UNIVERSITATEA DIN BUCUREȘTI

FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ

PROGRAMUL DE MASTER ARTIFICIAL INTELLIGENCE

Lucrare de disertație

GENERATIVE AI - GENERATION OF PROCEDURAL STORIES AND/OR FOR COMPUTER GAMES USING PROMPTS

Absolvent

Bogdan Radu Silviu Sielecki

Coordonator științific

Conf. univ. dr. Ciprian Păduraru

București, Iulie 2025

Abstract

This dissertation explores the use of generative AI (LLMs), in procedural storytelling for games and interactive media. The study investigates how multi-agent systems, powered by pre-trained models hosted locally via Ollama or via the Gemini API, can interact and respond dynamically to evolving world events from a specific character point of view. Agents are designed with persistent memory, defined roles, and structured goals, supported by a custom dialogue turn engine and a simple Retrieval-Augmented Generation (RAG)[13]. The system was evaluated across multiple narrative scenarios such as mystery, conflict, betrayal, and loss using both quantitative metrics—coherence, (reactivity, memory utilization, diversity, latency and RAG precision) and qualitative assessments of agent behavior and narrative flow.

Results demonstrate that while agents maintained consistent personalities and localized coherence, their contributions did not have any meaningful impact in the story progression. Dialogues frequently drifted without structured interventions, highlighting the need for more robust story progression mechanics. RAG integration showed promise for contextual grounding, but required more sophisticated retrieval and scoring strategies. High latency and passive agent behavior were also identified as limitations, particularly in real-time applications.

The findings suggest that although generative agents can enhance procedural storytelling, their effectiveness is highly dependent on surrounding systems for goal enforcement, memory management, and narrative scaffolding. Future work could explore fine-tuning LLMs for dialogue-driven scenarios, improving RAG pipelines, and integrating planning or plot management systems to support a more autonomous story generation.

Contents

Chapter 1

Introduction

1.1 Understanding Generative AI

Generative Artificial Intelligence (AI) refers to a class of machine learning models designed to create new content including text, images, audio and video by identifying patterns in vast datasets. Unlike traditional AI systems that rely on rule based logic and predefined responses, generative models can produce original and contextually coherent outputs. They achieve this by predicting the most probable next element in a sequence, whether it be a word in a sentence, an action in a game or a visual element in an image.

While early concepts of generative models date back to the 1960s, significant advancements emerged with the introduction of Generative Adversarial Networks (GANs)[11] in 2014. Since then, generative AI has evolved rapidly, particularly with the development of transformer based architectures, such as GPT[32], LLaMA [26] and Gemini[10]. These models were trained on large scale training datasets and employ attention mechanisms to generate highly sophisticated and context aware outputs.

Large Language Models (LLMs) are a subset of generative AI designed to operate as statistical predictors, they determine the most likely sequence of words based on previously observed data. Due to this behaviour LLMs do not inherently comprehend the meaning of the text they generate. This marks a key limitation for LLMs. Furthermore, because these models are trained on static datasets, they struggle to adapt to real time changes or new information without additional training or mechanisms like Retrieval Augmented Generation (RAG)[13]. LLMs may also struggle in long conversations or narratives by losing coherence. They are heavily reliant on the quality of the training data. If the data is noisy, incomplete or biased, the model's outputs will reflect those flaws. Furthermore, training data can embed societal biases, leading to prejudiced or inappropriate outputs.

Despite their limitations, generative AI models offer several advantages. They can produce coherent and contextually appropriate responses, making them useful for chatbots, virtual assistants and non player characters (NPCs). These models also support task

automation by handling repetitive and mundane activities such as text summarization, data extraction and content reformatting. Additionally, generative AI enhances human like interaction by simulating natural conversational patterns, which improves the realism and engagement of interactive systems. They are capable of generating creative and domain specific content, including poetry, scripts and personalized storytelling. Moreover, generative models can condense complex or lengthy texts into concise and meaningful summaries, aiding comprehension and decision making across various applications.

1.2 Procedural Storytelling

Procedural storytelling refers to the dynamic generation of narratives based on predefined rules, AI decisions or user input. Unlike static storytelling, which follows a fixed script, procedural storytelling enables adaptive narratives that react to player choices, game states and random events. This approach creates unique and replayable experiences, ensuring that no two interactions are exactly alike.

Procedural storytelling is widely used across various domains, particularly in gaming and interactive experiences. In video games, titles such as Dwarf Fortress[5], AI Dungeon[23] and The Elder Scrolls make extensive use of procedural generation to construct worlds, dynamic quests and narratives that evolve and change with player choices. In the realm of interactive fiction, AI driven storytelling enables branching dialogues in text based games and visual novels, enabling player driven narratives that respond to individual decisions. Furthermore, AI powered virtual assistants and non player characters (NPCs) can generate context aware responses in real time, significantly enhancing the realism and immersion of digital environments.

Traditional procedural storytelling systems typically rely on predefined logic, hand crafted rules and scripted decision trees. While effective in certain scenarios, these approaches often result in predictable and repetitive narratives. Generative AI introduces a significant shift by enabling adaptive, contextually rich and dynamically evolving stories that go beyond the constraints of static scripting. Conventional systems can only handle a finite number of narrative paths, limiting creativity and player engagement. As branching options are manually designed, they tend to become repetitive over time. In contrast, generative models offer the potential for unlimited storytelling, but as of now, not without flaws.

One of the most pressing issues with LLMs is the occurrence of hallucinations: outputs that are factually incorrect, inconsistent with prior context or illogical. In narrative applications, this can manifest as incoherent plots, contradictions in character behavior or implausible world building elements. Moreover, LLMs lack understanding or reasoning. They do not learn after deployment and remain static unless retrained. As a result, they require mechanisms like Retrieval Augmented Generation (RAG)[13] to incorporate real

time knowledge and maintain narrative relevance. Some scholars even argue that LLMs might be better described as "Artificial Knowledge" systems rather than true Artificial Intelligence, due to their lack of organization, adaptability and reasoning.

To overcome these challenges, emerging approaches seek to enhance generative storytelling through a combination of methods. With Retrieval Augmented Generation (RAG)[13] you can integrate external data sources to ground AI outputs in factual information. Training on domain specific narrative datasets can help in reducing hallucinations and maintaining coherence. Memory enhanced AI agents preserve consistency in character behavior and world logic across multiple dialogue turns. By incorporating these techniques, generative AI can elevate procedural storytelling, producing more coherent, responsive and compelling narratives that retain creative nuance while leveraging the power of automation.

For example, in a role playing game, a Generative AI model could create a unique quest based on a player's previous actions, alter NPC behavior dynamically in response to ingame development or introduce unexpected world events that reshape the storyline in real time. This dissertation explores how Retrieval Augmented Generation (RAG)[13] can further support these capabilities, ensuring that AI generated dialogues remain contextually grounded, responsive to the current world state and enriched by external knowledge sources.

Chapter 2

Literature Review

2.1 Background and Evolution

Procedural storytelling refers to the automatic generation of narrative elements using predefined rules, algorithms or systems that respond to player input or world changes rather than relying on predefined scripts or linear storytelling methods. This concept has deep roots in early computer science and interactive media. One of the first examples, ELIZA (Weizenbaum, 1966)[30], simulated a psychotherapist using pattern matching and scripted responses. The concept of procedural storytelling has evolved significantly over the past few decades, particularly with the advancement of generative AI technologies.

Early attempts at procedural storytelling were limited in scope and largely depended on rule based systems. These early systems relied on predefined scripts and decision trees, generating stories by combining various elements such as characters, locations and events based on user or system inputs. Games like Rogue (1980)[25] and Dwarf Fortress (2006)[5] exemplified early attempts at procedural generation, where the game world was procedurally created and player actions would shape the unfolding events. These systems, remained restricted by their static logic and repetitive patterns, which limited their potential for generating meaningful, dynamic narratives.

As technology progressed, games such as Zork (1980) introduced text based interactive fiction, allowing players to navigate branching narratives via typed commands. These early systems relied heavily on deterministic scripting and logic trees. Throughout the 1990s and early 2000s, procedural generation became more sophisticated with games like The Sims and Neverwinter Nights using behavior trees and rule based AI to simulate character interactions and quest generation. Intent based story generators like Façade (Mateas & Stern, 2005)[15] demonstrated how AI planning and emotional models could create more nuanced, real time character interactions. These systems allowed characters to make decisions dynamically, offering limited emergent behavior and responding to user input. Despite these advancements, such systems were still limited in narrative depth and

often resulted in repetitive or disjointed experiences.

The 2000s saw a shift toward data driven models. Statistical language models and Markov Decision Processes (MDPs)[22] were used to generate text and simulate user choices. Although less deterministic, these models lacked deep contextual understanding, often resulting in incoherent outputs in longer narratives. In 2010 there was a shift with the rise of deep learning, particularly Recurrent Neural Networks (RNNs) and Long Short Term Memory (LSTM)[28] networks were first applied to sequential storytelling, enabling models to learn patterns from large corpora of text. However, these early models suffered from context limitations and often generated incoherent or repetitive text.

The introduction of transformer based architectures, notably GPT (Radford et al., 2018)[19] and its successors marked a significant breakthrough. These models demonstrated an unprecedented ability to generate fluent, contextually rich narratives across a wide range of genres and topics. Transformers revolutionized narrative AI by introducing attention mechanisms that allowed models to maintain long term context. Models like OpenAI GPT 3[4], GPT 3.5[18],GPT 4[17]. LLaMA[26][27][16], Claude[3][2] and DeepSeek[8] became capable of producing human like storytelling with minimal prompt engineering. These models marked a leap forward in enabling open domain dialogue, emergent narratives and personalized story arcs. Trained on internet scale corpora, transformer models can simulate character dialogue, generate quests or even compose multi turn narratives with high levels of coherence.

Generative models, showed that AI could generate large, open ended text segments that made sense contextually and stylistically. These models relied on massive datasets to learn the relationships between words, sentence structures and common story tropes, leading to the creation of narratives that felt more human like. In the game AI Dungeon (2019)[23], the use of GPT 2[19] enabled the creation of dynamic stories where users could interact with the AI to shape the narrative in real time. AI Dungeon and similar platforms leveraged generative models to create branching, user directed narratives that react to player input, moving beyond traditional static storytelling systems. StoryGAN (Li et al., 2019)[14] and PlotMachines (Rashkin et al., 2020)[20] are examples of generative models trained to produce visual or textual stories. These systems focused on learning narrative coherence and character consistency across multiple story segments.

Despite their advances, these models are still bound by the limitations of the datasets they are trained on and often suffer from issues like hallucinations and narrative inconsistencies. More recent developments such as ReAct (Yao et al., 2022)[31] and Chain of Thought prompting (Wei et al., 2022)[29] have expanded the reasoning capabilities of LLMs, enabling them to maintain logical consistency across events and simulate causal consequences within a story. Recent innovations integrate large language models with retrieval systems (ex: Retrieval Augmented Generation (RAG)[13]) and memory modules. These hybrid systems allow AI agents to access external information, remember

past interactions and maintain coherent world states across long form dialogues, making it valuable in procedural storytelling, where consistency and adaptiveness are crucial.

This evolution has laid the foundation for dynamic storytelling agents capable of blending preauthored world logic with generative fluency. Today's narrative AIs are not only storytellers but interactive collaborators, guiding experiences that are immersive, responsive and personalized to each player or user.

Despite notable advancements, AI driven procedural storytelling continues to face several critical challenges. One persistent issue is consistency and coherence. AI generated narratives frequently exhibit inconsistencies in character behavior, plot development and world building. Characters may act out of alignment with their established traits and storylines can take abrupt or illogical turns due to limitations inherent in the model's training data and contextual awareness.

Another major challenge lies in memory and continuity. Most current AI narrative systems lack effective mechanisms for long term memory, restricting their ability to recall past events or adapt meaningfully to user input over extended interactions. This often results in disjointed storytelling, where characters fail to demonstrate learning or development across narrative arcs.

Furthermore, the emergence of genuinely emergent behavior remains limited. While human authored systems can occasionally foster complex, unexpected developments, AI models typically rely on predefined or semi scripted pathways, constraining their adaptability and the richness of emergent narratives.

This dissertation seeks to advance the field by integrating a Retrieval Augmented Generation (RAG)[13] architecture with a dialogue turn engine to simulate coherent, multi agent conversations within dynamic environments. This approach specifically targets the improvement of narrative consistency, persistent world state awareness and the evolution of personality driven dialogue core challenges that remain unresolved in previous procedural storytelling systems.

2.2 Memory Systems and Conversational Agents

Dialogue systems, particularly those powered by artificial intelligence, require some form of memory to maintain coherent, context aware conversations. Memory is essential for enabling systems to retain character backstory, recall previous interactions and respond consistently across time. This section explores the evolution of memory mechanisms in dialogue systems, from static context windows to advanced neural and retrieval augmented memory architectures.

Traditional transformer based language models like GPT 2[19] and GPT 3[4] utilize a context window that allows the model to "remember" a fixed length history of the conversation. While this enables short term continuity, it falls short in longer interactions.

Key character traits, long term goals or earlier events can easily be forgotten once they fall outside the context length.

To address this, developers and researchers introduced external memory buffers that store important past exchanges. Memory modules record significant user agent interactions and selectively inject them into the prompt, improving consistency over time. For example, ReAct[31] and AutoGPT[21] architectures maintain task relevant information in memory slots that are retrieved when needed. Dialogue agents like Replika, Character.AI and Inworld.ai use variations of this approach. They dynamically pull past character memories or facts to ground responses in previously established behavior, enhancing believability.

Another advancement involves storing interaction history as embeddings in vector databases, enabling semantic search to retrieve the most relevant information. This retrieval augmented memory improves long term consistency by surfacing past conversations, emotions or decisions relevant to the current interaction. Systems like LangChain[6] and private implementations using FAISS[9] or Pinecone integrate this technique into multi turn dialogues. For example, a fantasy character might remember that it previously met a dragon peacefully and thus treat it differently in the future, even if that event happened hundreds of tokens ago. Some systems simulate human like memory by distinguishing between episodic memory (specific events) and procedural memory (habits or skills). In narrative contexts, this allows agents to recall specific plot events (ex: "The dragon spared the village") and respond with learned behaviors (ex: always salute when entering the village). Projects like LIDA[7] and Cognitive Architectures (ex: ACT R[1], SOAR[12]) have explored memory organization inspired by cognitive science. These insights are now being reflected in neural architectures like MemGPT, which dynamically manages short and long term memory across dialogues.

Despite continued progress, several limitations remain in managing the memory for application of generative AI to procedural storytelling systems. One key issue is context prioritization: determining the most relevant past information from a large pool of memories is inherently complex. Without effective mechanisms for relevance ranking, AI systems may overlook critical narrative elements or retrieve unrelated details, resulting in weaker coherence. Another challenge is memory corruption. As dialogue and character memories evolve, inconsistencies can emerge, leading to contradictions in behavior or a gradual erosion of established personality traits over time. Scalability also poses a significant constraint, as embedding and searching large memory banks require substantial computational resources, particularly problematic in real time or resource limited environments. Finally, safety and privacy remain major concerns. Persisting user dialogue histories and other interaction data introduces ethical risks, especially when such data includes sensitive content. Ensuring secure handling, user consent and responsible data governance is essential for deploying these systems in a trustworthy manner.

In procedural storytelling, memory plays a crucial role in enabling characters to respond meaningfully to evolving narratives and dynamic world states. For instance, when a significant event such as a village being attacked by goblins occurs, memory allows characters to acknowledge and react to that change appropriately. Furthermore, it supports emotional and narrative consistency—for example, ensuring that a grieving character does not exhibit incongruent cheerful behavior. Memory also fosters a sense of continuity across different gameplay sessions or episodic story arcs, preserving the progression of character development and world events. This dissertation introduces a lightweight memory module integrated with retrieval techniques, enabling each agent to sustain both narrative awareness and personality coherence across multi turn dialogues, even as the environment and context shift over time.

2.3 Dialogue Management and Multi Agent Systems

Turn taking refers to the mechanism that governs who speaks next, when they speak and for how long. In human conversation, this process is naturally guided by social norms, body language and contextual cues. However, in AI driven dialogue—particularly in environments involving multiple agents—turn taking must be explicitly controlled through logic or algorithmic strategies. Basic implementations of turn management typically include round robin scheduling, where agents take turns in a fixed sequence; state based triggers, in which specific events like a world event or user action determine the next speaker; and dialogue trees, which define predetermined paths and speaker sequences, commonly used in traditional game dialogue systems.

While these simple approaches can work in constrained or linear scenarios, procedural storytelling requires more adaptive and context aware mechanisms. Effective turn taking must be responsive not only to external world changes but also to the internal motivations, personalities and emotional states of characters to maintain narrative believability and engagement.

Multi agent dialogue systems have been widely applied in role playing and open world games such as Skyrim and Baldur's Gate, where scripted interactions unfold between non player characters. Similarly, simulation based narratives like Prom Week and Façade employ social AI agents capable of dynamic interaction with players and other agents, driven by emotional models and social rules. Narrative planning systems such as Versu and Comme il Faut (CiF) go a step further by simulating beliefs, goals and interpersonal relationships to generate emergent dialogue and character behavior.

Despite these advances, most existing systems are still constrained by finite state machines and heavily authored logic. This limits their capacity for scalable and emergent storytelling, as the range of possible interactions remains bounded by predefined structures rather than evolving organically from context and character driven dynamics.

Recent advancements have enabled agents powered by Large Language Models (LLMs) to engage in open ended dialogues with one another. These multi agent LLM frameworks can simulate a wide range of interactions, including debates, collaborative problem solving and character driven conversations, by dynamically managing dialogue flow and role assignments. Notable examples include systems like ChatArena and multi agent debate simulations, which allow LLMs to represent distinct viewpoints within structured discussions. AutoGen offers a framework for coordinating autonomous agents capable of reasoning and planning through natural language, while Voyager demonstrates how LLM agents equipped with memory and external tools can learn and explore within open world environments such as Minecraft.

To support these capabilities, such systems often incorporate orchestration mechanisms or Dialogue Turn Engines. These components manage various aspects of the interaction, including agent selection (determining which agent should speak or respond), message routing (controlling how information is exchanged between agents) and context injection (ensuring each prompt includes relevant world state and memory for coherence and continuity).

A Dialogue Turn Engine (DTE) functions as a logical or procedural controller responsible for coordinating interactions among agents and the environment. It tracks conversation history and world events, determines which agent should respond next based on factors such as salience, urgency or individual character goals and injects relevant memories or emotional states into agent prompts. Additionally, the DTE is equipped to handle interruptions or external developments, such as a sudden dragon attack or an unexpected betrayal. These engines typically blend rule based logic with the reasoning capabilities of language models to strike a balance between structured control and creative adaptability. As a result, storytelling agents can respond not merely to the most recent utterance, but to multi dimensional contexts that include environmental triggers, emotional trajectories and narrative pacing.

Despite the significant potential of such systems, several challenges remain. Maintaining coherence is a persistent issue, as agents may contradict themselves or one another across multiple turns. Scalability becomes increasingly complex as the number of agents grows, complicating the management of shared and individual context. Handling interruptions requires dynamic systems that can reprioritize actions and adapt plans in real time. Furthermore, managing turn timing is essential for ensuring believable narrative flow and preserving the pacing of interactions.

This dissertation builds upon these foundations by implementing a turn based multi agent framework using local LLMs through Ollama. Each agent possesses its own memory, receives updates from a shared world state and is prompted within a structured turn engine loop that reflects evolving events. The Dialogue Turn Engine dynamically determines which agent should respond next and integrates pertinent world information into the

dialogue, fostering context aware, responsive and emergent conversations in procedurally generated narratives.

Chapter 3

Methodology

3.1 Research Design

The research design outlines the strategic framework used to guide the development, implementation and evaluation of the proposed multi agent dialogue system for procedural storytelling using generative AI. This study follows a design based research methodology, which is particularly suited for iterative development and testing of innovative technological systems in realistic settings.

This research adopts a qualitative, exploratory methodology. The system is developed through iterative prototyping, allowing for continuous refinement of agents, dialogue mechanisms and event responses. A working prototype is implemented using locally hosted language models (LLaMA3[16], DeepSeek[8] and Gemma[24] via Ollama and Gemini2[10]). Its functionality is tested in simulated narrative scenarios. The resulting agent dialogues are logged and analyzed thematically to assess narrative flow, memory retention, personality coherence and responsiveness to world events.

The study is guided by several research questions: How effectively can generative AI agents maintain coherent multi turn dialogue within a dynamic narrative context? To what extent can agents demonstrate consistent personality and memory throughout procedural storytelling? How do external world events influence the direction of agent dialogue and story progression? And finally, what design principles are necessary to manage dialogue turns within a multi agent storytelling framework?

The development of the procedural storytelling system followed a structured, iterative process comprising five main phases: requirement analysis, architecture design, implementation, testing and refinement and evaluation. This phased approach ensured both conceptual clarity and practical efficacy in building a dialogue driven multi agent simulation.

The process began with requirement analysis, where the essential features necessary for the system's operation were identified. These included memory simulation to enable

continuity across dialogue turns, personality modeling to differentiate agent behaviors, integration of dynamic world events to drive narrative progression and a structured dialogue turn mechanism to manage agent interactions effectively. The goal at this stage was to define a foundational blueprint that would guide subsequent development.

Following this, a modular architecture was designed to support the diverse components of the system in a scalable and maintainable way. This architecture included encapsulated agent classes with embedded memory and persona modules, a centralized world state handler to maintain environmental context, a dialogue turn engine to manage multi agent interactions and an extensible framework that could later incorporate additional features such as Retrieval Augmented Generation (RAG)[13] or emotion conditioned prompting.

In the implementation phase, the system was developed using Python, with large language models hosted through Ollama serving as the generative backbone. Agents were instantiated as object oriented classes, each defined by a unique persona, memory buffer and dialogue method. System prompts were dynamically constructed to include both world state updates and rolling conversation history, allowing each agent to respond in a manner consistent with both past events and their individual character definitions.

The implementation was followed by an extensive testing and refinement phase. Multiple handcrafted narrative scenarios were executed, featuring interactions between diverse characters under various environmental conditions. These scenarios included high stakes conflicts such as a dragon attack, emotionally charged events like the loss of a companion and unpredictable environmental changes such as storms or hidden treasures. The system's performance was observed through generated dialogues and iterative refinements were made to improve coherence, responsiveness and character fidelity.

Finally, the evaluation phase involved a combination of automated logging and manual qualitative analysis. Dialogue transcripts were analyzed for narrative coherence, turn taking fluidity and agent consistency. Both structural features, such as response length and entity mention accuracy and subjective qualities, such as believability and engagement, were assessed. This evaluation informed the overall effectiveness of the system in simulating believable, interactive storytelling.

This research adopts a design based research (DBR) methodology, which is particularly well suited for the development of functional artifacts intended for real world application. The core strength of DBR lies in its capacity to integrate theoretical principles with iterative design and implementation. In this project, DBR facilitated the direct application of concepts from generative AI and interactive narrative theory into the construction of a working multi agent storytelling framework.

Design based research also promotes continuous refinement through experimentation, which was critical in adjusting the system in response to unexpected outcomes or performance issues during testing. The ability to observe, evaluate and revise the system in real time allowed the design to evolve dynamically in alignment with the project's objectives.

Moreover, DBR supports the development of reusable design principles and frameworks. In the context of this work, the methodology enabled the creation of a modular, extensible platform for procedural storytelling that can be applied or adapted in various domains, such as gaming, simulation training or interactive entertainment. The practical orientation and adaptability of DBR make it a compelling choice for research that aims to bridge theory and application, especially within emerging fields like AI driven narrative generation.

3.2 Tools and Technologies

The development and experimentation of the multi agent dialogue system for procedural storytelling required a carefully selected set of tools and technologies. These tools enabled natural language generation, memory handling, world state integration and agent based modeling. This section details the key technologies used throughout the implementation phase. A central component of this research is the use of Ollama, a platform that enables the local execution of large language models (LLMs). Ollama was selected for its simplicity, reliability and emphasis on privacy, as it removes the dependency on external cloud based APIs. This not only enhances system performance and ensures reproducibility but also provides full control over the model querying pipeline, allowing for fine tuned adjustments to prompts, memory inputs and runtime parameters. For the core generative processes, Meta's LLaMA3 was chosen due to its advanced contextual reasoning capabilities and its strength in generating coherent, open ended responses suitable for dynamic storytelling. During the exploratory and evaluation phases, alternative models such as OpenAI's GPT 4[17], Google's Gemini 2[10] and DeepSeek R1[8] were also integrated to conduct comparative analyses. These models were assessed based on their coherence, memory consistency and adaptability in handling complex, evolving dialogue scenarios, offering broader insights into performance variability across leading generative systems.

The entire system was implemented using the Python programming language, chosen for its readability, rapid development capabilities and strong ecosystem supporting artificial intelligence, natural language processing and agent based simulations. Python enabled the development of object oriented agent classes, each of which encapsulated attributes such as personality, memory and behavioral logic. The integration with the Ollama API facilitated seamless interaction with LLMs for dialogue generation. Furthermore, the system included a dialogue turn engine for managing conversational flow, as well as modules for memory handling and conversation logging to support continuous, state aware storytelling.

At the heart of the interaction model lies a custom built Dialogue Turn Engine, which orchestrates the logic and sequencing of multi agent conversations. This engine is responsible for maintaining speaker turn order, tracking agents across exchanges and injecting

relevant world events between turns. It ensures narrative continuity and consistent character interactions, providing a framework where dialogue unfolds naturally in response to both agent driven goals and environmental stimuli. Designed with extensibility in mind, the engine supports future enhancements such as interrupt driven events, conditional branching or goal oriented planning.

Each agent in the system is equipped with a dedicated memory system, implemented as a structured buffer that stores past dialogue exchanges, personal decisions and relevant world events. This memory is continuously updated throughout the interaction and is reconstructed into the agent's prompt at every turn, enabling the simulation of long term character awareness. By allowing agents to recall and respond to past events, the memory architecture contributes significantly to behavioral consistency and narrative believability, simulating the effect of persistent internal states.

In parallel, a lightweight World State Integration module was implemented to simulate external dynamics such as environmental changes, moral dilemmas or the emergence of new threats. This World State Manager modifies the system prompts to reflect evolving narrative contexts, thereby influencing agent behavior in a context sensitive manner. Real time world events are appended to prompts as structured updates, creating a continuous feedback loop between agent perception and environmental evolution—similar to the mechanisms found in game engines and interactive fiction frameworks.

Several supporting tools were used to streamline the development and experimentation process. Jupyter Notebooks facilitated early stage prototyping and interactive testing of dialogue loops, while Visual Studio Code (VS Code) served as the primary integrated development environment for building the modular Python codebase. Git was employed for version control and collaborative development tracking, ensuring reproducibility and organized iteration across development phases. For documentation purposes, Markdown was used to log system architecture decisions, dialogue transcripts and structural design notes in a clear and accessible format.

3.3 System Design and Architecture

The system is designed to simulate a multi agent, AI driven storytelling environment where each agent participates in dialogue, responds to evolving world events and contributes to a procedurally generated narrative. The architecture follows a modular, extensible design that emphasizes agent autonomy, memory continuity and narrative coherence.

At a high level, the architecture comprises five primary components that interact to create a dynamic storytelling loop. These include autonomous agents that represent individual characters; a dialogue turn engine responsible for managing conversational flow and injecting narrative events; a world state manager that simulates external influences

and environmental changes; a local language model interface (facilitated through Ollama) that generates dialogue responses; and an interaction loop that orchestrates the temporal flow of communication, memory updates and event handling.

3.3.1 Agent Architecture

Each agent is implemented as a Python class instance and functions as an independent entity within the narrative environment. The agent encapsulates a set of defining attributes and behaviors, including a name and personality profile expressed through natural language to guide its tone and style of interaction. A system prompt serves as a persistent directive, defining the agent's role, worldview and behavioral tendencies. The agent also maintains a dialogue memory, which is a sequential log of all past utterances and experienced events, forming the basis of continuity and internal context. A response function is used to assemble relevant context into a prompt and query the local language model, ensuring that each response aligns with the agent's persona and memory.

This structure enables agents to sustain coherent dialogue over multiple interactions and to adapt their behavior based on both individual personality and shared narrative developments. As a result, each agent contributes uniquely to the procedural generation of the story, maintaining character identity while remaining responsive to environmental or social changes.

3.3.2 Dialogue Turn Engine

The dialogue turn engine acts as the temporal controller of the system, regulating the flow of interaction across agents. It maintains the turn order, tracks which agent is to respond next and inserts world events at appropriate moments between conversational turns. In doing so, it ensures orderly progression of the dialogue while also accommodating interruptions and scene changes that are essential to narrative dynamism.

The engine supports both synchronous exchanges—where one agent speaks and another replies—and asynchronous interjections, such as world events that interrupt or redirect the dialogue mid flow. It also enables potential extensions for handling more complex interactions, including interrupt driven storytelling or agent behaviors influenced by goals and motivations. All dialogue lines are recorded and may be broadcasted to other agents or external systems, forming a complete transcript of the evolving narrative.

3.3.3 World State Manager

The world state manager is responsible for simulating environmental dynamics that affect the behavior and decisions of agents. These changes may be manually defined or procedurally generated and typically represent events such as physical disturbances (ex:

a storm or battle), social developments (ex: betrayal by an ally) or narrative catalysts (ex: the arrival of a mysterious stranger).

Each world event is formatted as a textual update and inserted into the prompt context provided to the agents. Depending on the implementation, the event may be appended to the system prompt or embedded within the conversational input. This mechanism allows agents to perceive, interpret and react to ongoing changes in the environment, thereby reinforcing the illusion of a living, responsive story world. By managing both the timing and content of these updates, the world state manager plays a key role in shaping the trajectory of the emergent narrative.

3.3.4 LLM Interface with Ollama

Communication with the underlying language models is managed by a local interface to Ollama, which facilitates model querying and response handling. For each interaction, the interface constructs a prompt that incorporates the agent's personality directive, the current world state and the accumulated dialogue history. This layered prompt ensures that responses are grounded in both past events and the current narrative context.

Ollama's local hosting of language models allows for high responsiveness, offline functionality and greater control over model behavior. Model outputs are parsed and logged as part of the agent's memory, forming the basis for future interactions. This architecture provides a stable and reproducible environment for evaluating different generative models and fine tuning prompt structures and it supports iterative development without relying on external APIs or cloud based latency.

3.3.5 Interaction Flow

The interaction loop defines the runtime logic of the system. It begins with the initialization of agents and the establishment of their respective system prompts. The narrative is then initiated with an opening prompt, which may be delivered by a narrator agent or provided as user input. The agents proceed to take turns responding, with each response influenced by the agent's memory and the current state of the world.

At any point during this process, a world event can be introduced, altering the context in which subsequent dialogue occurs. This interplay between agent turns and environmental updates continues iteratively until the narrative reaches a natural stopping point—either through the achievement of a story goal, the unfolding of a climax or a predefined termination condition. The loop mirrors the structure of interactive fiction, wherein characters actively co create the story in response to unfolding developments.

3.3.6 Extensibility

The modular nature of the system allows for numerous future enhancements. One area of potential development is goal based planning, in which agents act according to long term objectives or internal motivations. Another is the introduction of emotion or affect tracking, enabling agents to alter their responses based on dynamic emotional states such as fear, anger or empathy. Retrieval Augmented Generation (RAG)[13] could also be integrated to allow agents to access external lore or knowledge bases, enhancing contextual depth and narrative richness. Lastly, the architecture is amenable to front end extensions, such as graphical user interfaces (GUIs) or integration with game engines, which would enable real time visualization and interactive storytelling experiences.

3.4 Model Training and Customization

In this research, the use of large language models (LLMs) is central to enabling dynamic dialogue and story progression between AI agents. Although foundational models such as Meta's LLaMA 3 were used without extensive re training, specific customization techniques were applied to tailor their behavior for multi agent procedural storytelling.

3.4.1 Rationale for Using Pretrained Models

Training a large language model (LLM) from scratch demands immense computational resources and access to expansive, high quality datasets. As this research focuses on agent interaction and narrative responsiveness rather than foundational language modeling, the use of pre trained models via the Ollama platform was deemed appropriate and strategically beneficial. Leveraging these existing models allowed for rapid prototyping and experimentation, enabling the focus to remain on dialogue dynamics, memory management and world state responsiveness rather than on model architecture or training procedures.

The adoption of pre trained models provided several key advantages. First, it significantly reduced setup time, allowing for quicker iteration across different interaction scenarios. Second, it enabled access to state of the art generative capabilities without the prohibitive costs associated with model training and infrastructure. Third, it ensured privacy and repeatability through local execution, an essential consideration for a system designed for procedural, interactive storytelling.

3.4.2 Model Selection

A variety of models were evaluated and integrated into the development process at different stages, each contributing unique strengths to the system's behavior and perfor-

mance.

The Meta LLaMA family, including both LLaMA 2 and LLaMA 3, formed the core of the local inference engine via Ollama. These models provided an excellent balance between performance and computational efficiency, making them ideal for running multiple agents in real time. LLaMA 3, in particular, demonstrated improved instruction following behavior and narrative adaptability, making it especially well suited for character driven dialogue and role based interaction.

Gemma, a newer open access model family from Google, was incorporated into the experimental stages for its emerging capabilities in structured text generation and context awareness. Gemma models showed promising results in maintaining character continuity and generating emotionally rich dialogue and were considered a valuable addition to the pool of models compatible with local deployment. Their lightweight architecture also aligned well with the system's modular design philosophy.

DeepSeek R1 was also tested due to its strong support for long context comprehension and compatibility with Retrieval Augmented Generation (RAG)[13] techniques. Its ability to maintain narrative threads across extended interactions made it a compelling candidate for experiments involving world state recall and memory based decision making.

Lastly, Google Gemini 2 was explored primarily as a reference point due to its advanced multimodal capabilities and proficiency in complex intent inference. Although not integrated into the local system due to infrastructure limitations, Gemini 2's capacity for structuring narrative arcs and interpreting contextual cues was noted as a potential asset for future expansions, particularly those involving visual storytelling or multimodal environments.

Through comparative analysis of these models, the system design was iteratively refined to accommodate varied model outputs while ensuring a consistent narrative flow. This multi model strategy enabled broader insights into the strengths and limitations of current generative systems when applied to agent based procedural storytelling

.

3.4.3 Agent Level Customization

The development of an effective prompt format was a central challenge in ensuring the coherence, believability and consistency of multi agent dialogue. Initial efforts began with a simple, declarative approach to character framing. Prompts such as "You are Aegon the Brave, a courageous and slightly overconfident warrior from the Northlands. Speak with valor and confidence" were used to establish tone and personality. While these initial formulations were sufficient for generating characterful responses, they lacked the constraints needed to prevent the model from speaking out of character or adopting multiple roles within a single turn.

To address these limitations, the prompt was expanded to include explicit instructions that reinforced the agent's narrative role and its bounded perspective. The revised version included guidance such as "I want to act according to this character in response to the world events and the dialogues from other companions. Please use dialogue and add a possible action" and "You are a character in a story. You can only see the world events and the dialogues from other companions. Please avoid using the same responses and continue the story as you see fit." This change helped restrict the scope of the model's replies but introduced new issues, including repetitive phrasing, verbosity and occasional breakdowns in speaker identity. Models would sometimes narrate from multiple perspectives or reintroduce other agents' dialogue inaccurately, leading to hallucinated content and disrupted story continuity.

In response to these challenges, a structured prompt format was introduced, designed to constrain output while encouraging variety and emotional nuance. This final iteration divided the prompt into clearly labeled sections: a Role declaration, a Memory Context summary and a Response Format guide. The Role section reiterated the character's identity and instructed the model to respond only in character using one to two sentences. The Memory Context provided a concise log of past interactions and events specific to the agent, helping ground the response in narrative history. The Response Format then asked for a standardized reply including an action, emotion and dialogue line — for example: Action: draws sword, Emotion: determined, Dialogue: "We end this now, together!"

This format significantly improved the model's consistency, limited hallucinations and helped generate shorter, more controlled responses. It also made it easier to parse and visualize outputs in downstream components such as the turn engine or logging tools. Each iteration of the prompt design process was informed by observed system behavior during scenario testing and small refinements—such as adjusting emotion phrasing, clarifying perspective constraints or tightening the memory buffer—yielded measurable improvements in narrative coherence and character integrity.

Prompt engineering, in this context, acted as both a behavioral guide and a narrative scaffold. It enabled the system to maintain structure across dynamic interactions while preserving enough flexibility for creative, emergent storytelling. The iterative nature of prompt design was essential to aligning model outputs with the goals of procedural narrative generation in a multi agent setting.

Over time, the prompt format evolved into a more structured and modular design that tightly aligned with the requirements of dynamic, multi agent storytelling. The current prompt format not only reinforces agent identity but also introduces intentionality and retrieved knowledge into the model's input context. This format was built to guide both coherence and expressiveness, ensuring responses that are not only relevant to the scene but also grounded in agent memory and the evolving world state.

Each prompt begins with a Role declaration, where the agent's name and personality

are reiterated, followed by an explicit instruction to remain in character and to respond using only dialogue, actions or emotions. This reduces the likelihood of the model slipping into narrator mode or adopting the voice of another character. The addition of a dedicated Intent section further refines the response space by aligning each utterance with the agent's current goal. This goal can evolve based on narrative developments, enabling goal driven behaviors and contributing to long term story arcs.

To enforce output discipline and reduce the variability of generative text, the prompt defines a Response Format with labeled fields: Action, Emotion and Dialogue. This not only makes the model's output easier to parse but also introduces narrative rhythm by pairing emotion with physical movement and concise dialogue. An EXAMPLE line is included in the prompt to provide a pattern for the model to mimic, helping stabilize the format during inference.

Two additional sections provide critical context: Relevant Knowledge, which is populated using a retrieval mechanism that fetches pertinent information (such as lore or past events) and Memory Context, which consists of the agent's internal memory buffer drawn from past interactions. These elements simulate long term memory and enable the agent to reference prior experiences or knowledge while maintaining internal consistency. Finally, a World State section captures the latest environmental or narrative updates that the agent should be aware of, such as external events or companion actions.

The adoption of this format yielded significant improvements in coherence, emotional tone and reactivity. Agents became more consistent in their speech patterns, their responses better reflected both current goals and past experiences and the likelihood of hallucinations or narrative derailments was substantially reduced. Moreover, this structure provided a foundation for future expansion, such as automated intent detection, emotion modeling and compatibility with GUI based visualizations or in game scripting engines.

This carefully engineered prompt format acts not only as an instruction interface but also as a behavioral constraint system, turning a general purpose language model into a controlled, responsive narrative agent embedded in a living story world.

3.4.4 Memory Simulation Techniques

To emulate agent memory and continuity of thought across interactions, the system employed several techniques designed to preserve narrative coherence and simulate an evolving internal state. These mechanisms were critical for enabling agents to maintain context, adapt to events and exhibit belief or attitude changes over time.

First, rolling context windows were used to include a selected history of previous dialogue turns and events within each new prompt. This preserved the conversational thread and allowed the agent to reference or respond to earlier developments naturally.

Second, speaker tagging was implemented to attribute each prior line of dialogue to a specific agent. This tagging was essential for distinguishing between characters during multi agent exchanges and for preventing identity confusion within the model's generative process.

Third, the reinsertion of relevant world events into the context window at regular intervals served to re anchor the agent in the ongoing narrative and environmental state. This helped mitigate drift over longer interactions and ensured that agents continued to respond in a manner consistent with the evolving world.

Together, these strategies provided a lightweight simulation of memory, enabling agents to demonstrate continuity, emotional evolution and narrative awareness without requiring persistent neural state or model fine tuning.

3.4.5 RAG Potential and Future Customization

A lightweight keyword based RAG[13] system was implemented as a proof of concept. This module allowed agents to retrieve relevant facts or narrative fragments from a growing knowledge base of world lore, past events and character histories. Retrieved entries were appended to the agent prompt under the [Relevant Knowledge] section, enabling grounded and contextually rich responses without overloading the agent's memory context. This method was particularly effective for reinforcing continuity in cases involving long running story arcs or repeated references to key events.

In its current form, the RAG[13] system supports basic textual retrieval based on keyword overlap. However, the architecture allows for future expansion into more advanced retrieval methods, including embedding based semantic search and long term episodic memory storage. Such improvements would enable agents to reason over extended timelines, recall previous missions or relationships and incorporate detailed world knowledge into their decision making processes.

Beyond retrieval capabilities, several customization paths are planned to enhance agent behavior and interaction fidelity. These include the use of LoRA (Low Rank Adaptation) fine tuning on small, domain specific corpora of fantasy dialogue, allowing agents to adopt more nuanced stylistic patterns. Persona specific tuning, driven by curated dialogue datasets or pre scripted interactions, could further refine consistency across agent responses. Emotional state conditioning—through control tokens or prompt modifiers—presents another avenue for aligning generative outputs with narrative tone and agent goals.

These enhancements, while not part of the initial implementation, are integral to the system's long term vision. By combining retrieval mechanisms with personalization strategies, future iterations of the platform aim to deliver richer, more believable agent interactions across procedurally generated narratives.

3.5 Agent Development

The development of individual agents was central to enabling interactive, personality driven storytelling within the system. Each agent was designed as a self contained entity capable of interpreting world events, recalling relevant knowledge and responding in character, all while maintaining continuity across interactions.

At the core of the agent design is a modular class structure that encapsulates behavior, memory and response generation. Each agent is instantiated with a unique identity composed of a name, a defining personality description and a current narrative goal. These parameters guide both the tone and content of generated responses, ensuring character consistency.

The architecture supports dynamic goal evolution via the 'update_goal' method, allowing for agents to shift objectives in response to plot developments or player interactions. Memory is maintained across two levels: a short term memory buffer (with a configurable size limit) used to simulate recent conversational context and a long term memory store that accumulates structured records of past turns.

3.5.1 Prompt Construction and Response Generation

Each agent constructs a prompt that blends several layers of context:

- Role and Intent: The prompt begins by reinforcing the agent's identity and current objective, framing the model's generative output around a role specific perspective.
- Relevant Knowledge: Information retrieved via the Retrieval Augmented Generation (RAG)[13] module is injected here. These facts are selected based on keyword relevance and help ground the agent's response in prior world lore or event history.
- Memory Context: This section aggregates both long and short term memories. Long term memory includes up to three past structured dialogue entries, while short term memory reflects recent conversational exchanges
- Dialogue Method: A function that constructs the prompt and interacts with the LLM.
- World State: A live description of the environment or current narrative scene is appended to anchor the response in a shared situational awareness.

The prompt is carefully formatted with rigid response expectations, including action, emotion and dialogue fields. A clearly stated example further constrains output formatting. This structured layout, along with reinforced instruction following via system messages, helped reduce hallucinations and discouraged overly verbose or off character completions.

The agent invokes a language model—typically LLaMA 3 through the Ollama platform—for each response. After generation, the output is parsed using regular expressions to extract its component parts. These parsed segments are then recorded back into memory, completing the interaction cycle.

3.5.2 Memory Simulation and Continuity

To emulate agent memory and support evolving narrative coherence, a combination of short term and long term memory is employed. Short term memory maintains dialogue level flow within the immediate context window, while long term memory stores structured event records that can be reintroduced when needed for deeper continuity.

Each memory entry includes the speaker, the utterance, the interpreted emotional state and action and the associated world state. This format allows agents to refer back to specific episodes and adjust behavior accordingly. Over time, this leads to agents forming apparent beliefs, emotional arcs and evolving social dynamics.

When a RAG[13] engine is provided, agents use it to search a shared knowledge base for facts relevant to the current input. This supports contextual retrieval for scenarios involving past events, geopolitical facts, character backgrounds or narrative threads that exceed the model's token limit. Retrieved entries are added to the prompt as bullet points under [Relevant Knowledge], enriching the agent's situational awareness without retraining or hardcoding responses.

Facts can be injected dynamically during runtime using the store_knowledge method, allowing the system to expand its informational grounding as the story unfolds. This lays the foundation for scalable long term memory beyond the constraints of a single interaction session.

Every agent response contributes to the evolving memory architecture and the separation between short and long term context enables gradual refinement of agent behavior over time. As agents accumulate past experiences, their responses naturally shift in tone and content, reflecting learned relationships or story developments.

This cumulative behavior, while not the result of weight based fine tuning, closely simulates persistent agent memory through engineered input structures and selective context reinsertion. The design thus supports the emergence of personality, narrative loyalty and plausible reactions to world events.

3.6 World State Management

In an AI driven narrative system, World State Management is the mechanism that tracks the evolving context of the fictional environment in which agents operate. The world state serves as the shared "reality" that informs the decisions, dialogue and behavior

of all participating agents. This section details how world state is represented, updated and used to influence agent interactions within the procedural storytelling framework.

3.6.1 Purpose of World State

The world state serves as a foundational mechanism within the dialogue system, enabling coherence, continuity and shared understanding across all agent interactions. It provides a consistent frame of reference that all agents can access, ensuring their responses are grounded in a synchronized understanding of the evolving story world. This shared context is essential for maintaining narrative alignment during multi agent conversations.

In addition to aligning agent perspectives, the world state tracks temporal and causal developments in the environment. This ensures continuity across dialogue turns, preventing contradictions and maintaining a logical progression in the narrative. As events unfold, the world state captures these changes, preserving the causal chain that agents rely on to reason about past and future actions.

The world state also introduces challenges and disruptions—such as natural disasters, enemy threats or shifting allegiances—that provide the narrative structure for conflict and resolution. These dynamics elevate story tension, create stakes and encourage agents to respond in ways that drive dramatic pacing. Furthermore, the system allows for external interference in the form of unexpected or autonomous world driven updates, such as a sudden storm, the arrival of reinforcements or the discovery of a hidden item. These changes can significantly alter the direction of a scene, disrupt ongoing plans or shift an agent's priorities, adding unpredictability and richness to the storytelling process.

Without a centralized world state, each agent would interpret prompts in isolation, relying only on local context. This would lead to narrative fragmentation, inconsistencies in world assumptions and diverging timelines, ultimately degrading the coherence of the interactive experience. The centralized world state thus functions as a narrative anchor, synchronizing all agents within a unified and evolving simulation.

The world state is implemented as a narrative sentence or paragraph, embedded in the prompt: *"The dragon is burning the village. It's night and stormy. Villagers are screaming and running."*

The current world state is passed to each agent as part of the system or user prompt at every turn. This ensures that agents not only respond to the last speaker but also adapt to global changes:

```
ollama.chat(
    model=self.model,
    messages=[
        {'role': 'system', 'content': system_prompt},
        {'role': 'user', 'content': memory + '\n' + user_prompt}
```

]
)

Agents then interpret the world event according to their personality and memory context, enabling diverse yet coherent reactions.

3.6.2 Centralization vs Decentralization World State

In designing the world state system, two architectural approaches were considered: centralized world state and decentralized perception. The centralized approach involves maintaining a single global structure that is accessible to all agents. This method offers a consistent narrative baseline, ensuring that every agent reacts to the same version of events. It simplifies coordination and implementation, making it particularly suitable for prototyping and smaller scale simulations. However, as the number of agents or the complexity of the environment increases, this centralized model can become a bottleneck, potentially limiting scalability and responsiveness.

In contrast, a decentralized perception model assigns each agent its own personal and potentially divergent view of the world. This allows for asymmetrical knowledge, where agents may operate based on incomplete, outdated or even false information. Such an approach enables the simulation of secrets, misunderstandings and subjectivity, enriching the storytelling experience with greater realism and narrative depth. However, this complexity comes at a cost, introducing additional system overhead for managing conflicting states and synchronizing key events when necessary.

The current prototype adopts a centralized world state for reasons of clarity, simplicity and consistency. It ensures all agents remain aligned with the core narrative and are grounded in the same environmental updates. Nevertheless, future iterations of the system may incorporate decentralized extensions. These would enable agents to operate with partial knowledge, harbor hidden motives and respond based on conflicting beliefs, opening the door to more intricate and emergent forms of interactive storytelling.

3.7 Evaluation and Performance Metrics

Evaluating a procedural storytelling system powered by generative AI and multi agent dialogue requires a combination of quantitative and qualitative metrics. Traditional accuracy based benchmarks do not fully capture narrative coherence, creativity or character consistency. Therefore, this section outlines a mixed method approach tailored to the unique goals of interactive storytelling and agent based dialogue systems.

3.7.1 Evaluation Goals

The evaluation is guided by the following key objectives:

Narrative Coherence: Assess whether the story progresses logically and consistently across dialogue turns. Each agent's contribution should align with past events and reinforce the overall narrative arc.

Character Consistency: Determine whether agents maintain their defined personalities, goals and behaviors throughout the simulation. This involves examining whether their responses remain faithful to their scripted traits.

Responsiveness to World State: Evaluate how well agents integrate external environmental events and changes into their dialogue. World events such as battles, storms or discoveries should be acknowledged and appropriately reacted to.

Dialogue Turn Fluidity: Measure how naturally turn taking occurs between agents. This includes detecting whether the flow of conversation feels organic and whether agents take turns without dominating or stalling the exchange.

User Perception (Optional): For human facing deployments, subjective feedback can be gathered from users to assess believability, engagement and enjoyment. This may be explored through small scale user studies or expert annotation.

3.7.2 Quantitative Metrics

While narrative experiences are inherently complex and subjective, certain structural and linguistic aspects can be quantified to provide objective insights. These metrics are automatically captured using the DialogueMetricsLogger:

Turn Completion Rate: The proportion of turns in which agents successfully generate responses without timing out or producing errors. This indicates the system's robustness during runtime.

Entity and State Mention Accuracy: Uses semantic similarity to evaluate how accurately agents reference relevant entities or the current world state. This reflects the agent's situational awareness and environmental reactivity.

Repetition Rate and Dialogue Diversity: Assesses linguistic variety using distinct 1 and distinct 2 metrics. High repetition may signal model stagnation or prompt fatigue, while greater diversity indicates rich and varied expression.

Embedding Based Coherence Score: Calculates average semantic similarity between consecutive responses to estimate topical consistency across dialogue turns.

Memory Utilization Score: Evaluates whether agents make use of their memory context in generating responses, offering insight into character development and long term coherence.

RAG[13] Precision at K: In retrieval augmented settings, this metric examines the relevance of the top K retrieved documents by comparing them semantically with the

agent's generated response.

Average Latency: Tracks the response time per turn to assess the system's efficiency, especially in real time or interactive deployments.

These metrics enable automated benchmarking and comparison across different runs, agent configurations and scenario types.

3.7.3 Qualitative Assessment

Beyond automated metrics, qualitative assessment provides deeper insight into subjective dimensions of story quality. Selected dialogues are manually reviewed and annotated based on the following criteria:

Narrative Coherence: Evaluators determine whether the story develops in a way that makes logical and temporal sense from beginning to end.

Personality Fidelity: Each agent is rated based on how well it reflects its predefined traits and motivations. This ensures that character behavior aligns with narrative expectations.

Reactivity: Judges assess whether agents acknowledge and adapt to key world events—such as danger, weather changes or new discoveries—appropriately and in character.

Engagement: Evaluators rate how emotionally or intellectually compelling the dialogue is, reflecting the simulation's overall entertainment value and believability.

3.7.4 Test Scenarios

Evaluation is based on a curated set of narrative scenarios with predefined world state transitions. These scenarios test:

- Conflict resolution (enemy attack)
- Ambiguous moral choices (negotiation vs combat)
- Emotional tension (loss of a companion)
- Environmental surprises (sudden storm or treasure discovery)

Each scenario is run multiple times to observe system variability and response adaptability.

Configuration	**Expected Outcome**
Without World State or Memory	Reduced contextual relevance
World State	Loss of continuity and character depth
World State Updates	Improved coherence and reactivity

3.7.5 Limitations of Evaluation

Despite the comprehensive set of evaluation metrics, several limitations persist. The assessment of qualitative aspects—such as narrative coherence or character fidelity—remains inherently subjective, relying on human interpretation and potentially introducing bias. Additionally, the generative nature of the system leads to non deterministic outputs, making consistent benchmarking across runs challenging. There is also a notable absence of standardized evaluation frameworks specifically tailored to procedural storytelling systems, which complicates direct comparison with related work. Furthermore, the scale of any user study conducted as part of this research is likely to be limited, reducing the generalizability of subjective findings. Nonetheless, the combination of quantitative logging and qualitative analysis provides a multi dimensional perspective on system performance and offers a solid foundation for addressing the core research questions.

Chapter 4

Results and Conclusions

4.1 Experimental Setup

To evaluate the performance of the proposed multi-agent dialogue system for procedural storytelling, a series of structured simulation runs were conducted. The system was implemented using a modular architecture that included autonomous agents, a centralized world state manager, a dialogue turn engine and a retrieval-augmented generation (RAG) component. The agents interacted with each other using the LLaMA 3.2, gemma3, deepseek-r1 model served locally via Ollama and Gemini in Google Colab, with each response generated through a carefully structured prompt that embedded the current world state, memory context and relevant conversation history.

Each agent in the system was assigned a distinct personality and role prompt, supported by a memory buffer that evolved throughout the simulation. These agents operated within curated narrative scenarios, which progressed turn-by-turn in real time. World events were dynamically introduced into the simulation either through predefined scripts or randomized triggers, allowing the narrative to adapt and shift organically. The agents' responses were recorded and evaluated using a custom logging system, DialogueMetricsLogger, which tracked both performance metrics (such as latency and turn completion) and dialogue-related metrics (including coherence, reactivity and diversity).

The experimental evaluation drew upon five core narrative scenarios. These included a dramatic conflict involving a dragon attack, a morally ambiguous decision between negotiation and combat, a moment of emotional tension marked by the loss of a companion, a setting characterized by environmental unpredictability and a political debate where agents represented competing ideologies. Each scenario featured approximately three agents interacting over the course of ten to twenty dialogue turns. To assess robustness and variability, each scenario was executed ten times.

All simulations used a centralized world state, represented as a narrative sentence or paragraph embedded directly into the system prompt. Memory management was handled

using turn-based updates and longer memory histories were periodically summarized to fit within token limits. The RAG system allowed agents to retrieve contextually relevant information and its impact was evaluated per turn based on semantic similarity between responses and retrieved content.

All tests were executed on a local machine with 32 GB of RAM and GPU acceleration, with the model running entirely offline to ensure reproducibility and full control over generation behavior.

4.2 Quantitative Results

The evaluation results in Table X present a comprehensive comparison of three generative models—LLaMA, Gemma3 and Gemini—under two interaction paradigms ("Static" and "Changing") across six procedurally designed narrative scenarios: Mystery, Conflict, Moral Ambiguity, Betrayal, Surprise and Loss. Each configuration is evaluated along six axes: Coherence, Reactivity, Memory, Diversity, Latency and Retrieval-Augmented Generation (RAG) contribution.

Across nearly all scenarios, Gemma3 in the Changing paradigm consistently outperforms the others in coherence and memory, suggesting its superior capacity to manage continuity and internal consistency over time within dynamic multi-agent dialogues. For instance, in the Moral Ambiguity scenario, Gemma3 (Changing) achieves a coherence score of 0.388 and a memory score of 0.793, notably surpassing LLaMA and Gemini under similar conditions. These results support the hypothesis that dynamic character state tracking in conjunction with agent memory contributes to more narratively consistent outputs.

Conversely, Gemini underperforms across most metrics, with particularly low memory retention and reactivity. For example, in the Mystery scenario, Gemini (Static) attains a memory score of just 0.458 and coherence of 0.189, indicating a lack of narrative depth and a tendency toward disjointed or shallow character responses. This performance suggests that Gemini's architecture may be less suited for sustained dialogue evolution or may suffer from weaker grounding in episodic memory and contextual carryover.

Interestingly, the Reactivity metric presents a more complex picture. While LLaMA in both modes exhibits relatively low reactivity (ex: 0.115 and 0.079 in Mystery), Gemma3's performance under the Changing setting improves significantly, achieving reactivity values up to 0.433 in the Conflict scenario. This improvement reflects the model's heightened sensitivity to external events and changes in the narrative state, likely due to its fine-tuned integration with world state updates and memory buffers.

Diversity scores, measured as confidence intervals over entropy, exhibit considerable variance. LLaMA's high diversity intervals in some scenarios (ex: [0.625, 0.944] in Mystery for Changing) suggest unstable but flexible response generation, whereas Gemma3 shows

tighter and more balanced intervals (ex: [0.570, 0.863]), indicating a better trade-off between creativity and consistency.

In terms of latency, all models show comparable performance with slight variations. LLaMA generally achieves lower average latency, which might reflect its lighter architecture or optimized inference pathways. However, this advantage comes at the cost of coherence and memory, as discussed earlier.

The RAG contribution metric—reflecting the model's ability to utilize retrieved documents effectively—remains moderate across all models, with LLaMA showing the most variation depending on scenario complexity. Gemma3, especially in the Changing mode, maintains more stable and effective RAG utilization, suggesting it integrates retrieved knowledge more fluently within the narrative flow.

Overall, these results validate the utility of agent memory, changing world states and narrative dynamics in enhancing storytelling quality. Models operating in static settings consistently underperform, reinforcing the hypothesis that procedural storytelling benefits from adaptive dialogue models responsive to evolving context and inter-agent history.

Model		**Scenario - Mystery**					
		Coherence	Reactivity	Memory	Diversity	Latency	RAG
LLaMA	Static	0.395	0.115	0.690	(0.299, 0.738)	0.434	0.128
	Changing	0.394	0.079	0.683	(0.625, 0.944)	0.404	0.063
Gemma3	Static	0.406	0.046	0.825	(0.283, 0.610)	0.585	0.060
	Changing	0.436	0.206	0.873	(0.570, 0.863)	0.580	0.105
Gemini	Static	0.189	0.066	0.458	(0.277, 0.662)	0.653	0.047
	Changing	0.243	0.239	0.398	(0.298, 0.715)	0.664	0.071

Model		**Scenario - Conflict**					
		Coherence	Reactivity	Memory	Diversity	Latency	RAG
LLaMA	Static	0.406	0.117	0.600	(0.489, 0.845)	0.457	0.183
	Changing	0.431	0.063	0.460	(0.570, 0.945)	0.402	0.132
Gemma3	Static	0.396	0.433	0.800	(0.533, 0.833)	0.569	0.027
	Changing	0.398	0.317	0.507	(0.591, 0.901)	0.579	0.021
Gemini	Static	0.272	0.166	0.466	(0.449, 0.773)	0.651	0.072
	Changing	0.266	0.238	0.444	(0.618, 0.943)	0.637	0.084

Model		**Scenario - Moral Ambiguity**					
		Coherence	Reactivity	Memory	Diversity	Latency	RAG
LLaMA	Static	0.411	0.167	0.600	(0.488, 0.906)	0.499	0.039
	Changing	0.421	0.032	0.667	(0.513, 0.911)	0.472	0.032
Gemma3	Static	0.369	0.200	0.866	(0.540, 0.842)	0.596	0.022
	Changing	0.388	0.142	0.793	(0.565, 0.851)	0.579	0.031
Gemini	Static	0.188	0.116	0.500	(0.527, 0.885)	0.676	0.044
	Changing	0.255	0.349	0.507	(0.607, 0.931)	0.647	0.058

Model		Scenario - Betrayal					
		Coherence	Reactivity	Memory	Diversity	Latency	RAG
LLaMA	Static	0.435	0.117	0.667	(0.561, 0.914)	0.430	0.061
	Changing	0.437	0.127	0.667	(0.600, 0.942)	0.406	0.116
Gemma3	Static	0.461	0.066	0.750	(0.597, 0.898)	0.592	0.094
	Changing	0.459	0.206	0.825	(0.641, 0.939)	0.560	0.068
Gemini	Static	0.310	0.033	0.666	(0.594, 0.930)	0.685	0.222
	Changing	0.235	0.365	0.698	(0.586, 0.912)	0.687	0.264

Model		Scenario - Surprise					
		Coherence	Reactivity	Memory	Diversity	Latency	RAG
LLaMA	Static	0.413	0.017	0.633	(0.599, 0.947)	0.432	0.094
	Changing	0.394	0.032	0.0825	(0.579, 0.887)	0.375	0.169
Gemma3	Static	0.376	0.066	0.783	(0.619, 0.889)	0.495	0.138
	Changing	0.402	0.174	0.682	(0.674, 0.936)	0.525	0.095
Gemini	Static	0.227	0.113	0.591	(0.251, 0.625)	0.644	0.101
	Changing	0.248	0.317	0.666	(0.576, 0.925)	0.668	0.211

Model		Scenario - Loss					
		Coherence	Reactivity	Memory	Diversity	Latency	RAG
LLaMA	Static	0.360	0.016	0.483	(0.619, 0.910)	0.386	0.183
	Changing	0.384	0.063	0.619	(0.593, 0.953)	0.459	0.089
Gemma3	Static	0.425	0.050	0.683	(0.650, 0.936)	0.505	0.055
	Changing	0.334	0.158	0.619	(0.629, 0.928)	0.568	0.058
Gemini	Static	0.182	0.133	0.566	(0.513, 0.854)	0.720	0.194
	Changing	0.244	0.365	0.634	(0.541, 0.880)	0.688	0.227

4.3 Qualitative Observations

Throughout the implementation and testing of the multi-agent procedural storytelling system, several qualitative patterns emerged that underscore both the strengths and current limitations of this generative approach.

Agents were generally successful in maintaining coherent dialogue and consistent character behavior when system prompts were well-defined and RAG-retrieved context was relevant. Distinct personalities could be convincingly portrayed, especially when prompt engineering clearly articulated the agent's background, tone and goals. However, this distinctiveness was often lost over longer interactions unless memory reinforcement and prompt clarity were continuously maintained. In scenarios where prompt specificity was lacking, agents tended to drift into generic, interchangeable responses, which diminished the believability of their personas.

World state information, when integrated properly into the prompt, had a clear impact on dialogue content. Agents frequently referenced recent environmental changes or story events, indicating a baseline contextual awareness. Nevertheless, their reactions were predominantly observational rather than initiative-driven. Instead of pushing the story

forward with decisions or consequences, many agents merely commented on events. This passive behavior limited the emergence of meaningful narrative progression and led to scenes that felt static or redundant without external intervention.

The evolution of dramatic tension was another area where the system showed mixed results. Tension occasionally emerged, particularly in scenarios involving betrayal or moral ambiguity, where character conflict was more naturally embedded in the prompt design. However, in many other cases, the dialogue plateaued rather than escalating, as agents lacked internal narrative objectives or conflict-resolution mechanisms. Without a goal-based behavior model or event-driven progression logic, conversations often stalled or cycled around the same themes.

Instances of repetition and logical drift were also observed. Agents sometimes reused phrases or emotional states when not exposed to new stimuli and in more complex scenarios, some produced illogical or contradictory responses, particularly when memory buffers became overloaded or retrieval produced irrelevant content. These issues highlighted both the benefits and fragility of the memory and RAG components—effective when well-configured, but vulnerable to derailment from weak context or prompt confusion.

Lastly, system latency emerged as a significant experiential barrier. The cumulative delay from memory retrieval, prompt construction, RAG searching and model inference contributed to sluggish dialogue pacing. In a real-time interactive setting such as a game or live simulation, such latency would likely disrupt immersion and responsiveness, especially when scaled to multiple concurrently interacting agents.

In summary, while the system demonstrated encouraging capabilities in maintaining agent consistency and recognizing world state, it fell short in generating proactive narrative behavior and sustaining dramatic momentum. Future iterations would benefit from the addition of structured narrative planning, tighter integration of goal-driven logic and optimized performance to reduce latency. The introduction of fine-tuned models specifically adapted for dialogue generation and storytelling may also improve coherence, diversity and narrative control in longer, dynamic scenarios.

4.4 Conclusion

The system presented in this research leverages pre trained language models, enhanced by carefully structured prompts and simulated memory mechanisms, to support believable and adaptive narrative agents. While no training was performed at the model weight level, significant system level customization enabled responsive, coherent and goal driven behavior within a procedurally evolving narrative.

This modular and extensible architecture lays the groundwork for future expansions in memory persistence, knowledge retrieval, emotional reasoning and character specific fine tuning. As such, it offers a flexible foundation for generative storytelling environments, whether as research platforms, interactive fiction systems or narrative engines for games and simulations.

While the use of generative AI and multi-agent systems in procedural storytelling demonstrates significant potential, this approach also presents several limitations that must be addressed. One of the key findings is the necessity of designing a structured system around the agents—one that constrains possible actions and guides responses within a narrative framework. Without such constraints, agent behavior becomes too open-ended, often leading to responses that, while contextually appropriate, fail to meaningfully advance the storyline.

Moreover, a more sophisticated and tightly integrated Retrieval-Augmented Generation (RAG) setup is essential to enable agents to interact more dynamically with the evolving world state. The current implementation allows agents to react to external events, but their contributions often remain observational or descriptive rather than directive, resulting in a narrative that lacks momentum or progression.

Another challenge is the latency introduced by multi-stage processing pipelines, which can become a critical issue in real-time applications such as interactive games. The delay in generating agent responses, combined with the computational load required for inference and retrieval, poses a significant barrier to smooth and immersive storytelling experiences.

Additionally, fine-tuning language models specifically for dialogue and storytelling tasks could improve both coherence and responsiveness. Tailoring models to the unique structure and flow of narrative discourse may allow agents to contribute more meaningfully to plot development and character interaction, reducing generic responses and enhancing immersion.

Addressing these issues will be vital for building more coherent, responsive and narratively rich systems capable of supporting complex procedural narratives in interactive environments.

Bibliography

[1] John R. Anderson, Michael Matessa, and Christian Lebiere. "ACT-R: a theory of higher level cognition and its relation to visual attention." In: *Hum.-Comput. Interact.* 12.4 (Dec. 1997), pp. 439–462. ISSN: 0737-0024. DOI: 10.1207/s15327051hci1204_5. URL: https://doi.org/10.1207/s15327051hci1204_5.

[2] Anthropic. *Introducing the next generation of Claude*. 2024. URL: https://www.anthropic.com/news/claude-3-family.

[3] Anthropic. *Model Card and Evaluations for Claude Models*. 2023. URL: https://www.anthropic.com/news/claude-2.

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. "Language Models are Few-Shot Learners." In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[5] James Cartlidge. "Interpreting Dwarf Fortress: Finitude, Absurdity, and Narrative." In: *Games and Culture* 19.2 (2024), pp. 218–236. DOI: 10.1177/15554120231162418. eprint: https://doi.org/10.1177/15554120231162418. URL: https://doi.org/10.1177/15554120231162418.

[6] Harrison Chase. *LangChain*. Oct. 2022. URL: https://github.com/langchain-ai/langchain.

[7] Edward Collins, Nikolai Rozanov, and Bingbing Zhang. *LIDA: Lightweight Interactive Dialogue Annotator*. 2019. arXiv: 1911.01599 [cs.CL]. URL: https://arxiv.org/abs/1911.01599.

[8] DeepSeek-AI et al. *DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning*. 2025. DOI: https://doi.org/10.48550/arXiv.2501.12948. arXiv: 2501.12948 [cs.CL]. URL: https://arxiv.org/pdf/2501.12948.

[9] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. *The Faiss library*. 2025. arXiv: 2401.08281 [cs.LG]. URL: https://arxiv.org/abs/2401.08281.

[10] Gemini Team Google. "Gemini: A Family of Highly Capable Multimodal Models." In: *arXiv preprint arXiv:2312.11805* (2023).

[11] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML]. URL: https://arxiv.org/abs/1406.2661.

[12] John E. Laird. *The Soar Cognitive Architecture*. The MIT Press, 2012. ISBN: 0262122960.

[13] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*. 2021. arXiv: 2005.11401 [cs.CL]. URL: https://arxiv.org/abs/2005.11401.

[14] Yitong Li, Zhe Gan, Yelong Shen, Jingjing Liu, Yu Cheng, Yuexin Wu, Lawrence Carin, David Carlson, and Jianfeng Gao. *StoryGAN: A Sequential Conditional GAN for Story Visualization*. 2019. arXiv: 1812.02784 [cs.CV]. URL: https://arxiv.org/abs/1812.02784.

[15] Michael Mateas and Andrew Stern. "Façade: An Experiment in Building a Fully-Realized Interactive Drama." In: *Game Developers Conference (GDC'03)*. 2003. URL: http://www.cp.eng.chula.ac.th/~vishnu/gameResearch/story_november_2005/MateasSternGDC03.pdf.

[16] Meta AI. *Introducing Meta Llama 3: The most capable openly available LLM to date*. 2024. URL: https://ai.meta.com/blog/meta-llama-3.

[17] OpenAI. "GPT-4 Technical Report." In: *arXiv preprint arXiv:2303.08774* (2023).

[18] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. "Training language models to follow instructions with human feedback." In: *arXiv preprint arXiv:2203.02155* (2022).

[19] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. "Improving language understanding by generative pre-training." In: (2018).

[20] Hannah Rashkin, Asli Celikyilmaz, Yejin Choi, and Jianfeng Gao. "PlotMachines: Outline-Conditioned Generation with Dynamic Plot State Tracking." In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Ed. by Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu. Online: Association for Computational Linguistics, Nov. 2020, pp. 4274–4295. DOI: 10.18653/v1/2020.emnlp-main.349. URL: https://aclanthology.org/2020.emnlp-main.349/.

[21] Significant Gravitas. *AutoGPT*. URL: https://github.com/Significant-Gravitas/AutoGPT.

[22] Marnix Suilen, Thiago D. Simão, David Parker, and Nils Jansen. *Robust Anytime Learning of Markov Decision Processes*. 2023. arXiv: 2205.15827 [cs.AI]. URL: https://arxiv.org/abs/2205.15827.

[23] AI Dungeon team. *AI Dungeon*. 2019. URL: https://ai-dungeon.com.

[24] Gemma Team et al. *Gemma: Open Models Based on Gemini Research and Technology*. 2024. arXiv: 2403.08295 [cs.CL]. URL: https://arxiv.org/abs/2403.08295.

[25] Rogue team. *Rogue*. 1984. URL: https://archive.org/details/RogueTheAdventureGameV1.11984MichaelC.ToyKennethC.R.C.ArnoldAdventureRolePlayingRPG.

[26] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. "LLaMA: Open and Efficient Foundation Language Models." In: *arXiv preprint arXiv:2302.13971* (2023).

[27] Hugo Touvron et al. "Llama 2: Open Foundation and Fine-Tuned Chat Models." In: *arXiv preprint arXiv:2307.09288* (2023).

[28] Christian Bakke Vennerød, Adrian Kjærran, and Erling Stray Bugge. *Long Short-term Memory RNN*. 2021. arXiv: 2105.06756 [cs.LG]. URL: https://arxiv.org/abs/2105.06756.

[29] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*. 2023. arXiv: 2201.11903 [cs.CL]. URL: https://arxiv.org/abs/2201.11903.

[30] J. Weizenbaum. "ELIZA—A Computer Program for the Study of Natural Language Communication between Man and Machine." In: *Communications of the ACM* 4 (1966), pp. 36–45. DOI: 10.1145/365153.365168. URL: https://doi.org/10.1145/365153.365168.

[31] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. *ReAct: Synergizing Reasoning and Acting in Language Models*. 2023. arXiv: 2210.03629 [cs.CL]. URL: https://arxiv.org/abs/2210.03629.

[32] Gokul Yenduri, Ramalingam M, Chemmalar Selvi G, Supriya Y, Gautam Srivastava, Praveen Kumar Reddy Maddikunta, Deepti Raj G, Rutvij H Jhaveri, Prabadevi B, Weizheng Wang, Athanasios V. Vasilakos, and Thippa Reddy Gadekallu. *Generative Pre-trained Transformer: A Comprehensive Review on Enabling Technologies, Potential Applications, Emerging Challenges, and Future Directions*. 2023. arXiv: 2305.10435 [cs.CL]. URL: https://arxiv.org/abs/2305.10435.