

STA5007 Project 1 Report

12112627 李乐平

- Q1 -

Paper Chosen

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

What is the problem addressed in the paper?

此论文中着力研究的问题是：在少样本预训练语言模型 (Pretrained Language Model, PLM) 的上下文学习 (In-context Learning) 中出现的顺序敏感性 (Order Sensitivity) 问题，即在使用预训练模型进行少样本学习 (Few-shot Learning, 在语境下与上下文学习意思相近) 的过程中，样本的顺序会对训练的效果产生相当的影响。此外，论文中还提到增大训练模型大小、增加样本数量等操作并不一定能够显著降低由顺序敏感性带来的方差。同时，在不同模型中展现良好表现的样本顺序并不一致。

在论文中，作者提出了问题：「（对于给定的预训练模型和样本）如何自动生成"探测集 (Probing Set)"以找寻最佳样本顺序？」并提供了一种可能的解决方案。

Is this a new problem?

论文中提到 2020 年 Gao 等人展示了上下文学习的顺序稳定性不如微调方法的，因此这不是一个新问题。但是，那篇文章并未深究样本顺序究竟如何影响上下文学习的表现。

What is the scientific hypothesis that the paper is trying to verify?

论文希望验证其提出的局部熵 (Local Entropy, LocalE) 和全局熵 (Global Entropy, GlobalE) 度量方法能够有效地从探测集中找到较优样本顺序，从而一定程度上规避顺序敏感性，同时还希望验证这些方法是稳健的。其主要想法是较差的顺序会使生成的分布更加极端，因此在一定程度上熵值越大，可以认为顺序越好。

What are the key related works and who are the key people working on this topic?

论文主要围绕基于如下概念展开：上下文学习和顺序敏感性。其中，与上下文学习相关的关键文章是 Brown 等人撰写的 Language Model are Few-Shot Learners，这篇论文昭示了扩大语言模型的规模能够极大地提高其在任务无偏、少样本下的表现，甚至能与微调模型相媲美。而 Gao 等人的 Making pre-trained language models better few-shot learners 则研究了基于提示词的微调方法，其提及了调换句子的顺序会导致表现的巨大偏差，但未进行更深入的研究。

What is the key of the proposed solution in the paper?

解决问题的关键在于"探测集"的构建以及评价样本顺序优劣的度量方法。一个探测集由若干随机选出的样本进行多次排列，分别供给预训练模型进行训练后生成的提示词拼接组成。文中提到了两种可能的评价方法：全局熵和局部熵。在选用一种评价方法应用于探测集后，对探测集中的提示词进行排序，可以得到一个基于评价方法的提示词排名。论文认为排名高的提示词表现更佳，可以有效地对应到一种较优的样本顺序。

How are the experiments designed?

论文中选取了若干不同的数据集，在不同规模下（0.1B~175B）的大语言模型上进行了实验。对于每一组实验，均设置了 4 个对比组：随机组、全局熵组、局部熵组和最优组，实验记录了这四组在对应数据集上的给出的样本顺序的表现与方差，并最终表格形式呈现。

对于每一次实验，作者均构建了 5 个探测集（GPT-3 组为 2 个），每个探测集均基于随机挑选的 4 个样本的 24 种组合生成（GPT-3 组为 12 种）。在探测集的提示词中，根据对应的指标进行排名，选出排名前 k（k = 4）的提示词在不同的数据集上进行表现评估。

	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-2 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}

What datasets are built/used for quantitative evaluation? Is the code open sourced?

实验采用了 SST-2、SST-5、DBPedia、MR、CR、MPQA、Subj、TREC、AGNews、RTE 和 CB 等 11 个数据集。文中附录给出了 11 个数据集对应的论文链接，但是并没有给出用于实验的源代码。

Is the scientific hypothesis well supported by evidence in the experiments?

实验表明基于局部熵指标和全局熵指标构筑的样本顺序相比基准模型（随机）的表现分别取得了 9.6%和 13%的平均相对提升。同时，文中还提到基于这些熵指标的探测方法具有稳健性和跨模板性（模板指将句子和标签拼接成格式化字符串的方式），基于此操作的顺序选择不会产生负面效果（即表现不会亚于随机选择）。

虽然相较于随机组的基准线，探测集法的表现有所提升，但提升并不明显，且几乎均未寻到最优顺序。而且文中最多使用了 5 个探测集——每个探测集均基于随机的 4 个样本生成——这很难排除实验的结果是否是由随机因素造成的。基于此，我认为应进行更多的实验，以进一步验证探测集法的有效性。

此外，论文中仅提出了全局熵和局部熵的概念以及计算方法，缺乏理论上的证明。同时，因为其没有源代码，所以可信度也存疑。

What are the contributions of the paper?

论文研究了上下文学习中的顺序敏感性，并昭示了其在少样本学习中的重要性。然后，其提出了一种能够不依赖额外数据的有效寻找良好顺序的探测方法，并为此提供了局部熵和全局熵两种评价指标。最后，其展示了这种方法在不同规模、不同模型和不同数据集上的通用性。

What should/could be done next?

论文中提出的两种评价方法均未达到最优，还有提升空间，因此可以探索其他的评价方法。另外，论文还提到对于较小模型上的句对任务（Sentence-pair task），即使采用最优的顺序，其效果仍欠佳。

除此之外，关于顺序敏感性的成因以及是否存在更加底层的规避方法也值得研究。

- Q2 -

List three tasks that you want the large language model (LLM, e.g., ChatGPT) to complement. The tasks should be diverse and complex. Find effective approaches to prompt the LLM to work for you. In your report, for each task,

1. describe how you try different prompts to improve the results, explain why each variation can improve. Write the prompts and screenshot of each trial (at least two).
2. include the LLM you use, the task description, final prompt, the answer you expect, and LLM-generated answer.

由于接下来涉及的对话均非常长，**prompt 非常多**，所以贴上分享链接和部分截图。真正复杂的任务不可能用单条 prompt 解决。因此接下来的展示中可能不会出现所谓的「final prompt」的说法。同时，在我提问时因为我并不知道答案，所以也没有「the answer I expect」的说法。

任务中使用的大模型均为 chatGPT-3.5。

在每项任务中，我均会根据 chatGPT 生成的回答比较与我遇到的问题之间的差异进行引导。在处理复杂的任务时，我也会先将复杂的任务分割为若干较为简单的任务，然后利用每一步得到的新的信息递进地交由 chatGPT 处理，如此的效果会好得多。

即是说，要想改进生成的结果，需要**划分困难的任务**，以及**提供更充分的信息**，接下来的 task 展示均表明了这两点。希望上面这句话已经充分阐释了任务能够推进并完成的原因。

Task 1:

此任务的目标是为一段通过 python 编写的执行通用化数据库批量插入更新语句的脚本进行 debug，使之能够成功实现插入。在此我将简单讲解编写时遇到的一些问题。

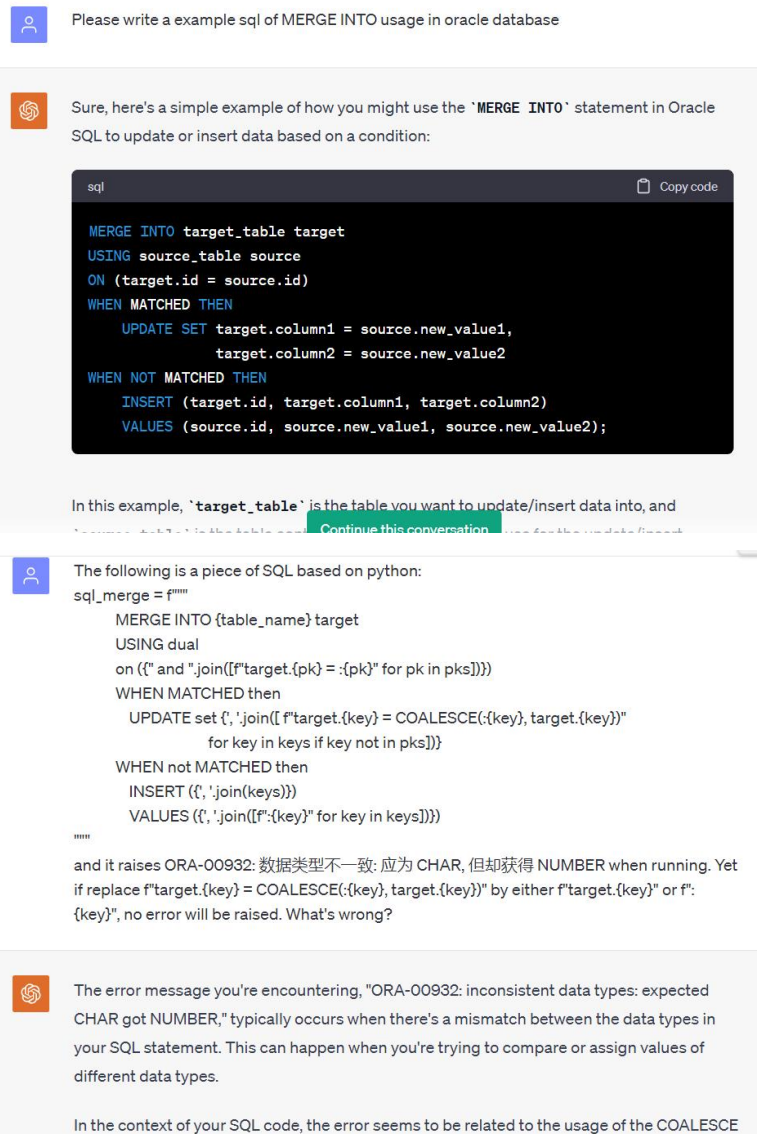
其一：部分列更新。这一点我自己解决了。

其二：报错 ORA-00932，数据类型不匹配，但是由于 SQL 语言的特性（缺点）使得定位问题非常困难。逐列检查并没有发现问题的根源，只好在执行时统一进行格式化，最后成功运行。此问题花费了 1 天半的时间才在 GPT 的辅助下解决，见 Trial1/2。

其三：期望更新，但是意外执行了插入。解决此问题也花费了我大量时间，最后查明原

因是 CHAR(2)类型在插入单字符时会在后面自动补空格,在匹配时用 TRIM 函数去空格即可成功。见 Trial3。

因为需要解决的子任务众多,所以没有「final prompt」,GPT 也未能给出完整的最终答案,但是最终的问题都被一一解决了。



Trial 0:

<https://chat.openai.com/share/6eb54a41-3f5a-434d-a5b0-e0815caf0514>

Trial 1:

<https://chat.openai.com/share/f0fbdc3b-6f9a-40a6-ab50-23c49b72c726>

Trial 2:

<https://chat.openai.com/share/1e7dd937-52a5-4831-92f8-d99ad912c9de>

Trial 3:

<https://chat.openai.com/share/739c5d16-36d0-44d8-8828-737c0578039b>

综合以上所有对话的成果,并加以改进,终于得到了最终的答案:

```
sql_merge = f"""
MERGE INTO {table_name} target
USING dual
on ({' and ".join([f"TRIM({pk}) = :{pk}" for pk in pks])'})
WHEN MATCHED then
    UPDATE set {' , '.join([ f"target.{key} = COALESCE(TO_{convert_dict[table_name][key]}(:{key}
                                for key in keys if key not in pks)])}
WHEN not MATCHED then
    INSERT ({' , '.join(keys)})
    VALUES ({' , '.join([f":{key}" for key in keys])})

```

Task 2:

此任务的目标是创建一个 Oracle 数据库连接池。因为最终结果依赖前面的结果，因此最终的 prompt（can you enclosed the functions above into a class）没有意义。最终结果很长，仅贴上部分截图。

Trial 1:

<https://chat.openai.com/share/2b1a8524-9cba-4541-bd2f-38a1839290d2>

Trial 2:

<https://chat.openai.com/share/d6ad6ca2-5409-4b69-a7ab-7fac4257bfac>

How to use python to create a connection pool to an oracle database

To create a connection pool to an Oracle database in Python, you can use the `cx_Oracle` library along with the `ThreadPoolExecutor` from the `concurrent.futures` module. `cx_Oracle` is a popular Python module that allows you to interact with Oracle databases, and `ThreadPoolExecutor` helps manage the connection pool.

Before proceeding, make sure you have the `cx_Oracle` library installed. You can install it using `pip`:

```
bash
pip install cx_Oracle
```

Here's a step-by-step guide on how to create a connection pool to an Oracle database using Python:

1. Import the necessary libraries:

```
python
```

最终答案：

can you enclosed the functions above into a class

Of course! Here's how you can encapsulate the connection pool functionality into a class:

```
python
import cx_Oracle
import queue

class OracleConnectionPool:
    def __init__(self, dsn, user, password, pool_size=5):
        self.dsn = dsn
        self.user = user
        self.password = password
        self.pool_size = pool_size
        self.connection_queue = queue.Queue(maxsize=pool_size)

        self._initialize_pool()

    def _initialize_pool(self):
        def create_connection():
            return cx_Oracle.connect(self.user, self.password, self.dsn)
```


Task 3:

此任务的目标是通过服务器的 JupyterLab 端口顺利导入一个 Python 库，并解决路径问题。

Trial 1:

<https://chat.openai.com/share/fb5fa92c-a238-4b90-9642-00231c0eddf1>

Trial 2:

<https://chat.openai.com/share/4b9226c6-c466-48b4-9cca-9dc296ac3a5b>

Trial 3:

<https://chat.openai.com/share/daecbce5-cbce-43a2-9cd1-d0c1a8cf245b>

解决Seaborn安装超时问题

October 16, 2023



如果jupyter notebook安装seaborn时访问Obtaining dependency information for seaborn from <https://files.pythonhosted.org/packages/7b/e5/83fcd7e9db036c179e0352bfcd20f81d728197a16f883e7b90307a88e65e/seaborn-0.13.0-py3-none-any.whl.metadata>超时怎么办



如果在安装 Seaborn 时遇到 "Obtaining dependency information for seaborn from <https://files.pythonhosted.org/packages/7b/e5/83fcd7e9db036c179e0352bfcd20f81d728197a16f883e7b90307a88e65e/seaborn-0.13.0-py3-none-any.whl.metadata>" 超时的问题，通常可以采取以下一些步骤来解决问题：

1. **检查网络连接**：首先确保你的网络连接正常，尤其是检查是否能够访问 <https://files.pythonhosted.org> 这个网址。有时候网络问题会导致超时错误。
2. **使用国内镜像源**：如果你位于中国或访问国际站点较慢，可以考虑使用国内的镜像源来加速安装。例如，使用清华大学的 PyPI 镜像站点。

Final prompt:

「

以下是运行

```
!python -m pip install -i https://pypi.tuna.tsinghua.edu.cn/simple --user --upgrade seaborn sys.path
```

的结果：

[报告中省略若干输出信息，详见 Trial 3]

应该如何做才能使 seaborn 正常 import

」

最终答案：



根据你提供的输出信息，seaborn 已经成功安装在用户目录下的 `~/local/lib/python3.10/site-packages` 中。然而，在你的 Jupyter Notebook 环境中，搜索路径似乎没有包含该目录，这可能是导致 `'seaborn'` 无法正常导入的原因之一。

你可以尝试以下方法来解决问题：

1. **将用户目录的 `local` 目录添加到 `sys.path` 中：**

在 Jupyter Notebook 中，你可以手动将 `~/local/lib/python3.10/site-packages` 添加到 `sys.path`，这样 Python 就能找到 `'seaborn'` 模块。在你的 Jupyter Notebook 中执行以下代码：

```
python
import sys
sys.path.append('/data/root/.local/lib/python3.10/site-packages')
```

这将添加 `~/local/lib/python3.10/site-packages` 到搜索路径中，使得 `'seaborn'` 可以正常导入。