Token-level improvement for LLM reasoning

本文介绍一下近期几篇在token级，针对大模型推理任务的一系列工作，文章列表如下：
Chain-of-Thought Reasoning Without Prompting：采样策略优化，解码时采样一些低排名的token，可以引出思维链
Top-nσ: Not All Logits Are You Need：采样策略优化，pre-softmax 阶段截取 logits 的 top-nσ 部分，大大提升高温采样时解码的准确率
Not All Tokens Are What You Need for Pretraining：预训练阶段优化，通过训练一个参考模型，对预训练语料做 token 级别的过滤，排除噪声token的干扰
Critical Tokens Matter: Token-Level Contrastive Estimation Enhence LLM's Reasoning Capability：
DPO算法+采样策略优化，发现有些关键token容易导向错误推理路径，提出token-level的偏好学习算法 cDPO 来消除关键 token

Chain-of-Thought Reasoning Without Prompting (2024.02)

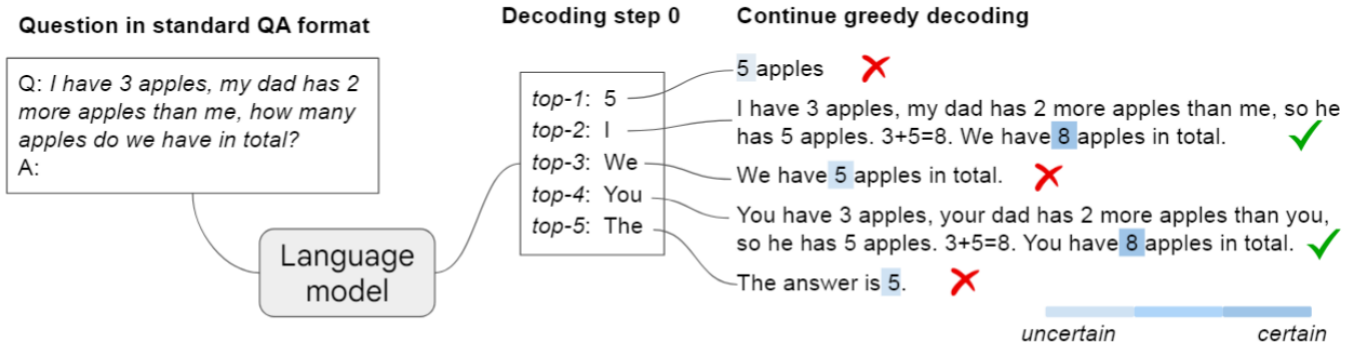


Figure 1 | **Illustration of CoT-decoding**. Pre-trained LLMs are capable of inherent reasoning without prompting by considering alternative top-k tokens, rather than solely relying on the top-1 greedy decoding path. Moreover, these models tend to display higher confidence in decoding the final answer (indicated by a darker shaded color) when a CoT reasoning path is present.

本文发现预训练后的LLM（没有经过专门的COT SFT）可以在 zero-shot 的情况下，通过在第一个token上进行多次解码尝试，后续token使用普通的贪心解码算法就可以引导模型自动输出COT推理（如上图中的top-2与top-4 token）。同时，文章也指出如果在第一个token上就直接使用贪心解码，模型倾向于直接输出问题答案，容易导致输出错误回答。
文章的第二点发现，当模型输出COT之后再输出答案，此时模型赋予答案token的置信度要高于直接回答时赋予答案的置信度。
基于基于以上两点观察，本文提出了一个简单解码策略 **CoT-decoding**，第一个token选取 top-k 个候选token，随后根据这些候选 token 生成 k 条answer，然后依据下式计算每个answer中答案token的置信度：

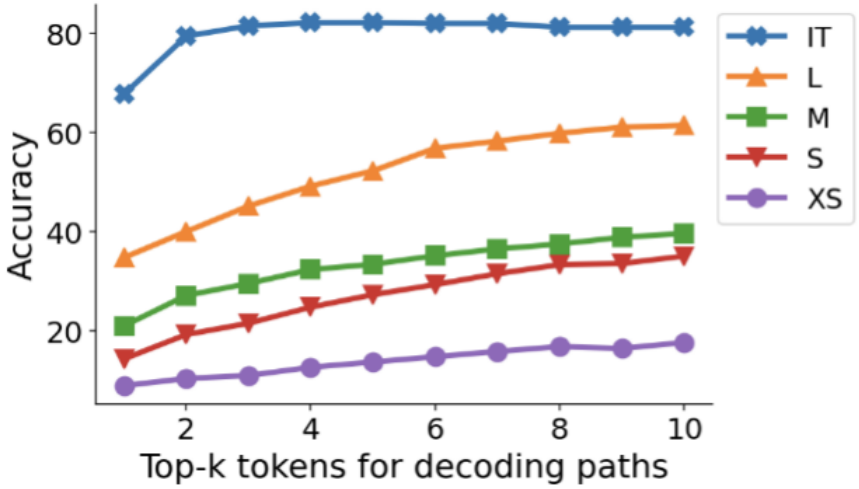
$$\Delta_{k,\text{ answer}} = \frac{1}{|\text{ answer }|} \sum_{x_t \in \text{ answer}} p\left(x_t^1 \mid x_{<t}\right) - p\left(x_t^2 \mid x_{<t}\right).$$

其中 x_t^1, x_t^2 指模型在第 t 个token位置上，post-softmax概率 top-2 的tokens。上面的式子衡量了概率最高的前两个token的概率之差，这个值越大，模型对该答案也就越自信。CoT-decoding 使用这个指标作为权重，对 k 条answer做加权投票，以此选出最终答案。

Table 4 | CoT-decoding is the only decoding strategy that can significantly enhance language models' reasoning.

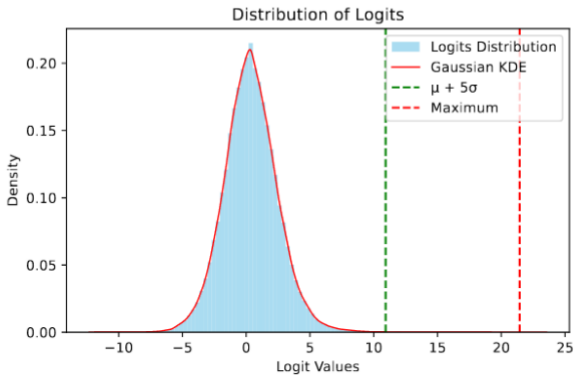
	GSM8K Acc
Top-k sampling ($k = 10$)	4.9%
Top-p / Nucleus sampling ($p = 0.9$)	6.4%
Beam search ($b = 10$)	6.7%
Temperature sampling ($T = 0.7$)	7.5%
Greedy decoding	9.9%
Self-consistency w/o CoT prompt (10 paths)	12.9%
CoT-decoding ($k = 10$)	**25.1%**

基于Mistral-7B的实验结果表明 CoT-decoding 在GAM8K上的性能远高于传统的解码策略以及不使用few-shot prompt 的 Self-consistency。作者也做了实验验证 CoT-decoding 中超参数k应该如何设置，实验结果如下

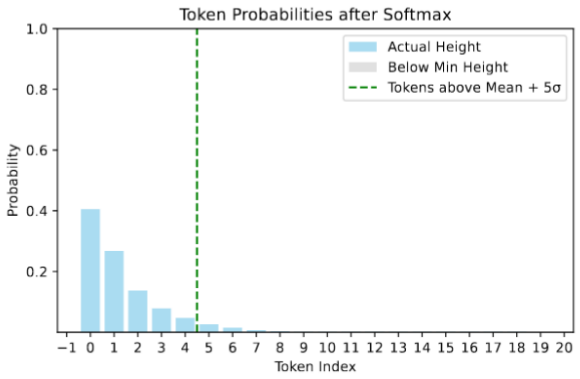


可以发现对于base model（未经过Instruct-tunning，IT），随着 k 增大，模型的准确率在不断提升，意味着模型本身拥有输出CoT的能力，但未经IT微调能诱发COT的 token 的概率排名较低。图中IT对应指令微调后的模型，可以看出 k 增大到2之后，模型的准确率几乎不再增加，说明指令微调使得诱发COT的token的排名提升，不需要探索更多token就可以输出COT。

Top-nσ: Not All Logits Are You Need (2024.11)



(a) Distribution of logits



(b) Descendingly sorted Probabilities. Only the top 20 tokens are shown.

Figure 1: Distribution of logits and descendingly sorted probabilities of LLaMA3-8B-Instruct on an AQuA sample. Note that the leading tokens in the right plot (with higher probabilities) correspond to the right-side region of the logits distribution. The maximum logit is approximately 10σ above the mean of the distribution.

作者可视化了llm最后一层输出的 logits value（张量尺寸 = (bacth_size, seq_len, vocab_size)）的分布，如上图可见 logits value 近似服从均值为0的高斯分布，然而在经过softmax操作之后，占logits value绝大部分的值都被映射到0概率，少量离群值（左图中横轴[10, 20]）经过softmax之后成为了主导 post-softmax 分布的token。由此作者假设，logits value 中存在噪声域以及信息域，分别对应高斯分布主体以及离群点，作者希望在 logit 空间找到一个决策边界划分开噪声与信息，从而避免post-softmax分布被噪声干扰。

传统采样策略的问题--概率空间截断

传统采样策略都是在设置一个概率阈值p，截断概率小于p的token，以近似真实的概率分布。

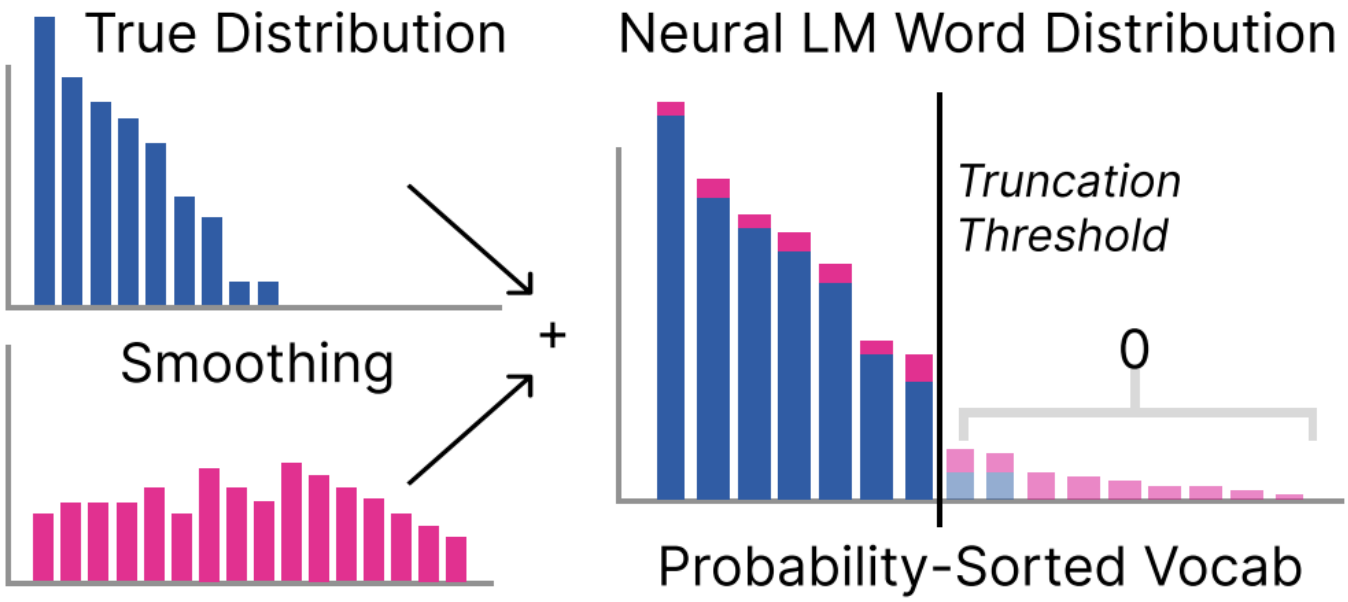
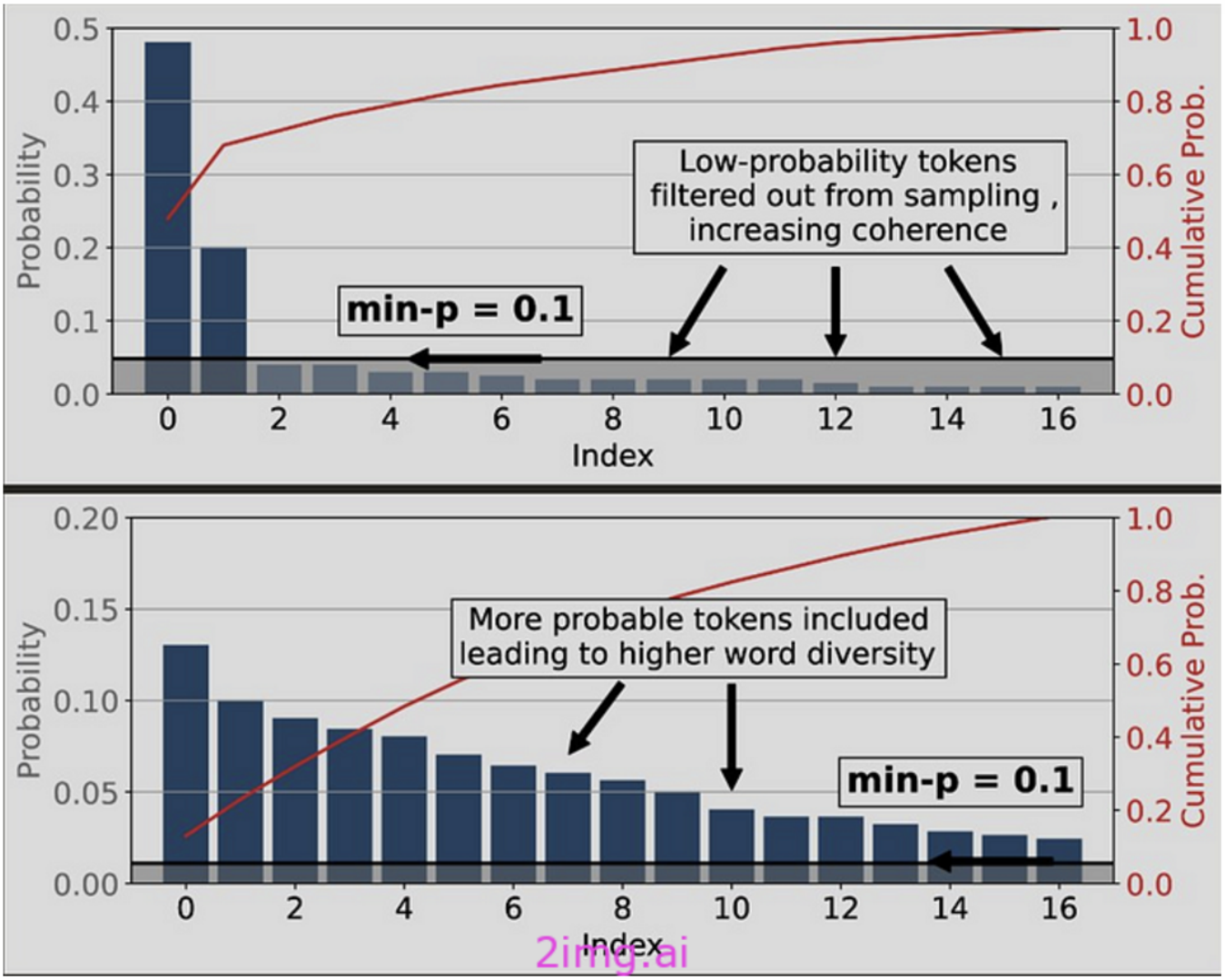


Figure 1: A neural LM as a mixture of the true distribution, and a uniform-like smoothing distribution. Truncation aims to approximate the true distribution support.

上图出自Truncation Sampling as Language Model Desmoothing (2022.10)一文，文章提出一个语言模型输出的分布，实际上是真实分布混合了一个类均匀分布的平滑分布，所以在概率空间做截断，相当于近似恢复真实分布。（本文作者指出，上述认为输出分布为真实分布与噪声分布混合的观点是正确的，但是噪声分布并不是类似均匀分布，而是服从对数正态分布，即在 logit 空间服从正态分布）
基于这样的观点，传统的采样算法都在设计各种启发式规则在概率空间对输出分布进行截断。像top-p之类的核采样算法，通过设定截断阈值为 p 来动态地在 post-softmax 空间进行截断，然后在截断后的采样池中进行随机采样，然而 post-softmax 空间并不能很好的排除噪声的干扰，例如当正确 token 被赋予0.87的概率，而此时 p 设为0.9，由此又将3个低概率的带噪 token 加入采样池，则原本87%的正确率会立马下降到25%。所以Top-p可能不适合数学推理等准确性要求较高的任务。
近期在数学与常识推理任务上取得显著成果的算法 min-p，同样在 post-softmax 空间通过依照最大概率值来动态截断，概率阈值= p_max * p，例如下图 p = 0.1 :



> - 当面对事实、数学或选项较窄的事件时，顶部分布更为常见
> - 底部分布在创意写作等任务中更为常见，我们期望模型具有创造力。
>
> 长话短说，虽然 top-p 采样选择了“最有可能的一个”，但**min-p 实际上考虑了分布的结构**，在结果明显时激励输出高度可能的标记，并且在分布具有长尾的情况下不会影响模型的创造力。

本文作者将上述方法在概率空间选定的阈值投影回 logit 空间，发现这些方法虽然声称根据模型输出分布自适应地截断，但是在 logit 空间仍然只是在做静态截断。
这就导致了在高温采样的情景下，由于 logits 值被除以一个大的温度系数T，整个 logits 分布实际在被压缩，原本高 logit value 的信息域被向低 logit value 的噪声域挤压，若仍使用静态截断可能会导致包含进许多噪声，从而降低高温采样下的准确性与稳定性。

在 logit 空间实现真正的动态截断

出于以上观察，论文提出 Top-nσ 算法，即在 logit 空间将 logit value 小于 M-nσ（M为最大的logit值，σ为标准差）的 logit 置为负无穷，从而在后续的 softmax 操作中被赋予0概率。算法流程如下：

Algorithm 1 Top-$n\sigma$ Sampling

Input: Input context x, temperature T, threshold multiplier n
Output: Next token
Compute logits $l = \text{LLM}(x)$
Scale logits: $l' = l/T$
Calculate $M = \max(l')$ and $\sigma = \text{std}(l')$
Create mask: $m_i = \begin{cases} 1 & \text{if } l'_i \geq M - n\sigma \\ 0 & \text{otherwise} \end{cases}$
Apply mask: $l'_i = \begin{cases} l'_i & \text{if } m_i = 1 \\ -\infty & \text{otherwise} \end{cases}$
$p = \text{softmax}(l')$
Sample token from distribution p

该操作实现了截断范围与温度系数解耦，即不论温度如何设置，Logit空间的截断范围会依据标准差而动态变化（温度系数影响的实际上是logits分布的标准差），所以实现了高温采样下也能准确过滤无关噪声。相关数学证明以及在信息域服从均匀分布或高斯分布两种假设下该算法的行为讨论见第3节。
与温度解耦后，通过设置合理的参数n，来决定一个划分有效域与噪声域的边界，在有效域内除以高温系数来平滑有效域分布，从而促进有效的探索。

Not All Tokens Are What You Need for Pretraining (NeurIPS 2024 oral)

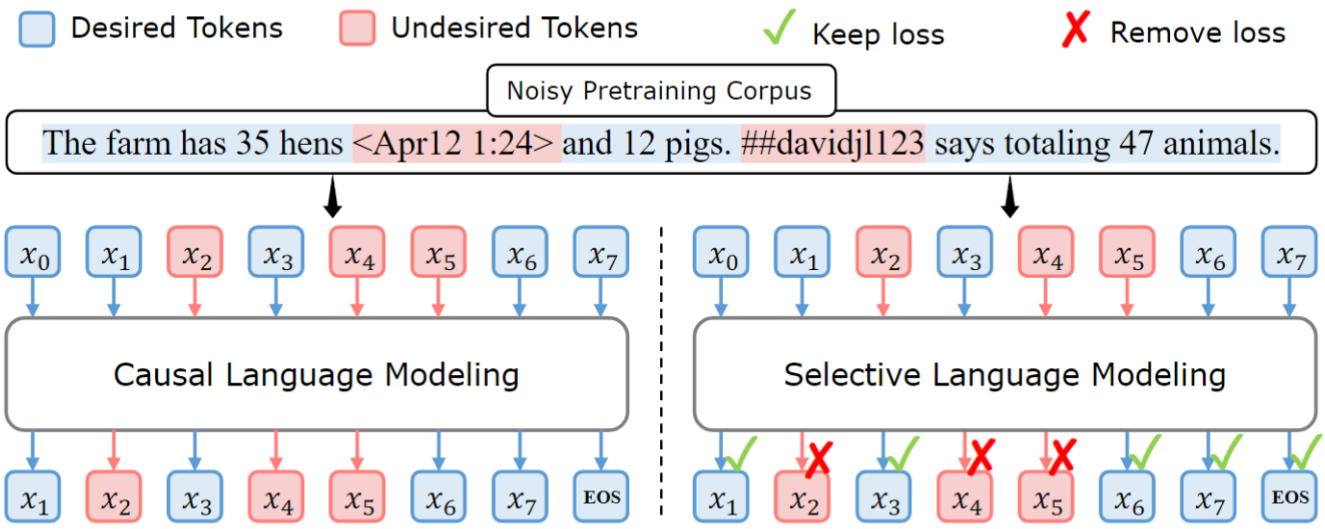


Figure 2: **Upper:** Even an extensively filtered pretraining corpus contains token-level noise. **Left:** Previous Causal Language Modeling (CLM) trains on all tokens. **Right:** Our proposed Selective Language Modeling (SLM) selectively applies loss on those useful and clean tokens.

本文提出LLM的预训练预料中充斥着大量噪声数据，并且这些数据仅靠文档级别的数据清洗是无法过滤掉的。所以文章提出了一种 token-level 噪声数据过滤的训练方法 SLM（Selective Language Modeling），具体为通过事先训练一个reference model (RM)，利用RM来对数据进行token级别的打分，再利用打分后的数据进行预训练，并且不计算被视为噪声token的损失，由此大大提高了预训练的效果。

Selective Language Modeling（SLM）

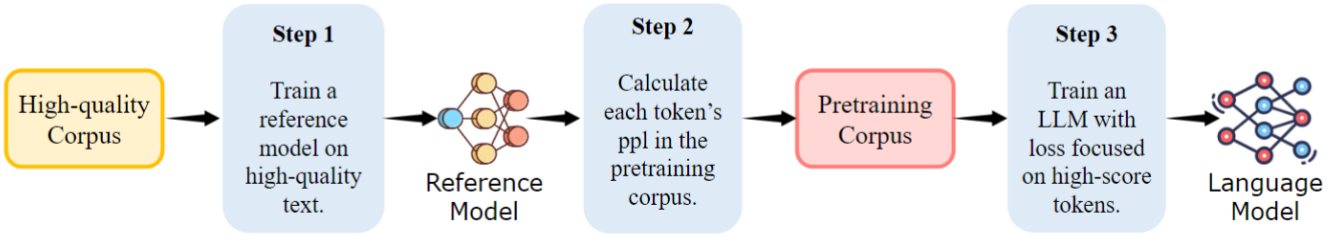


Figure 4: **The pipeline of Selective Language Modeling (SLM).** SLM optimizes language model performance by concentrating on valuable, clean tokens during pre-training. It involves three steps: (Step 1) Initially, train a reference model on high-quality data. (Step 2) Then, score each token's loss in a corpus using the reference model. (Step 3) Finally, selectively train the language model on tokens that have higher scores.

SLM的核心思想是事先利用高质量、低噪声的数据集训练一个参考模型，参考模型由此能反映高质量的目标数据集分布。随后我们可以利用该参考模型在预训练语料中的每一个token计算预训练数据的参考损失 $\mathcal{L}_{RM}$：

$$\mathcal{L}_{RM}(x_i) = -\log P(x_i \mid x_{<i})$$

拥有了参考损失，我们可以计算当前训练数据分布与期望数据分布的差异，文中称 excess loss：

$$\mathcal{L}_{\Delta}(x_i) = \mathcal{L}_{\theta}(x_i) - \mathcal{L}_{RM}(x_i)$$

基于 excess loss 可以将预训练数据中与期望分布相差过大的 token 筛选出来，文章发现这些 token 一部分来自自定义的符号、无意义的胡言乱语以及表格、参考文献的特殊字符；另一部分来自常用连词、词缀以及标点符号。后者的波动属于正常现象，但前者包含过多噪声不应计入loss计算。于是文章提出使用一个评分函数为每一个batch中的token评分，然后仅计算 top-k% 的 token 的 loss，其余 token 视为噪声 token，其 loss 被置 0。在评分函数的选择上，默认使用上面提到的 excess loss，形式化的表述为：

$$\mathcal{L}_{SLM}(\theta) = -\frac{1}{N * k\%} \sum_{i=1}^{N} I_{k\%}(x_i) \cdot \log(P(x_i \mid x_{<i}; \theta))$$

其中 $I_{k\%}(x_i)$ 为指示函数，

$$I_{k\%}(x_i) = \begin{cases} 1 & \text{if } x_i \text{ ranks in the top k\% by } S(x_i) \\ 0 & \text{otherwise} \end{cases}$$

$S(x_i)$ 为评分函数，后文中还讨论了除 excess loss 外，其他的评分函数选择。总之，基于上述$\mathcal{L}_{SLM}(\theta)$ 对LLM进行预训练，便可以动态地淘汰低质量 token，对齐参考模型所表征的高质量数据分布。

Self-Reference

但是高质量的训练数据并不是十分容易获得的，很多现实场景下我们其实没办法收集足够的高质量数据来训练参考模型。为了解决这一问题，作者提出 Self-Reference，顾名思义即直接用当前预训练数据训练参考模型，然后通过改变评分函数，利用当前数据上训练的模型来过滤低质量 token 。

由于，参考模型使用了同样的数据训练，所以可以视为训练过一轮的目标模型，此时可以计算参考模型的对每个 token 的 loss $\mathcal{L}_{RM}$（即负对数概率）或者信息熵 $\mathcal{H}_{RM}$，计算方法为：

$$\mathcal{H}_{\mathrm{RM}}(x_i) = -\sum_{k=1}^{V} P(t_k \mid x_{<i}) \log P(t_k \mid x_{<i})$$

上面两种指标均是表示了参考模型对该 token 的不确定程度。这么做的动机在于，经过一轮预训练后模型仍很不确定的 token 可能是一些没有意义的乱码噪声，很难预测，所以可以将其过滤。

作者在大量实验上证明了即使没有使用高质量数据训练RM，而是用Self-reference的方法训练RM，并基于不确定性过滤，仍然可以取得性能提升，只是没高质量数据方案的提升大。注意一下，这里作者提出过滤低对数概率 token 的做法似乎与我们过滤高对数概率正例的想法有些相悖，但其实并不冲突，因为本文主要关注的是预训练阶段，该阶段使用海量语料预训练，大部分训练数据质量堪忧，语料中充斥着大量符号、代码、乱码，但该阶段的目标只是让模型在海量数据中摸索学习语言规律，建立初步的世界知识，由于有意义的语段会在海量数据中高频出现，所以模型会在这些数据中学会预测有意义的 token，但是随机的噪声段出现频率不会很高，所以我们有信心说经过一轮预训练，还预测不出来的 token 很有可能就是噪声。而我们的想法是聚焦于模型的迭代微调阶段，这时主要矛盾已经变为防止模型在自己高频输出的例子上过拟合，当然此阶段的模型时不时也会输出低对数概率的乱码回复，不过我们很容易就可以通过验证最终答案将其过滤。

Critical Tokens Matter: Token-Level Contrastive Estimation Enhence LLM's Reasoning Capability (2024.11)

本文假设存在一些关键 token 容易将 LLM 导向错误的答案，于是做了个实验验证，作者使用 Llama-3-8B 在 GSM8K 上收集了100条错误的推理路径，然后在每条的每个 token 处重采样64次，找到第一个64次都全错的 token 。然后迫使模型在该步不采样这一 token，然后重新采样 k 次，实验结果表明 pass@k 显著增长。
证实了关键 token 的存在，接下来的任务是自动找出关键 token。作者设计了一个策略，分别使用推理产生的负例和正例 SFT 两个模型，然后计算正负模型对每个 token 的“分歧”，分歧最大的 token 可能就是关键 token，具体来说用下式计算每个 token 的得分 s_t：

$$\log s_t = (1+\beta)\log p^+(y_t \mid x, y_{<t}) - \beta \log p^-(y_t \mid x, y_{<t}) - logZ$$

其中 Z 是 softmax 中的配分函数（分母）。不难看出 s_t 越小，说明正模型对该 token 的置信度低，而负模型对该 token 置信度高，该 token 很可能就是导向错误答案的关键 token。
既然能自动识别关键 token 了，那就该想办法微调模型，降低这些 token 的出现几率。于是作者提出了 Aligning LLMs with critical tokens 的流程

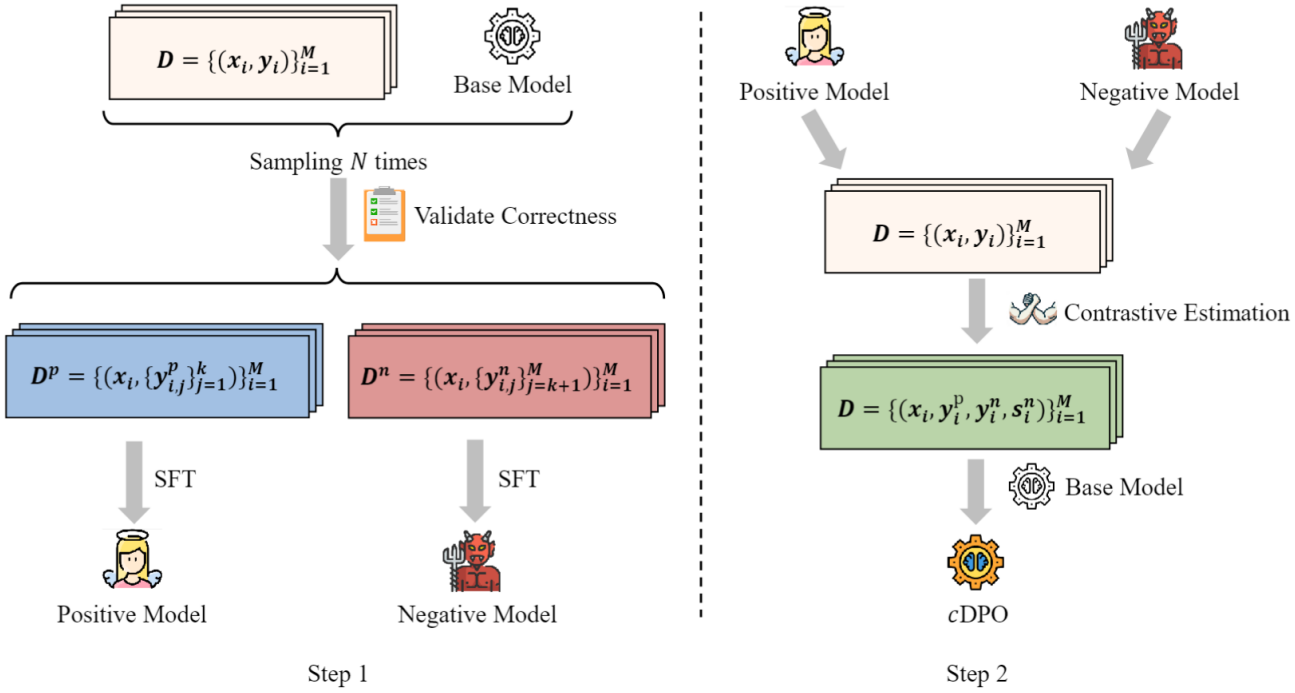


Figure 4. Overview of aligning LLMs with critical tokens. The pipeline consists of two steps: (1) fine-tuning positive and negative models on correct and incorrect reasoning trajectories, and (2) applying contrastive estimation for *c*DPO.

我们首先训练正、负模型，然后利用正、负模型来收集一个 token-level 的偏好数据集，具体来说对于一个prompt x_i ，随机挑选一条正确响应 y_i^p 作为正例，挑选一条最频繁出现的错误响应 y_i^n 作为负例，然后我们在负例上运用上文提到的方法为每一个 token 计算 s_t ，从而获得 $s_i^n = [s_{i,1}^n, s_{i,2}^n, \cdots, s_{i,T_i}^n]$，其中 T_i 为响应 y_i^n 的长度。
在得到偏好数据集之后，作者提出一种 cDPO 算法，想要将原本 sample-level 的偏好学习扩展到 token-level，具体来说其修改了原始DPO的损失函数中负例的奖励项：

$$\mathcal{L}_{\text{cDPO}} = -\sum_{i=1}^{M} \log \sigma \left(r(x_i, y_i^p) - r_s(x_i, y_i^n, s_i^n) \right)$$

其中，$r(x_i, y_i^p) = \gamma \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}$，$r_s(x_i, y_i^n, s_i^n) = \gamma \sum_{t=1}^{T} (1 - s_t) \log \frac{\pi_\theta(y_t|x, y_{<t})}{\pi_{\text{ref}}(y_t|x, y_{<t})}$ ，其实就是在原始DPO的负例奖励项上做了一个 token-level 的加权，权重为 $(1 - s_t)$ ，旨在惩罚 s_t 值较小的关键的 token。但个人感觉这样并没有做到真正的 token-level 的偏好学习，因为每个 token 的损失还是聚合为 sample 级别后再计算损失，这样只是会惩罚含有关键 token 的响应（或者说惩罚正负模型分歧较大的负例），而不能精准地识别并 ban 掉关键 token。

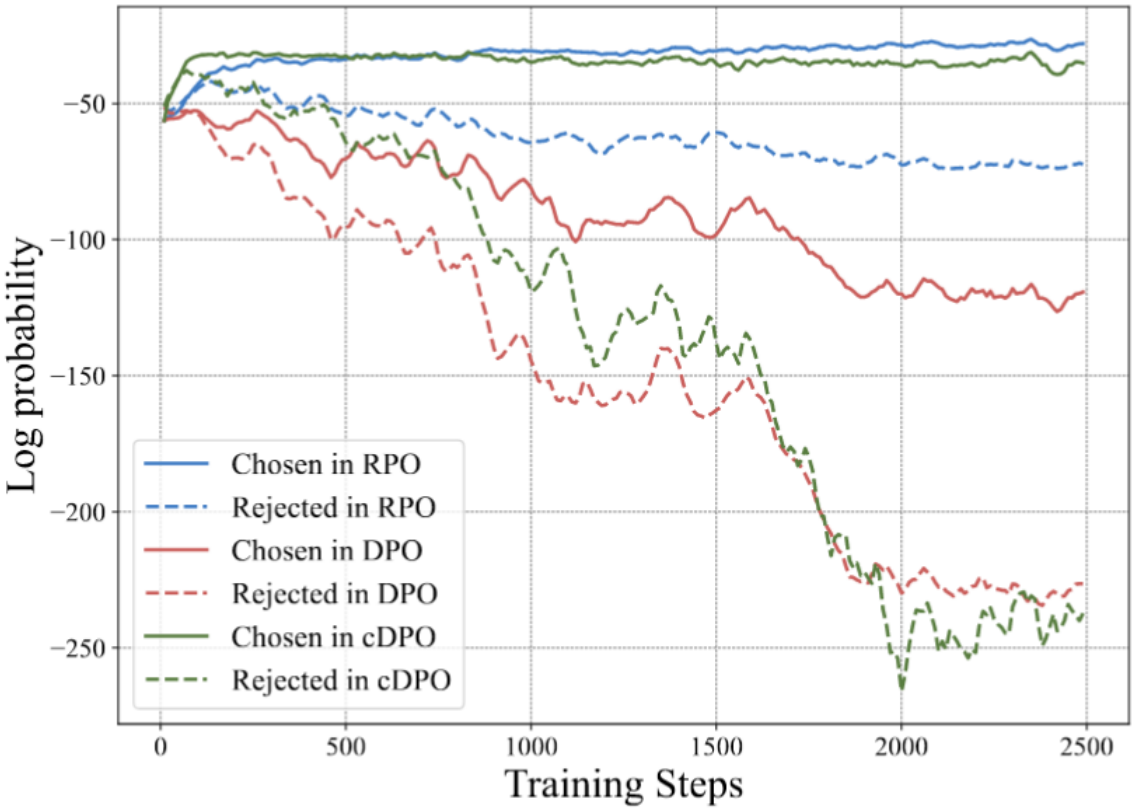


Figure 6. Log probabilities of chosen and rejected sequences during training on the GSM8K dataset using DPO, RPO, and *c*DPO. The solid lines represent chosen sequences, while the dashed lines represent rejected sequences. The figure demonstrates how *c*DPO achieves a greater separation between chosen and rejected sequences compared to DPO and RPO.

实验结果显示，cDPO避免了DPO降低正例的缺点，在保证增强正例的同时，大幅削弱了负例，可以看出随着训练进行cDPO正、负之间的间隙越来越大。

Method	**GSM8K**				**MATH500**			
	Llama-3		*DeepSeek*	**Avg.**	*Llama-3*		*DeepSeek*	**Avg.**
	8B	*70B*	*math-7B*		*8B*	*70B*	*math-7B*	
Baseline	56.4	80.4	64.1	67.0	16.8	42.2	31.4	30.1
+ SFT	61.2	82.1	67.1	70.1	17.2	43.0	32.6	30.9
+ DPO (Rafailov et al., 2024)	59.7	87.8	66.5	71.3	17.0	41.2	33.4	30.5
+ TokenDPO (Zeng et al., 2024)	62.3	83.3	69.6	71.7	17.8	42.2	32.4	30.8
+ DPO (Rafailov et al., 2024)	59.6	88.9	63.1	70.5	15.4	39.8	33.0	29.4
+ RPO (Pang et al., 2024)	67.5	89.7	68.9	75.4	18.4	43.8	34.8	32.3
+ *c*DPO (Ours)	**67.9***	**90.8***	**72.9***	**77.2***	**19.6***	**45.6***	**35.0***	**33.4***

Table 1. Experimental results on GSM8K and MATH500 datasets. Our proposed method surpasses all the strong baselines at a large margin on individual settings and average performance. * denotes the significance test where $p < 0.005$.