

亲测有效的几种研究选题方式

王 硕

CCMT 2024

三种选题方式

“先发制人”：选择一个全新的研究问题，或者用创新的方法解决现有的难题，开辟新的方向

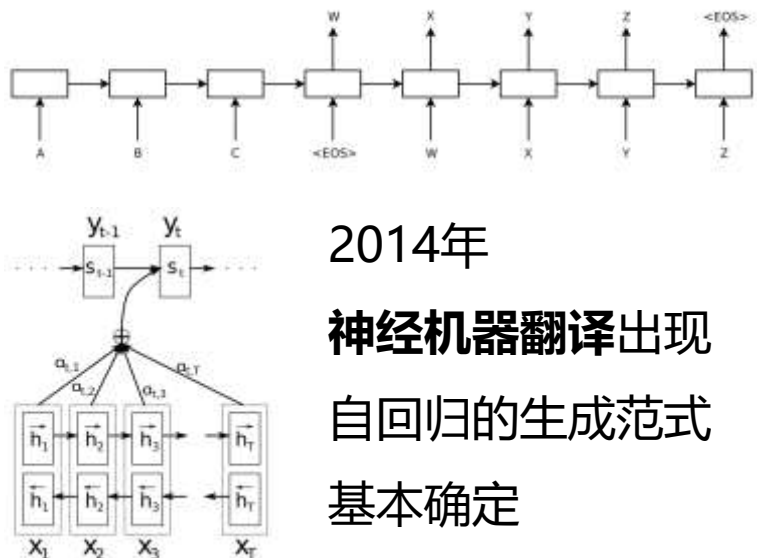
“后发制人”：根据当前主流方法中的不足，提出有针对性的优化或改进策略，或者扩展已有思想的应用场景，实现新的突破

“移花接木”：借鉴其他领域的思想或方法，如计算机科学的其他分支、甚至是跨学科的理论，来解决本领域的难题

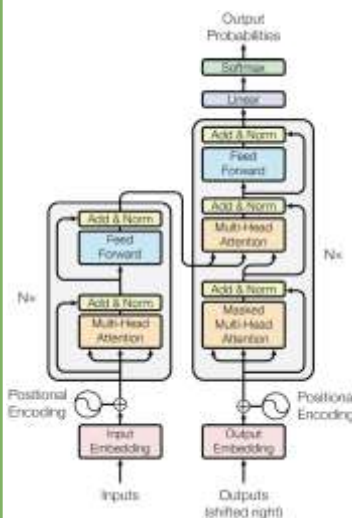
“先发制人”

- 不拘泥于现有的研究框架，开辟**全新的领域**，或者提出**全新的方法**
- 遵从**第一性原理**：一个最基本的命题或者假设，不能被省略，也不能被违反

全新的领域



全新的方法



2017年

Transformer出现

奠定大模型的模型基础

优势

竞争不激烈，好做

挑战

观点难以服人
新技术探索空间大

举例：使用大模型做狼人杀游戏

- **研究动机：**大模型是否具备群体博弈的能力

Idea形成的过程

使用大模型玩游戏



多模态模型能力欠佳

专注于文字冒险游戏



单人游戏已有TextWorld等前人工作

研究群体博弈游戏，狼人杀 or 谁是卧底

遇到的挑战

开源模型能力不足

没有成熟的GPT API服务

GPT-3.5也无法保证游戏的顺利完成

回合数多了之后超出GPT-3.5窗口大小

举例：使用大模型做狼人杀游戏

- **技术挑战：**从头搭建狼人杀模拟平台，有很多工程工作需要耐心处理

开发日志

• 基于Chatarena实现Werewolf Environment

- 7人制游戏，2狼人、2村民、1预言家、1女巫、1守卫

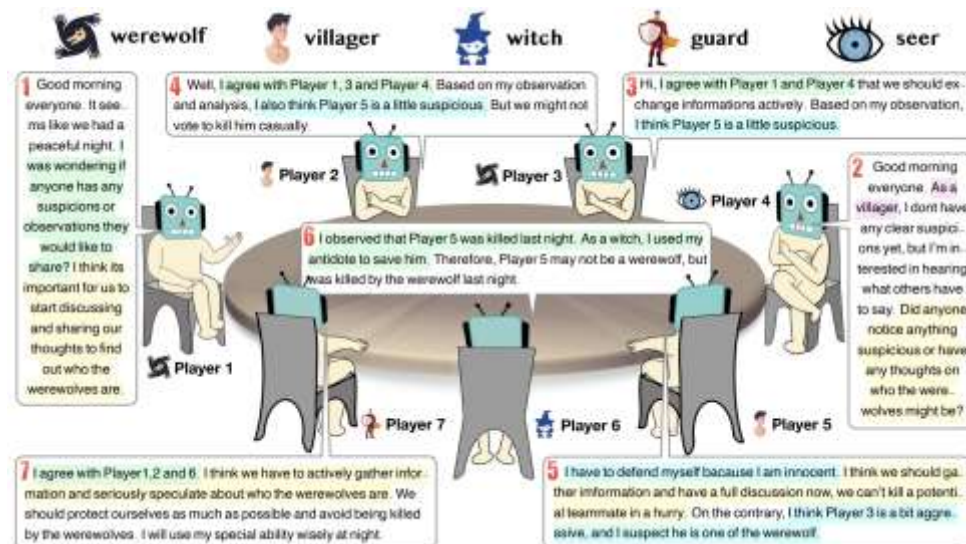
• 优化清单：

- ☐ 后端LLM模型选择，目前是gpt-3.5-turbo，后期可尝试其他模型
- ☒ GPT-3.5每分钟3次，成本无法满足需求，使用开源搭建的服务器
- ☒ 整体算法优化，尤其是黑夜决定谁死谁活的部分
- ☒ 单人说话时，常常有[player 1]、[player 2]这类引导信息，通过正则表达式
- ☒ LLM输出不可控，有冗余信息，考虑优化prompt
- ☒ 提升投票的准确度，采用选择题的方式
- ☒ 优化整个机制，使之从实际游戏场景到更贴近机器聊天实际情况
- ☐ 每个人额外提示本角色的详细的玩法
- ☐ 狼人投票的沟通环节（暂不实现）
- 目前仍然存在的问题：
 - 弱协同，没有表现出强协同，对其他玩家的发言敏感度不高
 - 存在幻觉，自编事实，通过设计prompt
 - prompt总长度受限，改用memory机制后该问题应该可以解决
 - 总结经验的能力不强，改进prompt，
- 目前对openai.py也进行少量改动
 - memory的实现或许在这里进行
- 可定制人数，身份的环境暂不考虑实现

☒ 优化openai.py中的prompt格式

- ☐ 每次检索最近观察的量，反思提问的量，必要时，调节最大token长度（已调节）
- ☒ 进一步优化prompt，仔细地调节user和assistant，尤其是反思过程的提示工程
- ☒ 前期仍然没有表现出足够智能的问题能否解决？已部分解决
- ☐ 在消息池的基础上实现经验池，并提供持久化方法（读写）
- ☒ 改openai调用方式为自建服务器调用
- ☒ 增加经验的写和读动作
- ☒ 经验的设计与使用
- ☒ 观察者身份等重要信息需要被检索访问
- ☒ 对观察者身份存在，提问，反思机制工作效果明显不高
- ☒ 消息重要度评级（无人说话=0，普通=1，暴身也/用预言家/获取连名=5，而用死了=3，人狼死亡=6，获取身份=10），检索时也检索重要消息，如果有反驳的时候，反驳的重要程度也需要评级？
- ☐ 后面加上经验时要注意message的多种类型问题，有些方法要做相应适配
- ☐ 使用环境的check_action解决典型的幻觉问题，如果后面有的话再解决
- ☒ 问答检索的相关度，存在幻觉问题，如果不相关，可以找不到，但不能胡说！另外，如果有若干条答案相关，应该最多允许不超过3条高度相似的答案拼接成最终答案。另外，身份竟然检索不到？！本地图解，让大模型精确地选择答案（或重新组织答案内容），让选择的个数可能更大些！！！！
- ☒ 白天讨论Talk with other players，投票的改进，只选取最后一句话输出。
- ☐ 观察到向第一个死了的狼人对话并且有输出的问题！暂时搁置，后面再查找原因！！
- ☐ 如何解决经验池中的玩家名和当前对局玩家名不匹配的问题，需要解决吗？
- ☒ 无人说话交互时仍然在反思——干脆就死人不说啥就完了！

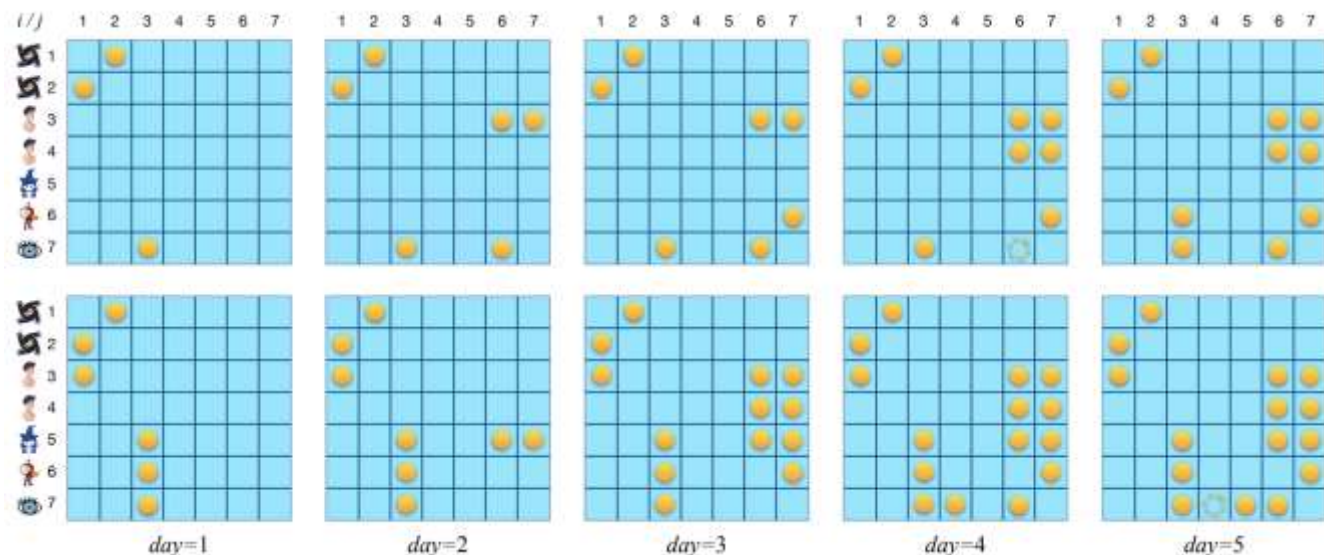
平台效果



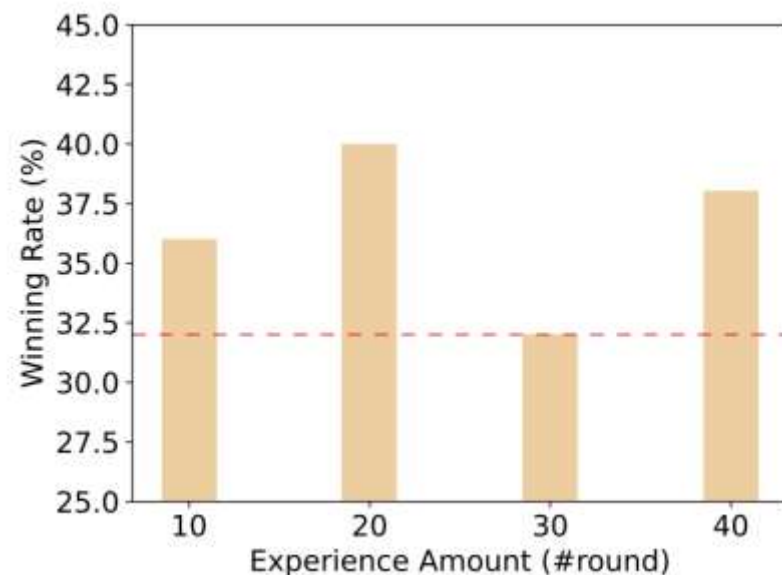
举例：使用大模型做狼人杀游戏

- **实验发现：**大模型在游戏过程中已经可以在一定程度上表现出类似人类的行为

信任关系的涌现



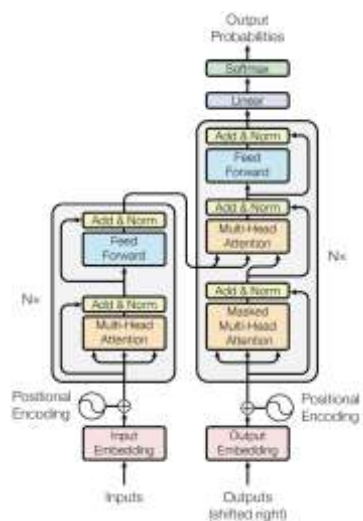
历史经验的影响



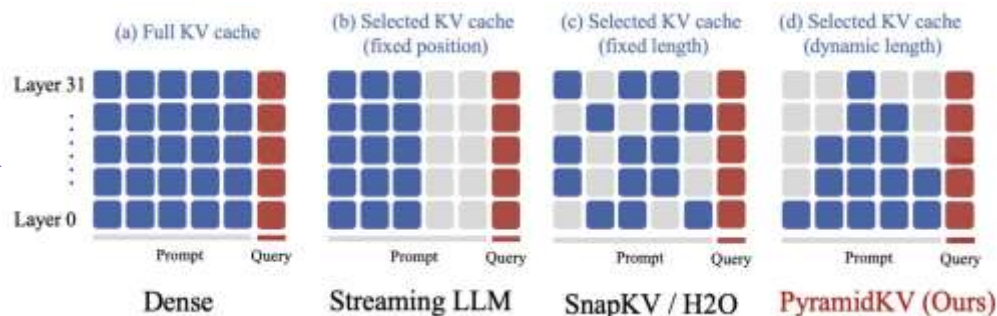
“后发制人”

- 分析当前最好的、或者最具代表性的方法存在的问题是什么
- “先发制人” 需要克服万难的信念感，“后发制人” 需要“敌进我退” 的灵活性

针对当前方法不足做改进



解决KV-Cache
过大的问题



优势

目标明确

一般超过基线方法即可

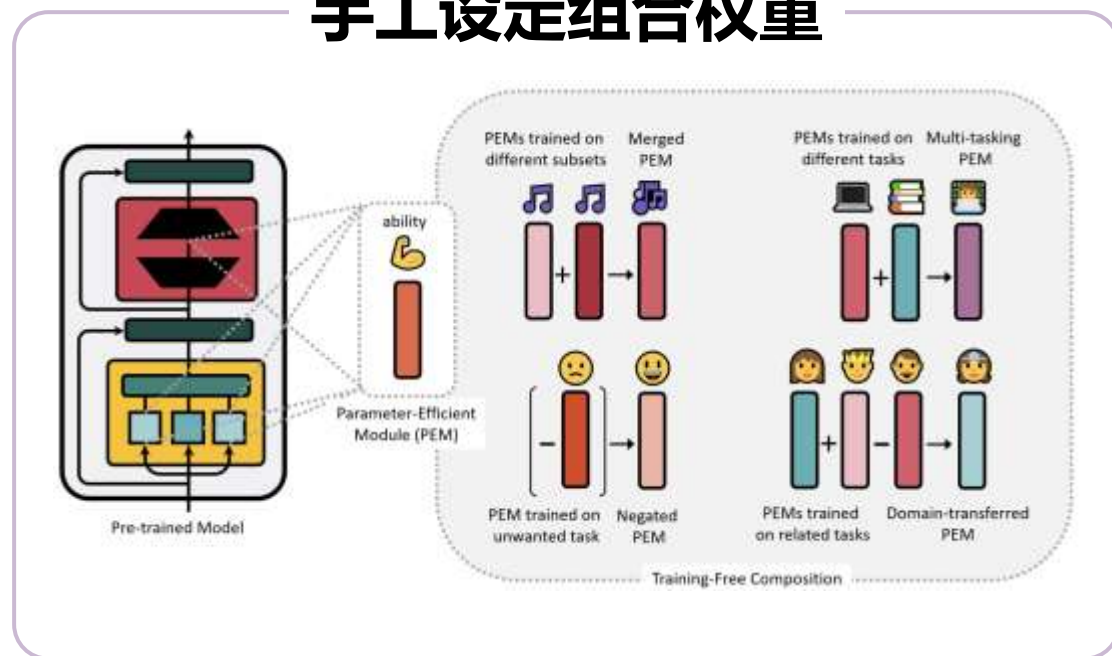
挑战

基线方法往往很强
同期工作往往很快

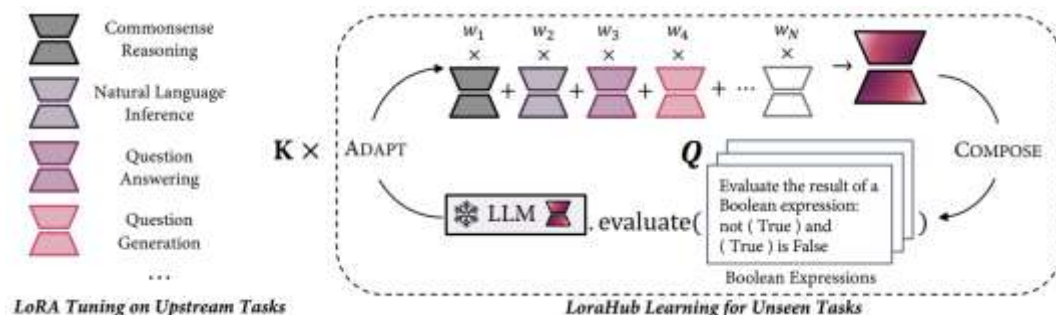
举例：面向生成式任务的LoRA组合方法

- **研究动机：**组合训练好的LoRA模块，实现模型能力的快速扩展
- **前人方法：**为每个LoRA模块分配一个组合权重，权重是人工设定的或学习得到的

手工设定组合权重



Few-Shot方式学习组合权重



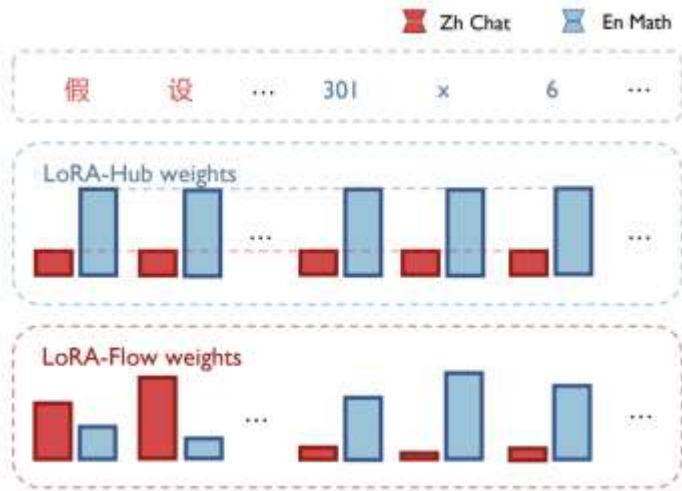
举例：面向生成式任务的LoRA组合方法

- **前期观察：**现有工作多数在分类任务上进行的实验，在生成式任务上表现如何？
- **猜想假设：**对于生成式任务，不同类型的token对各个LoRA模块的依赖程度是变化的

现有工作实验结果

Method		MNLI	RTE	SST-2	MRPC	QNLI	QQP	CoLA	STS-B
FFT	fullset	76.6	75.8	92.5	88.5	85.9	81.8	0.56	0.90
	s_0	72.0	72.9	90.4	85.8	83.4	79.2	0.42	0.88
	s_1	71.9	67.5	92.0	88.5	83.2	81.5	0.52	0.89
	m	74.2 $\uparrow 2.3$	75.1 $\uparrow 4.9$	92.1 $\uparrow 0.9$	89.2 $\uparrow 2.1$	83.8 $\uparrow 0.5$	81.9 $\uparrow 1.5$	0.55 $\uparrow 0.07$	0.89 $\uparrow 0.01$
LoRA	fullset	87.1	79.8	95.0	89.2	93.4	90.2	0.63	0.91
	s_0	71.4	72.2	92.2	86.3	83.1	79.0	0.50	0.88
	s_1	72.3	69.0	91.9	87.7	83.0	80.8	0.51	0.89
	m	73.5 $\uparrow 1.6$	75.8 $\uparrow 5.2$	92.2 $\uparrow 0.2$	88.0 $\uparrow 1.0$	83.3 $\uparrow 0.2$	81.1 $\uparrow 1.2$	0.52 $\uparrow 0.01$	0.89 $\uparrow 0.01$
(IA) ³	fullset	75.9	74.0	92.3	87.3	84.7	80.8	0.56	0.89
	s_0	71.7	72.9	90.8	85.8	83.0	78.3	0.44	0.87
	s_1	71.7	68.2	91.2	88.0	82.5	80.8	0.50	0.90
	m	74.0 $\uparrow 2.3$	74.7 $\uparrow 4.0$	92.3 $\uparrow 1.3$	88.2 $\uparrow 1.3$	84.8 $\uparrow 2.0$	81.3 $\uparrow 1.8$	0.50 $\uparrow 0.03$	0.90 $\uparrow 0.01$

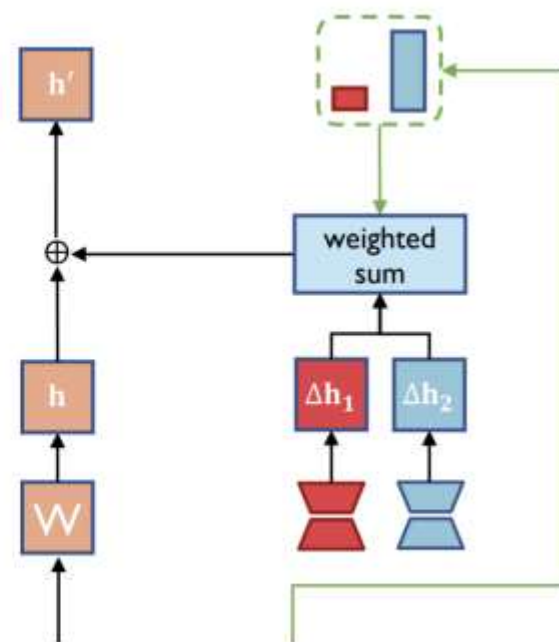
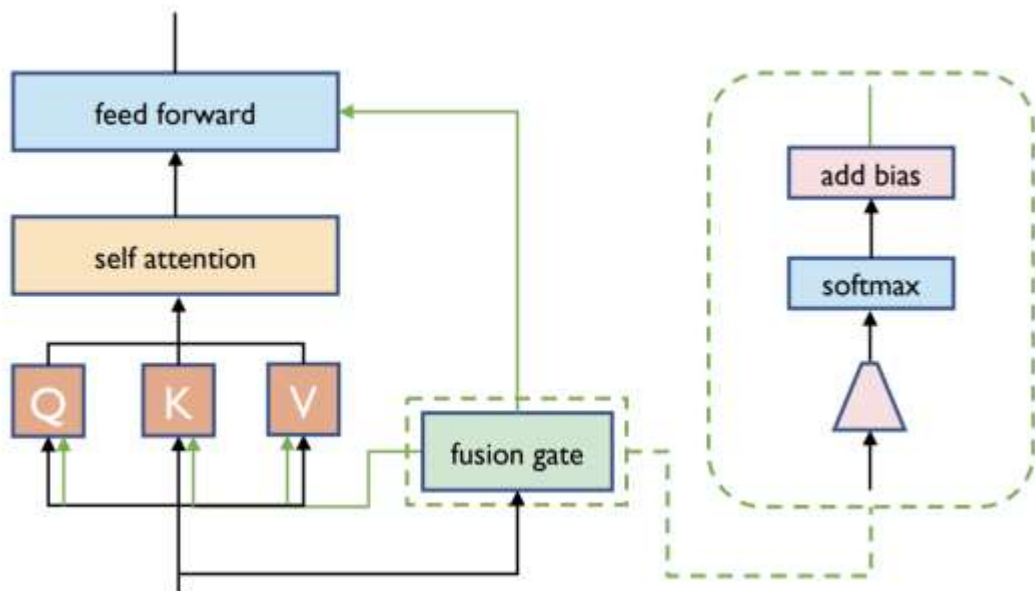
猜想：生成式任务需要动态组合权重



举例：面向生成式任务的LoRA组合方法

- **方法设计**：增加一个参数量很小的融合模块，以Few-Shot的方式学习动态组合权重
- **新增成本**：融合模块参数量仅为LoRA模块的0.2%，仅需200个样本训练

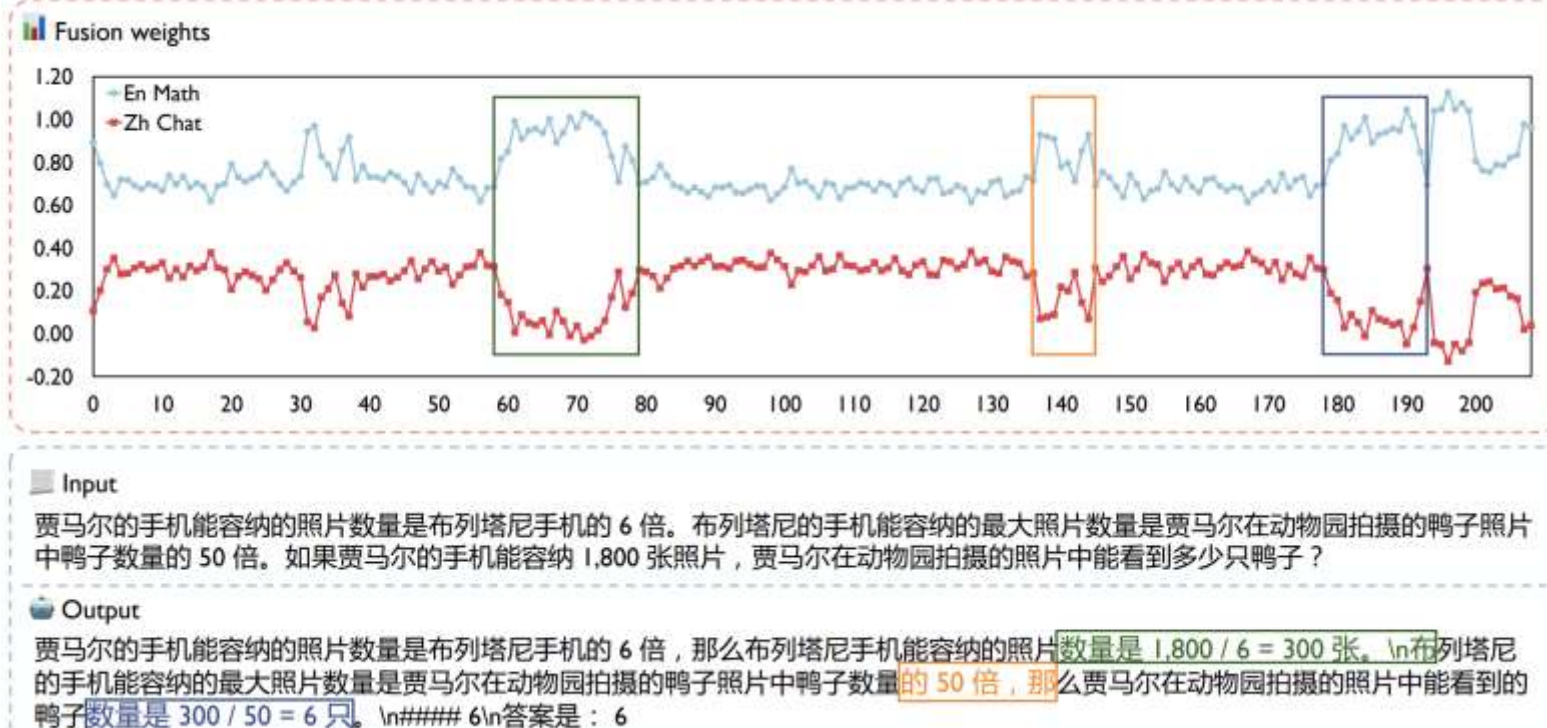
方法设计



举例：面向生成式任务的LoRA组合方法

- 实验分析，这个方法确实可以再生成式任务上学习到有效的LoRA组合模式

可视化分析



举例：面向生成式任务的LoRA组合方法

- 实验发现，在生成式任务上，动态组合权重可以明显提升方法的组合效果

与前人工作对比

Method		MGSM (Math)				HumanEval (Code)			
		Zh	Ru	Es	Avg.	Zh	Ru	Es	Avg.
BASE MODEL		4.4	3.2	2.4	3.3	0.0	0.0	2.4	0.8
SINGLE LoRA	LANG	5.2	3.6	3.6	4.1	12.2	14.0	10.4	12.2
	TASK	26.8	32.8	41.2	33.6	18.3	23.2	21.9	21.1
LoRA FUSION	AVERAGE	12.8	10.4	18.4	13.9	17.1	17.7	18.3	17.7
	LoRA-HUB	20.8	28.4	36.8	28.7	19.5	21.3	20.1	20.3
	LoRA-FLOW	33.2	37.6	42.0	37.6	20.7	23.8	23.2	22.6

消融实验

Method	MGSM			
	Zh	Ru	Es	Avg.
TASK LoRA	26.8	32.8	41.2	33.6
LoRA-HUB	20.8	28.4	36.8	28.7
STEP-LEVEL	30.0	32.4	44.0	35.5
LAYER-LEVEL	33.2	37.6	42.0	37.6
MODULE-LEVEL	30.4	34.0	42.4	35.6

“移花接木”

- 相信 **“它山之石可以攻玉”**，广泛阅读其他领域的前沿工作/经典工作，汲取灵感

AI其他领域

计算机视觉

强化学习

机器人

图神经网络

...

计算机学科其他领域

算法与数据结构

数据库系统

软件工程

操作系统

...

举例I：受Git版本管理启发的模型参数压缩

- 相信“它山之石可以攻玉”，广泛阅读其他领域的前沿工作/经典工作，汲取灵感

Git版本管理

存储所有版本



存储版本增量



查阅文件系统资料

差分编码 [编辑](#) 10种语言

条目 讨论 阅读 大陆简体

阅读 编辑 查看历史 工具

此条目没有列出任何参考或来源。(2018年6月24日)

维基百科所有的内容都应该可供查证。请协助补充可靠来源以改善这篇条目。无法查证的内容可能会因为异议提出而被移除。

差分编码（英语：Delta encoding），又称**增量编码**，是指在**序列式**资料之间以**数据差异**形式存储或发送资料的方式（相对于存储发送完整文件的方式）。通常应用于目标文件和参考文件之间存在高度冗余的情况。使压缩后文件大小比仅使用目标文件压缩小，可以提高网络和磁盘资源的使用效率。

差异存储称为“delta”或“diff”的不连续文件中。由于改变通常很小（平均占全部大小的2%），差分编码能大幅减少资料的重复。一连串独特的delta文件在空间上要比特未编码的相等文件有效率高了。

差分编码的简单例子是存储序列式资料之间的差异（而不是存储资料本身）：不存“2, 4, 6, 9, 7”，而是存“2, 2, 2, 3, -2”。单独使用用处不大，但是在序列式数值串出现时可以帮助压缩资料。

应用 [编辑](#)

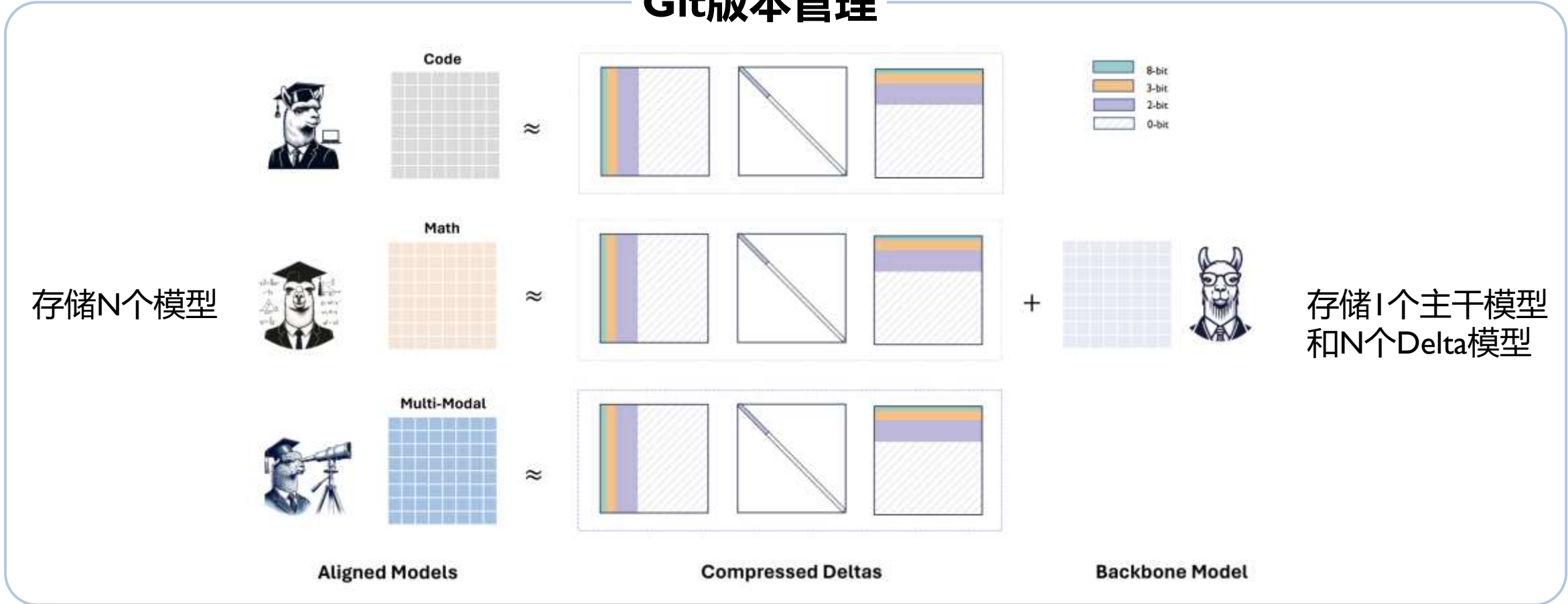
典型的应用场景包括管理有多版本的文件。透过网络传输软件的补丁或更新包，此时需要接收者已经拥有旧版本的资料，或是当网页有共同的布局和菜单结构时可以压缩。需要查看文件的历史更改记录（[版本控制](#)、git等），Windows中的**远程差分压缩**，[在线备份](#)等场景均用到了差分编码。大多数使用增量编码的文件为文字或二进制文件。

RCS和SCCS是两种传统系统，为单一文件提供增量压缩的版本控制。更新的版本控制系统，如CVS和PRCS强调对文件集合（而不是单一文件）提供版本控制的重要性，因此常使用RCS和SCCS进行后援存储而不是用于版本控制。

举例I：受Git版本管理启发的模型参数压缩

- 借鉴差分编码（Delta Encoding）思想，实现对模型的版本管理

Git版本管理



举例I：受Git版本管理启发的模型参数压缩

- **方法设计：**基于观察到的差值矩阵的低秩特性，使用高位宽编码特征值较大的参数，使用低位宽编码特征值较小的参数，实现混合精度压缩

对Delta参数进行混合精度压缩

模型压缩：low-bit或者low-rank

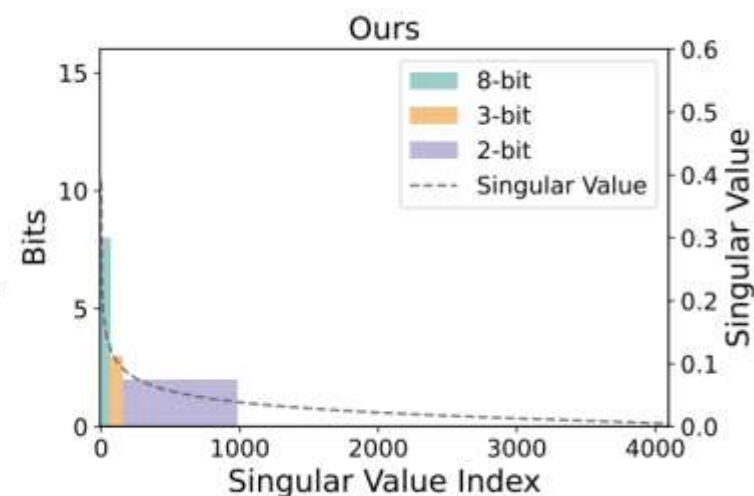
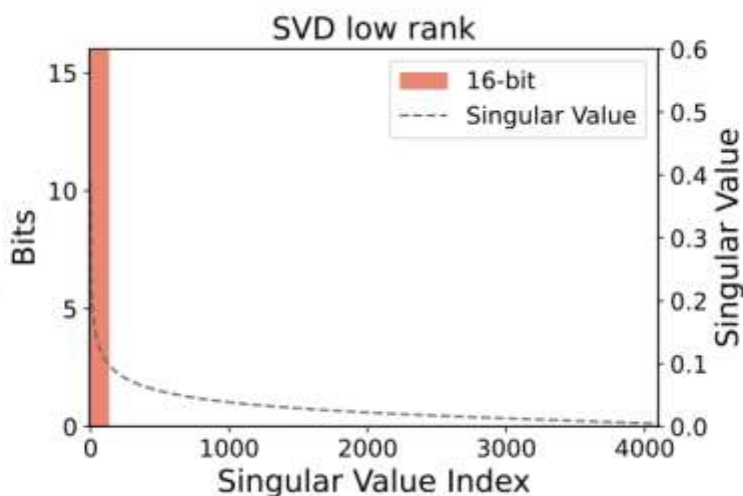
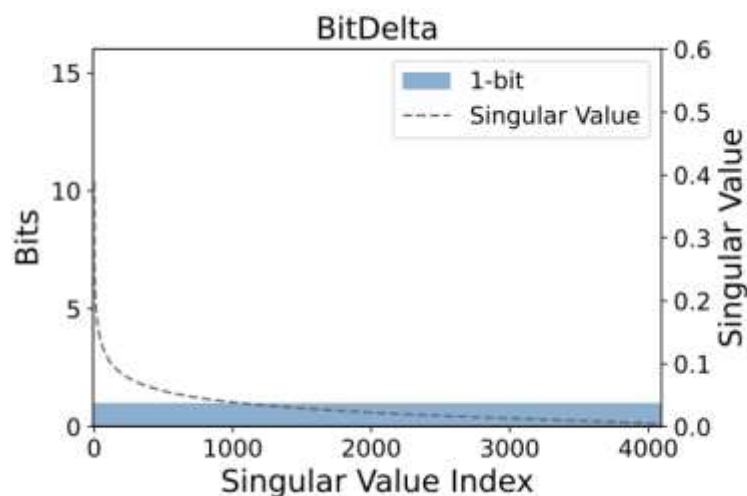


Delta压缩和模型压缩有什么不同？

Delta参数具有明显的低秩特性（长尾分布）



混合精度Delta压缩



举例I：受Git版本管理启发的模型参数压缩

- 实验发现：混合精度压缩方法平均可以达到与压缩前模型可比的效果

实验结果

Method	α	数学评测		代码评测		Chat评测		多模态评测		Ave.
		WIZARDMATH		MAGICODERS-CL		LLAMA-2-CHAT		LLAVA-V1.5		
		GSM8K	MATH	HumanEval	MBPP	TruthfulQA	SafetyBench	GQA	TextVQA	
Backbone	1	11.0	2.9	38.4	47.6	41.7	38.9	n/a	n/a	n/a
Aligned	1	55.2	10.9	70.7	69.2	59.5	44.6	62.0	58.2	53.5
Low-Rank	1/16	43.2	8.0	56.7	65.7	55.4	42.5	57.7	53.3	47.8
BitDelta	1/16	45.6	8.6	57.3	65.9	59.3	41.1	59.7	56.9	49.3
Delta-CoMe	1/16	53.6	10.3	67.1	67.9	59.8	47.0	61.7	58.5	53.2

举例I：受Git版本管理启发的模型参数压缩

- 实验发现：与传统PEFT方法相比，各有利弊

实验结果

Method	Math		Code		Ave.
	GSM8K	MATH	HumanEval	MBPP	
Backbone	11.0	2.9	10.5	17.7	10.5
Aligned	65.4	18.6	43.2	44.9	43.0
LoRA	58.3	11.4	17.6	31.8	29.8
Low-Rank	54.8	5.5	26.2	42.6	32.3
BitDelta	47.8	10.7	26.2	41.9	31.7
Delta-CoMe	65.1	18.0	39.6	44.9	41.9

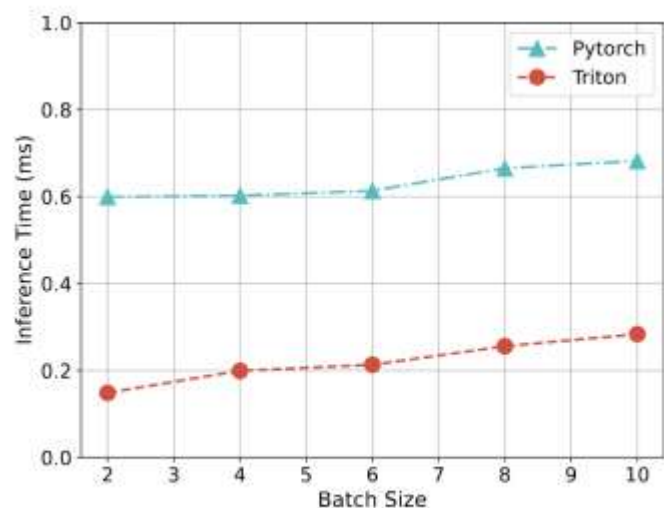
优势
方法效果好

不足
只能节省推理成本
不能节省训练成本

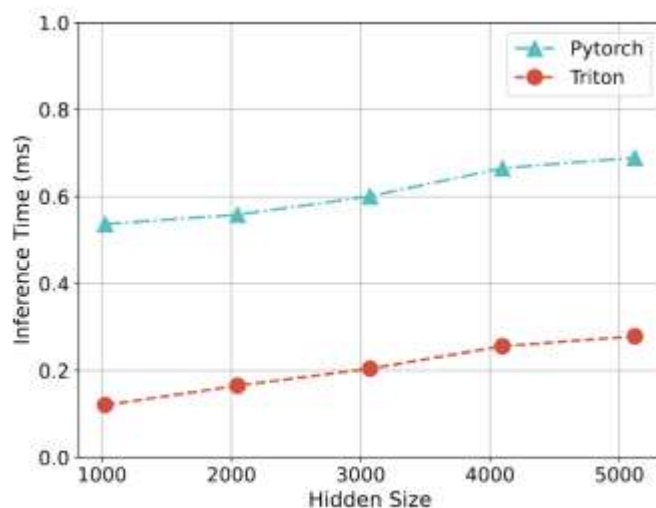
举例I：受Git版本管理启发的模型参数压缩

- **实验发现：**在多租户等需要部署多个模型的场景下，可以有效节约成本

实现Triton算子提升硬件兼容性



(a) Effect of batch size.



(b) Effect of hidden size.

节省Memory

Table 7: GPU memory cost (GB).

Num. of Models	w/o DC	w/ DC
2	26.67	15.54
4	52.24	18.17
8	OOM	23.44
16	OOM	33.95
32	OOM	55.06
50	OOM	78.70

举例2： LLM×MapReduce: 基于分治思想的长序列理解框架

- **优势**：长度可扩展，不受限于模型本身窗口大小，理论上支持任意长度输入
- **劣势**：将一个完整的长文档切分为多个片段，可能会破坏跨片段的关键信息，导致模型根据某个片段“**断章取义**”，产生错误结论

LLM×MapReduce技术挑战

跨片段依赖 (Inter-Chunk Dependency)：多个片段的信息相互依赖，综合起来，才能产生一个完整的答案
比如，我们要总结一个事件的时间脉络，需要从很多个片段中提取关键的信息，才能形成一个完整的时间线



结构化通信协议
Structured Information Protocol

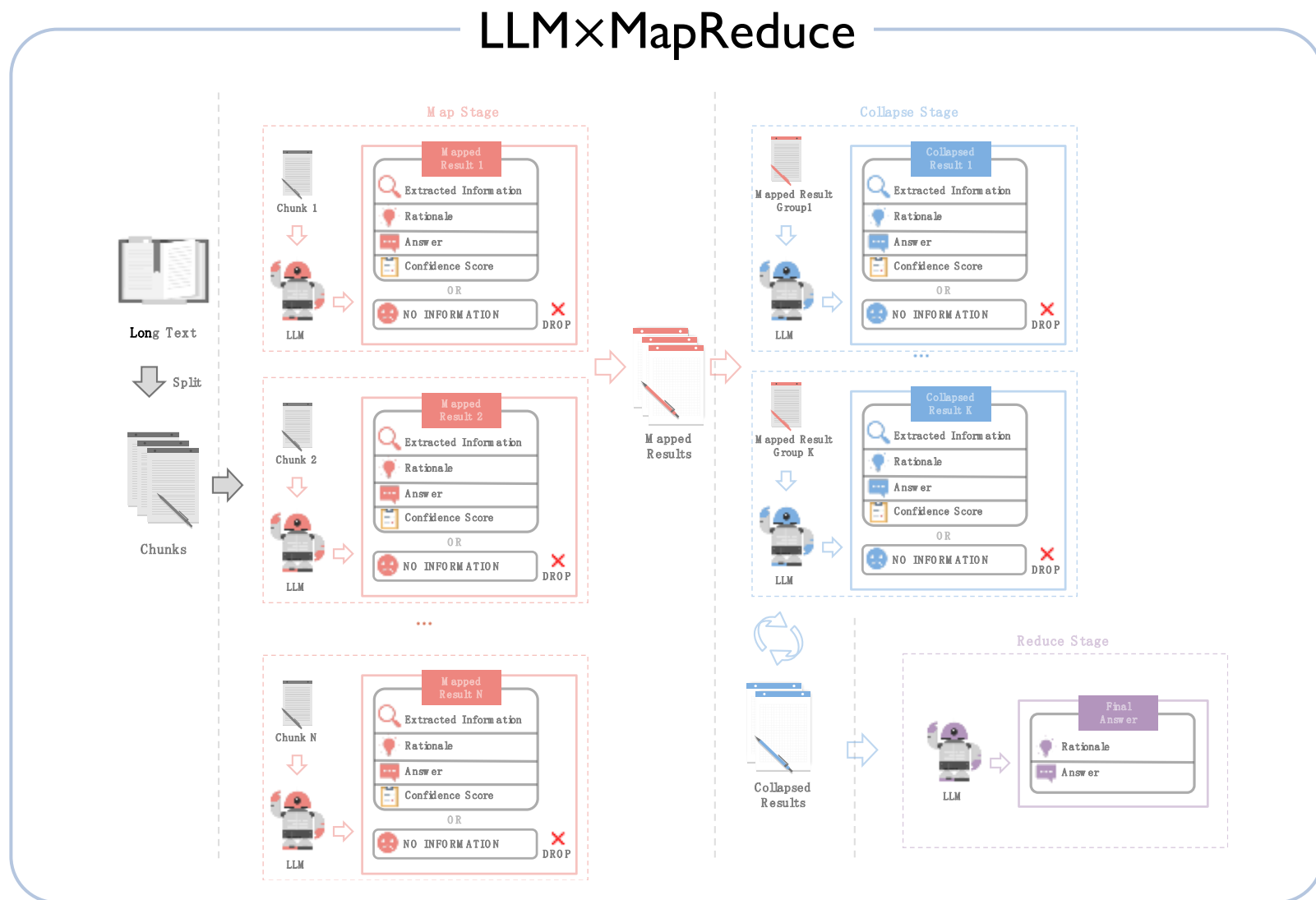
跨片段冲突 (Inter-Chunk Conflict)：多个片段的信息存在冲突，看不同的片段时可能得出不同的结论
比如，我们要问一位导演一生最伟大的作品是什么，在看他青年时期片段时，得出的是青年时的最大成就，而在看老年时期的片段，又会得出不同的结论



上下文置信度校准
In-Context Confidence Calibration

举例2： LLM×MapReduce: 基于分治思想的长序列理解框架

- **结构化通信协议：**模型在处理每个片段时，不是仅仅输出中间答案，而是输出结构体，包含丰富的相关信息
- **上下文置信度校准：**为了让模型在处理不同片段时具有一致的置信度评估标准



举例2： LLM×MapReduce: 基于分治思想的长序列理解框架

- Baseline：闭源模型、开源模型、其他分治框架（LongAgent & Chain-of-Agents）

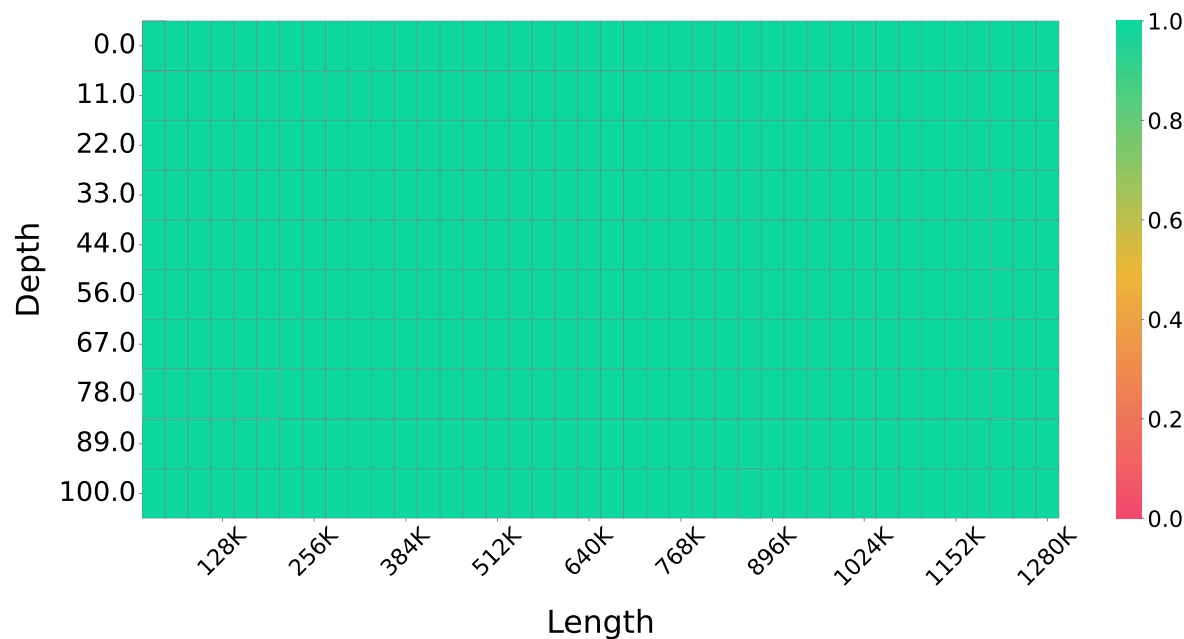
InfiniteBench评测结果

Methods	Re.Pa	Re.Nu	Re.KV	En.Sum	En.QA	En.MC	En.Dia	Co.De	Ma.Fi	Avg.
<i>Closed-Source Models</i>										
GPT-4*	100.00	100.00	89.00	14.73	22.44	68.12	7.50	54.31	60.00	57.34
Claude 2*	97.80	99.15	65.40	14.50	11.97	67.25	43.00	33.24	32.29	51.62
Kimi-Chat	99.32	97.45	69.20	29.94	18.81	79.91	15.50	38.32	18.57	51.89
<i>Open-Source Models</i>										
YaRN-Mistral*	92.71	58.31	0.00	9.09	9.55	29.26	4.50	23.60	17.14	27.13
Yi-6B-200K*	100.00	94.92	0.00	0.92	9.20	36.68	1.50	18.78	4.29	29.59
Yi-34B-200K*	100.00	100.00	0.00	1.33	12.17	46.29	3.50	21.32	25.71	34.48
Q2-72B-I	100.00	100.00	29.00	31.85	21.97	81.66	23.00	45.43	59.71	54.74
<i>Divide-and-Conquer Frameworks</i>										
L3-70B-I+LA	99.32	93.05	74.60	2.19	35.41	69.00	7.50	24.11	79.14	53.81
L3-70B-I+CoA	9.32	15.59	1.80	10.10	7.03	27.51	9.50	18.27	44.57	15.97
MiniCPM3×MR	98.81	99.32	93.80	25.89	28.39	66.38	12.50	25.63	83.43	59.35
L3-70B-I×MR	100.00	99.79	98.89	30.63	34.70	82.10	17.50	62.94	91.43	68.66
Q2-72B-I×MR	100.00	100.00	98.80	32.39	23.13	83.84	46.50	54.82	54.29	65.97

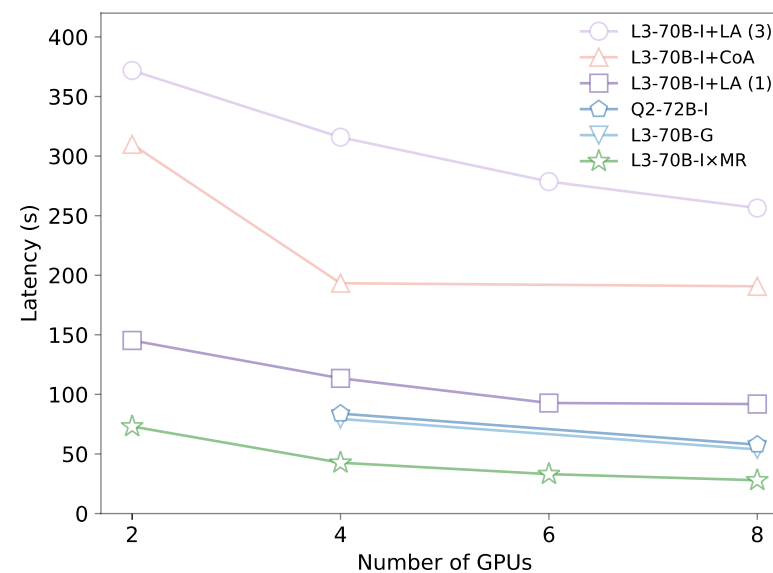
举例2： LLM×MapReduce: 基于分治思想的长序列理解框架

- 超长序列（1280K token）评测 & 效率评测

1280K NIAH



推理速度评测



请各位老师同学批评指正