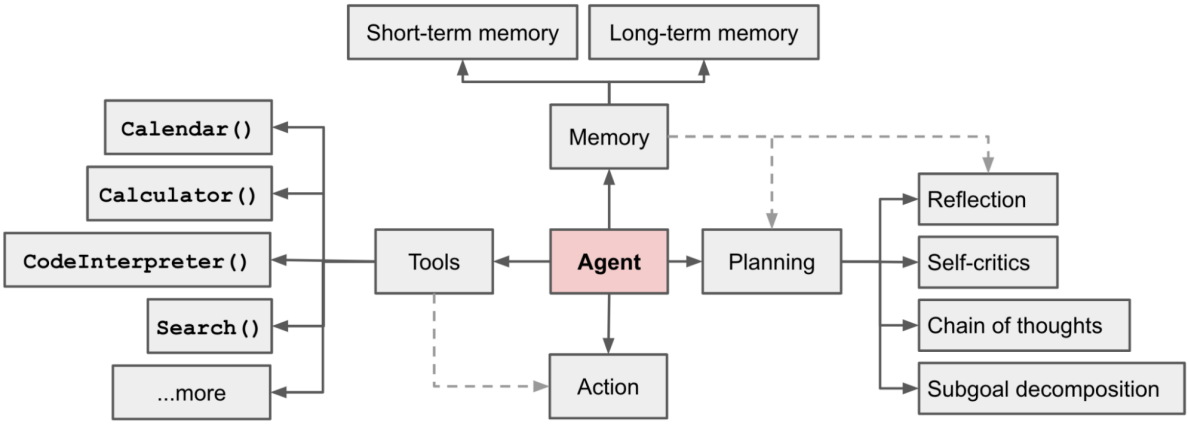
LLM-based Agent 调研

在由 LLM 驱动的自主代理系统中，LLM 充当代理的大脑，主要由几个关键组件组成：

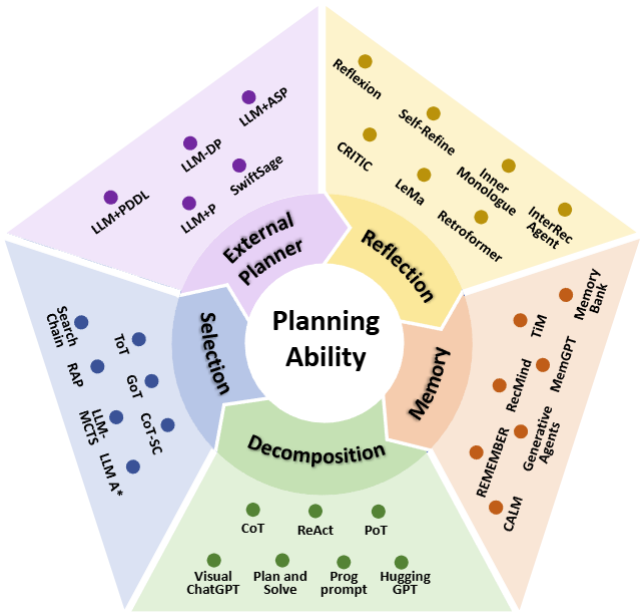
- **规划**
  - 子目标和分解：代理将大任务分解为更小的、可管理的子目标，从而有效地处理复杂任务。
  - 反思与改进：代理可以对过去的行为进行自我批评和自我反思，从错误中吸取教训，并为未来的步骤进行改进，从而提高最终结果的质量。
- **记忆**
  - 短期记忆：几乎所有的情境学习（参见提示工程）都是利用模型的短期记忆来学习。
  - 长期记忆：这为代理提供了在较长时间内保留和回忆（无限）信息的能力，通常是通过利用外部向量存储和快速检索来实现的。
- **工具使用**
  - 代理学习调用外部 API 来获取模型权重中缺少的额外信息（预训练后通常很难改变），包括当前信息、代码执行能力、专有信息源的访问等。



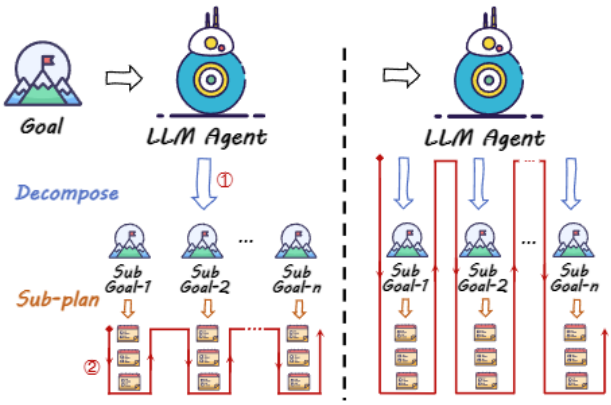
关于Agent 的规划能力

分类

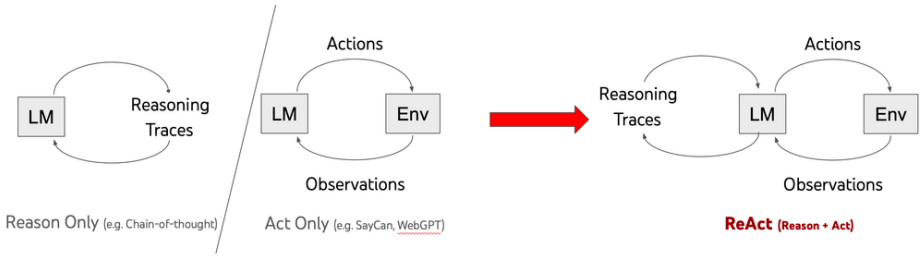
可分为**任务分解、多计划选择、外部模块辅助规划，反思和细化 和记忆增强规划**五个类别。



1. **任务分解Task Decomposition**
   - 基本思想：分而治之。分解成多个子任务，依次对每个子任务进行规划

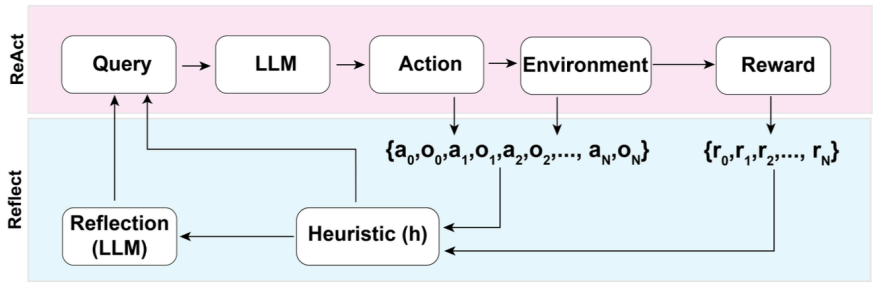


   - 代表性工作：**CoT [2022], ReAct [2022], HuggingGPT [2023]，Plan-and-Solve[2023], PAL [2023]**

> - CoT 通过一些构建的轨迹指导 LLM 推理复杂问题，利用 LLM 的推理能力进行任务分解。
> - ReAct 将推理与规划分离。它在推理（思维步骤）和规划（行动步骤）之间交替进行。通过将动作空间扩展为特定于任务的离散动作和语言空间的组合，将推理和动作集成到 LLM 中。前者使 LLM 能够与环境交互（例如使用 Wikipedia 搜索 API），而后者促使 LLM 生成自然语言中的推理轨迹。
>
>   格式：`Thought, Action, Action Input, Observation`
>
> 
>
>
> - HuggingGPT利用 Huggingface Hub 中的各种多模态模型构建了一个用于多模态任务的智能代理。LLM 充当控制器，负责分解人类输入的任务、选择模型并生成最终响应。

    - 利用 LLM 的编码能力改进 CoT，引导 LLM 在推理过程中生成代码。
  - 挑战：
    - 任务分解带来的额外开销。将一个任务分解为多个子任务需要更多的推理和生成
    - 规划受到 LLM 上下文长度的限制，导致规划轨迹被遗忘。
2. **多计划选择Multi-Plan Selection**
  - 基本思想：多计划生成和最优计划选择。考虑在生成模型的解码过程中使用不确定性。
  - 代表性工作：**ToT [2023], GoT [2023], CoT-SC [2022b]，Self-consistency[2022b]**
    - Self-consistency:通过解码过程中体现的采样策略（例如温度采样、top-k 采样）获得多条不同的推理路径。采用**朴素多数投票策略**，将获得最多票数的计划视为最优选择
    - 思维树 (ToT) 提出了两种生成计划（即想法）的策略：采样和提议。采样策略与自洽性一致，其中 LLM 会在解码过程中采样多个计划。提议策略明确指示 LLM 通过提示中的少量示例生成各种计划。 利用传统的 BFS 和 DFS进行最优选择计划
    - 思维图 (GoT) 通过添加想法的转换来扩展 ToT，从而支持任意想法的聚合。
    - LLM-MCTS [2023b] 和 RAP [2023] 利用 LLM 作为蒙特卡洛树搜索 (MCTS) 的启发式策略函数，其中通过多次调用可以获得多个潜在动作。
    - LLM A* [2023] 利用人工智能中的经典 A* 算法来协助 LLM 进行搜索。当前位置到目标位置的切比雪夫距离作为选择最佳路径的启发式成本函数。
  - 挑战：
    - 计算需求增加，资源限制
    - 过于依赖 LLM 来评估计划。由于 LLM 在排名任务中的表现仍在审查中，因此需要进一步验证和微调其在这种特定情况下的能力。LLM 的随机性增加了选择的随机性，可能会影响所选计划的一致性和可靠性。
3. **外部模块辅助规划External Module-aided planning**
  - 基本思想：将任务形式化，并利用外部计划
  - 代表性工作：**LLM+P [2023a], LLM+PDDL [2023]**
    - LLM+P [2023a]通过引入基于pddl的符号规划器来提高LLM的规划熟练度。利用LLM的语义理解和编码能力，作者将问题组织成文本语言提示输入到LLM。这将提示LLM将环境中的操作和指定任务组织成PDDL语言的格式。随后，在获得形式化描述后，作者采用快速向下求解器求解规划过程。
    - LLM- dp [2023]是专门为动态交互环境设计的。LLM在接收到来自环境的反馈后，对信息进行处理，将其形式化为PDDL语言，然后使用BFS 求解器生成计划。
    - LLM+PDDL [2023]也利用PDDL语言来形式化任务，合并了一个额外的手动验证步骤，以检查LLM生成的PDDL模型中的潜在问题。
    - SwiftSage [Lin et al.， 2023]利用认知心理学中的双过程理论，将规划过程分为慢速思维和快速思维。
4. **反思和细化（Reflection and Refinement）**
  - 基本思想：反思经验，完善计划。让自主代理能够通过改进过去的行动决策和纠正以前的错误来不断改进的重要方面。
  - 代表性工作：**Reflexion [NIPS 2023], CRITIC [2023], Self-Refine [2023]**

- Self-refine [2023]利用生成、反馈和改进的迭代过程。在每一代之后，LLM为计划生成反馈，便于根据反馈进行调整。
- reflection [2023]通过纳入评估器来评估轨迹，扩展了ReAct。LLM在检测到错误时产生自我反射，帮助纠错。



- RITIC [2023]使用知识库和搜索引擎等外部工具来验证LLM生成的操作。然后，它利用外部知识进行自我纠正，显著减少事实错误。
- InteRecAgent [2023b]采用一种名为ReChain的机制进行自我纠错。LLM用于评估交互式推荐代理生成的响应和工具使用计划，总结错误反馈，并决定是否重新启动计划。
- LEMA [2023]首先收集错误的规划样本，并使用更强大的GPT-4进行校正。然后，这些校正后的样本被用于微调LLM-Agent，从而在LLaMA模型的各个尺度上显著提高性能。

- 挑战：
  - 这种文本形式的更新的收敛性目前缺乏有保证的证明，这表明无法证明持续的反思最终可以使LLM代理达到指定的目标。

5. **记忆增强规划Memory-augmented planning**
   - 基本思想：利用记忆来帮助规划。基于**rag的记忆和具身记忆。**
   - 代表性工作：**REMEMBER [2023a], MemoryBank [2023]**

     - Generative Agent[2023]以文本形式存储类人代理的日常经验，并根据近代性和与当前情况的相关性的综合分数检索记忆。
     - MemoryBank [2023]、TiM [2023b]和RecMind [2023c]使用文本编码模型将每个内存编码为一个向量，并建立索引结构，如FAISS库。三者的区别在于记忆更新的方式。
     - MemGPT [Packer等人，2023]利用了计算机体系结构中多层存储的概念，将LLM的上下文抽象为RAM，并将额外的存储结构视为磁盘。LLM可以自发地决定是检索历史记忆还是将当前上下文保存到存储器中。
     - REMEMBER [2023a]以q值表的形式存储历史记忆，其中每条记录都是（环境、任务、动作、q值）元组。在检索过程中，LLM同时检索积极记忆和消极记忆，并根据环境和任务的相似性生成计划。
     - CALM [2020b]利用从文本世界环境中收集的ground-truth动作轨迹，使用下一个令牌预测任务对GPT-2进行调优，使其能够记忆与规划相关的信息，并在规划任务上进行良好的泛化。
     - AgentTuning [2023]将来自各种任务的计划轨迹组织到一个对话形式中，以微调LLaMA模型

实验评估

基准测试方法与数据集：

1. 交互式游戏环境，如Minecraft 4，
2. 基于文本的交互式环境，如知识密集型任务（HotpotQA、FEVER）和决策任务ALFWorld、ScienceWorld
3. 交互式检索环境：模拟了人类在现实生活中进行信息检索和推理的过程。例如基于 Wikipedia 引擎 WebShop、Mind2Web 和 WebArena 。
4. 交互式编程环境

Agent主流框架：

Agent	Type	Memory			Methodology	Domain	Environment Interaction		External Tools		
		Operation	Short-term	Long-term			Modality	Model	Web	Code	Other
CAMEL (Li et al., 2023a)	Communicative	Prompt	✓	-	Prompting	AI Society & Coding	Text	GPT-3.5-Turbo	-	-	-
Generative Agents (Park et al., 2023)	Communicative	Tree Search	✓	✓	Prompting	AI Society	Text	GPT-3.5-Turbo	-	-	-
Voyager (Wang et al., 2023c)	Communicative	Tree Search	✓	✓	Prompting	MineCraft	Text	GPT-4	-	-	-
GITM (Zhu et al., 2023)	Communicative	Tree Search	✓	✓	Prompting	MineCraft	Text	GPT-3.5-Turbo	-	-	-
MetaGPT (Hong et al., 2023)	Communicative	Text Retrieval	✓	✓	Prompting	AI Society & Coding	Text	GPT-4	✓	✓	✓
ChatDev (Qian et al., 2023)	Communicative	Text Summary	✓	✓	Prompting	Software Engineering	Text	GPT-3.5-Turbo	-	✓	-
MAD (Liang et al., 2023)	Communicative	Prompt	✓	-	Prompting	Reasoning	Text	GPT-3.5-Turbo	-	-	-
Multiagent Debate (Du et al., 2023)	Communicative	Prompt	✓	-	Prompting	Reasoning	Text	GPT-3.5-Turbo	-	-	-
FORD (Xiong et al., 2023a)	Communicative	Prompt	✓	-	Prompting	Reasoning	Text	GPT-4	-	-	-
AutoGPT (Richards, 2023)	Autonomous	Vector Search	✓	✓	Prompting	Task Management	Text Speech Image	GPT-4 DALL-e ElevenLabs	✓	✓	✓
BabyAGI (Nakajima, 2023)	Autonomous	Vector Search	✓	✓	Prompting	Task Management	Text	GPT-4	-	-	-
AgentGPT (Reworkd, 2023)	Autonomous	Vector Search	✓	✓	Prompting	Task Management	Text	GPT-4	✓	✓	✓
Auto-UI (Zhang & Zhang, 2023)	Autonomous	Prompt	✓	-	Finetuning	UI control	Text Image	FLAN-Alpaca BLIP-2	-	-	-
AITW (Rawles et al., 2023)	Autonomous	Prompt	✓	-	Prompting	UI control	Text	PaLM 2	-	-	-
DCACQ (Mehta et al., 2023)	Autonomous	Prompt	✓	-	Finetuning	Engineer	Text	BART	-	-	-
ChemCrow (Bran et al., 2023)	Autonomous	Prompt	✓	-	Prompting	Chemistry	Text	GPT-4	✓	✓	✓
Chatmof (Kang & Kim, 2023)	Autonomous	Text Search	-	✓	Prompting	Material Sciences	Text	GPT-4	✓	✓	✓
IASSE (Boiko et al., 2023)	Autonomous	Vector Search	-	✓	Prompting	Scientific Experiments	Text	GPT-4	✓	✓	✓
TE (Aher et al., 2023)	Communicative	Prompt	✓	-	Prompting	Social Science	Text	GPT-4	-	-	-
CodePlan (Bairi et al., 2023)	Autonomous	Tree Search	✓	✓	Prompting	Coding	Text	GPT-4	-	✓	-
VIMA (Jiang et al., 2022)	Communicative	Prompt	✓	-	Finetuning	Robot Manipulation	Text Image	T5 ViT	-	-	-
React (Yao et al., 2022)	Communicative	Text Retrieval	✓	✓	Finetuning	Decision-Making	Text	PaLM-8B	✓	-	-
Reflexion (Shinn et al., 2023)	Communicative	Text Retrieval	✓	✓	Prompting	Decision-Making	Text	GPT-4	✓	-	-
ToRA (Gou et al., 2023b)	Autonomous	Prompt	✓	-	Prompting	Mathematical reasoning	Text	GPT-4	-	✓	-
Toolformer (Schick et al., 2023)	Autonomous	Text Retrieval	✓	✓	Finetuning	Mathematical reasoning	Text	GPT-J	✓	✓	✓
Fireact (Chen et al., 2023a)	Autonomous	Prompt	✓	-	Finetuning	Question Answering	Text	GPT-3.5	✓	-	-

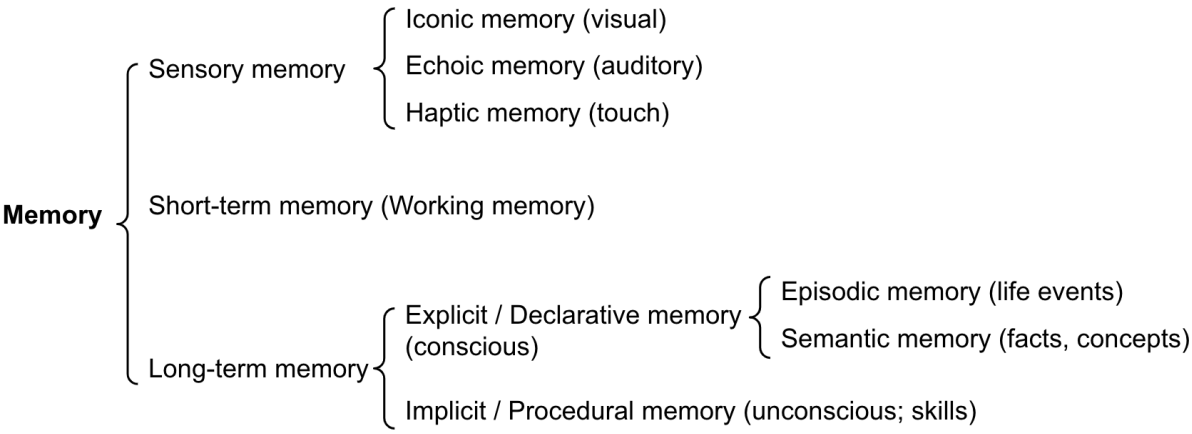
评估：

	AlfWorld		ScienceWorld		HotPotQA		FEVER	
Metrics	SR(%)	EX($)	AR	EX($)	SR(%)	EX($)	SR(%)	EX($)
Z-CoT	N/A	N/A	N/A	N/A	0.01	0.95	0.39	1.07
F-CoT	0.43	98.60	16.58	272.22	0.32	5.73	0.61	2.25
CoT-SC	0.57	105.37	15.24	274.33	0.33	7.86	0.62	3.21
SayCan	0.60	113.61	12.36	125.71	N/A	N/A	N/A	N/A
ReAct	0.57	152.18	15.05	356.03	0.34	66.00	0.63	22.20
Reflexion	0.71	220.17	19.39	724.48	0.39	112.49	0.68	37.26

- 随着开销的增加性能会提升。ReAct, Reflexion等工作涉及到多个计划、额外的思考和反思，所以执行时的开销更大但也带来了性能提升。

- 对复杂任务建议使用Fewshot例子。ZeroShot-CoT有推理能力，但是在两个QA任务上相比于few-shot CoT性能下降了很多。
  反思对于提高成功率很重要，特别是复杂任务而言。 Reflexioin相比ReAct消耗更多的token，但是在复杂任务上的性能也好很多。

关于Agent 的记忆能力



人脑记忆之于agent的记忆：

- 感觉记忆作为**原始输入的学习嵌入表示**，包括文本、图像或其他模态；
- 短期记忆作为**情境学习**。它很短而且有限，因为它受到 Transformer 有限的情境窗口长度的限制。
- 长期记忆作为代理在查询时可以关注的**外部向量存储**，可通过快速检索访问。

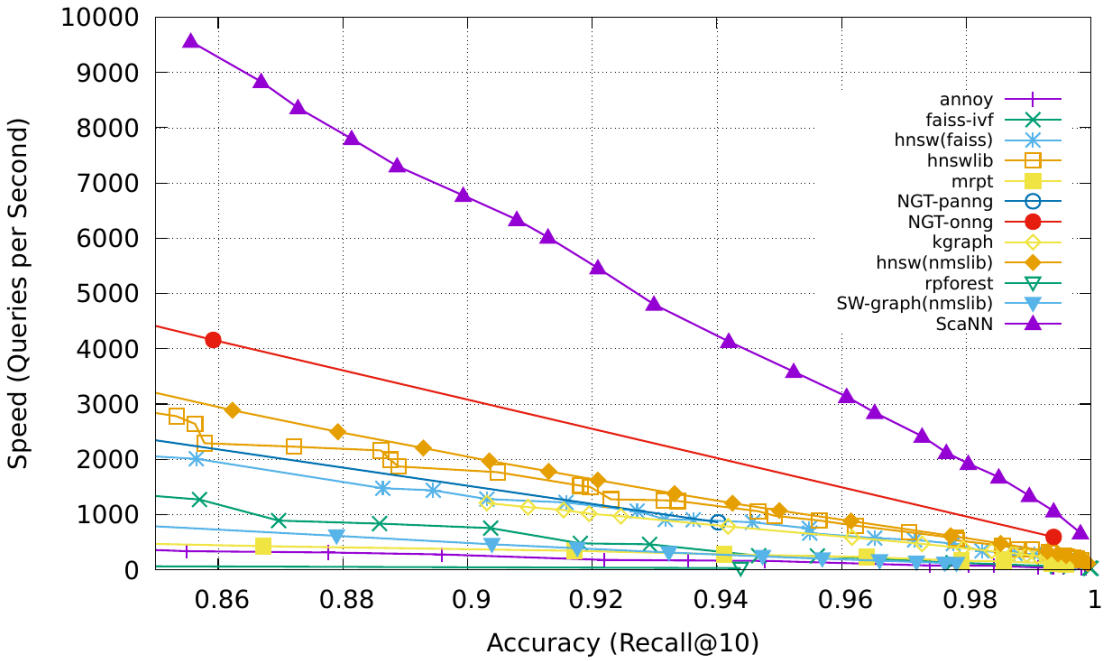
最大内积搜索(MIPS)

外部存储器可以缓解有限注意力范围的限制。标准做法是将信息的嵌入表示保存到可以支持快速最大内积搜索（MIPS）的向量存储数据库中。为了优化检索速度，常见的选择是*近似最近邻（ANN）*算法，以返回近似的前k个最近邻，以牺牲一点准确性换取巨大的加速。

对于快速 MIPS，有几种常见的 ANN 算法可供选择：

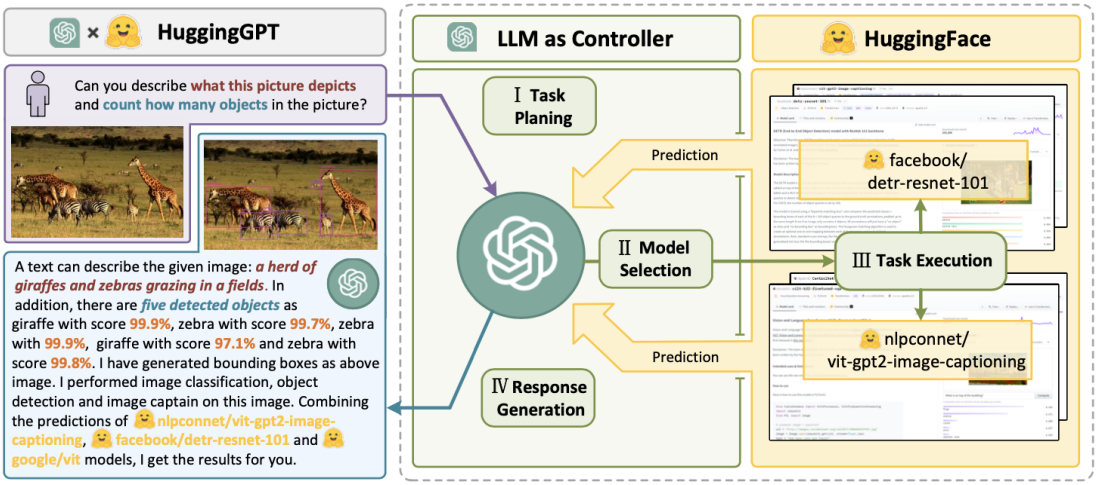
> - **LSH**（局部敏感哈希）：它引入了一种*哈希*函数，使得相似的输入项以高概率映射到相同的存储桶中，其中存储桶的数量远小于输入的数量。
> - **ANNOY**（近似最近邻）：核心数据结构是*随机投影树*，一组二叉树，其中每个非叶节点代表将输入空间分成两半的超平面，每个叶节点存储一个数据点。树是独立且随机构建的，因此在某种程度上，它模仿了哈希函数。ANNOY 搜索发生在所有树中，以迭代方式搜索最接近查询的一半，然后汇总结果。这个想法与 KD 树非常相关，但可扩展性更强。
> - **HNSW**（分层可导航小世界）：它的灵感来自于小世界网络的理念，在小世界网络中，任何其他节点都可以通过很少几步到达大多数节点；例如社交网络的“六度分离”特征。HNSW 构建了这些小世界图的分层层，其中底层包含实际数据点。中间的层创建快捷方式以加快搜索速度。执行搜索时，HNSW 从顶层的随机节点开始并导航至目标。当它无法再靠近时，它会向下移动到下一层，直到到达底层。上层的每一次移动都可能覆盖数据空间中的很大距离，而下层的每一次移动都会提高搜索质量。
> - **FAISS**（Facebook AI 相似性搜索）：它基于这样的假设：在高维空间中，节点之间的距离遵循高斯分布，因此应该*存在数据点的聚类*。FAISS 通过将向量空间划分为簇，然后在簇内细化量化来应用向量量化。搜索首先寻找具有粗量化的簇候选，然后进一步研究具有更精细量化的每个簇。

MIPS 算法比较:



关于Agent 使用工具能力

- **MRKL**[2022]是"模块化推理、知识和语言"的缩写，是一种用于自主代理的神经符号架构。MRKL 系统被提议包含一组"专家"模块，通用 LLM 充当路由器，将查询路由到最合适的专家模块。这些模块可以是神经的（例如深度学习模型）或符号的（例如数学计算器、货币转换器、天气 API）
- **TALM**（工具增强语言模型；2022）和**Toolformer**（2023）都对LLM 进行微调，以学习使用外部工具 API。
- **HuggingGPT**（2023）是一个框架，使用 ChatGPT 作为任务规划器，根据模型描述选择 HuggingFace 平台中可用的模型，并根据执行结果总结响应。



- **API-Bank**[2023]是评估工具增强型LLM性能的基准。

Agnet 应用

1. Single-Agent

- langchain，AutoGPT，GPT-Engineer 和 BabyAGI，HuggingGPT, Samantha

1. 面向任务的部署
    - 网络场景中:网络导航

        > **Mind2Web** 将多个针对 HTML 进行微调的 LLMs 结合在一起，使它们能够在真实世界的场景中总结冗长的 HTML 代码并提取有价值的信息。**WebGum** 通过使用包含 HTML 屏幕截图的多模态语料库，增强了具有视觉感知能力的Agent的能力。

    - 生活场景中：
        - Wu 等人介绍了 **PET 框架**，该框架通过早期纠错方法减少了环境信息中的无关物体和容器。PET 鼓励Agent更有效地探索场景和规划行动，并专注于当前的子任务。
2. 面向生命周期的部署
    - Minecraft: **Minecraft 中的Agent生存算法一般可分为两类：低级控制和高级规划。**Voyager 它引入了一个用于存储和检索复杂动作可执行代码的技能库，以及一个包含环境反馈和纠错的迭代提示机制。

2. Multi-Agent

多Agent系统（MAS）关注的重点是一组Agent如何有效地协调和协作解决问题。

- MetaGPT，AutoGen
- 斯坦福的虚拟小镇

1. **互补性合作交互**

- 无序合作:
    - ChatLLM
    - 在multi-Agent系统中引入一个专门的协调Agent，负责整合和组织所有Agent的响应，从而更新最终答案
    - 多数表决

- 有序合作:
  - CAME双Agent合作系统
  - AgentVerse
  - MetaGPT 从软件开发中的经典瀑布模型中汲取灵感，将Agent的输入/输出标准化为工程文档。通过将先进的人类流程管理经验编码到Agent提示中，多个Agent之间的合作变得更有条理。
  - **如果不制定相应的规则，多个Agent之间的频繁互动会无限放大轻微的幻觉。**

2. 对抗性互动促进进步

- 多Agent对抗系统
- 辩论
- ChatEval 建立了一个基于角色扮演的多Agent裁判团队。

3. 人类与Agent之间的互动参与

1. 不平等互动（即指导者-执行者范式）
2. 平等互动（即平等伙伴关系范式）