Jornadas de Automática

Generación y Ajuste Dinámico de Behavior Trees mediante LLMs para el Control de un Robot Social

Merino-Fidalgo, Sergio[a,*], Zalama, Eduardo[a,b], Gómez-García-Bermejo, Jaime[a,b], Duque-Domingo, Jaime[a]

[a] *ITAP-DISA, Universidad de Valladolid, C/ Doctor Mergelina, 47011, Valladolid, España.*
[b] *CARTIF Centro Tecnológico, 47151, Valladolid, España.*

To cite this article: Merino-Fidalgo, S., Zalama, E., Gómez-García-Bermejo, J., Duque-Domingo, J. 2025. Generation and Dynamic Adjustment of Behavior Trees using LLMs for the Control of a Social Robot. Jornadas de Automática, 46. https://doi.org/10.17979/ja-cea.2025.46.12071

Resumen

Los modelos de lenguaje a gran escala (LLMs) se han convertido en herramientas clave para generar comportamientos robóticos flexibles y conscientes del contexto. Sin embargo, adaptarse a los eventos imprevistos y garantizar la consecución de las tareas sigue siendo un reto importante. Este trabajo presenta un sistema que utiliza LLMs y Árboles de Comportamiento (BTs) para que un robot social genere, ejecute y adapte planes de tareas a partir de instrucciones en lenguaje natural. Combinando un planificador de BT y un módulo de interpretación de fallos, el sistema ajusta dinámicamente los BTs frente a errores de ejecución o cambios en el entorno. A diferencia de métodos basados en BTs estáticos, nuestro enfoque detecta problemas y propone alternativas o solicita aclaraciones al usuario, mejorando la interacción humano-robot. Validamos el sistema en múltiples escenarios reales, demostrando su flexibilidad y resiliencia en entornos dinámicos.

Palabras clave: Planificación de misiones y toma de decisiones, Aspectos cognitivos de los sistemas de automatización y los seres humanos, Robots móviles, Modelos de lenguaje a gran escala, Behavior Trees

Generation and Dynamic Adjustment of Behavior Trees using LLMs for the Control of a Social Robot

Abstract

Large Language Models (LLMs) have become key tools for generating flexible and context-aware robotic behaviors. However, adapting to unforeseen events and ensuring robust task completion remain significant challenges. This work presents a system that uses LLMs and Behavior Trees (BTs) to enable a social robot to generate, execute, and adapt task plans based on natural language instructions. By combining a BT planner with a failure interpretation module, the system dynamically adjusts BTs in response to execution errors or environmental changes. Unlike static BT-based methods, our approach detects problems and proposes alternatives or requests clarifications from the user in real time, improving human-robot interaction. We validate the system across various real-world scenarios, demonstrating its effectiveness in enhancing flexibility and resilience in dynamic environments.

Keywords: Mission planning and decision making, Cognitive aspects of automation systems and humans, Mobile robots, Large language models, Behavior Trees

1. Introducción

El control de tareas ha representado un desafío clave en los sistemas autónomos, especialmente en el ámbito de la robótica. A medida que estos sistemas aumentan en complejidad, se requiere una gestión más robusta y flexible. Las tradicionales máquinas de estados finitos (FSM) (Brand and Zafiropulo, 1983) resultan limitadas, dado que no pueden representar comportamientos no lineales o dependientes del contexto y

*Autor para correspondencia: sergio.merino.fidalgo@uva.es

carecen de memoria estructurada. Por ello, su uso se restringe al control de sistemas embebidos o aplicaciones con estrictos requisitos deterministas.

Los Árboles de Comportamiento (BTs) (Ögren and Sprague, 2022), surgidos en el contexto de los videojuegos, permiten estructurar tareas complejas de manera jerárquica, promoviendo así la modularidad y la claridad en su ejecución. Sin embargo, su implementación y ajuste requieren conocimientos en su programación y en el control de procesos.

Paralelamente, los Modelos de Lenguaje a Gran Escala (LLMs), como ChatGPT (Chang et al., 2024; OpenAI, 2024; Deng and Lin, 2022), se han consolidado como herramientas eficaces para la comprensión y el razonamiento en lenguaje natural. Estas capacidades han impulsado su uso como asistentes en la generación automática de código, incluyendo la creación completa de Árboles de Comportamiento a partir de instrucciones adecuadas. Sin embargo, la integración de BTs y LLMs conlleva importantes retos técnicos: los BTs demandan un diseño cuidadoso para afrontar eventos inesperados, mientras que los LLMs, al no poseer una comprensión semántica profunda comparable a la humana, pueden producir resultados imprecisos o erróneos.

Este artículo propone un marco que une BTs y LLMs para desarrollar un sistema robótico adaptativo capaz de ejecutar tareas de forma eficiente y flexible. El usuario proporciona una instrucción al robot, la cual es transmitida al ordenador central y procesada por un LLM. La respuesta generada por el LLM es luego analizada por el módulo Clarificador, que evalúa si la instrucción es válida o requiere aclaraciones adicionales. Si la instrucción es válida, se ejecuta el BT generado. Si se detecta un fallo durante la ejecución, el Manejador de Fallos actúa modificando el BT. Este enfoque permite que los robots no solo sean capaces de ejecutar las instrucciones de los usuarios en lenguaje natural, sino también de adaptar su comportamiento, mejorando así la robustez y usabilidad del sistema.

En un trabajo previo, presentamos un sistema inicial de generación de BTs a partir de instrucciones en lenguaje natural utilizando LLMs. Basándonos en esa propuesta, este artículo introduce una extensión significativa que no solo permite la generación de BTs, sino también su modificación y adaptación dinámica ante fallos de ejecución o cambios en el entorno, lo que aborda las limitaciones de robustez y fiabilidad detectadas en el sistema original.

El objetivo último de esta propuesta es su instalación en hogares de personas mayores, permitiendo que usuarios sin conocimientos técnicos puedan comunicarse fácilmente con el robot y este sea capaz de ejecutar las instrucciones solicitadas.

El resto del artículo se organiza como sigue: En la Sección 2 se introducen los BTs y los LLMs y se revisan sus aplicaciones en robótica. La Sección 3 detalla la metodología propuesta, incluyendo el proceso de generación y adaptación de BTs. La Sección 4 presenta los resultados experimentales y un ejemplo de una acción compleja como la búsqueda de una persona caída. La Sección 5 discute las contribuciones y ventajas del sistema, así como las áreas de mejora, y la Sección 6 completa con las conclusiones y el trabajo futuro.

2. Marco teórico y trabajos relacionados

2.1. Behavior Trees

Los Behavior Trees (BTs) son modelos estructurados de forma jerárquica invertida (véase Figura 1), diseñados para representar y gestionar tareas de forma eficiente (Colledanchise and Ögren, 2017). Originalmente desarrollados en el ámbito de los videojuegos para el control de PNJs (Personajes No Jugables) (Isla, 2005) como una solución más flexible frente a las máquinas de estados finitos o *Finite State Machines* (FSM) (Ben-Ari and Mondada, 2018), los BTs han sido adoptados ampliamente en robótica debido a su naturaleza modular, su facilidad de interpretación y su capacidad de adaptación. A diferencia de las FSM, los BTs ofrecen una estructura más sencilla de escalar y modificar, lo que los hace especialmente adecuados para sistemas complejos y dinámicos.

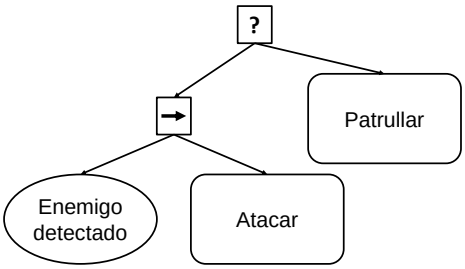


Figura 1: Ejemplo de un BT. Simula el comportamiento simplificado de un PNJ guardian en el videojuego Halo.

Recientes estudios destacan su adopción en sistemas autónomos complejos (Ghzouli et al., 2020), su versatilidad en planificación reactiva (Pezzato et al., 2020), y su capacidad para incorporar inferencia activa y sensores continuos (De la Cruz et al., 2020). Revisiones como las de Iovino et al. (2021) y Biggar et al. (2020) han sistematizado sus fundamentos y aplicaciones, mientras que Ghzouli et al. (2023) proporciona evidencia empírica de su efectividad frente a FSMs.

2.2. LLMs para tareas robóticas

La integración de modelos de lenguaje a gran escala (LLMs) en la generación automática de árboles de comportamiento ha emergido como una solución prometedora para abordar tareas robóticas complejas. Izzo et al. (2024) presentan BTGenBot, que utiliza LLMs compactos para generar BTs efectivos tras un ajuste fino específico. Lykov et al. (2023) introducen LLM-MARS, permitiendo diálogos dinámicos entre humanos y robots mediante LLMs ajustados desde Falcon 7B. Li et al. (2024) proponen una metodología innovadora que aprovecha las capacidades de representación y razonamiento de los LLMs para la generación automática de BTs. Ao et al. (2024) desarrollan LLM-as-BT-Planner, un marco que emplea LLMs para la planificación de tareas de ensamblaje robótico, mejorando la generación de BTs a través de métodos de aprendizaje en contexto.

2.3. Control de Robots Sociales mediante BTs y LLMs

La integración de modelos de lenguaje a gran escala en robots sociales ha permitido generar árboles de comportamiento adaptativos y expresivos. Mahadevan et al. (2024) exploran el uso de LLMs para generar comportamientos expresivos

en robots sociales, permitiendo una interacción más natural y comprensible para los usuarios. Wang et al. (2024) desarrollan SRLM, un sistema que combina LLMs y aprendizaje por refuerzo profundo para la navegación interactiva de robots sociales en entornos complejos. Por el contrario, Cao et al. (2023) presentan un enfoque de generación de tareas basado en BTs que utiliza LLMs y un diseño de prompt en fases para facilitar la generación jerárquica de tareas robóticas.

3. Metodología propuesta

El enfoque presentado combina el procesamiento de lenguaje natural con la generación de BTs por parte de LLMs, que posteriormente son ejecutados por un robot para llevar a cabo las acciones especificadas por el usuario. Este sistema ha sido concebido para su aplicación en entornos domésticos, especialmente en hogares de personas mayores, mediante el uso de un robot social que ofrezca compañía y responda a las órdenes emitidas por los residentes.

Aunque no es posible una comparación cuantitativa directa por las diferencias en configuraciones, dominios y criterios de evaluación, presentamos una evaluación cualitativa basada en las capacidades principales de cada sistema. LLM-as-BT-Planner destaca por su planificación jerárquica y uso de simulación, pero no maneja errores ni interactúa con el usuario. LLM-BRAIn genera BTs rápidamente con un modelo ligero, pero sus planes son estáticos y sin adaptación en ejecución. BTGenBot asegura corrección sintáctica y semántica con LLMs compactos, aunque solo opera offline y no responde a fallos ni consultas. LLM-BT permite cierta adaptación mediante mapas semánticos y un parser BERT, pero depende de plantillas predefinidas y no contempla la interacción social real ni retroalimentación del usuario. En cambio, nuestra propuesta integra comprensión del lenguaje natural, aclaraciones al usuario y adaptación en tiempo de ejecución, todo dentro de un sistema funcional y desplegado en un robot social real.

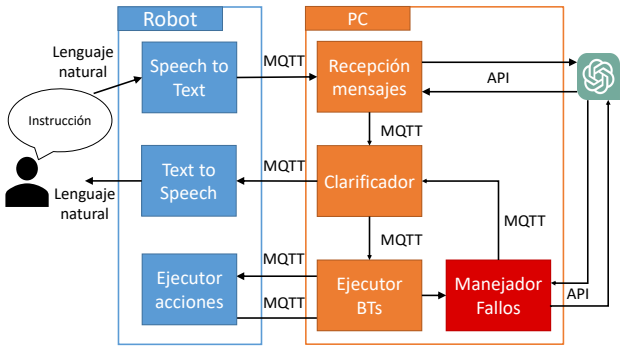


Figura 2: Representación gráfica del método propuesto. El marco azul muestra la parte del sistema ejecutada por el robot social, mientras que el naranja representa los módulos del ordenador.

La Figura 2 muestra la arquitectura general del sistema, diseñada para la generación y ejecución de BTs. Esta arquitectura está compuesta por un robot, un PC y un modelo LLM accesible a través de una API. Los componentes se comunican a través de un pipeline estructurado que permite interpretar comandos en lenguaje natural, traducirlos a BTs y ejecutar las tareas correspondientes. La principal diferencia respecto al trabajo presentado anteriormente corresponde al módulo de Manejador de Fallos y la realimentación al LLM cuyo fin es solucionar los errores producidos durante la ejecución de los BTs.

3.1. *Interpretación de órdenes en lenguaje natural*

El sistema se activa cuando el usuario emite una orden verbal, que es transcrita a texto mediante un módulo de reconocimiento de voz. Este texto es enviado al módulo de Recepción de Mensajes mediante MQTT (Message Queuing Telemetry Transport) (Mishra and Kertesz, 2020) y transmitido al modelo de lenguaje. El mensaje es procesado por ChatGPT 4 mediante un prompt estructurado y estático, compuesto por tres secciones principales: (1) directrices generales que instruyen al LLM a generar un Árbol de Comportamiento (BT) o solicitar una aclaración si la instrucción es ambigua o no ejecutable, (2) una descripción contextual del entorno y de las capacidades del robot, y (3) un conjunto de reglas específicas que definen la estructura y restricciones de los BTs generados. Este prompt es único para todas las instrucciones de entrada y no se modifica dinámicamente, lo que garantiza uniformidad en el comportamiento del modelo y facilita su escalabilidad sin necesidad de reconfiguración por tarea.

Este enfoque permite interpretar tanto instrucciones simples como órdenes complejas, sin necesidad de que el usuario detalle la lógica de ejecución, ya que el modelo infiere la intención del usuario y construye el BT.

3.2. *Clarificación, ejecución y adaptación de BTs*

Una vez que el modelo LLM ha generado una salida, distintos módulos del sistema intervienen para garantizar su correcta validación y ejecución.

Clarificador: Este módulo analiza la respuesta del LLM. Si el resultado es un BT válido, lo guarda en un archivo y notifica al Ejecutor de BTs que el plan está listo para su ejecución. En cambio, si el LLM indica que no ha comprendido la orden, que esta es ambigua o que la tarea solicitada no puede realizarse, el Clarificador genera una solicitud de aclaración al usuario. Esta interacción se transmite en lenguaje natural mediante el módulo Text-to-Speech del robot, asegurando así que el árbol generado refleja de forma precisa la intención del usuario.

Ejecutor de BTs: Al recibir la notificación de que hay un árbol disponible, este módulo inicia su ejecución como un subproceso independiente. Las acciones que debe realizar el robot se comunican mediante mensajes MQTT al módulo ejecutor del propio robot, que a su vez responde informando sobre el estado de cada acción (por ejemplo, si se ha completado, la ubicación actual, o la respuesta del usuario a una consulta). Al concluir la tarea, el Ejecutor de BTs finaliza el subproceso, elimina el archivo del BT y queda disponible para recibir nuevas órdenes. En caso de fallo —ya sea por errores en el código generado por el LLM o por un estado de Failure devuelto durante la ejecución— el subproceso se detiene, y el sistema envía un mensaje de error al Manejador de Fallos.

Manejador de Fallos: Este módulo garantiza la robustez y adaptabilidad del sistema durante la ejecución de tareas, abordando uno de los desafíos fundamentales en robótica: la

capacidad del robot para adaptarse ante situaciones en las que no puede completar una tarea como estaba prevista. Gracias a su integración con un LLM, el sistema combina razonamiento y generación creativa de soluciones. Si el BT devuelve un estado de *Failure* (cuando su código es semánticamente incorrecto u ocurre un fallo durante su ejecución), el Manejador de Fallos identifica el problema acontecido y consulta al LLM, proporcionándole el código del BT que ha fallado y el contexto específico del error (por ejemplo: "El robot no puede alcanzar la ubicación objetivo", "La ubicación objetivo no existe" o "Falta un argumento en la acción Ir"). Posteriormente, el módulo recibe la respuesta del LLM y la envía al Clarificador para asegurarse de que esta es un BT.

Esta capacidad de revisar dinámicamente los planes utilizando lenguaje natural permite al modelo interpretar y resolver problemas de forma flexible, dotando al robot de una capacidad de adaptación similar al razonamiento humano.

4. Experimentos y resultados

Se llevaron a cabo experimentos en un entorno de laboratorio con el objetivo de validar la viabilidad del sistema en contextos domésticos, específicamente orientados al cuidado de personas mayores. Para ello, se utilizó el robot social Temi (Temi, 2024), desplegado en un laboratorio adaptado para simular la distribución de una vivienda, con diferentes estancias representando habitaciones típicas de un hogar. En este entorno, se evaluó la capacidad del sistema para interpretar y ejecutar múltiples instrucciones formuladas en lenguaje natural.

La validación experimental se dividió en dos fases principales: primero, la generación y adaptación de BTs a partir de órdenes verbales, y segundo, la ejecución de tareas completas en el sistema completo con el robot real.

Se seleccionaron cinco tipos de acciones que el robot Temi es capaz de ejecutar para el desarrollo de las pruebas: *Hablar*, mediante la reproducción de un mensaje verbal; *Ir*, que permite al robot desplazarse de forma autónoma hacia una ubicación específica; *Preguntar*, donde formula una pregunta al usuario y almacena su respuesta; *Videollamada*, que activa una llamada de video con uno de los contactos registrados; y *Detección de caídas*, una función que obtiene la cantidad de personas caídas y no caídas presentes en la escena.

Para esta última funcionalidad, se integró una técnica basada en visión por computador y aprendizaje profundo, según la metodología descrita en (Sánchez-Girón et al., 2024). La cámara del robot captura una imagen del entorno, la cual es enviada mediante el protocolo MQTT a un modelo especializado en la detección de caídas, que devuelve el número de personas identificadas como caídas o no caídas en la escena.

4.1. Generación y adaptación de BTs

En un trabajo previo Merino-Fidalgo et al. (2025), ya se validó experimentalmente la capacidad del sistema para generar BTs a partir de instrucciones en lenguaje natural utilizando ChatGPT. En ese estudio, se demostró que el enfoque propuesto es capaz de traducir comandos expresados por el usuario en estructuras ejecutables por un robot social.

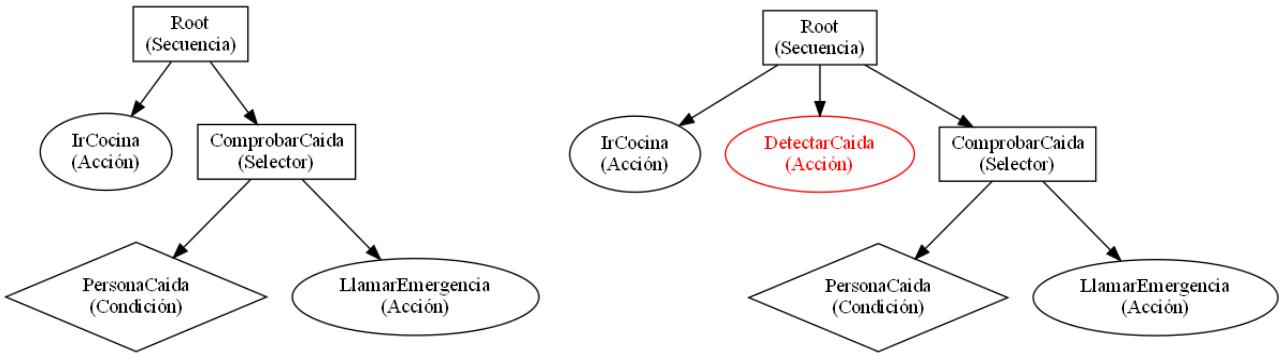
A partir de esa base, este apartado se enfoca principalmente en extender dicha capacidad, centrándose en la adaptación de los BTs. En lugar de evaluar nuevamente el proceso de generación, este estudio interpreta las causas de los fallos y, mediante la intervención del LLM, modifica dinámicamente el BT para replanificar o resolver el problema.

El procedimiento de evaluación de la adaptación de BTs comienza con la introducción de: (1) un prompt que establece que el objetivo del modelo es identificar errores en BTs proporcionados y generar una versión corregida del mismo; (2) una descripción textual del fallo ocurrido durante la ejecución del árbol; y (3) el código fuente del BT que ha fallado. Esta combinación de información permite al LLM razonar sobre el contexto del error, detectar posibles inconsistencias o estructuras incorrectas en el árbol, y devolver una versión funcional que corrija el problema reportado.

Los principales problemas encontrados en la generación del BTs del trabajo anterior corresponden a que el BT presenta errores semánticos en su código, su estructura lógica es incorrecta o que durante su ejecución se produce algún fallo, ya sea porque alguna de las acciones del robot no se encuentra disponible en ese momento por circunstancias del entorno (p. ej., no puede llegar a su destino porque hay un obstáculo en el camino).

En la Figura 3 se muestra un ejemplo de modificación de un BT. La instrucción en lenguaje natural introducida al LLM fue "*Vete a la cocina y si encuentras a alguien caído llama a emergencias*", cuya respuesta corresponde a la de la Figura 3(a), que se trata de un BT incorrecto, ya que devuelve error al comprobar si hay una persona caida puesto que no se ha realizado la detección. Al devolver *Failure*, entra en juego el Manejador de Fallos, que introduce en el LLM el código del BT generado inicialmente y el mensaje de error *Resultado de deteccion no encontrado*. La respuesta del modelo (Figura 3(b)) incluye el nodo resaltado en rojo de Detección de Caidas, que permite el correcto funcionamiento del BT.

En total se realizaron 80 experimentos para comprobar el funcionamiento del Manejador de Fallos. Esta evaluación se basó en 20 códigos de BTs distintos (generados previamente a partir de una instrucción en lenguaje natural) en los que en cada uno de ellos se han planteado cuatro situaciones de fallos como errores semánticos en el código, estructura lógica errónea o fallos simulados durante la ejecución. Posteriormente, se simuló en funcionamiento del sistema enviando un mensaje MQTT al Manejador de Fallos, que introdujo en el LLM el prompt de adaptación, la descripción del fallo planteado y el código del BT que contiene el error. 40 de esas pruebas corresponden a BTs de instrucciones simples —definidas como aquellas que requieren una o dos acciones—, que alcanzaron una tasa de éxito del 92.5 %, con tiempos de generación comprendidos entre 1.5 y 5 segundos. , lo que resultó en 37 ejecuciones correctas. En el caso de BTs de tareas complejas —aquellas que implican más de dos acciones o requieren la inclusión de nodos de control adicionales—, el número de casos exitosos fue de 35 sobre 40, que se traduce en un porcentaje de acierto del 87.5 % y un tiempo de generación que oscila entre 3 y 8 segundos. Esta latencia se considera aceptable ya que se trata de un proceso automático no supervisado, mucho más rápido que una revisión y modificación manual del BT. Los principales errores observados se debieron a que ocasinalmente el LLM malinterpretó la orden y preguntó por una aclaración al usuario cuando no era necesaria (suceso ocurri-



(a) Estructura inicial errónea de un BT.

(b) Estructura corregida de un BT.

Figura 3: Ejemplo de modificación de un BT.

Tabla 1: Resultados por tipo de instrucción.

Instrucción	**Adaptaciones exitosas**	**Total adaptaciones**	**Pruebas exitosas**	**Total pruebas**	**% Éxito**
Instrucciones simples	2	2	25	25	100 %
Instrucciones complejas	13	15	20	25	80 %
Instrucciones erróneas	9	10	22	25	88 %
Instrucciones imposibles	0	0	24	25	96 %
Total	24	27	91	100	91 %

do en dos ocasiones para BTs complejos), cambios en la estructura lógica del BT que no siempre solucionaban el error y dificultades eventuales para solucionar interferencias externas o del entorno que impedían la consecución del BT.

4.2. *Experimentos con el robot*

Con el fin de evaluar el desempeño del sistema, se realizó una prueba con 25 participantes (de entre 20 y 65 años) sin experiencia previa en el software donde, tras una breve explicación sobre qué hace el sistema y las capacidades del robot, se les solicitó que formularan cuatro instrucciones por voz distintas cumpliendo ciertos criterios específicos. La primera debía consistir en una tarea simple compuesta por dos acciones, como por ejemplo "*Vete al dormitorio y saluda a Miguel*". La segunda requería una tarea compleja con más de dos acciones e incluyendo, al menos, una condición lógica, como "*Ve al salón y pregunta a Laura si está viendo la televisión; si responde que sí, ve al dormitorio y dímelo; si no, dile que venga*". La tercera debía ser una tarea errónea, incorporando elementos no contemplados en el prompt, como ubicaciones inexistentes o contactos no registrados. Por último, se pidió una tarea imposible, que incluyera acciones fuera del alcance del robot, como recoger objetos (ya que el robot no tiene brazos) o abrir puertas.

Los resultados de los experimentos, evaluados por el autor de acuerdo a cómo debía comportarse nuestra propuesta en función de la instrucción introducida, se presentan en la Tabla 1, la cual refleja un desempeño sólido del sistema, con 91 casos de éxito sobre 100 instrucciones. Las tareas simples fueron resueltas con una precisión perfecta, mientras que las instrucciones complejas mostraron una ligera disminución en efectividad (80 %), en parte debido a su mayor carga lógica. En total, 24 de las 27 instrucciones que requirieron modificación fueron tratadas con éxito, lo que supone un porcentaje de éxito cercano al 89 %.

Al comparar con los resultados obtenidos en el trabajo previo realizado en el que no se contaba con ningún tipo de realimentación si el BT fallaba, la tasa global de éxito ha aumentado de un 82 % a un 91 %, lo que respalda la utilidad del módulo de manejo de fallos y del LLM en la adaptación dinámica del sistema ante problemas en la ejecución de los BTs.

5. Discusión

El sistema propuesto representa un nuevo marco en la planificación robótica al integrar modelos de lenguaje a gran escala no solo para la generación de árboles de comportamiento, sino también para su modificación y adaptación dinámica durante la ejecución. Esta capacidad de generar un BT modificado ante errores o situaciones imprevistas introduce una dimensión de resiliencia que tradicionalmente ha sido difícil de implementar con arquitecturas estáticas. El uso de lenguaje natural como interfaz de entrada permite una interacción intuitiva, especialmente útil para usuarios sin conocimientos técnicos. Además, el diseño estructurado del prompt y la incorporación del módulo Clarificador y del Manejador de Fallos refuerzan la robustez del sistema, facilitando la interpretación de instrucciones ambiguas o inválidas y la adaptación de BTs ante fallos o situaciones imprevistas.

No obstante, la fiabilidad del sistema depende en gran medida de la calidad de las respuestas del LLM, tanto en la generación inicial de BTs a partir de una instrucción en lenguaje natural como en la modificación y adaptación del código tras producirse un fallo, de tal forma que puede proporcionar respuestas que el sistema no es capaz de procesar. Asimismo, el nuevo módulo de tratamiento de fallos introduce una consulta remota del LLM, cuya latencia se traduce en un aumento del tiempo de ejecución de tareas cuando estas necesitan ser corregidas.

Por último, a medida que se incorporen más acciones y se incremente la complejidad de las instrucciones, será necesario explorar estrategias de optimización en la estructura del prompt y en la validación previa de los BTs generados, con el fin de mantener la escalabilidad y eficiencia del sistema.

6. Conclusiones

En este artículo, se ha presentado un marco innovador para la generación, ejecución y adaptación dinámica de Árboles de Comportamiento (BTs) mediante el uso de Modelos de Lenguaje a Gran Escala (LLMs) en tareas robóticas. La propuesta combina la modularidad y claridad estructural de los BTs con las capacidades de interpretación y razonamiento de los LLMs, lo que permite que un robot social ejecute tareas complejas expresadas en lenguaje natural con un alto grado de autonomía y flexibilidad. A través del módulo de Clarificación, el sistema puede gestionar órdenes ambiguas o no ejecutables, previniendo fallos en la ejecución. Asimismo, se introduce un mecanismo de modificación basado en LLMs que permite monitorizar la ejecución del BT y adaptar su código cuando se detectan errores, lo que aumenta significativamente la resiliencia del sistema ante imprevistos. Los resultados experimentales evidencian la eficacia del enfoque tanto en la generación inicial de planes como en su capacidad de adaptación, demostrando su aplicabilidad práctica en entornos no estructurados como una vivienda.

El trabajo futuro abordará varias líneas de desarrollo. En primer lugar, se planea la integración de entradas multimodales, como datos de sensores del entorno, para mejorar la percepción contextual del sistema. En segundo lugar, se explorarán métodos de optimización en la generación y validación de BTs, ampliando el repertorio de acciones disponibles para cubrir tareas más complejas, como enviar correos electrónicos o reconocer objetos mediante visión artificial. Finalmente, se contempla la evaluación del sistema en escenarios reales, mediante su instalación en hogares de personas mayores, lo que permitirá validar su rendimiento en condiciones de uso cotidiano.

Agradecimientos

La investigación que se presenta en este trabajo ha recibido financiación del proyecto ROSOGAR PID2021-123020OB-I00 financiado por MCIN/ AEI / 10.13039/501100011033 / FEDER, UE, y del proyecto EIAROB Financiado por Consejería de Familia de la Junta de Castilla y León - Next Generation EU.

Referencias

Ao, J., Ren, Z., Su, H., Zhu, J., 2024. Llm-as-bt-planner: Large language models for behavior tree based robotic task planning. arXiv preprint arXiv:2402.08013.

Ben-Ari, M., Mondada, F., 2018. Finite state machines. Elements of Robotics, 55–61.
DOI: 10.1007/978-3-319-62533-1_4

Biggar, O., Zamani, M., Shames, I., 2020. A principled analysis of behavior trees and their generalisations. arXiv preprint arXiv:2008.11906.

Brand, D., Zafiropulo, P., 1983. On communicating finite-state machines. Journal of the ACM (JACM) 30 (2), 323–342.

Cao, Y., Rajeswaran, A., Grover, A., Feinberg, V., 2023. Robot task planning with large language models via behavior tree generation. arXiv preprint arXiv:2310.06796.

Chang, Y., Wang, X., Wang, J., Wu, Y., Yang, L., Zhu, K., Chen, H., Yi, X., Wang, C., Wang, Y., Ye, W., Zhang, Y., Chang, Y., Yu, P. S., Yang, Q., Xie, X., 3 2024. A survey on evaluation of large language models. ACM Transactions on Intelligent Systems and Technology 15, 39.
DOI: 10.1145/3641289

Colledanchise, M., Ögren, P., 8 2017. Behavior trees in robotics and ai: An introduction. Behavior Trees in Robotics and AI.
URL: http://arxiv.org/abs/1709.00084http://dx.doi.org/10.1201/9780429489105
DOI: 10.1201/9780429489105

De la Cruz, P., Piater, J., Saveriano, M., 2020. Reconfigurable behavior trees: Towards an executive framework meeting high-level decision making and control layer features. arXiv preprint arXiv:2007.10663.

Deng, J., Lin, Y., 1 2022. The benefits and challenges of chatgpt: An overview. Frontiers in Computing and Intelligent Systems 2, 81–83.
DOI: 10.54097/FCIS.V2I2.4465

Ghzouli, R., Berger, T., Johnsen, E. B., Dragule, S., Wasowski, A., 2020. Behavior trees in action: A study of robotics applications. In: Proceedings of the 13th ACM SIGPLAN International Conference on Software Language Engineering. ACM, pp. 127–140.
DOI: 10.1145/3426425.3426942

Ghzouli, R., Berger, T., Johnsen, E. B., Dragule, S., Wasowski, A., 2023. Behavior trees and state machines in robotics applications. IEEE Transactions on Software Engineering 49 (1), 1–14.
DOI: 10.1109/TSE.2021.3115347

Iovino, M., Scukins, E., Styrud, J., Ögren, P., Smith, C., 2021. A survey of behavior trees in robotics and ai. Robotics and Autonomous Systems 136, 103710.
DOI: 10.1016/j.robot.2020.103710

Isla, D., 2005. Handling complexity in the halo2 ai. Game Developers Conference.

Izzo, S., Ardulov, V., Scioni, E., Mohan, R., Cheng, X., He, Y., Hager, G. D., 2024. Btgenbot: Behavior tree generation using compact language models. arXiv preprint arXiv:2402.00560.

Li, M., Lin, L., Lin, Z., Li, X., Du, Y., 2024. A study of behavior tree generation from natural language with large language models. arXiv preprint arXiv:2403.12658.

Lykov, A., Ravichandar, H., Klee, S., Park, D., Nagarajan, P., Troniak, D., Liu, Z., Jenkins, R., Khante, A., Chai, J. Y., et al., 2023. Llm-mars: Leveraging large language models for multi-turn, multi-modal, multi-scene human-robot interaction. arXiv preprint arXiv:2310.02291.

Mahadevan, R., Wang, S., Thomaz, A. L., 2024. Generative language models for expressive robot social behaviors. arXiv preprint arXiv:2402.09013.

Merino-Fidalgo, S., Casanova, E. Z., García-Bermejo, J. G., Domingo, J. D., 6 2025. Generación de behavior trees mediante llm para el control de un robot social. Simposios del Comité Español de Automática (CEA) 1.
URL: https://ingmec.ual.es/ojs/index.php/RBVM25/article/view/10

Mishra, B., Kertesz, A., 2020. The use of mqtt in m2m and iot systems: A survey. IEEE Access 8, 201071–201086.
DOI: 10.1109/ACCESS.2020.3035849

OpenAI, 2024. Introducing chatgpt — openai. Accessed: 26-12-2024.
URL: https://openai.com/index/chatgpt/

Pezzato, C., Hernandez Corbato, C., Bonhof, S., Wisse, M., 2020. Active inference and behavior trees for reactive action planning and execution in robotics. arXiv preprint arXiv:2011.09756.

Sánchez-Girón, C., Gómez, M. G., Domingo, J. D., García-Bermejo, J. G., Casanova, E. Z., 2024. Integración convnext-yolo mediante cvv para detectar caídas en robot social. Jornadas de Automática.
DOI: https://doi.org/10.17979/jacea.2024.45.10788

Temi, 2024. Introducing temi robot v3. Accessed: 03-01-2025.
URL: https://www.robotemi.com/product/temi-sales-contact/

Wang, T., Patel, R., Qian, Y., Mataric, M. J., 2024. Srlm: Social robot language modeling for goal-conditioned navigation. arXiv preprint arXiv:2401.11317.

Ögren, P., Sprague, C. I., 5 2022. Behavior trees in robot control systems. Annual Review of Control, Robotics, and Autonomous Systems 5, 81–107.
DOI: 10.1146/ANNUREV-CONTROL-042920-095314/CITE/REFWORKS