

DM-AI: un director de juego multiagente basado en modelos de lenguaje para juegos de rol narrativos

Salvador Baez, Juan Manuel Fernández

Universidad Nacional de Luján, Buenos Aires, Argentina
totitobaez1011@gmail.com, jmfernandez@unlu.edu.ar

Resumen DM-AI es un director de juego automatizado para juegos de rol narrativos inspirados en Dungeons and Dragons. Los sistemas de narración interactiva basados en modelos de lenguaje ofrecen libertad de acción, pero no distinguen las acciones que requieren resolución por reglas ni preservan un estado consistente del mundo. DM-AI descompone la dirección de la partida en cinco agentes especializados —arbitraje de acciones, actualización de estado, diálogo de personajes, narración y verificación de finales— que se comunican mediante salidas estructuradas con esquemas de valores acotados, y resuelve las acciones con tiradas de d20 cuya dificultad propone el modelo dentro de un conjunto discreto y el sistema revalida. El prototipo se ejecuta sobre modelos de lenguaje locales de escala reducida. Su funcionamiento sobre una campaña de tres escenarios muestra que la arquitectura sostiene, en la mayoría de los casos, la coherencia del estado y la aplicación de las reglas, y expone una tensión de diseño: los mecanismos incorporados para contener las alucinaciones del modelo pueden reintroducir parte de la rigidez que su uso buscaba superar, al punto de rechazar acciones creativas pero válidas.

Keywords: Modelos de lenguaje · Sistemas multiagente · Narración interactiva · Salida estructurada

1. Introducción

Los juegos de rol, como Dungeons and Dragons (DnD), ofrecen una experiencia narrativa caracterizada por la libertad de acción del jugador [9]. En los últimos años, diversos videojuegos han intentado trasladar esta experiencia al ámbito digital. Títulos como Baldur's Gate 3¹ incorporan muchas de las mecánicas de DnD, aunque continúan limitados por un conjunto finito de acciones y caminos predefinidos, lo que restringe la libertad propia de una sesión dirigida por un Director de Juego (DM). Los modelos de lenguaje de gran escala (LLM) [1] permiten superar esa rigidez, pero un director de juego automatizado debe además aplicar reglas y sostener un estado consistente del mundo.

¹ <https://baldursgate3.game>

Este trabajo presenta DM-AI, un director de juego basado en modelos de lenguaje que administra el estado de la partida, interpreta las acciones del jugador y resuelve sus consecuencias mediante un sistema de reglas con tiradas de d20. El prototipo descompone la dirección en agentes especializados que se comunican mediante salidas estructuradas y se ejecuta sobre modelos locales de escala reducida, aunque la arquitectura no depende de un modelo específico: cada componente configura el suyo de manera independiente.

Las contribuciones del trabajo son: (1) una arquitectura multiagente que separa el arbitraje de acciones, la actualización de estado, el diálogo de personajes, la narración y la verificación de finales; (2) un mecanismo de resolución donde el modelo propone la dificultad dentro de un conjunto discreto de valores y el sistema la revalida antes de aplicarla; y (3) la documentación de una tensión de diseño entre contener las alucinaciones del modelo y preservar la libertad de acción del jugador.

2. Antecedentes

Los juegos de rol de mesa se consolidaron como dominio de estudio en procesamiento del lenguaje natural por combinar diálogo abierto con un estado de juego verificable: dirigir una partida se formula como la tarea doble de generar el turno de diálogo y predecir el estado del juego [2], y FIREBALL mostró que condicionar la generación en un estado estructurado mejora la calidad narrativa, tanto en métricas automáticas como en juicio humano [10].

Entre los sistemas existentes, plataformas como FableAI² generan historias dinámicas a partir de las acciones del usuario, y CALYPSO asiste al director humano generando texto y sugerencias que él cura antes de presentarlas a los jugadores [11]. En ambos casos, la resolución de acciones mediante reglas y el mantenimiento de un estado persistente quedan fuera del sistema: en el primero porque no se contemplan, en el segundo porque permanecen a cargo del DM humano. Automatizar el rol por completo habilita el juego en solitario, pero traslada al sistema la responsabilidad de verificar lo que el modelo produce.

La tensión entre respetar las reglas y preservar la libertad del jugador ya fue reportada en este dominio: los directores de juego basados en LLM tienden a ser excesivamente permisivos, produciendo narrativas inconsistentes, o excesivamente rígidos, restringiendo la agencia del jugador [4]. DM-AI aborda ese compromiso mediante una arquitectura multiagente con estado explícito, resolución por reglas y validación externa al modelo, y documenta dónde queda el punto de equilibrio resultante.

3. Arquitectura del sistema

La dirección de una partida combina tareas heterogéneas: interpretar la intención del jugador, aplicar reglas, mantener el estado del mundo y producir

² <https://fableai.app/>

texto narrativo. DM-AI las separa en agentes especializados —coordinados por un motor de control central— de modo que cada uno resuelva una responsabilidad acotada y comunique su salida en un formato verificable, en lugar de delegar el conjunto a un único *prompt*.

Como se observa en la Figura 1, el flujo comienza cuando el jugador ingresa una acción, que un agente **Árbitro** analiza para determinar si es válida y si requiere resolución mediante una tirada de d20.

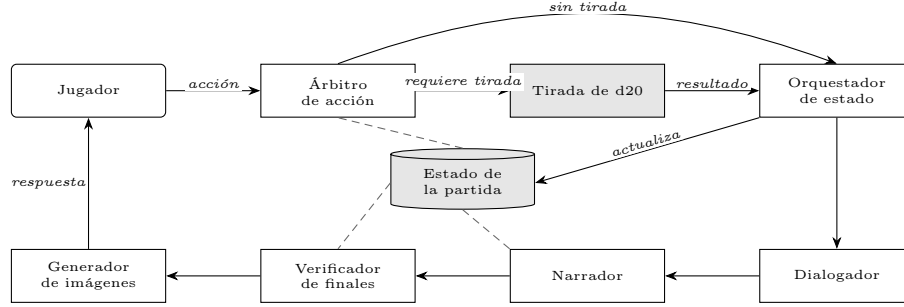


Figura 1: Arquitectura de DM-AI. Rectángulos blancos: agentes que invocan al modelo; sombreados: componentes resueltos en código; líneas punteadas: consulta del estado.

Si la requiere, el resultado es procesado por un **Orquestador de Estado**, encargado de actualizar la información del mundo. Cuando corresponde, un **Dialogador** genera las intervenciones de los personajes no jugadores, mientras que el **Narrador** produce la descripción de la escena considerando el nuevo estado. Un **Verificador de Finales** determina si se alcanzó una condición de cierre, y un generador de imágenes ilustra la escena antes de presentar el resultado al jugador.

La resolución de acciones sigue el mecanismo de DnD [9]: un dado de veinte caras se compara contra un umbral de dificultad. El reparto de responsabilidades es deliberado: el Árbitro propone la dificultad, pero restringido a un conjunto discreto de cinco valores que van de trivial a casi imposible, mientras que la tirada y su comparación se ejecutan en código, fuera del modelo. El sistema revalida el valor propuesto: si no pertenece al conjunto, lo sustituye por el intermedio.

Los agentes se comunican mediante llamadas a herramientas con esquemas declarados, y los esquemas acotan los valores admisibles: el Orquestador no puede introducir una ubicación inexistente, porque el campo correspondiente enumera los escenarios válidos, ni un estado de salud arbitrario, porque solo admite ausencia de cambios, daño o curación. Dado que restringir el formato de salida puede degradar el desempeño en tareas de razonamiento [7], cada esquema incluye como primer campo un texto libre obligatorio donde el agente explicita su razonamiento antes de comprometerse con el veredicto, siguiendo el principio de las cadenas de pensamiento [8].

El estado de la partida se mantiene en una estructura de esquema fijo —ubicación, personajes presentes, estado de cada personaje, inventario y final

alcanzado— que se serializa e inyecta en el contexto de cada agente. La memoria narrativa, en cambio, es una ventana deslizante de los últimos eventos y decisiones, acotada para no saturar el contexto disponible.

Los agentes comparten el modelo subyacente pero no su configuración de muestreo: la narración se genera con temperatura alta y las llamadas estructuradas con temperatura baja, dado que la temperatura afecta de manera distinta la resolución de problemas [6] y la generación creativa [5].

4. Implementación y funcionamiento

4.1. Entorno de ejecución

El prototipo ejecuta los modelos de lenguaje de forma local mediante Ollama, sin dependencia de servicios externos para la generación narrativa ni para la lógica del juego³. Los modelos de texto y de imagen se configuran por separado, lo que independiza la arquitectura del modelo subyacente. La implementación evaluada utilizó Llama 3.1 de 8 mil millones de parámetros [3] para los agentes textuales, un modelo de escala reducida cuyas limitaciones condicionan las observaciones posteriores.

Dos dificultades surgieron durante el desarrollo, ambas derivadas de la brecha entre el formato que el sistema exige y el que el modelo produce. La primera fue obtener respuestas JSON válidas, resuelta con un parser que extrae la estructura del texto circundante y repara errores sintácticos frecuentes. La segunda fue la estabilidad de las invocaciones al modelo, para la que se incorporó un mecanismo de reintentos con retroceso exponencial.

4.2. Comportamiento observado

La campaña implementada consta de tres escenarios encadenados, cinco personajes y cinco finales alcanzables. Sobre ella se recorrió el conjunto de funcionalidades del sistema, desde la narración inicial hasta el desbloqueo de los finales.

La Figura 2 muestra un recorrido representativo: ante la escena de la sala custodiada por dos goblins, el jugador intenta intimidarlos; el Árbitro clasifica la acción como válida y sujeta a tirada; el sistema resuelve el d20 contra el umbral propuesto y, ante un resultado exitoso, el Narrador produce una descripción consistente con el desenlace.

El sistema aplicó las reglas y sostuvo la coherencia del estado en la mayoría del recorrido: los objetos recogidos persistieron en el inventario, los desplazamientos respetaron las conexiones entre escenarios y los finales se activaron al

³ Python 3; interfaz en Pygame (<https://www.pygame.org>); imágenes mediante Hugging Face (<https://huggingface.co>) o Gemini (<https://ai.google.dev>); ejecución local con Ollama (<https://ollama.com>). Contexto de 8192 tokens; dificultades admitidas {0,5,10,15,20}, con 0 para acciones sin tirada; memoria narrativa de los últimos cinco eventos y cinco decisiones. Código fuente: https://github.com/Baisa54/DM_ai.



Figura 2: Resolución de una acción mediante tirada de d20: (a) escena generada por el sistema para la sala custodiada por los goblins; (b) tirada contra el umbral propuesto por el Árbitro.

cumplirse sus condiciones. La coherencia no fue uniforme: los diálogos presentaron inconsistencias menores respecto del estado vigente, y en algunos episodios el modelo devolvió texto conversacional donde el esquema exigía una llamada a herramienta, falla absorbida por los reintentos pero indicativa de la fragilidad del acoplamiento entre generación y formato.

El comportamiento más relevante para el diseño se observó en el Árbitro. Las directivas incorporadas para impedir que el jugador invocara objetos o criaturas ausentes lo llevaron a rechazar también acciones creativas y lógicamente posibles, calificándolas de imposibles. Cada excepción agregada para recuperar esas acciones —permitir el desplazamiento descrito en términos genéricos, admitir el abandono de la misión— amplió la superficie de acciones aceptadas y, con ella, la de acciones indebidamente aceptadas.

5. Conclusiones y trabajo futuro

Se desarrolló un director de juego multiagente basado en modelos de lenguaje que supera la rigidez de los sistemas de acciones predefinidas y sostiene un estado de juego coherente mediante salidas estructuradas y resolución por reglas. Su construcción enfrentó los desafíos propios de operar modelos locales de escala reducida y expuso una tensión de diseño ya reportada en la literatura [4]: contener las alucinaciones del modelo reintroduce parte de la rigidez que su uso buscaba evitar. El punto de equilibrio entre control y libertad de acción no constituye un óptimo sino una decisión de diseño, que este prototipo resolvió a favor de la jugabilidad y cuya ubicación depende de la capacidad del modelo empleado. Dado que los modelos pueden sustituirse por componente sin modificar la lógica del sistema, como trabajo futuro se propone evaluar la arquitectura con modelos de mayor capacidad, para determinar si ese punto de equilibrio se desplaza; incorporar memoria episódica mediante recuperación aumentada; y extender el sistema de combate para aproximarlos al de DnD.

Referencias

1. Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., et al.: Language models are few-shot learners. *Advances in Neural Information Processing Systems* **33**, 1877–1901 (2020)
2. Callison-Burch, C., Tomar, G.S., Martin, L.J., Ippolito, D., Bailis, S., Reitter, D.: Dungeons and Dragons as a dialog challenge for artificial intelligence. In: *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. pp. 9379–9393. Association for Computational Linguistics (2022). <https://doi.org/10.18653/v1/2022.emnlp-main.637>
3. Grattafiori, A., Dubey, A., Jauhri, A., et al.: The Llama 3 herd of models (2024), arXiv:2407.21783
4. Jørgensen, N.H., Tharmabalan, S., Aslan, I., Hansen, N.B., Merritt, T.: Guiding, not railroading: design and evaluation of a multi-agent system for narrative redirection in role-playing games. In: *Proceedings of the 31st International Conference on Intelligent User Interfaces (IUI '26)*. pp. 1551–1601. ACM, New York (2026). <https://doi.org/10.1145/3742413.3789218>
5. Peepkorn, M., Kouwenhoven, T., Brown, D., Jordanous, A.: Is temperature the creativity parameter of large language models? In: *Proceedings of the 15th International Conference on Computational Creativity (ICCC'24)*. pp. 226–235. Association for Computational Creativity (2024)
6. Renze, M.: The effect of sampling temperature on problem solving in large language models. In: *Findings of the Association for Computational Linguistics: EMNLP 2024*. pp. 7346–7356. Association for Computational Linguistics (2024). <https://doi.org/10.18653/v1/2024.findings-emnlp.432>
7. Tam, Z.R., Wu, C.K., Tsai, Y.L., Lin, C.Y., Lee, H.y., Chen, Y.N.: Let me speak freely? a study on the impact of format restrictions on large language model performance. In: *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*. pp. 1218–1236. Association for Computational Linguistics (2024)
8. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E.H., Le, Q.V., Zhou, D.: Chain-of-thought prompting elicits reasoning in large language models. In: *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)*. pp. 24824–24837 (2022)
9. *Wizards of the Coast: Player's Handbook*. Wizards of the Coast, 2024 edn. (2024)
10. Zhu, A., Aggarwal, K., Feng, A., Martin, L.J., Callison-Burch, C.: FIREBALL: a dataset of Dungeons and Dragons actual-play with structured game state information. In: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. pp. 4171–4193. Association for Computational Linguistics (2023). <https://doi.org/10.18653/v1/2023.acl-long.229>
11. Zhu, A., Martin, L.J., Head, A., Callison-Burch, C.: CALYPSO: LLMs as dungeon masters' assistants. In: *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*. vol. 19, pp. 380–390 (2023). <https://doi.org/10.1609/aiide.v19i1.27534>