浙江工业大学

本科毕业设计说明书（论文）

Undergraduate International Students' Graduation Project Report (Thesis)

外文翻译

Foreign Language Translation

译文题目：Fantastically Ordered Prompts and Where to Find Them: Overcoming Few-Shot Prompt Order Sensitivity

译文出处：arXiv preprint arXiv:2104.08786 (2021)

学　　院：计算机科学与技术学院、软件学院

专　　业：软件工程（中外合作办学）

班　　级：2020 软件工程（中外合作办学）01

学　　号：202003340133

学生姓名：庄景文

指导老师：李小薪

提交日期：2024 年 3 月

神奇顺序提示在哪里：克服对少量提示顺序的敏感性

Yao Lu†　Max Bartolo†　Alastair Moore‡　Sebastian Riedel†　Pontus Stenetorp†
†University College London　‡Mishcon de Reya LLP
{yao.lu,m.bartolo,s.riedel,p.stenetorp}@cs.ucl.ac.uk
alastair.moore@mishcon.com

摘要：在只有少量训练样本的情况下，如 GPT-3 这样的大型预训练语言模型显示出了与完全监督的、微调过的大型预训练语言模型相当的结果。我们展示了样本提供的顺序可能导致接近最先进水平和随机猜测表现之间的差异：本质上，一些排列是“奇妙的”，而有些则不是。我们详细分析了这一现象，确定了：它在不同模型大小中普遍存在（即使是最大的当前模型），它不关联于特定的样本子集，以及一个给定的好的排列对一个模型是好的并不意味着它能转移到另一个模型。虽然可以使用开发集来确定哪些排列是有效的，但这将偏离真正的少量样本设置，因为它需要额外的标注数据。相反，我们利用语言模型的生成性质构建一个人工开发集，并基于这个集合上候选排列的熵统计，我们识别出了表现良好的提示。我们的方法使得 GPT 系列模型在十一个不同的已建立文本分类任务上平均提高了 13%。

1　引言

大型预训练语言模型（PLMs，Devlin et al., 2019; Peters et al., 2018; Raffel et al., 2020; Liu et al., 2019; Yang et al., 2019; Radford et al., 2019）在配以适当的文本背景时，表现出了显著的性能。例如，当以一个长文档和一个“TL;DR:”标记为条件时，它们可以生成该文档的摘要；当提供一个部分问题（“相对论是由__发展的”）时，它们可以生成正确的答案。最引人注目的是，当以非常少的训练示例作为上下文时，它们产生的文本分类结果可以与完全监督的模型相匹配。这种类型的少量样本设置通常被称为“上下文学习”（Brown et al., 2020）。

上下文学习的一个核心组成部分是作为上下文的基于文本的提示。构建一个提示需要：（i）使用模板进行文本线性化；以及（ii）训练样本的拼接（参见表 1 中的一个例子）。已经确定模板的结构对性能有很大的影响（Shin et al., 2020；Gao et al., 2020；Schick and Schütze, 2020；Jiang et al., 2020）。然而，据我们所知，没

有工作研究样本排序对于上下文学习性能的影响。

	Example
training set	(the greatest musicians, 1) (redundant concept, 0)
linearization	Review: the greatest musicians. Sentiment: positive Review: redundant concept. Sentiment: negative
concatenation	Review: the greatest musicians. Sentiment: positive. Review: redundant concept. Sentiment: negative *OR* Review: redundant concept. Sentiment: negative. Review: the greatest musicians. Sentiment: positive

表 1 提示构建的流程

或许有悖直觉的是，我们发现正确的样本顺序可以像正确的模板一样产生巨大的差异，如图 1 所示，有些排列的表现可以与监督训练相媲美（准确率超过 85%），而其他排列的表现接近随机（约 50%）。这种顺序敏感性在模型中是普遍存在的，并且尽管增加模型大小在某种程度上可以解决这个问题，但对某些文本分类任务（图 1 中的 Subj）仍然存在问题，即使是拥有数十亿参数的模型。

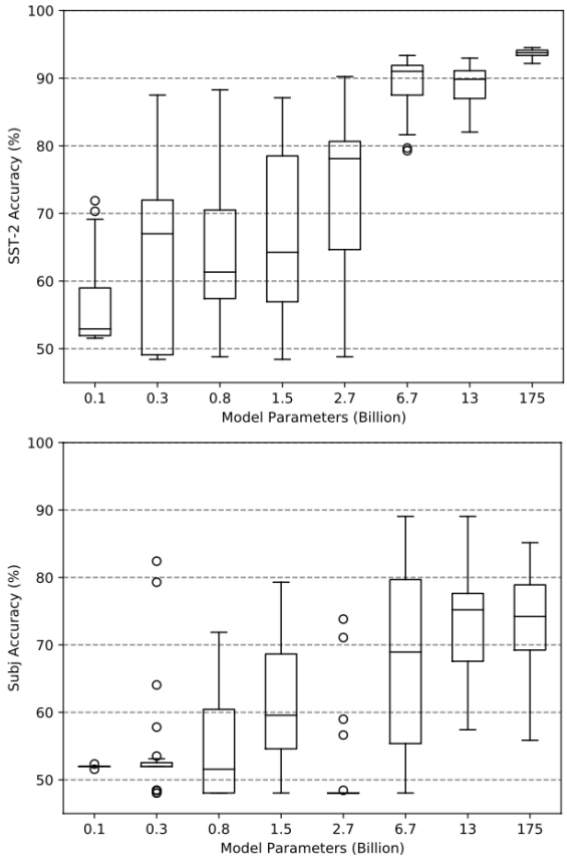

图 1 针对 SST-2 和 Subj 数据集，不同规模的 GPT 系列模型（GPT-2 和 GPT-3）在 24 个不同取样顺序下的四次拍摄性能。

在我们的分析中，我们发现没有一个普遍的规则可以定义哪些样本顺序是有效的，它们也不能在不同的模型大小和任务之间转移。在一个完全监督的设置中，我们可以依赖一个开发集来选择样本顺序。然而，这在一个真正的少量样本设置中并不可取，因为开发集的规模非常有限，甚至是不可用的（Perez et al., 2021）。相反，我们利用语言模型的生成性质构建一个未标记的人工开发集，并将其称为探针集。由于探针集是未标记的，我们使用预测标签分布统计，并提出基于熵的指标来衡量候选提示的质量。实验结果显示，我们可以在不同大小（四个数量级）的 PLMs 上平均实现 13%的相对提高，覆盖十一个不同的已建立文本分类任务。

总结我们的贡献如下：

1. 我们研究了上下文学习中的顺序敏感性，我们显示这对于预训练语言模型在少量学习中的成功至关重要。
2. 我们提出了一种简单的、基于生成的探针方法来识别性能良好的提示，而不需要额外的数据。
3. 我们的探针方法普遍适用并有效，适用于不同大小的预训练语言模型和不同类型的数据集——平均而言，实现了 13%的相对改进。

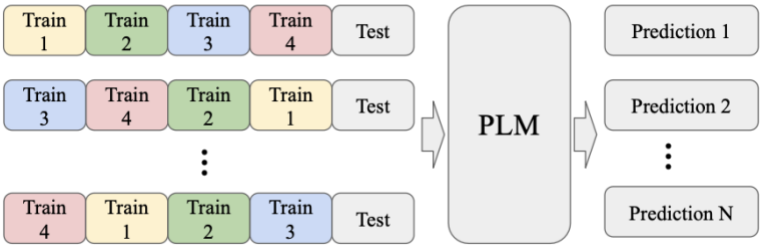

图 2 上下文学习设置中的训练样本排列。训练样本及测试数据的串联将分类任务转变为序列生成任务。

2 顺序敏感性与提示设计

在本节中，我们研究排列性能与各种因素之间的关系。为了便于可视化，我们使用 SST-2 数据集的四个样本的固定随机子集，并考虑所有 24 种可能的样本顺序排列。这种设置如图 2 所示。我们还测试了五组随机选取的示例，并在实验部分（第 5 节）总结了方差统计数据。

虽然增加模型大小有益，但不能保证低方差。我们评估了四种不同大小的 GPT-2（0.1B-1.5B）和 GPT-3（2.7B-175B）的顺序排列。如我们在图 1 中观察到

的，模型可以获得显著的少量样本性能。我们看到，GPT2-XL（1.5B）模型甚至可以在仅有四个样本的情况下超过 90%的准确率。这一结果与在超过 60,000 个样本上训练的监督模型相媲美。然而，不同排列的性能变化仍然是一个大问题，特别是对于“较小”的模型。同一模型在给定一种样本顺序时可能表现几乎完美，但换成另一种顺序则可能回到与随机基线相当的水平。尽管增加模型大小（数量级增加几个）有时可以缓解这个问题，但它仍然不能完全解决问题（特别是如果我们考虑除 SST-2 之外的任务）。相比之下，监督式微调方法的不同初始化通常在测试集性能上的标准偏差小于 1%（Gao et al., 2020）。

增加训练样本并不能显著减少方差。为了进一步探索少量样本提示的顺序敏感性，我们增加了训练样本的数量，然后抽取最多 24 种不同的排序。我们使用 GPT2 系列模型进行这项实验。在图 3 中，我们可以观察到增加训练样本数量导致性能提高。然而，即使在大量样本的情况下，高水平的方差仍然存在，甚至可能增加。基于此，我们得出结论，顺序敏感性可能是上下文学习的一个基本问题，无论训练样本的数量如何。

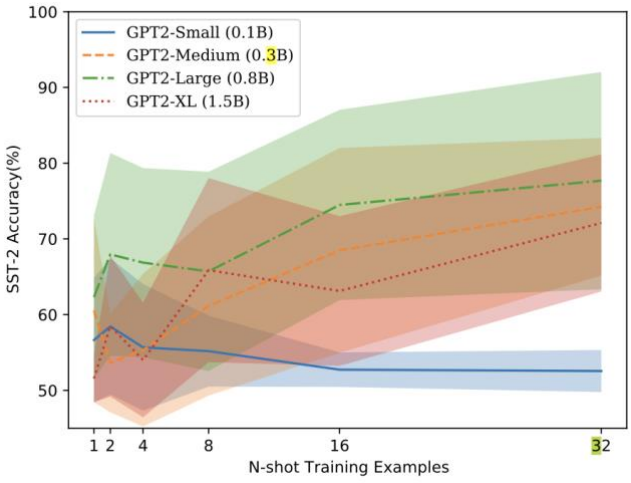

图 3 使用不同数量的训练样本的顺序敏感性

性能良好的提示在模型之间不可转移。我们发现，通过将底层模型从 GPT2-XL（1.5B）更改为 GPT2-Large（0.8B），特定排列的性能可能从 88.7%下降到 51.6%。这表明，对于一个模型表现良好的特定排列，并不意味着它也会为另一个模型提供好的结果。为验证这一假设，我们使用四个样本的所有可能的顺序排列作为提示——总共 24 种。然后，我们对每种提示条件下的不同模型进行预测，并计算得分之间的成对 Spearman 等级相关系数。这些结果如图 4 所示。

如果存在表现良好的提示的共同模式，那么我们应该能够观察到跨模型的高相关性。然而，排列的行为在同一模型的不同大小之间似乎是随机的，例如，175B 和 2.7B 模型之间的相关性仅为 0.05，这意味着对于 2.7B 模型的良好排列，并不能保证它也会为 175B 模型带来良好的性能。

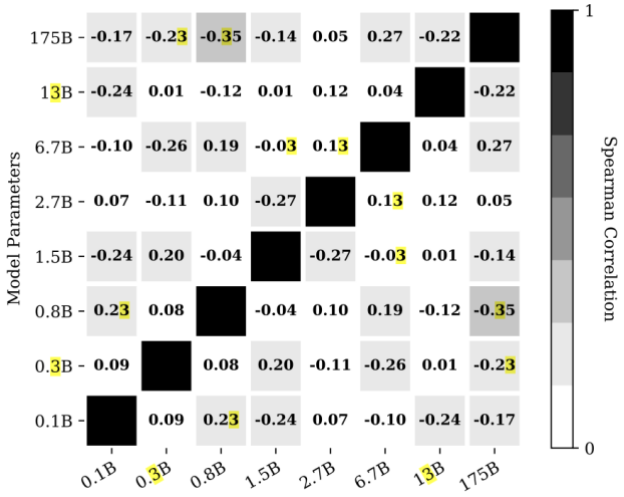

图 4 不同模型间训练样本排列性能的相关性

性能良好的标签排序在模型之间不一致。除了训练示例排序之外，我们还探索了训练提示的标签排序。我们使用上述完全排列的所有模式——六种不同的标签模式。然后我们按照前一段描述的方式计算不同模型之间的成对 Spearman 相关系数。如图 5 所示，标签排序的行为再次似乎是随机的，跨不同大小的同一模型。因此，不可能识别出一个表现良好的标签排序。

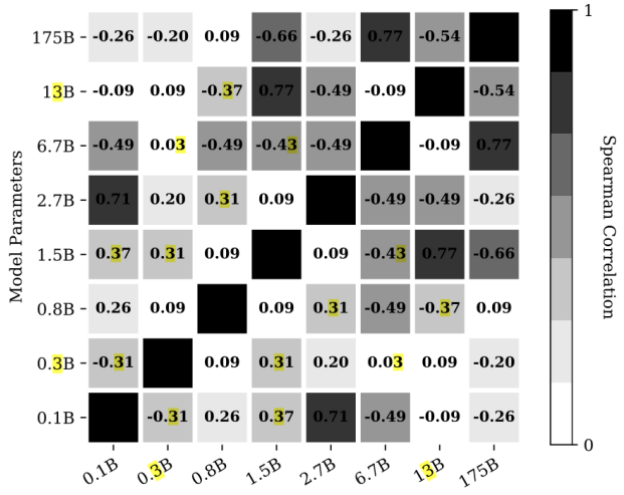

图 5 不同模型间训练标签模式排列性能的相关性

不良提示的退化行为。我们对表现良好和表现不佳的提示进行错误分析，观

察到大多数失败的提示都遭受了高度不平衡的预测标签分布的困扰（见图 6 左图）。解决这一问题的一个直观方法是通过校准输出分布，类似于 Zhao 等人（2021）的方法。然而，我们发现，尽管校准可以显著提高性能，但方差仍然很高（见图 6 右图）。

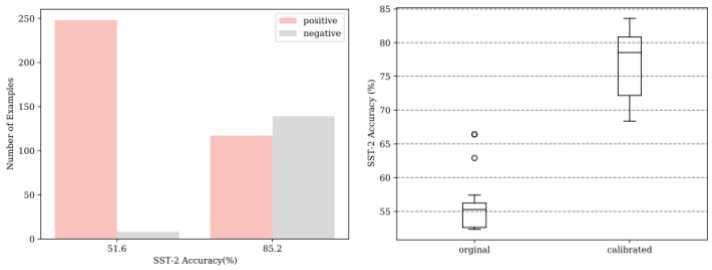

图 6 左：不同提示下预测的 SST-2 标签分布。右：GPT2-XL（1.5B）上所有可能排列的 2-shot 校准性能（Zhao 等，2021）。

3 方法论

前一节表明，提示的顺序可能对性能产生实质性的影响，对于同一模型的同一提示的不同排序，有些可能导致随机性能，而其他“更好”的排序则能提供与监督方法竞争的性能。这表明，选择提示顺序的不同方式可能有助于提高性能，但挑战在于如何自动且不需额外标签（例如，开发集）来做到这一点。

因此，在本节中，我们探索以下问题：“我们如何自动生成‘探针集’以找到性能良好的提示顺序？”我们的方法包括：（i）对随机选定的训练样本集，使用该集的每种可能的排序排列作为候选；（ii）通过使用所有候选提示作为上下文来查询语言模型，构建探针集；以及（iii）使用这个探针集通过一个探测度量来识别最佳排序。

3.1 从语言模型中采样以构建探针集

我们提出了一种自动构建“探针集”的简单方法，即直接从语言模型本身进行采样。这种方法使我们能够自动生成探针集，而无需访问任何额外的数据。具体来说，给定一组训练样本 $S = \{(x_i, y_i)\}$，其中 $i = 1, \ldots, n$，x_i 和 y_i 分别表示第 i 个训练样本的句子和标签。然后我们定义一个转换 T，将每个样本映射到自然语言空间，使得 $t_i = T(x_i, y_i)$。因此，t_i 是使用 T 定义的模板的第 i 个训练样本的文本序列。在本工作中，我们使用一个简单的转换函数 T，使得 $T(x_i, y_i) = \text{input:}x_i\ \text{type:}y_i$。这将每个样本转换成一个标准格式的句子，将集合

中的每个元素线性化到自然语言空间中定义的 $S' = \{t_i\}, i = 1, \ldots, n$ 。

然后我们定义一个 n 个训练样本的完全排列函数组 $F = \{f_m\}, m = 1, \ldots, n!$ ，其中每个函数 f_m 将 S' 作为输入，并输出 c_m ：一个独特排列的串联。在我们的案例中，随机采样四个训练样本可以产生多达 24 种可能的排序排列的转换样本。

对于每个提示候选 c_m ，我们随后从语言模型中采样以获得探测序列 $g_m \sim P(\cdot \mid c_m; \theta)$ ，其中 θ 表示预训练语言模型的参数。我们在生成由模板定义的特殊句子结束标记时停止从语言模型解码，或者达到生成长度限制。我们的探针集构建方法如图 7 所示，其目标是生成一个与训练样本分布相似的探针集。

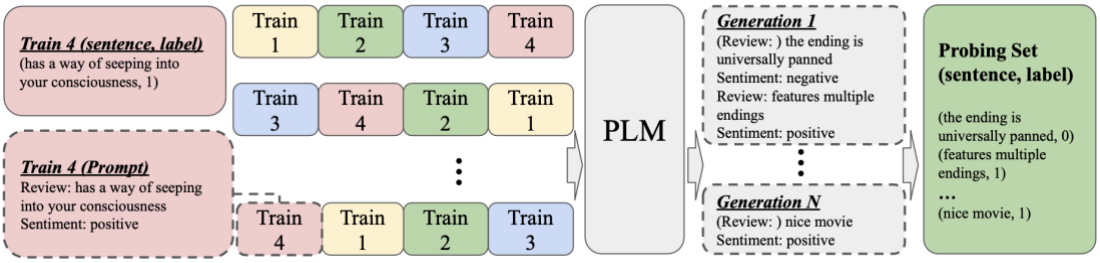

图 7 我们的探针集构建方法，展示了随机选定的训练样本的各种可能的排序排列，每种排列的结果生成，以及它们的串联成一个探针集。请注意，我们丢弃了生成的标签，因为这些生成的标签没有正确性保证。

我们对所有可能的提示排序排列运行此采样过程，并从中提取探针样本 $T^{-1}(g)$ 。然后将提取的样本聚集在一起形成探针集 $D = T^{-1}(g_1) \oplus \ldots \oplus T^{-1}(g_{n!})$ 虽然探针集包含每个句子的预测标签，但这些标签的有效性无法保证。因此，我们在探针集中丢弃它们，因为我们只对从语言模型中采样的与输入分布相对应的探针感兴趣。

一旦我们为给定的样本集构建了探针集，我们现在可以使用这个探针集来识别特定样本集的最佳可能提示排序。这里，我们探索两种方法：全局熵（GlobalE）和局部熵（LocalE）。

全局熵（GlobalE）的动机是识别特定样本排序的提示，这些提示避免了极端不平衡的预测问题（正如我们之前确定的，这是非表现良好提示的关键问题）。我们如下计算在上下文 c_m 下数据点 (x_i', y_i') 的预测标签 $\widehat{y_{i,m}}$ ：

$$\widehat{y_{i,m}} = \arg\max_{v \in V} P\left(v \mid c_m \oplus T(x_i'); \theta\right)$$

对于每个标签 $v \in V$ （其中 V 表示目标标签集），我们计算在探针集上的标签概率：

$$p_{v,m} = \frac{\sum_i 1_{\{\widehat{y_{i,m}}=v\}}}{|D|}$$

然后我们使用预测的类别标签熵作为 c_m 的全局熵分数：

$$\text{GlobalE}_m = -\sum_{v \in V} p_{v,m} \log p_{v,m}$$

局部熵（LocalE）的动机是，如果模型对所有探测输入过于自信，则可能意味着模型表现不佳。至少，它是校准不良的，这也可能是模型无法适当区分类别的一个迹象。类似于全局熵的计算，我们计算在上下文 c_m 下，数据点 (x_i', y_i') 对目标标签 $v \in V$ 的预测概率：

$$p_{v,i,m} = P_{(x_i',y_i')\sim D}(v \mid c_m \oplus T(x_i'); \theta), v \in V$$

然后我们计算每个数据点的平均预测熵作为局部熵分数：

$$\text{LocalE}_m = \frac{1}{|D|} \sum_i \sum_{v \in V} -p_{v,i,m} \log p_{v,i,m}$$

现在我们已经有了一种方法来评分每个提示排序，根据其对探针集的影响，我们可以分别根据全局熵或局部熵对每个提示排序进行性能排名。

4 实验设置

我们使用四种不同大小的 GPT-2（Radford et al., 2019）（0.1B、0.3B、0.8B、1.5B 参数）和两种大小的 GPT-3（Brown et al., 2020）（2.7B 和 175B 参数）。由于 GPT-2 系列模型的上下文窗口大小限制（最多 1024 个词片），我们对所有数据集除了 AGNews 和 DBPedia 使用 4-shot 设置。我们的实验基于开源的 GPT-2 模型检查点和使用 OpenAI GPT-3 API。对于探针集生成，我们限制最大生成长度为 128。我们还使用温度为 2 的采样，并使用块 n-gram 重复（Paulus et al., 2018）来鼓励多样性生成。

我们对每组随机选定的训练样本使用 24 种不同的排列，并使用 5 组不同的样本集（除了 GPT-3 的 175B 参数模型，由于高成本，我们只进行了两组，每组 12 种不同的排列）进行每项实验，总共进行了 120 次运行。我们报告了 5 组不

同样本集上相应评估指标的均值和标准差。

为了选择表现良好的提示，我们使用局部熵（LocalE）和全局熵（GlobalE）对候选提示进行排序，这些排序基于我们自动生成的探针集。然后我们选择根据最高熵值排名前 k 的样本，其中 k=4 是我们实验中的数量，从可用的 24 种排列中选择表现良好的提示。最后，我们使用这些表现良好的提示在各种数据集上评估性能，并展示了改进的性能和减少的方差。我们还为一个始终预测数据集中多数标签的主要基线提供了性能的下限。我们还提供了一个 oracle，以显示通过选择基于验证集提示性能的前四种表现良好的排序的上限性能。

4.1 评估数据集

类似于以前的工作（Gao et al., 2020; Zhao et al., 2021），我们使用了 11 个文本分类数据集，范围从情感分类到文本蕴含。数据集的更多详细信息在附录中提供。为了评估，我们对所有数据集的验证集样本进行了 256 个样本的子抽样，以控制 GPT-3 推理成本，因为它需要使用付费 API。

5 结果

我们在表 2 中报告了实验结果，并观察到无论是使用 LocalE 还是 GlobalE，所有任务都显示了一致的改进。

基于熵的探测对于表现良好的提示选择有效，无论模型大小如何。我们发现，与不使用探测的提示相比，全局熵平均在十一个不同的句子分类任务中实现了 13％的相对改进。局部熵提供的结果略低于全局熵，平均相对改进为 9.6％。我们选定的表现良好的提示也显示出比使用所有候选提示显著更低的方差。

使用基于熵的探测进行排序是稳健的。在图 8 中，我们展示了在变化 K 的情况下，对前 K 个提示选择的平均性能。K=24 相当于使用所有抽样的提示顺序，相当于表 2 中的基线模型性能。我们可以观察到，所有数据集的曲线斜率均为负，表明我们的方法能够有效地对表现良好的提示进行排名。虽然 K=1 在大多数情况下可以提供良好的性能，但在我们的实验中，我们使用 K=4，因为初步实验表明它能够在各数据集中提供稳定的性能。

	SST-2	SST-5	DBPedia	MR	CR	MPQA	Subj	TREC	AGNews	RTE	CB
Majority	50.9	23.1	9.4	50.0	50.0	50.0	50.0	18.8	25.0	52.7	51.8
Finetuning (Full)	95.0	58.7	99.3	90.8	89.4	87.8	97.0	97.4	94.7	80.9	90.5
GPT-2 0.1B	$58.9_{7.8}$	$29.0_{4.9}$	$44.9_{9.7}$	$58.6_{7.6}$	$58.4_{6.4}$	$68.9_{7.1}$	$52.1_{0.7}$	**49.2**$_{4.7}$	$50.8_{11.9}$	$49.7_{2.7}$	$50.1_{1.0}$
LocalE	**65.2**$_{3.9}$	$34.4_{3.4}$	$53.3_{4.9}$	$66.0_{6.3}$	**65.0**$_{3.4}$	$72.5_{6.0}$	$52.9_{1.3}$	$48.0_{3.9}$	$61.0_{5.9}$	**53.0**$_{3.3}$	$49.9_{1.6}$
GlobalE	$63.8_{5.8}$	**35.8**$_{2.0}$	**56.1**$_{4.3}$	**66.4**$_{5.8}$	$64.8_{2.7}$	**73.5**$_{4.5}$	**53.0**$_{1.3}$	$46.1_{3.7}$	**62.1**$_{5.7}$	**53.0**$_{3.0}$	**50.3**$_{1.6}$
Oracle	$73.5_{1.7}$	$38.2_{4.0}$	$60.5_{4.2}$	$74.3_{4.9}$	$70.8_{4.4}$	$81.3_{2.5}$	$55.2_{1.7}$	$58.1_{4.3}$	$70.3_{2.8}$	$56.8_{2.0}$	$52.1_{1.3}$
GPT-2 0.3B	$61.0_{13.2}$	$25.9_{5.9}$	$51.7_{7.0}$	$54.2_{7.8}$	$56.7_{9.4}$	$54.5_{8.8}$	$54.4_{7.9}$	$52.6_{4.9}$	$47.7_{10.6}$	$48.8_{2.6}$	$50.2_{5.3}$
LocalE	$75.3_{4.6}$	$31.0_{3.4}$	$47.1_{3.7}$	$65.2_{6.6}$	**70.9**$_{6.3}$	$67.6_{7.2}$	**66.7**$_{9.3}$	$53.0_{3.9}$	$51.2_{7.3}$	**51.8**$_{1.0}$	$47.1_{4.2}$
GlobalE	**78.7**$_{5.2}$	**31.7**$_{5.2}$	**58.3**$_{5.4}$	**67.0**$_{5.9}$	$70.7_{6.7}$	**68.3**$_{6.9}$	$65.8_{10.1}$	**53.3**$_{4.6}$	**59.6**$_{7.2}$	$51.1_{1.9}$	**50.3**$_{3.7}$
Oracle	$85.5_{4.3}$	$40.5_{6.3}$	$65.2_{7.6}$	$74.7_{6.1}$	$80.4_{5.4}$	$77.3_{2.3}$	$79.4_{2.4}$	$63.3_{2.9}$	$68.4_{8.0}$	$53.9_{1.3}$	$62.5_{7.4}$
GPT-2 0.8B	$74.5_{10.3}$	$34.7_{8.2}$	$55.0_{12.5}$	$64.6_{13.1}$	$70.9_{12.7}$	$65.5_{8.7}$	$56.4_{9.1}$	$56.5_{2.7}$	$62.2_{11.6}$	$53.2_{2.0}$	$38.8_{8.5}$
LocalE	$81.1_{5.5}$	$40.3_{4.7}$	$56.7_{7.5}$	$82.6_{4.2}$	$85.4_{3.8}$	$73.6_{4.8}$	**70.4**$_{4.2}$	$56.2_{1.7}$	$62.7_{8.1}$	$53.3_{1.6}$	$38.4_{5.2}$
GlobalE	**84.8**$_{4.1}$	**46.9**$_{1.1}$	**67.7**$_{3.6}$	**84.3**$_{2.9}$	**86.7**$_{2.5}$	**75.8**$_{3.1}$	$68.6_{6.5}$	**57.2**$_{2.3}$	**70.7**$_{3.6}$	**53.5**$_{1.5}$	**41.2**$_{4.5}$
Oracle	$88.9_{1.8}$	$48.4_{0.7}$	$72.3_{3.3}$	$87.5_{1.1}$	$89.9_{0.9}$	$80.3_{4.9}$	$76.6_{4.1}$	$62.1_{1.5}$	$78.1_{1.3}$	$57.3_{1.0}$	$53.2_{5.3}$
GPT-2 1.5B	$66.8_{10.8}$	$41.7_{6.7}$	$82.6_{2.5}$	$59.1_{11.9}$	$56.9_{9.0}$	$73.9_{8.6}$	$59.7_{10.4}$	$53.1_{3.3}$	$77.6_{7.3}$	$55.0_{1.4}$	$53.8_{4.7}$
LocalE	$76.7_{8.2}$	**45.1**$_{3.1}$	$83.8_{1.7}$	$78.1_{5.6}$	$71.8_{8.0}$	$78.5_{3.6}$	$69.7_{5.8}$	$53.6_{3.1}$	$79.3_{3.7}$	**56.8**$_{1.1}$	$52.6_{3.9}$
GlobalE	**81.8**$_{3.9}$	$43.5_{4.5}$	**83.9**$_{1.8}$	**77.9**$_{5.7}$	**73.4**$_{6.0}$	**81.4**$_{2.1}$	**70.9**$_{6.0}$	**55.5**$_{3.0}$	**83.9**$_{1.2}$	$56.3_{1.2}$	**55.1**$_{4.6}$
Oracle	$86.1_{1.5}$	$50.9_{1.0}$	$87.3_{1.5}$	$84.0_{2.7}$	$80.3_{3.3}$	$85.1_{1.4}$	$79.9_{5.7}$	$59.0_{2.3}$	$86.1_{0.7}$	$58.2_{0.6}$	$63.9_{4.3}$
GPT-3 2.7B	$78.0_{10.7}$	$35.3_{6.9}$	$81.1_{1.8}$	$68.0_{12.9}$	$76.8_{11.7}$	$66.5_{10.3}$	$49.1_{2.9}$	$55.3_{4.4}$	$72.9_{4.8}$	$48.6_{1.9}$	$50.4_{0.7}$
LocalE	**81.0**$_{6.0}$	$42.3_{4.7}$	$80.3_{1.7}$	$75.6_{4.1}$	$79.0_{5.5}$	$72.5_{5.8}$	$54.2_{4.2}$	$54.0_{2.6}$	$72.3_{4.6}$	$50.4_{1.9}$	$50.5_{0.8}$
GlobalE	$80.2_{4.2}$	**43.2**$_{4.3}$	**81.2**$_{0.9}$	**76.1**$_{3.8}$	**80.3**$_{3.4}$	**73.0**$_{4.3}$	**54.3**$_{4.0}$	**56.7**$_{2.0}$	**78.1**$_{1.9}$	**51.3**$_{1.8}$	**51.2**$_{0.8}$
Oracle	$89.8_{0.7}$	$48.0_{1.1}$	$85.4_{1.6}$	$87.4_{0.9}$	$90.1_{0.7}$	$80.9_{1.4}$	$60.3_{10.3}$	$62.8_{4.2}$	$81.3_{2.9}$	$53.4_{3.1}$	$52.5_{1.4}$
GPT-3 175B	**93.9**$_{0.6}$	$54.4_{2.5}$	$95.4_{0.9}$	**94.6**$_{0.7}$	$91.0_{1.0}$	$83.2_{1.5}$	$71.2_{7.3}$	$72.1_{2.7}$	$85.1_{1.7}$	$70.8_{2.8}$	$75.1_{5.1}$
LocalE	$93.8_{0.5}$	**56.0**$_{1.7}$	$95.5_{0.9}$	$94.5_{0.7}$	$91.3_{0.5}$	**83.3**$_{1.7}$	$75.0_{4.6}$	$71.8_{3.2}$	**85.9**$_{0.7}$	**71.9**$_{1.4}$	$74.6_{4.2}$
GlobalE	**93.9**$_{0.6}$	$53.2_{2.1}$	**95.7**$_{0.7}$	**94.6**$_{0.2}$	**91.7**$_{0.4}$	$82.0_{0.8}$	**76.3**$_{3.5}$	**73.6**$_{2.5}$	$85.7_{1.0}$	$71.8_{1.9}$	**79.9**$_{3.3}$
Oracle	$94.7_{0.2}$	58.2	$96.7_{0.2}$	$95.5_{0.2}$	$92.6_{0.4}$	$85.5_{0.8}$	$81.1_{4.9}$	$77.0_{1.2}$	$87.7_{0.6}$	$74.7_{0.4}$	$83.0_{0.9}$

表 2 我们在验证集子集上的主要结果。为了适应 GPT-2 模型上下文窗口大小，我们对 DBPedia 使用 1-shot，对 AGNews 使用 2-shot，对其他数据集使用 4-shot。所有基线结果都是基于 24 种训练上下文排列的 5 个不同随机种子计算得出的。LocalE 和 GlobalE 结果是基于我们提出的方法使用前 4 种上下文排列计算得出的。对于 GPT-3 175B，由于计算预算有限，我们只使用 2 个种子和 12 种不同的排列。

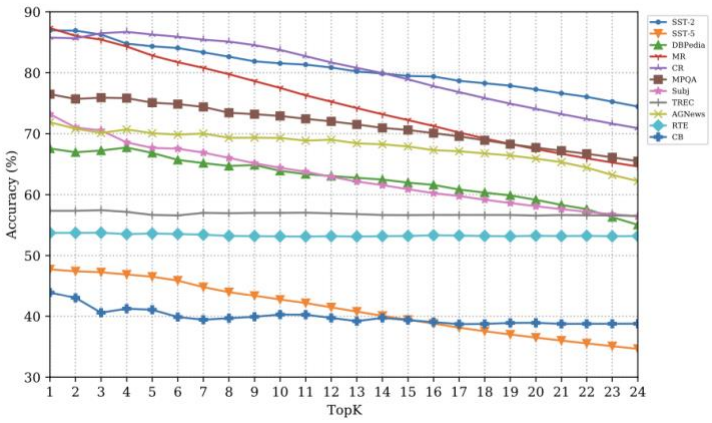

图 8 GPT2-Large (0.8B) 上不同 Top K 排列选择的平均性能

基于熵的探测在模板间也有效。我们针对 SST-2 数据集评估了四种不同的模板（类似于 Gao et al.，2020 和 Zhao et al.，2021 的工作，见表 4），在表 3 中展示了实验结果，表明基于熵的探测对不同模板有效。我们还观察到，不同模板间的随机性与第 2 节中的相似。这些发现表明，基于熵的探测对特定模板不敏感，因为它始终能够为所有案例提供改进。

	Template 1	Template 2	Template 3	Template 4
GPT-2 0.1B	$58.9_{7.8}$	$57.5_{6.8}$	$58.1_{7.4}$	$56.6_{6.6}$
LocalE	$\mathbf{65.2}_{3.9}$	$\mathbf{60.7}_{4.6}$	$\mathbf{65.4}_{4.8}$	$61.0_{4.7}$
GlobalE	$63.8_{5.8}$	$59.0_{2.9}$	$64.3_{4.8}$	$\mathbf{63.5}_{4.8}$
GPT-2 0.3B	$61.0_{13.2}$	$63.9_{11.3}$	$68.3_{11.8}$	$59.2_{6.4}$
LocalE	$75.3_{4.6}$	$70.0_{7.2}$	$80.2_{4.2}$	$62.2_{3.4}$
GlobalE	$\mathbf{78.7}_{5.2}$	$\mathbf{73.3}_{4.5}$	$\mathbf{81.3}_{4.1}$	$\mathbf{62.8}_{4.3}$
GPT-2 0.8B	$74.5_{10.3}$	$66.6_{10.6}$	$70.3_{10.5}$	$63.7_{8.9}$
LocalE	$81.1_{5.5}$	$80.0_{5.6}$	$73.7_{6.2}$	$\mathbf{71.3}_{4.5}$
GlobalE	$\mathbf{84.8}_{4.1}$	$\mathbf{80.9}_{3.6}$	$\mathbf{79.8}_{3.9}$	$70.7_{5.3}$
GPT-2 1.5B	$66.8_{10.8}$	$80.4_{7.6}$	$54.5_{7.9}$	$69.1_{10.5}$
LocalE	$76.7_{8.2}$	$83.1_{3.6}$	$66.9_{7.5}$	$72.7_{5.5}$
GlobalE	$\mathbf{81.8}_{3.9}$	$\mathbf{83.4}_{3.2}$	$\mathbf{67.2}_{6.1}$	$\mathbf{74.2}_{5.3}$

表 3 不同模板在 SST-2 上的提示选择性能

ID	Template	Label Mapping
1	Review: {Sentence} Sentiment: {Label}	positive/negative
2	Input: {Sentence} Prediction: {Label}	positive/negative
3	Review: {Sentence} Sentiment: {Label}	good/bad
4	{Sentence} It was {Label}	good/bad

表 4 SST-2 的不同模板

性能良好的排列选择对于上下文学习是一个安全的选择。我们发现，对于那些提示方差较高的模型，我们的提示选择过程可以显示出大幅度的改进——最高可达 30%的相对改进。此外，对于初始提示性能方差较低的任务，我们的方法不会对性能产生负面影响。我们的提示选择在最差情况下提供边际改进，在大多数情况下平均提高了 13%。

对于小型模型，即使使用表现良好的排列选择，句子对任务也仍然具有挑战性。对于 CB 和 RTE 数据集，GPT-2 模型的性能与随机基线没有显著差异。尽管如此，我们发现我们的识别表现良好的提示方法仍然可以提供最小的性能增益，尽管这些仍然处于随机猜测或多数票水平。其中一个原因可能是，对于这些特定的模型大小和任务，不存在良好的提示。因此，在这种设置中优化提示并不特别有效。这进一步得到验证，即提示选择在更大的模型尺寸（尤其是 GPT-3 175B 参数模型）上显著提高了 CB 和 RTE 的性能。实际上，我们发现，使用 GlobalE

的提示选择在 CB 上为 GPT-3 175B 提高了 4.9%的性能。这表明我们的方法适用于所有模型大小和所有任务，只要它们已经具有一些可以通过提示设计改进的现有分类能力。

基于熵的探测优于使用训练数据子集进行调整。如果不依赖于生成，选择提示的另一种方法可能是将（有限的）训练数据分割成一个验证集。为了与这种方法进行比较，我们将 4-shot 训练样本（与表 2 中的设置相同）分成两半。然后我们选择使用验证集性能的前四种表现良好的提示。如表 5 所示，这种方法始终优于基线。然而，基于熵的两种探测方法始终提供了更好的性能，跨所有模型大小。

	GPT-2 0.1B	GPT-2 0.3B	GPT-2 0.8B	GPT-2 1.5B
Baseline	$58.9_{7.8}$	$61.0_{13.2}$	$74.5_{10.3}$	$66.8_{10.8}$
LocalE	$\mathbf{65.2}_{3.9}$	$75.3_{4.6}$	$81.1_{5.5}$	$76.7_{8.2}$
GlobalE	$63.8_{5.8}$	$\mathbf{78.7}_{5.2}$	$\mathbf{84.8}_{4.1}$	$\mathbf{81.8}_{3.9}$
Split Training Set	$62.8_{5.3}$	$64.2_{6.1}$	$75.1_{6.8}$	$71.4_{7.8}$

表 5 将我们的方法与将训练集分割为训练和开发集进行比较，用于 SST-2。

6 相关工作

NLP 的统一界面设计。大多数先前的工作都集中在共享参数模型上，先在一些任务上进行预训练，然后为不同的任务进行微调，例如 ELMo（Peters et al., 2018）、BERT（Devlin et al., 2019）等。这最终导致了多个特定于任务的模型。一段时间以来，人们一直在尝试为 NLP 任务设计一个统一的界面（Kumar et al., 2016; Raffel et al., 2020）。与这些工作并行，GPT-2（Radford et al., 2019）表明，在语言模型输入末尾添加触发令牌（例如“TL;DR”）可以使语言模型表现得像总结模型。语言模型的零样本能力显示了将 NLP 任务统一到一个语言建模框架中的潜力，其中无需微调即可实现良好性能。此外，GPT-3（Brown et al., 2020）表明，通过扩展语言模型，任务非依赖的少样本性能可以得到改善，有时甚至可以与以前的最佳微调方法竞争。

PLM 的提示设计。提示设计的核心挑战是将训练数据（如果存在）转换为文本序列。关于提示设计的大多数工作都集中在如何使提示与语言模型更兼容。Petroni et al.（2019）使用人力设计自然语言句子，然后在给定输入上下文的情况下进行令牌预测。然而，手工制作的模板需要大量的人力，并且很可能导致性能不佳。最近的工作已经探索了自动模板构建：Schick 和 Schütze（2020）使用填

空式任务构建模板，Gao et al.（2020）使用外部语言模型生成模板，Shin et al.（2020）使用基于梯度的搜索找到最大化性能的模板。Jiang et al.（2020）使用基于挖掘的方法自动创建多个不同的模板。

提示设计的顺序敏感性。Gao et al.（2020）表明，微调基准方法并不像上下文学习那样对顺序敏感。利用标准大小的训练集，Liu et al.（2021）使用最近邻搜索检索特定测试样本的最相关训练样本。他们在检索相关样本后成功，并得出结论，提供这些样本的顺序对性能几乎没有影响。虽然我们的研究与他们的根本不同，因为我们没有使用标准大小的训练集，但我们得出了相反的结论。所有先前关于提示设计的工作都集中在提示的文本质量上，并且据我们所知，没有详细研究过顺序敏感性。

真正的少样本学习。Perez et al.（2021）评估了在没有预留验证集的情况下 LM 的少样本能力。实验结果表明，先前的工作高估了 LM 在这种（真正的少样本学习）设置下的少样本能力。我们的工作则利用语言模型的生成性质构建探针集，而不依赖于预留示例。我们表明，我们的探针方法优于依赖预留示例（如图 5 所示），因此可以实现真正的少样本学习。

7 结论

我们已经表明，少量样本的提示会受到顺序敏感性的影响，即对于同一个提示，样本提供的顺序可能是实现最先进性能与随机性能之间的区别。在我们对这个问题的分析中，我们确定它存在于各种任务、模型大小、提示模板、样本以及训练样本数量之间。为了缓解这个问题，我们引入了一种新的探针方法，该方法利用语言模型的生成性质来构建一个人工开发集。我们能够使用这个集合上的熵统计来识别性能良好的排列，从而在十一个文本分类任务中平均实现了 13%的改进。

参考文献

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, pages 177–190. Springer.

Joe Davison, Joshua Feldman, and Alexander M Rush. 2019. Commonsense knowledge mining from pre- trained models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP- IJCNLP)*, pages 1173–1178.

Marie-Catherine De Marneffe, Mandy Simons, and Judith Tonhauser. 2019. The commitmentbank: Investigating projection in naturally occurring discourse. In *proceedings of Sinn und Bedeutung*, pages 107– 124.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2020. Making pre-trained language models better few-shot learners. *arXiv preprint arXiv:2012.15723*.

Minqing Hu and Bing Liu. 2004. Mining and summa- rizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177.

Zhengbao Jiang, Frank F Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *International conference on machine learning*, pages 1378–1387. PMLR.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2021. What makes good in-context examples for gpt-3? *arXiv preprint arXiv:2101.06804*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining ap- proach. *arXiv preprint arXiv:1907.11692*.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 271–278.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment

categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Compu- tational Linguistics (ACL'05)*, pages 115–124.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive sum- marization. In *International Conference on Learning Representations*.

Ethan Perez, Douwe Kiela, and Kyunghyun Cho. 2021. True few-shot learning with language models. *arXiv preprint arXiv:2105.11447*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227– 2237.

Fabio Petroni, Patrick Lewis, Aleksandra Piktus, Tim Rocktäschel, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. 2020. How context affects language models' factual predictions. In *Automated Knowledge Base Construction*.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP- IJCNLP)*, pages 2463–2473, Hong Kong, China. As- sociation for Computational Linguistics.

A. Radford, Jeffrey Wu, R. Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language mod- els are unsupervised multitask learners. In *OpenAI Blog*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

Timo Schick and Hinrich Schütze. 2020. It's not just size that matters: Small language mod- els are also few-shot learners. *arXiv preprint arXiv:2009.07118*.

Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv preprint arXiv:2010.15980*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Ellen M Voorhees and Dawn M Tice. 2000. Building a question answering test collection. In

Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval, pages 200–207.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emo- tions in language. *Language resources and evaluation*, 39(2):165–210.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Car- bonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Xiang Zhang, Junbo Zhao, and Yann Lecun. 2015. Character-level convolutional networks for text classification. *Advances in Neural Information Processing Systems*, 2015:649–657.

Tony Z Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Im- proving few-shot performance of language models. *arXiv preprint arXiv:2102.09690*.