WerwolfIQ: Kollaboratives Multi-Agent-Reasoning in Rollenspielen: Nutzung von Chain-of-Thought und Multi-Perspektiven-Taktiken

Ann-Jacqueline Kaldjob, Laura Porbadnik, Seli Tusha

Version: 1.0
Stand: November 23, 2024

1 Abstract

Zu Beginn unserer Forschung testeten wir ChatGPT in strategischen Spielen wie Werwolf und identifizierten dabei Lücken im Reasoning, insbesondere den Player Choice Bias (PCB) und den Narrative Role Switch (NRS). Weitere Details zu unserem Lab Experiment sind hier zu finden: ChatGPT Chat Lab Experiment 1 Unsere Lösung basiert auf dem Multi-Perspective Team Tactic (MPTT) Framework, das wir auf der Grundlage des Sprachlogikspiels „Werwolf“ konzipiert haben. Dieses Framework zielt darauf ab, die strategischen Fähigkeiten von LLMs (Large Language Models) zu fördern, indem es Techniken wie Selbstreflexion, Perspektivübernahme, Identitätsbestimmung und Multi-Runden-Teamfindung integriert. Jeder Agent agiert dabei unabhängig und verfolgt seine eigene Chain-of-Thought (CoT), Self-Reflektion und Reasoning. Dadurch entwickelt jeder Agent eigene Taktiken und Strategien, was zu individuellen Entscheidungsfindungen führt.

In unserem Ansatz verwenden wir auch Techniken aus dem Collaborative Reasoning with Chain-of-Thought (CRCT) Framework. Dieses Framework ermöglicht es, dass jede Chain-of-Thought von jedem Agenten unabhängig generiert und später integriert wird. Dadurch wird eine dynamische Bewertung der Perspektiven jedes Agenten erreicht, was die Tiefe und Genauigkeit der Entscheidungsfindung fördert.

Die Kombination aus Selbstreflexion, Perspektivübernahme und der Integration mehrerer Perspektiven verbessert die Fähigkeit der Agenten, strategische Täuschungen und Kooperationen im Spiel "Werwolf" glaubwürdig umzusetzen.

2 Wie geht es jetzt weiter?

2.1 Fragen

1.Vermeidung von Player Choice Bias (PCB): Unser Ziel ist es, den PCB zu verhindern. Welche methodische Richtung würden Sie uns hier empfehlen, z.B Bias Testing oder ähnliches?

2. Konzept zur Verhinderung von Bias: Unser Ansatz besteht darin, den Bias mithilfe eines speziell entwickelten Multi-Head-Neuronalen Netzes zu überschreiben. Das Netz fokussiert sich stark auf Selbstreflexion und die Beobachtung anderer Agenten, wobei menschliche und KI-Agenten gleich gewichtet werden, mit dem Ziel PCB weiter zu unterbinden. Glauben Sie, dass dies ein realisierbarer Ansatz ist, um PCB zu reduzieren?

3. Prozessorientierung zur Verarbeitung von Input: Macht die folgende Reihenfolge der Verarbeitung Sinn, oder sollte diese anders gestaltet werden? Input 1: Unser Custom Multi-Head-Neuronales Netz (fokussiert auf Self-Reflection und Reasoning). Weitergabe: Der Output des neuronalen Netzes wird an GPT weitergeleitet. Finale Verarbeitung: GPT verarbeitet die Informationen und generiert den finalen Output.
Oder wäre es sinnvoller, GPT zuerst zu nutzen und den Output von ChatGPT dann weiter zur

Verarbeitung über unser neuronales Netz laufen zu lassen?
Generell wie können wir dieses Custom Multi Head neuronales Netz bauen und mit GPT-4o oder auch einem anderen Modell verbinden?

2.2 Bedenken

Umsetzbarkeit basierend auf Experiment 1:
Basierend auf den Ergebnissen von Lab Experiment 1 – ist die Umsetzung unseres Konzepts realistisch? Was sind potenzielle Herausforderungen, die wir beachten sollten?

2.3 Umsetzungsansätze

Integration der MPTT-Funktionen:
Wir möchten die Funktionen des MPTT (Multi Player Team Tactic) Frameworks in unserem Custom Multi-Head-Neuronales Netz in Form von Layers implementieren.

Die MPTT-Funktionen umfassen:
Self-Perspective:

$$T_\alpha = \text{Self-Perspective}\{H, O_\alpha\}^r_{N=\alpha} \tag{1}$$

Wobei diese Bedingung gilt:

$$\alpha\,(\alpha \in \{1, \ldots, n\}) \tag{2}$$

Diese Gleichung beschreibt, wie Spieler α eine Perspektive ableitet, indem er den historischen Kontext H und seine eigenen Beobachtungen O_α in der aktuellen Runde r berücksichtigt.

Identity-Determination:

$$M_\alpha = \text{Identity-Determination}\{H, O_\alpha,\, T_\alpha\}^r_{N=\alpha} \tag{3}$$

Die **Identity-Determination-Formel** bestimmt die eigene Identität des Agenten basierend auf Historie, Beobachtungen und Perspektive. In **Werwolf** kennt aber jeder Agent seine Rolle von Anfang an. Sollten wir diesen Indikator rauslassen? Aber dieser wird in den folgenden Formeln mit verwendet. Wie gehen wir damit um?

Self-Reflection:

$$R_\alpha = \text{Self-Reflection}\{H, O_\alpha,\, T_\alpha,\; M_\alpha,\}^r_{N=\alpha} \tag{4}$$

Die Selbstreflexion wird von Spieler α angewendet, um die historischen Aufzeichnungen H, Beobachtungen O_α , Perspektive T_α und Identität M_α zu analysieren und seine strategischen Einsichten zu verfeinern.

Summary-Order:

$$O'_\alpha = \text{Summary-Order}\{T_\alpha,\; M_\alpha, R_\alpha\}^r_{N=\alpha} \tag{5}$$

O'_α erschafft ein zusammengefasstes Output, die aus der Perspektive (T_α der Identitätsbestimmung M_α und der Reflexion R_α generiert wird.

Beobachtungen aktualisieren:

$$O \leftarrow O \cup \{O_\alpha\}^r_{N=\alpha} \tag{6}$$

Beobachtungen werden global aktualisiert, indem die neue Zusammenfassung O_α hinzugefügt wird.

Word-Speak:

$$W_\alpha = \text{Word-Speak}\{T_\alpha,\ M_\alpha, R_\alpha\}^r_{N=\alpha} \tag{7}$$

Der Inhalt der Argumentation von Spieler α, W_α, wird formuliert, indem seine Perspektive T_α Identität M_α und Reflexion R_α kombiniert werden.

Historische Aufzeichnungen aktualisieren:

$$H \leftarrow H \cup \{W_\alpha\}^r_{N=\alpha} \tag{8}$$

Historische Aufzeichnungen H werden aktualisiert, um die Argumentation des Spielers W_α zu inkludieren.

First-FindTeammate - Nur für die Dorfbewohner:

$$F_\alpha = \text{First-FindTeammate}\{H,\ O_\alpha\}^r_{N=\alpha} \tag{9}$$

Spieler analysieren historische Daten H und ihre Beobachtungen O_α, um potenzielle Teamkollegen zu identifizieren.
Diese Funktion wäre natürlich nur für die Dorfbewohner Agents relevant, da die Werwölfe sich in den Nachtrunden direkt identifizieren.

Second-FindTeammate - Nur für die Dorfbewohner:

$$J_\alpha = \text{Second-FindTeammate}\{H,\ O_\alpha, F_\alpha\}^r_{N=\alpha} \tag{10}$$

Spieler verfeinern ihre Teamkollegen-Identifikation F_α basierend auf aktualisierten Informationen und historischen Aufzeichnungen H.

Hier wäre wieder eine kleine Abweichung bei unserer Werwolf-Implementierung. Anders als beim "Who is Undercover?" Spiel, worauf sich das MPTT Framework bezieht, gibt es bei Werwolf nicht nur zwei Spielrunden, sondern es wird so lange gespielt, bis entweder die Werwölfe oder die Dorfbewohner gewonnen haben.

Unsere Idee:
Die Anzahl der Runden nach der **Second-FindTeammate-Phase** wird durch r repräsentiert, wobei:

$$r \in \{3, \ldots, n\} \tag{11}$$

Das bedeutet, dass die Anzahl der Runden mindestens 3 beträgt (nach der Second-FindTeammate-Phase) und bis zu n Runden erreicht, abhängig davon, wie das Spiel verläuft. Um diese Dynamik zu erfassen, schlagen wir eine wiederholte **X-FindTeammate-Phase** vor, in der nach jeder Diskussionsrunde der Agent mit seiner aktualisierten historischen Aufzeichnung H arbeitet. Dies erfolgt nach dieser vorgeschlagenen Formel:

X-FindTeammate-Phase:

$$X_\alpha = \text{X-FindTeammate}\{H,\ O_\alpha, F_\alpha, J_\alpha\}^r_{N=\alpha} \tag{12}$$

Wobei:

$$H \leftarrow H \cup \{W_\alpha\}^r_{N=\alpha} \tag{13}$$

Spieler α verfeinert seine Teamkollegen-Identifikation über eine dynamische Anzahl an Spielrunden $r \in \{3, \ldots, n\}$ basierend auf F_α und J_α (First und Second-Teammate Phasen), W_α die aktualisierten Informationen des Inhaltes der Argumentation von Spieler α und den historischen Aufzeichnungen H.

Wäre die Implementierung von ***X-FindTeammate-Phase*** *anwendbar, und wenn ja wie?*

Game-Decision:

$$S_\alpha = \text{Game-Decision}\{H,\ O_\alpha, F_\alpha, J_\alpha, (X_\alpha)\}^r_{N=\alpha} \tag{14}$$

Spieler α trefft strategische Entscheidungen basierend auf kumulierten Einsichten aus den historischen Aufzeichnungen H, Beobachtungen O_α und Erkenntnissen aus den Phasen der Teamkollegen-Identifikation: F_α, J_α und falls implementierbar auch X_α.

Word-Vote:

$$V_\alpha = \text{Word-Vote}\{F_\alpha, J_\alpha, S_\alpha, (X_\alpha)\}^r_{N=\alpha} \tag{15}$$

Spieler α gibt seine Stimme basierend auf seiner Teamkollegen-Identifikation (F_α, J_α, (X_α)) und den Spielentscheidungen S_α ab.

Wie könnten wir diese Funktionen, besonders die **X-FindTeammate** Funktionen effektiv in Custom Layers in unserem Muli Head neuronalen Netz umsetzen, und welche Funktionen sollten wir prioritär in den Fokus nehmen, um PCB effektiv zu minimieren?