一　书生·浦语大模型全链路开源体系

https://www.bilibili.com/video/BV1Vx421X72D/?vd_source=c321adfb85c49c9f0da0ffc729de2292

第一章精彩呈现了书生·浦语大模型的全链路开源体系，并探讨了通用人工智能的最新发展趋势，尤其是行业从专用模型逐渐过渡到通用大模型的重要转变。深入介绍了书生模型在近几个月中的关键升级，特别是其在8K语境支持、多模态输入及不同规模模型的支持方面的进步，这些提升极大增强了该模型在语言建模、对话交互以及智能体框架的处理能力。此外，还讨论了英特尔M2对于模型性能提升的重要影响，使其更加擅长处理复杂场景。

视频还详尽展示了书生·浦语大模型的不同尺寸和类型，旨在适应多样化的使用需求。通过详细的时间线，能够清楚地把握到每个关键环节的详细讲解，包括模型对长上下文的理解、智能体框架以及模型轻量化和多模态智能体工具箱的部署。同时，还特别强调了循环评测策略和全面的客观评测如何突出模型在语言知识推理、数学代码处理等领域的先进性。

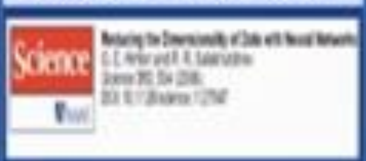
最终，全面总结了书生·浦语大模型的开源体系，涵盖了从数据集的准备、模型的预训练、微调、部署，到评测和实际应用等各个环节，彰显了其在推动语言模型技术的进步及实际应用的全面性和深度。

大模型成为发展通用人工智能的重要途径

专用模型：

针对特定任务，一个模型解决一个问题

深度学习理论突破



Reducing the Dimensionality of Data with Neural Networks
G. E. Hinton and R. R. Salakhutdinov
Science 313, 504 (2006);
DOI: 10.1126/science.1127647

深度置信网络

2006

ImageNet竞赛

1000类，100万数据

2012

围棋比赛

AlphaGo 4:1 李世乭

2016

AlphaFold

蛋白质结构预测准确率新高

2021

2011

大规模语音识别

Switchboard错误降低9%

2014

人脸识别

LFW识别率99%,超过人类

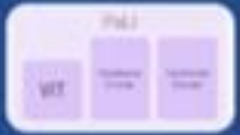
2019

德州扑克

首次在多人复杂对局中超越人类

通用大模型：

一个模型应对多种任务、多种模态



ChatGPT GPT-4

书生·浦语全链条开源开放体系

数据

书生·万卷

2TB数据，
涵盖多种模态与任务

预训练

InternLM-Train

并行训练，极致优化
速度达到 3600 tokens/sec/gpu

微调

XTuner

支持 **全参数微调，**
支持LoRA**等低成本微调**

部署

LMDeploy

全链路部署，性能领先
每秒生成 2000+ tokens

评测

OpenCompass

全方位评测，性能可复现
100 套评测集，50 万道题目

应用

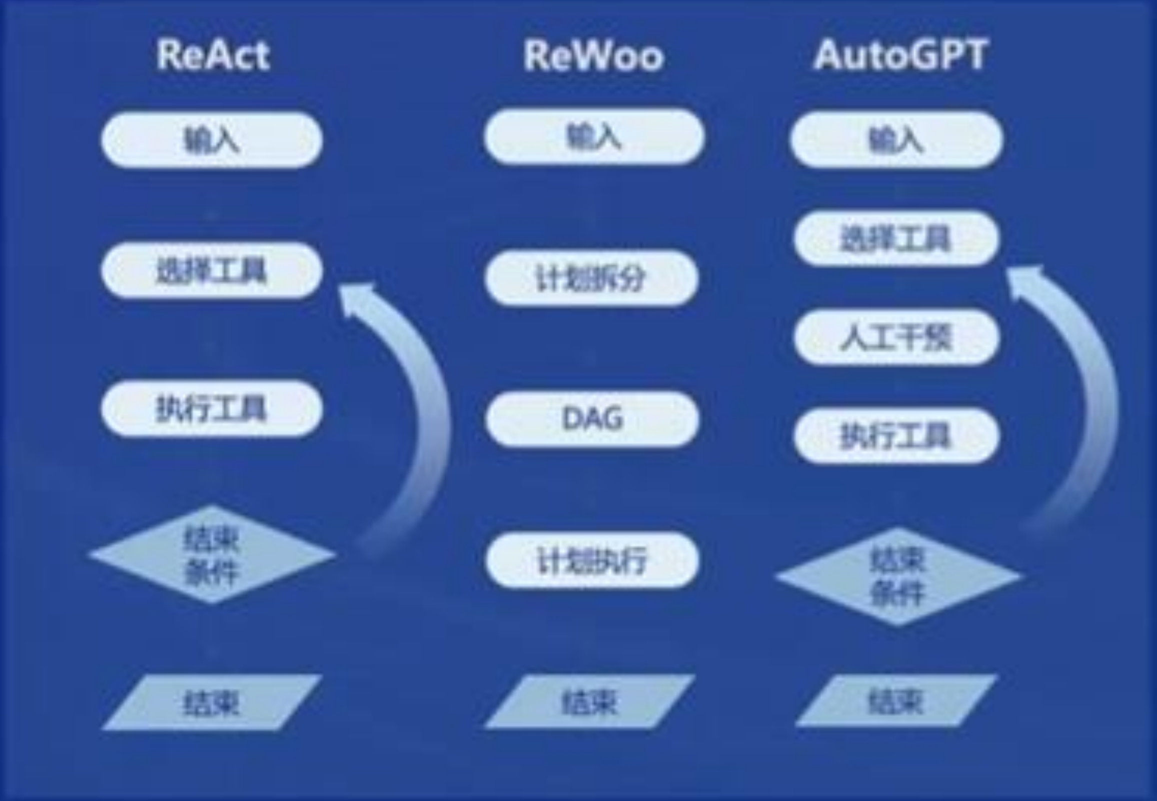
Lagent
AgentLego

支持多种智能体，支持**代**
码解释器等多种工具

全链条开源开放体系｜智能体

轻量级智能体框架 Lagent

支持多种类型的智能体能力



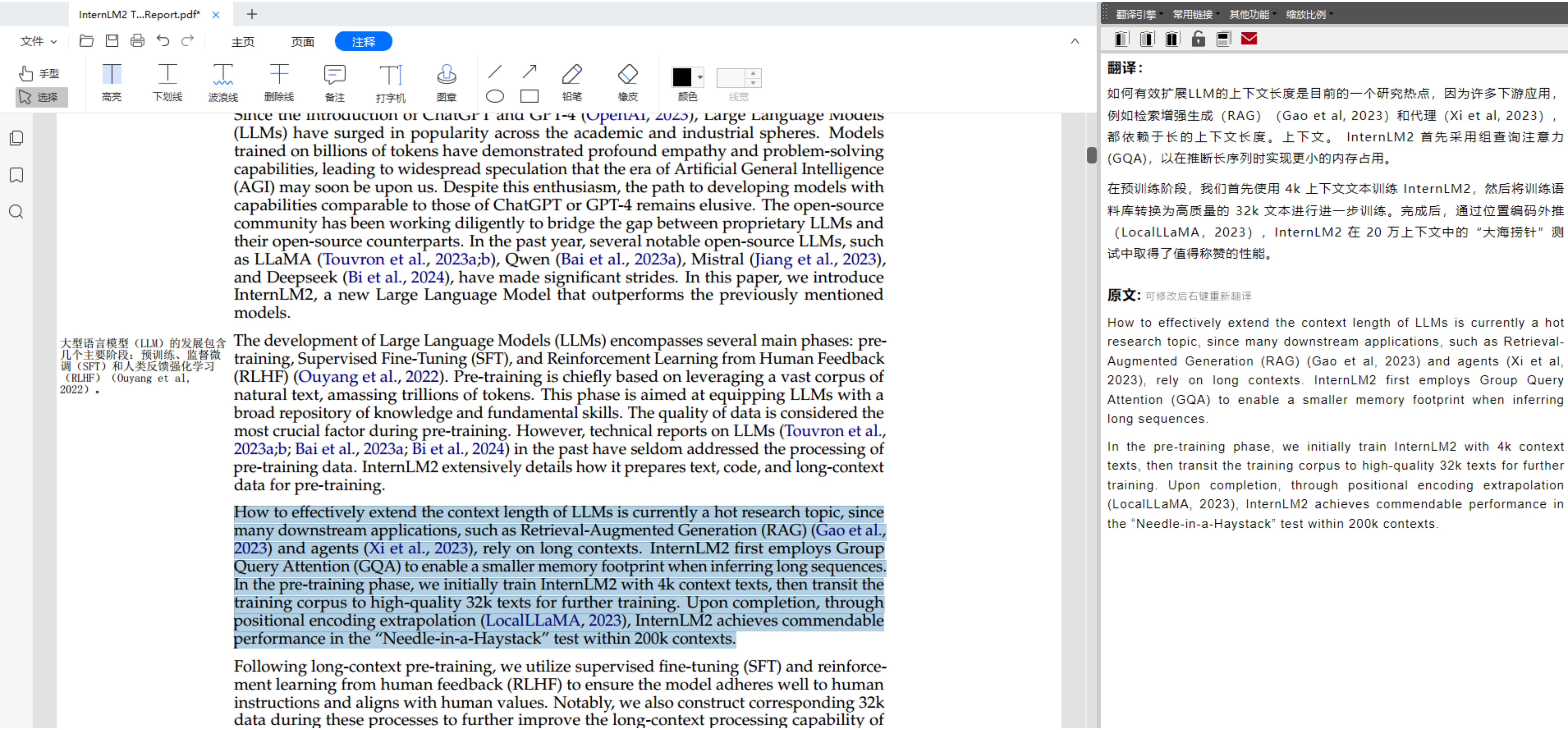
灵活支持多种大语言模型

- GPT-3.5/4
- InternLM
- Hugging Face Transformers
- Llama

简单易拓展，支持丰富的工具

AI 工具	能力拓展	Rapid API
文生图	搜索	出行 API
文生语音	计算器	财经 API
图片描述	代码解释器	体育资讯 API

https://arxiv.org/pdf/2403.17297.pdf



Since the introduction of ChatGPT and GPT-4 (OpenAI, 2023), Large Language Models (LLMs) have surged in popularity across the academic and industrial spheres. Models trained on billions of tokens have demonstrated profound empathy and problem-solving capabilities, leading to widespread speculation that the era of Artificial General Intelligence (AGI) may soon be upon us. Despite this enthusiasm, the path to developing models with capabilities comparable to those of ChatGPT or GPT-4 remains elusive. The open-source community has been working diligently to bridge the gap between proprietary LLMs and their open-source counterparts. In the past year, several notable open-source LLMs, such as LLaMA (Touvron et al., 2023a;b), Qwen (Bai et al., 2023a), Mistral (Jiang et al., 2023), and Deepseek (Bi et al., 2024), have made significant strides. In this paper, we introduce InternLM2, a new Large Language Model that outperforms the previously mentioned models.

大型语言模型（LLM）的发展包含几个主要阶段：预训练、监督微调（SFT）和人类反馈强化学习（RLHF）（Ouyang et al, 2022）。

The development of Large Language Models (LLMs) encompasses several main phases: pre-training, Supervised Fine-Tuning (SFT), and Reinforcement Learning from Human Feedback (RLHF) (Ouyang et al., 2022). Pre-training is chiefly based on leveraging a vast corpus of natural text, amassing trillions of tokens. This phase is aimed at equipping LLMs with a broad repository of knowledge and fundamental skills. The quality of data is considered the most crucial factor during pre-training. However, technical reports on LLMs (Touvron et al., 2023a;b; Bai et al., 2023a; Bi et al., 2024) in the past have seldom addressed the processing of pre-training data. InternLM2 extensively details how it prepares text, code, and long-context data for pre-training.

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.

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

翻译：

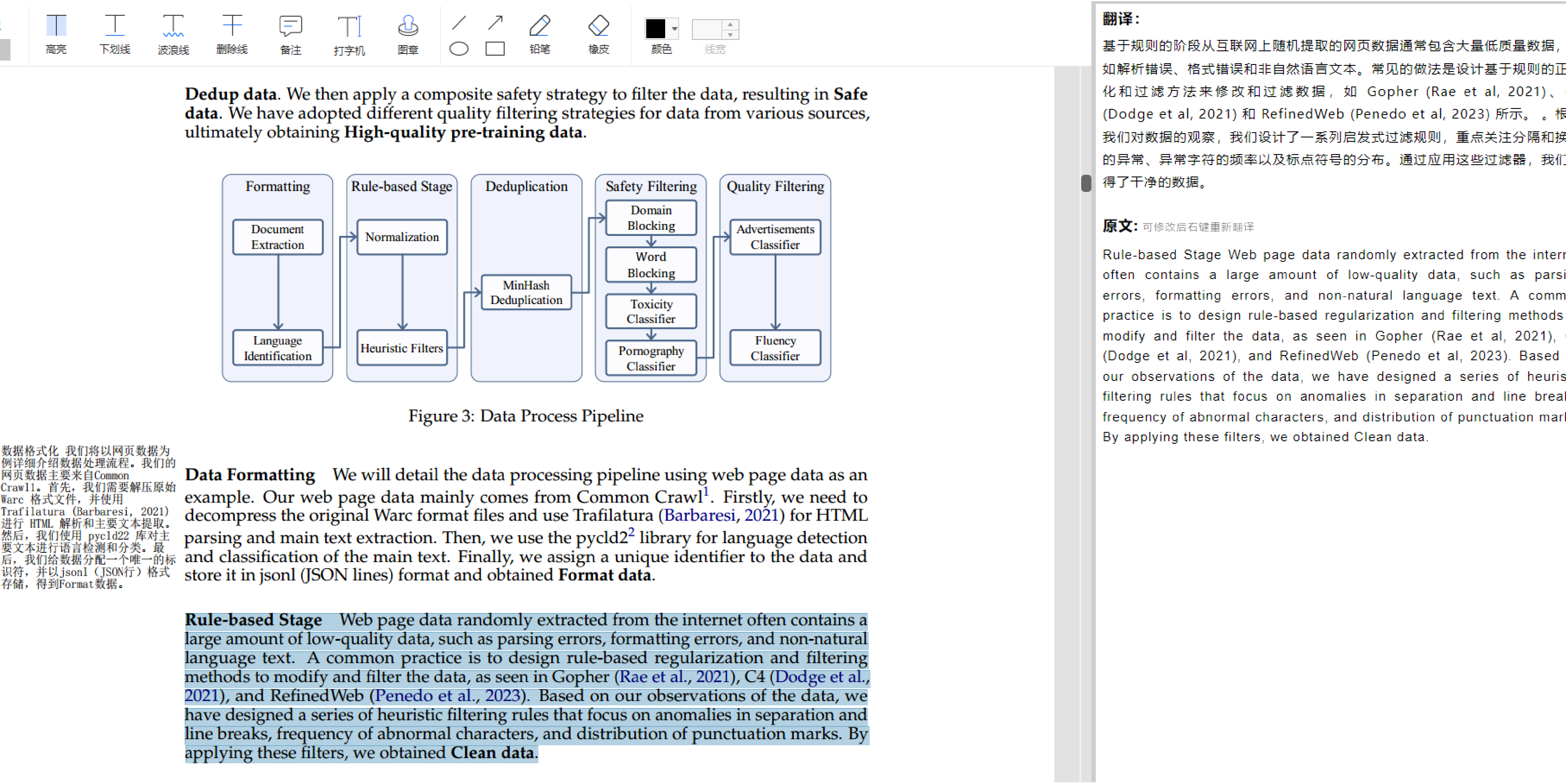
如何有效扩展LLM的上下文长度是目前的一个研究热点，因为许多下游应用，例如检索增强生成（RAG）（Gao et al, 2023）和代理（Xi et al, 2023），都依赖于长的上下文长度。上下文。 InternLM2 首先采用组查询注意力(GQA)，以在推断长序列时实现更小的内存占用。

在预训练阶段，我们首先使用 4k 上下文文本训练 InternLM2，然后将训练语料库转换为高质量的 32k 文本进行进一步训练。完成后，通过位置编码外推（LocalLLaMA，2023），InternLM2 在 20 万上下文中的“大海捞针”测试中取得了值得称赞的性能。

原文： 可修改后右键重新翻译

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.



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**.

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**.

数据格式化 我们将以网页数据为例详细介绍数据处理流程。我们的网页数据主要来自Common Crawl。首先，我们需要解压原始Warc 格式文件，并使用Trafilatura (Barbaresi, 2021) 进行 HTML 解析和主要文本提取。然后，我们使用 pycld22 库对主要文本进行语言检测和分类。最后，我们给数据分配一个唯一的标识符，并以jsonl（JSON行）格式存储，得到Format数据。

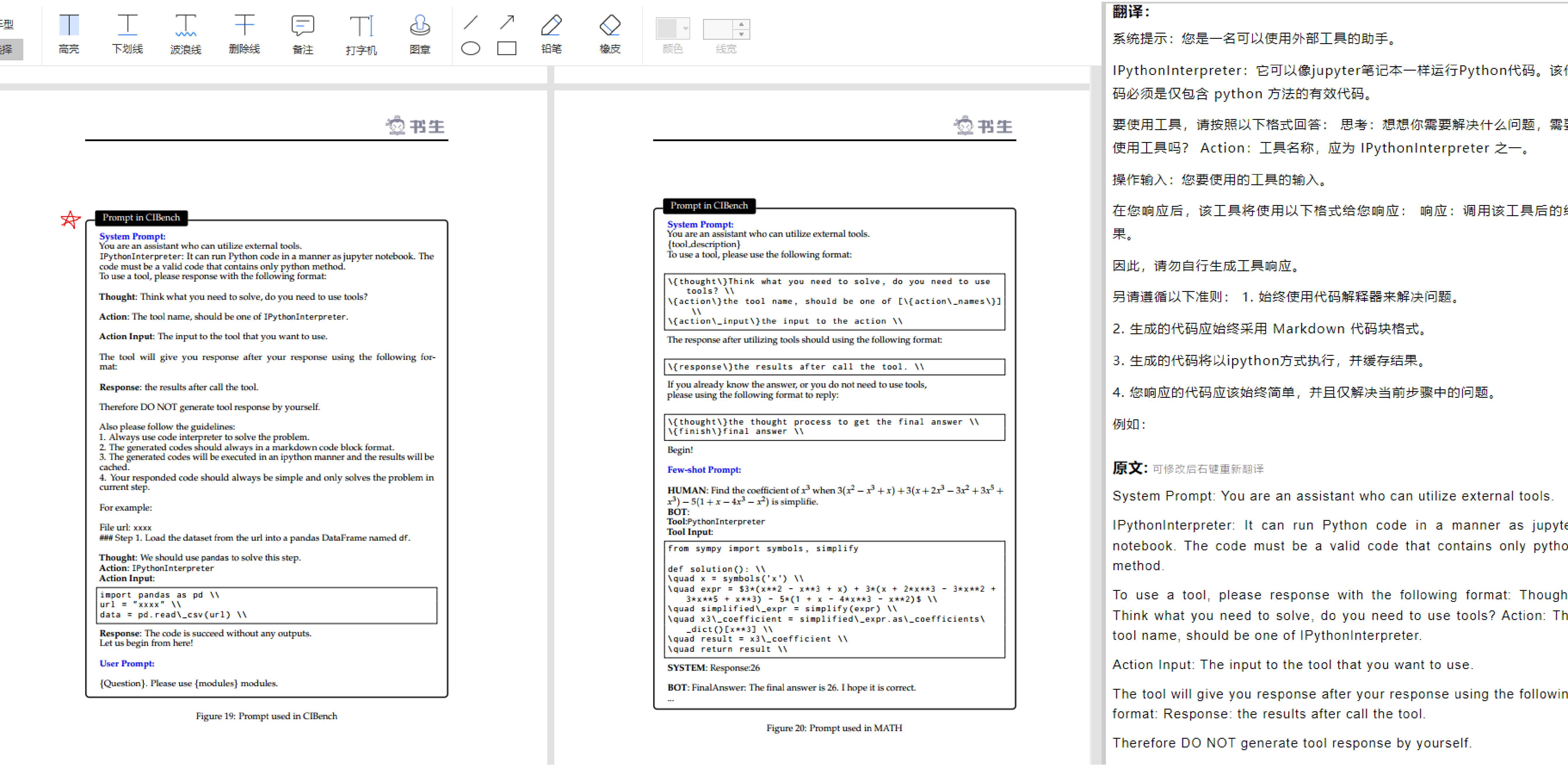
Rule-based Stage Web page data randomly extracted from the internet often contains a large amount of low-quality data, such as parsing errors, formatting errors, and non-natural language text. A common practice is to design rule-based regularization and filtering methods to modify and filter the data, 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**.

翻译：

基于规则的阶段从互联网上随机提取的网页数据通常包含大量低质量数据，
如解析错误、格式错误和非自然语言文本。常见的做法是设计基于规则的正
化和过滤方法来修改和过滤数据，如 Gopher (Rae et al, 2021)、
(Dodge et al, 2021) 和 RefinedWeb (Penedo et al, 2023) 所示。 。根
我们对数据的观察，我们设计了一系列启发式过滤规则，重点关注分隔和换
的异常、异常字符的频率以及标点符号的分布。通过应用这些过滤器，我们
得了干净的数据。

原文: 可修改后右键重新翻译

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



Prompt in CIBench

System Prompt:
You are an assistant who can utilize external tools.
IPythonInterpreter: It can run Python code in a manner as jupyter notebook. The code must be a valid code that contains only python method.
To use a tool, please response with the following format:

Thought: Think what you need to solve, do you need to use tools?

Action: The tool name, should be one of IPythonInterpreter.

Action Input: The input to the tool that you want to use.

The tool will give you response after your response using the following format:

Response: the results after call the tool.

Therefore DO NOT generate tool response by yourself.

Also please follow the guidelines:
1. Always use code interpreter to solve the problem.
2. The generated codes should always in a markdown code block format.
3. The generated codes will be executed in an ipython manner and the results will be cached.
4. Your responded code should always be simple and only solves the problem in current step.

For example:

File url: xxxx
Step 1. Load the dataset from the url into a pandas DataFrame named df.

Thought: We should use pandas to solve this step.
Action: IPythonInterpreter
Action Input:

```
import pandas as pd \\
url = "xxxx" \\
data = pd.read\_csv(url) \\
```

Response: The code is succeed without any outputs.
Let us begin from here!

User Prompt:

{Question}. Please use {modules} modules.

Figure 19: Prompt used in CIBench

Prompt in CIBench

System Prompt:
You are an assistant who can utilize external tools.
{tool_description}
To use a tool, please use the following format:

```
\{thought\}Think what you need to solve, do you need to use
   tools? \\
\{action\}the tool name, should be one of [\{action\_names\}]
    \\
\{action\_input\}the input to the action \\
```

The response after utilizing tools should using the following format:

```
\{response\}the results after call the tool. \\
```

If you already know the answer, or you do not need to use tools,
please using the following format to reply:

```
\{thought\}the thought process to get the final answer \\
\{finish\}final answer \\
```

Begin!

Few-shot Prompt:

HUMAN: Find the coefficient of x^3 when $3(x^2 - x^3 + x) + 3(x + 2x^3 - 3x^2 + 3x^5 + x^3) - 5(1 + x - 4x^3 - x^2)$ is simplifie.
BOT:
Tool:PythonInterpreter
Tool Input:

```
from sympy import symbols, simplify

def solution(): \\
\quad x = symbols('x') \\
\quad expr = $3*(x**2 - x**3 + x) + 3*(x + 2*x**3 - 3*x**2 +
   3*x**5 + x**3) - 5*(1 + x - 4*x**3 - x**2)$ \\
\quad simplified\_expr = simplify(expr) \\
\quad x3\_coefficient = simplified\_expr.as\_coefficients\
   _dict()[x**3] \\
\quad result = x3\_coefficient \\
\quad return result \\
```

SYSTEM: Response:26

BOT: FinalAnswer: The final answer is 26. I hope it is correct.
...

Figure 20: Prompt used in MATH

翻译：

系统提示：您是一名可以使用外部工具的助手。

IPythonInterpreter：它可以像jupyter笔记本一样运行Python代码。该
码必须是仅包含 python 方法的有效代码。

要使用工具，请按照以下格式回答： 思考：想想你需要解决什么问题，需
使用工具吗？ Action：工具名称，应为 IPythonInterpreter 之一。

操作输入：您要使用的工具的输入。

在您响应后，该工具将使用以下格式给您响应： 响应：调用该工具后的
果。

因此，请勿自行生成工具响应。

另请遵循以下准则： 1. 始终使用代码解释器来解决问题。

2. 生成的代码应始终采用 Markdown 代码块格式。

3. 生成的代码将以ipython方式执行，并缓存结果。

4. 您响应的代码应该始终简单，并且仅解决当前步骤中的问题。

例如：

原文: 可修改后右键重新翻译

System Prompt: You are an assistant who can utilize external tools.

IPythonInterpreter: It can run Python code in a manner as jupyte
notebook. The code must be a valid code that contains only pytho
method.

To use a tool, please response with the following format: Though
Think what you need to solve, do you need to use tools? Action: Th
tool name, should be one of IPythonInterpreter.

Action Input: The input to the tool that you want to use.

The tool will give you response after your response using the followin
format: Response: the results after call the tool.

Therefore DO NOT generate tool response by yourself.