Paper shredding: InternML2

关于本轮存在的疑惑

1. 本文档 Page 21 关于 COOLRLHF 为了防止过拟合只训练一个周期是否不妥？
2. 本文档 Page 23 这里的 lambda 指的是损失函数的 lambda 参数？也就是几乎所有权重都由惩罚函数提供，那么前者排序损失函数的意义岂不是变得很小？

1. Introduction

InternLM2 extensively details how it prepares text, code, and long-context data for pre-training.
InternML2 数据预处理主要包含三部分： 文本，代码，长文本（包含强位置关联性）的数据。

> How to effectively extend the context length of LLMs is currently a hot research topic, since many downstream applications, such as Retrieval-Augmented Generation (RAG) (Gao et al., 2023) and agents (Xi et al., 2023), rely on long contexts. InternLM2 first employs Group Query Attention (GQA) to enable a smaller memory footprint when inferring long sequences. In the pre-training phase, we initially train InternLM2 with 4k context texts, then transit the training corpus to high-quality 32k texts for further training. Upon completion, through positional encoding extrapolation (LocalLLaMA, 2023), InternLM2 achieves commendable performance in the "Needle-in-a-Haystack" test within 200k contexts.

延长文本长度：使用 GQA（分组队列注意力机制，一种注意力机制的变体）用以长序列推断，并采用位置信息外扩法进行位置信息的建模和调参，最终用以实现“大海捞针”，即长文本中的语义理解和信息关联问题。

> Following long-context pre-training, we utilize supervised fine-tuning (SFT) and reinforcement learning from human feedback (RLHF) to ensure the model adheres well to human instructions and aligns with human values. Notably, we also construct corresponding 32k data during these processes to further improve the long-context processing capability of InternLM2. Besides, we introduce **CO**nditional **On**Line RLHF (COOL RLHF), which adopts a conditional reward model to reconcile diverse but potentially conflicting preferences and executes Proximal Policy Optimization (PPO) over multiple rounds to mitigate emerging reward hacking in each phase. To elucidate the impact of RLHF within the community, we

使用监督调优（SFT）和人类回馈强化学习（RLHF）帮助模型更好地与人类所给出的 instructions 和价值观念相符。

2. Contributions

> Our contributions are twofold, highlighting not only the model's exceptional performance across diverse benchmarks but also our comprehensive approach to its development in different stages. Key points include

本文的两方面成就：提出新的模型，以及阐释不同层级的实现方法，其主要包含四个方面：

1. **Open-Sourcing InternLM2 with Exceptional Performance:** We have open-sourced models of various sizes, including 1.8B, 7B, and 20B, all of which perform well in both subjective and objective evaluations. Additionally, we have released models from different stages to facilitate community analysis of changes post-SFT and RLHF training.

2. **Designed with a 200k Context Window:** InternLM2 exhibits impressive long-context performance, nearly perfectly identifying all "needles" in the "Needle-in-a-Haystack" experiment with a 200k context. Furthermore, we provide experience of training long-context LLMs across all stages, including pretraining, SFT, and RLHF.

200K 长度 token 中的语义窗口。

3. **Comprehensive Data Preparation Guidance:** We have elaborated on the preparation of data for LLMs, including pre-training data, domain-specific enhancement data, SFT data, and RLHF data. These details will benefit the community in better training of LLMs.

数据清洗和预处理方法。

4. **Innovative RLHF Training Techniques:** We introduced Conditional Online RLHF (COOL RLHF) to harmonize various preferences, significantly enhancing the performance of InternLM2 in various subjective dialogue evaluations. We have also conducted a preliminary analysis and comparison of RLHF's subjective and objective results to offer the community insights into RLHF.

COOLRLHF：在线条件强化学习，用以在不同场景下泛化模型表现。

Infrastructure

InternEvo

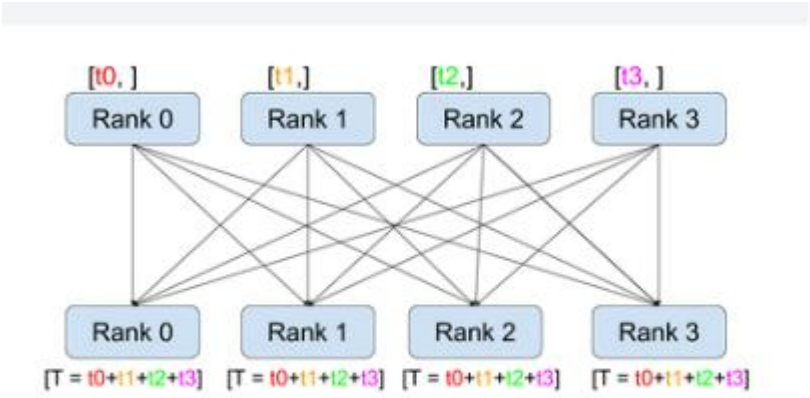
Reducing Communication Overhead A trade-off exists between memory utilization and communication cost in distributed LLM training. Initially, the communication cost can be effectively reduced by diminishing the communication scale. This involves limiting communications to a smaller group of GPUs, potentially within the same node, which mitigates the overall communication cost. Building upon this principle, InternEvo addresses communication challenges by implementing a suite of adaptive sharding techniques to achieve strong scaling performance (Chen et al., 2024b). These include Full-Replica, Full-Sharding, and Partial-Sharding, which allow each component of the model states—parameters, gradients, and optimizer states—to independently select the most appropriate sharding approach and device mesh configuration. This flexibility facilitates a more nuanced distribution of model states across the GPU infrastructure. InternEvo also introduces an optimization framework designed to identify the most efficient sharding factors. This aims to minimize communication expenses while adhering to the memory constraints of the GPU.

减少通信负载：通常需要调整通信规模，即划分为同一节点内组更小的集群 GPU。InternEvo 采用自适应分层方法，包含三种分层模式，将包含参数，梯度，优化器状态在内的模型参数独立进行最优化分层策略。同时也提出了最优化框架用以设计和调整分层参数。

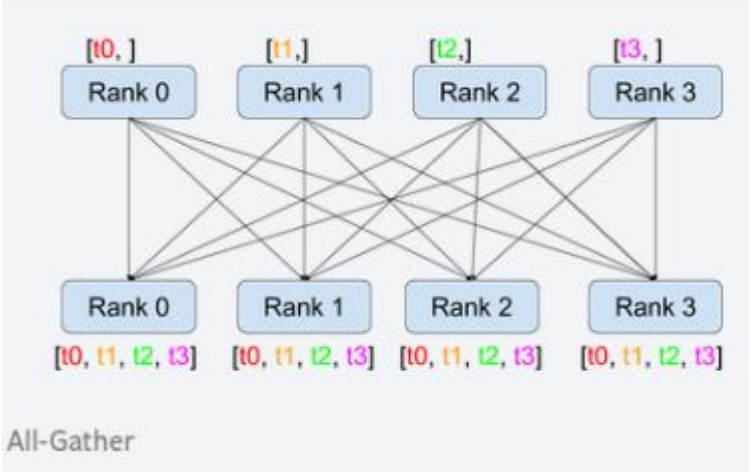
Communication-Computation Overlap Further reducing communication overhead, InternEvo strategically coordinates communication and computation to optimize overall system performance. When employing parameter sharding, the model's entire parameters are distributed across multiple GPUs to conserve GPU memory. During each forward and backward pass of every micro-batch in a training step, InternEvo efficiently pre-fetches the complete parameter set for upcoming layers through AllGather, while concurrently computing the current layer. The generated gradients undergo synchronization within the parameter sharding group through ReduceScatter and are subsequently synchronized across parameter sharding groups using AllReduce. These communication processes are skillfully overlapped with the backward computation, maximizing efficiency in the training pipeline. In the case of optimizer state sharding, where the GPU broadcasts updated parameters within the sharding group through Broadcast, InternEvo employs a strategic overlap with the forward computation of the next training step. These innovative overlap approaches effectively balance communication overhead and computation execution time, resulting in a significant enhancement in overall system performance.

通信-计算重叠：

当使用参数分片时，模型的整个参数分布在多个 GPU 上，以节省 GPU 内存。在训练步骤中每个微批的每次正向和反向传递中，inmenEvo 通过 AllGather 有效地预获取即将到来的层的完整参数集，同时计算当前层。生成的梯度通过 ReduceScatter 在参数分片组内进行同步，随后使用 AllReduce 在参数分片组之间进行同步。这些通信过程与反向计算巧妙地重叠，使训练管道中的效率最大化。



All-Reduce

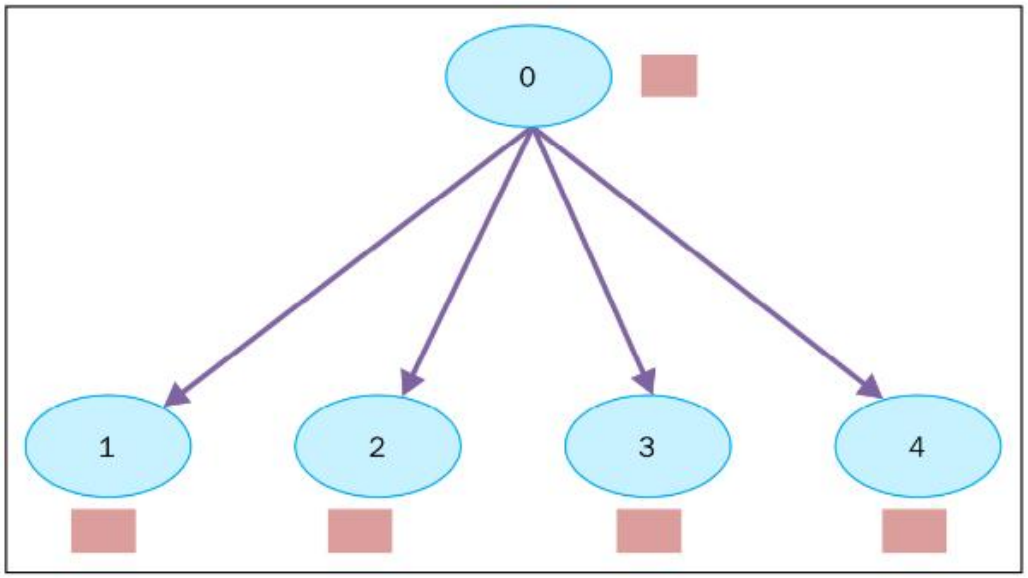


All-Gather

两者都是分布式数据的通讯方式。

Broadcast

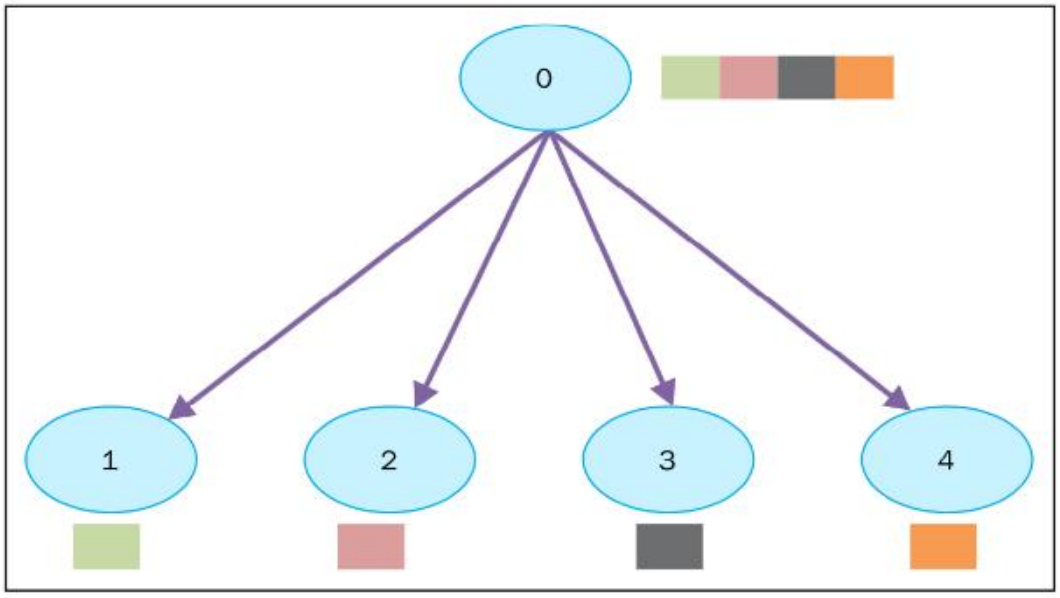
看名字就很好理解了，其实就是把同一份数据分发广播给所有人，示意图如下：



Broadcasting data from process 0 to processes 1, 2, 3, and 4

Scatter

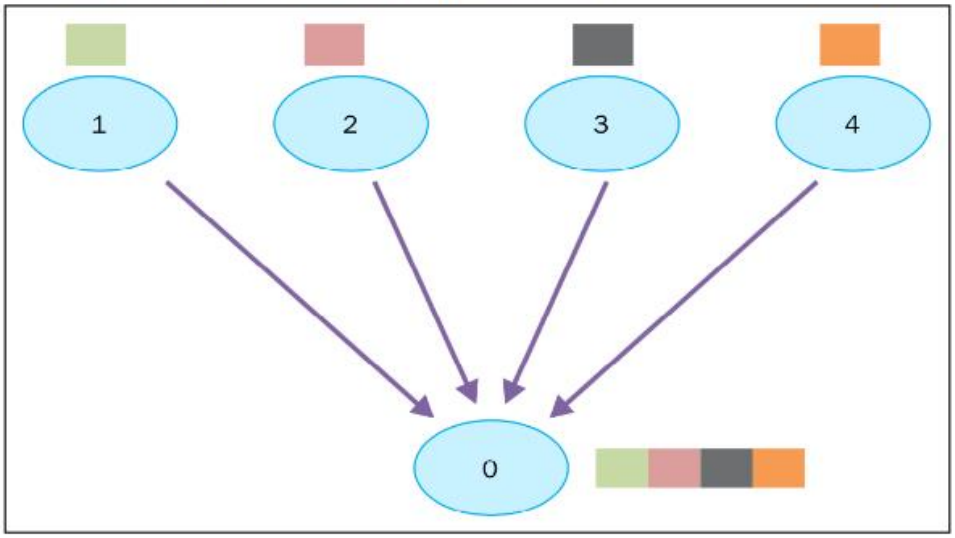
不同于Broadcast， scatter可以将不同数据分发给不同的进程。



Scattering data from process 0 to processes 1, 2, 3, 4

Gather

这个也很好理解，就是把多个进程的数据拼凑在一起。

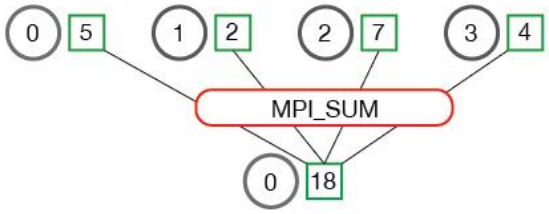


Gathering data from processes 1, 2, 3, 4

Reduce

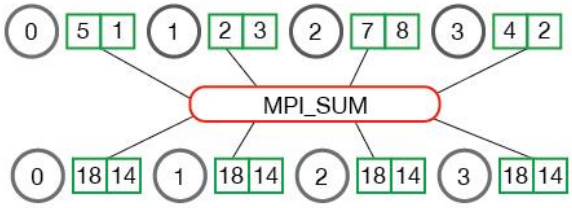
reduce就是将多个进程中的数据按照指定的映射函数进行运算得到最后的结果存在一个进程中，例如下面两个图中的归约操作都是求和，将4个不同进程的数据归约求和后存在了第一个进程中

MPI_Reduce



All-reduce

All-reduce与reduce的区别就在于后者最后的结果是只保存在一个进程中，而All-reduce需要每个进程都有同样的结果。所以All-reduce一般包含scatter操作，所以有时候也会看到reduce-scatter这种说法，其实reduce-scatter可以看成是all reduce的一种实现方式



Long-Sequence Training One of the primary challenges in long-sequence training is the trade-off between computation speed and communication overhead. InternEvo breaks down GPU memory management into a hierarchical space with four parallel dimensions—data, tensor, sequence, and pipeline—and three sharding dimensions—parameter, gradient, and optimizer state (Chen et al., 2024a). We conduct a thorough analysis of memory and communication costs for each dimension, utilizing an execution simulator to identify and implement the optimal parallelization strategy. The optimal execution plan can be automatically searched based on the training scale, sequence length, model size, and batch size. With this execution plan, InternEvo exhibits the capability to handle long contexts (up to 1 million tokens) during training. InternEvo also implements memory management techniques to reduce GPU memory fragmentation, a common issue in long-sequence training scenarios. It uses a memory pool for unified memory management and introduces a defragmentation

长序列训练涉及到计算速度和通讯成本之间的权衡，InternEvo 将 GPU 内存分为四个平行维度：数据，张量，序列和流水线。分块维度分成三个：参数，梯度和优化器状态。通过分析总体内存和通讯成本构建最优化平行策略，用来解决长文本训练时的内存溢出问题。

Fault Tolerance We have also addressed the challenges of efficiently training LLMs in GPU datacenters, which often face issues such as frequent hardware failures, complex parallelization strategies, and imbalanced resource utilization. To tackle these issues, we conducted an in-depth characterization study of a six-month LLM development workload trace from our GPU datacenter (Hu et al., 2024). This study identified discrepancies between LLMs and previous deep learning workloads and explored resource utilization patterns and job failure impacts. Based on our analysis, we introduced two system efforts: a fault-tolerant pretraining system that enhances fault tolerance through LLM-involved failure diagnosis and automatic recovery, and a decoupled scheduling system for evaluation tasks that provides timely model performance feedback. In our implementation, we have incorporated an asynchronous saving mechanism that regularly archives model weights and optimizer states to distributed file and object storage at predefined intervals. Throughout the training process, each GPU first saves its model states in local storage and subsequently asynchronously uploads this data to the remote distributed storage system. This dual-step process ensures that, in the event of occasional hardware or network failures automatically detected by our system, only a minimal amount of training progress is lost. To optimize storage space, we systematically transfer these temporary model checkpoints from hot storage to cost-effective cold storage. Furthermore, our system is designed to seamlessly resume model training even when the parallelization configuration is altered, providing flexibility and continuity in the training pipeline.

报错恢复：针对可能出现的硬件问题和资源使用不平衡进行的改善。通过演剧主要有两个成果：错误容忍和恢复系统，以及分析性能系统。类似联邦学习，GPU 分别训练并存储模型，并异步分发给相邻的数据节点，通过这种方式可以最小化数据通讯量，并减小出现错误时造成的数据损失。为了优化存储空间，使用冷存储（离线存储，访问频率低但需要硬性记录的数据）进行节约数据。

Interactive Training The efficiency of InternEvo has also been successfully demonstrated in the Reinforcement Learning from Human Feedback (RLHF) stage, where multiple LLMs are deployed for interactive training. For instance, in the Proximal Policy Optimization (PPO) process, we utilize four equally-sized models and train two of them; InternEvo enables each model to be executed at its optimal configuration. To enhance the coordination of multiple models, we have developed an innovative RLHF framework that builds upon InternEvo and Ray. This framework is characterized by its flexibility and scalability, allowing it to perform effectively at scale. It is capable of integrating with various LLM execution engines and supporting diverse algorithmic designs. For a comprehensive description of the "Alignment" concept, please refer to Section 4.

交互训练：融合封装多种 LLM 进行训练用于强化学习。

Model Structure

Transformer (Vaswani et al., 2017) has been predominantly used as the backbone for past Large Language Models (LLMs) due to its excellent parallelization capabilities, which fully leverage the power of GPUs (Brown et al., 2020; Chowdhery et al., 2023; Zeng et al., 2023). LLaMA (Touvron et al., 2023a) builds on Transformer architecture by replacing LayerNorm (Ba et al., 2016) with RMSNorm (Zhang & Sennrich, 2019) and setting the activation function to SwiGLU (Shazeer, 2020), resulting in improved training efficiency and performance. Since the unveiling of LLaMA (Touvron et al., 2023a), the community has vigorously engaged in augmenting the ecosystem built around the LLaMA architecture. This includes advancements in high-efficiency inference (lla, 2023) and operator optimization (Dao, 2023), among others. To ensure our model, InternLM2, aligns seamlessly with this well-established ecosystem, alongside other renowned LLMs, such as Falcon (Almazrouei et al., 2023), Qwen (Bai et al., 2023a), Baichuan (Yang et al., 2023), Mistral (Jiang et al., 2023), we opt to adhere to the structural design principles of LLaMA. In pursuit of enhancing efficiency, we consolidate the W_k, W_q, and W_v matrices, which has resulted in a training acceleration of over 5% during the pre-training phase. Moreover, to better support diverse tensor parallelism (tp) transformations, we have reconfigured the matrix layout. Rather than stacking the W_k, W_q, and W_v matrices in a straightforward manner, we adopt an interleaving approach for each head's W_k, W_q, and W_v, as depicted in Figure 2. This design modification facilitates the adjustment of the tensor parallel size through either splitting or concatenating the matrices along their last dimension, thereby enhancing the model's flexibility in varying distributed computing environments. InternLM2 aims to infer beyond 32K context, there-

基于 LLaMa 模型架构，对 k,q,v 矩阵进行分插来提高运算性能。

Figure 2: Different weight matrix layouts cause different complexity when changing the tensor parallelism (TP) size.

3. Pre-train

3.1 Pre-training data

The pre-training of LLMs is critically shaped by data, which presents a multifaceted challenge. It encompasses handling sensitive data, covering comprehensive knowledge, and balancing efficiency and quality. In this section, we will depict our data processing pipeline for preparing general domain textual data, programming language-related data, and long textual data.

预训练数据集主要分为三部分，分别为普通文本，代码，以及长文本数据。.

3.1.1. Text Data

3.1.1 Text Data

The text data in our pre-training dataset can be categorized by source into web pages papers, patents, and books. To transform these sources into a pre-training dataset, we first standardize all data into a specified format, categorize them by type and language, and store them in JSON Lines (jsonl) format. Then, for all data, we apply several processing steps including rule-based filtering, data deduplication, safety filtering, and quality filtering This results in a rich, safe, and high-quality text dataset.

文本数据预处理流程可以简化为获取数据、标准化数据、分类数据、存储数据、处理数据。

Data Source Distribution We have statistically analyzed the number of documents, data storage size, and data Bytes proportion in the pre-training dataset according to their sources, as shown in Table 1. Among them, the Chinese and English data from web pages account for 86.46% of the total, making it the primary source. Although the data volume from other sources is relatively small, such as books and technical literature(abbreviated as techlit), the average document length is longer and the content quality is relatively higher, making them equally important.

数据分布：主要来源于中英文网页，其他书籍、杂志、报纸等占比较小，但相对质量较高。

Data Process Pipeline The data processing pipeline used in this work is shown in Figure 3. The entire data processing pipeline first standardizes data from different sources to obtain **Format data**. Then, heuristic statistical rules are used for data filtering to get **Clean data**. Next, the Locality-Sensitive Hashing (LSH) method is used for data deduplication to obtain

Dedup data. We then apply a composite safety strategy to filter the data, resulting in **Safe data**. We have adopted different quality filtering strategies for data from various sources, ultimately obtaining **High-quality pre-training data**.

数据处理流程：标准化为标准格式数据、统计学规则清洗为干净数据、位置信息清洗为无冗余数据、安全策略检测为安全数据、质量评估检测为高质量数据。

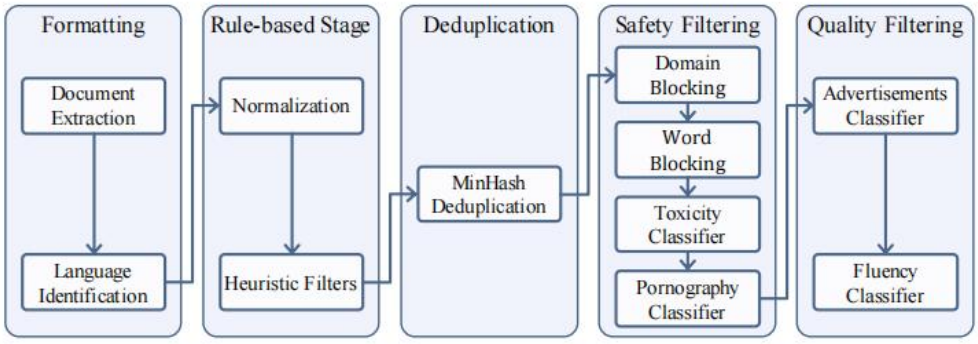


Figure 3: Data Process Pipeline

Data Formatting We will detail the data processing pipeline using web page data as an example. Our web page data mainly comes from Common Crawl[1]. Firstly, we need to decompress the original Warc format files and use Trafilatura (Barbaresi, 2021) for HTML parsing and main text extraction. Then, we use the pycld2[2] library for language detection and classification of the main text. Finally, we assign a unique identifier to the data and store it in jsonl (JSON lines) format and obtained **Format data**.

调库干活，没啥好说的

Rule-based Stage Web page data randomly extracted from the internet often contains a large amount of low-quality data, such as parsing errors, formatting errors, and non-natural language text. A common practice is to design rule-based regularization and filtering methods to modify and filter the data, as seen in Gopher (Rae et al., 2021), C4 (Dodge et al., 2021), and RefinedWeb (Penedo et al., 2023). Based on our observations of the data, we have designed a series of heuristic filtering rules that focus on anomalies in separation and line breaks, frequency of abnormal characters, and distribution of punctuation marks. By applying these filters, we obtained **Clean data**.

大部分网页数据质量很低或带有格式错误，需要进行过滤

Deduplication A large amount of duplicate texts exist on the Internet, which can negatively impact model training. Therefore, we employed a method based on Locality-Sensitive Hashing (LSH) to perform fuzzy deduplication on the data. More specifically, we used the MinHash method (Broder, 1997), establishing signatures with 128 hash functions on the 5-gram of the documents, and using 0.7 as the threshold for deduplication. We aimed to retain the most recent data, that is, prioritizing data with larger CC dumps numbers. We obtained the **Dedup data** after LSH deduplication.

数据去重：位置敏感哈希方法。

Safety Filtering The internet is rife with toxic and pornographic content, the use of which for model training can negatively impact performance and increase the likelihood of unsafe content generation. Therefore, we employed a comprehensive safety strategy combining "domain blocking", "word blocking", "pornography classifier", and "toxicity classifier" to filter the data. Specifically, we constructed a block domain list comprising approximately 13M unsafe domains and a block word list containing 36,289 unsafe words for preliminary data filtering. Given that word blocking might inadvertently exclude a large amount of data, we opted for a cautious approach in compiling the block word list.

To further improve the detection rate of unsafe content, we fine-tuned the BERT model using the "Toxic Comment Classification Challenge" dataset from Kaggle, resulting in a toxicity classifier. We sampled some data from **Dedup data** and annotated it using the Perspective API[3] to create a pornography classification dataset. We then fine-tuned the BERT model with this dataset, yielding a pornography classifier. Finally, we used these two classifiers for secondary filtering of the data, filtering out data with scores below the threshold, resulting in **Safe data**.

安全性：过滤词+BERT 调参

Quality Filtering Compared to sources such as books, papers, and patents, internet-sourced data contains a significant amount of low-quality content. Based on our observations, the primary causes for this low-quality content are twofold: 1. The internet is rife with marketing advertisements, which tend to be repetitive and carry little information. 2. Many web pages consist of lists of article abstracts or product descriptions, resulting in extracted text that is difficult to read and lacks logical coherence.

To filter out these low-quality contents, we first organized manual data annotation. For the advertisements classification task, annotators were asked to identify whether a piece of data contains advertising content (both overall and partial advertisements are marked as low quality). For the fluency classification task, annotators were asked to rate the data on four dimensions: consistency, noise, information content, and grammar, resulting in a comprehensive fluency score. We then fine-tuned the BERT model using the manually annotated data, obtaining an advertisements classifier and a fluency classifier. Finally, we used these two classifiers for secondary filtering of the data, filtering out data with scores below the threshold, resulting in **High-quality pre-train data**.

质量过滤：手动标注+BERT 调参

3.1.2. Code Data

3.1.2 Code Data

Programming is a crucial skill for a LLM, offering support for a variety of downstream applications, such as coding assistance, software development, and building tool-use agents. Moreover, Groeneveld et al. (2024) indicate the possibility of enhancing reasoning capabilities by training on code data, as code is generally well-structured, rigorous, and predictable than natural language.

代码数据通常比自然语言更结构清晰，易于建模和预测（通常来说越接近汇编语言逻辑性和规则性越强，越易于建模）

Data Source Distribution We collect data from various sources, including direct crawling from GitHub, public datasets, and online resources related to coding and programming, like Q&A forums, tutorial sites, and API documentation, the statistics is shown in Figure 4.

Table 2 reflects the data quality assessment based on a scoring model we trained. High-quality data will have a higher sampling weight and can undergo multiple training iterations in the pre-training phase. Moderate-quality data has a normal sampling weight and is typically trained once. Low-quality data are excluded, as our empirical findings affirm that removing them is vital for optimizing model performance and ensuring training stability despite their proportion being relatively small.

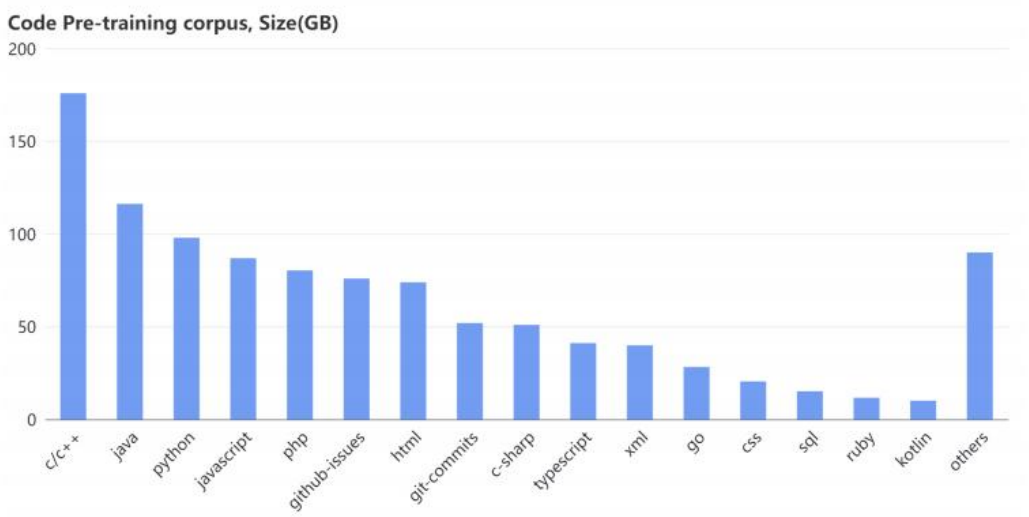


Figure 4: Statistics of code data in our pre-training corpus.

High		Moderate		Low	
Bytes (GB)	Percentage	Bytes (GB)	Percentage	Bytes (GB)	Percentage
105.6	16.8%	440.1	69.9%	83.85	13.3%

代码数据集：根据代码质量进行评分，高质量的代码数据采样权重更高，低质量的数据将会被排除。

Format Cleaning All data is converted to a unified markdown format. Nevertheless, a very small fraction of the data still exhibited corrupted HTML or XML formats. We applied a set of heuristic rules to minimize these occurrences, though we did not invest too much effort in format purification. Markdown was selected for its simplicity—minimizing the token overhead for formatting—and its compatibility with interleaving code and natural language.

数据清洗：统一简化格式

The real format used for the pre-training is more complex, involving the concatenation of multiple code files based on their dependencies. The main idea is to utilize the interleaving data, which is pivotal for teaching the model about programming. This point is also mentioned in recent studies (Guo et al., 2024).

需要注意真正的数据包含多个代码文件的结合，通过使模型理解他们之间的依赖关系来理解分散的数据，从而增强代码建模能力。

Data Deduplication Deduplicating code data is similar to processing natural language except for tokenization, which impacts hyperparameter selection. For instance, Python examples use two spaces, four spaces, or a tab character to signify indentation. A conventional whitespace tokenizer, or one tailored for natural language, might mistakenly assess these samples as different data, which is inaccurate. Our insight is that an effective tokenizer is essential for applying a universal deduplication strategy. Although recent studies have explored fine-grained deduplication at the paragraph or line level, our approach remains at the file level to preserve context integrity.

去重：与文本数据类似，涉及部分编程语言的特定格式转换。

Quality Filtering Quality is a pivotal yet nebulous aspect of pre-training in LLM research, primarily due to the difficulty in quantifying its impact on model performance regarding the scale. We adopted a hybrid, multi-stage filtering process including rule- and model-based scorers. Rule-based scorers are heuristic and varied, though we discovered that code style is not a reliable quality metric and can misclassify too many codes as low-quality. For the model-based scoring, we evaluated several backbone models, training them with roughly 50,000 labeled samples. However, we observed that the correlation between scorer assessments and human judgments varied across languages, and enlarging the training set did not substantially enhance scorer accuracy. Consequently, we only employed model-based scoring for languages where the model predictions align well with human evaluations on a human-annotated validated set.

质量评估：评分器与人类结果的相关性随着编程语言不同有较大差异，且不随数据集增大有所改善。

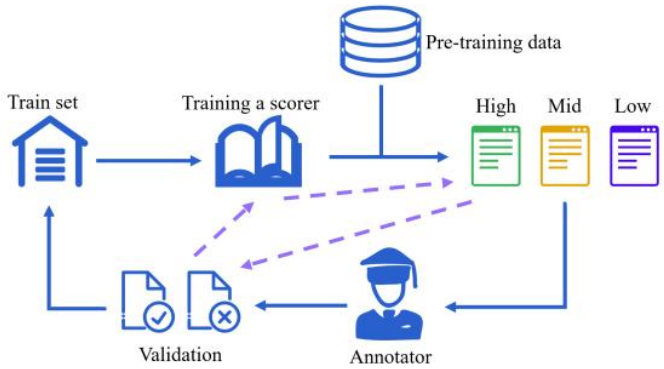


Figure 5: The pipeline of iterative refinement annotation process for a code quality classifier.

code quality data is vague. Identifying code that would be helpful for teaching an LLM is also non-trivial for human experts, for instance, a widely recognized code repository might be overly complex for a beginner. The proposed iterative workflow allows annotators to verify model predictions and refine the guidelines accordingly. To improve the annotation efficiency, we only ask the annotator to check the samples labeled by the scorer as high-quality and low-quality with high confidence. Besides, there is an automatic validation process in each iteration to ensure the previously annotated samples are correctly classified by the scorer, which is shown as yellow dot lines in the figure. In practice, we took three iterations to finalize our scoring model.

通过循环标注判断代码质量为低/高

Dependency sorting The training context window of InternLM2 has been expanded to 32,000 tokens, allowing the utilization of the entire context of a code repository. The structure of the repository may have been broken in previous data processing steps, such as the filtering of code files by their extensions and deduplication. So we first regroup code files originating from the same repository and perform dependency sorting to establish a sequence for concatenating these files. Therefore, a code repository will be viewed as a single long markdown file with code blocks, which allows the model to learn dependencies across files.

We employ regular expressions to detect the "import" relations across various programming languages, and we use topological sorting to determine the concatenation order of files. In practice, the arrangement of files may break folder boundaries, resulting in an interleaving sequence of files from multiple sub-folders. Non-code files, such as markdowns and other documentation, are positioned preceding the first code file located in the same sub-folder.

For the corner cases like multiple paths between an ascendant and descendant and loops within the "import" relation graph, we take the shorter path for the former and use the alphabet order to decide the start point for the latter. A trick in finding "import" relations is to resolve the batched import, such as "__init__.py" or "#include xx.h". Those files may import a bunch of unused dependencies, so we apply heuristic rules to refine our detection of "import" relationships, ensuring that we accurately identify and process these relations at a finer level.

依赖判断：通过检测 import 引入判断依赖关系

3.1.3. Long Text Data

3.1.3 Long Context Data

The ability to handle very long contexts ($> 32K$ tokens) is an increasingly popular topic in the study of LLMs, broadening and facilitating applications, such as book summarization, supporting long-term conversations, and enabling the handling of tasks involving complex reasoning steps. Pre-training data is one crucial factor for expanding the model's context window. We follow the process of preparing long text pre-training data mentioned in Lv et al. (2024), which includes additional experiments and discussions. The following text only outlines the data preparation employed for InternLM2.

Data Filtering Pipeline Our data processing pipeline is designed to filter out low-quality long text data. It comprises three stages: a) Length selection, a rule-based filter that selects data samples exceeding 32K bytes; b) Statistical filters, which leverage statistical features to identify and remove anomalous data; c) Perplexity filters, which utilize the difference in perplexity to assess the coherence between text segments, filtering out samples with distracting context. Note that all data chosen for the long context training is a subset of the standard pre-training corpus, meaning that the long context data will be learned at least twice during the pre-training.

长文本处理：三个阶段：

1. 长度选择，选择大于 32Kbytes 的数据
2. 统计学过滤：从统计学特征分辨不规则数据
3. 复杂数据过滤：通过计算 perplexity 分析文本关系

所有的长文本数据样本都来源于基础的语料库，已经被预处理过一遍。

Statistical Filters We employ a variety of lexical and linguistic features to construct our statistical filters. Data samples failing to comply with established rules are excluded from the pre-training corpus. The full list of these filters can be found in Lv et al. (2024). A remarkable filter is the presence of conjunctions and other words that imply discourse structure, such as "Especially", "Formally", etc. The overall guidance of designing such filters is to filter out the meaningless data rather than selecting the most high-quality data. Statistical filters are particularly effective with long text data, as the statistical features are far more consistent than those in short text data. For instance, a 20-token text may not yield reliable statistics, but a 32K-token text will have a much clearer statistical feature distribution.

使用词义和语义学特征进行统计分析。关联词往往指代上下文结构。目的是去除无意义的数据而不是选取高质量数据。在长文本中统计学特征更有效。

Perplexity定义

PPL是用在自然语言处理领域（NLP）中，衡量语言模型好坏的指标。它主要是根据每个词来估计一句话出现的概率，并用句子长度作normalize，公式为

$$
\begin{aligned}
PP(S) &= P(w_1 w_2 \dots w_N)^{-\frac{1}{N}} \\
&= \sqrt[N]{\frac{1}{p(w_1 w_2 \dots w_N)}} \\
&= \sqrt[N]{\prod_{i=1}^{N} \frac{1}{p(w_i | w_1 w_2 \dots w_{i-1})}}
\end{aligned}
$$

Perplexity的影响因素

这些是听报告了解的：

1. 训练数据集越大，PPL会下降得更低，1billion dataset和10万dataset训练效果是很不一样的；

2. 数据中的标点会对模型的PPL产生很大影响，一个句号能让PPL波动几十，标点的预测总是不稳定；

3. 预测语句中的"的，了"等词也对PPL有很大影响，可能"我借你的书"比"我借你书"的指标值小几十，但从语义上分析有没有这些停用词并不能完全代表句子生成的好坏。

所以，语言模型评估时我们可以用perplexity大致估计训练效果，作出判断和分析，但它不是完全意义上的标准，具体问题还是要具体分析。

Perplexity filters Perplexity is often treated as an estimator for the probability of a text sequence, $P(X)$, and we slightly change its usage to estimate the conditional probability between two text segments $P(S_2|S_1)$, where S_1 is preceding of S_2. When the S_1 and S_2 are strongly correlated, the conditional probability should be higher than estimating the probability of S_2 alone, which also means a negative perplexity difference. Conversely, if the probability changed in the reverse direction, meaning that the S_1 is a distracting context, it should be removed from the pre-training corpus. Ideally, adding more context should not negatively impact the predictability of subsequent text. However, we've observed exceptions in cases of improperly joined texts, such as failed HTML parsing, random social media snippets, and other instances stemming from recognition errors in sources with complex layouts. Note that we only filter the data based on the perplexity difference rather than the perplexity itself, and this could largely mitigate the bias introduced by the estimator itself (using which model to compute the perplexity). The bias of the perplexity estimator has been discussed in Wettig et al. (2024); Sachdeva et al. (2024).

使用 Perplexity 的差值估计前后文间是否存在关联。关联性强的前后文应在条件概率上相比原始概率有较大增长，意味着 Perplexity 负差值。

Threshold Selection Selecting appropriate thresholds is a challenging yet essential part of the data filtering process, and this challenge becomes more severe because we have built many filters. We have two lessons on setting thresholds:

Tailoring thresholds to each domain rather than seeking a universal solution. For example, the statistical filter for conjunction words would not apply to long code data, which typically does not have any conjunction. Similarly, textbooks, research papers, novels, and patents each exhibit unique characteristics. A universal threshold would likely misclassify a significant amount of data. The same logic applies to setting thresholds across different languages; thus, we adjust thresholds for each domain individually.

阈值选取：

1.针对不同的语料类型分别调整阈值

Employing a validation set to streamline the process, focusing only on borderline cases. Unlike learning-based feature extractors or scorers, our statistical and perplexity filters yield smooth results within the same domain. It allows us to focus on samples lying near the threshold, simplifying the adjustment of thresholds since we only need to determine whether to lower or raise them. Lv et al. (2024) illustrates the data clusters alongside scores from specific filters, demonstrating the interpretability of our proposed filters.

2.只关注边界样例。统计学和 perplexity 在界内的表现平滑，因此只需关注边界值的情况来决定阈值调整。

3.2.1. Tokenization

3.2.1 Tokenization

We have chosen to utilize the tokenization approach of GPT-4 due to its exceptional efficiency in compressing a wide array of textual content. Our primary reference is the cl100k vocabulary [4], which mainly encompasses English and programming language tokens, totaling 100,256 entries, with a minor inclusion of fewer than 3,000 Chinese tokens. To optimize compression rates for InternLM when processing Chinese text, while maintaining the overall vocabulary size under 100,000, we carefully selected the top 60,004 tokens from the cl100k vocabulary and integrated them with 32,397 Chinese tokens. Additionally, we included 147 spare tokens to round out the selection, culminating in a vocabulary size that aligns with a multiple of 256, thereby facilitating efficient training.

采用 GPT4 的方式进行分词。

3.2.2. Pre-training Hyper-parameters

3.2.2 Pre-training Hyper-parameters

The basic hyper-parameters are listed in Table 3. During training, AdamW (Loshchilov & Hutter, 2019) with $\beta_1 = 0.9, \beta_2 = 0.95, \epsilon = 1e-8$ and $weight\_decay = 0.1$ is used to optimize the model. We use the cosine learning rate decay and the learning rate decays to 10% of its maximum.

超参数：经典 Adam 优化器，学习率。

3.3 Pre-training Phases

The total number of tokens used for pre-training the 1.8B, 7B, and 20B models ranges from 2.0T to 2.6T, and the pre-training process consists of three distinct phases. In the first phase, we used pre-training corpora with lengths not exceeding 4k. In the second phase, we included 50% of pre-training corpora with lengths not exceeding 32k. In the third phase, we utilized capability-specific enhancement data. During each phase, we mixed data in English, Chinese, and code.

根据文本长度不同在三次预训练过程中分别训练语料库

3.3.1 4k Context Training

For approximately 90% of the training steps, we trained using data with lengths of up to 4096 tokens. If the length of the data exceeded 4096, we would forcibly truncate it and use the remaining part for training as well.

大多数情况下使用短语料进行训练（先从较短语料提取特征，然后推广到长语料情形）

3.3.2 Long Context Training

The utilization of extended context windows significantly enhances the performance of Large Language Models (LLMs) in a variety of applications, such as retrieval-augmented generation (Gao et al., 2023) and intelligent agents (Xi et al., 2023). Motivated by cutting-edge advancements in training for long context (Rozière et al., 2023; Xiong et al., 2023; Liu et al., 2023b), our training process for InternLM2 begins with a 4K context corpus and subsequently transitions to a corpus with 32K context. Instead of only using 32k corpus, 50% of the data is still shorter than 4096 tokens. This long context training phase accounts for about 9% of the total steps. When accommodated to these longer sequences, we adjusted the Rotary Positional Embedding (RoPE) base from 50,000 to 1,000,000, ensuring more effective positional encoding for long contexts (Liu et al., 2023b). Owing to the good scalability of InternEvo (Chen et al., 2024a) and flash attention (Dao, 2023), the training speed only decrease 40% when changing context window from 4K to 32K.

长语料训练情形约占总共的 9%。调整位置信息编码方式更好地获取位置信息从而加强长语料建模能力。

3.3.3 Capability Specific Enhancement Training

Capabilities such as reasoning, mathematical problem-solving, and knowledge memorizing are key abilities expected from large language models. However, in the pre-training process, high-quality capability-related data is sparsely distributed in the entire corpus, which makes it hard for models to be proficient at these mentioned capabilities. Previous works, such as Qwen (Bai et al., 2023a), GLM-130B (Zeng et al., 2023), Nemotron-4 (Parmar et al., 2024), have tried to incorporate instruction-based or high quality data during the pre-training stage to enhance these abilities. In InternLM2, we collect an enriched dataset with a meticulously curated mix of high-quality retrieved data and various types of open-source data from the huggingface datasets platform [5]. In total, we collect 24 Billion tokens in this dataset, and details of this corpus are shown in Table 4. We filter out test set related data and run a contamination test as illustrated in Section 5.4. To make the model fit these data well, we employed a smaller learning rate and batch size.

After this enhancement training phase, the InternLM2 model exhibited substantial performance improvements in coding, reasoning, question answering, and examinations, with detailed evaluations results shown in the following evaluation section. To aid the research community, we have released checkpoints before and after the enhancement training, naming them InternLM2-{size}-Base and InternLM2-{size}, respectively.

特定能力增强训练：针对特定能力（原因推理归纳，数学问题等），专门建立高质量语料库

进行训练，同时使用较小的学习率和块尺寸更好拟合。

4. Alignment

4 Alignment

The pre-training stage empowers LLMs with the foundation abilities and knowledge that are necessary for solving various tasks. We further fine-tune the LLMs to fully elicit their capabilities and guide LLMs to serve as helpful and harmless AI assistants. This stage, also

Alignment 的原因：更好地满足人类需求，与人类价值观相符，遵循人类 instruction 生成结果。

commonly referred to as 'Alignment', typically contains two phases: supervised fine-tuning (SFT) and reinforcement learning from human feedback (RLHF). During SFT, we fine-tune the model to follow diverse human instructions by high-quality instruction data (Sec.4.1). Then we propose **CO**nditional**On**Line RLHF, which applies a novel conditional reward model that can reconcile different kinds of human preferences (*e.g.*, multi-step reasoning accuracy, helpfulness, harmlessness), and conducts three-round online RLHF to reduce reward hacking (Sec. 4.2. In the alignment stage, we keep the long-context capability of LLMs by utilizing long-context pre-training data during SFT and RLHF 4.3. We also introduce our practices of improving the tool utilization capability of LLMs 4.4.

包含两个方面：

1. SFT，监督调参，用于遵循人类 instructions 生成结果
2. COOLRLHF，条件-在线-基于人类反馈的强化学习，用于满足不同人类需求（每一步的置信度，无害程度等）

4.1. Supervised Fine-Tuning

In the supervised fine-tuning (SFT) stage, we use a dataset of 10 million instruction data instances, which have been screened to ensure their helpfulness and harmlessness. The dataset encompasses a diverse range of topics, including general conversation, NLP tasks, mathematical problems, code generation and function calls, etc. Figure 7 shows the detailed distribution of SFT data topics. To facilitate a versatile representation of such various tasks, we transform the data samples into the ChatML (Cha) format. Both the 7B and 20B models undergo training for one epoch using the AdamW optimizer with an initial learning rate of 4e-5.

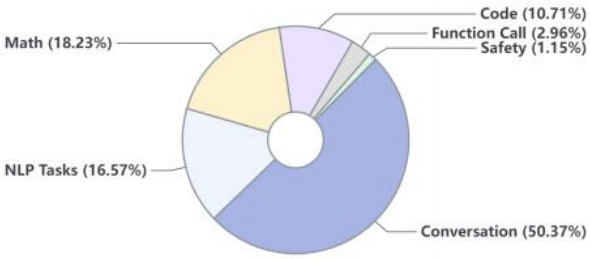


Figure 7: The distribution of SFT data instances.

使用不同 instructions 进行训练，主要为对话

4.2 COOL Reinforcement Learning from Human Feedback

4.2 COOL Reinforcement Learning from Human Feedback

Reinforcement Learning from Human Feedback (RLHF) (Christiano et al., 2017; Ouyang et al., 2022) is an innovative approach within the realm of large language models. By incorporating human feedback, RLHF creates reward models that serve as proxies for human preferences, thereby providing reward signals for LLMs to learn through the use of Proximal Policy Optimization (PPO) (Schulman et al., 2017). This methodology enables models to better understand and execute tasks that are difficult to define through traditional methods.

Despite the achievements of RLHF, there are still some issues in its practical application. The first is the preference conflicts. For example, in developing a dialogue system, we expect it to provide useful information (helpful) while not producing harmful or inappropriate content (harmless). However, these two preferences often cannot be satisfied simultaneously in practice, as providing useful information might involve sensitive or high-risk content in some cases. Existing RLHF methods (Touvron et al., 2023b; Dai et al., 2023; Wu et al., 2023) usually rely on multiple preference models for scoring, which also introduces more models in the training pipeline thus increases computational cost and slows down training speed. Second, RLHF faces the issue of reward hacking, especially when the policy becomes more powerful with the scale increasing (Manheim & Garrabrant, 2018; Gao et al., 2022), where the model might learn to "cheat" the reward system by shortcuts to obtain high scores,

通过近端优化策略的强化学习过程提高性能。同时存在一些问题：偏好冲突，指某些需求很难同时实现（例如提供准确信息的同时不泄露隐私等）。另一个问题是奖励函数“骇入”，

即模型通过欺骗奖励函数获得高评分，而不是真正进行学习。

rather than truly learning the expected behavior. This leads the model to maximize rewards in unintended ways, significantly affecting the effectiveness and reliability of LLMs.

To address these issues, we propose **Co**nditional **On**Line RLHF (COOL RLHF). COOL RLHF first introduces a Conditional Reward mechanism to reconcile diverse preferences, which allows the reward model to dynamically allocate its attention to various preferences based on specific conditions, thereby optimally integrating multiple preferences. Furthermore, COOL RLHF adopts a multi-round Online RLHF strategy to enable the LLM to promptly adapt to new human feedback, mitigating the occurrence of reward hacking.

针对这两点问题提出了条件-在线-强化学习机制，即通过“条件”奖励函数在不同的场景下动态调整侧重，从而满足多个奖励函数的需要。通过“在线”强化学习策略及时更新人类反馈，以防止奖励函数骇入的问题。

4.2.1. Conditional Reward Model

The Conditional Reward Model represents an innovative solution to the challenges inherent in the previous preference modeling of RLHF methods. Unlike conventional approaches that typically rely on multiple preference models to address preference conflicts across different domains (Figure 8(a)), the Conditional Reward Model incorporates different system prompts for different kinds of preferences to effectively model a variety of preferences within a singular reward model.

不同于传统方式，条件奖励模型将不同系统的偏好结合到一个模型中。

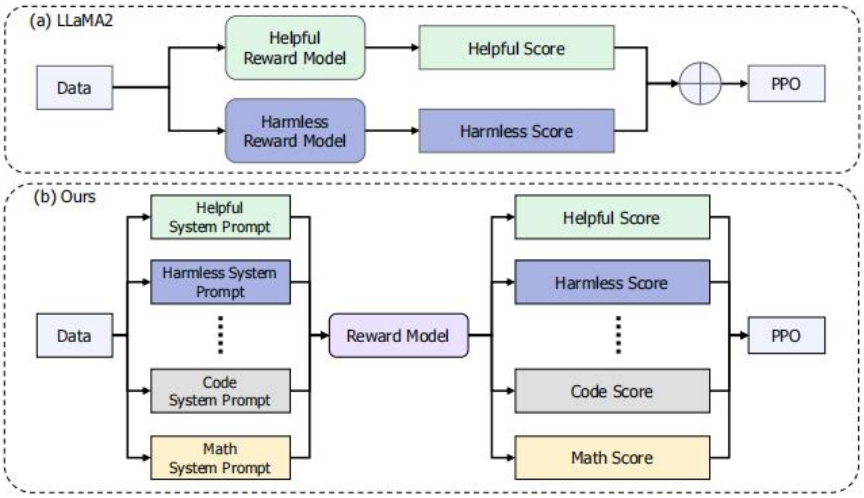


Figure 8: Architecture of the Conditional Reward Model. (a) LLaMA2 employs distinct reward models to address the issue of preference conflicts. (b) The proposed conditional reward model (ours) utilizes conditional system prompts to reconcile preference data across various domains, enabling the modeling of multiple preferences using a single reward model.

用同一个奖励模型，当时使用不同的 prompt 进行评估（应该根据需要进行加权），最终算出评估分数。

Specifically, as depicted in Figure 8(b), the Conditional Reward Model employs different system prompts to seamlessly blend data from various fields. Since the reward model is initialized from a SFT model, which already learned to follow diverse human instructions, we also let the reward model follow different system prompts to adapt to diverse preferences of different scenarios. In the Conditional Reward Model, system prompts are not simply a component of its input; they are also a crucial tool for directing the reward score in alignment with specific preferences in varied scenarios. Such an integration facilitates the management of contradictory and complex human preferences within a unified reward model without sacrificing accuracy.

借由不同的 prompts 对不同场景下的偏好进行校准。

Data Composition The training process of the Conditional Reward Model involves an extensive dataset, encompassing various fields such as dialogue, article writing, poetry, summarization, coding, mathematics, and formatted output, with up to 2.4 million binarized preference pairs. This comprehensive dataset ensures the model's broad adaptability and enhances its capability to undertake reinforcement learning in a broader and more complex array of situations. Thus, by employing the conditional system prompt method, the reward

强化学习的数据分布

Loss Function Furthermore, to reduce the influence of the imbalance between easy and difficult samples in the dataset, we modify the original ranking loss function (Burges et al., 2005) inspired by Focal Loss (Lin et al., 2017). We add a difficulty decay coefficient to the ranking loss, making the loss value larger for difficult samples and smaller for easy ones, preventing overfitting on a large number of easy samples. The focal ranking loss is formulated as

$$L_{ranking} = -(1 - 2 \times \max(0, P_{i,j} - \frac{1}{2}))^{\gamma} \log(P_{i,j})), \tag{1}$$

损失函数：基于排序损失函数，增加难度衰减系数，问题越难对应的损失函数值越大

where $P_{i,j} = \sigma(r_i - r_j)$ represents the probability that $reward_i$ is greater than $reward_j$. The difficulty decay coefficient only takes effect when the model correctly predicts the preference of the training sample, i.e., $P_{i,j} > 0.5$, otherwise it equals to 1. The term γ represents a hyper-parameter that is instrumental in modulating the difficulty decay ratio. Here we set it to 2 by default. Concurrently, to ensure the stability and consistency of the output scores from the reward model across different training, we introduce a logarithmic barrier penalty to the reward scores to confine the score distribution within a range of -5 to 5, defined as

$$L_{penalty} = -(\log(x + 5) + \log(5 - x)). \tag{2}$$

This constraint is critical as it obviates the need to modify additional reward-related hyper-parameters in the PPO stage, which could potentially arise due to variations of the reward score distribution in different reward models. Overall, the loss function of the reward model is

$$L = L_{ranking} + \lambda L_{penalty}. \tag{3}$$

还有一个惩罚函数，最终的损失函数是两者加权和

Training Details In our experiment, we align the sizes of the reward models with those of the actor models used in PPO. Following the methods described in InstructGPT(Ouyang et al., 2022), we initialize the reward models using the SFT model weights, modifying the output layer to a one-dimensional linear mapping layer, which was randomly initialized. Our batch construction strategy focuses on fixing the total length of preference data at 16384 tokens per batch, rather than limiting the number of preference pairs, to avoid training inefficiencies due to data padding. The maximum context length is set at 8192. A special token is appended to each sequence's end, with its output value used as the reward score. We adapt AdamW as the optimizer. The learning rate follows a cosine annealing schedule, decreasing from 1e-5 to 5e-6 and weight decay is set to 0.01. To prevent overfitting, the models are trained for one epoch.

通过只训练一个周期防止过拟合？那要是没有收敛怎么办？

4.2.2. Online RLHF

4.2.2 Online RLHF

After obtaining a conditional reward model, we conduct Proximal Policy Optimization (PPO) to align the LLMs to the human preferences modeled by the reward model Ouyang et al. (2022). To address the challenge of reward hacking in the PPO stage, we introduce an Online RLHF approach, divided into two distinct pathways: a Fast Path for immediate, targeted improvements and a Slow Path for long-term, comprehensive refinement of the reward model. The Fast and Slow Paths are complementary to provide an adaptive framework for mitigating reward hacking and enhancing the performance and reliability of LLMs trained with human feedback.

一个快速的、有针对性的改进的快速路径和一个对奖励模型的长期的、全面的改进的缓慢路径。快速路径和慢路径是互补的，提供了一个自适应的框架，以减少奖励黑客攻击和提高由人工反馈训练的 llm 的性能和可靠性。

Fast Path The fast path in Online RLHF focuses on quickly identifying and rectifying reward hacking incidents through targeted patches to improve the reliability of the reward model. As the PPO training goes by, the LLMs are encouraged to gravitate towards high-reward regions, which usually expose more reward hacking scenarios that can be easily detected. After identifying the hacking pattern after each round of RLHF, we construct preference pairs that highlight these patterns by comparing responses generated by early and late-stage PPO models in the current round. Incorporating 20 to 100 such preference pairs into the training process is sufficient to prevent the reward model from the corresponding hacking pattern significantly. This process allows for a swift fix of the reward model to the emerging hacking behaviors, enhancing the reward model's reliability and adherence to desired outcomes.

快速路径：出现奖励骇入的模型会对某些提供较高奖励的区域偏移，因此可以采用偏好对的方法比较模型在学习早期和后期对这一区域的偏好，以确定是否存在某种程度的奖励骇入现象。

Slow Path In comparison to the fast path that focuses on fixing reward hacking, the slow path aims at a general improvement of the reward model's upper bound, especially the reliability and robustness of the reward model at the high-reward regions, by covering the LLMs responses from the most recent and capable models, following previous work (Bai et al., 2022). To achieve this, responses generated by models at various stages of training (including the SFT model, early-stage PPO model, and late-stage PPO model) are used to form pairwise comparisons. These pairs are then presented to professional human annotators to label their preferences.

Such a process provides a more nuanced and thorough refinement of the reward model but requires extensive human annotation time. To improve the efficiency of online RLHF, we only use the accumulated human preferences of all previous models at the launch time of our experiments (*i.e.*, which may not include the preferences of responses produced by models at the current round due to the time cost of human labeling). By continuously updating the model based on human feedback, the Slow Path ensures that the reward model evolves in tandem with the complexities and subtleties of human preferences.

慢速路径：更偏好于提高奖励模型的上限。对不同阶段的模型生成的响应进行比较，由人类进行标注，来确定偏移程度和及时更正。他耗费更多的时间但可以达到更好的效果，在模型复杂性和人类偏好上取得不错的折中效果。

Implementation In our implementation of Online RLHF, we conducted three rounds of refinement. In these cycles, we gathered thousands of preference patches and online preference data in the Fast Path to update the reward model, and use all the existing human preference data of responses from previous models. Each round of Online RLHF provided valuable insights, allowing us to dynamically adjust and refine the reward model, thereby enhancing the overall performance and reliability of language models trained with human feedback.

在三个轮次中优化强化学习模型。采用快速路径的偏好对和慢速路径的人类偏好数据进行调整。

4.2.3. PPO Training Details

During the RL alignment phase, we adopted the standard PPO (Proximal Policy Optimization) algorithm and made several adaptions to it, ensuring a more stable training process. The framework involves four models: the actor model, critic model, reference model, and reward model. During training, the latter 2 models are frozen, and only the former 2 models are actively trained. Notably, all these models are of the same size, ensuring consistency in their capacity to process and generate data. We iterate through about 200k diverse queries in about 400 iterations and choose the best checkpoints on the validation sets for release.

4 models，前两个在训练过程中被训练，所有模型的大小相同。

Model Initialization As is common practice, we initialize the reference model and the actor model from the SFT model weights. The critic model is initialized from the reward model (excluding the linear head) and undergoes a 50-iteration pre-training phase, during which the actor model is frozen. This phase is critical for stabilizing the value estimation in early training, thus preventing the potentially detrimental effects of unstable values. We conducted ablation studies comparing the initialization of the critic model from the reward model versus the SFT model, as show in Figure 9. Our results show that the critic model initialized from the reward model experiences larger losses during the first few iterations of PPO training, but after approximately 20 iterations, it consistently exhibits lower losses and leads to higher rewards for the actor model. We hypothesize that the higher loss observed during the initial stage may reveal fundamental differences between the tasks of reward modeling and critic modeling. The subsequent lower loss could be attributed to a

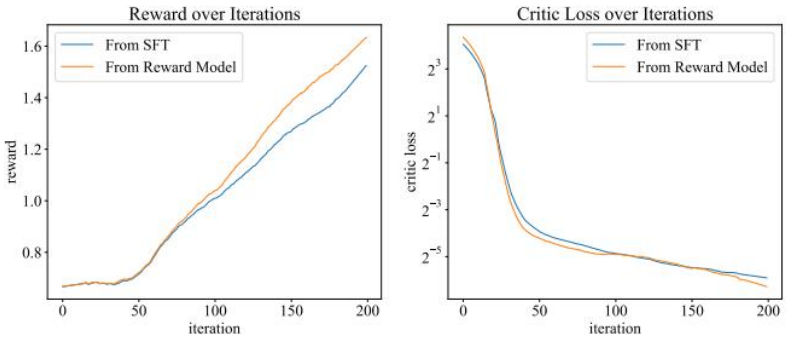


Figure 9: Ablation study on critic model initialization. Note that we used logarithmic coordinates in critic loss due to its large dynamic range.

使用 critic loss 时初始的损失函数值都很高，可能与奖励建模和批评建模的区别有关。

Hyperparameters We set the KL divergence coefficient to 0.01. The learning rates for the actor model and the critic model are set to 1e-6 and 5e-6, respectively. We found that a larger λ value for PPO leads to higher rewards in our case, so we set it to 0.99. We adopted a slightly conservative sampling strategy, with $top\_p = 0.9$, to strike a balance between sampling diversity and convergence speed. Unlike some conventional approaches, we do not apply value loss clipping or advantage normalization. Despite extensive RL tricks, the training remains remarkably stable, partially due to our meticulous Online RLHF efforts.

在概率论或信息论中，KL散度(Kullback–Leibler divergence)，又称相对熵（relative entropy），是描述两个**概率分布P和Q差异的一种方法**。它是非对称的，这意味着D(P||Q) ≠ D(Q||P)。特别的，在信息论中，D(P||Q)表示当用概率分布Q来拟合真实分布P时，产生的信息损耗，其中P表示真实分布，Q表示P的拟合分布。
有人将KL散度称为KL距离，但事实上，KL散度并不满足距离的概念，应为：1）KL散度不是对称的；2）KL散度不满足三角不等式。

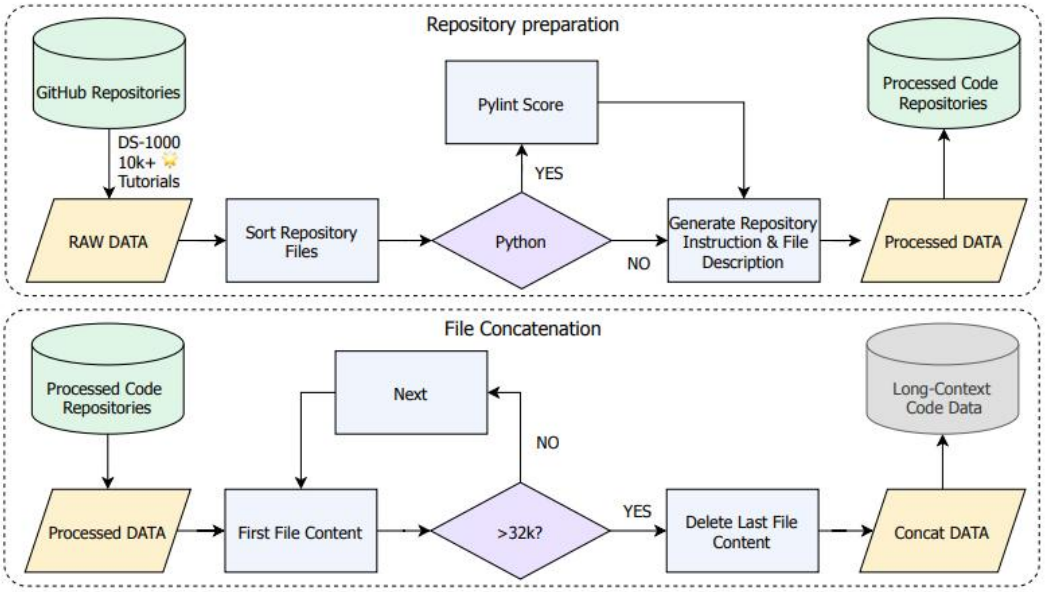
这里的 lambda 是损失函数里那个？那不是相当于加权和都加在了惩罚函数，而没有排序损失算法的权重上？

4.3 Long-Context Finetuning

To preserve the long-context capability of LLMs after fine-tuning, we keep using the long-context pre-training data in SFT and RLHF, inspired by previous work that adopts long-context pre-training corpus in SFT (Xiong et al., 2023). Specifically, we utilize two types of data: one comprises long-context data from books, while the other is long-context data obtained from GitHub repositories and concatenated through specific paradigms, described below.

具体来说，我们使用了两种类型的数据：一种包含来自书籍的长上下文数据，而另一种是从 GitHub 存储库中获得的长上下文数据，并通过特定的范式进行连接，如下所述。

To enhance the data analysis capabilities of InternLM2, we choose the code repositories used in DS-1000 (Lai et al., 2023) as the core repositories, which include Pandas, Numpy, Tensorflow, Scipy, Scikit-learn, PyTorch, and Matplotlib. We then search for repositories on GitHub with over 10,000 stars that referenced these core repositories and conducted the same filtering and data cleaning process as pre-training. For each repository, we initially sort the acquired raw data using a depth-first approach, while simultaneously generating the required prompts that briefly describe the file content, as shown in Figure 11. Subsequently, we concatenate the processed data in sequence until reaching a length of 32k. The experimental results show that long-context code data improves not only the long-context capability of LLMs but also the code capabilities.



4.4 Tool-Augmented LLMs

General Tool Calling We adopt a modified version of the ChatML format to enable general tool calling by introducing the "environment" role. Such a modification shares the same format in chat scenarios but provides more clear signal to the model when adopting agents. Additionally, we define two specific keywords to support the diverse purpose of AI agents, *i.e.*, code interpreter (<|interpreter|>) and external plugins (<|plugin|>). This enables us

采用两个关键词：代码编译器和外部插件。

to adopt a unified streaming format that can handle various types of plugin extensions and AI environments while being compatible with general chat. Figure 17 shows a concrete example of streaming chat format. To fully elicit the agent ability of InternLM2, we align the agent corpus to the chat domain and disentangle it along the basic capabilities of the language model for fine-grained training, as described in Agent-FLAN (Chen et al., 2024c).

Code Interpreter We also enhance the ability of InternLM2-Chat to solve the math problems by code interpreters, by treating the Python code interpreter as a special tool using the same schema described in Tool Learning. We adopt the reasoning interleaved with coding (RICO) strategy and construct the data in an iterative hard example mining manner, described in InternLM-Math (Ying et al., 2024).

InternLM2 Streaming Format

```
<|im_start|>system
You are InternLM2-Chat, a harmless AI assistant<|im_end|>
<|im_start|>system name=<|plugin|>
[
    {
        "name": "get_current_weather",
        "description": "Get the current weather in a given location",
        "parameters": {
            "type": "object",
            "properties": {
                "location": {
                    "type": "string",
                    "description": "The city and state, e.g. San Francisco, CA",
                },
                "unit": {"type": "string"},
            },
            "required": ["location"],
        },
    }
]
<|im_end|>
<|im_start|>user
I want to know today's weather in Shanghai<|im_end|>
<|im_start|>assistant
Sure, I will search for the weather of Shanghai.<|action_start|><|plugin|>
{"name": "get_current_weather", "parameters": {"location": "Shanghai"}}<|action_end|><|im_end|>
<|im_start|>environment name=<|plugin|>
{"temperature": 22}<|im_end|>
<|im_start|>assistant
The weather in Shanghai is 22 celsius<|im_end|>
```

5 Evaluation and Analysis

5.1 Overview

5.1 Overview

In this section, we provide a comprehensive evaluation and analysis of the language model's performance across various domains and tasks. The evaluation is structured into two main categories: *(a) downstream tasks* and *(b) alignment*. For each category, we further break down the evaluation into specific subtasks to provide a detailed understanding of the model's strengths and weaknesses. Lastly, we discuss the potential issue of data contamination in language models and its impact on model performance and reliability. All evaluations are performed using OpenCompass (Contributors, 2023b) unless explicitly specified otherwise[6].

模型评估分为两个种类：下游任务和对齐。存在潜在的数据污染问题。

5.2 Performance on Downstream Tasks

5.2 Performance on Downstream Tasks

We start by detailing evaluation protocols and performance metrics for multiple NLP tasks. We introduce datasets, explain our experimental setup, and then present results with in-depth analysis, comparing to state-of-the-art methods to showcase the effectiveness of our model. The performance assessment will be dissected through six key dimensions: *(1) comprehensive examinations, (2) language and knowledge, (3) reasoning and mathematics, (4) multiple programming language coding, (5) long-context modeling, (6) tool utilization.*

总的来说表现评估围绕六个维度，基本都是不同方向的关键能力。

Needle-in-the-Haystack "Needle-in-the-Haystack" is a single-needle retrieval task, which is designed to test the Large Language Models' (LLMs) ability to recall a single piece of key information. This is done by inserting a crucial piece of information into a Haystack text of a target length[9] at various positions and then querying the model about this key information at the end of the prompt. This method precisely visualizes LLMs' recall capabilities at different positions within long texts of varying lengths. We design a Chinese Haystack

大海捞针测试。将关键信息安插到语料的不同位置中，然后进行搜索和队列

5.3 Performance on Alignment

主要是模型校准能力，满足人类需求的能力

6 Conclusion

In this report, we present the InternLM2 large language model, which demonstrates exceptional performance in both subjective and objective evaluations. InternLM2 has been trained on over 2T of high-quality pre-training corpora, covering model sizes of 1.8B, 7B, and 20B, making it suitable for a variety of scenarios. To better support long contexts, InternLM2 employs GQA to reduce inference costs, and has been additionally trained on up to 32k contexts. Besides open-sourcing the model itself, we also make available the checkpoints from various phases of the training process, facilitating studies by future researches.

In addition to open-sourcing the model, we provide a detailed description of how we trained InternLM2, including the training framework, pre-training text data, pre-training code data, pre-training long text data, and alignment data. Furthermore, to address preference conflicts encountered during the RLHF process, we propose Conditional Online RLHF to harmonize various preferences. This information can offer insights into how to prepare pre-training data and how to train larger models more effectively.