实验部分 1

刘昊

2025 年 1 月 14 日

1 基线：自回归采样

1.1 基本介绍

为了比较多种推测采样的提速效果，先采用两个目标模型 Llama-2-7b-chat-hf 以及 Llama-2-13b-chat-hf 在 humaneval 数据集上自回归解码推理得到结果，记录正确率和采样速度作为比较的基线。本部分自回归采样的基本代码来自LLMSpeculativeSampling. 基线、实验一、实验二的基础环境配置如表 1。所有模型 (目标模型或起草模型) 在自回归采样部分生成 token 采用贪婪采样策略 (top k=1,temperature=0)

环境参数	配置
显卡	910B
显存	65536MB
目标模型	Llama-2-7b-chat-hf/ Llama-2-13b-chat-hf
top k	1
top p	/
temperature	0
生成 token 个数	200
数据集	humaneval

表 1: 基础配置

1.1.1 测试数据集

humaneval 是一个开源的 python 代码生成评估数据集，其包括 164 条 python 代码题目，每道题包括 prompt 部分和测试用例。每道题的 prompt 部分采用 zero-shot 的提问方式，其中函数头详细给出了函数名称以及输入变量名，doc string 部分用注释的方式给出了函数功能的描述。测试用例是一系列断言语句，用于检测特定输入下函数能否给出正确的输出。本项目使用的 humaneval 数据集已附在文件夹内，评测代码来自于相关开源的 python 工具包。

1.1.2 指标设置

- pass@k[1] 是一个代码生成领域常用的衡量代码正确性的评估指标，其计算方式为:

$$pass@k := E_{\text{Problems}}\left[1 - \frac{\binom{n-c}{k}}{\binom{n}{k}}\right]. \tag{1}$$

其中 n 是每道题重复采样以生成回复的次数，c 是 n 个回复中正确的个数，k 代表从 n 个回复中无放回的抽样次数。最终的 pass@k 指标会对数据集的所有题目求期望得到模型在该数据集上的代码生成正确性。在本实验中，我们采用 pass@1，即 n=k=1，对每个问题生成一次回复后检测回复中代码的通过率。

- 在实验一实验二以及对应基线中，采样速度是指某采样策略得到回复部分的单个 token id 所花费的平均 walltime 时长，即计时的起始时刻是 prompt 部分对应的 input id 送入相应的生成函数，结束时刻是生成函数生成到了限制个数的 token id，或是生成了 <EOS> token 的时刻。用生成的总 token 个数除以该段时长作为采样速度。在代码中采用 time.time() 分别记录开始和结束的时刻。需要额外注意的是：
  - 回复部分不包含 prompt 的输入 token id；计时阶段针对采样过程，不包含采样结束得到的 token id 序列被 tokenizer 解码为自然语言的时间。
  - 采用 walltime 计算解码速度作为评估指标有一定的局限性，因为随着系统不同以及显卡的性能不同会使得不同工作的效果难以比较 [2]. 为了避免这一局限性，在比较实验一和实验二两种推测采样时系统和显卡维持一致，尽可能做到了排除无关变量。
  - 对于推测采样策略，起草模型生成 token 的过程同样被计入耗时之内，但是自回归采样策略并没有起草模型生成 token 这一过程，也不会涉及到多个模型的协调、通信、调用等相关程序执行时长。因此采用 walltime 计算的速度难以公平比较自回归策略和推测采样之间的效果。虽有论文提出自己的计时指标 [2], 但是难以兼容到其余的采样策略上。
  - 在本项目的实验一和实验二中，为了从用户的等待时长和感受出发，并统一比较各个采样策略的响应时长，速度的测量选择用 walltime 时长这一宏观指标，同时尽可能做到实验环境相同。

1.2 结果分析

基线的实验结果如表 2所示。为验证 pass@1 结果的可靠性，此处对比了 Llama2 发布论文中的结果。touvron 等人 [3] 测试 Llama2 7b 在 humaneval 上的 pass@1(temperature=0.1,top_p=0.95) 为 0.128, Llama2 13b 在 humaneval 上的 pass@1(temperature=0.1,top_p=0.95) 为 0.158 考虑到本项目中自回归采用贪婪解码，正确率稍微下降是合理的。

目标模型	Llama-2-7b-chat-hf	Llama-2-13b-chat-hf
平均速度	15.97 token/s	12.95 token/s
pass@1	0.1159	0.1463

表 2: 自回归结果

2 实验一：朴素的推测采样

2.1 基本介绍

采用推测采样策略，在同样的配置下对 humaneval 数据集推理得到结果，记录正确率和采样速度。本部分推测采样的基本代码为 Google 论文 [4] 介绍的版本，由于原论文并未对代码开源，

因此实现的代码来自代码仓LLMSpeculativeSampling，相关代码对于论文的推测解码策略实现了带有 KV Cache 版本的改进。在沿袭了基线配置的基础上，对于 Llama-2-7b-chat-hf 将起草模型设置为含有 160M 参数的Llama-160m. 选择它的目的是与实验二论文中的起草模型对齐，此外与目标模型保持相同的 tokenizer. 起草模型一次性起草的 token 序列长度为 4. 对于 Llama-2-13b-chat-hf，起草模型设置为 Llama-2-7b-chat-hf，从而比较起草模型参数大小的影响。

2.2 结果分析

朴素推测采样的实验结果如表 3所示。通过和基线的比较可以发现，生成 token 的速度以及 pass@1 都有所波动。进一步做具体的分析。

目标模型	Llama-2-7b-chat-hf	Llama-2-13b-chat-hf
起草模型	Llama-160m	Llama-2-7b-chat-hf
平均速度	9.541 token/s	10.658 token/s
pass@1	0.1098	0.1524

表 3: 朴素推测采样结果

- 在配置一中，pass@1 指标前后有轻微的下降 (0.0061)，即 164 题中推测解码比自回归解码少做对一道题。可以理解成目标模型由于接受了小参数的起草模型错误的 token 导致整体回复质量下降，也可以理解为测量误差，两者的效果并没有显著差别。在配置二中，pass@1 指标前后有轻微的上升 (0.0061)，即 164 题中推测解码比自回归解码多做对一道题。可以理解成 13b 的模型有了 7b 模型的辅助整体回复质量提升，也可以理解为测量误差，两者的效果并没有显著差别。

- 两个推测采样的速度相比于对应的自回归采样平均速度下降较大，即在同样生成 200 个 token 的情况下，采用推测解码耗时更多。这与先前提到的 walltime 计时的局限性有关，也因为自回归采样不需要对两个模型交替采样，具有一定的机制优势。下面做进一步分析。

  - 在目标模型 7b，起草模型 160M 的实验中，经过计算发现平均每条回复 200 个 token 中由起草模型产生被目标模型接受的 token 个数为 91.6097，剩余 token 为目标模型自己生成的。目标模型生成的 token 又分为两种情况，第一类是完全接受了起草模型所生成的 gamma 个 token 后开始自己产生下一个 token，该类的平均个数为 11.8841；第二类是起草模型生成的 gamma 个 token 中存在某个位置的 token 没有被接受，于是在该位置自己重新生成 token，该类个数为 97.5548。三类合并得到 200 个 token。
  - 在目标模型 13b，起草模型 7b 的实验中，经过计算后得到由起草模型产生被目标模型接受的 token 个数为 152.37，目标模型完全接受了起草模型所生成的 gamma 个 token 后开始自己产生下一个 token 个数为 33.31，目标模型重新采样个数为 16.15。
  - 因此可以发现，推测采样通过让起草模型自回归得到 gamma 个 token 后让目标模型并行对 gamma 个 token 决定是否接受，理论上减少了目标模型的串行采样的次数从而加速 token 生成，但是当起草模型所生成的 token 大部分都被目标模型拒绝时（即配置一的情况），额外耗费了起草模型生成被拒绝的 token 以及目标模型判断草稿 token 是否被接受的时间，导致最终 walltime 时长变长。当起草模型和目标模型参数量差距较大

时或是起草模型的参数量过小时，较有可能出现这种情况。而当起草模型采用 7b 的情况下，目标模型重新采样的次数大幅减少，因此速度有所提升。

- 通过自回归采样内部的横向对比以及推测采样的内部横向对比也可以得出推测采样确实可以提升解码速度这一结论。因为在自回归的内部对比中，目标模型从 7b 上升到 13b，推理速度降低是必然的；但是两个目标模型各自配了相应的起草模型下（即使 13b 目标模型的起草模型也存在参数量远大于 7b 目标模型的起草模型而存在起草部分的自回归更耗时的情况），13b 的更大的目标模型的推测采样推理速度反而比 7b 的目标模型的推测采样推理速度更快。

3 实验二：Cascade Speculative Drafting

3.1 基本介绍

采用 Chen 等人提出的 Cascade 推测采样策略 [2]，在同样的配置下对 humaneval 数据集推理得到结果，记录正确率和采样速度。本部分推测采样的基本代码为论文配套代码仓的版本。在沿袭了基线配置的基础上，将 7b 的目标模型的起草模型同样设置为论文中使用的含有 160M 参数的Llama-160m, 此处由于显存限制，给 13b 配 160M 参数的起草模型. k 矩阵按照代码仓默认的矩阵 $\begin{bmatrix} 5 & 10 \\ 0 & 10 \end{bmatrix}$. 二元组模型来源为 wiki__bigram__naive__bayers__greedy__llama__next__token.json

3.2 结果分析

Cascade 推测采样的实验结果如表 4所示。通过和朴素推测采样的比较可以发现，生成 token 的速度有所下降，但是 pass@1 有所上升。进一步做具体的分析。

目标模型	Llama-2-7b-chat-hf	Llama-2-13b-chat-hf
起草模型	Llama-160m	Llama-160m
平均速度	8.504 token/s	8.257 token/s
pass@1	0.1220	0.1463

表 4: cascade 推测采样结果

- 在配置 1 中，pass@1 指标的上升 (0.0122) 代表 164 题中 Cascade 推测解码比朴素推测解码多做对两道题。可以认为 Cascade 推测解码额外加入了起草模型对于二元组模型产生 token 的修正过程从而提升了回复质量。考虑到 humaneval 本身测试样例的局限性，也可以认为两者的效果并没有显著差别。在配置 2 中，pass@1 相比于基线没有变化。

- 相比于实验一的推测采样，平均速度有所下降，这与 Cascade 推测解码的算法原理相对更为复杂有关。其分为 Vertical Cascade 和 Horizontal Cascade.

  – 在 Vertical Cascade 上通过使用更小的草稿模型来协助草稿生成，并用原始草稿模型来审查更小草稿模型的生成内容。引入的更小的草稿模型虽然参数量更小，但是该步相比于常规的推测采样增加了原始草稿模型对于更小的草稿模型的审查过程以及递归调用的耗时。

– 在 Horizontal Cascade 上根据草稿模型生成的 token 被目标模型接受的概率不同，合理分配生成 token 的时间。将最大的草稿模型用于生成第一个草稿 token，随着新草稿 token 被接受的概率降低，逐渐使用更小的模型进行生成，以减少生成不重要草稿 token 的时间成本。这涉及到多个模型切换、协调和通信，对草稿 token 进行拒绝采样的额外计算和指定草稿模型的决策计算，递归回溯修改草稿 token 等过程，都会增加程序的 walltime 耗时。

参考文献

[1] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[2] Ziyi Chen, Xiaocong Yang, Jiacheng Lin, Chenkai Sun, Kevin Chen-Chuan Chang, and Jie Huang. Cascade speculative drafting for even faster llm inference, 2024.

[3] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

[4] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding, 2023.