预训练语言模型研究进展

刘群

语音语义首席科学家
华为诺亚方舟实验室

2020-09-06
CCF-NLP走进华南理工大学

HUAWEI

目录

目录

自然语言的表示学习

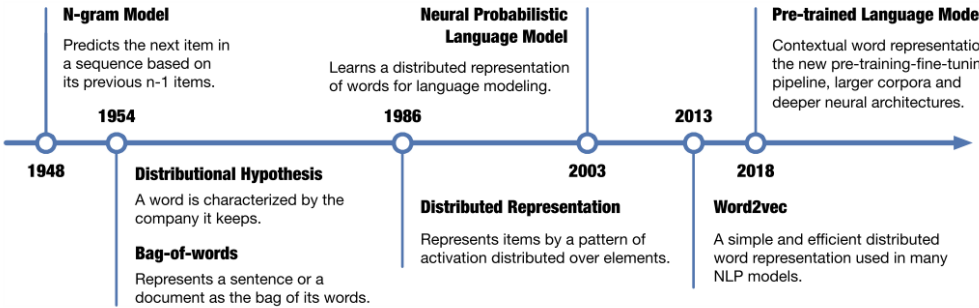
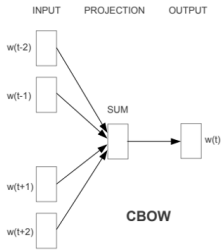


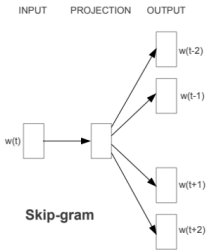
Fig. 1.3 The timeline for the development of representation learning in NLP. With the growing computing power and large-scale text data, distributed representation trained with neural networks and large corpora has become the mainstream

Liu et al., Representation Learning for Natural Language Processing, Springer, 2020

第一代自然语言预训练模型：词向量模型

- 典型代表：CBOW, Skip-gram, Glove, Fasttext
- 词向量表示是固定，不会随着上下文的改变而变化





CBOW: predicts the current word based on the context

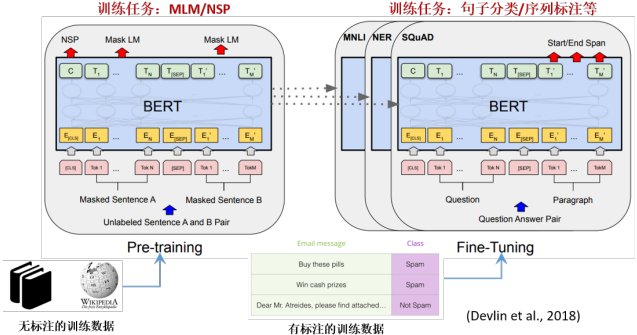
Skip-gram: predicts surrounding words given the current word

(Mikolov et al., 2013)

语义越相近的词在空间中越接近

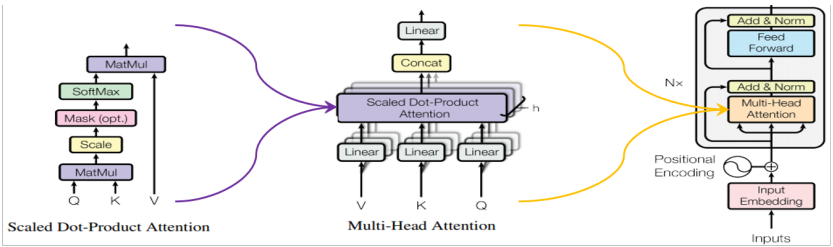
第二代自然语言预训练模型：预训练语言模型

- 典型代表：ELMo, BERT, GPT
- Pre-training-then-fine-tuning已经成为NLP研究新范式
- 将在pre-training阶段学习到的语言表示迁移到下游任务



第二代自然语言预训练模型：预训练语言模型

- 核心技术：Transformer模型



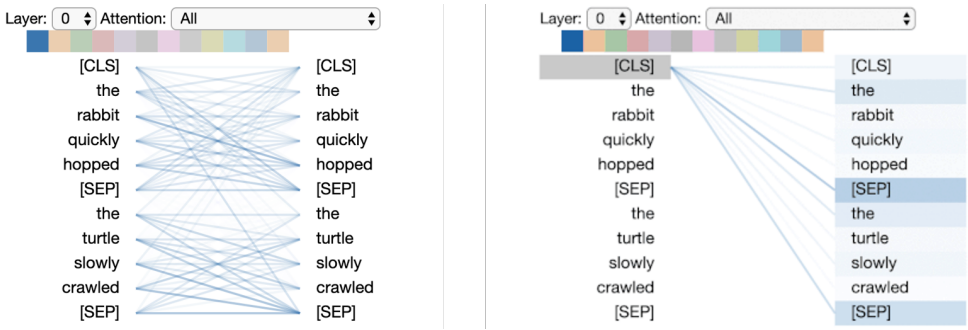
$$\text{MultiHead}(Q,K,V) = \text{Concat}(\text{head}_1,...,\text{head}_\text{h})W^O$$
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

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

(Vaswani et al., 2017)

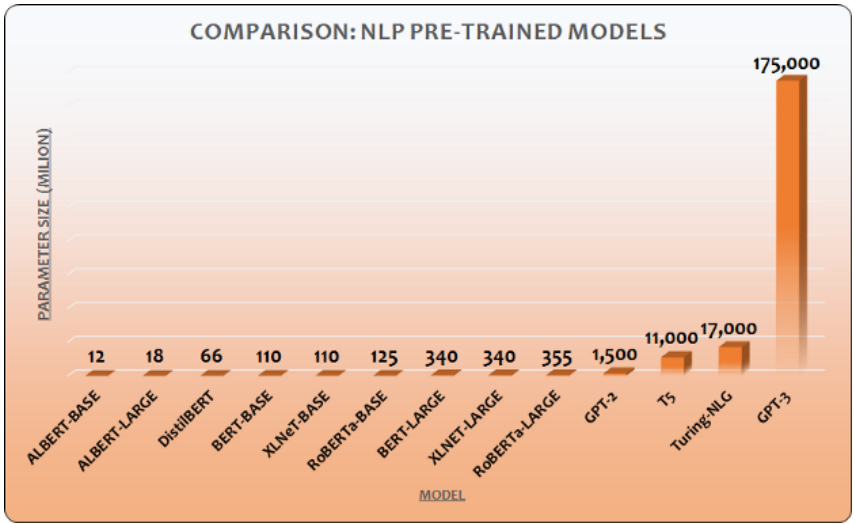
第二代自然语言预训练模型：预训练语言模型

- 核心技术：自注意力机制（self-attention）
  - 每个token是通过所有词动态加权得到
  - 动态权重会随着输入的改变而变化



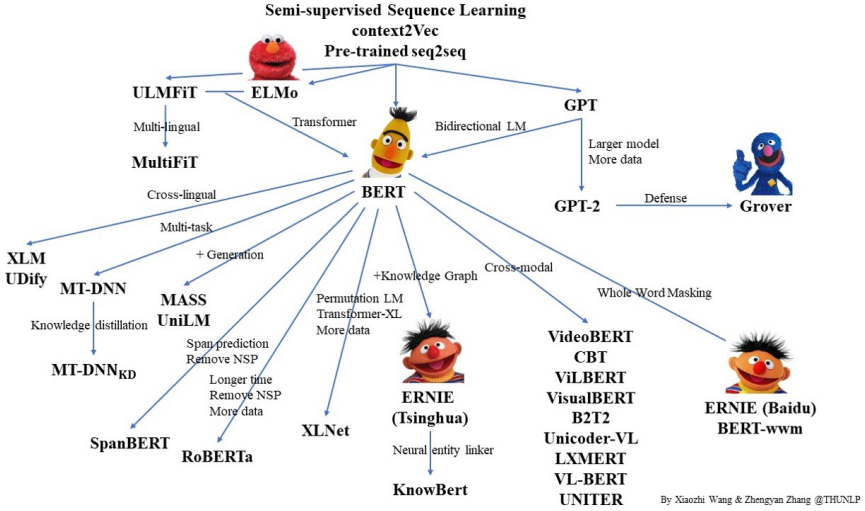
(BertViz tool, Vig et al., 2019)

预训练语言模型发展



https://medium.com/analytics-vidhya/openai-gpt-3-language-models-are-few-shot-learners-82531b3d3122

预训练语言模型家族



https://github.com/thunlp/PLMpapers

目录

诺亚方舟实验室的预训练语言模型研究

- 开发自研预训练语言模型：哪吒（NEZHA）
- 预训练语言模型小型化和加速（TinyBERT、DynaBERYT、TernaryBERT）
- 多语言预训练语言模型（MultiNEZHA）
- 预训练语言模型的解释（Perturbed Masking BERT for Unsupervised Parsing）
- 预训练语言模型用于任意词序生成（Probabilistically Masked Language Model）
- 融合知识的预训练语言模型（ERNIE-Tsinghua，BERT-MK）
- 预训练语言模型的底层移植和优化（to Da Vinci NPU and MindSpore）
- 预训练语言模型落地应用
  - 华为多语言语音助手
  - 华为应用市场、华为浏览器（推荐场合）
  - 乐府作诗机

哪吒预训练语言模型

NEZHA: NEURAL CONTEXTUALIZED REPRESENTATION FOR CHINESE LANGUAGE UNDERSTANDING

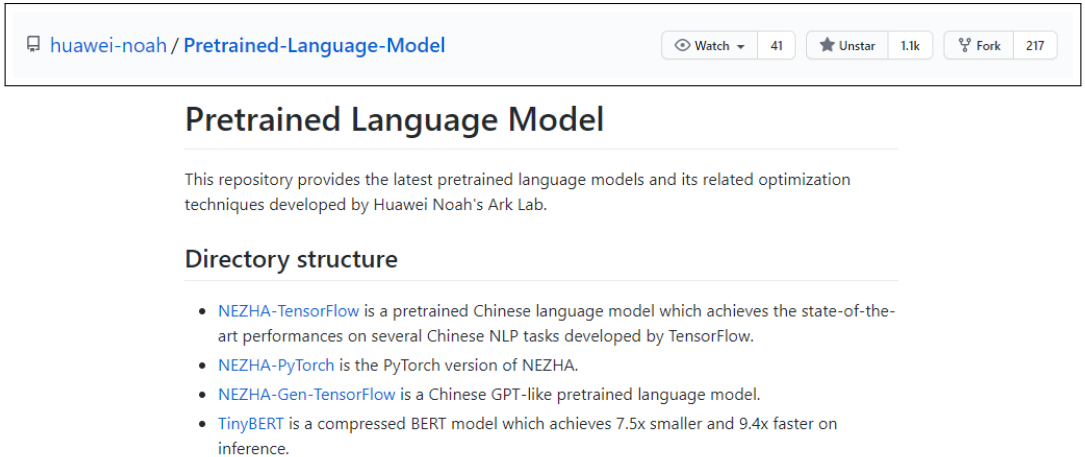
TECHNICAL REPORT

Junqiu Wei, Xiaozhe Ren, Xiaoguang Li, Wenyong Huang, Yi Liao, Yasheng Wang, Jiashu Lin*, Xin Jiang, Xiao Chen, Qun Liu
Noah's Ark Lab, *HiSilicon, Huawei Technologies
{wei.junqiu1, renxiaozhe, lixiaoguang11, wenyong.huang, liao.yi, wangyasheng, linjiashu, jiang.xin, chen.xiao2, qun.liu}@huawei.com

September 4, 2019

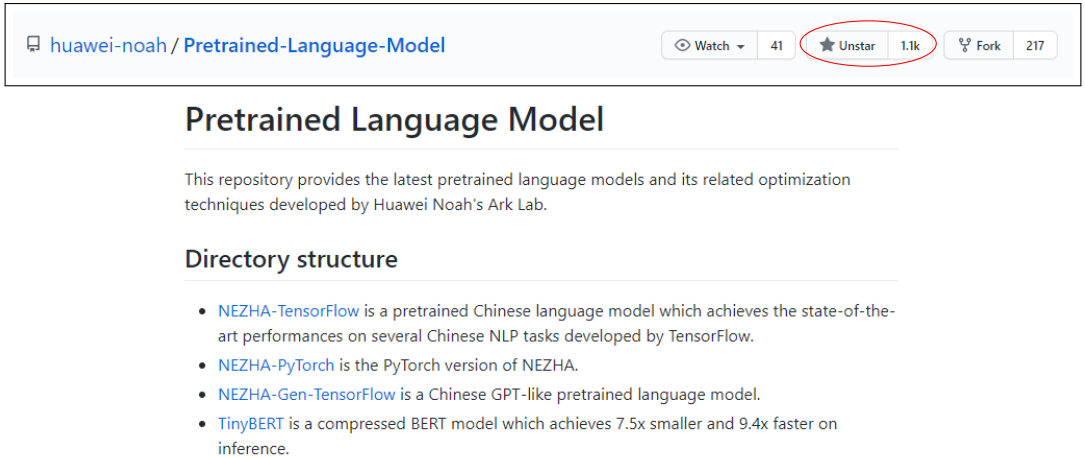
- 技术报告：https://arxiv.org/abs/1909.00204

哪吒的开源



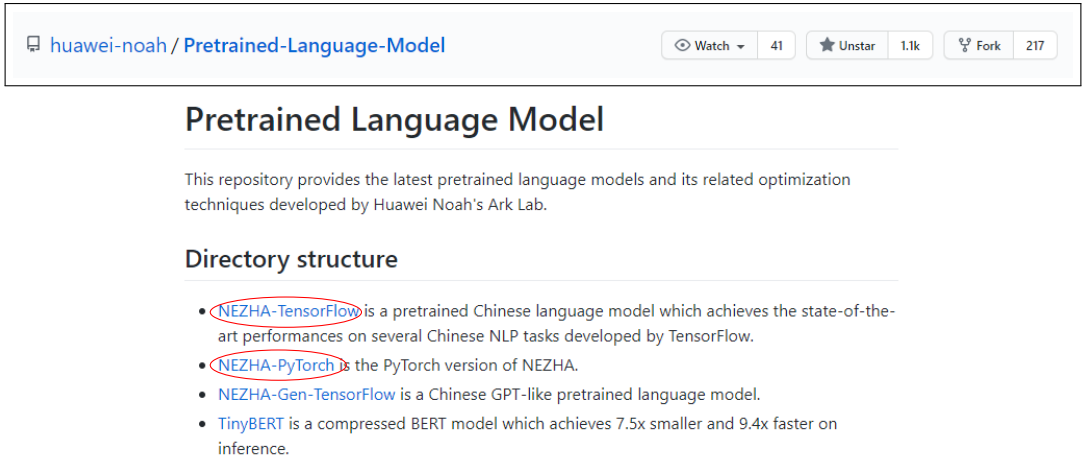
- 开源代码：https://github.com/huawei-noah/Pretrained-Language-Model

哪吒的开源



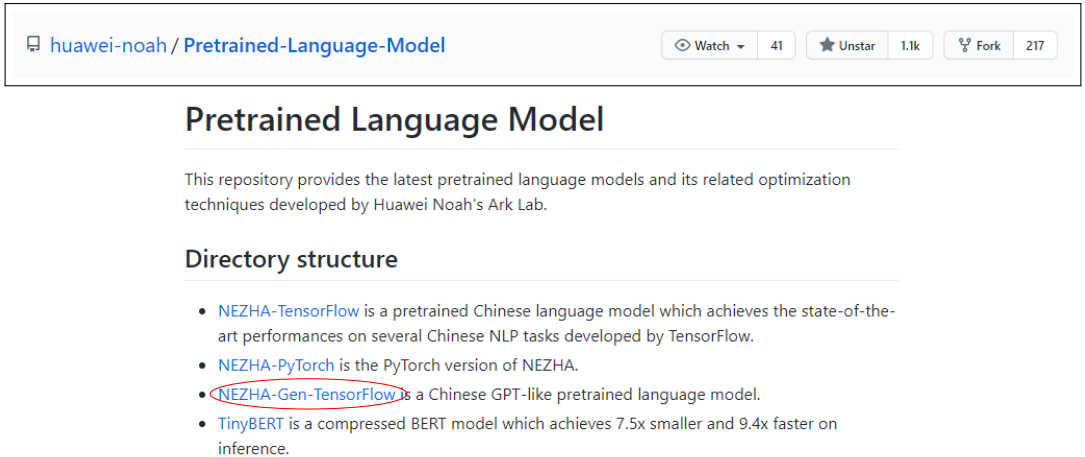
- 开源代码：https://github.com/huawei-noah/Pretrained-Language-Model

哪吒的开源



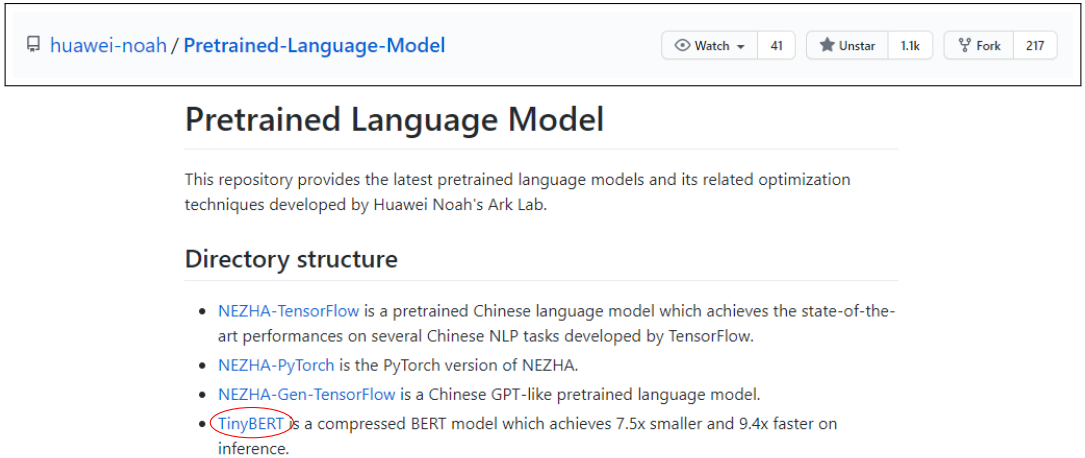
- 开源代码：https://github.com/huawei-noah/Pretrained-Language-Model

哪吒的开源



- 开源代码：https://github.com/huawei-noah/Pretrained-Language-Model

哪吒的开源



- 开源代码：https://github.com/huawei-noah/Pretrained-Language-Model

哪吒的技术特色

与BERT模型共同点

多层Transformer编码器

相同的预训练语言任务

相近的模型大小与参数量

与BERT模型不同点

相对位置编码

全词掩码

混合精度训练

LAMB优化器

支持GPT自回归模型

概率掩码

其他

训练语料：wiki，百度百科，新闻

训练资源：8 V100 * 10

哪吒的排名

- 中文CLUE：第一

CLUE总排行榜*记最佳得分

排行	模型	研究机构	测评时间	Score	认证	AFQMC	TNEWS	IFLYTEK	CMNLI	CLUEWSC 2020	CSL	CMRC2018	CHID	C3
1	HUMAN	CLUE	19-12-01	85.089	已认证	81.000	71.000	80.300	76.000	98.000	84.000	92.400	87.100	96.000
2	NEZHA-large	Huawei Noah's Ark ...	20-05-27	78.274	待认证	76.094	68.360	61.615	81.460	89.310	84.033	77.900	86.535	79.162
3	UER	Tencent Oteam	20-04-30	77.745	待认证	76.819	71.260	63.231	83.471	82.759	84.633	79.150	85.493	72.893
4	RoBERTa-wwm-large	CLUE	19-12-01	66.614	已认证	76.550	58.610	62.980	82.120	0.0	82.130	77.950	85.370	73.820
5	ALBERT-xxlarge	CLUE	19-12-01	66.256	已认证	75.600	59.460	62.890	83.140	0.0	83.630	75.150	83.150	73.280
6	NEZHA-base	Huawei Noah's Ark ...	20-03-10	66.060	待认证	74.488	67.380	59.538	81.088	0.0	83.733	72.150	84.377	71.788
7	RoBERTa-large	CLUE	19-12-01	65.3				62.550	81.700	0.0	81.360	78.500	84.500	67.550
8	RoBERTa-wwm-ext	CLUE	19-12-01	64.236	已认证	74.040	56.940	60.310	80.510	0.0	81.000	75.200	83.620	66.500

获取排行榜数据成功!

哪吒的排名

- 英文GLUE：第七

	Rank	Name	Model	URL	Score	CoLA	SST-2	MRPC	STS-B	QQP	MNLI-m	MNLI-mm	QNLI	RTE	WNLI	AX
+	1	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
	2	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
+	3	Alibaba DAMO NLP	StructBERT		90.3	75.3	97.1	93.9/91.9	93.0/92.5	74.8/91.0	90.9	90.7	96.4	90.2	94.5	49.1
	4	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
	5	Microsoft D365 AI & MSR AI & GATECHMT-DNN-SMART			89.9	69.5	97.5	93.7/91.6	92.9/92.5	73.9/90.2	91.0	90.8	99.2	89.7	94.5	50.2
+	6	ELECTRA Team	ELECTRA-Large + Standard Tricks		89.4	71.7	97.1	93.1/90.7	92.9/92.5	75.6/90.8	91.3	90.8	95.8	89.8	91.8	50.7
+	7	Huawei Noah's Ark Lab	NEZHA-Large		88.7	67.4	97.2	93.2/91.0	92.2/91.6	74.1/90.2	90.8	90.2	95.7	88.5	93.2	45.0
+	8	Microsoft D365 AI & UMD	FreeLB-RoBERTa (ensemble)		88.4	68.0	96.8	93.1/90.8	92.3/92.1	74.8/90.3	91.1	90.7	95.6	88.7	89.0	50.1
	9	Junjie Yang	HIRE-RoBERTa		88.3	68.6	97.1	93.0/90.7	92.4/92.0	74.3/90.2	90.7	90.4	95.5	87.9	89.0	49.3
	10	Facebook AI	RoBERTa		88.1	67.8	96.7	92.3/89.8	92.2/91.9	74.3/90.2	90.8	90.2	95.4	88.2	89.0	48.7

哪吒的排名

- 英文SUPERGLUE：第五

Leaderboard Version: **2.0**

	Rank	Name	Model	URL	Score	BoolQ	CB	COPA	MultiRC	ReCoRD	RTE	WiC	WSC	AX-b	AX-g
	1	SuperGLUE Human Baselines	SuperGLUE Human Baselines	SuperGLUE Human Baselines			95.8/98.9	100.0	81.8/51.9	91.7/91.3	93.6	80.0	100.0	76.6	99.3/99.7
✚	2	T5 Team - Google	T5		89.3	91.2	93.9/96.8	94.8	88.1/63.3	94.1/93.4	92.5	76.9	93.8	65.6	92.7/91.9
	3	Zhuiyi Technology	RoBERTa-mtl-adv		85.7	87.1	92.4/95.6	91.2	85.1/54.3	91.7/91.3	88.1	72.1	91.8	58.5	91.0/78.1
	4	Facebook AI	RoBERTa		84.6	87.1	90.5/95.2	90.6	84.4/52.5	90.6/90.0	88.2	69.9	89.0	57.9	91.0/78.1
✚	5	Huawei Noah's Ark Lab	NEZHA-Large		83.8	85.8	93.3/95.6	91.2	78.7/42.4	87.1/86.4	88.5	73.1	90.4	58.0	87.1/74.4
✚	6	Infosys : DAWN : AI Research	RoBERTa-iCETS		77.4	84.7	88.2/91.6	85.8	78.4/37.5	82.9/82.4	83.8	69.1	65.1	35.2	93.8/68.8
	7	IBM Research AI	BERT-mtl		73.5	84.8	89.6/94.0	73.8	73.2/30.5	74.6/74.0	84.1	66.2	61.0	29.6	97.8/57.3
	8	Ben Mann	GPT-3 few-shot - OpenAI		71.8	76.4	52.0/75.6	92.0	75.4/30.5	91.1/90.2	69.0	49.4	80.1	21.1	90.4/55.3
	9	SuperGLUE Baselines	BERT++		71.5	79.0	84.8/90.4	73.8	70.0/24.1	72.0/71.3	79.0	69.6	64.4	38.0	99.4/51.4

目录

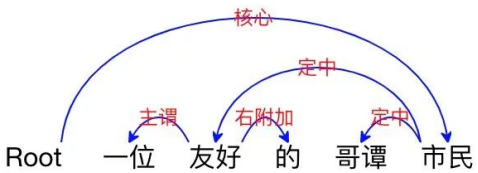
预训练语言模型扰动掩码: Perturbed Masking

Perturbed Masking: Parameter-free Probing for Analyzing and Interpreting BERT

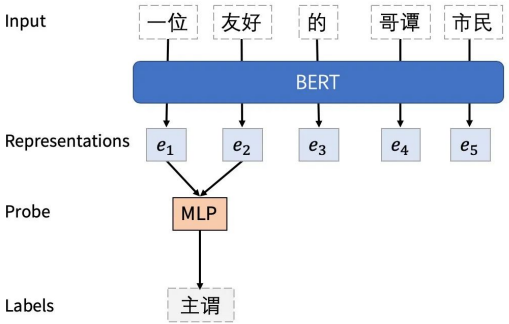
Zhiyong Wu[1], **Yun Chen**[2], **Ben Kao**[1], **Qun Liu**[3]
[1]The University of Hong Kong, Hong Kong, China
[2]Shanghai University of Finance and Economics, Shanghai, China
[3]Huawei Noah's Ark Lab, Hong Kong, China
{zywu,kao}@cs.hku.hk, yunchen@sufe.edu.cn, qun.liu@huawei.com

https://www.aclweb.org/anthology/2020.acl-main.383/

问题：预训练语言模型学到了多少句法知识？



依存句法树



探针测试（Probing Test）

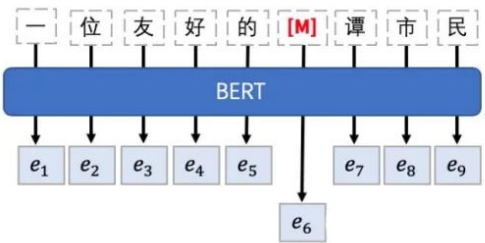
- 探针测试虽然可以达到很高的准确率，但并不能有效回答上述问题(John Hewitt et al., 2019)
- 我们的工作：直接用BERT做无监督句法分析，看看能否从BERT的表示中推导出合理的句法结构

扰动掩码(Perturbed Masking)

- 借鉴依存句法分析的常见做法，我们先利用BERT建立一个词语之间的影响力矩阵（impact matrix），然后再利用这个矩阵推导出依存关系树；
- 利用BERT探测一个词对另一个词的影响，我们提出了“扰动掩码（Perturbed Masing）”的方法：

扰动掩码(Perturbed Masking)

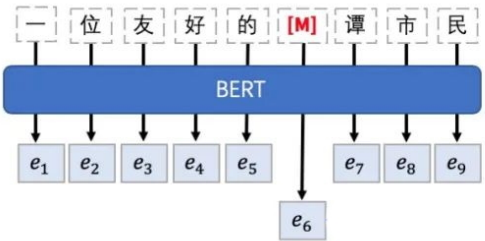
- 借鉴依存句法分析的常见做法，我们先利用BERT建立一个词语之间的影响力矩阵（impact matrix），然后再利用这个矩阵推导出依存关系树；
- 利用BERT探测一个词对另一个词的影响，我们提出了“扰动掩码（Perturbed Masing）”的方法：



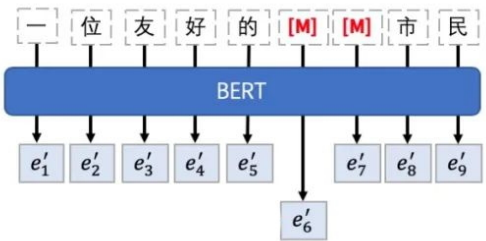
Masked Word Prediction

扰动掩码(Perturbed Masking)

- 借鉴依存句法分析的常见做法，我们先利用BERT建立一个词语之间的影响力矩阵（impact matrix），然后再利用这个矩阵推导出依存关系树；
- 利用BERT探测一个词对另一个词的影响，我们提出了“扰动掩码（Perturbed Masing）”的方法：



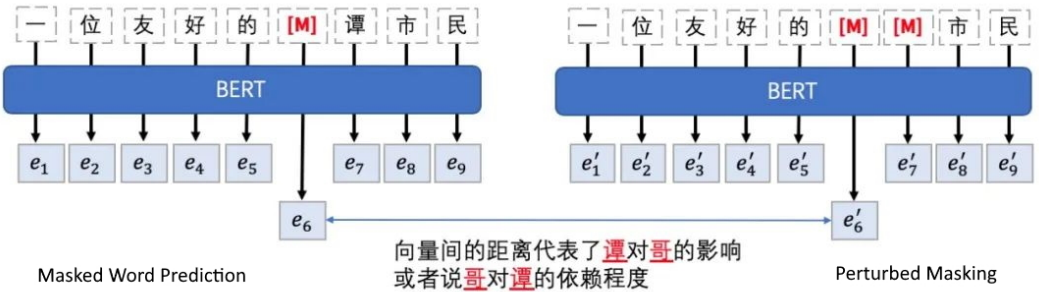
Masked Word Prediction



Perturbed Masking

扰动掩码(Perturbed Masking)

- 借鉴依存句法分析的常见做法，我们先利用BERT建立一个词语之间的影响力矩阵（impact matrix），然后再利用这个矩阵推导出依存关系树；
- 利用BERT探测一个词对另一个词的影响，我们提出了“扰动掩码（Perturbed Masing）”的方法：



影响力矩阵（impact matrix）

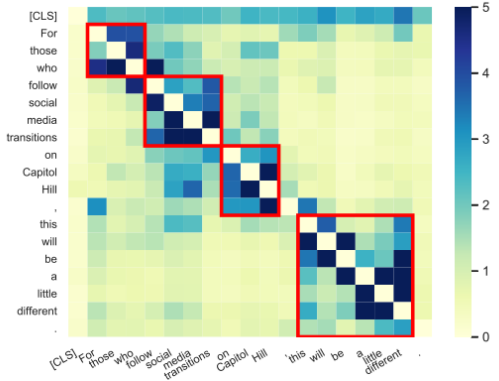


Figure 1: Heatmap of the impact matrix for the sentence “For those who follow social media transitions on Capitol Hill, this will be a little different.”

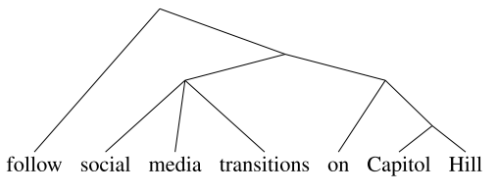


Figure 2: Part of the constituency tree.

依存句法分析

- 投射式依存分析:

  Eisner算法(1996)
- 非投射依存分析:

  Chu-Liu/Edmonds(CLE)算法(1965; 1967)
- 系统：
  - Left-chain: 每个词依存于前一个词
  - Right-chain: 每个词依存于后一个词
  - Random-BERT: 随机参数化BERT，采用我们的方法做句法分析
  - *-Dist: 我们的方法，采用该词输出的logit的欧式距离计算影响力
  - *-Prob: 我们的方法，采用该词输出的softmax概率之差计算影响力

Model	Parsing UAS	
	WSJ10-U	PUD
Right-chain	49.5	35.0
Left-chain	20.6	10.7
Random BERT	16.9	10.2
Eisner+Dist	**58.6**	**41.7**
Eisner+Prob	52.7	34.1
CLE+Dist	51.5	33.2

Table 1: UAS results of BERT on unsupervised dependency parsing.

Model	UAS	UUAS	NED
Eisner+Dist	41.7	52.1	69.6
Right-chain	35.0	39.9	41.2

Table 2: Performance on PUD when evaluated using UAS, UUAS, and NED.

成分句法分析

Model	Parsing F1		Accuracy on PTB23 by Tag				
	WSJ10	PTB23	NP	VP	PP	S	SBAR
PRPN-LM	70.5	37.4	63.9	-	24.4	-	-
ON-LSTM 1st-layer	42.8	24.0	23.8	15.6	18.3	48.1	16.3
ON-LSTM 2nd-layer	66.8	49.4	61.4	51.9	55.4	54.2	15.4
ON-LSTM 3rd-layer	57.6	40.4	57.5	13.5	47.2	48.6	10.4
300D ST-Gumbel w/o Leaf GRU	-	25.0	18.8	-	9.9	-	-
300D RL-SPINN w/o Leaf GRU	-	13.2	24.1	-	14.2	-	-
MART	58.0	42.1	44.6	47.0	50.6	66.1	51.9
Right-Branching	56.7	39.8	25.0	71.8	42.4	74.2	68.8
Left-Branching	19.6	9.0	11.3	0.8	5.0	44.1	5.5

Table 3: Unlabeled parsing F1 results evaluated on WSJ10 and PTB23.

我们提出的无监督成分句法分析算法MART：

- 根据影响力矩阵，自顶向下，每一步寻找当前span最薄弱点切开。

篇章分析

Model	UAS	Accuracy by distance			
		0	1	2	5
Right-chain	10.7	20.5	-	-	-
Left-chain	**41.5**	**79.5**	-	-	-
Random BERT	6.3	20.4	7.5	3.5	0.0
Eisner+Dist	34.2	61.6	7.3	**7.6**	**12.8**
CLE+Dist	34.4	63.8	3.3	3.5	2.6

Table 4: Performance of different discourse parser. The distance is defined as the number of EDUs between head and dependent.

无监督句法分析结果分析

- 无论是依存句法，还是成分句法，或者篇章分析，基于BERT扰动掩码的方法均取得了较好的结果；
- 比随机模型，或者单纯左/右分叉（依存）的方法均有显著提高；
- 我们的方法与无监督方法句法分析的SOTA还有一定差距，原因在于：
  - 现有SOTA无监督句法分析方法都引入了较多的语言学先验知识（比如左分叉bias），而我们没有引入任何先验知识。
  - 有些无监督句法分析方法利用了POS等外部特征，而我们完全没有采用。
  - 依存句法结构的定义主观性较强，与无监督方法学到的结构可能有较大差异，而我们的系统没有没有针对这种结构定义做任何针对性调试。
- 无监督篇章分析，采用左分叉方法已经可以取得较好结果，我们的方法没有引入左分叉bias，比左分叉方法略低。

无监督句法分析结果用于下游任务

- 问题：虽然无监督句法分析得到的句法结构与人类语言学家定义的句法结构有较大差异，这种句法结构是否可以用在下游NLP任务中，以改善下游任务的性能呢？
- 我们以Aspected-based Sentiment Analysis（ABSC）为例，采用Zhang et al.(2019)提出的Proximity-Weighted Convolution Network (PWCN)方法，分别引入各种不同的语言学特征，进行了对比实验：

Model	Laptop		Restaurant	
	Acc	Macro-F1	Acc	Macro-F1
LSTM	69.63	63.51	77.99	66.91
PWCN				
+Pos	75.23	71.71	81.12	71.81
+Dep	76.08	72.02	80.98	72.28
+Eisner	75.99	72.01	**81.21**	**73.00**
+right-chain	75.64	71.53	81.07	72.51
+left-chain	74.39	70.78	80.82	72.71

Table 5: Experimental results of aspect based sentiment classification.

实验表明，引入我们的无监督句法分析得到的依存结构，比引入其他语言学信息的结果都要好，包括引入人类语言学家定义的依存特征信息（左图中+Dep）。

目录

概率掩码的预训练语言模型PMLM

Probabilistically Masked Language Model Capable of Autoregressive Generation in Arbitrary Word Order

Yi Liao, Xin Jiang, Qun Liu
Huawei Noah's Ark Lab
{liaoyi9, jiang.xin, qun.liu}@huawei.com

https://www.aclweb.org/anthology/2020.acl-main.24/

概率掩码的预训练语言模型PMLM

- 基本思想：取代BERT训练中预定比例（如15%）掩码的思想，我们对源语言句子中每一个词按一个预定义的概率分布进行解码
- 模型特点：
  - 可以像BERT一样用于NLU任务，效果甚至略好于BERT和XLNET
  - 可以像BERT一样用于NLG任务，效果略逊于GPT，
  - 比GPT更强大的生成能力：可以按照任意词序生成文本
  - 训练代价高于BERT、XLNET和GPT

按照任意词序生成文本

The wolf has an extraordinary speed , and it can often jump from a spot **quick** enough to escape a spot already occupied by an adult wolf . Unlike the **brown** and black bear , where it is easily distracted by wolves , the gray **fox** does not run over a wolf , and is often driven mad . Having **jumps** with high speed that breaks the wolf ' s legs before it is run **over** , a grey wolf could defend itself against an adult of other species as **the** best predator at any time . The black bear may kill packs of four **lazy** , though the gray fox can inflict significant wounds on a **dog** .

应用场合：

- 藏头诗、藏尾诗
- 辅助诗歌生成
- 词汇约束解码
- 加密

PMLM: 模型定义

- Assume:
  - N is the sequence length
  - K is the number of masked tokens
  - $X = \{x_1, x_2, ..., x_N\}$ is a sequence of tokens
  - $M = \{m_1, m_2, ..., m_N\}$ denotes the sequence of binary variables indicating the masks
  - $\Pi = \pi_1, \pi_2, ..., \pi_K$ denotes the indexes of masked tokens
- Then the log-likelihood function of PMLM for observing X_Π is:

$$L_{pmlm} = \mathbb{E}_{X,M}\left[\log p(X_\Pi | X_{-\Pi})\right] = \sum_X \sum_M \left[\log p(X_\Pi | X_{-\Pi})\right] p(M)$$

概率掩码

- M是定义在X之上的任何一个掩码，但与X的内容无关，只与X中token的位置有关
- 假设M服从某种概率分布$p(M)$
  - 在PMLM中$p(M)$可以任意定义
  - 为简化起见，我们可以假设X中每个token被mask的概率r是独立的：
    - BERT中假设每个token出现的概率是固定的：$p(r) = 15\%$
    - 我们考虑一种比BERT更一般化的模型μ-PMLM：假设$p(r)$在[0,1]区间上均匀分布
- μ-PMLM的概率掩码生成算法：
  - 对于X中的每一个词x_i:
    1. 依据均匀分布在[0,1]随机选取一个概率r_i；
    2. 根据概率r_i随机决定m_i是否替换为[MASK]。
- 除了掩码生成部分，μ-PMLM的实现与BERT完全相同。

在自然语言理解（NLU）任务上的实验结果

Model	COLA	SST2	MRPC	STSB	QQP	MNLI-m/mm	QNLI	RTE	AVG.
BERT(A)	52.1	93.5	88.9/84.8	87.1/85.8	71.2/89.2	84.6/83.4	90.5	66.4	78.3
u-PMLM-A	56.5	94.3	88.8/84.4	87.0/85.9	71.4/89.2	84.5/83.5	91.8	66.1	79.0
u-PMLM-R	58.0	94.0	89.7/85.8	87.7/86.8	71.2/89.2	85.0/84.1	92.3	69.8	80.0
u-PMLM-R*	56.9	94.2	90.7/87.7	89.7/89.1	72.2/89.4	86.1/85.4	92.1	78.5	81.3

Table 5: Evaluation on GLUE test set.

- μ-PMLM在GLUE总成绩和SQuaD任务上都超过了BERT-base
- 相对位置编码可以进一步改进μ-PMLM的性能
- μ-PMLM在MNLI和SQuaD等主要任务上超过了XLNET
- 注：
  - μ-PMLM-A：采用绝对位置编码的μ-PMLM
  - μ-PMLM-R：采用相对位置编码的μ-PMLM
  - μ-PMLM-R*:在μ-PMLM-R加多任务训练

Model	F1	EM
BERT(A)	76.85	73.97
u-PMLM-A	78.31	74.62
u-PMLM-R	81.52	78.46

Table 6: Evaluation on SQUAD 2.0.

Model	SQUAD 2.0 F1/EM	MNLI m/mm	SST2
XLNet (R)	81.33/78.46	85.84/85.43	92.66
u-PMLM-R	81.52/78.46	85.99/85.60	93.58

Table 7: Comparison with XLNet.

在自然语言生成（NLG）任务上的实验结果

Model	PPL(sequential)	PPL(random)
BERT	23.12	25.54
GPT	21.23	N/A
u-PMLM-R	19.58	21.51
u-PMLM-A	19.32	21.30

Table 2: Perplexity on Wikitext103.

Model	PPL(sequential)	PPL(random)
BERT	140.67	56.97
GPT	24.25	N/A
u-PMLM-R	35.24	38.45
u-PMLM-A	49.32	42.46

Table 3: Perplexity on One-Billion Words.

- μ-PMLM在生成任务上表现远好于BERT，无论是顺序生成还是乱序生成
- μ-PMLM跟GPT相比：
  - μ-PMLM可以乱序生成，GPT不能
  - 在小训练数据上，μ-PMLM的生成性能好于GPT
  - 在大训练数据上，μ-PMLM的生成性能跟GPT相比还有一定差距

μ-PMLM乱序生成示例

Step	Prediction Index	State of the sequence							
0	n/a	_	_	_	_	_	_	_	_
1	3	_	_	a	_	_	_	_	_
2	7	_	_	a	_	_	_	random	_
3	1	This	_	a	_	_	_	random	_
4	2	This	is	a	_	_	_	random	_
5	4	This	is	a	sentence	_	_	random	_
6	6	This	is	a	sentence	_	in	random	_
7	5	This	is	a	sentence	generated	in	random	_
8	8	This	is	a	sentence	generated	in	random	order
Generation Order: 3→7→1→2→4→6→5→8									
Output: This is a sentence generated in random order									

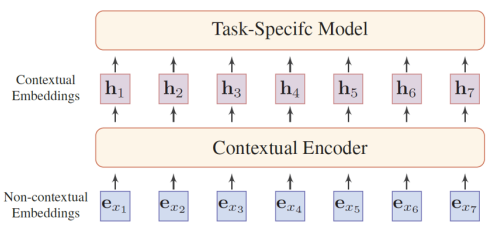
Table 1: An example of how u-PMLM generates a sequence in random order. The special token `[MASK]` is simplified as the symbol "_".

目录

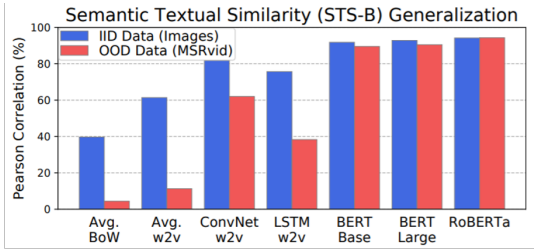
目录

预训练模型已经成为各种NLP任务的基石

- 各种预训练模型被各大公司竞相提出
- 先做大阶段:“大算力+大模型+大数据+创意任务”探索能力边界
- 再做小阶段: 在各种下游任务上形成生产力（对话/阅读理解/搜索等）



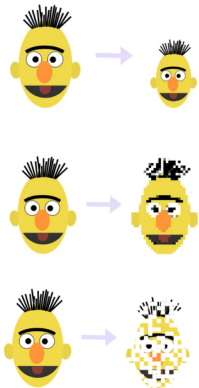
(Qiu et al., 2020)



(Hendrycks et al., 2020)

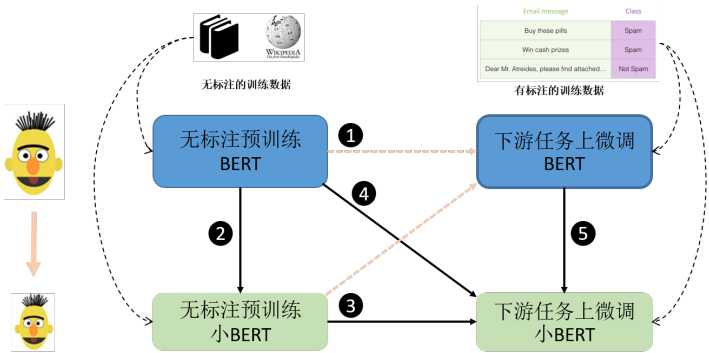
预训练模型压缩

- 蒸馏(Knowledge Distillation)
  - 用一个小模型来模拟大模型
  - DistilBERT/BERT-PKD/TinyBERT
  - MobileBERT/MiniLM (Task agnostic)
- 量化(Quantization)
  - 用低bit来表示权重和激活函数
  - Q-BERT/Q8BERT
  - TernaryBERT
- 剪枝/可伸缩 (Pruning/Slimmable)
  - 将一些不重要的head/层/神经元去掉
  - LayerDrop/DynaBERT
- 其他：参数共享/矩阵分解等
- 华为自研 TinyBERT 背后的模型压缩技术实践



目录

预训练模型蒸馏–知识迁移视角



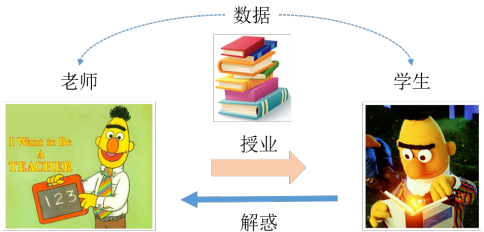
迁移1：基于无标注预训练的BERT到基于下游任务微调的BERT

迁移2+3：通过两步，将在无标注语料学到的知识迁移到小模型

迁移4：通过一步，将无标注语料学到的知识迁移到小模型

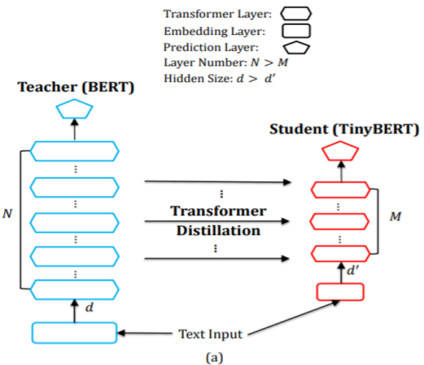
迁移5：将下游任务上的老师迁移到小模型

预训练模型蒸馏–面临的问题



- 好的老师模型：包含的知识应该更容易迁移到小模型，（比如：MobileBERT专门学习了瘦高的老师）
- 好的学生模型：应该稀疏、低比特、参数少，同时可以包含更多的知识；（比如：AdaBERT通过AutoML搜索学生结构）
- 好的学习方法：
  1. 定义更好的知识表示函数和损失函数；
  2. 通过各种策略学习怎么教给小模型，比如：逐步地将知识迁移到学生不同的层，找一个助教解决超大老师的知识不易于迁移的难题；
  3. 通过更加自动化的方法扩充语料，将更多的知识教给小模型。

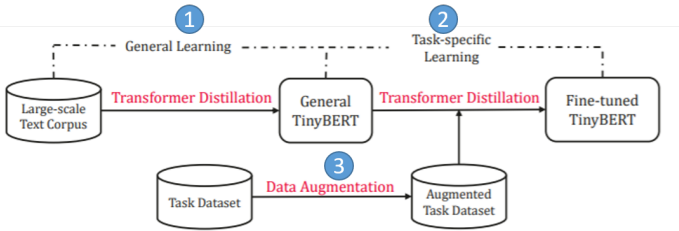
预训练模型蒸馏–损失函数



$$\sum_{x \in X} \sum_{m=1}^{M+1} L(f(s_m(x)), f(t_{g(m)}(x)))$$

- X: Dataset
- m: index of a student layer
- s_m: the m^{th} student layer
- $t_{g(m)}$: the teacher layer corresponding to the m^{th} student layers
- $f(*)$: the knowledge function
- $L(*)$: the loss function

TinyBERT知识蒸馏的基本流程：



(1) 第一步蒸馏 GD(General Distillation)

- 将pre-trained teacher BERT知识迁移到general TinyBERT

(2) 第二步蒸馏 TD(Task-specific Distillation)

- 将fine-tuned teacher BERT知识迁移到fine-tuned TinyBERT

(3) 数据增强 DA(Data Augmentation)

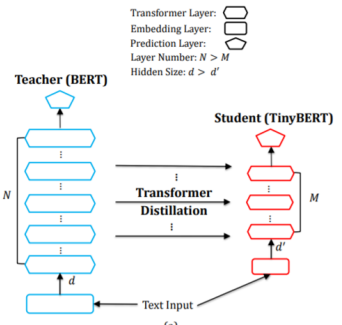
- 论文: Jiao et al. TinyBERT: Distilling BERT for Natural Language Understanding, link
- 开源代码：link

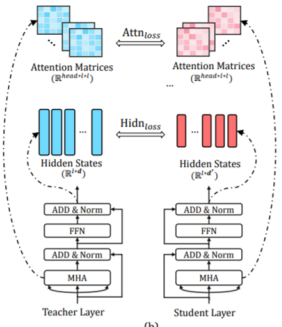
TinyBERT知识蒸馏：损失函数

训练目标函数：

$$\mathcal{L}_{model} = \sum_{m=0}^{M+1} \lambda_m \mathcal{L}_{layer}(S_m, T_{g(m)})$$

- 同时计算词嵌入层、Transformer层和预测层的损失
- 在Transformer层，同时计算隐状态的损失和注意力矩阵的损失





$$\mathcal{L}_{layer}(S_m, T_{g(m)}) = \begin{cases} \mathcal{L}_{embd}(S_0, T_0), & m = 0 \\ \mathcal{L}_{hidn}(S_m, T_{g(m)}) + \mathcal{L}_{attn}(S_m, T_{g(m)}), & M \geq m > 0 \\ \mathcal{L}_{pred}(S_{M+1}, T_{N+1}), & m = M+1 \end{cases}$$

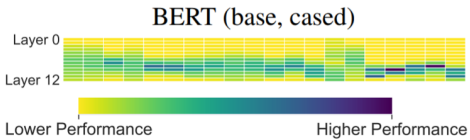
TinyBERT知识蒸馏：选层策略

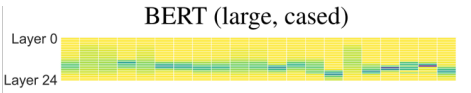
选层策略：学生模型从老师的哪些层学习

- g(m)函数：选层策略需要学习优化，但是GD非常耗费时间
- 均匀选层策略并不是最优，我们通过进化算法来搜索最优的选层方案

Student	Strategy	Layer Mapping	SST-2 (Acc)	MNLI (Acc)	SQuAD v1.1 (EM/F1)	MRPC (Acc/F1)	CoLA (Mcc)	QNLI (Acc)	QQP (Acc/F1)	SQuAD v2.0 (EM/F1)	Avg
4-Layer	Uniform	(3,6,9,12)	87.4	77.0	66.7/77.4	76.5/84.6	21.3	84.9	86.0/81.7	58.9/62.5	72.1
	Last-Layer	(0,0,0,12)	88.1	77.6	69.2/79.4	83.8/88.6	21.4	85.7	87.2/82.9	59.4/63.3	73.4
	Contribution-based	(1,10,11,12)	86.8	76.1	64.4/76.4	79.4/86.3	15.5	85.8	86.1/81.4	61.6/65.1	71.7
	ELM (ours)	(0,0,5,10)	89.9	78.6	71.5/81.2	85.0/89.5	23.9	86.0	87.9/83.6	62.9/66.2	74.9
6-Layer	Uniform	(2,4,6,8,10,12)	90.7	81.2	76.0/84.6	85.0/89.6	27.2	89.2	88.2/84.1	68.0/71.3	77.2
	Last-Layer	(0,0,0,0,0,12)	89.8	81.3	76.0/84.7	85.5/89.7	34.3	89.0	88.7/84.6	68.7/71.9	78.2
	Contribution-based	(1,6,7,10,11,12)	90.0	80.9	75.0/84.1	84.6/89.3	28.5	88.8	88.0/84.2	66.5/70.0	77.0
	ELM (ours)	(0,5,0,0,0,10)	91.5	82.4	77.2/85.7	86.0/90.1	36.1	89.3	89.2/85.4	70.3/73.2	79.2

- 我们发现，预训练大模型的中间层，知识更容易迁移（GD不考虑M+1层）



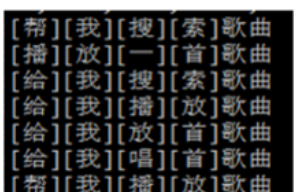


TinyBERT知识蒸馏：数据增强

下游任务通常训练数据不足，我们采用数据增强方法生成更多的伪数据用于TD（面向任务的蒸馏）

- 对于下游任务的每一个训练样本，随机选择一些词语进行替换
- 替换时我们采用BERT和Glove，替换相近的词语
- 通过一个阈值控制被替换词语的比例
- 我们发现伪数据数量为原始下游任务训练数据20倍时蒸馏效果最好

[Mask][Mask][Mask][Mask]歌曲



TinyBERT实验结果：GLUE

System	#Params	#FLOPS	Speedup	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	Avg
$\text{BERT}_{\text{BASE}}$ (Teacher)	109M	22.5B	1.0x	83.9/83.4	71.1	90.9	93.4	52.8	85.2	87.5	67.0	79.5
$\text{BERT}_{\text{TINY}}$	14.5M	1.2B	9.4x	75.4/74.9	66.5	84.8	87.6	19.5	77.1	83.2	62.6	70.2
$\text{BERT}_{\text{SMALL}}$	29.2M	3.4B	5.7x	77.6/77.0	68.1	86.4	89.7	27.8	77.0	83.4	61.8	72.1
BERT_4-PKD	52.2M	7.6B	3.0x	79.9/79.3	70.2	85.1	89.4	24.8	79.8	82.6	62.3	72.6
DistilBERT_4	52.2M	7.6B	3.0x	78.9/78.0	68.5	85.2	91.4	32.8	76.1	82.4	54.1	71.9
$\text{MobileBERT}_{\text{TINY}}$†	15.1M	3.1B	-	81.5/81.6	68.9	**89.5**	91.7	**46.7**	80.1	**87.9**	65.1	**77.0**
TinyBERT_4 (ours)	14.5M	1.2B	9.4x	**82.5/81.8**	**71.3**	87.7	**92.6**	44.1	**80.4**	86.4	**66.6**	**77.0**
BERT_6-PKD	67.0M	11.3B	2.0x	81.5/81.0	70.7	89.0	92.0	-	-	85.0	65.5	-
DistilBERT_6	67.0M	11.3B	2.0x	82.6/81.3	70.1	88.9	92.5	49.0	81.3	86.9	58.4	76.8
TinyBERT_6 (ours)	67.0M	11.3B	2.0x	**84.6/83.2**	**71.6**	**90.4**	**93.1**	**51.1**	**83.7**	**87.3**	**70.0**	**79.4**

TinyBERT实际性能和应用情况

- 相比于BERT-base模型，4层TinyBERT模型加速9.4倍，参数量降为1/7，精度平均下降2.5%，优于现有的所有同量级的模型
- 6层TinyBERT模型精度接近BERT-base模型
- TinyBERT成为终端语音助手NLU算法核心，相对传统模型提高6 10%，B+潜高专利
- TinyBERT目前已成为预训练模型压缩的主流方案，被微软、谷歌、IBM等公司引用，多家媒体报道，Google Scholar引用30+
- 4层TinyBERT基于诺亚AI系统工程实验室开发的Bolt框架，对于常见的短文本（长度小于9个词）在2.5GHz的ARM A76单核float16推理时间仅有2ms, int8推理时间1.3ms ，模型存储只有10MB左右。

TinyBERT: CLUE小模型排行榜

TinyBERT在CLUE小模型排行榜上稳居第一已经很长时间

CLUE小模型排行榜*记最佳得分(模型名称需带有tiny标识，模型参数低于1/9的bert-base参数量)

排行	模型	研究机构	测评时间	Score	认证	AFQMC	TNEWS	IFLYTEK	CMNLI	CLUEWSC 2020	CSL
1	Tiny-NEZHA-4L(9x faster than ...	Huawei Cloud & Noah's Ark lab	20-04-10	59.826	待认证	74.903	67.230	58.808	79.245	0.0	78.767
2	BERT-base	CLUE	19-12-01	58.437	已认证	73.700	56.580	60.290	79.690	0.0	80.360
3	electra-small	electra-small	20-04-17	56.502	待认证	69.956	54.460	56.500	76.929	0.0	81.167
4	electra-official-dis_tiny	electra-official-dis_tiny	20-03-15	55.097	待认证	70.319	54.280	53.538	73.745	0.0	78.700
5	ebm-tiny	ebm-tiny	20-04-14	54.960	待认证	69.904	54.410	55.654	74.823	0.0	74.967

TinyBERT: NLPCC高性能小模型评测

TinyBERT在NLPCC2018的高性能小模型评测比赛中获第一名

结果：

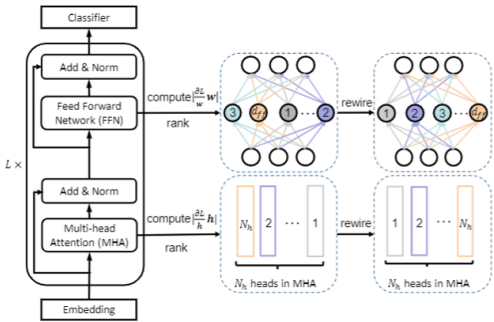
首先感谢大家本次的参与，我们本次比赛的评定规则如下：

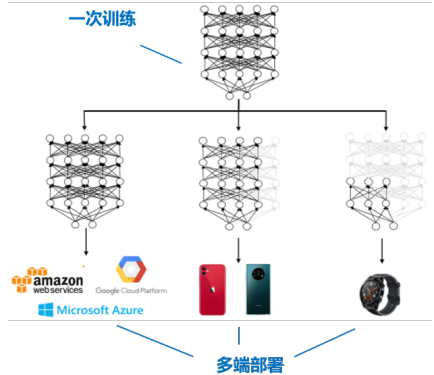
1. 由于测速涉及到硬件条件的限制，所以我们将依据参赛同学自己测定的速度作为模型是否符合规范的依据。
2. 最终分数评定由我们实际测算的速度和参数量按照任务指南中的公式进行计算，分数可能会存在误差，以不影响名次为可接受误差范围内。
3. 具体名次如下：第一名：Huawei Cloud & Noah's Ark lab，得分：0.777126778 第二名：Tencent Oteam，得分：0.768969257 第三名：Xiaomi AI Lab，得分：0.758543871 结果公示三天，如果有问题可以和我们联系。 本次比赛是CLUE第一次正式举办的对外比赛，不免有很多疏漏之处，也非常感谢得到来自各个学校、公司的各位同学大神的指导和帮助。 本次比赛的其他信息会在最终的评估报告中有详细的解释。 CLUE保留对结果的最终解释权。关于奖励发放的事宜会在后续和各位获奖者沟通。 恭喜这些同学，也非常感谢各位的支持。

目录

DynaBERT：动态可伸缩的预训练语言模型

- 一次训练、多设备/多场景部署
- 部署时可以灵活选择不同的子网络进行推理
- 针对Transformer模型，定义网络的深度和宽度，根据神经元/头的重要性进行排序，使得重要的神经元/头被更多子网络共享



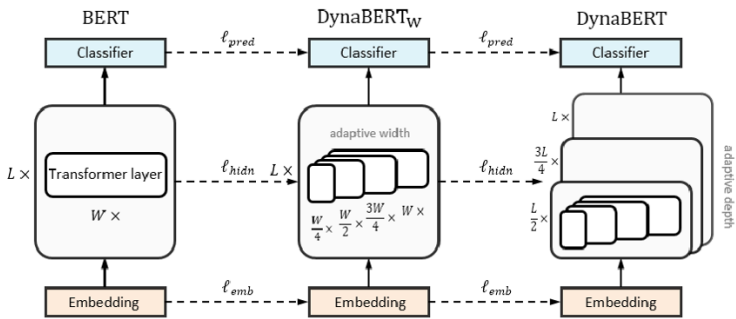


论文： Hou et al., DynaBERT: Dynamic BERT with Adaptive Width and Depth

链接： https://arxiv.org/abs/2004.04037

DynaBERT训练

- 先进行宽度可伸缩网络的训练，再通过宽度和深度同时可伸缩网络的训练。
- 借鉴TinyBERT的Transformer蒸馏技术。

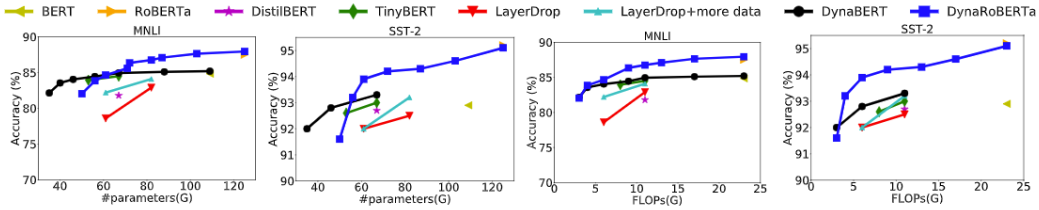


DynaBERT实验结果

Table 1: Development set results of the GLUE benchmark using DynaBERT and DynaRoBERTa with different width and depth multipliers (m_w, m_d).

Method		CoLA			STS-B			MRPC			RTE		
$\text{BERT}_{\text{BASE}}$		58.1			89.8			87.7			71.1		
	(m_w, m_d)	1.0x	0.75x	0.5x	1.0x	0.75x	0.5x	1.0x	0.75x	0.5x	1.0x	0.75x	0.5x
DynaBERT	1.0x	59.7	59.1	54.6	**90.1**	89.5	88.6	86.3	85.8	85.0	72.2	71.8	66.1
	0.75x	**60.8**	59.6	53.2	90.0	89.4	88.5	**86.5**	85.5	84.1	71.8	**73.3**	65.7
	0.5x	58.4	56.8	48.5	89.8	89.2	88.2	84.8	84.1	83.1	72.2	72.2	67.9
	0.25x	50.9	51.6	43.7	89.2	88.3	87.0	83.8	83.8	81.4	68.6	68.6	63.2
		MNLI-(m/mm)			QQP			QNLI			SST-2		
$\text{BERT}_{\text{BASE}}$		84.8/84.9			90.9			92.0			92.9		
	(m_w, m_d)	1.0x	0.75x	0.5x	1.0x	0.75x	0.5x	1.0x	0.75x	0.5x	1.0x	0.75x	0.5x
DynaBERT	1.0x	**84.9/85.5**	84.4/85.1	83.7/84.6	**91.4**	**91.4**	91.1	92.1	91.7	90.6	93.2	93.3	92.7
	0.75x	84.7/85.5	84.3/85.2	83.6/84.4	**91.4**	91.3	91.2	92.2	91.8	90.7	93.0	93.1	92.8
	0.5x	84.7/85.2	84.2/84.7	83.0/83.6	91.3	91.2	91.0	**92.2**	91.5	90.0	**93.3**	92.7	91.6
	0.25x	83.9/84.2	83.4/83.7	82.0/82.3	90.7	91.1	90.4	91.5	90.8	88.5	92.8	92.0	92.0

DynaBERT：精度与响应速度的平衡



(a) #parameters(G). (b) FLOPs(G).

(c) Nvidia K40 GPU latency(s). (d) Kirin 810 ARM CPU latency(ms).

DynaBERT：注意力矩阵可视化

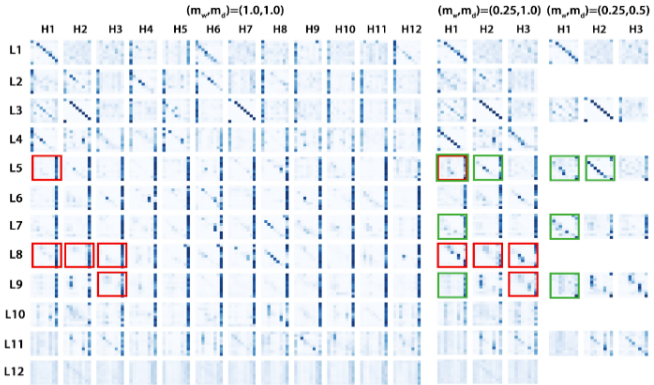


Figure 5: Attention maps in sub-networks with different widths and depths in DynaBERT trained on CoLA.

通过观察我们发现，小模型的注意力模式有可能融合了大模型的多个注意力模式。

目录

目录

Thank you!

把数字世界带入每个人、每个家庭、
每个组织，构建万物互联的智能世界。

Bring digital to every person, home and organization for a fully connected, intelligent world.

HUAWEI