文档排序中同构架构知识蒸馏的实证研究

Xubo Qin Xiyuan Liu Xiongfeng Zheng 京东 科罗拉多大学博尔德分校 腾讯平台与内容事业群
中国 美国 中国 qratosone@live.com heathclief.liu.physics@gmail.com
peacezheng@tencent.com

Jie Liu 腾讯平台与内容事业群 中国 jesangliu@tencent.com

Yutao Zhu yutao.zhu@umontreal.ca 计算机科学与运筹学系 蒙特利尔大学 加拿大

arXiv:2302.04112v1 [cs.IR] 8 Feb 2023

摘要

尽管基于 BERT 的排序模型已广泛应用于商业搜索引擎，但它们通常难以满足在线排序任务的时效要求。知识蒸馏旨在学习一个性能可与大型模型媲美的小型模型，是降低在线推理延迟的常用策略。本文研究了不同损失函数对基于 BERT 的排序模型进行统一架构蒸馏的效果。这里的"统一架构"是指教师模型和学生模型均采用交叉编码器架构，而学生模型包括小规模的预训练语言模型。实验结果表明，排序任务的最优知识蒸馏配置与通用自然语言处理任务大不相同。具体而言，当学生模型采用交叉编码器架构时，基于硬标签的成对损失对于训练学生模型至关重要，而以 Transformer 中间层为目标的知识蒸馏可能会损害性能。这些发现凸显了针对采用成对训练样本的文档排序任务，精心设计适用于交叉编码器学生模型的知识蒸馏策略的必要性。

1 引言

近年来，深度学习方法在信息检索任务中的应用取得了长足进展 [19]。尤其是在文档排序方面，BERT 等预训练语言模型（PLM） [4], 已取得最先进的性能。然而，由于这些预训练模型通常包含大量参数，因此在推理阶段会带来不可避免的计算成本和延迟 [6]。在部署预训练模型以执行对延迟敏感的在线排序任务时，这一问题会更加严重。为解决这一问题，研究人员广泛研究了许多基于 PLM 的知识蒸馏（KD）方法 [14] 。知识蒸馏的原理可概括如下：

首先，使用训练数据中的标签训练教师模型，然后同时利用训练数据和教师模型训练学生模型。具体而言，训练学生模型时，将数据中的真实标签作为硬标签，并将教师模型的输出逻辑值作为软标签。与仅使用硬标签直接优化学生模型相比，加入软标签可以帮助学生模型学习模仿教师模型的行为，从而提升性能。选择合适的损失函数（也称为蒸馏目标 [7]）来学习软标签至关重要。

现有的大多数方法 [7, 15–17, 22] 旨在通过语言模型知识蒸馏来提升学生模型在通用自然语言理解（NLU）任务中的性能 [21]。特别是在文档检索和排序任务中，近期的一些方法正专注于跨架构蒸馏方法 [5, 10] ，使用交叉编码器 [12]教师模型，以及基于 ColBERT [8]或双编码器 [2]的学生模型。对于这些跨架构方法，教师模型和学生模型都使用相同规模的预训练模型（即 ERNIE2.0 [18] ，采用12层或24层 Transformer [20] 编码器）。由于文档检索和排序任务的训练样本量通常非常庞大，训练大规模教师模型和学生模型可能需要数千个 GPU时长，这对于一些商业搜索引擎来说通常难以承受。因此，仍有必要探索采用小规模交叉编码器学生模型的统一架构蒸馏方法。

本文开展了一项实证研究，以探究不同知识蒸馏目标在文档排序任务中的有效性。我们重点分析**给定一个采用4层Transformer的小规模PLM的交叉编码器学生模型时的蒸馏效果**。对于排序任务中的统一架构蒸馏，最新的方法之一是简化版TinyBERT [3]（以下简称STinyBERT），该方法针对一般自然语言理解任务对TinyBERT [7] 进行了一些简化。一个主要问题是，TinyBERT和STinyBERT中的知识蒸馏目标都基于逐点训练样本，而这类样本适用于分类问题。然而，在文档排序中，成对训练得到了广泛应用，并且已被证明更加有效 [1, 9]。因此，本文的第一个研究问题是如何将成对排序与知识蒸馏目标相结合。我们提出了一种将成对排序融入STinyBERT的直接方法，并设计了一个成对蒸馏目标。随后，在分析成对排序的机制后，我们认为

特此授权免费制作本作品的全部或部分数字副本或纸质副本，供个人或课堂使用，但前提是不得出于营利或商业利益而制作或分发副本，并且副本必须包含本声明及首页上的完整引文。对于本作品中由非美国计算机协会所有的组件，必须尊重其版权。经注明出处后允许进行摘要。若要以其他方式复制、重新发表、发布到服务器或重新分发到邮件列表，则须事先获得具体许可和/或支付费用。请通过 permissions@acm.org 申请许可。*Conference'17, 2017*年7月，美国华盛顿哥伦比亚特区© 2022年 美国计算机协会。美国计算机协会 ISBN 978-x-xxxx-xxxx-x/YY/MM...$15.00 https://doi.org/XXXXXXX.XXXXXXX

在原始STinyBERT中，基于Transformer中间层的蒸馏目标是多余的。我们进一步开展的实证研究证实了这一推测。基于实验结果，我们简化了STinyBERT中的知识蒸馏目标，最终提出了一种**PD-BERT**（PD代表成对蒸馏）知识蒸馏文档排序策略。

我们在 MS MARCO数据集上进行了实验。实验结果表明，我们提出的 PD-BERT 虽然采用了更少的蒸馏目标，但性能优于现有方法。结果清楚地表明，要使排序模型达到最佳性能，应精心设计一些基于成对训练样本的新型蒸馏目标，并确保这些目标能够表示成对样本之间的相对关系。

2 相关工作

基于 BERT 的排序模型有多种架构，包括交叉编码器 [12],双编码器 [2]和 ColBERT [8]。与其他架构相比，交叉编码器能够取得最佳性能，但其推理延迟也高于其他架构。对于商业搜索引擎而言，这是一个关键问题。为了解决这一问题，研究人员提出了各种知识蒸馏方法，旨在训练蒸馏BERT模型，以提升推理速度，同时尽可能减少性能损失；交叉编码器模型通常用作教师模型。

典型的预训练语言模型知识蒸馏方法 [7, 15–17, 22] 主要关注通用自然语言理解（NLU）任务 [21]。对于文档检索和排序任务，相关方法可分为**跨架构** 方法和**同架构** 方法。对于采用大规模预训练语言模型的交叉编码器教师模型，Margin-MSE [5]或 ERNIE-Search [10] 等跨架构方法，侧重于将知识蒸馏到 ColBERT 或双编码器架构的学生模型中，而这些学生模型所用的预训练语言模型与教师模型规模相当（例如包含 12 个 Transformer 层）。而在同架构方法中，学生模型是采用较小规模预训练语言模型的交叉编码器。表1展示了跨架构与同架构之间的差异。

对于排序任务，现有最先进的同架构知识蒸馏方法是简化版TinyBERT（STinyBERT） [3] ，它是面向通用自然语言理解任务的 TinyBERT [7] 的扩展。秉承 TinyBERT 的思路，STinyBERT 采用三个目标来学习部分 Transformer 中间层的权重，包括嵌入层权重、隐状态和注意力矩阵。与原始 TinyBERT 相比，STinyBERT 将任务特定蒸馏的两个步骤合并为一步，并基于真实标签增加了逐点损失。然而，由于排序任务的训练数据通常以成对训练样本 [9], 的形式组织，现有蒸馏目标在成对训练样本上的效果仍不明确。

不同于已有研究，我们提出了 PD-BERT 策略，将基于点式的蒸馏方法适配到成对样本上。我们进一步开展了实证研究，并据此简化了蒸馏目标，取得了更好的结果。

表1：交叉架构与统一架构方法的比较。“PLM 规模”表示学生模型所用预训练语言模型的规模。“Arc.”和“Enc.”分别是“架构”和“编码器”的缩写。

Type	学生架构	预训练语言模型规模
统一架构	交叉编码器（与教师相同）	比教师更小
交叉架构	ColBERT 或双编码器	与教师相同

3 所提方法

3.1 背景

在详细介绍我们的方法之前，我们先简要介绍一些背景知识，包括点式/成对排序，以及 STinyBERT 中常用的蒸馏目标。

3.1.1 逐点排序与成对排序。逐点方法和成对方法是学习排序框架中两种常用的方法。**逐点方法**在损失函数中每次考虑一个文档。它们通常会训练一个分类器（使用交叉熵损失），以判断文档与查询是否相关。随后，根据文档的预测分数进行排序，得到最终排名。在逐点方法中，每个文档的分数都独立于其他文档的分数。相反，**成对方法**在损失函数中考虑一对文档，其目标是在同一查询下学习两个文档的相对顺序。实践中，成对方法通常比逐点方法表现更好，因为预测相对顺序更符合排序的本质 [1, 9]。具体而言，对于查询 q，假设有一个相关文档 d^+ 和一个不相关文档 d^-，成对排序损失可定义为：

$$\mathcal{L}_{\text{pair}} = \max(0, 1 - f(q, d^+) + f(q, d^-)), \tag{1}$$

其中，$f(q,\ d)$是 q 与 d之间的排序分数。与逐点方法类似，最终的排序列表可通过按照排序分数对文档进行排序获得。

3.1.2 STinyBERT 中的知识蒸馏。简化版TinyBERT（STinyBERT）专为文档检索而设计，它在 TinyBERT [7] 的基础上进行了若干简化。它通过以下五个目标从 BERT 中进行知识蒸馏：

$$\mathcal{L}_1 = \mathcal{L}_{\text{attn}} + \mathcal{L}_{\text{hidn}} + \mathcal{L}_{\text{emb}} + \mathcal{L}_{\text{hard}} + \mathcal{L}_{\text{logits}}. \tag{2}$$

前三个目标用于从教师模型中学习注意力权重、中间层隐藏层和嵌入层，其定义为：$\mathrm{L}_i = \mathrm{MSE}(L_i^T,\ L_i^S)\ i \in \{\text{attn、hidn、emb}\}$。$L_i^T$和 L_i^S分别是教师模型和学生模型的注意力矩阵/隐藏层矩阵/嵌入矩阵。通过优化这些目标，学生模型可以使其参数度量更接近教师模型，从而获得可比的性能。最后两个目标定义于标准知识蒸馏中。L_{hard} 是根据学生模型的输出逻辑值$\mathbf{z}^S$和标签计算得到的交叉熵损失，而$\mathrm{L}_{\text{logits}}$ 是用于衡量$\mathbf{z}^S$与教师模型逻辑值$\mathbf{z}^T$之间差异的软交叉熵损失。为适应知识蒸馏，将$\mathbf{z} \in \mathbb{R}^2$排序任务构造为二分类任务。相关的概率（即，$\mathbf{z}^S[0]$）被用作排序分数 $f(q,\ d)$。

更多详情请参阅TinyBERT [7] 和STinyBERT [3]的原论文。

3.2 成对蒸馏目标

在原始 STinyBERT 中，$L_{硬}$ 被定义为交叉熵损失，可视为一种逐点方法。正如第 3.1.1 节所述，研究人员已证明，在实践中，成对方法的表现优于逐点方法。因此，我们的第一个研究问题是：如何将成对排序与知识蒸馏目标相结合？

一个直接的想法是用公式1中定义的 L_{pair} 替换公式2中的 $L_{硬}$。然而，与逐点排序不同，成对排序涉及两篇文档（即，d^+ 和 d^-）。为了在知识蒸馏中考虑这样的文档对，我们提出了如下成对蒸馏目标：

$$\mathcal{L}_2 = \mathcal{L}_{\text{pair}} + \mathcal{L}^+ + \mathcal{L}^-, \tag{3}$$

$$\mathcal{L}^+ = \mathcal{L}^+_{\text{attn}} + \mathcal{L}^+_{\text{hidn}} + \mathcal{L}^+_{\text{emb}} + \mathcal{L}^+_{\text{logits}}, \tag{4}$$

$$\mathcal{L}^- = \mathcal{L}^-_{\text{attn}} + \mathcal{L}^-_{\text{hidn}} + \mathcal{L}^-_{\text{emb}} + \mathcal{L}^-_{\text{logits}}, \tag{5}$$

其中，L^+ 和 L^- 分别在正样本对（q，d^+）和负样本对（q，d^-）上计算。使用 L_2 进行知识蒸馏的 STinyBERT 记为“STinyBERT+成对蒸馏”。

在公式 3 中可以看到，总共需要计算九个目标，这非常复杂。因此，我们接下来的研究问题是：这些目标是否都是必要的？

原始 STinyBERT 是为逐点排序而设计的。针对不同层或注意力的知识蒸馏目标，可以帮助学生模型获得与教师模型相似的参数，从而使学生模型输出相似的排序分数。直观而言，对于我们的成对蒸馏目标而言，这种约束可能并非必要，因为我们的模型经过训练，学习的是两个文档之间的相对顺序，而非为单个文档计算特定的排序分数。为验证这一假设，我们开展了一项实证研究，发现针对中间参数（即，L_i^+ 和 L_i^+，$i \in$ {attn，emb，hidn}）的蒸馏目标甚至会损害模型的性能（实验结果见第4.3.1节）。因此，我们对成对蒸馏目标进行了简化，如下所示：

$$\mathcal{L}_3 = \mathcal{L}_{\text{pair}} + \mathcal{L}^+_{\text{logits}} + \mathcal{L}^-_{\text{logits}}. \tag{6}$$

使用我们提出的 L_3 进行蒸馏的 STinyBERT 记作“PD-BERT”。

4 实验

4.1 数据集与实现细节

按照先前的工作 [3]，我们在 MSMARCO [11] 段落排序数据集的一个子集上进行实验，仅使用前 320 万个样本进行训练。查询和段落的长度分别截断为 32 和 120。评估时，我们使用MSMARCO官方提供的完整开发集，其中包含约 7,000 个查询，每个查询包含 1,000 个用于重排序的段落。我们使用 MRR@10 作为评估指标，这是官方评估所建议的指标。

表2：不同方法的性能。

模型	Type	MRR@10
BERT-base（教师）	微调	**0.359**
TinyBERT-L4	微调	0.324
STinyBERT	蒸馏	0.330
STinyBERT+成对	蒸馏	0.334
Margin-MSE	蒸馏	0.335
PD-BERT	蒸馏	**0.340**

我们使用 PyTorch [13] 和 HuggingFace 的 Transformers [23] 实现模型。对于教师模型，我们以 3e-6 的学习率和 32 的批次大小，对一个预训练的 BERT-base [4] 模型进行一轮微调。该模型有 12 个 Transformer 层，所有层的隐藏维度均为 768。对于所有学生模型，我们使用通用蒸馏版 TinyBERT 检查点初始化模型。[1] 该模型称为 TinyBERT-L4，由四个隐藏维度为 312 的 Transformer 层组成。$L_{逻辑值}$ 的温度固定为 1。其他超参数和知识蒸馏设置与原始 STinyBERT [3]中的相同。

4.2 基线

我们将提出的 PD-BERT 与三组方法进行比较：

(1) **微调方法**。 我们在该数据集上对 BERT-base 和 TinyBERT-L4 进行微调，并报告其性能。该 BERT-base 模型用作后续知识蒸馏方法中的教师模型。

(2) **蒸馏方法**。 为验证我们提出的成对蒸馏目标的有效性，我们报告原始 STinyBERT 和 STinyBERT+Pairwise 的性能。这两种方法都包含基于 Transformer 的蒸馏目标 L_{attn},L_{hidn}, L_{emb}。我们还报告 PD-BERT 的性能；如公式 6 所述，该方法忽略了所有这些基于 Transformer 的目标函数。

(3) **STinyBERT 变体**。 为探究不同知识蒸馏目标的影响，我们从公式 3 中移除部分目标，并测试这些变体的性能。

4.3 结果与讨论

4.3.1 总体结果。 表2展示了不同微调方法和知识蒸馏方法的实验结果。可以看出，我们提出的 PD-BERT 优于所有其他基线，这充分证明了其优越性。此外，我们还有以下观察：

(1) 所有知识蒸馏方法的性能都优于经过微调的 TinyBERT-L4，从而缩小了学生模型与教师模型（BERT-base）之间的差距。此外，我们的 PD-BERT 在所有这些方法中取得了最佳性能。

(2) 与 STinyBERT 相比，STinyBERT+Pairwise 的 MRR@10 绝对提升了 0.004。这两个模型的区别在于排序损失。因此，这一提升验证了我们提出的成对蒸馏目标的有效性。

[1] https://huggingface.co/huawei-noah/TinyBERT_通用_4L_312D。请注意，Tiny-BERT 同时提供通用知识蒸馏和任务特定蒸馏的检查点，我们使用通用检查点来初始化学生模型。

表3：消融实验结果。$\mathcal{L}_{\text{pair}}$ 用于所有变体中，因此省略。所有蒸馏目标均在正样本和负样本上计算，因此标记 +/− 也被省略。

模型	蒸馏目标	MRR@10
STinyBERT+成对	L^{logits}，L_{emb}，L_{hidn}，L_{attn}	0.334
↪不含中间层	L^{logits}、L_{emb}	0.339
↪不含嵌入	$\mathcal{L}^{\text{logits}}$，$\mathcal{L}_{\text{attn}}$，$\mathcal{L}_{\text{hidn}}$	0.334
↪不使用 逻辑值	L_{attn}，L_{hidn}，L_{emb}	0.319
PD-BERT	L^{logits}	**0.340**

表4：不同知识蒸馏设置下的性能。“教师层”表示用于知识蒸馏的教师模型的 Transformer 层。

模型	教师层	MRR@10
STinyBERT+成对	3,6,9,12	0.334
↪ 最后 4 层	9,10,11,12	0.333
↪ 最后 1 层	12	0.339
PD-BERT	-	**0.340**

这一结果与先前研究的发现一致：成对排序通常优于逐点排序 [9]。

(3) 有趣的是，与 STinyBERT+Pairwise 相比，我们的 PD-BERT 使用更少的蒸馏目标进行训练，却取得了更好的性能，这表明使用成对训练样本时，基于 Transformer 的蒸馏目标会损害性能。这证实了我们的假设：成对蒸馏方法可以放宽对中间层参数学习的约束。通过更灵活地调整参数，在提供成对训练样本时，学生模型能够更好地适应下游任务。

4.3.2 消融研究。先前的实验结果表明，某些知识蒸馏目标并不适用于文档排序任务，尤其是成对排序。因此，我们进行了消融实验，以探究不同蒸馏目标对 STinyBERT+Pairwise 的影响。实验结果如表3所示。我们可以看到：

首先，移除任何针对中间层参数的蒸馏目标都会带来性能提升。这再次证实了我们的假设：使用成对蒸馏目标进行训练时，无须模拟中间 Transformer 层中的参数。其次，与 L_{emb} 相比，加入 L_{attn} 和 L_{hidn} 会对模型性能产生负面影响。由于中间层的参数数量远多于嵌入层，我们推测，学习过多参数可能会干扰成对蒸馏。最后，在移除

$L_{\text{逻辑值}}$，模型性能显著下降。这表明，从教师模型学习输出逻辑值至关重要。

4.3.3 不同知识蒸馏设置的影响。先前研究 [24]分析了 BERT 在文档排序中不同层的作用，并报告称，BERT 在很大程度上依赖最后几层之间的交互

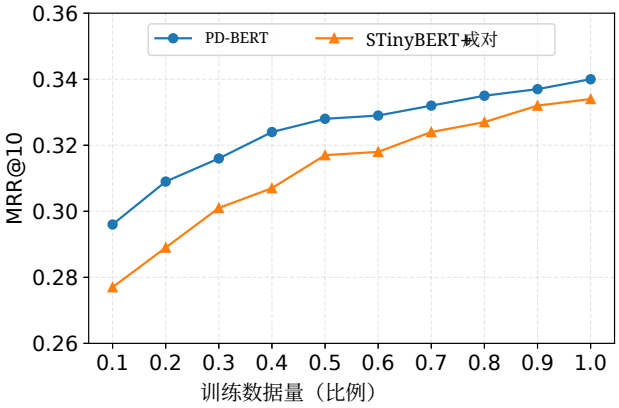


图1：不同训练数据量下的性能比较。

来预测查询与文档之间的相关性。类似地，在知识蒸馏中，BERT 中蒸馏层的选择也可能影响性能。遵循先前的工作 [7], , 我们测试了 BERT 各层的几种不同知识蒸馏设置（也称为映射函数）。结果见表4，我们得出以下发现：

首先，使用最后四层进行知识蒸馏并不能带来进一步提升。对此的一种解释是，尽管最后四层被认为对于捕捉查询与文档之间的匹配信号至关重要，但它们也依赖底层。由于STinyBERT的层数有限，它们无法像原始BERT那样提取大量信息。其次，仅对BERT的最后一层进行知识蒸馏时，STinyBERT+成对的表现略逊于我们的PD-BERT。这表明，对BERT中间层进行知识蒸馏并无作用。因此，我们对蒸馏目标的简化直接明了且行之有效。

4.3.4 训练数据量的影响。知识蒸馏通常会受到训练数据数量的影响。为研究这种影响，我们测试了仅使用一定比例的训练数据（从 0.1 到 1.0）训练的 PD-BERT 和 PTiny-BERT+Pairwise 的性能。结果如图1所示。

显而易见，随着使用的训练数据增多，PD-BERT 和 STinyBERT+Pairwise 的性能都在提升。这是很自然的，因为知识蒸馏受益于充足的数据。此外，我们还可以观察到，提供更多训练数据后，STinyBERT+Pairwise 与 PD-BERT 之间的性能差距会缩小。一种可能的解释是，在成对训练中，许多候选文档会在多个文档对中重复使用。我们在第 3.2 节中指出，这些蒸馏目标无法衡量这些文档对之间的相对顺序。因此，训练集中大多数此类文档对于蒸馏目标而言都是冗余的，这可能导致过拟合。与针对输出逻辑值的蒸馏目标相比，旨在学习中间参数的蒸馏目标更容易过拟合，并且需要更多不同的训练文档进行调整。

5 结论

本文研究了排序任务中广泛使用的统一架构蒸馏目标的效果。对于采用交叉编码器架构的学生模型，我们证明，TinyBERT 中现有的基于 Transformer 的目标函数并不适用于成对训练样本。针对 Transformer 中间层的蒸馏目标甚至会损害学生模型的性能。这可能是因为现有的蒸馏目标基于逐点训练样本，而在成对训练中，由于不同的逐点样本数量不足，基于 Transformer 的目标函数更容易过拟合。为了适应排序任务，应谨慎设计新型蒸馏目标，以衡量成对样本中的相对顺序。

参考文献

[1] Christopher JC Burges。2010。从 RankNet 到 LambdaRank 再到 LambdaMART：概述。*Learning* 11, 23-581 (2010), 81. [2] Yinqiong Cai、Yixing Fan、Jiafeng Guo、Fei Sun、Ruqing Zhang 和 Xueqi Cheng。2021。第一阶段检索的语义模型：全面综述。*CoRR* abs/ 2103 04831 2021 2103 04831 2103 04831 . ( ). arXiv: . [3] https://arxiv.org/abs/ 2021 。Xuanang Chen、Ben He、Kai Hui、Le Sun 和 Yingfei Sun。。简化版TinyBERT：用于文档检索的知识蒸馏。载于信息检索进展——第*43*届欧洲信息检索研究会议 *ECIR 2021*，线上活动，*2021*年*3*月*28*日至*4*月*1*日，论文集，第二部分（《计算机科学讲义》第*12657*卷），Djoerd Hiemstra、Marie-Francine Moens、Josiane Mothe、Raffaele Perego、Martin Potthast 和 Fabrizio Sebastiani（编）。施普林格，241–248。https://doi.org/ 10 1007 978 3 030 72240 1_21 . / - - - - Jacob Devlin、[4] Ming-Wei Chang、Kenton Lee 和 Kristina Toutanova。2019。BERT：用于语言理解的深度双向变换器预训练。载于*2019*年北美计算语言学协会会议论文集：人类语言技术，*NAACL-HLT 2019*，美国明尼阿波利斯，*2019*年*6*月*2*日至*7*日，第一卷（长篇与短篇论文），Jill Burstein、Christy Doran 和 Thamar Solorio（编）。计算语言学协会，4171–4186。https://doi.org/10.18653/v1/n19-1423[5] Sebastian Hofstätter、Sophia Althammer、Michael Schröder、Mete Sertkan 和 Allan Hanbury。2020。通过跨架构知识蒸馏改进高效神经排序模型。arXiv: 2010 02666 [ ] . [6] cs.IR Sebastian Hofstätter 和 Allan Hanbury。。让我们测量运行时间！扩展信息检索可复现性基础设施，以纳入性能方面的内容。arXiv:1907.04614 [cs.IR][7] Xiaoqi Jiao、Yichun Yin、Lifeng Shang、Xin Jiang、Xiao Chen、Linlin Li、Fang Wang 和 Qun Liu。2020。TinyBERT：通过蒸馏 BERT 实现自然语言理解。载于*2020*年自然语言处理实证方法会议论文集：论文集，*EMNLP 2020*，线上活动，*2020*年*11*月*16*日至*20*日，Trevor Cohn、Yulan He 和 Yang Liu（编）。计算语言学协会，4163 4174 10 18653 1 2020 372 – 。[8] https://doi.org/ 。/v / .findings-emnlp 2020 。Omar Khattab 和 Matei Zaharia。。*ColBERT*：通过 *BERT* 上下文化晚期交互实现高效且有效的段落检索。美国计算机协会，美国纽约州纽约市， 39 48 10 1145 3397271 3401075 – 。[9] https://doi.org/ 。/ 。2010 Tie-Yan Liu。。信息检索中的学习排序。载于第*33*届国际 *ACM SIGIR* 信息检索研究与发展会议论文集，*SIGIR 2010*，瑞士日内瓦，*2010*年*7*月*19*日至*23*日，Fabio Crestani、Stéphane Marchand-Maillet、Hsin-Hsi Chen、Efthimis N. Efthimiadis 和 Jacques Savoy（编）。ACM，904。https://doi.org/10.1145/1835449.1835676[10] Yuxiang Lu、Yiding Liu、Jiaxiang Liu、Yunsheng Shi、Zhengjie Huang、Shikun Feng、Yu Sun、Hao Tian、Hua Wu、Shuaiqiang Wang、Dawei Yin 和 Haifeng Wang。2022。ERNIE-Search：通过自适应在线蒸馏连接交叉编码器与双编码器，以实现稠密段落检索。https://doi.org/10.48550/ARXIV.2205.09153[11] Tri Nguyen、Mir Rosenberg、Xia Song、Jianfeng Gao、Saurabh Tiwary、Rangan Majumder 和 Li Deng。2016。MS MARCO：人工生成的机器阅读理解数据集。载于认知计算研讨会论文集：整合神经与符号方法 *2016*（与第*30*届神经信息处理系统年会同期举行），西班牙巴塞罗那，*2016*年*12*月*9*日（*CEUR* 研讨会论文集，第*1773*卷），Tarek Richard Besold、Antoine Bordes、Artur S. d'Avila Garcez 和 Greg Wayne（编）。CEUR-WS.org。http://ceur-ws.org/Vol-1773/CoCoNIPS_2016_paper9.pdf [12] Rodrigo Frassetto Nogueira、Wei Yang、Kyunghyun Cho 和 Jimmy Lin。2019。基于 BERT 的多阶段文档排序。*CoRR* abs/1910.14424 (2019)。arXiv:1910.14424 http://arxiv.org/abs/1910.14424

[13] Adam Paszke、Sam Gross、Francisco Massa、Adam Lerer、James Bradbury、Gre- gory Chanan、Trevor Killeen、Zeming Lin、Natalia Gimelshein、Luca Antiga、Alban Desmaison、Andreas Köpf、Edward Yang、Zachary DeVito、Martin Rai- son、Alykhan Tejani、Sasank Chilamkurthy、Benoit Steiner、Lu Fang、Junjie Bai 和 Soumith Chintala。2019年。PyTorch：一款命令式风格、高性能深度学习库。载于神经信息处理系统进展第*32*卷：神经信息处理系统年会 *2019*，*NeurIPS 2019*，*2019*年*12*月*8*日至*14*日，加拿大不列颠哥伦比亚省温哥华，Hanna M. Wallach、Hugo Larochelle、Alina Beygelzimer、Florence d'Alché-Buc、Emily B. Fox 和 Roman Garnett（编）。8024–8035。https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html[14] Adriana Romero、Nicolas Ballas、Samira Ebrahimi Kahou、Antoine Chassang、Carlo Gatta 和 Yoshua Bengio。2015年。FitNets：精简深度网络的提示。arXiv: 1412 6550 [ ] 。cs.LG Victor [15] Sanh、Lysandre Debut、Julien Chaumond 和 Thomas Wolf。2019年。DistilBERT：BERT 的蒸馏版本，更小、更快、更便宜、更轻。*CoRR* abs/ 1910 01108 2019 1910 01108 1910 01108 。（）。arXiv: 。[16] http://arxiv.org/abs/ 2019 。Siqi Sun、Yu Cheng、Zhe Gan 和 Jingjing Liu。。BERT模型压缩的渐进式知识蒸馏。载于*2019*年自然语言处理实证方法会议暨第九届国际自然语言处理联合会议论文集，*EMNLP-IJCNLP 2019*，中国香港，*2019*年*11*月*3*日至*7*日，Kentaro Inui、Jing Jiang、Vincent Ng 和 Xiaojun Wan（编）。计算语言学协会，4322–4331。https://doi.org/ 10 18653 1 19 1441 。/v /D - Siqi Sun、Zhe Gan、Yu [17] Cheng、Yuwei Fang、Shuohang Wang 和 Jingjing Liu。2020年。面向语言模型压缩的中间表示对比蒸馏。*CoRR* abs/2009.14167（2020年）。arXiv:2009.14167 https: // arxiv.org/abs/2009.14167[18] Yu Sun、Shuohuan Wang、Yukun Li、Shikun Feng、Hao Tian、Hua Wu 和 Haifeng Wang。2019年。ERNIE 2.0：一种面向语言理解的持续预训练框架。https://doi.org/ 10 48550 1907 12412 。/ ARXIV. 。Dacheng Tao。。[19] 2020 深度学习如何应用于信息检索。美国计算机协会，纽约，美国，5。https://doi.org/10.1145/ 3397271.3402429[20] Ashish Vaswani、Noam Shazeer、Niki Parmar、Jakob Uszkoreit、Llion Jones、Aidan N Gomez、Ł ukasz Kaiser 和 Illi a Polosukhin。2017年。Attention Is All You Need。载于神经信息处理系统进展第*30*卷，I. Guyon、U. V. Luxburg、S. Bengio、H. Wallach、R. Fergus、S. Vishwanathan 和 R. Garnett（编）。Curran Associates, Inc.，5998–6008。http://papers.nips.cc/paper/7181-attention- is-all-you-need.pdf [21] Alex Wang、Amanpreet Singh、Julian Michael、Felix Hill、Omer Levy 和 Samuel R. Bowman。2018年。GLUE：自然语言理解的多任务基准与分析平台。*CoRR* abs/1804.07461（2018年）。arXiv:1804.07461 http://arxiv.org/abs/ 1804 07461 。Wenhui [22] Wang、Furu Wei、Li Dong、Hangbo Bao、Nan Yang 和 Ming Zhou。2020年。MiniLM：用于任务无关压缩预训练变换器的深度自注意力蒸馏。arXiv:2002.10957 [cs.CL] [23] Thomas Wolf、Lysandre Debut、Victor Sanh、Julien Chaumond、Clement Delangue、Anthony Moi、Pierric Cistac、Tim Rault、Rémi Louf、Morgan Funtowicz 和 Jamie Brew。2019年。HuggingFace 的 Transformers：最先进的自然语言处理。*CoRR* abs/1910.03771（2019年）。arXiv:1910.03771 http://arxiv.org/abs/1910.03771[24] Jingtao Zhan、Jiaxin Mao、Yiqun Liu、Min Zhang 和 Shaoping Ma。2020年。BERT在文档排序中的应用分析。载于第*43*届国际 *ACM SIGIR* 信息检索研究与发展会议论文集，*SIGIR 2020*，线上活动，中国，*2020*年*7*月*25*日至*30*日，Jimmy Huang、Yi Chang、Xueqi Cheng、Jaap Kamps、Vanessa Murdock、Ji-Rong Wen 和 Yiqun Liu（编）。美国计算机协会，1941–1944。https://doi.org/10.1145/3397271.3401325