AutoAgents: A Framework for Automatic Agent Generation

Künstliche Intelligenz Seminar 2025

Prof. Dr. Schmidt-Schuß

Mahmoud Alhag Ali

Autoren:
Guangyao Chen, Siwei Dong,
Yu Shu, Ge Zhang,
Jaward Sesay, Börje Karlsson,
Jie Fu and Yemin Shi

Instituten
Peking University,
Hong Kong University of Science and Technology
Beijing Academy of Artificial Intelligence
University of Waterloo
{gy.chen, shiyemin}@pku.edu.cn, jiefu@ust.hk

Conferenc
Proceedings of the Thirty-Third International Joint Conference on
Artificial Intelligence (IJCAI-24), site (22 - 30)

Agenda

1. Definitionen
2. Vorarbeiten
3. Entwurfsphase
4. Agent-Architektur und Komponentendesign
5. Plan Erstellung
6. Ausführungsphase
7. Workflow
8. Experimentelle Validierung und Leistungsanalyse
9. Praktische Anwendung und Fallstudie

Einleitung

Die meisten bestehenden LLM-basierten Multi-Agent-Ansätze basieren auf vordefinierten Agenten zur Bewältigung einfacher Aufgaben, was die Anpassungsfähigkeit der Multi-Agent-Kollaboration an verschiedene Szenarien stark einschränkt. AutoAgents löst dieses fundamentale Problem durch die adaptive Generierung und Koordination mehrerer spezialisierter Agenten zur Bildung eines KI-Teams entsprechend unterschiedlicher Aufgaben

1. Definitionen

AI Agent:

Ein AI Agent (deutsch: KI-Agent) ist ein autonomes Softwaresystem, das mithilfe künstlicher Intelligenz in der Lage ist, eigenständig Aufgaben in einer definierten Umgebung auszuführen, um bestimmte Ziele zu erreichen.

LLM Agent

LLM-basierte autonome Agenten: LLMs werden häufig als Kernsteuerungen für autonome Agenten verwendet, die bestimmte Ziele erreichen können.

Multiagentensystem:

Ein Multiagentensystem (kurz: MAS) bezeichnet ein System, das aus mehreren autonomen Agenten besteht, die miteinander und mit ihrer Umgebung interagieren, um individuelle oder gemeinsame Ziele zu erreichen.

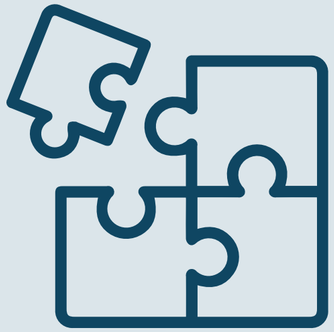
2.Vorarbeiten

Auto-GPT [Gravitas, 2023]: ist eine frühe Arbeit, die LLM als KI-Agent einsetzt, jedoch unterstützt es keine Multi-Agenten-Kolaboration und kann nur isoliert arbeiten.

SSP [Wang et al., 2023c]: hat die Aufgabenlösungsfähigkeiten von LLMs durch Integration von Multi-Agenten-Diskussionssystem. Es erlaubt LLMs für oder gegen eine Aufgabe zu argumentieren.

AgentVerse [Chen et al., 2023a]: Framework für die automatische Erzeugung unbegrenzter Agenten, es generiert den Ausführungsplan durch die Diskussionen der generierten Agenten und fügt Bewertungsstrategien für die zyklische Ausführung hinzu

3. Zweistufige Architektur



Entwurfsphase

- Synthetisiert ein angepasstes Agententeam und Ausführungsplan
- Planner, Agent Observer, Plan Observer

Ausführungsphase

- Verfeinerung den Plan und produziert das endgültige Ergebnis
- Action Observer

3.1 Entwurfsphase

synthetisiert ein angepasstes Agententeam und einen Ausführungsplan durch kollaborative Diskussion zwischen drei vordefinierten Agenten

Planner (P)

Generiert und verfeinert ein Agententeam sowie einen Ausführungsplan basierend auf dem Aufgabeninhalt

Agent Observer (O_agent)

Bietet Vorschläge zur Rationalität der Agententeammitglieder und deren Übereinstimmungsgrad mit der Aufgabe

Plan Observer (O_plan)

Liefert Vorschläge zur Rationalität des Ausführungsplans und dessen Übereinstimmungsgrad mit der Aufgabe und dem Agententeam

3.2 Ausführungsphase

Ausführungsphase

Die Ausführungsphase verfeinert den Plan durch Inter-Agent-Kollaboration und Feedback und produziert das endgültige Ergebnis

Action Observer (O_action)

fungiert als Aufgabenmanager für die verschiedenen Agenten, weist ihnen unterschiedliche Aufgaben zu, überprüft die Ausführungsergebnisse jedes Agenten und passt den Ausführungsplan dynamisch basierend auf dem Ausführungsstatus an

4. Agentenstruktur und Komponentendesign

Prompt (P)

- Bietet eine detaillierte und angepasste Darstellung der Expertenidentität für jeden spezifischen Agenten, bestehend aus Profil, Ziel und Beschränkungen

Description (D)

- Verleiht zusätzliche konkrete Identität zur Etablierung einer umfassenderen Rolle

Toolset (T)

- Stattet den Agenten mit ausgewählten Tools aus einem vordefinierten Set aus, um Entscheidungsverwirrung durch übermäßige Tools zu verhindern

Suggestions (S)

- Liefert Vorschläge für jeden Agenten zur Ausführung der aktuellen Aufgabe

4. Agent Observer

Planner (P)

- Auf der Grundlage der vom Planner wird Agentenliste {A1, A2, … , An} erstellt

Agent Observer (O_agent)

1. Prüft, ob jeder Agent den genannten Spezifikationen entspricht und identifiziert alle fehlenden Elemente {P, D, T}
2. bewertet die Kompatibilität jedes Agenten mit der Aufgabe anhand seiner Beschreibungsinformationen und des Aufgabeninhalts.
3. Prüft die Agentenliste auf überflüssige oder fehlende Rollen und eliminiert oder ergänzt sie entsprechend.

5. Plan Erstellung

Zeitgleiche Ausführungsplanung:

- Während Agenten für eine Aufgabe erstellt werden, entwickelt der Planer parallel einen Ausführungsplan zur Bewältigung der Aufgabe.

Planerstellung:

- Der Planer legt für die Aufgabe konkrete Einzelschritte {S1, S2, ..., Sn} fest.
- Jeder Schritt enthält:
  - Den zuständigen Agenten (A(j)),
  - die notwendigen Eingaben und
  - die erwarteten Ergebnisse.

Planprüfung durch den Plan Observer:

- Der Plan Observer kontrolliert, ob jedem Schritt ein klarer Agent zugeordnet ist und ob die Inhalte der Einzelschritte verständlich und konsistent sind.
- Außerdem prüft er, ob die Schrittfolge vollständig und geeignet ist, um die Aufgabe vollständig zu lösen.

5. Plan Erstellung

Iterative Verbesserung:

- Durch wiederholten Austausch zwischen Planer und Plan_Observer wird der Plan schrittweise optimiert.

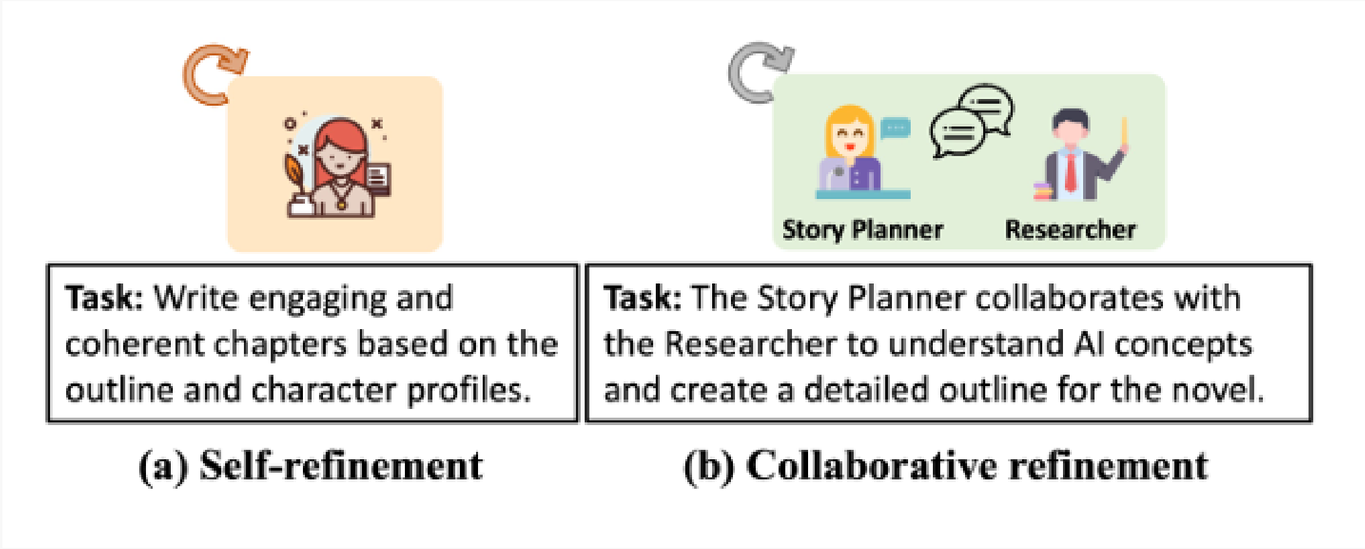
Abschluss:

- Nach mehreren Überprüfungsschleifen entsteht ein konsistenter, vollständiger Ausführungsplan zur Erreichung der vorgegebenen Aufgabe.

5.1 Aktionen zur Aufgabenausführung

Der Ausführungsplan umfasst zwei Aktionen der Aufgabenausführung:

- Selbstverfeinerung durch einen einzelnen Agenten
- Kollaborative Verfeinerung durch mehrere Agenten



6. Ausführungsphase

Action Observer O_action: fungiert als Aufgabenmanager für die verschiedenen Agenten, indem er ihnen verschiedene Aufgaben zuweist, die Ausführungsergebnisse der einzelnen Agenten überprüft und den Ausführungsplan auf der Grundlage des Ausführungsstatus dynamisch anpasst.

Dieser Mechanismus der Verfeinerung und Kommunikation wiederholt sich so lange, bis der Aktion_Observer eine einstimmige Einigung über die Ausführungsantworten erzielt oder der Prozess seine maximale Iteration erreicht.

6.1 Mechanismus zum Wissensaustausch.

Wenn jedoch die Anzahl der Agenten groß ist und ein einzelner Agent mehr Selbstiterationen durchführt, generiert er mehr historische Informationen. Aufgrund der Token-Beschränkung von LLM-Modellen können diese oft nicht alle Informationen umfassen. In diesem Rahmen gibt es also

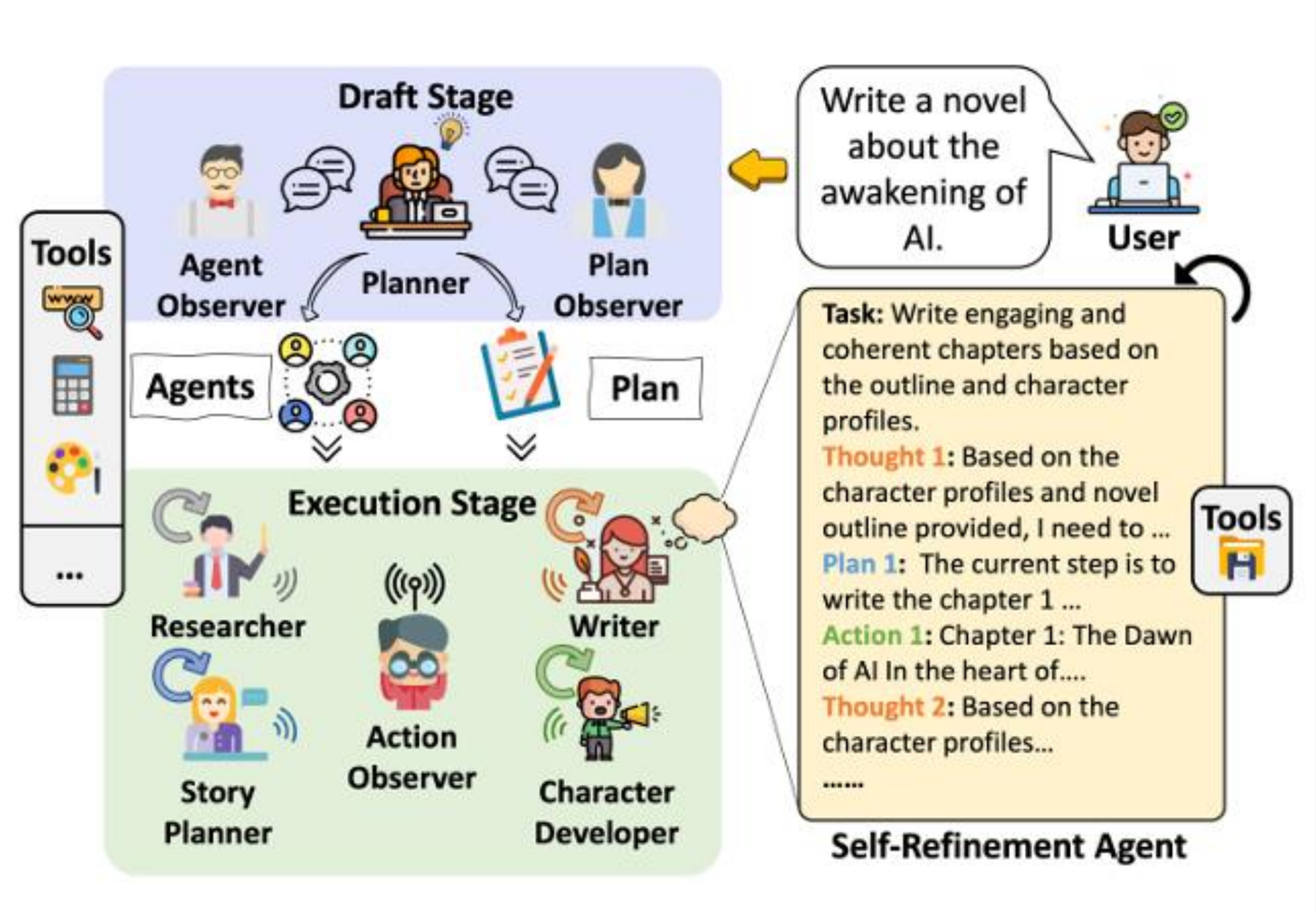
- ein Kurzzeitgedächtnis,
- ein Langzeitgedächtnis,
- ein dynamisches Gedächtnis.

6.1 Innovatives Speichermanagementsystem

AutoAgents implementiert ein dreischichtiges Speichersystem zur Bewältigung komplexer Kontextdaten

Speichertyp	Zweck	Inhalt
Kurzzeitgedächtnis	Einzelne Aktion	Zwischengedanken, Strategien und Ergebnisse während Self-/Collaborative Refinement
Langzeitgedächtnis	Historische Trajektorie	Ausgeführte Ergebnisse jeder Aufgabe und Synthese wichtiger Feedback-Informationen
Dynamisches Gedächtnis	Spezialisierte Aufmerksamkeit	Vom Action Observer extrahierte Hilfsinformationen für spezifische Aktionsanforderungen

7. Workflow



Algorithm 1 AutoAgents Execution Process.

Input: User task/Question
Output: Task solution/Answer

Drafting Stage
Initialize Planner $\mathcal{P}$, Agent Observer $\mathcal{O}_{\text{agent}}$, and Plan Observer $\mathcal{O}_{\text{plan}}$.
$\mathcal{P}$ generates initial agent team $\{\mathcal{A}_1, \mathcal{A}_2, \cdots, \mathcal{A}_n\}$ and execution plan $P = \{\mathcal{S}_1, \mathcal{S}_2, \cdots \mathcal{S}_n\}$.
repeat
  $\mathcal{O}_{\text{agent}}$ provides feedback on agent team.
  $\mathcal{P}$ refines agent team based on feedback.
  $\mathcal{O}_{\text{plan}}$ provides feedback on execution plan.
  P refines execution plan based on feedback.
until No feedback or reached the maximum iteration limit.
Execution Stage:
Initialize Action Observer $\mathcal{O}_{\text{action}}$ and long-term memory M_L.
for $\{\mathcal{S}_1, \mathcal{S}_2, \cdots \mathcal{S}_n\}$ **do**
  $\mathcal{O}_{\text{action}}$ generate dynamic memory M_D.
  $\mathcal{O}_{\text{action}}$ assign task $\mathcal{S}_k$ and M_D to corresponding agents $\{\mathcal{A}_i, \cdots, \mathcal{A}_j\}$.
  Initialize short-term memory M_S.
  repeat
    for $\{\mathcal{A}_i, \cdots, \mathcal{A}_j\}$ **do**
      Agent $\mathcal{A}_m$ analyzes $\mathcal{S}_k$, M_S and M_D.
      Agent $\mathcal{A}_m$ plans the current step and executes this step.
      The execution result is stored in M_S.
    end for
  until No step or reached the maximum iteration limit.
  The execution results of task $\mathcal{S}_k$ are stored in M_L.
  $\mathcal{O}_{\text{action}}$ coordinates $\{\mathcal{S}_1, \mathcal{S}_2, \cdots \mathcal{S}_n\}$ and monitors execution.
end for
return Execution results of final step.

8. Experimentelle Validierung und Leistungsanalyse

Open-ended Question Answering

AutoAgents wurde auf dem MT-bench-Datensatz mit 80 hochwertigen offenen Fragen aus verschiedenen Kategorien wie gesunder Menschenverstand, kontrafaktische Szenarien und Codierung getestet

Evaluator	v.s. **GPT-4**	v.s. **AgentVerse**
FairEval [Wang *et al.*, 2023b]	76.3%	91.3%
HumanEval	62.5%	77.5%

8. Experimentelle Validierung und Leistungsanalyse

Trivia Creative Writing

Diese Aufgabe fordert die Fähigkeiten großer Sprachmodelle heraus, vielfältige Informationen aus ihrem internen selbst komprimierten Wissen abzurufen und zu integrieren. Das Modell muss eine kohärente Geschichte um ein gegebenes Thema verfassen und dabei Antworten auf N Trivia-Fragen einbeziehen.

Methods	N (# trivia questions) = 5		N (# trivia questions) = 10	
	Score (%)	Δ (v.s Standard %)	Score (%)	Δ (v.s Standard %)
Standard	74.6	0.0%	77.0	0.0%
CoT [Yao *et al.*, 2023]	67.1	-10.0%	68.5	-11.1%
SPP-Profile [Wang *et al.*, 2023c]	79.1	+5.9%	83.0	+7.8%
SPP [Wang *et al.*, 2023c]	79.9	+7.1%	84.7	+10.0%
AutoAgents	**82.0**	**+9.9%**	**85.9**	**+11.6%**

8.1 Further Analysis

Ablationsstudie: Komponentenwichtigkeit

Eine detaillierte Ablationsstudie an 20 Instanzen der Trivia Creative Writing-Aufgabe offenbart die Bedeutung jeder Komponente

Methods	N (# trivia questions) = 5	
	Score (%)	Δ (v.s Standard %)
Standard	74.6	0.0%
CoT [Yao *et al.*, 2023]	66.0	-11.5%
SPP-Profile [Wang *et al.*, 2023c]	74.0	-0.01%
SPP [Wang *et al.*, 2023c]	84.4	+13.1%
AutoAgents w/o observers	87.0	**+16.6%**
AutoAgents w/o self-refinement	87.0	**+16.6%**
AutoAgents w/o dynamic memory	89.0	**+19.3%**
AutoAgents	**90.0**	**+20.6%**

8.2 Vergleich mit bestehenden Frameworks

AutoAgents unterscheidet sich fundamental von bestehenden Multi-Agent-Frameworks durch mehrere Schlüsselinnovationen

Framework	Dynamic Agent Generation Method	Number of Agent	Multi-agent Conversation	Self-Refinement Action	Collaborative Refinement Action
AutoGPT [Gravitas, 2023]	✗	1	✗	✓	✗
BabyAGI [Nakajima, 2023]	✗	3	✓	✗	✗
Generative Agents [Park *et al.*, 2023]	✗	25	✓	✓	✗
Camel [Li *et al.*, 2023]	✗	2	✓	✗	✗
GPT-bargaining [Fu *et al.*, 2023]	✗	3	✓	✓	✗
MetaGPT [Hong *et al.*, 2023]	✗	Unlimited	✓	✗	✗
AutoGen [Wu *et al.*, 2023]	✗	Unlimited	✓	✗	✗
Social Simulacra [Park *et al.*, 2022]	Single Agent	Unlimited	✓	✗	✗
Epidemic Modeling [Williams *et al.*, 2023]	Single Agent	Unlimited	✓	✗	✗
ExpertPrompting [Xu *et al.*, 2023]	Single Agent	1	✗	✗	✗
SSP [Wang *et al.*, 2023c]	Single Agent	Unlimited	✓	✗	✗
AgentVerse [Chen *et al.*, 2023a]	Single Agent	Unlimited	✓	✗	✗
AutoAgents	Multi-agent Discussion	Unlimited	✓	✓	✓

9. Praktische Anwendungen und Fallstudien

Software-Entwicklung

Eine Fallstudie zur Entwicklung eines Python-basierten Tetris-Spiels zeigt die Überlegenheit kollaborativer Diskussion in der Entwurfsphase

- Ohne Observer: Der Planner generiert ausschließlich Programmierer und vernachlässigt den ganzheitlichen Prozess
- Mit Observer: Einbeziehung von Spieldesign-Experten, UI-Design-Experten und Test-Experten

Conclusion

AutoAgents präsentiert ein innovatives Framework für dynamische Multi-Agenten-Systeme auf Basis großer Sprachmodelle. Es kombiniert flexible Agentengenerierung, kollaborative Planung und effektive Beobachtermechanismen, um komplexe Aufgaben effizient zu lösen. Die Ergebnisse zeigen klare Vorteile gegenüber bisherigen Ansätzen und machen AutoAgents zu einem wichtigen Schritt in Richtung leistungsfähiger, adaptiver KI-Systeme.

Danke für Ihre Aufmerksamkeit