通过随机填充轻松改进位置嵌入

Mingxu Tao[1,2], Yansong Feng[1,3],Dongyan Zhao[1,2][1]北京大学王选计算机研究所，北京大学，中国[2]北京大学数据科学研究中心，中国[3]北京大学教育部计算语言学重点实验室，中国 {thomastao, fengyansong, zhaody}@pku.edu.cn

arXiv:2305.04859v1 [cs.CL] 8 May 2023

摘要

位置嵌入对文本序列中词元之间位置关系进行编码，对于基于 Transformer 的预训练语言模型建模局部上下文特征至关重要。然而，在抽取式问答中，使用不同上下文长度的实例训练的位置嵌入，效果可能不如预期。由于后部位置的嵌入更新次数少于前部位置的嵌入，后部位置的嵌入可能没有得到充分训练。本文提出了一种简单而有效的策略：随机填充，且无需修改现有预训练语言模型的架构。在微调时，我们调整输入序列中词元的顺序，以平衡每个位置嵌入的更新次数。实验表明，随机填充能够显著提升答案位于后部位置的实例上的模型性能，尤其是在模型使用短上下文训练、但在长上下文上评估时。我们的代码和数据将会公开，以供未来研究使用。

1 引言

预训练语言模型（Devlin 等人，2019年；Liu 等人，2019年；Clark 等人，2020年；Joshi 等人，2020年）在多种自然语言处理任务中取得了巨大成功，包括文本分类、关系抽取和抽取式问答（QA）。采用 Transformer 架构的这些模型（Vaswani 等人，2017年）具有强大的局部上下文建模能力，这对问答至关重要（Salant 和 Berant，2018年；Peters 等人，2018年）。与预测主要基于句子级表示的文本分类模型相比，许多抽取式问答模型必须从上下文中的所有词元里确定答案片段的起止边界。抽取式问答模型倾向于选取与上下文和问题之间重叠词的相邻词作为答案（Jia 和 Liang，2017年；Sugawara 等人，2018年），因此应当对上下文中词的相对位置保持敏感。因此，如何表示词之间的位置关系对问答模型至关重要。

基于 Transformer 的模型仅使用位置嵌入来识别词元的顺序，从而编码词元之间的位置关系。许多流行的基于 Transformer 的模型，例如 BERT（Devlin 等人，2019年），采用绝对位置嵌入，可将其视为一组可学习的向量。对于每个词元，其对应的位置嵌入都会附加到其词元嵌入上，以形成最终的输入表示。因此，在微调过程中，如果词元序列的长度短于最大输入长度（例如，BERT 的最大输入长度为 512），则位置嵌入向量的后半部分不会像前半部分那样得到同等程度的更新。在实际应用中，不可能所有实例的长度都恰好等于预训练语言模型的最大输入长度，因此，位置嵌入的后半部分可能比前半部分得到的更新次数更少。这可能会阻碍问答模型为位于上下文末尾的答案词元生成更加准确的表示。

许多近期研究也聚焦于 短训练、长测试（Su 等人，2021年；Press 等人，2022年；Sun 等人，2022年）。这些研究采用相对位置嵌入，以避免部分绝对位置嵌入无法更新的情况。然而，这些方法需要修改 Transformer 层的注意力机制，并重新预训练语言模型。与这些研究不同，我们致力于增强现有的绝对位置嵌入模型。

在本文中，我们首先在抽取式问答任务中开展了一项试验性实验，以展示训练不足将如何影响位置嵌入的质量。我们

提出了一种简单但有效的方法，即随机填充。我们通过随机移动填充标记，减少位置嵌入前部的更新次数，并使后部的位置嵌入得到更多次更新。为检验随机填充是否能够有效改进问答模型，我们首先在一种严苛场景下进行实验：模型在具有较短上下文的实例上进行训练，但必须在更长的上下文上执行推理。我们还在训练集和测试集的上下文长度分布相近的数据集上进行了实验。实验结果表明，随机填充在这两种场景下都能改进问答模型。我们进一步提供了实证分析，结果表明，随机填充通过使输入序列中的后部词元更好地建模局部上下文的位置特征，在改进这些词元的表示方面发挥着重要作用。

我们的主要贡献如下：（1）我们提出了一种简单但有效的方法，称为随机填充（*Random Padding*），用于改进后部位置的嵌入，尤其是在模型使用短上下文进行训练时。（2）我们将随机填充（*Random Padding*）引入流行的基于 Transformer 的问答模型中，无需对其架构进行任何修改，即可帮助模型在答案位于后部位置时做出更准确的预测。（3）我们进一步揭示了随机填充（*Random Padding*）能够提升模型在抽取式问答和文档级关系抽取的公开基准数据集上的表现。

2 背景

2.1 任务定义

在抽取式问答中，模型应从给定的段落或文档中提取一个文本片段来回答问题。给定的段落或文档也称为上下文。形式化地，抽取式问答任务可定义为：给定一个问题 Q 和一个上下文 $C = \langle c_0, \cdots, c_{N_c-1} \rangle$，模型应从 C 中选择一个文本片段 $\langle c_s, \cdots, c_e \rangle (0 \le s \le e < N_c)$ 来回答 Q，其中 N_c 是 C 中的词数，s 和 e 分别是答案的起始边界和结束边界。

2.2 研究的模型

预训练语言模型（PLM）通常接收由一系列词元组成的输入。这里，我们以 BERT（Devlin 等人，2019年）为例，介绍如何构造预训练语言模型的输入。按照 BERT 原始的预处理方式，我们使用

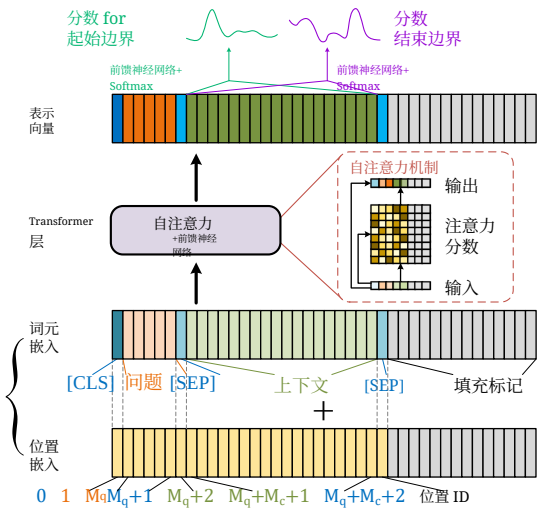

图 1：采用绝对位置嵌入的问答模型处理流程。灰色部分表示与掩码词元对应的权重，其梯度无法进行反向传播。非填充词元的表示向量、嵌入向量和注意力分数以彩色区域表示。

特殊词元 [CLS] 和 [SEP] 来分隔问题和上下文。由于 BERT 必须接收固定长度的输入序列，因此短于最大长度的输入序列会在末尾补上填充标记（[PAD]）。因此，在原版 BERT 中，包含给定问题和上下文的输入序列可以构造为：

[CLS] 问题 [SEP] 上下文 [SEP] 填充标记

形式上，假设问题和上下文在分词后分别包含 M_q 和 M_c 个词元。那么，我们可以将位于第 0 个位置的 [CLS] 视为起始标记，相应地，上下文包含位于 $M_q + 2$ 至 $M_q + M_c + 1$ 位置的词元。为便于说明，我们将非填充词元的数量记为 $m = M_q + M_c + 3$，并将最大输入长度记为 n。因此，应当有 $(n - m)$ 个填充标记。

在采用预训练语言模型的通用抽取式问答框架中（Devlin 等人，2019年；Liu 等人，2019年；Clark 等人，2020年），对于一个包含 m 个非填充标记的输入序列，我们将其表示向量记为 $\{T_i\}_{i=0}^{m-1}$，$T_i \in \mathbb{R}^H$。随后，我们使用两个可训练向量 S，$E \in \mathbb{R}^H$ 来预测答案跨度的起始边界和结束边界。词元 s 作为答案跨度起始边界的概率可写为：

$$\text{Prob}(\text{start} = s) = \frac{\exp(S \cdot T_s)}{\sum_{i=M_q+2}^{M_q+M_c+1} \exp(S \cdot T_i)}.$$

类似地，词元 j 作为结束边界的概率为：

$$\text{Prob}(\text{end} = e) = \frac{\exp(E \cdot T_e)}{\sum_{i=M_q+2}^{M_q+M_c+1} \exp(E \cdot T_i)}.$$

最后，从第 s 个词元到第 e 个词元的答案跨度得分可以简化为 $S \cdot T_s + E \cdot T_e$。抽取式问答模型的最终目标是在满足 $M_q + 2 \leq s \leq e < M_q + M_c + 2$ 约束的条件下，从任意 $\langle s, e \rangle$ 对中选出最大的 $S \cdot T_s + E \cdot T_e$。

图 1 展示了基于预训练语言模型和绝对位置嵌入的抽取式问答框架的流程。如前所述，我们在较短的输入序列后添加填充标记，并在微调期间对这些标记进行掩码。与被掩码词元对应的梯度不会反向传播（图1中的灰色部分）。

2.3 初步实验

我们观察到，问答数据集中的实例具有不同的问题和上下文长度，例如 SQuAD（Rajpurkar 等人，2016年）。经过预处理后，问题与上下文组合而成的词元序列通常短于 BERT 等热门预训练语言模型的最大输入长度。由于掩码词元的梯度无法反向传播，与填充标记对应的位置嵌入不会更新。因此，前部位置的嵌入向量更新次数可能多于后部位置的向量。我们想知道，微调不足是否会导致预训练语言模型表示后部词元的能力较弱。

作为一项初步实验，我们在维基百科领域的 TriviaQA（Joshi 等人，2017）上训练一个 BERT-base 模型。我们记录微调过程中每个位置嵌入被更新的次数。我们发现，第一个位置嵌入（即位置编号 0）在每一步都会更新，而最后一个位置嵌入（即位置编号 511）仅在全部步骤的 62.58% 中更新。我们进一步考察在不同位置预测答案时模型性能的差异。为简洁起见，我们主要关注预测的起始边界 P_s，并将其视为整个答案的位置。

如表1所示，模型在查找出现在位置序列前四分之一处的答案时，F1值可达65.36%，而在最后四分之一处仅为57.03%。这表明模型倾向于在预测词元序列后部位置的答案时犯下更多错误。从前四分之一到最后四分之一，我们发现五次运行的平均F1分数持续下降。尽管这并非严格公平的比较，但仍表明问答模型倾向于对位置嵌入微调次数较少的词元给出更多错误预测。

答案位置	平均 F1 分数
$0 \leq P_s < 128$	65.36 ± 0.49
$128 \leq P_s < 256$	58.45 ± 0.74
$256 \leq P_s < 384$	57.42 ± 0.63
$384 \leq P_s < 512$	57.03 ± 0.51

表 1：按答案位置划分的子集的平均 F1 分数。

3 我们的方法：随机填充

如初步实验所示，当我们在短上下文的实例上训练模型时，前部位置的嵌入更新次数远多于后部位置的嵌入。因此，直观上，我们应使整个位置范围内的更新次数保持均衡，即减少前部位置嵌入的更新次数，并将更多更新分配给后部位置嵌入。

回想一下，在针对抽取式问答对预训练语言模型进行微调时，我们只更新非填充词元的位置嵌入。由于填充标记始终位于整个输入序列的末尾，因此在绝对位置嵌入方案中，这些后部位置嵌入往往会被忽略。如果在微调过程中，填充标记可以随机放置在整个序列中，那么我们可以预期，每个位置嵌入被更新或忽略的机会几乎相同。然而，如果将填充标记插入问题或上下文中，就会改变非填充词元的位置关系，这可能会损害模型性能。因此，我们应保留问题词元和上下文词元，使其各自保持为连续序列。具体而言，在微调过程中，我们提出将随机数量的填充标记移至输入序列前端，如图2所示。这样，非填充词元就会被推向输入序列末尾，从而使后部位置嵌入得到更新。

在微调阶段 与第 2.2 节类似，我们假设输入长度上限为 n 词元。对于包含 m 个非填充词元（包括 `[CLS]`, `[SEP]`, 问题 和 上下文）的输入序列，应当存在 $(n - m)$ 个填充标记

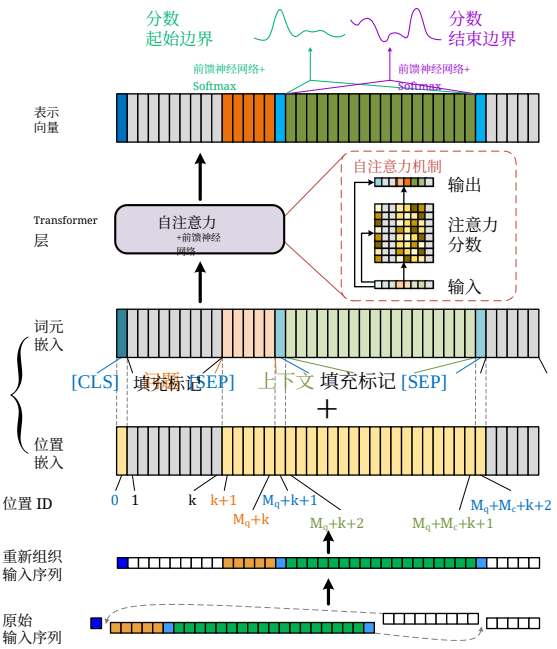

图2：通过随机填充增强问答模型的过程。

在末尾。然后，我们随机选择一个整数 $k \in [0,\ m-n]$，并将 k 个填充标记从输入序列的末尾移动到开头。由此得到的输入序列将为：

$$\texttt{[CLS]}\ \{\boldsymbol{k}\texttt{[PAD]}\}\ \textit{question}\ \texttt{[SEP]}\ \ldots$$
$$\ldots\ \textit{context}\ \texttt{[SEP]}\ \{(\boldsymbol{m-n-k})\texttt{[PAD]}\}$$

注意，我们将 [CLS]保留在 第 0 个位置，作为输入序列的第一个词元。原因是，在广泛采用的 SQuAD 2.0 设置下（Rajpurkar 等人，2018年），问答模型会利用 [CLS] 的分数来判断能否找到给定问题的答案。因此，我们认为[CLS] 在微调期间应放置在固定位置，以减少干扰并加快收敛。

在推理阶段 推理时，我们不会移动填充标记，而是将原始输入序列直接输入训练好的模型：

$$\texttt{[CLS]}\ \textit{question}\ \texttt{[SEP]}\ \ldots$$
$$\ldots\ \textit{context}\ \texttt{[SEP]}\ \{(\boldsymbol{m-n})\texttt{[PAD]}\}$$

请注意，推理期间的输入序列将与基线方法相同。

总而言之，我们只需在微调时调整输入序列中词元的顺序，将随机数量的填充标记移至 [CLS] 与 问题 词元之间的位置。我们的方法不会对现有预训练语言模型的架构做任何修改，因此适用于采用绝对位置嵌入的预训练语言模型。

4 实验

我们在三种不同的预训练语言模型上进行实验，以回答以下问题：(1) 当模型在短上下文上训练、却在长上下文上评估时，随机填充能否提升问答模型的性能？(2) 如果模型在上下文长度分布相似的数据集上训练和评估，随机填充是否仍然有效？(3) 当答案位于上下文末尾时，随机填充是否确实能更显著地提升模型性能？(4) 随机填充如何发挥作用以提升问答模型的性能？

4.1 数据集准备

与第 2.3 节类似，我们在 TriviaQA 的维基百科领域上进行实验（Joshi 等人，2017）。由于我们无法访问官方测试集，因此将官方开发集随机划分为两个大小相等的子集，分别用于验证和测试。

为了更好地了解我们方法的行为，我们研究了在各种训练/测试条件下使用或不使用随机填充的模型，例如，训练集和测试集的上下文长度分布可能不同。具体而言，我们准备了两种不同类型的数据集。

第一种方法是将所有较长的上下文截断为固定长度，例如 100 或 800 个词。我们将截断为固定长度 L 的数据集记作 $\mathcal{D}_L$，并将其对应的训练集/验证集/测试集分别记作 $\mathcal{D}_L^{train}$, $\mathcal{D}_L^{val}$, $\mathcal{D}_L^{test}$。

第二种类型是将较长的上下文截断至一定的长度范围。我们将最小和最大长度限制预先定义为 L_1 和 L_2。对于原始 TriviaQA$_{Wiki}$ 数据集中的每个样本，我们随机选取一个整数 $k \in [L_1,\ L_2]$，并将其上下文截断至 k词。类似地，其训练集/验证集/测试集分别为 $\mathcal{D}_{L_1\sim L_2}^{train}$, $\mathcal{D}_{L_1\sim L_2}^{val}$, $\mathcal{D}_{L_1\sim L_2}^{test}$。

这两种类型的数据集对应两种实际场景。对于自动标注的数据集，如 TriviaQA（Joshi 等人，2017）和 SearchQA（Dunn 等人，2017），其上下文是从与问题相关的多篇文档中收集的，并在创建数据集时截断为固定长度。对于人工标注的数据集，如 SQuAD v1 和 v2（Rajpurkar 等人，

2016年、2018年），人工标注员会为每个段落分别编写问题和答案，其中上下文不会被截断，且长度各不相同。

4.2 实现细节

我们研究了三种不同的抽取式问答模型，分别采用 BERT 的 基础版本（Devlin 等人，2019年）、RoBERTa（Liu 等人，2019年）和 ELECTRA（Clark 等人，2020）。我们在基线设置下对所有模型进行微调，并将原始预处理序列作为输入（其形式与第 2.2 节相同）。对于每个模型，我们采用 Adam（Kingma 和 Ba，2015）作为优化器，并使用不同的随机种子运行五次实验。对于每次基线模型训练，我们还记录每轮训练实例的打乱顺序。

对使用随机填充增强的模型进行微调时，我们采用与基线模型相同的超参数。在五次运行的每一轮中，我们也保持训练实例的顺序与基线设置一致，以消除训练顺序的随机性对模型性能的影响。更多实现细节和数据集统计信息见附录B。

5 主要结果

5.1 短训练、长测试

我们首先考察在训练集中不存在长上下文时，随机填充是否能帮助模型获得更好的后部位置嵌入。具体而言，所有模型都仅使用短上下文实例进行训练（例如，$\mathcal{D}^{train}_{100}$在第 4.1 节中定义的实例），但必须在具有更长上下文的实例上进行评估（例如，$\mathcal{D}^{val}_{100\sim800}$和 $\mathcal{D}^{test}_{100\sim800}$）。这是一个实际且具有挑战性的场景，因为训练数据和测试数据是从不同分布中采样的。

如表 2所示，当模型在包含约 100 个词的上下文（$\mathcal{D}^{train}_{100}$）上进行训练时，随机填充可以使测试集上的 F1 值提高超过 +1%，具体而言，BERT 提高了 +1.07%，ELECTRA 提高了 +1.45%，RoBERTa 提高了 +1.37%。当我们采用 $\mathcal{D}^{train}_{250}$作为训练集时，随机填充分别使 BERT / ELECTRA / RoBERTa 的 F1 值提高了 +1.06% / +0.80% / +0.46%。较短上下文（$\mathcal{D}^{train}_{100}$ 相比于 $\mathcal{D}^{train}_{250}$ 或 $\mathcal{D}^{train}_{400}$）上训练的模型能够从随机填充中获得更大的提升，这并不令人意外，因为后部位置上的更多位置嵌入无法

训练集 Set	模型	验证集 F1	验证集 EM	Test F1	Test EM
$\mathcal{D}^{train}_{100}$	BERT	$58.78_{0.25}$	$52.17_{0.51}$	$58.75_{0.45}$	$51.87_{0.47}$
	+RP	$\mathbf{59.86}_{0.47}$	$\mathbf{54.12}_{0.40}$	$\mathbf{59.82}_{0.40}$	$\mathbf{53.60}_{0.48}$
	ELEC	$61.67_{0.21}$	$55.54_{0.08}$	$61.31_{0.15}$	$54.80_{0.16}$
	+RP	$\mathbf{63.32}_{0.35}$	$\mathbf{57.16}_{0.62}$	$\mathbf{62.76}_{0.44}$	$\mathbf{56.45}_{0.63}$
	RoBE	$61.74_{0.17}$	$55.39_{0.35}$	$61.07_{0.43}$	$54.44_{0.62}$
	+RP	$\mathbf{62.98}_{0.16}$	$\mathbf{56.73}_{0.32}$	$\mathbf{62.44}_{0.35}$	$\mathbf{55.94}_{0.58}$
$\mathcal{D}^{train}_{250}$	BERT	$62.61_{0.13}$	$57.02_{0.32}$	$62.91_{0.15}$	$56.94_{0.27}$
	+RP	$\mathbf{63.46}_{0.32}$	$\mathbf{57.84}_{0.51}$	$\mathbf{63.97}_{0.40}$	$\mathbf{58.01}_{0.49}$
	ELEC	$66.16_{0.28}$	$60.66_{0.57}$	$66.14_{0.28}$	$60.30_{0.55}$
	+RP	$\mathbf{66.88}_{0.14}$	$\mathbf{61.46}_{0.25}$	$\mathbf{66.94}_{0.27}$	$\mathbf{61.41}_{0.34}$
	RoBE	$65.11_{0.20}$	$59.44_{0.39}$	$65.01_{0.40}$	$59.00_{0.37}$
	+RP	$\mathbf{65.60}_{0.20}$	$\mathbf{60.01}_{0.12}$	$\mathbf{65.47}_{0.33}$	$\mathbf{59.63}_{0.40}$
$\mathcal{D}^{train}_{400}$	BERT	$64.03_{0.14}$	$58.48_{0.46}$	$64.79_{0.29}$	$58.96_{0.37}$
	+RP	$\mathbf{64.90}_{0.27}$	$\mathbf{59.55}_{0.35}$	$\mathbf{65.21}_{0.33}$	$\mathbf{59.55}_{0.34}$
	ELEC	$67.76_{0.35}$	$62.61_{0.40}$	**68。07**$_{0.34}$	$62.28_{0.44}$
	+随机填充	**68.09**$_{0.14}$	**62.77**$_{0.25}$	67。94$_{0.21}$	**62.37**$_{0.23}$
	RoBE	$66.67_{0.16}$	$61.25_{0.19}$	$66.34_{0.26}$	$60.75_{0.29}$
	+RP	$\mathbf{66.82}_{0.09}$	$\mathbf{61.38}_{0.16}$	$\mathbf{66.78}_{0.30}$	$\mathbf{61.26}_{0.27}$

表2：在短上下文上训练、但在长上下文（$\mathcal{D}^{val}_{100\sim800}$ 和$\mathcal{D}^{test}_{100\sim800}$）上评估的模型性能。ELECTRA和RoBERTa分别缩写为*ELEC* 和*RoBE*。

在模型使用较短的上下文进行训练时，也需要更新。

5.2 使用相似上下文长度进行训练/测试

我们还想知道，当训练集和测试集的上下文长度分布相似时，随机填充能否帮助这三种预训练语言模型都取得提升。这里，我们将使用第4.1节中创建的第二类数据集。例如，所有模型都将在 $\mathcal{D}^{train}_{800}$上训练，在 $\mathcal{D}^{val}_{800}$ 上验证，并分别在 $\mathcal{D}^{test}_{800}$上测试。

表格3 展示了三种预训练语言模型采用和未采用我们的随机填充策略时的性能。可以看到，随机填充使 BERT 在 +0上提升了 .73% 的 F1值 $\mathcal{D}^{test}_{100\sim800}$，+0使 ELECTRA 提升了 .29%，使 RoBERTa 提升了 .32%。在 $\mathcal{D}_{800}$上进行训练和测试时，模型也能从随机填充中获益，具体而言，+0BERT 的 F1值提升了 .38%，ELECTRA 的 F1值提升了 .22%，+0RoBERTa 的提升为 .26%。

我们发现，三个预训练语言模型在 $\mathcal{D}^{test}_{100\sim800}$ 上的随机填充带来的提升都大于在 $\mathcal{D}^{test}_{800}$上的提升。这与5.1中的结果相似，表明如果训练集包含更多短上下文实例，随机填充就能更有效地提升模型性能。比较三个预训练语言模型在同一数据集上的结果，我们

(a) $\mathcal{D}_{800}$ 上的模型性能

模型	验证集		Test	
	F1	EM	F1	EM
BERT	$60.74_{0.28}$	$54.48_{0.33}$	$61.45_{0.23}$	$55.54_{0.26}$
+RP	$\mathbf{61.18}_{0.09}$	$\mathbf{54.92}_{0.30}$	$\mathbf{61.83}_{0.33}$	$\mathbf{55.98}_{0.35}$
ELECTRA	$64.56_{0.29}$	$58.92_{0.28}$	$65.48_{0.23}$	$59.59_{0.14}$
+RP	$\mathbf{64.77}_{0.17}$	$\mathbf{59.07}_{0.25}$	$\mathbf{65.70}_{0.26}$	$\mathbf{59.81}_{0.38}$
RoBERTa	$63.11_{0.26}$	$56.87_{0.36}$	$63.24_{0.14}$	$57.23_{0.22}$
+RP	$\mathbf{63.24}_{0.17}$	$\mathbf{57.08}_{0.14}$	$\mathbf{63.50}_{0.30}$	$\mathbf{57.44}_{0.27}$

(b) $\mathcal{D}_{100\sim800}$ 上的模型性能

模型	验证集		Test	
	F1	EM	F1	EM
BERT	$63.65_{0.16}$	$58.11_{0.34}$	$64.33_{0.41}$	$58.36_{0.39}$
+RP	$\mathbf{64.40}_{0.08}$	$\mathbf{58.97}_{0.16}$	$\mathbf{65.06}_{0.29}$	$\mathbf{59.28}_{0.24}$
ELECTRA	$67.14_{0.18}$	$61.75_{0.23}$	$67.21_{0.20}$	$61.66_{0.24}$
+RP	$\mathbf{67.35}_{0.13}$	$\mathbf{61.81}_{0.25}$	$\mathbf{67.50}_{0.14}$	$\mathbf{61.84}_{0.26}$
RoBERTa	$65.77_{0.21}$	$60.22_{0.38}$	$66.05_{0.15}$	$60.21_{0.16}$
+RP	$\mathbf{66.11}_{0.28}$	$\mathbf{60.51}_{0.40}$	$\mathbf{66.37}_{0.19}$	$\mathbf{60.50}_{0.35}$

表 3：训练集和测试集的上下文长度分布相似的数据集上的模型性能。

还发现，随机填充对 BERT 性能的提升大于对 RoBERTa 和 ELECTRA 的提升。这可能是因为 BERT 在 90% 的训练步数中使用长度为 128 的序列进行预训练，而 ELECTRA 和 RoBERTa 的所有训练步数都使用包含 512 个词元的序列进行预训练。因此，ELECTRA 和 RoBERTa 在预训练中获得了比 BERT 更好的位置嵌入。此外，ELECTRA 和 RoBERTa 使用更大的批次大小，并在更大的语料库上进行预训练，从而增强了它们生成上下文表示的能力。我们猜测，这使得随机填充在提升 ELECTRA 和 RoBERTa 的性能方面可发挥的空间更小。

6 分析与讨论

实验结果表明，随机填充能够有效提升抽取式问答模型的表现，尤其是在模型使用短上下文训练、但在长上下文上评估时（表2）。考虑到随机填充能够使靠后的位置嵌入获得更多次更新，我们想知道，当模型预测上下文后部的答案时，随机填充是否能带来更大的提升。我们还想知道随机填充如何提升问答模型的表现。

(a) 在 $\mathcal{V}$（$\mathcal{D}_{100\sim800}$的验证集）上的结果

训练集 Set	Seg.	F1 分数 BERT	F1 分数 BERT+RP	ΔF1
$\mathcal{D}_{100}^{train}$	$\mathcal{V}_{100}^{(1)}$	72.03±0.55	71.88±0.47	-0.15
	$\mathcal{V}_{100}^{(2)}$	38.97±1.00	41.88±0.72	**+2.91**
$\mathcal{D}_{250}^{train}$	$\mathcal{V}_{250}^{(1)}$	68.76±0.24	69.32±0.47	+0.56
	$\mathcal{V}_{250}^{(2)}$	36.65±0.65	38.63±1.06	**+1.98**
$\mathcal{D}_{400}^{train}$	$\mathcal{V}_{400}^{(1)}$	67.03±0.19	67.86±0.35	+0.83
	$\mathcal{V}_{400}^{(2)}$	37.19±0.99	38.39±0.91	**+1.21**

(b) $\mathcal{T}$（$\mathcal{D}_{100\sim800}$的测试集）

训练 Set	Seg.	F1 分数 BERT	F1 分数 BERT+RP	ΔF1
$\mathcal{D}_{100}^{train}$	$\mathcal{T}_{100}^{(1)}$	71.92±0.85	71.14±0.54	-0.78
	$\mathcal{T}_{100}^{(2)}$	39.67±0.57	43.43±0.71	**+3.76**
$\mathcal{D}_{250}^{train}$	$\mathcal{T}_{250}^{(1)}$	68.53±0.15	69.48±0.35	+0.95
	$\mathcal{T}_{250}^{(2)}$	39.60±1.00	41.24±1.35	**+1.64**
$\mathcal{D}_{400}^{train}$	$\mathcal{T}_{400}^{(1)}$	67.28±0.50	67.71±0.32	**+0.43**
	$\mathcal{T}_{400}^{(2)}$	41.92±1.32	42.20±2.02	+0.28

表4：在 $\mathcal{D}_{100\sim800}^{val}$ 和 $\mathcal{D}_{100\sim800}^{test}$的每个片段上使用随机填充带来的提升（ΔF1值）。

6.1 答案位置分析

为了研究随机填充带来的改进与答案位置之间的关系，我们将测试集划分为两个分段，$\mathcal{S}$ 即 $\mathcal{S}_X^{(1)}$ 和 $\mathcal{S}_X^{(2)}$。第一个分段，$\mathcal{S}_X^{(1)}$ 包含至少有一个答案位于输入序列前部X 词元中的实例，而所有其他实例则归入第二个分段，$\mathcal{S}_X^{(2)}$。我们在附录C中详细解释了分段划分方法。

我们使用 BERT 训练一个问答模型，其 $\mathcal{D}_X^{train}$上下文被截断至约 X词（100、250 或 400）。随后，我们根据 X将验证集和测试集划分为两个部分，以考察使用或不使用随机填充的模型在答案位置方面是否存在差异。在这里，我们采用 $\mathcal{D}_{100\sim800}^{val}$ 和$\mathcal{D}_{100\sim800}^{test}$ 作为验证集和测试集。为简化表述，我们在表 4中分别将它们记为 $\mathcal{V}$ 和 $\mathcal{T}$。该表展示了随机填充在各个部分带来的性能提升。

我们发现，随机填充能显著提升第二个分段的表现，例如，在 $\mathcal{D}_{100}^{train}$ 上训练时，$\mathcal{T}_{(2)\ 100}$ 的 F1 值提升了 +3.76%。然而，我们惊讶地发现，模型在第一段上的 F1 值提升也不超过 +1%

Segment	训练集 Set	模型	F1分数	ΔF1
$\mathcal{V}^{(2)}_{100}$	$\mathcal{D}^{train}_{100}$	BERT +RP	38.97±1.00 41.88±0.72	**2.91**
	$\mathcal{D}^{train}_{250}$	BERT +RP	45.78±0.46 47.53±0.77	1.75
	$\mathcal{D}^{train}_{400}$	BERT +RP	49.52±0.46 51.20±0.40	1.68
$\mathcal{T}_{(2)\ 100}$	$\mathcal{D}^{train}_{100}$	BERT +RP	39.67±0.57 43.43±0.71	**3.76**
	$\mathcal{D}^{train}_{250}$	BERT +RP	47.38±0.86 49.10±0.70	1.72
	$\mathcal{D}^{train}_{400}$	BERT +RP	51.51±1.13 52.44±0.66	0.93

表5：固定分段划分下的性能。

在第一段上的表现则是如此。在 $\mathcal{D}^{train}_{100}$ 上训练时，经过随机填充增强后，模型在第一段上的表现甚至略有下降。这表明，随机填充的主要作用是提升那些答案全部位于后部位置的实例的模型性能。

我们还注意到，当 X 为 400 时，随机填充对$\mathcal{D}^{test}_{100\sim800}$ 的第二个分段 带来的提升小于对第一段的提升。在 $\mathcal{T}_{(1)\ 400}$ 和 $\mathcal{T}_{(2)\ 400}$中，实例数分别为 4,247 和 462；而在 $\mathcal{T}_{(1)\ 100}$ 和 $\mathcal{T}^{(2)}_{100}$ 中，实例数分别为 2,786 和 1,923。很小的数据集（如 $\mathcal{T}_{(2)\ 400}$）上的评估可能会导致分数波动加剧，这并不奇怪，因此 X =400 的结果成为了异常值。

随机填充与更长上下文的比较 表 2和4 中的结果表明，在更长上下文上训练的模型从随机填充中获得的提升较小。考虑到上述随机填充的主要贡献，我们希望了解，如果模型在更长的上下文上进行训练，其在第二个分段 上的性能提升是否会有所下降。在这里，我们使用具有不同上下文长度（$\mathcal{D}^{train}_{100}$、$\mathcal{D}^{train}_{250}$ 或 $\mathcal{D}^{train}_{400}$）的实例训练 BERT 模型，然后在固定的分段划分（$\mathcal{V}^{(2)}_{100}$ 和 $\mathcal{T}_{(2)\ 100}$）上评估这些模型。

结果如表 5 所示。对于在 $\mathcal{D}^{train}_{100}$ 上训练的模型，随机填充可使 +2.91% F1值和 +3.76% F1值分别在 $\mathcal{V}^{(2)}_{100}$ 和 $\mathcal{T}_{(2)\ 100}$ 上得到提升。当我们使用包含更长上下文的训练集时，提升幅度降至 +0.93%∼+1.75% F1值。这表明，在更长上下文上训练的模型，在后部位置预测答案时获得的提升较小。原因之一可能是，更长的上下文也会使后部位置嵌入得到更多次更新，其作用与随机填充类似。

6.2 随机填充如何提升问答性能?

为了研究随机填充的工作原理，我们考察了两个类别中特定实例的改进效果：(1) 基线模型未能找到正确答案，而我们改进后的模型能够纠正错误预测。(2) 基线模型几乎找到了正确答案，但它选择了真实答案附近的冗余词元，而这些词元随后被我们改进后的模型过滤掉。我们在表6中提供了示例案例。

Type	示例案例
Make 更正	问题：在电视剧《雷鸟神机队》中，帕克是谁的司机? 上下文：帕克.....出现在电影续集中雷鸟神机队出发（1966）.....帕克受雇位于克赖顿-沃德庄园，由 佩内[redacted]担任她的管家和司机.....
Find 精确 弹跳- 达里斯	问题：贝塞mer 流程? 上下文：贝塞麦炼钢法是首个用于大规模生产的低成本工业工艺将[redacted]熔融生铁中炼制出来，早于平炉出现炉底熔炉.....

表 6：我们的方法所纠正的不同类型示例。带下划线的文本是基线模型给出的错误预测。 红色文本是改进模型给出的正确预测。 黄色背景的文本是[redacted]

对于每个实例，我们都会比较基线模型和改进模型给出的预测。如果基线模型的预测所得 F1分数低于改进模型，我们就会比较这两个预测。如果两个序列之间没有共同子序列，我们会认为改进模型选出了正确答案，而基线模型没有。否则，我们会认为改进模型找到了更精确的答案边界。

我们将基于 $\mathcal{D}^{train}_{100}$训练的 BERT 模型作为基线模型。随后，我们通过随机填充或更长的上下文（$\mathcal{D}^{train}_{400}$）来增强模型。所有模型均使用五个不同的随机种子进行训练，并在 $\mathcal{D}^{test}_{100\sim800}$上进行测试。在随机填充带来改进的实例中，36.11% ±2.75%的改进源于找到更精确的边界，其余则是因为纠正了错误答案。然而，对于通过使用更长的上下文进行训练而得到改进的实例，31.73 %±2.59%的改进可归因于更精确的边界。

模型	Natural Questions		HotpotQA		SQuAD 2.0	
	评估 F1	测试 F1	评估 F1	测试 F1	评估 F1	测试 F1
BERT	78.49±0.20	78.18±0.24	75.46±0.12	73.41±0.29	77.69±0.50	77.71±0.55
+RP	**78.86**±0.35	**78.67**±0.33	**76.22**±0.34	**74.23**±0.15	**78.04**±0.41	**78.00**±0.47
ELECTRA	81.30±0.17	80.71±0.18	78.59±0.25	76.55±0.13	**84.44**±0.38	83.85±0.34
+RP	**81.46**±0.13	**80.88**±0.13	**78.71**±0.29	**76.67**±0.28	84.40±0.55	**83.95**±0.34
RoBERTa	80.67±0.15	80.43±0.18	79.63±0.20	76.87±0.28	83.03±0.41	82.85±0.26
+RP	**80.81**±0.22	**80.74**±0.27	**79.73**±0.41	**77.42**±0.26	**83.37**±0.14	**82.99**±0.18

表7：在公开抽取式问答基准数据集上的表现。

从结果来看，我们认为随机填充能够增强问答模型更好地表示相邻词元之间位置信息的能力，从而在答案边界方面带来更大的改进。

模型	测试 F1	测试集 Ign F1
BERT-base	73.57±0.20	72.74±0.19
BERT-base+随机填充	**73.92**±0.09	**73.10**±0.14
BERT-large	76.06±0.11	75.40±0.09
BERT-large+RP	**76.18**±0.10	**75.53**±0.11
RoBERTa-base	75.29±0.19	74.57±0.20
RoBERTa-base+RP	**75.42**±0.23	**74.71**±0.31
RoBERTa-large	77.78±0.24	77.10±0.25
RoBERTa-large+RP	**78.00**±0.25	**77.34**±0.27

表 8：Re-DocRED 上的模型性能。

7 更多基准数据集上的结果

现在，我们继续考察随机填充对更多基准数据集的影响。我们使用各种预训练语言模型训练并评估问答模型，包括 NaturalQuestions（Kwiatkowski 等，2019年）、HotpotQA（Yang 等，2018年）和 SQuAD 2.0（Rajpurkar 等，2018年）。由于这些基准仅提供验证集，我们将原始验证集随机划分为两部分，分别作为我们的验证集和测试集。

如表7所示，我们可以观察到，简单的随机填充使 BERT 在 HotpotQA 上获得了 +0.82% 的 F1 值提升，在 Natu-ralQuestions 上获得了 +0.49% 的 F1值提升，在 SQuAD v2 上获得了 +0.29% 的 F1值提升；而 ELECRTA 和 RoBERTa 的提升仅约为 +0.3% 的 F1值，甚至没有显著提升。这与第5节中的结果类似。由于 ELECTRA 和 RoBERTa 使用更大的批次大小和更大的语料库进行训练，因此留给随机填充用于改进的空间更小。

除了抽取式问答外，我们还研究了文档级关系抽取（RE）任务，在该任务中，模型应提供高质量的

表示以编码局部上下文（Zhang 等人，2021年；Wang 等人，2020年）。如第 6 节所述，随机填充可以增强模型编码相邻词元相对位置的能力，这为表示局部上下文提供了重要信息。本文采用最先进的关系抽取框架 ATLOP（Zhou 等人，2021年），并考察 ATLOP 中作为预训练语言模型组件的 BERT 和 RoBERTa。表 8 展示了模型在 Re-DocRED（Tan 等人，2022年）基准上的模型性能。我们发现，随机填充使 BERT 的 F1 值提升了 +0.35%，使 Ign F1 提升了 +0.36%。结果还表明，随机填充对 BERT 的提升大于对 RoBERTa 的提升，对基础版模型的提升大于对大型模型的提升。

8 结论

在这项工作中，我们提出了一种简单的策略——随机填充，以提升抽取式问答模型的性能，尤其是在模型使用短上下文训练、却在更长的上下文上评估时。我们的方法仅在微调时重新组织输入词元序列，不对预训练语言模型的架构进行任何修改。实验表明，我们的简单方法能够有效提升问答模型在靠后位置预测答案时的表现；若不采用随机填充，这些位置嵌入可能无法得到充分更新。我们还表明，在更多基准和任务中，只要需要对更长上下文进行准确的局部上下文表示，我们的简单策略就能提升预训练语言模型组件的性能。

局限性

本文主要关注随机填充是否以及如何提升模型在抽取式问答任务中的表现。需要注意的是，随机填充并不能提升基于 Transformer 的模型在所有任务上的表现，尤其是使用绝对位置嵌入的模型。例如，我们在句子分类这类任务上进行了实验，例如 Natural

语言推理。我们使用 BERT 在 MNLI（Williams 等人，2018年）上训练了一个文本分类模型。由于 MNLI 中大多数文本的长度不超过 100 个词，因此无需更新第 100 个位置之后的位置嵌入。因此，我们在微调期间将可移动填充标记的最大数量限制为 32。不使用随机填充时，原始 BERT 的准确率为 92.69%±0.09%，而使用随机填充的模型准确率为 92.61%±0.23%。我们发现，随机填充几乎不会对 BERT 在 MNLI 上的表现产生影响。由于 MNLI 是一项文本分类任务，我们应将 `[CLS]` 词元的表示输入分类解码器，而不输入任何其他表示。`[CLS]` 的表示应捕获整篇文本的全局上下文信息，而随机填充的主要作用是增强模型表示局部上下文的能力。因此，随机填充对文本分类无效。

我们还研究了当模型在具有极长上下文的实例上训练时，随机填充是否仍然有效。这里，我们以 NewsQA（Trischler 等人，2017）为例。NewsQA 中超过 30% 的实例具有不少于 1,000 个词的上下文。由于 BERT 只能接收长度为 512 个词元的输入序列，因此这些超长上下文将被划分为多个序列。因此，大多数训练序列都将包含 512 个非填充词元，每个位置嵌入都能够得到适当的微调。我们发现，原始 BERT 模型在 NewsQA 上取得了 70.96%±0.22% 的 F1值，而通过随机填充增强的 BERT 则取得了 70.93%±0.37% 的 F1值。这表明，随机填充对在超长上下文上训练的问答模型没有任何帮助，因为在微调过程中，几乎不存在可以移动的填充标记。

总而言之，我们认为，随机填充应应用于需要模型捕捉局部上下文特征、而非全局特征的任务。对于在超长上下文上训练且位置嵌入均已得到充分微调的模型，随机填充无法发挥提升作用。

参考文献

Steven Bird、Ewan Klein 和 Edward Loper。2009。《使用 Python 进行自然语言处理》。O'Reilly Media Inc.

Kevin Clark、Minh-Thang Luong、Quoc V. Le 和 Christopher D. Manning。2020。Electra：将文本编码器预训练为判别器，而非生成器。载于国际学习表征会议。

Jacob Devlin、Ming-Wei Chang、Kenton Lee 和 Kristina Toutanova。2019年。BERT：用于语言理解的深度双向 Transformer 预训练。载于《2019年计算语言学协会北美分会人类语言技术会议论文集》第1卷（长篇论文与短篇论文），第4171–4186页，美国明尼阿波利斯，计算语言学协会。

Matthew Dunn、Levent Sagun、Mike Higgins、V. Ugur Güney、Volkan Cirik 和 Kyunghyun Cho。2017年。SearchQA：结合搜索引擎上下文增强的新问答数据集。CoRR，abs/1704.05179。

Adam Fisch、AlonTalmor、Robin Jia、Minjoon Seo、Eunsol Choi 和 Danqi Chen。2019年。MRQA 2019年共享任务：评估阅读理解的泛化能力。载于 EMNLP 自然语言处理经验方法会议第二届机器阅读理解（MRQA）研讨会论文集。

Robin Jia 和 Percy Liang。2017年。用于评估阅读理解系统的对抗样例。载于 2017年自然语言处理经验方法会议论文集，第 2021–2031 页，丹麦哥本哈根。计算语言学协会。

Mandar Joshi、Danqi Chen、YinhanLiu、Daniel S. Weld、Luke Zettlemoyer 和 Omer Levy。2020年。SpanBERT：通过表示和预测跨度改进预训练。计算语言学协会会刊，8：64–77。

Mandar Joshi、Eunsol Choi、Daniel Weld 和 Luke Zettlemoyer。2017 年。TriviaQA：用于阅读理解的大规模远程监督挑战数据集。载于第 55 届计算语言学协会年会论文集（第 1 卷：长篇论文），第 1601–1611 页，加拿大温哥华。计算语言学协会。

Diederik P. Kingma 和 Jimmy Ba。2015。Adam：一种随机优化方法。载于第 3 届国际学习表征会议（ICLR 2015）会议论文集。

Miyoung Ko、Jinhyuk Lee、Hyunjae Kim、Gangwoo Kim 和 Jaewoo Kang。2020年。看第一句话：问答中的位置偏差。载于《2020年自然语言处理经验方法会议（自然语言处理经验方法会议）》论文集，第1109–1121页，线上。计算语言学协会。

Tom Kwiatkowski、Jennimaria Palomaki、Olivia Redfield、Michael Collins、Ankur Parikh、Chris Alberti、Danielle Epstein、Illia Polosukhin、Jacob Devlin、Kenton Lee、Kristina Toutanova、Llion Jones、Matthew Kelcey、Ming-Wei Chang、Andrew Dai、Jakob Uszkoreit、Quoc Le 和 Slav Petrov。2019 年。Natural Questions：一项用于问答研究的基准。计算语言学协会会刊，7(0):452–466。

Yinhan Liu、Myle Ott、Naman Goyal、Jingfei Du、Man- dar Joshi、Danqi Chen、Omer Levy、Mike Lewis、Luke Zettlemoyer 和 Veselin Stoyanov。2019年。RoBERTa：一种稳健优化的 BERT 预训练方法。CoRR，abs/1907.11692。

Matthew E. Peters、Mark Neumann、Mohit Iyyer、Matt Gardner、Christopher Clark、Kenton Lee 和 Luke Zettlemoyer。2018年。深度上下文化词表示。载于2018年计算语言学协会北美分会会议论文集：人类语言技术，第1卷（长篇论文），第2227–2237页，新奥尔良州路易斯安那州。计算语言学协会。

Ofir Press、Noah Smith 和 Mike Lewis。2022年。短训练、长测试：带线性偏置的注意力能够实现输入长度外推。载于国际学习表征会议。

Pranav Rajpurkar、Robin Jia 和 Percy Liang。2018年。了解自己不知道什么：SQuAD 中无法回答的问题。载于计算语言学协会第 56 届年会论文集（第 2 卷：短文），第 784–789 页，澳大利亚墨尔本。计算语言学协会。

Pranav Rajpurkar、Jian Zhang、Konstantin Lopyrev 和 Percy Liang。2016年。SQuAD：100,000+ 个用于文本机器阅读理解的问题。载于 2016年自然语言处理实证方法会议论文集，第 2383–2392 页，得克萨斯州奥斯汀。计算语言学协会。

Shimi Salant 和 Jonathan Berant。2018年。用于阅读理解的上下文化词语表示。载于 2018年计算语言学协会北美分会会议：人类语言技术论文集，第 2 卷（短文），第 554–559 页，路易斯安那州新奥尔良。计算语言学协会。

Kazutoshi Shinoda、Saku Sugawara 和 Akiko Aizawa。2022年。看向右侧：缓解抽取式问答中的相对位置偏差。CoRR，abs/2210.14541。

Jianlin Su、Yu Lu、Shengfeng Pan、Bo Wen 和 Yunfeng Liu。2021年。RoFormer：通过旋转位置嵌入增强的 Transformer。CoRR，abs/2104.09864。

Saku Sugawara、Kentaro Inui、Satoshi Sekine 和 AkikoAizawa。2018年。什么让阅读理解问题更容易？ 载于《2018年自然语言处理经验方法会议论文集》，第 4208–4219 页，比利时布鲁塞尔。计算语言学协会。

Yutao Sun、Li Dong、Barun Patra、Shuming Ma、Shao-han Huang、Alon Benhaim、Vishrav Chaudhary、Xia Song 和 Furu Wei。2022年。一种可进行长度外推的 Transformer。

Qingyu Tan、Lu Xu、Lidong Bing、Hwee Tou Ng 和 Sharifah Mahani Aljunied。2022年。重新审视 DocRED——解决关系抽取中的假阴性问题。发表于自然语言处理经验方法会议论文集。

Adam Trischler、Tong Wang、Xingdi Yuan、Justin Har- ris、Alessandro Sordoni、Philip Bachman 和 Ka- heer Suleman。2017年。NewsQA：机器理解数据集。发表于第三届自然语言处理表示学习研讨会论文集，第191–200页，温哥华，加拿大。计算语言学协会。

Ashish Vaswani、Noam Shazeer、Niki Parmar、Jakob Uszkoreit、Llion Jones、Aidan N. Gomez、Łukasz Kaiser 和 Illia Polosukhin。2017年。《Attention is all you need》。载于《第31届神经信息处理系统国际会议论文集》，NIPS'17，第6000–6010页，纽约州雷德胡克，美国。Curran Associates Inc.

Difeng Wang、Wei Hu、Ermei Cao 和 Weijian Sun。2020 年。用于文档级关系抽取的全局到局部神经网络。载于《2020年自然语言处理经验方法会议》（自然语言处理经验方法会议）论文集，第 3711–3721 页，线上。计算语言学协会。

Adina Williams、Nikita Nangia 和 Samuel Bowman。2018年。通过推理实现句子理解的广覆盖挑战语料库。载于2018年计算语言学协会北美分会会议论文集：人类语言技术，第1卷（长篇论文），第1112–1122页，新奥尔良，路易斯安那州。计算语言学协会。

Zhilin Yang、PengQi、Saizheng Zhang、Yoshua Bengio、William Cohen、Ruslan Salakhutdinov 和 Christo- pher D. Manning。2018年。HotpotQA：一个用于多样化、可解释多跳问题解答的数据集。载于《2018年自然语言处理经验方法会议论文集》，第 2369–2380 页，比利时布鲁塞尔。计算语言学协会。

Ningyu Zhang、Xiang Chen、Xin Xie、Shumin Deng、Chuanqi Tan、MoshaChen、Fei Huang、Luo Si 和 Huajun Chen。2021年。将文档级关系抽取视为语义分割。载于会议论文集

载于第三十届国际人工智能联合会议、IJCAI-21，第3999–4006页。国际人工智能联合会议组织。主会场。

Wenxuan Zhou、Kevin Huang、Tengyu Ma 和 Jing Huang。2021年。采用自适应阈值化和局部上下文池化的文档级关系抽取。载于《AAAI 人工智能会议论文集》。

PilotExperiments 的设置

在试点实验中，我们研究了在 维基百科 领域的 TriviaQA（Joshi 等人，2017）数据集上训练的原始 BERT-base 模型。与开放域问答基准 MRQA（Fisch 等人，2019）类似，我们将原始数据集转换为 SQuAD 格式（Rajpurkar 等人，2016），并通过 [1]提供的脚本，将较长的文档截断为 800 个词 Joshi 等人(2017)。由于 BERT 接收的输入序列长度不能超过 512，因此较长的上下文会被划分为多个序列。因此，训练集中的每个实例都会被划分为多个分词序列，且最后一个序列的长度会远小于 512 个词元。因此，在微调过程中，第一个位置嵌入在每一步都会得到更新，而最后一个位置嵌入仅能在所有步骤中的 62.58% 中得到更新。我们基于 BERT-base 对问答模型进行 4 轮微调，批次大小为 16，学习率为 3×10^{-5}。

我们在验证集上评估模型，并收集每个实例的预测答案位置。随后，根据答案位置将这些实例划分为四个子集，并分别计算每个子集的平均 F1 分数。

此外，值得注意的是，答案位置偏差也会影响模型的表现（Ko 等人，2020年；Shinoda 等人，2022年）。例如，如果训练集中的所有答案都标注在每个上下文的第一句话中，问答模型也会学会在第一句话中给出预测。

为消除答案的位置偏差，我们重新标注训练集中的答案片段：对于给定的训练样本，如果答案在上下文中出现多次，我们会从所有出现位置中随机选择一个作为最终标注。这样一来，训练集中的答案便倾向于均匀分布在整个上下文中。

[1]https://github.com/mandarjoshi90/triviaqa/blob/master/utils/convert_to_squad_format.py

B 更多实现细节和数据集统计

B.1 实现细节

在第 5 节中，我们使用批次大小为 16 的 $\mathcal{D}_{800}^{train}$、$\mathcal{D}_{100\sim800}^{train}$、$\mathcal{D}_{100}^{train}$、$\mathcal{D}_{250}^{train}$ 或 $\mathcal{D}_{400}^{train}$训练模型。我们采用 Adam（Kingma 和 Ba，2015）作为优化器。在每项实验中，我们每训练 1000 步就在验证集上评估一次模型性能，并保存最佳检查点用于测试。每项实验运行五次，然后报告平均分数及标准差。对于所有实验，我们从以下选项中选取最佳超参数组合：

- 学习率： $\{2 \times 10^{-5}, 3 \times 10^{-5}\}$;
- 训练轮数：{2, 3, 4, 6, 8};

B.2 数据集统计

我们将原始 Trivia 数据集中的实例Joshi 等人，2017截断至不同长度，以研究随机填充能否提升问答模型在上下文长度分布各异的数据集上的表现。

数据集	划分	平均长度 问题数	平均长度 上下文数	数量 实例数
$\mathcal{D}_{800}$	训练集	17.47	746.32	110,647
	Val	17.18	732.97	4,924
	Test	17.24	732.58	4,925
$\mathcal{D}_{100\sim800}$	训练集	17.47	275.17	110,647
	Val	17.20	316.58	4,706
	Test	17.14	317.14	4,709
$\mathcal{D}_{100}$	训练集	17.47	98.03	110,647
$\mathcal{D}_{250}$	训练集	17.47	242.34	110,647
$\mathcal{D}_{400}$	训练集	17.47	384.07	110,647

表9：第5节中采用的所有数据集的统计信息。

由于我们主要关注问答模型能否从完整上下文中找到答案跨度的精确边界，因此我们从验证集和测试集中移除了无法作答的实例。然而，为了确保所有训练集规模相同，我们没有从训练集中移除任何实例。我们使用NLTK 的 _word tokenize函数（Bird 等人，2009）将问题和上下文拆分为词。然后，我们可以确定一个问题或上下文包含多少个词。数据集统计信息见表9。

C 答案位置分析的片段划分

为了研究随机填充带来的改进与答案位置之间的关系，我们将测试集划分为两个分段。第一段由答案位于上下文前部位置的实例组成。相应地，第二个分段由答案全部位于上下文末尾的实例组成。

我们注意到，在同一上下文中可能存在多个答案标注，只要模型能够找到答案的任意一处提及，就会被视为预测正确。因此，我们对数据集 $\mathcal{S}$ 的划分具体如下：

- 我们首先对答案位置进行划分，记作 X。
- 对于数据集 $\mathcal{S}$ 中的每个实例，如果至少存在一个答案提及，其位置位于 0 和 X 之间，则该实例属于 第一段，记作 $\mathcal{S}_{X}^{(1)}$ 。
- 第二个分段 包含 $\mathcal{S}$ 的其余实例。在这些实例中，所有答案提及都位于 $X+1$ 和 512 之间。同样，我们将此分段记为$\mathcal{S}_{X}^{(2)}$ 。

例如，我们按照第 100 个位置对测试集进行划分。假设某个实例的答案在上下文中出现了两次，分别位于位置 99 和 200，那么该实例应属于 第一段。这是因为至少有一个答案标注位于 0 到 100 之间。然而，如果这两个答案标注分别位于位置 101 和 200，那么该实例将属于第二个分段，因为模型必须从后面的词元中寻找答案。

D 低资源场景

我们认为，随机填充可以让位置嵌入的后部得到更多次更新。然而，如果模型是在固定的数据集上训练固定轮次，那么所有位置嵌入的总更新次数就是确定的，这意味着采用随机填充后，前部位置嵌入的更新次数会减少。我们想知道，在低资源场景下，随机填充是否仍然有效，因为此时所有位置嵌入可能都无法得到充分训练。

在这种情况下，随机填充可能会降低模型在前部位置预测答案时的性能，进而可能损害模型在整个数据集上的总体性能。

我们随机选取 $\mathcal{D}_{100}^{train}$的子集来训练基于 BERT-base 的问答模型。我们构造了不同程度的低资源场景，其中训练集分别包含原始训练集的 5%、10%、25% 和 50% 的实例。与上述实验类似，我们使用 Adam（Kingma 和 Ba，2015）作为优化器，学习率设为 3e-5。我们使用三个不同的随机种子为低资源数据集抽样实例，然后分别在这些数据集上进行实验。

与第2.2节类似，我们假设预处理后问题中存在M_q 个词元，上下文中存在 M_c 个词元。输入序列中还存在一个 [CLS] 词元和两个 [SEP] 词元。如果预训练语言模型接收到包含 n 个词元的输入序列，我们应在上下文后添加 $n-M_q-M_c-3$ 个填充标记。在训练经过随机填充增强的模型时，我们随机选取 $k \in [0, n-M_q-M_c-3]$，并将 k 个填充标记移动到 [CLS] 与问题之间的位置。在低资源场景中，我们认为移动的填充标记数量应限制为 K，以保证答案位于前部位置时的模型性能。因此，随机填充可以修改为：训练时，我们选取$k \in [0, \min\{K, n-M_q-M_c-3\}]$，然后将 k 个填充标记移动到问题词元的前面。

我们通过实验将 K 设为64、128、192或256。表10 展示了使用不同随机种子抽样得到的低资源数据集上训练的模型结果。结果表明，训练集规模越小，就越应该适当减少移动填充标记的最大数量。例如，对于使用随机种子=42抽样得到的数据集，当模型使用 5%的实例进行训练时，具有 $K=64$的随机填充模型性能最佳，而具有 $K=128$ 的模型性能排名第二。

规模	K	评估 F1	测试 F1	Δ测试 F1
		(a) 使用随机种子进行采样=42		
5%	0	$47.49_{1.23}$	$47.17_{1.26}$	–
	64	$48.72_{0.86}$	$48.94_{0.86}$	+1.77
	128	$49.16_{0.69}$	$48.86_{0.60}$	+1.69
	192	$48.82_{0.84}$	$48.65_{0.71}$	+1.48
	256	$48.89_{0.72}$	$48.84_{0.81}$	+1.67
	512	$48.89_{0.76}$	$48.62_{0.84}$	+1.45
10%	0	$52.50_{0.33}$	$52.29_{0.65}$	–
	64	$53.04_{0.48}$	$52.90_{0.44}$	+0.61
	128	$53.19_{0.16}$	$52.72_{0.34}$	+0.43
	192	$53.03_{0.19}$	$52.97_{0.54}$	+0.68
	256	$53.22_{0.33}$	$52.81_{0.69}$	+0.52
	512	$53.10_{0.37}$	$53.28_{0.54}$	+0.99
25%	0	$56.30_{0.36}$	$56.39_{0.65}$	–
	64	$56.82_{0.16}$	$56.89_{0.41}$	+0.50
	128	$56.96_{0.33}$	$57.09_{0.31}$	+0.70
	192	$56.89_{0.26}$	$57.00_{0.33}$	+0.61
	256	$56.84_{0.13}$	$57.16_{0.52}$	+0.77
	512	$57.06_{0.45}$	$57.30_{0.29}$	+0.91
50%	0	$59.37_{0.16}$	$59.89_{0.41}$	–
	64	$60.09_{0.09}$	$60.24_{0.52}$	+0.35
	128	$60.22_{0.19}$	$60.45_{0.26}$	+0.56
	192	$60.26_{0.30}$	$60.29_{0.40}$	+0.40
	256	$60.32_{0.14}$	$60.61_{0.35}$	+0.72
	512	$60.21_{0.17}$	$60.57_{0.51}$	+0.68

K	评估 F1	测试 F1	Δ测试 F1
	(b) 使用随机种子=43进行采样		
0	$47.18_{0.96}$	$46.82_{1.03}$	–
64	$48.26_{0.78}$	$48.04_{0.95}$	+1.22
128	$48.63_{0.61}$	$48.17_{0.95}$	+1.35
192	$48.58_{0.48}$	$48.62_{1.08}$	+1.80
256	$48.70_{0.66}$	$48.71_{0.61}$	+1.89
512	$48.77_{0.59}$	$48.71_{0.57}$	+1.89
0	$52.27_{0.74}$	$52.23_{0.98}$	–
64	$52.93_{0.50}$	$52.89_{0.48}$	+0.66
128	$52.99_{0.19}$	$52.91_{0.41}$	+0.68
192	$52.74_{0.46}$	$52.46_{0.32}$	+0.23
256	$53.16_{0.27}$	$53.16_{0.39}$	+0.93
512	$53.32_{0.49}$	$53.10_{0.34}$	+0.87
0	$56.38_{0.30}$	$56.44_{0.21}$	–
64	$56.65_{0.36}$	$56.85_{0.57}$	+0.41
128	$56.95_{0.22}$	$56.83_{0.32}$	+0.39
192	$56.86_{0.39}$	$56.94_{0.40}$	+0.50
256	$56.90_{0.30}$	$56.94_{0.28}$	+0.50
512	$56.96_{0.17}$	$56.92_{0.27}$	+0.48
0	$59.11_{0.28}$	$59.41_{0.09}$	–
64	$59.47_{0.23}$	$59.81_{0.17}$	+0.40
128	$60.05_{0.26}$	$60.20_{0.32}$	+0.79
192	$59.79_{0.09}$	$60.13_{0.61}$	+0.72
256	$59.88_{0.28}$	$59.86_{0.30}$	+0.45
512	$59.81_{0.18}$	$59.94_{0.54}$	+0.53

K	评估 F1	测试 F1	Δ测试 F1
	(c) 使用随机种子=44进行采样		
0	$47.20_{0.63}$	$46.63_{0.46}$	–
64	$48.57_{0.69}$	$47.86_{0.60}$	+1.23
128	$48.56_{0.61}$	$48.37_{0.74}$	+2.03
192	$48.64_{0.60}$	$47.69_{0.87}$	+1.06
256	$48.43_{0.58}$	$47.83_{0.65}$	+1.20
512	$48.67_{0.68}$	$48.01_{0.75}$	+1.38
0	$51.92_{0.52}$	$51.93_{0.74}$	–
64	$52.46_{0.26}$	$52.13_{0.55}$	+0.20
128	$52.49_{0.37}$	$52.36_{0.63}$	+0.43
192	$52.59_{0.27}$	$52.58_{0.24}$	+0.65
256	$52.60_{0.62}$	$52.83_{0.60}$	+0.90
512	$52.61_{0.26}$	$52.53_{0.34}$	+0.60
0	$56.63_{0.38}$	$56.68_{0.42}$	–
64	$56.94_{0.25}$	$56.94_{0.49}$	+0.26
128	$57.17_{0.38}$	$56.99_{0.26}$	+0.37
192	$56.94_{0.52}$	$56.82_{0.29}$	+0.14
256	$57.17_{0.23}$	$56.79_{0.23}$	+0.11
512	$57.11_{0.45}$	$57.17_{0.43}$	+0.49
0	$59.25_{0.43}$	$59.52_{0.52}$	–
64	$59.69_{0.16}$	$59.78_{0.23}$	+0.26
128	$59.71_{0.40}$	$59.91_{0.26}$	+0.39
192	$59.81_{0.34}$	$59.74_{0.35}$	+0.22
256	$59.96_{0.27}$	$60.05_{0.31}$	+0.53
512	$59.91_{0.27}$	$60.07_{0.25}$	+0.55

表10：不同规模的低资源数据集上的模型性能。 $K = 0$ 表示未使用随机填充的基线模型的性能，$K = 512$ 表示使用原始随机填充的模型性能。