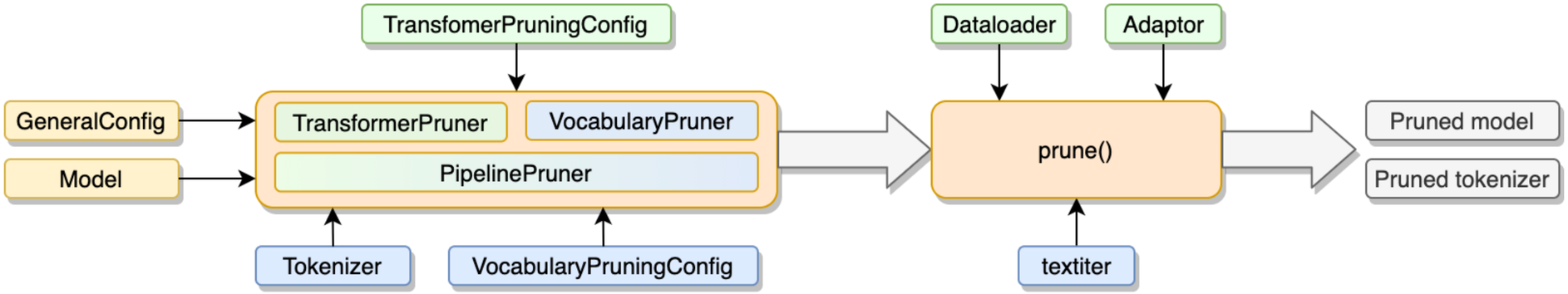
☞ 有监督裁剪 $\mathscr{L}_{CE}(x) = -\sum_{x} y \log q(x)$

☞ 无监督裁剪 $\mathscr{L}_{KL}(x) = \mathbf{KL}(\mathbf{stopgrad}(q(x)) \,||\, p(x))$

[Yang et al., 2022] TextPruner: A Model Pruning Toolkit for Pre-trained Language Models. In ACL 2022.

TextPruner

- 裁剪方法



Pruning with Python API

```
from textpruner import PipelinePruner
from textpruner import TransformerPruningConfig

config = TransformerPruningConfig(
    pruning_method='iterative', n_iters=4,
    target_ffn_size=2048,target_num_of_heads=8)
pruner = PipelinePruner(model, tokenizer,config)
pruner.prune(dataloader=dataloader, dataiter=texts)
```

▲ 通过Python API裁剪

Pruning with CLI

```
textpruner-cli \
 --pruning_mode pipeline \
 --configurations vc.json trm.json \
 --model_class BertForClassification \
 --tokenizer_class BertTokenizer \
 --model_path models \
 --vocabulary texts.txt \
 --dataloader_and_adaptor dl.py
```

▲ 通过命令行裁剪

[Yang et al., 2022] TextPruner: A Model Pruning Toolkit for Pre-trained Language Models. In ACL 2022.

TextPruner

- **模型裁剪效果：词表裁剪**
  - 实验设置
    - 实验以多语言预训练模型XLM-R在自然语言推断任务XNLI为基准进行测试
    - 使用英文数据训练，使用中文数据测试 (zero-shot)
    - 词表裁剪后，预训练模型只保留对应语种的单词
  - 实验结果表明，缩减词表后模型体积减小60%左右，基本保持了与基线系统持平的效果
  - 在指定语种的情况下，词表裁剪特别适合应用在多语言预训练模型

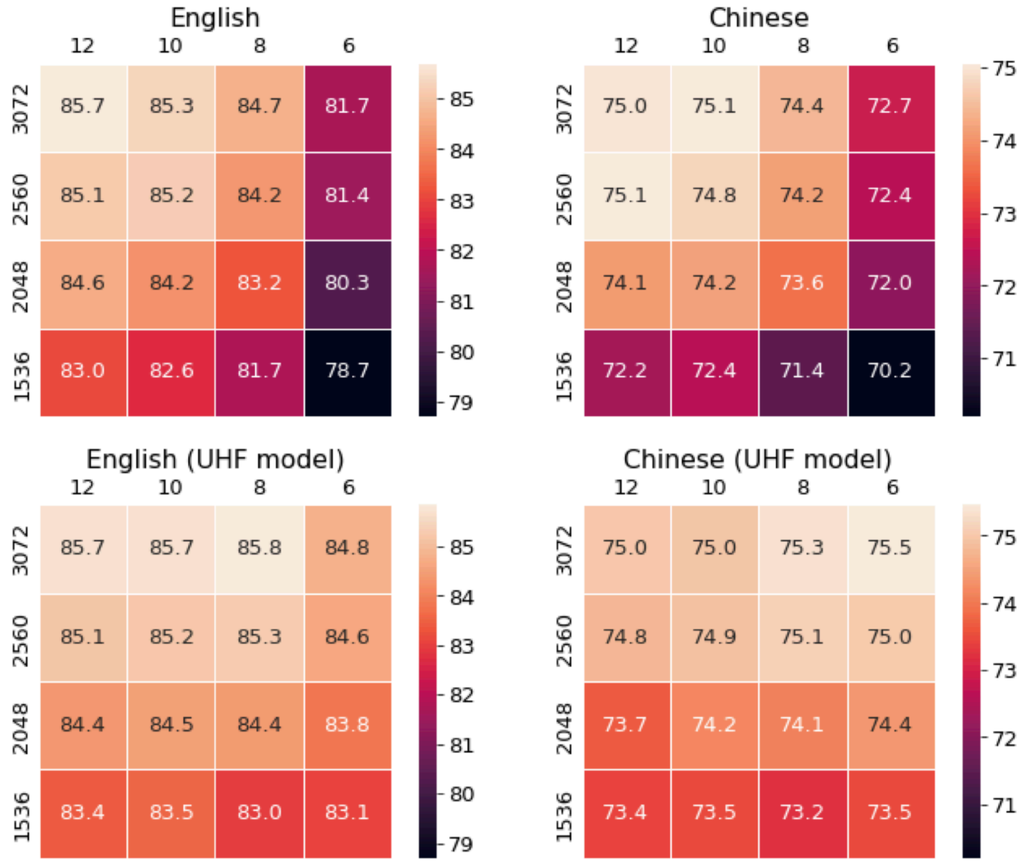
	Model	Vocabulary size	Model size	Dev (en)	Dev (zh)	Test (en)	Test (zh)
基模型	XLM-R	250002	1060 MB (100%)	84.8	75.1	85.7	75.0
裁剪模型	+ Vocabulary Pruning on en	26653	406 MB (38.3%)	84.6	-	85.9	-
	+ Vocabulary Pruning on zh	23553	397 MB (37.5%)	-	74.7	-	74.5
	+ Vocabulary Pruning on en and zh	37503	438 MB (41.3%)	84.8	74.3	85.8	74.5

▲ 词表裁剪效果

[Yang et al., 2022] TextPruner: A Model Pruning Toolkit for Pre-trained Language Models. In ACL 2022.

TextPruner

- **模型裁剪效果：Transformer裁剪**
  - 使用XNLI英文开发集计算Importance Scores
  - 两种裁剪方法
    - UHF (**U**neven attention **H**eads and **F**FN sizes)
    - HP (**H**omogenous **P**runing)
  - 观察结论
    - UHF相比HP具有更大的灵活度，能够获得更好的裁剪效果
    - 中文测试集（zero-shot）上的性能损失与英文相近
      - 对中文任务重要的神经元仍然被保留
      - 提供通用跨语言理解能力的神经元被保留

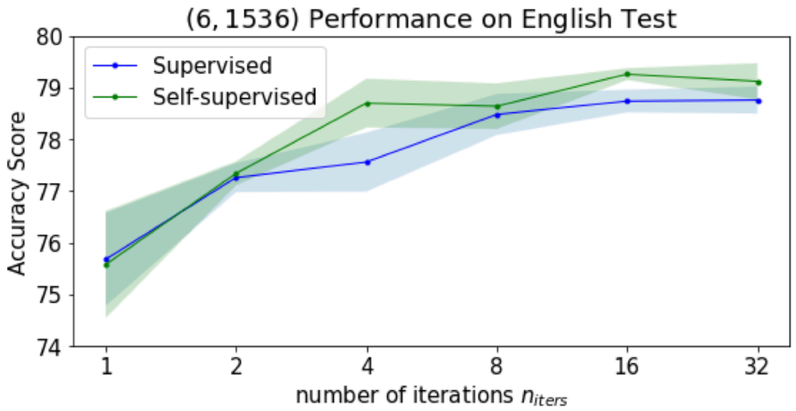


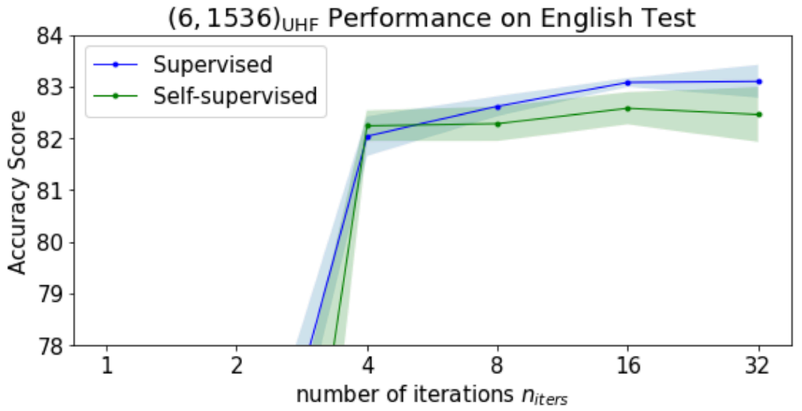
▲ XNLI测试集结果，横轴：平均注意力头数，
纵轴：FFN大小

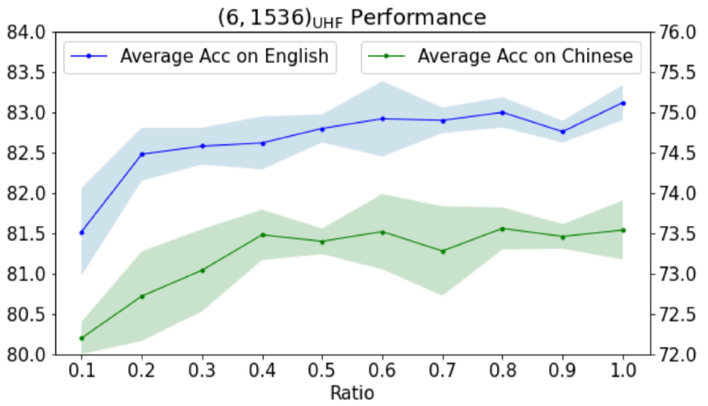
[Yang et al., 2022] TextPruner: A Model Pruning Toolkit for Pre-trained Language Models. In ACL 2022.

TextPruner

- **分析：有监督 v.s. 无监督（Fig. 1&2）**
  - 自监督方法能够在不利用标签的情况下达到与有监督方法可比甚至更优的实验结果
  - 为了保证IS计算准确，建议迭代轮数不低于8轮
- **分析：需要多少数据来计算IS？（Fig. 3）**
  - 当使用70%的开发集用于计算IS，其效果与使用全量开发集达到可比状态







[Yang et al., 2022] TextPruner: A Model Pruning Toolkit for Pre-trained Language Models. In ACL 2022.

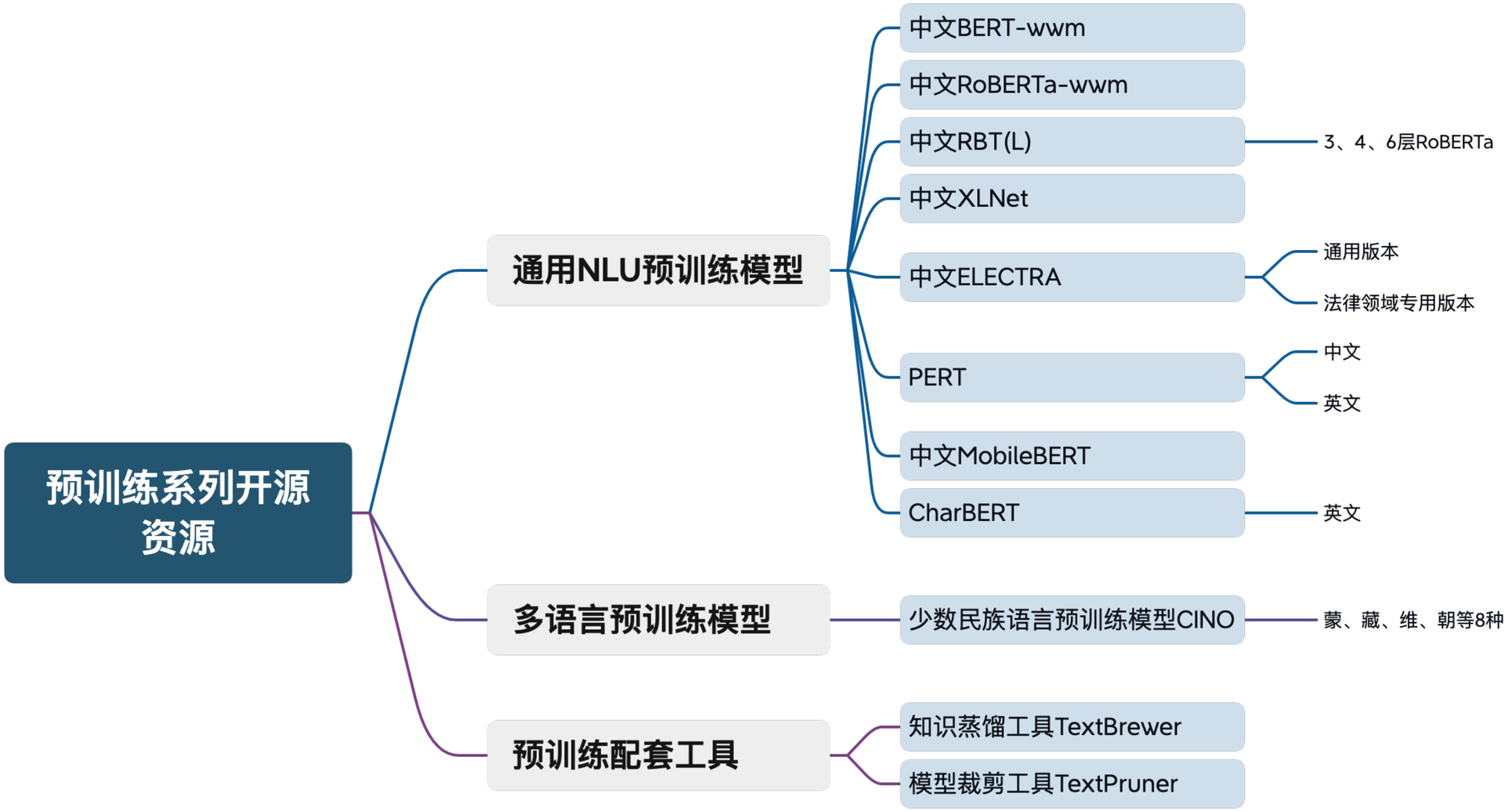
总结

SUMMARY

总结

- **基于上下文的语言模型：**过渡产物，有时有用
  - CoVe, ELMo
- **经典预训练语言模型：**经典系列，常读常新
  - GPT系列（GPT, GPT-2, GPT-3）
  - BERT系列（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、PERT
- **面向预训练语言模型的知识蒸馏与压缩：**高效推理，助推应用
  - 知识蒸馏工具TextBrewer、模型裁剪工具TextPruner

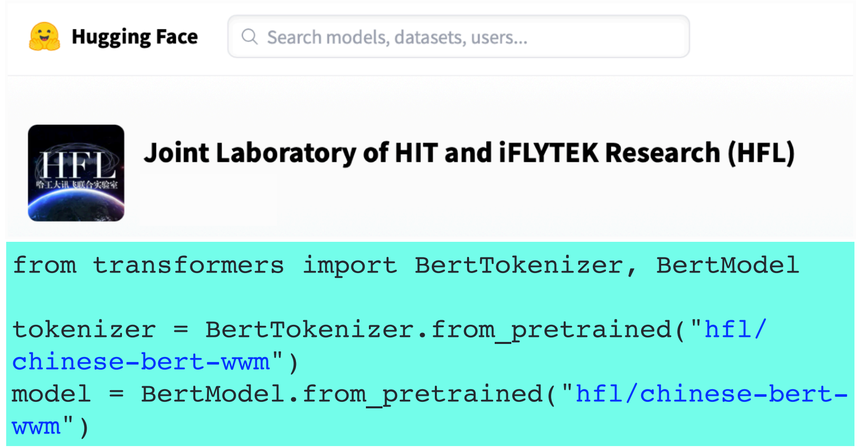
相关资源


预训练系列开源资源
通用NLU预训练模型
中文BERT-wwm
中文RoBERTa-wwm
中文RBT(L)
3、4、6层RoBERTa
中文XLNet
中文ELECTRA
通用版本
法律领域专用版本
PERT
中文
英文
中文MobileBERT
CharBERT
英文
多语言预训练模型
少数民族语言预训练模型CINO
蒙、藏、维、朝等8种
预训练配套工具
知识蒸馏工具TextBrewer
模型裁剪工具TextPruner

相关资源

- **GitHub**
  - 中英文预训练模型：BERT-wwm, XLNet, ELECTRA, MobileBERT, MacBERT, PERT, CINO等
  - 工具包：TextBrewer, TextPruner等
  - 数据集：阅读理解、文本分类、文本纠错等
  - HFL典藏集：https://github.com/ymcui/HFL-Anthology
- **Model Hub**
  - 搭配transformers库，快速加载各类预训练模型
  - 目前已开放44个预训练语言模型
  - 地址：https://huggingface.co/HFL

哈工大讯飞联合实验室
微信公众号



THANK YOU!

https://github.com/ymcui

https://ymcui.com

me@ymcui.com

讲义下载

哈工大讯飞联合实验室

科大讯飞 iFLYTEK