Universidad de Concepción
Facultad de Ingeniería
Ingeniería Civil Informática

Integrando ABMs a LLMs para mejorar su coherencia espacial y temporal

Memoria de Título

Autor:	**Paulo Ruggero Antonio Alarcón Quevedo**
Profesor Patrocinante:	**Pedro Pablo Pinacho Davidson**
Profesor Co-patrocinante:	**Fernando Andree Tercero Gutiérrez Gómez**

Chile, Región del Biobío, Concepción
30 de octubre de 2025

Este trabajo fue parcialmente financiado por Proyecto ANID Fondecyt N°11230359, “AN IMMUNE INSPIRED MODEL OF INTRUSION PREVENTION SYSTEM (IPS) FOR COLLABORATIVE AND DISTRIBUTED ENVIROMENTS"

El autor, Paulo Ruggero Antonio Alarcón Quevedo, declara haber utilizado herramientas de inteligencia artificial generativa, como ChatGPT, de manera ética y responsable para apoyar la realización de este trabajo. A continuación, se detalla el uso otorgado:

- **Redacción, Estructuración, Mejora del Texto Corrección Ortográfica:** Uso de IA para reescribir ideas originales, organizar secciones, mejorar la coherencia, claridad, estilo, corregir errores gramaticales y ortográficos.
- **Asesoría Técnica o Conceptual:** Consulta sobre conceptos técnicos complejos. La información proporcionada por la IA ha sido revisada, contrastada y validada con fuentes académicas o científicas adecuadas para asegurar su precisión y pertinencia.
- **Otros usos:** El autor declara que todo contenido generado o asistido por IA ha sido revisado, adaptado y validado para asegurar su originalidad y pertinencia. Él es el único responsable del trabajo presentado y se compromete a que las fuentes utilizadas sean debidamente citadas.

Índice

1. Introducción

El desarrollo de grandes modelos de lenguaje (LLMs) ha permitido avances significativos en la generación de texto coherente, fluido y gramaticalmente correcto [1, 2]. Estas capacidades han impulsado el surgimiento de chatbots y agentes conversacionales que pueden interactuar de manera cada vez más natural con los usuarios en contextos variados, desde asistentes virtuales hasta sistemas de soporte especializado [3]. Sin embargo, una limitación persistente de estos agentes radica en su capacidad para mantener coherencia espacial y temporal a lo largo de interacciones extensas [4, 5].

La ausencia de esta coherencia se vuelve problemática en contextos donde se espera que un agente actúe de manera verosímil dentro de un entorno conversacional continuo, como ocurre en simulaciones interactivas, videojuegos narrativos o entornos de entrenamiento virtual [6, 7]. En estos casos, la incapacidad para sostener una narrativa persistente reduce la credibilidad del agente y su utilidad en aplicaciones que dependen de interacciones prolongadas. Esto ocurre porque, la falta de coherencia genera contradicciones en la continuidad de los eventos y en la consistencia del entorno descrito, lo que rompe la ilusión de realismo y mina la confianza del usuario en la fiabilidad del agente [5, 8].

Esta problemática relacionada con la degradación de la coherencia en interacciones prolongadas trasciende el ámbito técnico y alcanza dimensiones éticas y de seguridad. Por ejemplo, OpenAI -organización detrás de ChatGPT-, tras el caso de suicidio de un joven asistido por ChatGPT, ha señalado explícitamente que "nuestras salvaguardias funcionan mejor en intercambios comunes y breves. Con el tiempo, aprendimos que estas salvaguardias a veces se tornan menos confiables en las interacciones largas: a medida que se desarrollan las idas y vueltas, parte del entrenamiento de seguridad del modelo puede debilitarse." [1]. Lo anterior refleja

[1] OpenAI *Helping people when they need it most*, 26 de Agosto 2025 [9]

que, incluso en modelos avanzados, la consistencia tiende a degradarse con conversaciones extensas, evidenciando que la pérdida de coherencia no solo compromete la credibilidad del agente, sino que también puede derivar en consecuencias potencialmente riesgosas para los usuarios más vulnerables.

En este contexto, surge la motivación de este trabajo: explorar un enfoque que permita a los LLMs mantener coherencia espacial y temporal en escenarios interactivos mediante la integración con un entorno simulado. La idea central es que un espacio controlado pueda proporcionar a los modelos una representación persistente del estado del mundo. De este modo, sus respuestas pueden reflejar no solo la interacción inmediata, sino también las consecuencias que dichas interacciones puedan tener en este espacio y en interacciones futuras.

La mejora en la coherencia espacial y temporal implica a su vez un aumento en la credibilidad y utilidad de los agentes conversacionales. Un modelo capaz de mantener consistencia en la representación del entorno puede sostener narrativas prolongadas y verosímiles, lo cual es un aspecto central en ámbitos como simulaciones educativas, videojuegos narrativos o sistemas de entrenamiento virtual. De esta forma, el presente trabajo busca contribuir a superar una de las limitaciones más relevantes de los LLMs actuales, reforzando su potencial de generar experiencias interactivas más realistas y coherentes.

La solución propuesta para enfrentar este desafío es la utilización de entornos basados en *Agent-Based Modeling* (ABM). Este enfoque permite representar sistemas como un conjunto de agentes autónomos que interactúan siguiendo reglas locales, generando dinámicas globales emergentes [10–12]. A diferencia de modelos agregados o de los sistemas multi-agente orientados (MAS) a la coordinación, ABM ofrece un marco idóneo para mantener un estado explícito y persistente del entorno, lo que permite representar de manera controlada la dinámica espacio-temporal de la simulación. Integrar un LLM con un entorno de este tipo proporciona un *estado persistente y verificable*, mitigando las limitaciones observadas en

ventanas de contexto largas [4].

2. Objetivos

Objetivo General

El objetivo de este proyecto es desarrollar un framework que integre un entorno simulado y un LLM para estudiar la coherencia espacio-temporal en agentes conversacionales.

Objetivos Específicos

1. Desarrollar un entorno simulado cuyo estado de mundo sea visible e interpretable para un LLM.
2. Implementar el método para lograr la interacción entre el LLM, un agente conversacional (el cual puede ser otro LLM) y el entorno simulado.
3. Explorar mecanismos para la evaluación del desempeño de los LLMs, considerando métricas relevantes para la coherencia temporal y espacial.
4. Evaluar el desempeño de distintos LLMs en constante interacción con el entorno simulado implementado.

3. Análisis de Riesgos

Este proyecto se enmarca dentro de una investigación exploratoria, es decir, un tipo de investigación orientada a examinar fenómenos poco estudiados y a generar nuevos enfoques más que a validar teorías consolidadas. Debido a esto los riesgos principales se encuentran en el ámbito técnico más que en el ámbito de gestión, principalmente por la potencial inviabilidad de las aproximaciones abordadas. Del mismo modo, al ser una memoria de carácter exploratorio no hay garantías de mejora del estado del arte.

1. **Inviabilidad de una implementación**: Dado el carácter exploratorio del proyecto, existe el riesgo de que alguna de las aproximaciones propuestas para la simulación resulte impracticable, lo que podría llevar a invertir tiempo en un desarrollo que no genere los resultados esperados.

2. **Tiempo invertido en estrategias no óptimas**: La exploración de distintos enfoques conlleva el riesgo de dedicar esfuerzo a metodologías que, en retrospectiva, no sean las más efectivas, lo que podría ralentizar el progreso del proyecto.

Para mitigar los riesgos anteriores, se adoptó un enfoque de trabajo y evaluación iterativos que permite prever la viabilidad de cada estrategia en fases tempranas, evitando prolongar desarrollos que no conduzcan a soluciones viables.

4. Marco Teórico

El presente marco teórico aborda los conceptos fundamentales que sustentan esta investigación. En primer lugar, se describen las características de los Grandes Modelos de Lenguaje (LLMs) y sus limitaciones actuales. Posteriormente, se examinan las distintas dimensiones de coherencia —contextual, temporal y espacial— y los enfoques recientes que incorporan memoria persistente en LLMs. Finalmente, se introduce el enfoque de Modelado Basado en Agentes (ABM), contrastándolo con otras metodologías de simulación y destacando su relevancia como base para este trabajo.

4.1. Grandes Modelos de Lenguaje (LLM)

Los *Grandes Modelos de Lenguaje* (LLM, por sus siglas en inglés) son redes neuronales entrenadas con grandes volúmenes de texto para modelar la probabilidad de aparición de secuencias lingüísticas. Estos modelos se basan en la arquitectura *Transformer* [13], cuyo mecanismo de autoatención permite representar dependencias entre palabras en diferentes posiciones de la secuencia. Gracias a ello, el modelo no solo considera el contexto inmediato, sino que pondera la relevancia de cada elemento de entrada al predecir la siguiente palabra. De esta manera, la secuencia de entrada funciona como guía para identificar la salida más probable en cada paso, lo que hace posible generar respuestas coherentes y contextualmente adecuadas. Algunos ejemplos representativos incluyen *GPT*, *Gemma* o *LLaMA*. Estos modelos han demostrado una alta capacidad para generar texto fluido, realizar razonamiento lógico, traducir idiomas y responder preguntas.

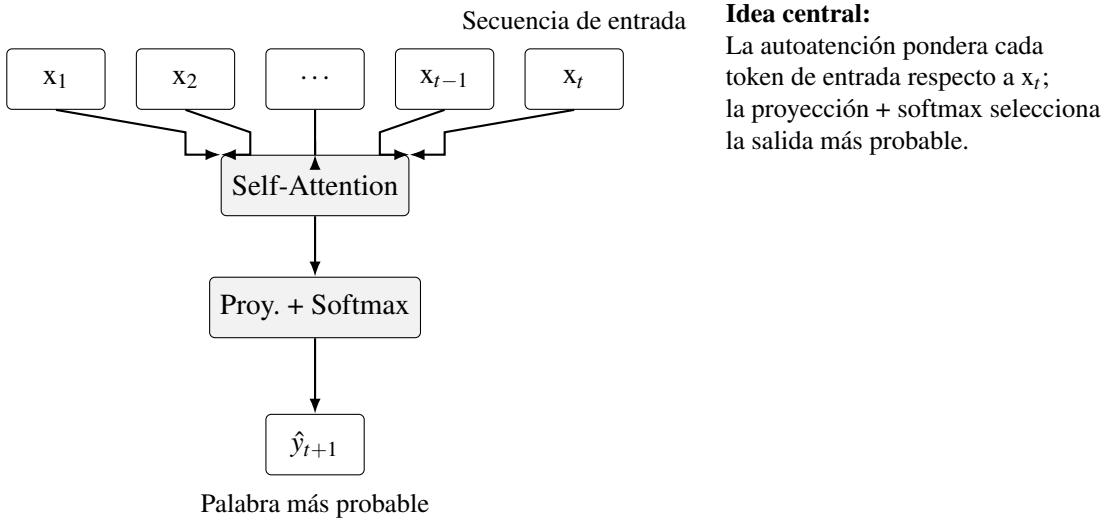


Figura 1: Esquema simplificado del proceso autoregresivo en un Transformer: *self-attention* pondera la secuencia de entrada para construir un contexto y, mediante una proyección seguida de *softmax*, se selecciona la siguiente palabra más probable [13].

4.1.1 Prompts

Un elemento pivotal en la interacción con los LLMs son los *prompts*, es decir, las instrucciones textuales que definen el contexto y guían la respuesta del modelo. La formulación adecuada de un prompt puede modificar drásticamente el rendimiento del modelo en tareas específicas. Este mecanismo permite condicionar la salida hacia objetivos concretos, ya sea respondiendo preguntas, realizando tareas de clasificación o adoptando un rol dentro de una simulación. La formulación adecuada de *prompts* posibilita adaptar un mismo modelo a múltiples aplicaciones sin necesidad de reentrenamiento.

No obstante, la efectividad de un *prompt* depende en gran medida de su diseño, lo que ha dado lugar a la llamada *prompt engineering*, un área activa de investigación dedicada a encontrar estrategias más sistemáticas para lograr resultados coherentes y confiables. En este contexto, han surgido diversas estrategias para generar prompts que sistematizan la manera en que se formulan las instrucciones, permitiendo aprovechar mejor las capacidades de los LLMs y mejorar su cohe-

rencia en tareas complejas. Estas estrategias, llamadas *técnicas de prompting* se describen en la sección siguiente.

4.1.2 Técnicas de prompting

Las técnicas de prompting corresponden a estrategias utilizadas para diseñar las instrucciones que guían la generación de un modelo de lenguaje. Un mismo modelo puede producir respuestas muy distintas según cómo se formule el *prompt*, de modo que su construcción se ha convertido en un aspecto clave para aprovechar las capacidades de los LLMs. Estas técnicas permiten orientar al modelo hacia tareas específicas, mejorar la calidad de las respuestas y, en muchos casos, compensar limitaciones como la falta de memoria persistente o la tendencia a producir salidas inconsistentes.

Se ha demostrado que un diseño cuidadoso de los *prompts* puede mejorar significativamente el razonamiento paso a paso, la recuperación de información relevante y la coherencia en interacciones de varias etapas [14]. Por esta razón, el *prompting* se ha consolidado como un mecanismo práctico y flexible para adaptar modelos generales a contextos de uso concretos, sin necesidad de modificar sus parámetros internos ni realizar procesos de reentrenamiento costosos.

En primer lugar, *Zero-shot Prompting* [1], consiste en formular directamente la tarea deseada en lenguaje natural, sin proporcionar ejemplos previos que guíen la respuesta del modelo. En este enfoque, se asume que el modelo ya adquirió durante su entrenamiento el conocimiento suficiente para interpretar la instrucción y generar una salida adecuada. Por ejemplo, si se requiere que un LLM devuelva un vector de números, basta con indicarle explícitamente: “Devuelve 49 enteros separados por espacios”. La ventaja de este método es su simplicidad y generalidad: el mismo modelo puede adaptarse a una amplia variedad de tareas con solo cambiar la instrucción. No obstante, su limitación principal es que el desempeño depende de qué tan bien el modelo entienda la tarea desde la formulación, lo cual puede ser

inconsistente si la instrucción es ambigua.

Chain-of-Thought (CoT) Prompting [15] es otra técnica que introduce un nivel adicional de control al instruir al modelo para que explicite un razonamiento intermedio antes de producir la respuesta final. En lugar de generar una salida directa, el LLM debe descomponer el problema en pasos lógicos, generando una *cadena de pensamientos* que conecta el enunciado con la conclusión. Este proceso favorece que el modelo explore rutas de razonamiento más estructuradas, lo que ha mostrado mejoras en tareas aritméticas, de lógica simbólica y de planificación. Un ejemplo típico es añadir al *prompt* instrucciones como "razona paso a paso antes de responder", lo que lleva al modelo a describir cómo llega a la respuesta. Su desventaja es que puede aumentar el costo computacional y, en ocasiones, inducir razonamientos redundantes o demasiado largos.

Otra técnica utilizada es *Prompt Chaining* [14], la cual se aplica en escenarios donde la interacción se desarrolla en múltiples turnos. En este caso, el modelo recibe no solo la nueva instrucción, sino también el historial acumulado de interacciones previas, lo que genera una cadena de *prompts* conectados. Así, cada respuesta del modelo incorpora el contexto anterior, manteniendo continuidad narrativa y factual a lo largo de la sesión. Este método resulta útil en simulaciones, juegos o agentes conversacionales, donde el estado del mundo debe conservarse y reflejarse en cada turno. Sin embargo, su eficacia depende de la longitud de la ventana de contexto: cuando el historial se hace demasiado extenso, puede saturar la memoria inmediata del modelo y degradar el desempeño.

Finalmente, la técnica de *Auto-consistency* [16] busca aumentar la fiabilidad de los resultados generados. Su funcionamiento implica pedir al modelo que produzca múltiples trayectorias de razonamiento independientes para un mismo *prompt* y, posteriormente, seleccionar la respuesta más frecuente o más consistente entre ellas. De este modo, se reduce la probabilidad de obtener salidas erróneas causadas por decisiones aleatorias en una única ejecución. Aunque este procedimiento

completo exige generar varias muestras en paralelo, una aplicación parcial consiste en reintentar la consulta cuando el modelo produce resultados inválidos o incoherentes. Incluso este uso limitado aproxima los beneficios de la técnica, ya que introduce redundancia y filtra errores, aumentando la confiabilidad general de las respuestas.

4.1.3 Ventana de contexto

Otro aspecto central en el desempeño de los LLMs es la *ventana de contexto*, que determina la cantidad máxima de *tokens* que el modelo puede procesar simultáneamente como entrada. En el contexto de los LLMs, los *tokens* corresponden a las unidades mínimas de procesamiento en que se fragmenta el texto de entrada y salida. Un *token* puede representar una palabra completa, parte de una palabra o incluso signos de puntuación, dependiendo del esquema de tokenización utilizado.

Esta ventana establece los límites de la memoria inmediata del modelo: todo lo que quede fuera de ella no puede influir en la generación de respuestas. En la práctica, esto significa que el *prompt* —que incluye tanto la instrucción actual como el historial de interacción relevante— debe encajar dentro de la ventana de contexto para que el modelo pueda aprovecharlo al producir su salida.

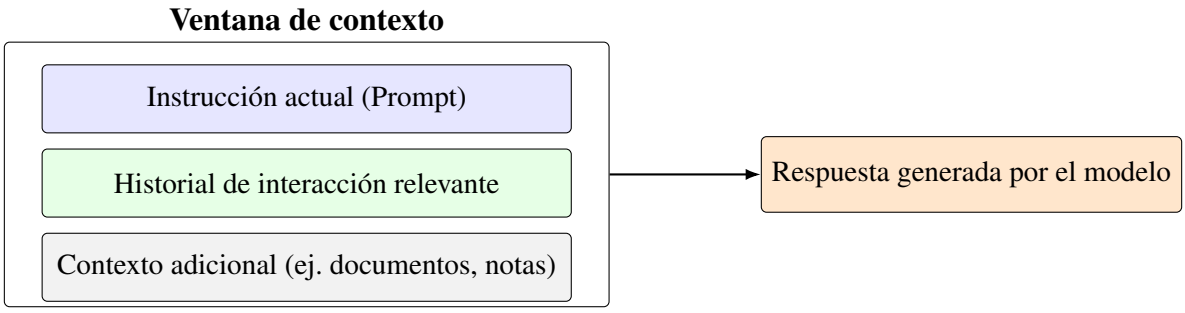


Figura 2: La ventana de contexto contiene el *prompt* (instrucción + historial), que determina la información disponible para el modelo al generar una respuesta. Todo lo que quede fuera de la ventana no influye en la salida.

Aunque en versiones recientes se han expandido desde algunos miles hasta cien-

tos de miles de *tokens*, lo cual representa un avance técnico significativo, esta ampliación no garantiza necesariamente un mejor uso del contexto. Investigaciones han demostrado que los LLMs no siempre emplean de manera eficiente contextos largos [4]. Los modelos tienden a priorizar la información ubicada al inicio y al final de la ventana, mientras que los detalles intermedios suelen degradarse o ser ignorados. Esto limita la capacidad de los modelos para mantener coherencia en interacciones extensas, aun cuando dispongan de un mayor "espacio de memoria".

Ampliar la ventana de contexto aumenta la capacidad teórica de memoria del LLM, pero no resuelve por sí mismo los problemas de coherencia a largo plazo ni de integración estructurada de la información [4, 5]. En respuesta a estas limitaciones, se han propuesto técnicas complementarias que buscan superar lo que una ventana fija no puede garantizar. Por ejemplo, BiTimeBERT [17] incorpora información temporal bidireccional durante el preentrenamiento, lo que mejora el modelado de relaciones secuenciales más allá de los límites de dicha ventana. De forma similar, métodos de decodificación como Self-Consistency [16] apuntan a estabilizar las respuestas al generar y comparar múltiples trayectorias de razonamiento, mitigando parcialmente la pérdida de información intermedia.

El problema de fondo se relaciona con la arquitectura Transformer [13] que sustenta a la mayoría de los LLMs actuales. Estos modelos aprenden de forma autoregresiva o mediante enmascaramiento, y a pesar de su capacidad para producir salidas plausibles y coherentes a corto plazo, muestran limitaciones para integrar información distribuida en secuencias largas [1,2]. En escenarios conversacionales complejos o simulaciones interactivas, esto se traduce en una pérdida progresiva de contexto: las decisiones y descripciones generadas en turnos posteriores no siempre mantienen consistencia con los eventos previos, reduciendo así la credibilidad narrativa del agente [3, 8].

La ventana de contexto representa tanto un avance como una restricción. Su expansión técnica permite manejar entradas de mayor longitud, pero la literatura

señala que la verdadera solución a los problemas de coherencia temporal y espacial requiere mecanismos adicionales. Estos mecanismos incluyen memorias persistentes externas [6, 7], arquitecturas híbridas de integración temporal [17], o técnicas de prompting estructurado [14] que guíen explícitamente la recuperación y el uso de información relevante a lo largo de múltiples interacciones.

4.2. Coherencia Contextual, Espacial y Temporal en LLMs

La coherencia en agentes conversacionales puede entenderse en tres dimensiones complementarias.

La *coherencia contextual* se refiere a la capacidad de un modelo para mantener consistencia temática y factual en el contenido de la conversación. Esto implica que el agente no contradiga información previamente mencionada y que las respuestas sean relevantes respecto al tema en curso. En este marco, la cohesión lingüística, o sea, los mecanismos lingüísticos que enlazan oraciones y párrafos entre sí, actúa como un aspecto complementario, ya que asegura la correcta conexión entre frases y oraciones, reforzando la continuidad del discurso. La falta de coherencia contextual puede llevar a contradicciones internas, pérdida de sentido o cambios abruptos de tópico que afectan la credibilidad del sistema conversacional [3].

La *coherencia espacial* está relacionada con la capacidad de sostener descripciones consistentes del entorno. Incluye mantener la ubicación de objetos, personajes y elementos del espacio, así como las relaciones entre ellos. En simulaciones interactivas o entornos narrativos, este tipo de coherencia es el pilar fundamental para que la representación del "mundo" conserve verosimilitud y el usuario pueda confiar en la continuidad de la experiencia [6].

La *coherencia temporal* corresponde a la habilidad de recordar y actuar en consecuencia a eventos pasados, preservando la continuidad a lo largo del tiempo. Esto significa que las acciones, decisiones o descripciones del modelo deben reflejar la evolución lógica de la interacción. La ausencia de coherencia temporal se

traduce en inconsistencias narrativas, olvidos de información clave o repeticiones innecesarias que reducen la calidad de la interacción prolongada [4, 5].

Estudios recientes destacan la importancia de evaluar y garantizar estas dimensiones en modelos de diálogo, dado que la falta de consistencia factual o narrativa reduce la credibilidad de los agentes conversacionales [5]. En entornos simulados, la necesidad de mantener memoria persistente y un "estado del mundo" explícito se vuelve aún más evidente, ya que la coherencia espacial y temporal es esencial para sostener interacciones prolongadas y verosímiles [6]. A pesar de que los LLMs manejan, en mayor o menor medida, cada una de estas dimensiones, carecen de mecanismos internos robustos para mantener un "estado del mundo" más allá de lo que cabe en su ventana de atención inmediata.

4.2.1 Trabajo Relacionado

Para abordar el problema de la coherencia, han surgido múltiples propuestas que integran diferentes formas de memoria persistente y mecanismos de control del contexto. Estas aproximaciones pueden entenderse en un continuo que va desde soluciones prácticas de ingeniería hasta arquitecturas más sofisticadas orientadas a entornos dinámicos.

En el extremo más simple se encuentran los enfoques basados en recuperación de información, como los sistemas de *Retrieval-Augmented Generation* (RAG) [18] o el uso de archivos externos para almacenar y recuperar datos relevantes. De manera práctica, frameworks como los módulos de memoria de LangChain [19] ofrecen interfaces reutilizables que permiten a un modelo recordar información de interacciones previas. Su principal ventaja es la facilidad de integración, ya que proporcionan una infraestructura estandarizada para dotar a chatbots o asistentes virtuales de memoria externa. Sin embargo, estas soluciones dependen fuertemente de la calidad de los embeddings y suelen ser insuficientes para mantener coherencia en interacciones prolongadas o escenarios complejos.

Estos enfoques constituyen el nivel más básico de memoria externa en LLMs, pues permiten extender la ventana de contexto con información recuperada sin necesidad de modificar la arquitectura del modelo. Sin embargo, al depender de búsquedas sobre bases de datos o historiales de interacción, su alcance se limita a proveer fragmentos relevantes en cada consulta, sin garantizar una narrativa continua ni una integración semántica profunda. Esta limitación ha motivado el desarrollo de arquitecturas más sofisticadas, como MemGPT [7] o Smallville [6], que buscan incorporar mecanismos de memoria persistente capaces de mantener coherencia a lo largo de interacciones prolongadas en entornos dinámicos.

MemGPT [7] introduce mecanismos para alternar entre almacenamiento a corto y largo plazo, reduciendo la carga computacional y preservando información crítica en sesiones largas. Por su parte, el proyecto Smallville [6] demostró cómo agentes impulsados por LLMs pueden interactuar en un entorno simulado mientras acceden a memorias de largo plazo. Esto permitió observar fenómenos emergentes como rutinas, amistades o coordinación colectiva, aunque a costa de altos requerimientos de cómputo y con un control limitado sobre la coherencia en detalles específicos.

En paralelo, algunos trabajos han buscado mejorar la representación de la dimensión temporal a nivel de arquitectura lingüística. Un ejemplo es BiTimeBERT [17], una extensión de BERT que incorpora información temporal bidireccional durante el entrenamiento. Esta propuesta mejora la representación de relaciones cronológicas en set de datos estructurados, aunque su alcance se limita a textos estáticos y no aborda directamente la coherencia en entornos interactivos donde el estado cambia dinámicamente.

Finalmente, también se han explorado técnicas de decodificación orientadas a estabilizar las salidas de los LLMs. El método de *Self-Consistency Decoding* [16] genera múltiples trayectorias de razonamiento y selecciona la más coherente, lo que reduce la variabilidad y los errores en tareas de razonamiento paso a paso.

Si bien esto mejora la coherencia local de las respuestas, no incorpora memoria a largo plazo ni representación estructurada del contexto, por lo que su aplicabilidad en interacciones prolongadas es limitada.

En conjunto, los trabajos revisados evidencian dos grandes líneas de investigación orientadas a mejorar la coherencia en los modelos de lenguaje. Por un lado, las estrategias basadas en *memoria externa*, como MemGPT [7] y RAG [18], buscan extender la capacidad de los modelos mediante la incorporación de mecanismos de almacenamiento y recuperación de información persistente. Por otro lado, los enfoques inspirados en la *simulación*, como Smallville [6], trasladan la representación del contexto a un entorno dinámico en el cual las consecuencias y estados del mundo se derivan de la propia evolución del sistema y no exclusivamente del razonamiento del modelo. Este último enfoque resulta particularmente prometedor, ya que permite evaluar la coherencia de manera explícita y controlada dentro de escenarios interactivos, ofreciendo un marco más robusto para el estudio de la consistencia temporal y espacial en LLMs.

4.3. Modelado Basado en Agentes (ABM)

Agent-Based Modeling (ABM) representa un sistema como un conjunto de agentes autónomos que perciben su entorno y actúan siguiendo reglas locales. De las interacciones de estos agentes emergen patrones globales que no están programados explícitamente [10, 11]. Este enfoque de simulación es útil cuando la heterogeneidad, la interacción local y los efectos espaciales/temporales son relevantes. ABM permite registrar el estado de cada agente (atributos, posición, historial) y del entorno, proporcionando trazabilidad fina sobre cómo micro decisiones producen dinámicas a mayor escala [11, 12].

En comparación con enfoques de simulación agregada, como las ecuaciones diferenciales [20] o los modelos de sistemas dinámicos [21], que describen el comportamiento promedio de un sistema mediante variables continuas, el ABM

trabaja al nivel de cada agente individual. A diferencia de los modelos agregados, donde se asume que los actores son homogéneos y sus interacciones se diluyen en promedios globales, el ABM permite que cada entidad conserve atributos propios y tome decisiones autónomas. Esto implica que los resultados de la simulación no son la consecuencia de ecuaciones deterministas sobre poblaciones completas, sino de trayectorias discretas de agentes que interactúan en el tiempo y el espacio. Mientras los modelos agregados carecen de memoria explícita de eventos y de representación espacial detallada, el ABM facilita rastrear cómo las reglas locales generan patrones colectivos observables. Por esta razón, la literatura lo reconoce como un marco más fiel y auditable para estudiar la emergencia de comportamientos complejos a partir de condiciones simples [10, 11].

Un enfoque que suele confundirse con ABM son los Sistemas Multi-Agente (MAS). Ambos enfoques utilizan múltiples agentes como unidades básicas, lo que genera la impresión de que persiguen los mismos objetivos, aunque en realidad responden a tradiciones y finalidades distintas. A pesar de que ambos operan con múltiples agentes, MAS nace de la IA distribuida y busca resolver problemas de coordinación/planificación entre agentes de software (énfasis ingenieril). Por otro lado ABM se emplea como herramienta de simulación y análisis para estudiar fenómenos emergentes y explorar escenarios contrafactuales [11, 12]. Es decir, MAS se orienta a diseñar soluciones multi-agente; ABM a comprender y explicar dinámicas en sistemas complejos. Esta diferencia de objetivos es uno de los aspectos más relevante al elegir una base metodológica.

Para unificar simulación y LLM con el objetivo de mejorar la coherencia espacial y temporal, ABM ofrece ventajas prácticas documentadas recientemente. Por un lado, entrega un estado persistente y estructurado (posiciones, relaciones, eventos), mitigando las limitaciones de los long prompts y la degradación de información intermedia. Por otro lado, estudios sobre agentes basados en LLM y simulación social muestran que integrar LLMs con memorias o estados explícitos

facilita mantener narrativas consistentes y evaluables a lo largo del tiempo [8, 22]. Además, se ha observado que marcos híbridos LLM + ABM permiten escalar gradualmente la complejidad (más agentes, reglas y atributos) manteniendo control experimental y trazabilidad de causas y efectos [22, 23].

En síntesis, la literatura respalda que ABM constituye un marco metodológico particularmente adecuado para este trabajo. Esto porque permite representar heterogeneidad y dinámicas espacio-temporales a nivel de agente, al mismo tiempo que ofrece un estado explícito y auditable que puede funcionar como memoria persistente para un LLM. Estas características abordan directamente las limitaciones documentadas en el uso de ventanas de contexto largas y en la degradación de información intermedia, al proporcionar una fuente estructurada y verificable de continuidad. Además, a diferencia de los enfoques agregados, que pierden trazabilidad individual, y de los marcos MAS, centrados en problemas de coordinación, ABM entrega un balance entre flexibilidad experimental y control analítico. Por estas razones, se consolida como la base metodológica más robusta para explorar la integración entre simulación y LLMs en la evaluación de coherencia espacial y temporal.

5. Solución Propuesta

En este trabajo se propone combinar un entorno simulado basado en ABM en conjunto con un LLM. El ABM proporciona un estado persistente y controlado del entorno, mientras que el LLM actúa como un observador que interpreta dicho estado y lo traduce en descripciones narrativas en lenguaje natural. Esta integración busca superar las limitaciones de coherencia espacial y temporal de los LLMs, ya que el modelo no depende únicamente de su memoria implícita, sino que dispone de un contexto estructurado que evoluciona paso a paso.

5.1. Motivación de la Arquitectura

La motivación central detrás de la arquitectura propuesta es ofrecer a los LLMs un mecanismo de representación de estado de mundo a través de un entorno simulado. Estudios recientes han mostrado que los modelos tienden a perder información contextual en interacciones largas [4], mientras que enfoques que integran memorias persistentes, han logrado sostener narrativas más estables. Inspirado por estas ideas, se plantea que un ABM puede funcionar como una estructura externa que almacena el estado del mundo y lo actualiza en cada turno. De este modo, el LLM recibe en cada interacción un resumen fiel de la simulación, lo que le permite generar respuestas coherentes con la historia acumulada y con el espacio en el que se desarrolla la interacción.

5.2. Arquitectura Propuesta

La arquitectura diseñada integra tres componentes principales:

- **Simulador ABM**: mantiene la representación estructurada del entorno, aplicando reglas simples de interacción y actualizando el estado global en cada turno. Esto asegura un *ground truth* verificable y reproducible.

- **LLM Observador**: interpreta el estado generado por el simulador y lo comunica en lenguaje natural. Su función principal es transformar la representación estructurada en descripciones comprensibles para el usuario, asumiendo un *rol conversacional definido* orientado a mantener la coherencia espacial de la interacción, pero además puede enriquecer dichas descripciones ya que no emite una respuesta explícita del estado del mundo sino que narra el estado actual de la simulación al usuario, de manera similar a un narrador literario. De esta forma, se evita el uso de lenguaje técnico que puede resultar en una traducción inadecuada del estado al añadir información no relevante para el usuario.

- **Agente conversacional**: interactúa con el sistema tomando decisiones dentro del entorno y recibiendo descripciones del LLM Observador. Esto permite evaluar cómo el modelo mantiene la coherencia en escenarios prolongados. Puede ser un usuario humano o un LLM.

De manera complementaria, se incluye un *módulo de registro*, encargado de almacenar tanto el estado de la simulación como las salidas generadas por el LLM. Este componente asegura la trazabilidad de cada interacción y facilita el posterior análisis experimental.

La Figura 3 ilustra la interacción entre estos componentes. El ABM actualiza el entorno en cada turno, el LLM Observador traduce ese estado a lenguaje natural y el usuario recibe la descripción y toma nuevas decisiones. Este ciclo se repite, creando una dinámica en la que la coherencia espacial y temporal del agente puede ser evaluada de forma sistemática.

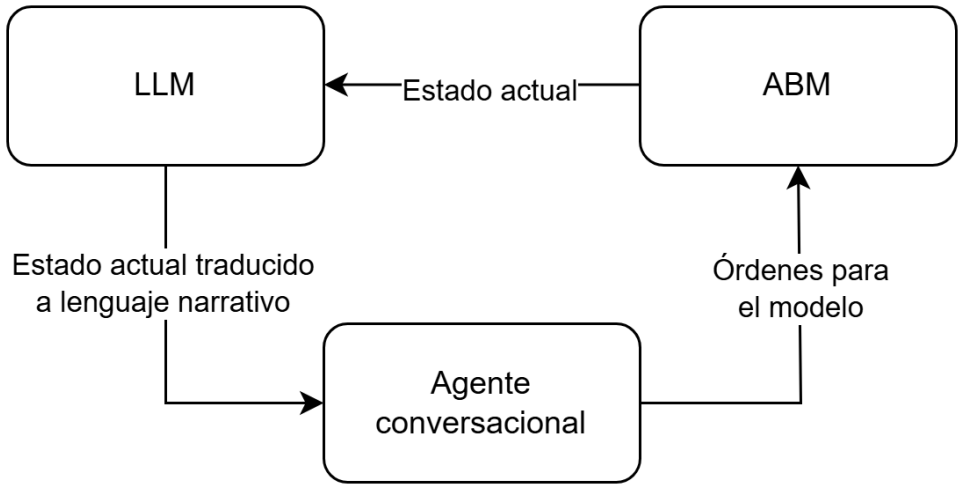


Figura 3: Arquitectura propuesta: integración entre simulador ABM, LLM Observador y agente conversacional.

5.3. ¿Por qué un entorno simulado y no datos estáticos?

Una alternativa natural al uso de un simulador sería trabajar con **datos estáticos**, entendidos como colecciones de interacciones ya registradas (por ejemplo, *logs* o historiales de chat). Sin embargo, actualmente no existen *datasets* diseñados para evaluar la coherencia espacial y temporal en LLMs. En este contexto, los registros de chat podrían considerarse la opción más intuitiva, ya que permitirían analizar cómo un modelo mantiene consistencia a lo largo de una conversación. No obstante, el uso de un ABM ofrece ventajas frente a este tipo de datos:

- **Ground truth controlado**: el simulador define con exactitud el estado en cada paso de la simulación, lo que permite realizar pruebas y comparaciones objetivas y reproducibles.
- **Escalabilidad**: se pueden modificar reglas, complejidad del entorno o condiciones iniciales para explorar distintos escenarios o escalar el problema.
- **Alineación con el objetivo**: la coherencia espacial y temporal requiere un entorno que evolucione dinámicamente; los datos estáticos, incluso los registros

de chat, no capturan estas dependencias de largo plazo ni ofrecen un estado explícito del entorno.

5.4. Alternativas Consideradas

En una primera etapa del proyecto se desarrolló una implementación basada en `Mesa`, un framework para simulación multiagente en Python. El objetivo era construir un mundo ficticio de gran escala, poblado por múltiples agentes con atributos heterogéneos y un terreno cargado desde mapas pixelados de distinta tipología (bosques, montañas, ríos, casas, etc.). En este diseño, coexistían dos LLMs con roles diferenciados:

- **LLM Narrador**: observaba el estado global de la simulación y actuaba como un narrador de estilo literario, describiendo en lenguaje natural la situación del entorno.
- **LLM Protagonista**: representaba a un agente específico de la simulación, encargado de mantener una conversación con otro agente conversacional. Este modelo respondía en base a la información que el narrador le entregaba, como si se tratara de una sesión de *juego de rol*.

Este enfoque buscaba emular la dinámica de un juego de rol como *Dungeons & Dragons*, en el que el narrador (Dungeon Master) traduce el estado del mundo al jugador, y el protagonista interactúa con el usuario de manera verosímil. La intención era evaluar la capacidad de los LLMs para sostener un rol persistente dentro de un entorno complejo y dinámico.

Sin embargo, esta alternativa fue finalmente descartada por varias razones. En primer lugar, la dificultad de construir una simulación lo suficientemente realista en `Mesa` excedía el alcance de este trabajo. Adicionalmente, la integración del LLM Narrador con la simulación en tiempo real presentaba importantes desafíos: los modelos tenían problemas para generar eventos consistentes y afectar el estado

del mundo de manera coherente. Estos factores hicieron que la propuesta resultara demasiado ambiciosa para el marco de la memoria, lo que motivó el cambio hacia un entorno más controlado y simple como el implementado actualmente.

A pesar de haber sido descartada, esta aproximación representó un paso importante hacia la solución final. El LLM Narrador de este diseño inicial evolucionó conceptualmente hasta convertirse en el LLM Observador de la propuesta actual, encargado de traducir el estado de la simulación en descripciones accesibles. Del mismo modo, el LLM Protagonista fue reemplazado por un agente conversacional, el cual puede estar representado por un LLM o un usuario, simplificando la interacción sin perder la lógica de un agente central que participa en el entorno. Esto con el fin de facilitar la implementación y evaluación de la solución propuesta. En este sentido, la experiencia adquirida en esta primera etapa permitió delimitar el alcance del proyecto y sentar las bases de la integración entre simulación y LLM que constituye el núcleo de la presente memoria.

6. Experimentos

En esta sección se detalla el objetivo de los experimentos, el contexto y las reglas de mundo de la simulación implementada, la metodología empleada, el diseño experimental y la implementación del sistema para la recolección de resultados.

6.1. Objetivos de los Experimentos

El propósito principal de estos experimentos es evaluar la capacidad de diferentes LLMs para reconstruir y representar el estado de un entorno simulado. Para ello se utilizará una variación del juego de mesa *Mind MGMT* [24], que ofrece una mecánica clara pero con suficiente complejidad del estado de mundo para poner a prueba las capacidades de los modelos [25, 26] bajo dos condiciones de acceso a la información:

1. **Observer**: El modelo recibe acceso completo al estado del juego (simulación).
2. **Blind**: El modelo sólo recibe un resumen textual de los sucesos del turno y debe inferir el resto del estado del juego.

Los experimentos tienen como objetivos específicos: medir y comparar la precisión de los LLMs en la reconstrucción del estado del juego, cuantificada mediante la distancia de *Hamming* (métrica definida en la Sección 6.4) entre el estado real y el estado predicho; analizar cómo la cantidad y tipo de información disponible afectan el rendimiento del modelo; comparar el desempeño entre modelos de diferentes arquitecturas y tamaños de parámetros. La elección de la distancia de Hamming se justifica porque el estado del tablero se representa mediante un vector de enteros discretos —resultado de la vectorización de sus celdas y atributos—, de modo que esta métrica permite contabilizar de manera directa el número de discrepancias entre posiciones equivalentes en ambos vectores. A diferencia de otras

medidas de distancia continua, la distancia de Hamming se ajusta de forma natural a la comparación de representaciones categóricas y discretas, lo que la convierte en la opción más adecuada para evaluar la coherencia en este contexto.[2]

El objetivo es aprovechar la estructura formal de estos juegos como marco de evaluación. Se plantea medir la coherencia de manera explícita a través de la representación y evolución de un entorno simulado turno a turno.

6.2. Modelos Utilizados

Zhang et al. [5] destacan la importancia de realizar evaluaciones sistemáticas con múltiples modelos para obtener una visión comparativa de su desempeño en coherencia. Por lo que se emplearon tres modelos, disponibles en la plataforma Ollama, para estos experimentos :

- **Gemma3:4b:** modelo ligero, orientado a eficiencia y buen balance entre calidad y costo computacional.
- **Qwen3:8b:** modelo intermedio en tamaño, diseñado para cubrir un rango más amplio de tareas generales.
- **Ollama3.1:8b:** modelo de mayor tamaño en esta selección, que sirve como referencia de rendimiento robusto bajo las mismas condiciones.

En este sentido, las diferencias en escalas de modelo y familias arquitectónicas pueden influir significativamente en tareas de razonamiento y en la consistencia a lo largo de interacciones prolongadas. Por ello, la elección de los modelos utilizados en este trabajo responde tanto a su disponibilidad en Ollama como a su diversidad en arquitectura y número de parámetros, lo que permite comparar de

[2]Otras métricas de distancia, como la Euclídea o la del coseno, son más apropiadas para vectores continuos o embeddings distribuidos, ya que capturan magnitudes y ángulos en espacios vectoriales de alta dimensionalidad. En este trabajo, sin embargo, lo relevante no es la proximidad geométrica, sino el conteo exacto de coincidencias y discrepancias en representaciones categóricas.

manera controlada el impacto de estas variables en la coherencia espacial y temporal de los agentes.

Con la selección de estos modelos establecida, se diseñó una metodología experimental orientada a evaluar su desempeño de manera sistemática, garantizando condiciones comparables entre las diferentes configuraciones de observación y asegurando la reproducibilidad de los resultados.

6.3. Diseño de Experimentos

El entorno experimental se implementa como un tablero de 7×7 celdas donde interactúan dos entidades:

- **Jugador**: controlado por el usuario, puede realizar hasta dos movimientos ortogonales por turno.
- **Oponente**: se desplaza una celda ortogonal por turno, sin poder quedarse en la misma posición ni volver a una casilla ya visitada. En la fase inicial realiza cinco movimientos aleatorios desde la posición central.

Cada celda del tablero contiene información estructurada: coordenadas, color, dos valores booleanos (nombrados `know_if` y `know_when`, los cuales indican si el jugador sabe si el oponente visitó dicha celda y, en caso afirmativo, si conoce cuándo lo hizo respectivamente), y, finalmente, cuando corresponde, el turno en el que el oponente la visitó.

El flujo básico es el siguiente:

1. El simulador ABM actualiza el estado del tablero siguiendo las reglas descritas.
2. El LLM genera su representación vectorizada del estado.
3. Se comparan ambos vectores y se calcula la discrepancia (distancia de Hamming).

El ciclo se repite durante un máximo de 10 turnos, concluyendo si el jugador coincide con el oponente en una celda, si el oponente queda sin movimientos legales o si se alcanza el límite de turnos.

Este diseño mantiene un balance entre simplicidad y riqueza: es lo suficientemente controlado para asegurar reproducibilidad, pero también lo bastante complejo para desafiar la capacidad de los modelos de mantener coherencia en escenarios de múltiples pasos.

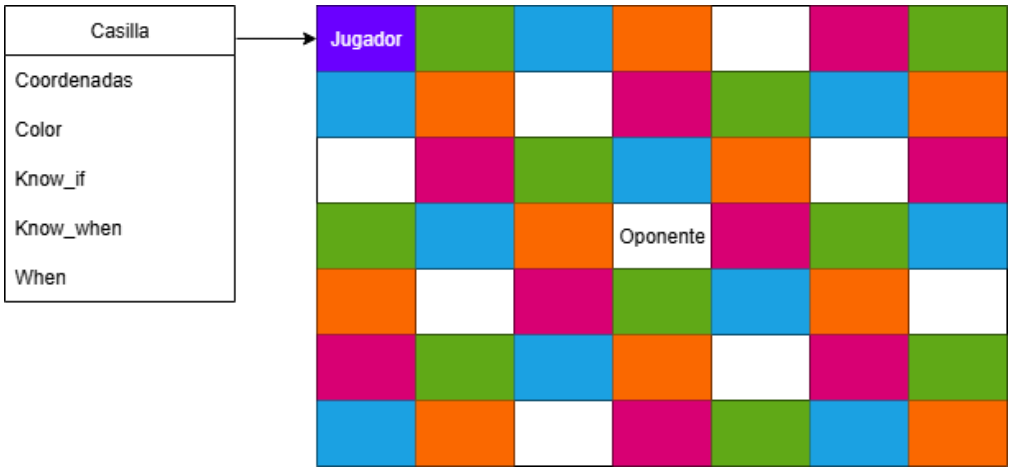


Figura 4: Representación visual del tablero de juego al inicializarlo

6.3.1 Reglas del juego

El entorno se define sobre una grilla de 7×7 celdas, en la cual interactúan dos entidades principales:

- **Jugador**: controlado por el usuario, realiza hasta dos movimientos ortogonales por turno.
- **Oponente**: se mueve una celda ortogonal por turno, sin poder quedarse en la misma posición ni volver a una casilla ya visitada.

Cada celda del tablero contiene información contextual:

- `coords`: coordenadas (x, y) de la celda.

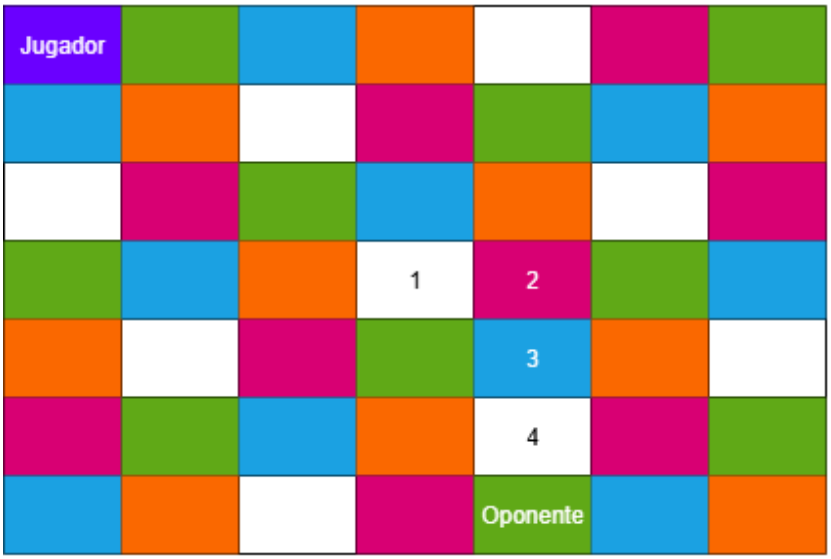
- `color`: uno de cinco valores posibles (blanco, azul, negro, rojo, verde).
- `know_if`: booleano que indica si el jugador sabe que el oponente pasó por esa celda.
- `know_when`: booleano que indica si el jugador sabe en qué turno el oponente pasó por la celda.
- `when`: número de turno en el que ocurrió el paso del oponente (si corresponde).

Las reglas básicas de la simulación son:

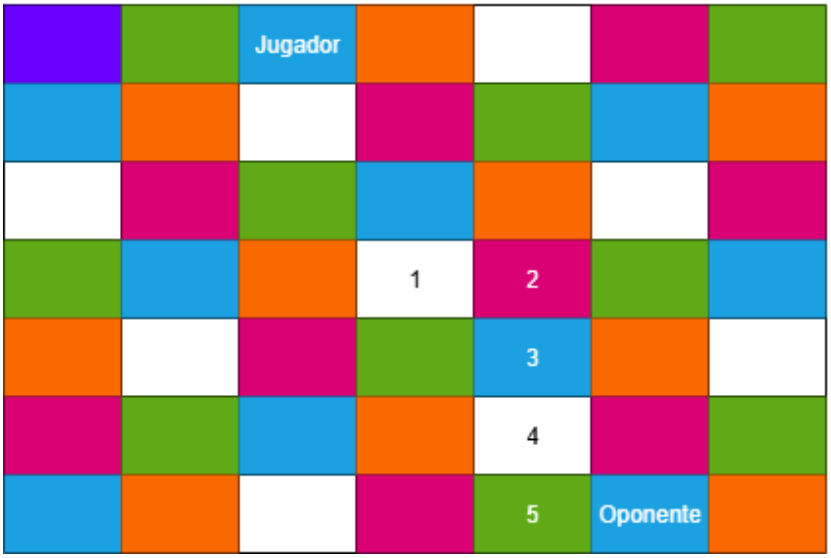
- **Inicio**: el jugador comienza en la posición $(0,0)$, mientras que el oponente inicia en el centro $(3,3)$ y realiza 5 movimientos aleatorios iniciales.
- **Movimiento del oponente**: en cada turno, se desplaza a una celda ortogonal válida.
- **Movimiento del jugador**: en cada turno, luego del oponente, el jugador realiza hasta dos movimientos ortogonales. Puede revisitar celdas anteriores.
- **Investigación**: al finalizar cada turno, el jugador revisa el color de la celda en la que se encuentra.
    - Si el oponente ha pasado por alguna celda de ese color, la más reciente se marca con `know_if = True`.
    - Si el jugador se encuentra en una celda con `know_if = True`, esta pasa a tener también `know_when = True` y se registra el turno en `when`.
- **Condición de victoria**: el jugador gana si coincide con el oponente en la misma celda o si este queda atrapado sin movimientos legales.
- **Condición de término**: la partida finaliza tras un máximo de 10 turnos.

La Figura 5 muestra una secuencia de los primeros turnos de una partida simulada. Se representa la grilla completa y sus casillas. En cada casilla se indica, su atributo de color, la posicion del jugador u oponente, y un entero el cual corresponde al número del movimiento del oponente en el cual pasó por dicha casilla.

Inicializacion



Turno 1



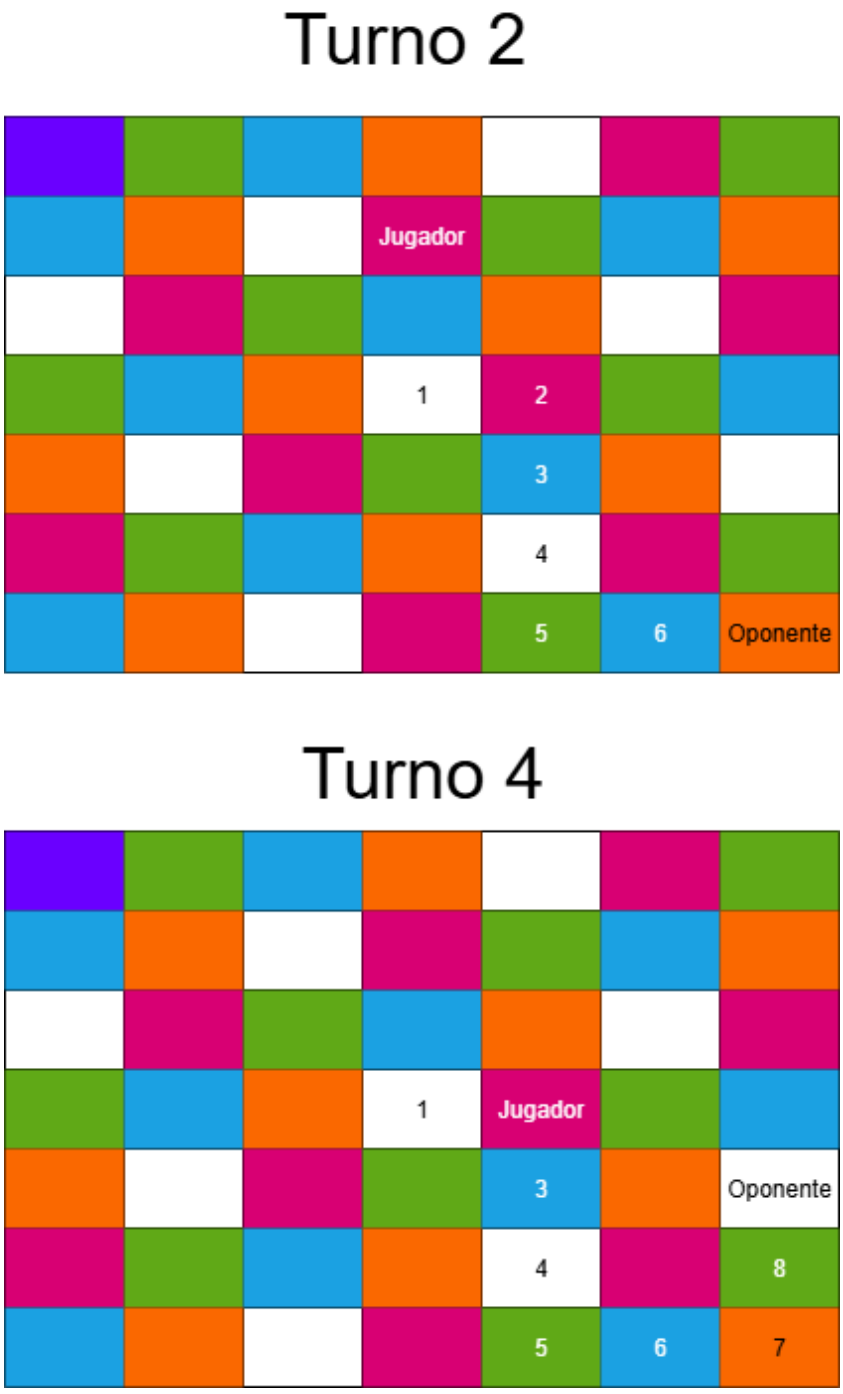


Figura 5: Representación visual del tablero de juego durante el transcurso de múltiples turnos de juego

6.3.2 Componentes de la Implementación

- **Simulador ABM**: mantiene la grilla y ejecuta las reglas anteriores, generando un estado persistente del juego.
- **LLM Observer**: recibe el estado del simulador y produce descripciones narrativas que reflejan lo que el jugador sabe hasta ese turno. Este módulo cumple un rol de mediador, transformando el estado estructurado en lenguaje natural, lo que permite evaluar la coherencia en la representación del entorno.
- **Agente conversacional)**: interactúa con el sistema ingresando movimientos en cada turno y recibe retroalimentación del LLM Observer.
- **Módulo de Registro**: guarda el estado y las narraciones generadas en cada turno, asegurando trazabilidad y la posibilidad de análisis posterior.

6.4. Metodología

La métrica utilizada para cuantificar la discrepancia entre el estado real de la simulación y el estimado por el LLM es la *distancia de Hamming*. Dadas dos secuencias binarias x y y de longitud n, la distancia de Hamming se define como el número de posiciones en las que ambos vectores difieren:

$$d_H(x,y) = \sum_{i=1}^{n} \mathbb{1}_{\{x_i \neq y_i\}} \tag{1}$$

donde $\mathbb{1}_{\{x_i \neq y_i\}}$ es la función indicadora que vale 1 si $x_i \neq y_i$ y 0 en caso contrario.

En este trabajo, los vectores representan el estado de la grilla en cada turno, de modo que $d_H(x,y)$ mide cuántos elementos del estado fueron representados de manera incorrecta por el modelo en comparación con el simulador. El enfoque experimental propuesto se centra en cuantificar la precisión con que cada LLM representa el estado interno de la simulación, expresado como un vector de 49

enteros que codifica la grilla 7x7 del juego. Esta representación incluye tanto la ubicación del jugador y del oponente, como información derivada de las reglas de investigación.

En cada turno:

1. Se actualiza el estado de la simulación en función de los movimientos del jugador y del oponente.
2. Se genera un *prompt* adaptado al tipo de observador (completo para *Observer*, parcial para *Blind*).
3. El LLM devuelve un vector de 49 enteros entre 0 y 5 siguiendo la codificación establecida.
4. Se compara el vector devuelto con el vector real de la simulación mediante la distancia de *Hamming*.
5. Se registra la métrica junto con el número de turno en un archivo CSV.

En caso de que el LLM no entregue un vector válido, se registra un valor `NaN`. La coherencia se evalúa comparando, turno a turno, los vectores reales de la simulación con los vectores producidos por el LLM. La distancia de Hamming entre ambas representaciones se emplea como métrica principal, ya que permite cuantificar de forma directa el grado de discrepancia entre el estado verdadero y el estimado por el modelo.

Esta metodología de experimentación es realizada sobre los tres modelos mencionados anteriormente (Gemma3:4b, Qwen3:8b y Ollama3.1:8b). Sobre cada modelo se simularon un total de 200 partidas del juego simulado. Del total de partidas 100 fueron realizadas con el LLM teniendo acceso constante al estado de la simulación, y 100 fueron realizadas con el LLM sin acceso al estado de la simulación. Los experimentos fueron ejecutados en un equipo con procesador Intel® Core™ i9-13900K (13ª generación), 128 GB de memoria RAM y dos GPU

NVIDIA® RTX 4090, lo que permitió realizar las pruebas de manera eficiente y con alta capacidad de procesamiento paralelo.

7. Resultados y Discusión

En esta sección se presentan y discuten los resultados de los experimentos para cada combinación de LLMs y modos de acceso a la información utilizados.

7.1. Resultados

En esta sección se presentan los resultados de los experimentos realizados, organizados por modelo y modo de observación. Para cada configuración se incluyen dos tipos de gráficos: el promedio de distancias de Hamming por turno, y las envolventes (valores mínimo y máximo). Con estos gráficos se puede comparar la estabilidad y consistencia de los modelos bajo condiciones de acceso completo (*Observer*) o parcial (*Blind*) al estado de la simulación.

7.1.1 Resultados con Ollama

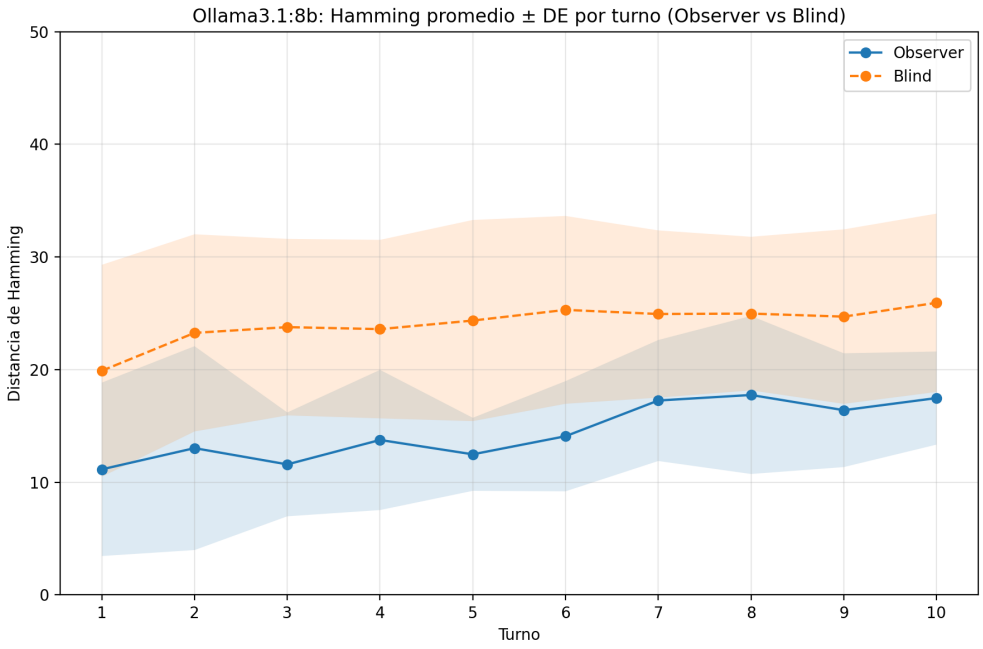


Figura 6: Distancia de Hamming promedio (±DE) por turno para Ollama3.1:8b en condiciones Observer y Blind.

En la Figura 6 se presentan conjuntamente el promedio de distancia de Hamming por turno y las desviaciones estándar en los modos *Observer* y *Blind*. En el modo *Observer*, la distancia Hamming se mantiene en valores relativamente bajos y con una progresión estable a lo largo de los 10 turnos, lo que indica que el modelo es capaz de reconstruir el estado del juego con un nivel de consistencia aceptable cuando dispone de acceso completo a la información de la simulación.

Por otro lado, en el modo *Blind* se aprecia un aumento sostenido de la distancia de Hamming a medida que avanzan los turnos, alcanzando valores significativamente más altos hacia el final de la partida. Este patrón observado evidencia la limitación del modelo para conservar coherencia temporal cuando su desempeño se apoya exclusivamente en la memoria implícita y en la interpretación de resúmenes parciales.

A su vez, las bandas de envolventes complementan este análisis mostrando los valores mínimos y máximos de Hamming obtenidos en cada turno. En el modo *Observer*, la dispersión entre mínimos y máximos es reducida, lo que confirma que el modelo entrega resultados consistentes en partidas diferentes. En el modo *Blind*, en cambio, la variabilidad es mucho mayor: algunos intentos logran distancias relativamente bajas en los primeros turnos, pero otros divergen rápidamente hacia valores altos.

En resumen, los resultados de Ollama muestran un desempeño robusto en condiciones de acceso completo (*Observer*), con errores bajos y consistentes, pero presentan una degradación clara en el modo *Blind*, tanto en magnitud como en variabilidad.

7.1.2 Resultados con Gemma

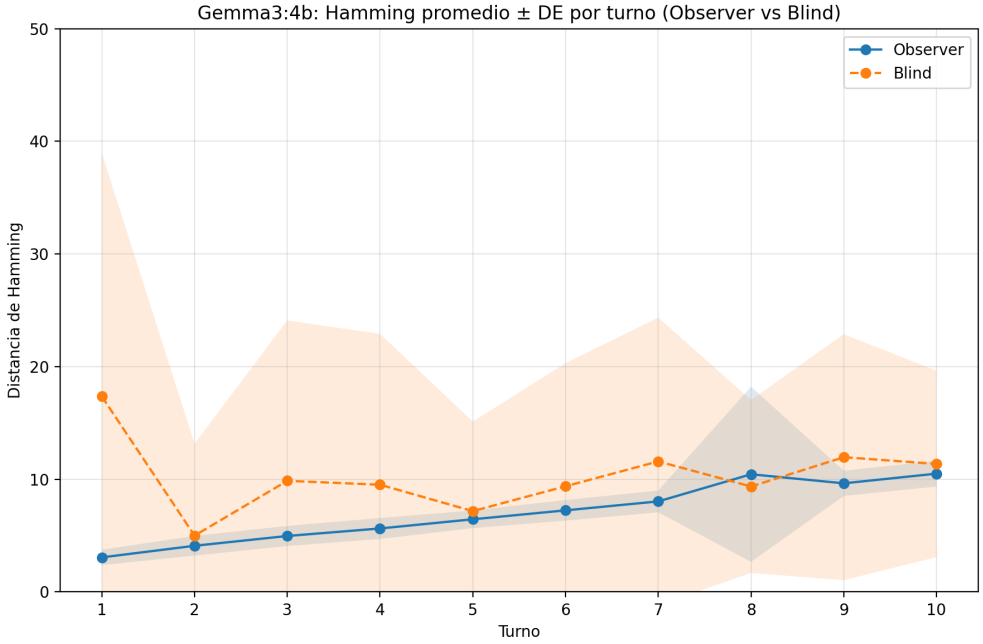


Figura 7: Distancia de Hamming promedio (±DE) por turno para Gemma3:4b en condiciones Observer y Blind.

En la Figura 7 se presentan en conjunto los resultados de Gemma, incluyendo el promedio de distancia de Hamming y las desviaciones estándar por turno en los modos *Observer* y *Blind*. En el modo *Observer*, las distancias se mantienen en niveles bajos y con un incremento moderado a lo largo de los turnos.

En el modo *Blind*, en cambio, se observan valores más irregulares: la distancia Hamming comienza con valores elevados en los primeros turnos, seguida de fluctuaciones notorias a lo largo de la partida.

En el modo *Observer*, los valores mínimos permanecen cercanos a cero y los máximos se mantienen altos pero relativamente constantes, lo que sugiere que, aunque existen discrepancias notables en algunos intentos, la mayoría de las partidas producen resultados cercanos entre sí. En el modo *Blind*, la variabilidad es mucho mayor: los valores máximos alcanzan picos significativos en turnos inter-

medios y finales, mientras que los mínimos muestran un crecimiento progresivo.

7.1.3 Resultados con Qwen

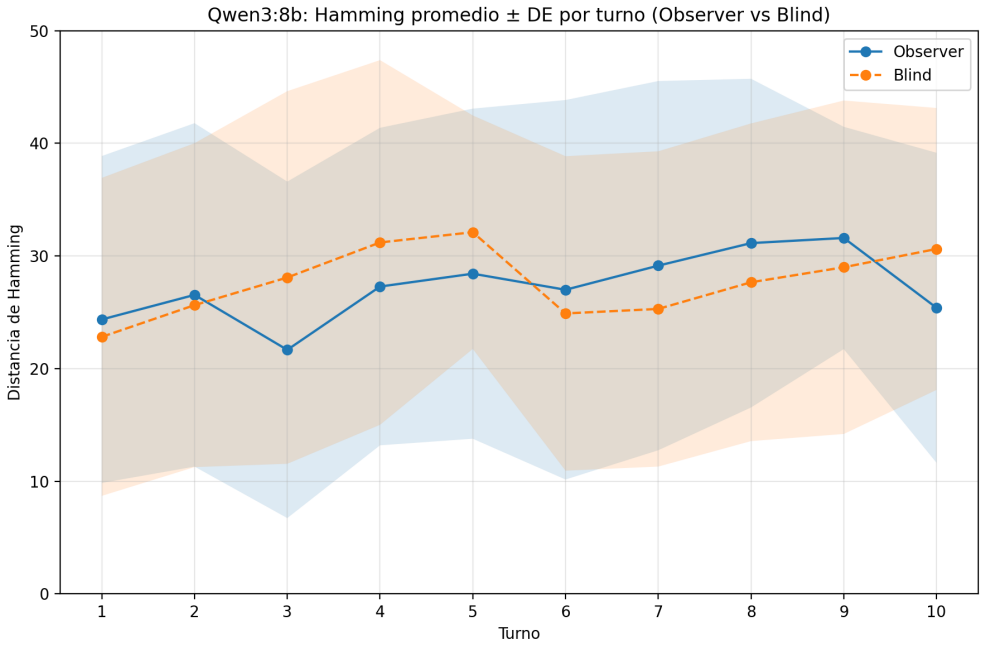


Figura 8: Distancia de Hamming promedio (±DE) por turno para Qwen3:8b en condiciones Observer y Blind.

En la Figura 8 se presentan en conjunto los promedios de distancia de Hamming y las desviaciones estándar por turno en los modos *Observer* y *Blind*. En el modo *Observer*, la distancia Hamming comienza con valores moderados que aumentan de manera progresiva durante los primeros turnos, alcanzando un máximo en torno al turno 5, para luego descender y estabilizarse en la segunda mitad de la partida.

En el modo *Blind*, en cambio, los valores promedios muestran un incremento más marcado y sostenido en los turnos intermedios, alcanzando picos elevados antes de descender abruptamente en los últimos turnos.

En el modo *Observer*, los valores mínimos se mantienen bajos a lo largo de toda la partida, mientras que los máximos se mantienen altos pero relativamente estables, lo que indica cierta consistencia en la calidad de las reconstrucciones. En

el modo *Blind*, sin embargo, los valores máximos alcanzan niveles muy elevados en los turnos intermedios y finales, mientras que los mínimos presentan una fuerte dispersión.

7.1.4 Comparación global (Observer vs Blind)

En la Figura 9 se presenta la comparación entre las condiciones Observer y Blind para los tres modelos evaluados (Gemma3:4b, Qwen3:8b y Ollama3.1:8b). El eje X corresponde al número de turno (1–10) y el eje Y a la distancia de Hamming promedio por turno. Para cada modelo se trazan dos series: Observer (línea continua) y Blind (línea discontinua). En términos generales, Observer exhibe menores distancias de Hamming promedio que Blind en la mayor parte de los turnos, lo que sugiere que el acceso al estado del entorno reduce la discrepancia entre el estado real y el estimado por el modelo.

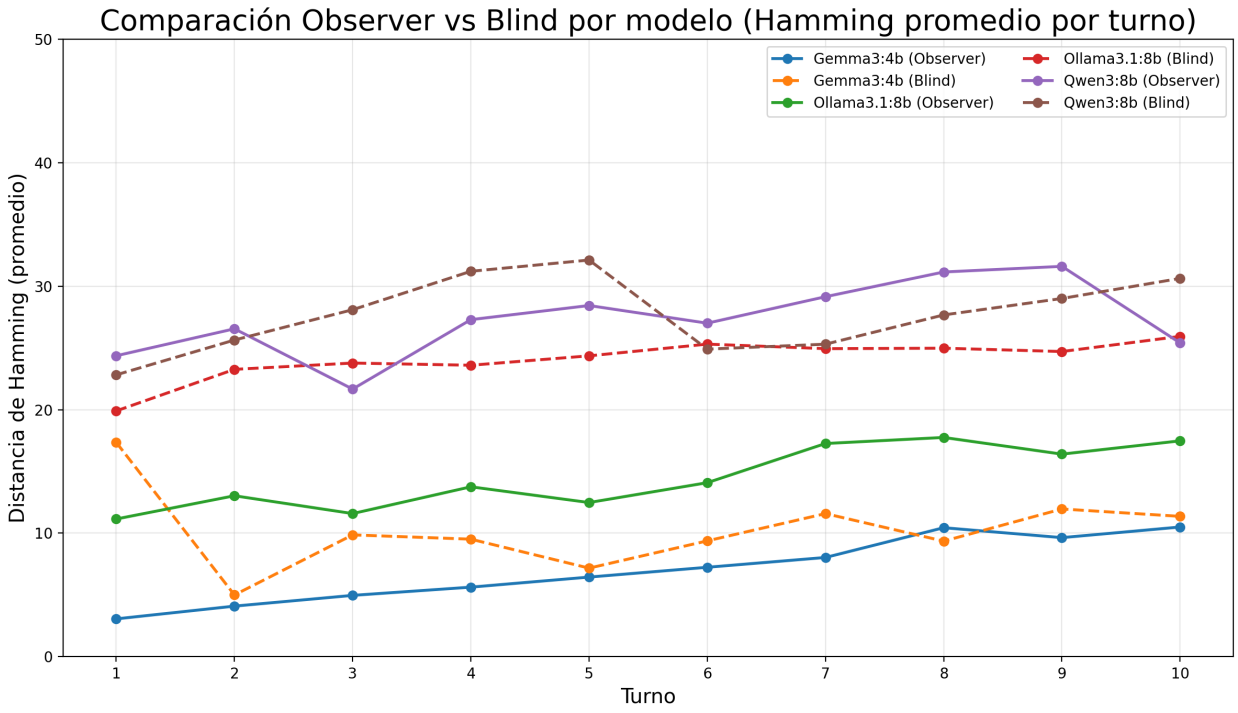


Figura 9: Comparación Observer vs Blind por modelo: Hamming promedio por turno (1–10).

7.2. Validez Estadística

Para evaluar la robustez de los resultados se calcularon promedios y desviaciones estándar de la distancia de Hamming a partir de 100 ejecuciones por combinación de modelo y modo (Observer y Blind). Esto permitió analizar tanto la evolución turno a turno como los valores globales medios.

La Figura 6 muestra la evolución de la distancia de Hamming promedio para el modelo Ollama3.1:8b, comparando Observer y Blind. Se observa que la condición Observer mantiene sistemáticamente valores más bajos de discrepancia, mientras que Blind presenta mayores distancias acumuladas. Resultados similares se observan en la Figura 7 para Gemma3:4b. En contraste, la Figura 8 muestra que en Qwen3:8b las curvas de Observer y Blind son cercanas, sin diferencias claras.

Además, la Figura 10 resume los valores globales (media de medias por ejecución ±DE) para cada modelo y modo. Puede apreciarse que, en promedio, Observer reduce la distancia de Hamming en Ollama y Gemma, mientras que en Qwen las diferencias entre modos son mínimas.

Los valores globales se presentan en la Tabla 1, donde se observa que Ollama3.1:8b y Gemma3:4b presentan diferencias marcadas entre modos, mientras que en Qwen3:8b ambas condiciones obtienen valores cercanos.

Modelo	Modo	Media de medias	DE
Gemma3:4b	Observer	6.61	1.36
Gemma3:4b	Blind	9.79	3.91
Ollama3.1:8b	Observer	14.22	4.44
Ollama3.1:8b	Blind	23.41	3.99
Qwen3:8b	Observer	25.49	13.89
Qwen3:8b	Blind	25.52	14.47

Cuadro 1: Resumen global de la distancia de Hamming promedio (media $\pm$ DE) por modelo y modo.

Para comprobar la significancia estadística de estas diferencias, se aplicó un ANOVA de dos vías con factores *Modelo* y *Modo*. Los resultados (Tabla 2) indican

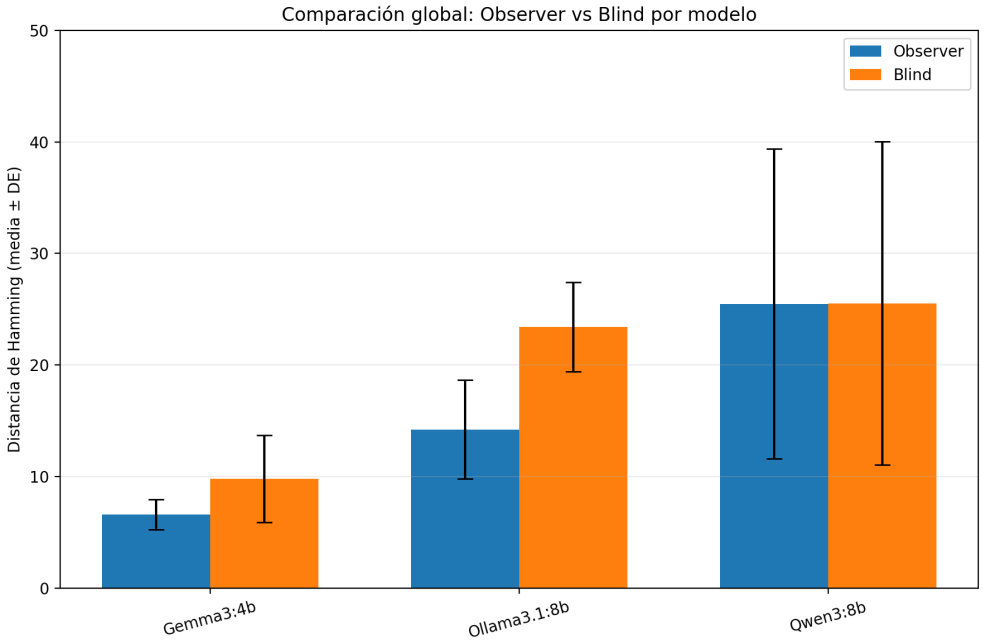


Figura 10: Comparación global de la distancia de Hamming promedio (±DE) entre condiciones Observer y Blind para cada modelo.

efectos principales altamente significativos de ambos factores y una interacción significativa. Esto confirma que las diferencias entre Observer y Blind dependen del modelo evaluado.

Factor	F	gl	p
Modelo	118.24	2	$< 0{,}001$
Modo	115.59	1	$< 0{,}001$
Modelo $\times$ Modo	12.30	2	$< 0{,}001$

Cuadro 2: Resultados del ANOVA de dos vías (Modelo $\times$ Modo).

En conjunto, los resultados sugieren que la condición Observer mejora la coherencia (menor distancia de Hamming) en Ollama3.1:8b y Gemma3:4b, mientras que en Qwen3:8b no se observan diferencias significativas entre modos. Esto refuerza la conclusión de que el acceso al estado del entorno contribuye a la consistencia en algunos modelos, pero no garantiza mejoras universales.

7.3. Discusión

Los resultados no solo describen diferencias entre Observer y Blind, sino que también permiten entender por qué se producen. En el modo Blind, el modelo debe depender únicamente de su memoria implícita y de resúmenes textuales parciales. Esto genera una *acumulación de errores*: equivocaciones iniciales se propagan y amplifican en turnos posteriores, tal como predicen trabajos como *Lost in the Middle* [4], donde se refleja la dificultad de los LLMs para mantener información consistente en interacciones largas cuando no existe un estado explícito.

En contraste, el modo Observer provee al modelo con una representación estructurada del entorno en cada turno. Esto reduce la dependencia de la memoria implícita y permite mantener una coherencia más estable. Qwen, en tanto, no mejoró significativamente en Observer, lo que concuerda con observaciones de la literatura donde algunos modelos presentan inductores de coherencia más débiles pese a contar con contexto adicional [5].

El bajo rendimiento observado en Qwen3:8b durante los experimentos iniciales se puede explicar principalmente por la formulación inadecuada del prompt, más que por limitaciones inherentes al modelo. Las instrucciones originales exigían una salida numérica estrictamente estructurada —una secuencia fija de 49 enteros separados por espacios—, lo que forzaba al modelo a operar fuera de su marco de entrenamiento lingüístico. Este tipo de formulación, carente de ejemplos o guías previas, es propensa a generar ambigüedades y errores de formato, tal como se ha documentado en estudios sobre sensibilidad al diseño del prompt [27]. Al reformular la instrucción como una tarea de parafraseo del estado del entorno, el modelo mostró una mejora significativa, lo que evidencia que el problema radicaba en el tipo de instrucción y no en su capacidad general de comprensión. Este resultado coincide con recomendaciones recientes de ingeniería de prompts, que recomiendan emplear estrategias como Few-shot prompting o Prompt Chaining para tareas

estructuradas, y Chain-of-Thought en aquellas que requieran razonamiento narrativo o contextual [14]. En consecuencia, se plantea la hipótesis de que el desempeño deficiente de Qwen en el modo Observer se debió principalmente a un desajuste entre la naturaleza del prompt utilizado y la forma en que el modelo procesa y representa la información, más que a una carencia técnica de la arquitectura subyacente.

Para poner a prueba esta hipótesis se realizaron experimentos simples en los cuales se le proporcionó un *prompt* distinto al de los experimentos originales. Este nuevo set de instrucciones solicitaba al modelo parafrasear el estado de la simulación en lugar de vectorizarlo. Esto con el objetivo de comprobar que el modelo efectivamente posee la capacidad de analizar el estado del mundo de manera correcta.

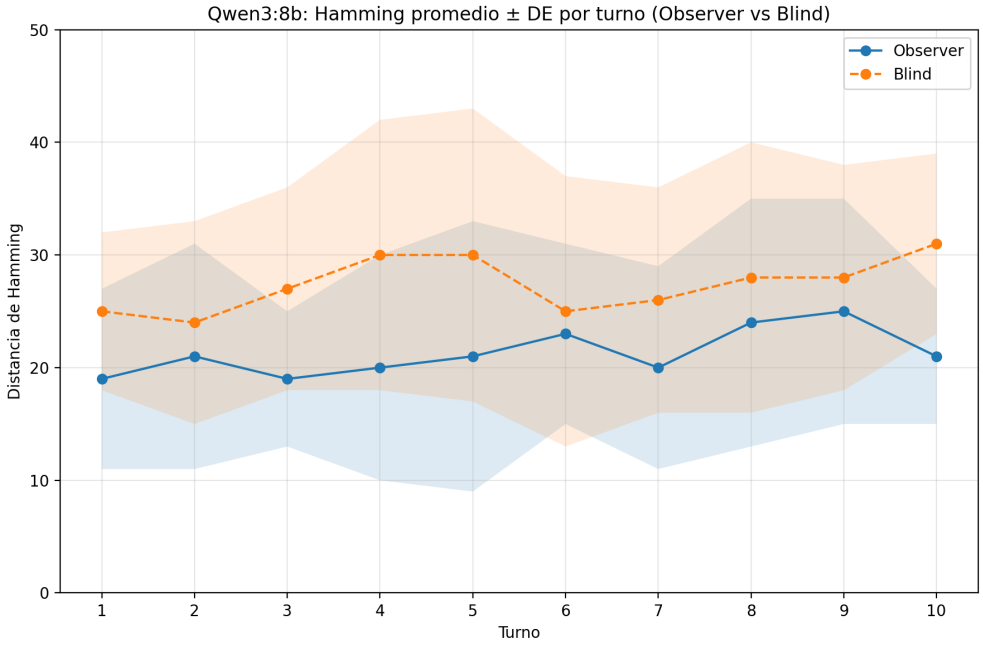


Figura 11: Distancia de Hamming promedio (±DE) por turno para Qwen3:8b en condiciones Observer y Blind (bajo el nuevo set de instrucciones).

Los resultados de este segundo set de experimentos confirman la hipótesis planteada. Tal como se observa en la Figura 11, el rendimiento de Qwen3:8b mejora

de manera consistente en comparación con los experimentos originales, mostrando menores distancias de Hamming promedio y una variabilidad más controlada entre turnos. Esto evidencia que el modelo es capaz de mantener una representación coherente del entorno cuando las instrucciones se alinean con su entrenamiento lingüístico. En otras palabras, al reemplazar la tarea de vectorización por una de parafraseo, el modelo opera dentro de un dominio semántico más natural, lo que reduce los errores sistemáticos observados anteriormente. Este comportamiento refuerza la noción de que la formulación del *prompt* es un componente determinante en la evaluación de coherencia, y que una instrucción mal adaptada puede inhibir las verdaderas capacidades de razonamiento de un LLM. Además, los resultados apoyan la idea, planteada en trabajos recientes sobre ingeniería de *prompts* [14], de que la adaptación del formato de entrada a la naturaleza del modelo puede ser tan determinante para el desempeño como la arquitectura o el tamaño del mismo.

Esta dualidad refleja lo planteado en propuestas recientes sobre memorias persistentes. En *Smallville* [6] y *MemGPT* [7], la incorporación de memorias externas y jerárquicas permitió a los agentes sostener interacciones coherentes en el tiempo. Asimismo, los resultados de este proyecto sugieren que el acceso explícito a un estado simulado, logran una mejora en la estabilidad narrativa en el modelo Ollama y Gemma. No obstante, la variabilidad observada confirma que la calidad final depende también del razonamiento que cada arquitectura puede desplegar sobre ese estado.

En conjunto, los hallazgos apoyan la idea de que los LLMs requieren *mecanismos complementarios de memoria o representaciones persistentes* para mantener coherencia espacial y temporal en escenarios interactivos. La evidencia presentada en este trabajo se alinea con la discusión del estado del arte y refuerza la necesidad de continuar explorando soluciones híbridas que integren tanto memorias externas como mejoras en la capacidad de razonamiento de los modelos.

8. Conclusión

El presente trabajo abordó el problema de la falta de coherencia espacial y temporal en los grandes modelos de lenguaje (LLMs), una limitación crítica cuando se espera que estos agentes sostengan interacciones prolongadas y verosímiles. La ausencia de mecanismos internos que permitan mantener una narrativa consitente más allá de la ventana de contexto restringe la credibilidad y aplicabilidad de los LLMs en escenarios interactivos y dinámicos.

Como propuesta de solución, se desarrolló un entorno de simulación basado en *Agent-Based Modeling* (ABM), en el que un LLM Observador actúa como intérprete del estado de la simulación y lo comunica en lenguaje natural. Este diseño buscó aprovechar las fortalezas de ABM —control del estado, trazabilidad y evolución dinámica— para proporcionar a los modelos un estado explícito y constante que permita evaluar su capacidad de mantener coherencia a lo largo de múltiples pasos de interacción.

Los resultados obtenidos muestran que, si bien algunos modelos (como Ollama y Gemma) evidenciaron mejoras en el modo Observer, otros (como Qwen), presentaron un comportamiendo sin mayores cambios independientemente de su nivel de acceso a un contexto adicional. Estos hallazgos indican que el simple acceso a una representación estructurada no garantiza coherencia, y que la capacidad de cada modelo para razonar sobre datos estructurados y aprovecharlos de manera efectiva juega un rol determinante.

El marco desarrollado abre múltiples líneas de investigación. Una primera dirección como trabajo a futuro es la de ampliar la complejidad del entorno simulado. Por ejemplo, incorporar tableros de mayor tamaño, reglas adicionales o dinámicas más cercanas a juegos de estrategia complejos, con el fin de evaluar cómo los modelos manejan dependencias de más largo plazo. Otra posibilidad es introducir múltiples agentes controlados por distintos LLMs, de modo que se estudie no solo

la coherencia individual, sino también la consistencia en interacciones colectivas. Asimismo, resulta relevante experimentar con técnicas de prompting más avanzadas y con memorias externas híbridas que combinen estado estructurado y narrativas. Estas extensiones permitirían evolucionar el marco hacia un escenario de pruebas flexible para explorar la coherencia en escenarios cada vez más complejos, contribuyendo a sentar bases más sólidas para el diseño de agentes conversacionales.

Referencias

[1] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 1877–1901, 2020. [Online]. Available: https://arxiv.org/abs/2005.14165

[2] W. Zhao, J. Chen *et al.*, "A comprehensive review of chatgpt: Evolution, applications, and future directions," *arXiv preprint arXiv:2304.01852*, 2023.

[3] S. Zhang, E. Dinan, J. Weston *et al.*, "Improving the coherence of open-domain dialogue systems via context-aware learning," *arXiv preprint arXiv:1906.01250*, 2019.

[4] D. Liu, B. Peng, Q. Chen, Z. Yu, K. Zhao, C. Zhu, A. H. Awadallah, and J. Gao, "Lost in the middle: How language models use long contexts," *arXiv preprint arXiv:2307.03172*, 2023. [Online]. Available: https://arxiv.org/abs/2307.03172

[5] Y. Zhang, A. Elgohary, N. A. Smith, and L. W. Lee, "Benchmarking coherence in dialogue models," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2023. [Online]. Available: https://arxiv.org/abs/2305.06979

[6] J. S. Park, J. C. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein, "Generative agents: Interactive simulacra of human behavior," in *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*. ACM, 2023. [Online]. Available: https://arxiv.org/abs/2304.03442

[7] T. Wu, Z. Chen, S. Choudhury, Y. Fu, Z. Song, Z. Yang, S. Zha, and D. Yu, "Memgpt: Towards llms as operating systems," *arXiv preprint arXiv:2310.08560*, 2023. [Online]. Available: https://arxiv.org/abs/2310.08560

[8] C. Zhang, C. Qian *et al.*, "From individual to society: A survey on social simulation driven by llm-based agents," *ACM Computing Surveys*, 2024. [Online]. Available: https://arxiv.org/abs/2307.01033

[9] OpenAI, "Helping people when they need it most," https://openai.com/index/helping-people-when-they-need-it-most, 2023, Último acceso: febrero 2025.

[10] E. Bonabeau, "Agent-based modeling: Methods and techniques for simulating human systems," *Proceedings of the National Academy of Sciences*, vol. 99, no. suppl 3, pp. 7280–7287, 2002.

[11] N. Gilbert, *Agent-Based Models*, ser. Quantitative Applications in the Social Sciences. SAGE Publications, 2008, vol. 07-153.

[12] Wikipedia contributors, "Agent-based model," https://en.wikipedia.org/wiki/Agent-based_model, 2025.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS)*. Curran Associates Inc., 2017, pp. 6000–6010. [Online]. Available: https://arxiv.org/abs/1706.03762

[14] T. Schick and otros, "Prompt engineering guide - técnicas de prompting," 2025, Último acceso: agosto 2025. [Online]. Available: https://www.promptingguide.ai/es/techniques

[15] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. H. Chi, Q. V. Le, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. [Online]. Available: https://arxiv.org/abs/2201.11903

[16] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. H. Chi, D. Zhou *et al.*, "Self-consistency improves chain of thought reasoning in language models," *arXiv preprint arXiv:2203.11171*, 2023. [Online]. Available: https://arxiv.org/abs/2203.11171

[17] W. Sun, W. Hu, H. Cheng, K. McKeown, and X. Ren, "Bitimebert: Incorporating bi-temporal information into pre-trained language representations," *arXiv preprint arXiv:2204.13032*, 2023. [Online]. Available: https://arxiv.org/abs/2204.13032

[18] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive nlp tasks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. [Online]. Available: https://arxiv.org/abs/2005.11401

[19] H. Chase, "Langchain," https://www.langchain.com/, 2022, Último acceso: septiembre 2025.

[20] J. D. Murray, *Mathematical Biology I: An Introduction*, 3rd ed. Springer, 2002.

[21] S. H. Strogatz, *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering*, 2nd ed. CRC Press, 2018.

[22] J. Hernández, B. Silva *et al.*, “Llm-augmented agent-based modelling for social simulations: Challenges and opportunities,” in *Frontiers in Artificial Intelligence and Applications*, vol. 386. IOS Press, 2024, pp. 190–205.

[23] F. Li, W. Zhang *et al.*, “Learning agent-based modeling with llm companions,” in *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, 2024.

[24] Off the Page Games, “Mind mgmt: The psychic espionage "game",” https://offthepagegames.com/mind-mgmt, 2020, Último acceso: septiembre 2025.

[25] R. Wang, Z. Li, Z. Zhang *et al.*, “Character-llm: A trainable agent for role-playing,” in *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2024. [Online]. Available: https://arxiv.org/abs/2309.11419

[26] U. Author, “Exploring the potential of llm-based agents as dungeon masters in tabletop role-playing games,” Master’s thesis, Master Thesis, [University name if available], 2024. [Online]. Available: https://studenttheses.uu.nl/handle/20.500.12932/47209

[27] X. Lu, S. Mishra, S. Arora *et al.*, “On the sensitivity of language models to prompt format,” *arXiv preprint arXiv:2109.08496*, 2021. [Online]. Available: https://arxiv.org/abs/2109.08496