Andries Jaeken
Ihno van de Sande
2TI/3AI

Technical Report – Advanced AI: Dungeon Master

Introduction:

Problem Statement:

Interactive storytelling has always relied heavily on human creativity, limiting scalability and requiring manual effort. Our project addresses the challenge of automating a Dungeon Master (DM) experience using artificial intelligence to dynamically generate narratives, control world logic, and interact meaningfully with players in real time.

Motivation:

We both share an interest in D&D and gaming, so we thought choosing this subject for the project would be interesting and fun at the same time. We wanted to create an AI that knows the player’s stats and equipment and can change these accordingly.

Approach:

We developed a Python-based game that uses Gemini-2.0-flash to handle narrative generation, while maintaining structured game logic for stats, equipment, and combat. The AI generates story outputs and game state updates in JSON format, allowing for consistent interaction between the player and the game world.

Link to GitHub repository:

https://github.com/ihno999/advanced_ai_semester_project

Data:

Description:

Our "data" consists of dictionaries for items, equipment slots, character stats, and spell definitions. Instead of traditional datasets, we built rule-based systems to constrain and guide the model’s behaviour.

Preprocessing:

We created:

- A stat-boosting item dictionary (item_stats.py)
- A structured spell list with mana costs and effects (magic_spells.py)
- A file with global variables used throughout the application (globals_variables.py)

Challenges:

The biggest challenge was preventing hallucinations from the LLM, such as it inventing non-existent items or misformatting JSON. This required iterative prompt engineering and runtime validation logic to catch and reject bad responses.

Another challenge was the output display. At some stages during the development, the application displayed multiple times the current player stats and story so far or the display was incorrect. The origin of this challenge were multiple reasons, for example: there was a circular loop because the print was called multiple times, or we called a function at the wrong time causing the output display to be incorrect.

We were also challenged with finding an AI LLM that was usable and were the API end point works with the API KEY. We tried multiple LLM's to find one that was usable and also free to use. We experimented with various LLMs such as ChatGPT and OpenAI's models. Eventually, we selected the Gemini-2.0-flash API.

Model & Methods

Techniques Used:

- **Gemini-2.0-flash** for natural language generation
- **Prompt Engineering** to embed game logic (stat calculations, JSON formatting, valid equipment slots)
- **Stat System** using RPG mechanics (e.g., strength affects attack, defense reduces damage, endurance influences stamina)

Key Logic Features:

- Damage = base_damage + (strength × 0.5)
- Damage Taken = incoming × (1 - defense × 0.03)
- JSON state updates handled via custom <META>{...}</META> tags.
- Equipment limited to 8 slots, all validated against item_stats.py

Results & Evaluation:

Performance:

- Generated coherent, interactive narratives 90–95% of the time.
- Successfully maintained a consistent game state across dozens of interactions.
- High engagement due to dynamic responses and meaningful stat impact

Insights and key findings:

- Strict JSON formatting reduced AI freedom but increased system stability
- Prompt tuning was essential for balancing creativity and control.

Contributions:

What we did:

Most of the time we were working together in a discord call, but these are some of the things each of us did.

Andries:

- Added items that provide stat boosts.
- Introduced magical spells with mana costs.
- Updated game configuration to clarify narrative control and player limitations.
- Implemented game logic for dealing and taking damage.

Ihno:

- Added new stats: mana, stamina, and magic.
- Enabled player name input.
- Implemented game difficulty selection.
- Introduced a levelling system.

Samen:

- Added equipment interactions handled by the AI/LLM.
- Implemented save, load, and delete game functionalities.
- Added functionality to print the current game state.

What we found online:

- We identified and selected a suitable LLM for the project.
- How to use META tags with JSON

Use of GenAI:

- Generated spells, item names, and stat descriptions
- Refined our prompt.
- Occasionally debugged issues or brainstormed new mechanics with GenAI assistance.

Challenges & Future Work:

Challenges:

- Ensuring Gemini output was always JSON-compliant.
- Preventing cheating or unrealistic actions (For example: "I win instantly", "I gain 50 levels", "I shoot laser beams")
- Avoiding logic loops or repeated narrative patterns
- The AI uses only predefined spells and items, modifies stats like health and stamina, and fully integrates these into the interaction.

Future Improvements:

- Add turn-based combat with enemy AI.
- Expand spellcasting system with cooldowns.
- Implement multiplayer support.
- Integrate visuals or map-based exploration.
- Add more magic spells.
- Add more items.
- Application can have multiple save files.
- Implement skill trees or character classes.
- Implement achievements or in-game challenges.