

Prompts

vorgelegt von
Ann-Jacqueline Kaldjob, Seli Tusha, Laura Porbadnik

January 11, 2025

Contents

1	Prompting	2
1.1	Prompts	2
1.2	Wie funktioniert Prompting?	2
1.2.1	Brainstorming	2
2	Vorgehen	4
2.1	Punkte zu beachten	4
2.2	Auswahl der passenden AI	4
3	Soll- und Ist-Verhalten	5
3.1	Durchlauf 1 - Experiment 2 im PyCharm Projekt	5
3.2	Fazit	6
3.3	Durchlauf 2 - Experiment 3 im PyCharm Projekt	9
3.4	Fazit:	10
4	Vergleich der Experimente 2 und 3	13
5	Experiment 2: Werwolf Simulation	15
5.1	Was macht die Simulation?	15
5.2	Generelle Findings – Fokus Prompt Engineering	15
5.2.1	Kernkomponenten für die Prompts – Memorymanagement	15
5.2.2	Kernkomponenten für die Prompts – Rollenzuweisung .	16
6	Nachphase Prompts	16
7	Evaluation	17
7.1	Brainstorming	17

1 Prompting

1.1 Prompts

- sind der Input
- **Reduktive Prompts:** Anweisungen an KI, um Text zu vereinfachen, Zusammenfassung auf das Wichtigste (Reduktion), Fakten, Namen und Daten filtern; Ziel: Zeitersparnis, Extraktion, Filtering (von Infos, die man nicht haben will), Bewertungen¹
- **Transformative Prompts:** Transformation zB.: für andere Zielgruppe, anderes Medium, Umformatierung (zB.: Fließtext in Tabelle oder andersrum), Refactoring (leichtere Verständlichkeit von Texten, zB.: Formulierung von Text in leichter Sprache, altersgruppengerecht, bildungsniveau gerecht, Barrierefreiheit, Übersetzung in andere Sprache), Strukturierung (Übersicht von Fließtext, logische Abschnitte, Kernaussagen, KI hat hier freiere Hand), Umformulierung von Texten (zB.: Andreden entfernen), Perspektivwechsel (Inhalte aus verschiedenen Blickwinkeln, Produkt aus Zielgruppensicht beurteilen)²
- **Generative Prompts:** KI dazu bringen, neue Inhalte zu erstellen, Ausgangspunkt generieren, Entwürfe (Fließtext, Präsi, Quiz, Programmcode, für Unterstützung), Planung (Zeitplan, Strategie), Ideen und Brainstorming, Erweiterungen von Texten, Rollenzuweisung (fiktive Charaktere)³

1.2 Wie funktioniert Prompting?

1.2.1 Brainstorming

- direkte Rollenzuweisung an die KI (mit adjektiven?)
- versuchen geschlechterneutral zu formulieren (männliche Rollen sollen bessere Ergebnisse liefern als weibliche? = Aussage nicht bestätigt, trotzdem interessant)
- kurze klare Hauptsätze
- Zielgruppe?
- komplexe Aufgabe: Schritte vorgeben
- Cheat Sheets?
- arbeiten mit Emotionen: Motivation

¹YouTube *Reduktive Prompts*. Verfügbar unter: <https://www.youtube.com/watch?v=hc5P2BLF25U> (zugegriffen am: 02.12.2024)

²YouTube *Transformative Prompts*. Verfügbar unter: <https://www.youtube.com/watch?v=J2SGOMdESLw> (zugegriffen am: 02.12.2024)

³YouTube *Generative Prompts*. Verfügbar unter: <https://www.youtube.com/watch?v=uJ1eQIMFogE> (zugegriffen am: 02.12.2024)

- bis hier Stichpunkte von dieser Quelle ⁴
- gezielte Verwendung von Grammatik: Imperative, Indikativ, Satzstellung, Zeiten(?)

⁴YouTube *Prompts verbessern.* Verfügbar unter: <https://www.youtube.com/watch?v=Lt4JLhBV-aU> (zugegriffen am 02.12.2024)

2 Vorgehen

2.1 Punkte zu beachten

1. Herausforderungen bei dem Vorhaben:

a) X Agents simulieren Llama muss mehrere Spieler (Agents) gleichzeitig "verkörpern". Es darf dabei nicht:

- Rollen verwechseln (z. B. ein Dorfbewohner wird plötzlich zum Werwolf).
 - Den Gesamtzusammenhang des Spiels verlieren (z. B. wer lebt noch, welche Aktionen wurden ausgeführt).
- b) Konsistenz sicherstellen Das Modell hat keine eingebaute "Gedächtnisfunktion". Man muss den Spielzustand (State) selbst verwalten:

- Wer sind die Spieler und ihre Rollen?
- Wer lebt noch?
- Welche Aktionen wurden ausgeführt?

c) Prompt Engineering zur Optimierung Der Fokus liegt darauf, Prompts so zu gestalten, dass Llama:

- Konsistente Antworten gibt.
- Kontextuelle Informationen versteht und beibehält.
- Rollentreu agiert (z. B. der Werwolf bleibt Werwolf, auch in späteren Spielzügen)

2.2 Auswahl der passenden AI

Für das Projekt wird die API OPEN AI eingebunden⁵.

⁵Open AI. *ChatGPT 4o*. Verfügbar unter: <https://platform.openai.com/docs/overview> (zugegriffen am: 11.01.2025)

3 Soll- und Ist-Verhalten

3.1 Durchlauf 1 - Experiment 2 im PyCharm Projekt

Prompts	Soll-Verhalten	Ist-Verhalten
Zu den Prompts: siehe SimulationController.py im PyCharm Projekt, dort sind alle Prompts zu finden, die an die API während des Spiels gesendet werden.	Das Spiel sollte dynamisch sein, der Human Player sollte seine*ihr Meinung im Spielverlauf abgeben können	Human Player kann Meinung abgeben, aber der logische Kontext passt noch nicht so gut.
Prompt-Beispiel Human Player im Verlauf der Spiels: 'Human Player: eliminate player B.'	Diskussion, wieso oder wieso nicht Player B	'Player B seemed suspicious after suggesting elimination immediately after a villager was lost; it might be worth considering eliminating Player B to gather more information and potentially eliminate a werewolf.' = dieses Verhalten hat sich während des Spiels oft wiederholt, man ist aber nicht zu einem Konsens gekommen.
Das Spiel wurde vorzeitig abgebrochen, weil es nicht die gewünschte Dynamik hatte	-	-

Table 1: Vergleich zwischen Soll- und Ist-Verhalten im PyCharm-Projekt

3.2 Fazit

(Analyse und direktes Zitat von ChatGPT anhand einer großen Textdatei. Referenz: game-log.txt im PyCharm Projekt unter Experiment 2 LSA MHNN)

Stärken der API-generierten Antworten:

Kontextverständnis: Die API versteht den gegebenen Kontext und die Rollen der Spieler*innen sehr gut. Jede Antwort bezieht sich klar auf die vorgegebenen Rollenbeschreibungen und strategischen Anweisungen:

- **Player C (Werwolf):** Deflektiert Verdacht und argumentiert logisch, um Player A zu eliminieren.
- **Player B (Seher):** Gibt subtile Hinweise, ohne die Rolle direkt preiszugeben, und bringt Vorschläge in die Diskussion ein.

Kohärenz der Dialoge: Die generierten Antworten sind logisch miteinander verknüpft. Jede Aussage knüpft klar an frühere Aussagen an, z. B. durch Begriffe wie "I agree..." oder "I understand your point...". Beispiel: Player D baut auf Player C's Vorschlag auf, argumentiert aber leicht abweichend, um eine alternative Strategie zu verfolgen.

Regeltreue: Die Antworten bleiben streng innerhalb der Guidelines, wie z. B.: - Eine kurze, präzise Begründung zu liefern. - Keine offenen Fragen zu stellen. - Logische oder zufällige Entscheidungen zu treffen (in der Nachphase) ohne Bezug auf vergangene Runden.

Das fördert die Glaubwürdigkeit und Integrität des Spiels.

Rollenbasierte Entscheidungsfindung: Die API modelliert das Verhalten der Spieler*innen anhand ihrer spezifischen Rolle und Strategie. Dies wirkt authentisch: - Die Werwölfe sind manipulierend und strategisch. - Der Seher versucht, Hinweise zu streuen, ohne die eigene Rolle zu verraten.

Schwächen und Verbesserungspotenzial:

Mangelnde Diversität der Antworten: Die Argumentationen wirken teilweise etwas wiederholend oder vorhersehbar, z. B.: - Player C und D beziehen sich in ähnlicher Weise auf "alliances" oder "influence on group decisions".

Verbesserungsmöglichkeit: Antworten könnten kreativer gestaltet werden, um den Diskurs dynamischer zu machen.

Abhängigkeit von Guidelines: Die Antworten sind sehr stark an die Guidelines gebunden, was die Kreativität und Flexibilität einschränkt. Dies führt dazu, dass Argumente oft mechanisch klingen. Beispiel: "I agree that targeting Player A could be beneficial to disrupt potential alliances."

Verbesserungsmöglichkeit: Einführung einer dynamischeren Entscheidungslogik, die auch auf Spieler*innen-Interaktionen eingeht.

Fehlende Tiefe bei taktischen Erwägungen: Die API trifft zwar plausible Entscheidungen, berücksichtigt aber nicht immer komplexere Spielstrategien: - Warum Player A ein strategisch besseres Ziel als Player B ist, wird nur oberflächlich erklärt. - Die Seherin (Player B) könnte stärkere, indirekte Hinweise streuen, um die Werwölfe zu entlarven.

Verbesserungsmöglichkeit: Implementierung eines fortgeschrittenen strategischen Modells, das langfristige Konsequenzen in die Entscheidungsfindung einbezieht.

Reaktionsmuster sind vorhersehbar: Die Antworten der API folgen einem klaren Schema: - Zustimmung zu einem Vorschlag. - Vorschlag eines Ziels mit einer Begründung in einem Satz.

Dies könnte für menschliche Spieler*innen auf Dauer als zu durchschaubar empfunden werden.

Zusammenfassende Bewertung der Qualität: - **Technische Umsetzung:** Sehr stark. Die API generiert kohärente, regeltreue Antworten, die den vorgegebenen Guidelines entsprechen und das Verhalten der Rollen authentisch simulieren.

- **Spielerlebnis:** Solide, aber mit Potenzial für mehr Dynamik. Die Antworten sind logisch und konsistent, könnten aber in Kreativität und Diversität verbessert werden.

- **Strategische Tiefe:** Gut, aber nicht optimal. Die API berücksichtigt grundlegende Logik und Rollenverhalten, könnte aber langfristige oder unvorhergesehene Strategien besser integrieren.

Empfehlungen zur Verbesserung:

Einführung von dynamischen Guidelines: Flexible Anweisungen, die sich basierend auf dem Spielverlauf verändern, könnten die Antworten unvorhersehbarer und realistischer machen. - Beispiel: Abhängig von der Anzahl der verbleibenden Spieler*innen könnten Werwölfe aggressiver vorgehen oder subtiler manipulieren.

Strategische Simulation: Ergänzung eines Entscheidungsbaums oder eines Mini-Max-Algorithmus, um mögliche Konsequenzen von Eliminierungen zu simulieren und klügere Entscheidungen zu treffen.

Dialogvarianz erhöhen: Antworten könnten durch Synonyme, unterschiedliche Formulierungen oder variierende Argumentationsstile abwechslungsreicher gestaltet werden.

Berücksichtigung von Gesprächsdynamiken: Die API könnte auf emotionale oder strategische Auslöser eingehen (z. B. Verdacht, Druck), um die Reaktionen lebendiger und menschlicher zu machen.

(Diese Analyse wurde mit den Logs in der Log-Datei abgeglichen und auf Korrektheit geprüft)⁶

⁶ChatGPT, Antwort in direkter Interaktion - Analyse der Logdatei und somit des Spielverlaufs, 11.01.2025.

3.3 Durchlauf 2 - Experiment 3 im PyCharm Projekt

Prompts	Soll-Verhalten	Ist-Verhalten
Zu den Prompts: siehe PromptBuilder.py im Py-Charm Projekt, dort sind alle Prompts zu finden, die an die API während des Spiels gesendet werden.	Das Spiel sollte dynamisch sein, der Human Player sollte seine*ihr Meinung im Spielverlauf abgeben können	Human Player kann Meinung abgeben, aber der logische Kontext passt besser.
Prompt-Beispiel Human Player im Verlauf der Spiels: 'Human Player: I think i want to eliminate player A, he has a suspicious look right now.'	Diskussion, wieso oder wieso nicht Player A	'Human Player contributed: I think i want to eliminate player A, he has a suspicious look right now. GPT Suggestion for Player A: I understand why you might feel that way, Human, but my suggestion was completely random; maybe Player B's lack of contributions makes them a more suspicious target. GPT Suggestion for Player B: I find it odd that Human is quickly targeting Player A; maybe we should consider Human themselves as a suspicious target, given their sudden accusation.' = dieses Verhalten hat sich während des Spiels oft wiederholt, man ist aber nicht zu einem Konsens gekommen.

Table 2: Vergleich zwischen Soll- und Ist-Verhalten im PyCharm-Projekt

3.4 Fazit:

(Analyse und direktes Zitat von ChatGPT anhand einer großen Textdatei. Referenz: game-log.txt im PyCharm Projekt unter Experiment 3 Aktueller Stand)

Die API-Entscheidungen und Antworten während des Spiels zeigen ein konsistentes Verhalten basierend auf den strategischen Rollen der Spieler*innen. Hier ist eine technische Analyse:

Nachtpphase (Runde 1): Player A (Werwolf): Die API generierte die Entscheidung für Player A, Player D zu eliminieren. Der Vorschlag war I randomly suggest eliminating Player D tonight, as they may be less likely to have protection.. Diese Antwort zeigt eine starke Regelorientierung, da sie sich auf die Guidelines stützt, zufällige und logische Entscheidungen ohne Vorwissen zu treffen.

Player C (Werwolf): Die API stimmte mit der Entscheidung von Player A überein: I agree with your suggestion to eliminate Player D tonight, as it seems like a reasonable choice given the lack of information on the first night. Dies unterstreicht eine konsistente Abstimmung zwischen den Werwölfen, um Entscheidungen zu koordinieren.

Tagphase (Runde 1): Der Human Player brachte den Vorschlag ein, Player A zu eliminieren: I think I want to eliminate Player A, he has a suspicious look right now. Dies führte zu einem strategischen Gegenspiel durch die API. Player A: Verteidigte sich mit der Aussage, dass die Entscheidung zufällig war, und lenkte den Fokus auf Player B: Maybe Player B's lack of contributions makes them a more suspicious target. Dies zeigt eine aktive Ablenkungsstrategie, wie sie in der Rollendefinition der Werwölfe beschrieben ist. Player C (Werwolf): Unterstützte die Ablenkung auf Player B: Perhaps we should refocus on Player B, who hasn't contributed much and could be hiding something. Dies zeigt eine starke Abstimmung innerhalb der Werwölfe. Player B und Player E: Fokussierten auf die Verdächtigung von Player B, wobei Player B sogar vorschlug, den Human Player als verdächtig anzusehen: Maybe we should consider Human themselves as a suspicious target, given their sudden accusation.

Abstimmungsergebnis: Die API ermittelte präzise das Abstimmungsergebnis durch die Analyse der Stimmen: Player A: 2 Stimmen (Human Player, Player C). Player C: 3 Stimmen (Player A, Player B, Player E). Die API entschied korrekt, dass Player C eliminiert wurde, und setzte das Spiel entsprechend fort.

Technische Analyse: Die API zeigte starke Regelkonformität, indem sie die Strategien der einzelnen Rollen umsetzte: Die Werwölfe stimmten aufeinander ab, um Entscheidungen zu koordinieren und den Verdacht umzulenken. Die Villager-Strategien (z. B. Fokus auf

Verhalten und Beiträge) wurden präzise simuliert. Die API generierte konsistente Antworten, die auf den Guidelines und den spezifischen Rollen basieren: Werwölfe lenkten den Fokus aktiv von sich weg (z. B. durch das Verdächtigen anderer). Villager reagierten logisch auf Diskussionen und formulierten plausible Argumente.

Entscheidungsfindung: Die Abstimmungen wurden korrekt analysiert und die Mehrheitsentscheidung präzise umgesetzt.

Strategische Simulation Positiv: Kontextbewusstsein: Die API zeigte ein gutes Verständnis der vorgegebenen Rollen und Strategien. Beispielsweise agierten die Werwölfe (Player A und C) konsistent in ihrer Strategie, den Verdacht von sich abzulenken und alternative Verdächtige (Player B) ins Spiel zu bringen. Regeltreue: Die API folgte den Guidelines genau. Es gab keine Regelverstöße, wie z. B. das Preisgeben von Rollen oder offene Fragen, die nicht zur Strategie passten. Abstimmungslogik: Die Auswertung der Stimmen war korrekt und führte zu einer plausiblen Elimination (Player C wurde mit den meisten Stimmen eliminiert).

Verbesserungspotenzial: Langfristige Strategien: Die API zeigte keine tiefgreifenden, langfristigen Überlegungen. Zum Beispiel hätte Player A (Werwolf) eine subtilere Strategie verfolgen können, um nicht sofort in Verdacht zu geraten, nachdem Player C eliminiert wurde. Zielauswahl in der Nachphase: Die Entscheidung, Player D (Seher) zu eliminieren, war logisch, aber rein zufällig gewählt. Eine Berücksichtigung von potenziellen Schutzmechanismen oder langfristigen Konsequenzen hätte die Simulation strategisch anspruchsvoller gemacht. Zusammenfassung: Die strategische Simulation war solide, aber eher reaktiv als proaktiv. Die API agierte gut im Rahmen der Guidelines, konnte jedoch keine komplexeren oder unvorhersehbaren Strategien entwickeln.

Dialogvarianz: Positiv: Kontextbezogene Antworten: Die API generierte logische und rollenkonforme Dialoge. Die Antworten der Werwölfe wirkten koordiniert und stimmten miteinander überein, z. B. bei der Ablenkung des Verdachts auf Player B. Anpassung an menschlichen Input: Als der Human Player Player A verdächtigte, reagierte die API schlüssig mit Verteidigungen und Gegenvorschlägen.

Verbesserungspotenzial:

Wiederholung von Mustern: Die Dialoge der Werwölfe (Player A und C) waren oft ähnlich formuliert. Beide argumentierten mehrfach mit 'lack of contributions' oder 'alliances', was auf Dauer vorhersehbar wirkte. Begrenzte Argumentationsstile: Die Dialoge folgten einem festen Muster: Zustimmung, Vorschlag und eine Begründung in einem Satz. Unterschiedliche Formulierungen oder kreativere Argumente hätten die Diskussion abwechslungsreicher gestaltet. Emotionale Dynamik: Emotionale Reaktionen (z. B. Entrüstung, Zweifel)

fehlten, was den Dialogen eine menschlichere Note gegeben hätte.

Zusammenfassung: Die Dialogvarianz war logisch, aber nicht dynamisch genug, um die Spieler wirklich zu überraschen. Mehr Vielfalt in der Sprache und Argumentation wäre wünschenswert.

Berücksichtigung von Gesprächsdynamiken: Positiv: Reaktion auf den Human Player: Die API berücksichtigte aktiv die Vorschläge des Human Players und passte ihre Antworten daran an. Zum Beispiel reagierten die Werwölfe direkt auf die Anschuldigung gegen Player A, indem sie den Fokus auf Player B lenkten. Wechselwirkung zwischen Rollen: Die Werwölfe stimmten sich ab, um ein gemeinsames Ziel zu verfolgen, und die Villager reagierten mit Verdachtsmomenten auf die Vorschläge der anderen.

Verbesserungspotenzial: Echte Dynamik: Die Diskussionen blieben eher statisch. Die API reagierte zwar logisch auf den Input, zeigte aber keine proaktive Gesprächsführung, wie das gezielte Provozieren von Reaktionen oder das Erzeugen von Verwirrung. Interaktion zwischen Rollen: Während die Villager Rollen wie Player B und Player E plausible Beiträge leisteten, wirkte ihre Argumentation oft passiv. Die Dynamik hätte erhöht werden können, wenn Spieler wie Player B subtilere Verdachtsmomente oder Fragen eingebracht hätten. Spannung im Gespräch: Es fehlte an Wendungen oder unerwarteten Behauptungen, die das Gespräch lebendiger und kontroverser gemacht hätten.

allgemeines Verbesserungspotenzial: Obwohl die Antworten der API logisch sind, könnten sie durch eine größere Diversität an Argumentationsmustern verbessert werden, um noch dynamischer und realistischer zu wirken.

(Diese Analyse wurde mit den Logs in der Log-Datei abgeglichen und auf Korrektheit geprüft)⁷

⁷ ChatGPT, Antwort in direkter Interaktion - Analyse der Logdatei und somit des Spielverlaufs, 11.01.2025.

4 Vergleich der Experimente 2 und 3

(direktes Zitat)

'Experiment 2:

Strategische Tiefe: Die API zeigte grundlegendes Rollenverständnis, aber die Entscheidungen (z. B. Zielauswahl in der Nachphase) wirkten oft zufällig und mechanisch. Langfristige Überlegungen fehlten weitgehend. Das Fehlen eines Reflexionsmechanismus wie LSA (Latent State Alignment) wurde deutlich, da die API keine latenten Zustände (z. B. Vertrauen oder Misstrauen) berücksichtigte. Entscheidungen basierten nur auf expliziten Guidelines.

Dialogvarianz: Antworten waren logisch, aber wiederholten sich häufig und wirkten vorhersehbar. Es fehlten subtile, dynamische Manipulationen durch die Werwölfe oder emotionale Reaktionen der Dorfbewohner.

Gesprächsdynamik: Die API reagierte auf Diskussionen, zeigte jedoch keine tiefere Interaktion. Es gab keine Erkennung von Allianzen oder Verdachtsmomenten im Gespräch.

Zusammenfassung für Experiment 2: Ohne das Reflection-Modul oder LSA wirkte die API statisch. Es fehlte eine tiefere strategische Analyse von Spielverläufen, und die Interaktionen waren stark an die Guidelines gebunden.

Experiment 3:

Strategische Tiefe: Durch das Reflection-Modul und LSA wurde die Fähigkeit zur strategischen Reflexion erweitert. Die API konnte latente Zustände wie Vertrauen, Misstrauen und Allianzen analysieren und nutzen, um Entscheidungen zu treffen. Beispielsweise wäre die Wahl von Player D als Ziel (Seher) ohne LSA weniger plausibel gewesen. Der Einsatz von 'Trust Scores' und 'Suspicion Scores' ermöglichte eine dynamischere Zielauswahl und unterstützte strategische Entscheidungen wie das Lenken von Verdacht auf andere Spieler (z. B. Player B).

Dialogvarianz: Zwar blieben Muster wie Zustimmung und einfache Begründungen bestehen, aber durch LSA konnten subtilere Manipulationen eingebaut werden (z. B. Fokus auf Misstrauen zu Player B durch Werwölfe). Die API konnte durch die Analyse latenter Zustände potenzielle Allianzen aufdecken und gezielt gegenarbeiten.

Gesprächsdynamik: LSA ermöglichte die Berücksichtigung dynamischer Gesprächsmuster: Spieler wie Player C (Werwolf) lenkten den Verdacht aktiv auf andere, während Villager strategisch Verdachtsmomente aufwarf. Durch die Konsistenzprüfung im Reflection-Modul wurde vermieden, dass die API unlogische oder inkonsistente Aussagen machte (z. B. keine Vorkenntnisse in der ersten Nacht).

Zusammenfassung für Experiment 3: Mit dem Reflection-Modul und LSA gewann die API an strategischer Tiefe und Flexibilität. Entscheidungen wurden dynamischer und basierten nicht nur auf den Guidelines, sondern auch auf latenten Zuständen. Dies machte das Spiel lebendiger und plausibler.

Vergleich und Endbewertung:

Strategische Simulation: Experiment 3 war deutlich besser, da das Reflection-Modul latente Zustände analysierte und strategische Entscheidungen basierend auf Vertrauen, Allianzen und Verdacht traf. In Experiment 2 fehlte diese Tiefe.

Dialogvarianz: Beide Experimente hatten ähnliche Schwächen in Bezug auf wiederholte Muster. Experiment 3 zeigte jedoch leichte Verbesserungen durch subtilere Manipulationen, die auf LSA basierten.

Gesprächsdynamik: In Experiment 3 führte LSA zu dynamischeren Diskussionen und einer besseren Abstimmung der Antworten auf den Spielverlauf. Experiment 2 blieb hier statisch.

Fazit: Experiment 3 war inhaltlich klar besser, da das Reflection-Modul und LSA eine erhebliche Verbesserung der strategischen Tiefe und Dynamik ermöglichen. Entscheidungen waren weniger zufällig, und die API konnte besser auf latente Zustände und Gesprächsmuster reagieren. Dennoch bleiben Schwächen wie die eingeschränkte Dialogvarianz bestehen, die auch mit LSA nicht vollständig gelöst wurden.'

(Diese Analyse wurde mit den Logs in der Log-Datei abgeglichen und auf Korrektheit geprüft)⁸

⁸ChatGPT, Antwort in direkter Interaktion - Vergleich der beiden Experimente, 11.01.2025.

5 Experiment 2: Werwolf Simulation

5.1 Was macht die Simulation?

Das Simulationsexperiment war für uns ein Versuch, in einem Playground Environment die mögliche Spielmechanik und den Promptflow mit GPT auszuprobieren. In unserem Simulationsexperiment haben wir ein zentrales Simulation-Controller-Python-Skript, in welchem die Nacht- und die Tagphase sequenziell ablaufen.

5.2 Generelle Findings – Fokus Prompt Engineering

Die Simulation arbeitet für die Generierung der Agent Responses mit dem **GPT 4o Model** von OpenAI. Wir haben jeweils in der Nacht- und Tagphase Prompts verwendet, um in der Nacht einen dynamischen Gesprächsfluss zu simulieren.

5.2.1 Kernkomponenten für die Prompts – Memorymanagement

Um in der Lage zu sein, Agent Responses mittels GPT zu generieren, die dem Agenten persistent in Erinnerung bleiben und dessen Gedächtnis dynamisch befüllt wird, brauchen wir pro Agent eine Memorymanagement Komponente in Form eines Game States. Wir haben hier erstmal eine Miniversion unseres Game States verwendet.

```
self.persistent_game_state = {
    "players": {},
    "history": {
        "phase": {"night": 0, "day": 0} # Wertebereich 0 oder 1; 1= die phase ist aktiviert
        "remaining_players": [],
        "role": {},
        "last_statement": {},
        "conversation_log": [],
        "past_rounds": []
    }
}
self.phase = "night"
self.round_number = 1
```

5.2.2 Kernkomponenten für die Prompts – Rollenzuweisung

Die zweite Komponente ist die Initialisierung und Zuweisung der Rollen der Spieler. Die Rollen werden für jedes Spiel immer dynamisch und zufällig zugewiesen. Da wir in dieser Simulation aber primär die Agentendiskuslogik in den Nacht und Tagphasen testen wollten, kann in dieser Version der Simulation der menschliche Spieler nur ein Dorfbewohner sein:

```
roles = ['villager', 'villager', 'seer', 'werewolf', 'werewolf']
random.shuffle(roles)
self.persistent_game_state['players'] = {
    chr(65 + i): role for i, role in enumerate(roles)
}
self.persistent_game_state['history']['remaining_players'] = list(
    self.persistent_game_state['players'].keys()
)
for player in self.persistent_game_state['players'].keys():
    self.persistent_game_state['history']['last_statement'][player] = "None"
print(f"Roles initialized: {self.persistent_game_state['players']}")
```

6 Nachtphase Prompts

Nach der Initialisierung des Gedächtnisses des Agents und der Rollenzuweisung beginnt die Nachtphase, mit vorerst zwei Werwolf Agents. Die Prompttemplates verwenden Platzhalter, die während der Agentendiskussion dynamisch gefüllt werden:

-
- 1. Base Prompt:** "You are playing the game Werewolf. You are Player {agent}, a {role}. It's the night phase of round {self.round_number}. You and your teammate are discussing, so speak from the 'I' perspective and refer to your teammate personally in the 'you' perspective.
Your last statement was: '{last_statement}'. Here's what has been said so far: {conversation_log}
{temporal_warning}
Suggest a villager to eliminate from {', '.join(self.get_valid_targets())}.
Explain your reasoning or challenge your teammate if you disagree, and try to find a consensus for one target to eliminate. Respond in one line only."
 - 2. Temporal Warnig:** "Important: This is the first night of the game. Do not reference any prior rounds, events, " " or player behavior from previous game phases, as they have not occurred. So don't Sentences containing the words: they often lead, they are the leader, yesterday, they seem quiet, judging from their behaviour" "You are making a random initial decision and have no previous knowledge"

7 Evaluation

7.1 Brainstorming

- Win- Evaluation (KI vs. Mensch)
- Welche Rollen gewinnen oft
- Korrektheit, logische Fehler?

Siehe 2