Simposio CEA de Robótica, Bioingeniería, Visión Artificial y Automática Marina 2025

RBVM
Simposio CEA de Robótica, Bioingeniería, Visión por Computador y Automática Marina

Generación de behavior trees mediante LLM para el control de un robot social

Merino-Fidalgo, S.[a,*], Zalama, E.[a,b], Gómez-García-Bermejo, J.[a,b], Duque-Domingo, J.[a]

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

Resumen

Los árboles de comportamiento (Behavior Trees), utilizados como estructuras para controlar robots sociales, suelen ser diseñados manualmente por expertos y adaptados a situaciones específicas. Esta metodología limita la flexibilidad y la capacidad de personalización del comportamiento del robot en tiempo real, lo que dificulta su adaptación a contextos dinámicos y variados.

Este artículo presenta un sistema innovador para el control de robots sociales basado en la generación automática de árboles de comportamiento a partir de instrucciones en lenguaje natural. Utilizando un modelo de lenguaje grande (LLM), las instrucciones humanas son interpretadas y traducidas en una estructura jerárquica de acciones y decisiones, que el robot ejecuta para interactuar de forma coherente y natural con usuarios sin conocimientos técnicos avanzados. Esta propuesta reduce la necesidad de programación manual y favorece una mayor personalización y adaptabilidad en la interacción humano-robot. A modo de experimentación se exponen casos de uso donde el robot ejecuta tareas como desplazarse a una ubicación y reproducir un mensaje, o iniciar una videollamada al detectar que una persona ha caído, todo ello generado dinámicamente a partir de instrucciones simples en lenguaje natural.

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 grandes, Behavior Trees

Generation of behavior trees using LLM for the control of a social robot.

Abstract

Behavior Trees, used as structures to control social robots, are usually designed manually by experts and adapted to specific situations. This methodology limits the flexibility and the ability to customize the robot's behavior in real time, making it difficult to adapt to dynamic and varied contexts.

This paper presents an innovative system for the control of social robots based on the autonomous generation of behavior trees from natural language instructions. Using a large language model (LLM), human instructions are interpreted and translated into a hierarchical structure of actions and decisions, which the robot executes to interact in a coherent and natural way with users without advanced technical knowledge. This approach reduces the need for manual programming, and favors greater customization and adaptability in human-robot interaction. Use cases are presented where the robot executes tasks such as moving to a location and playing a message, or initiating a video call when it detects that a person has fallen, all dynamically generated from simple instructions in natural language.

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 sido un reto clave en sistemas autónomos, especialmente en robótica. A medida que estos sistemas se vuelven más complejos, es necesaria una gestión más robusta y flexible. Las conocidas máquinas de estados finitos (FSM) (Brand and Zafiropulo, 1983) resultan limitadas debido a su incapacidad para representar sistemas con comportamientos no lineales o dependientes del contexto, así como por su falta de memoria estructurada, por lo que han sido relegadas

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

al control de sistemas embebidos o aplicaciones con requisitos deterministas estrictos.

Los Árboles de Comportamiento (BTs) (Ögren and Sprague, 2022), usados inicialmente en videojuegos, ofrecen modularidad y transparencia en tareas complejas mediante estructuras jerárquicas de condiciones y acciones. Sin embargo, su programación requiere conocimiento técnico.

Por otro lado, los Modelos de Lenguaje de Gran Tamaño (LLMs), como ChatGPT (Chang et al., 2024; OpenAI, 2024; Deng and Lin, 2022), han emergido como una herramienta destacada en comprensión y razonamiento en lenguaje natural. Estas capacidades abren la posibilidad de explorarlos como asistentes en la generación automática de código, llegando a ser capaces de generar el código completo de los BTs con el prompt adecuado. Sin embargo, integrar BTs y LLMs presenta desafíos técnicos importantes. Los Árboles de Comportamiento, aunque modulares, requieren una programación minuciosa para gestionar adecuadamente eventos inesperados, lo cual demanda experiencia en diseño de arquitecturas reactivas. Por su parte, los Modelos de Lenguaje de Gran Tamaño carecen de una comprensión semántica profunda como los humanos y pueden generar respuestas erróneas.

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. La metodología se basa en el uso de ChatGPT para interpretar comandos en lenguaje natural proporcionados por el usuario y traducirlos en estructuras de BTs que luego son ejecutadas por un robot social orientado a la atención y cuidado de personas mayores. Además, el sistema incorpora el módulo Clarificador, encargado de notificar al usuario cuando las instrucciones no son entendidas o no pueden ser llevadas a cabo por el robot debido a limitaciones operativas o ambigüedades. Este enfoque permite que los robots no solo sean capaces de ejecutar en tiempo real las instrucciones de lo usuarios, sino también que mantengan una comunicación efectiva con los usuarios, mejorando así la robustez y usabilidad del sistema. El objetivo último de esta propuesta es su instalación en casas de personas mayores, de tal forma que usuarios sin conocimientos técnicos sean capaces de comunicarse con el robot y que este sea capaz de ejecutar las instrucciones requeridas por los ancianos.

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 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. Árboles de Comportamiento (Behavior Trees)

Los Árboles de Comportamiento (BTs) son estructuras jerárquicas invertidas (Figura 1) que representan tareas y permiten conmutar entre ellas (Colledanchise and Ögren, 2017). Surgieron en videojuegos (Isla, 2005) como alternativa a las máquinas de estados finitos (FSM) (Ben-Ari and Mondada, 2018), pero se han extendido a la robótica por su adaptabilidad, claridad y modularidad, sustituyendo a las FSM, más difíciles de modificar y poco flexibles.

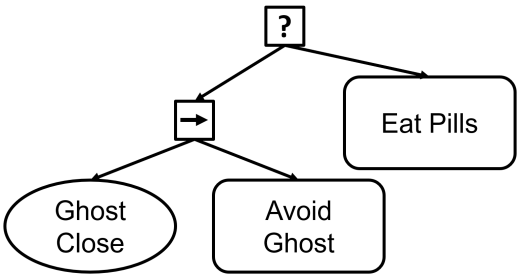

Figura 1: Ejemplo de un BT. Simula el comportamiento básico de Pacman, de huir si hay un fantasma cerca o en caso contrario comer puntos.

En un BT, la ejecución parte de la raíz y propaga 'ticks' hacia los nodos hijos. Cada nodo devuelve *Éxito*, *Fracaso* o *En ejecución*. Existen:

- **Nodos de Control**:
  - *Secuencia (→)*: ejecuta nodos de izquierda a derecha hasta que uno falle. Solo retorna Éxito si todos lo hacen.
  - *Selector (?)* : ejecuta nodos hasta que uno tenga éxito. Solo retorna Fracaso si todos fallan.
- **Nodos de Ejecución**:
  - *Acción*: ejecutan tareas específicas.
  - *Condición*: evalúan condiciones booleanas.

Los BTs se usan cada vez más en robótica por su transparencia y escalabilidad (Iovino et al., 2022), permitiendo incorporar nuevos comportamientos sin afectar los existentes, especialmente en entornos dinámicos como robots móviles (Pezzato et al., 2023), tareas de manipulación (Pezzato et al., 2023) y HRI (Axelsson and Skantze, 2019).

2.2. LLMs para tareas robóticas

La integración de Modelos de Lenguaje Grandes (LLMs) en robótica ha mejorado la planificación y ejecución de tareas. Modelos como GPT (Kalyan, 2024) pueden interpretar lenguaje natural, razonar y generar salidas estructuradas, lo que los hace ideales para generar BTs en entornos dinámicos.

Modelos tempranos como Palm-E combinan visión, lenguaje y acción para tareas embebidas (Driess et al., 2023), mientras que RT-2 transfirió conocimiento web a robots mediante modelos VLA (Brohan et al., 2023).

En la generación de BTs para aplicaciones robóticas destacan:

- **LLM-BRAIn**: basado en Alpaca 7B, genera BTs estáticos desde texto (Lykov et al., 2023).
- **BTGenBot**: usa LLMs ligeros con un dataset de BTs existentes (Izzo et al., 2024).
- **LLM-BT**: combina ChatGPT y mapas semánticos generados por reconocimiento de objetos para tareas adaptativas en simulación (Zhou et al., 2024).

- **LLM as BT-Planner**: aplica LLMs en planificación de ensamblaje con aprendizaje contextual (Ao et al., 2024).

2.3. *Robots Sociales*

Los robots sociales deben interpretar señales humanas, comunicarse naturalmente y adaptarse al contexto. Los BTs son útiles en HRI por su capacidad de adaptación. Cooper et al. diseñaron un BT para interacciones sociales a largo plazo (Cooper and Lemaignan, 2022).

Los LLMs permiten interacciones verbales naturales y variadas. Saleshat, por ejemplo, usa un robot comercial y ChatGPT para controlar tanto el diálogo como la apariencia física (Hanschmann et al., 2024).

No obstante, los BTs no capturan toda la complejidad del comportamiento humano, y los LLMs suelen limitarse al lenguaje. Por eso proponemos combinar ambos: usar LLMs para la comprensión del lenguaje natural y los BTs para ejecutar acciones estructuradas, más allá de la conversación (Kim et al., 2024).

3. Metodología propuesta

El método propuesto integra el procesamiento de lenguaje natural para gestionar órdenes que derivan en la generación de árboles de comportamiento por parte de LLMs, los cuales son posteriormente ejecutados por un robot, realizando las acciones previamente indicadas. Este sistema está pensado para ser implementado en hogares de personas mayores, mediante un robot social que proporcione compañía y ejecute las órdenes dadas por los residentes.

La arquitectura del sistema propuesto para generar y ejecutar BTs se ilustra en la Figura 2. El sistema integra un robot, un PC y un LLM al que se accede a través de una API. Cada componente interactúa a través de un pipeline estructurado que facilita la comprensión del lenguaje natural, la generación de BTs y la ejecución de tareas. Las partes principales del sistema se describen a continuación:

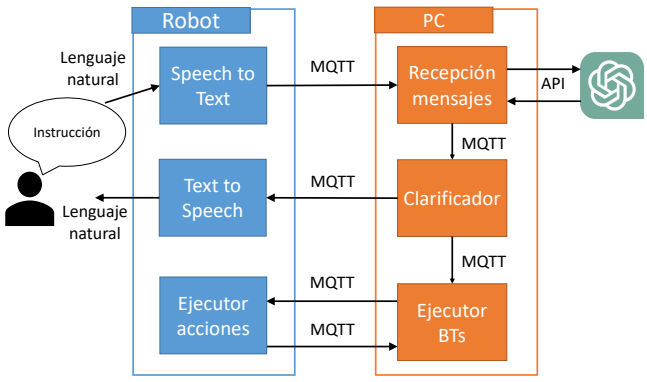

Figura 2: Representación gráfica del método propuesto. El marco azul muestra la parte del sistema ejecutada por el robot social, mientras que el naranja representa los módulos del ordenador.

3.1. *Interacción en lenguaje natural*

El proceso comienza cuando el usuario emite una orden de tarea en lenguaje natural. Esta entrada se procesa en el robot a través de un módulo Speech-to-Text, que convierte la orden por voz en un formato textual. A continuación, este texto pasa al módulo de Recepción de Mensajes para su preprocesamiento mediante mensajes MQTT (Message Queuing Telemetry Transport) (Mishra and Kertesz, 2020), protocolo usado para las comunicaciones del sistema.

3.2. *Interpretación de la orden y generación del BT*

El mensaje es recibido por el módulo de Recepción de Mensajes. Este bloque extrae la información del mensaje y la envía al LLM vía API. Utilizamos ChatGPT 4 con un prompt estructurado que consta de tres partes principales:

- Una parte básica, común a cualquier escenario, define el comportamiento del LLM, que debe generar un árbol de comportamiento o pedir al usuario que aclare la instrucción si no la entiende o si la tarea es imposible debido a que el robot no puede realizar las acciones ordenadas.
- Una descripción del entorno, las acciones que el robot puede realizar y la interacción entre el sistema y el robot. Esta información aporta contexto al modelo de lenguaje, lo cual facilita la obtención de resultados más precisos y contribuye a la detección de posibles errores.
- Reglas críticas de funcionamiento, donde definimos aspectos clave de los árboles de comportamiento, como estructuras fijas para un nodo específico, detalles de los nodos de acción o cuándo finaliza el BT.

Gracias a este prompt, el LLM puede generar BTs a partir de todo tipo de instrucciones, desde las más simples que especifican directamente las acciones que debe realizar el robot (*"Ve a mi habitación y llama a María"*) hasta órdenes más complejas como *"Dime si hay alguien en la cocina, quiero lavar los platos"*, en las que ChatGPT extrae las acciones que debe realizar el robot (ir a la cocina, buscar si hay alguien y luego regresar para informar si hay una persona o no) y genera el BT correspondiente o, si no entiende la instrucción, pide una aclaración al usuario. No es necesario incluir la lógica de ejecución en las instrucciones, ya que el LLM es capaz de comprenderla a partir del lenguaje natural y traducirla a la estructura del BT.

3.3. *Clarificación y ejecución del BT*

A continuación, explicamos cada uno de los módulos del sistema que intervienen una vez que el LLM genera una salida.

Clarificador: Este módulo recibe y analiza la respuesta generada por el LLM. Si la salida es el código de un BT, el módulo lo guarda en un archivo y envía un mensaje al Ejecutor para indicar que hay un plan pendiente de ejecutar. Sin embargo, si el LLM responde que no entendió lo que el usuario desea, que la instrucción es ambigua o que la tarea solicitada es imposible de realizar, el Clarificador envía un mensaje para pedir una aclaración o sugerir posibles interpretaciones al usuario mediante lenguaje natural, gracias al módulo Text-to-Speech del robot. Esto garantiza que el BT generado se corresponde con la intención del usuario.

Ejecutor de BTs: El Ejecutor de BTs recibe el mensaje de que hay un plan listo para ejecutar e inicia la ejecución del árbol de comportamiento como un subproceso. Las acciones que debe realizar el robot se envían mediante mensajes MQTT al módulo Ejecutor de acciones del robot, y de igual forma este

responde informando si ha finalizado una acción, en qué posición se encuentra o cuál fue la respuesta del usuario a una pregunta. Al finalizar la ejecución del árbol, el Ejecutor de BTs termina, elimina el subproceso y borra el archivo del BT quedando listo para ejecutar la siguiente tarea. Si la ejecución del árbol de comportamiento falla (el subproceso devuelve que no se ha ejecutado con éxito), ya sea por un error en el código generado por el LLM o porque durante la ejecución se ha devuelto un estado de *Failure*, el Ejecutor detiene el subproceso e informa al usuario del error acontecido mediante un mensaje de voz.

4. Experimentos y resultados

Se han realizado experimentos en un laboratorio para demostrar que el sistema puede implementarse en hogares para personas mayores. Se ha utilizado un robot social Temi (Temi, 2024), el cual fue desplegado en dicho laboratorio que fue dividido en habitaciones para recrear la distribución de una vivienda, y se probaron múltiples instrucciones en lenguaje natural.

Para verificar nuestra propuesta, se llevaron a cabo varios experimentos divididos en dos partes: la generación de árboles de comportamiento tras introducir una orden en lenguaje natural, y la prueba final que incluye la ejecución de tareas en un robot social real, como se muestra en la Figura 4.

Se eligieron cinco acciones diferentes que Temi puede realizar para los experimentos: *Hablar*, donde el robot reproduce un mensaje; *Ir*, que hace que el robot navegue hasta una ubicación objetivo; *Preguntar*, que formula una pregunta al usuario y guarda su respuesta; *Videollamada*, que inicia una videollamada con uno de los contactos guardados; y *Detección de caída*, que se utiliza para comprobar si una persona se ha caído, tomando una foto, analizándola y devolviendo el número de personas caídas y no caídas. Para detectar personas caídas en una imagen, se ha implementado una metodología basada en aprendizaje profundo y procesamiento de imágenes (Sánchez-Girón et al., 2024). La imagen capturada por la cámara del robot se envía al modelo de detección de caídas a través de MQTT.

4.1. Reconocimiento de lenguaje natural y generación de BTs

En esta fase, se evaluó la capacidad del sistema para generar correctamente árboles de comportamiento interpretables a partir de comandos en lenguaje natural. El proceso se iniciaba proporcionando una descripción de la tarea; en esta prueba, dicha descripción consistió en una instrucción escrita introducida directamente al sistema en vez de un comando de voz al robot, que era interpretada por el LLM para generar el código correspondiente de un árbol de comportamiento. Algunas de las órdenes introducidas fueron:

Temi, ve a la cocina y di "Hola, me llamo Temi"
¿Puedes ir al garaje y, si hay alguien, llamar a David?
Tráeme mi teléfono móvil, está en mi habitación

Los resultados demostraron la capacidad del sistema para generar árboles de comportamiento de forma eficiente, resolviendo todas las aclaraciones en un máximo de dos iteraciones en promedio. Por ejemplo, la primera orden fue identificada rápidamente y se generó el BT de manera ágil y correcta, cuyo código resultó perfectamente ejecutable.

Para la segunda orden, se introdujo una ubicación que no estaba registrada en la información de la casa, por lo que el robot no pudo navegar hacia ella. En consecuencia, el LLM reportó este problema al módulo Clarificador, que comunicó al usuario el mensaje: "*La ubicación solicitada no existe*". Tras seleccionar el "salón" como nueva ubicación objetivo, ChatGPT generó el BT mostrado en la Figura 3.

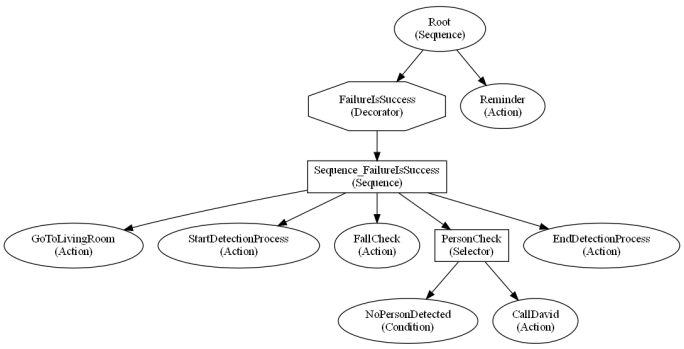

Figura 3: Representación gráfica del BT generado para la segunda instrucción.

El último ejemplo incluye una acción que el robot no puede realizar (recoger y traer objetos), por lo que LLM respondió a través del módulo Clarificador señalando este problema. Y como se trata de algo que el robot no puede realizar, no se generó ningún BT. Los BTs generados para tareas sencillas —definidas como aquellas que requieren una o dos acciones— alcanzaron una tasa de éxito del 85,3 % con un tiempo de generación entre 2 y 5 segundos. Esta evaluación se basó en la ejecución de 30 instrucciones distintas, cada una probada en cinco ocasiones, resultando en 128 ejecuciones correctas de un total de 150 intentos. En el caso de tareas complejas —que implican más de dos acciones o la inclusión de nodos de control adicionales—, el porcentaje de aciertos se redujo al 76 %, con 114 ejecuciones exitosas sobre el mismo número total de pruebas. El tiempo de generación aumentó considerablemente, con un rango entre 4 y 10 segundos, de gran amplitud debido a la variedad de posibles instrucciones. Los fallos se debieron principalmente a que la lógica del BT implementada por el LLM no se correspondía con la instrucción proporcionada o a que el código generado presentaba errores sintácticos, haciendo imposible su ejecución.

4.2. Experimentos con el robot

Para evaluar el sistema, se invitó a 25 usuarios sin conocimientos del software a dar instrucciones al robot, tras una breve guía sobre las acciones que el robot puede realizar. Se les pidió que formularan cuatro instrucciones diferentes con los siguientes requisitos: una tarea simple de dos acciones, como *"Di hola y luego ve al baño"*; una tarea compleja con más de dos acciones e introduciendo una condición, por ejemplo, *"Ve al salón y pregunta a Laura si está viendo la televisión, si responde que sí, ven al dormitorio y dímelo; de lo contrario, dile que venga"*; una tarea errónea, que incluya argumentos para acciones no definidas en el prompt, como una ubicación inexistente o contactos no guardados; y una tarea imposible que incluya acciones que el robot no puede realizar, como coger objetos o grabar un vídeo.

(a) Temi preguntando al usuario.

(b) Temi comprobando si el usuario está caido.

Figura 4: Robot Temi realizando distintas acciones.

Los resultados de los experimentos se presentan en la Tabla 1. La columna *Aclaradas* indica la eficacia del mecanismo de clarificación cuando una instrucción inicial no puede ser procesada correctamente por el modelo. Esto ocurre en situaciones en las que el LLM no comprende la orden, la localización de destino o el contacto especificado no existen, o bien la acción solicitada no es ejecutable por el robot. En tales casos, el modelo solicita al usuario que aclare o reformule la instrucción. El valor se presenta en formato *x*/*y*, donde *y* representa el número total de experimentos en los que se requirió algún tipo de aclaración, y *x* corresponde al número de casos en que, tras dicha interacción, el LLM logró generar correctamente el BT. Por ejemplo, si una instrucción fue aclarada tres veces hasta que el modelo generó un BT válido, se contaría como un único éxito de clarificación (1/1) en la tabla Finalmente, la columna *General* ilustra las pruebas exitosas de cada tipo de instrucción sobre el total de 25. Las tareas se dividieron en simples (máximo dos acciones y un nodo de control), complejas (más de dos acciones o más de un nodo de control), erróneas (destinos o contactos de videollamada distintos a los posibles) e imposibles (acciones que el robot no puede realizar).

Tabla 1: Resultados por tipo de tarea

Tarea	**Aclaradas**	**General**
Tarea simple	2/2	20/25
Tarea compleja	7/11	17/25
Tarea errónea	23/25	21/25
Tarea imposible	24/25	24/25
Total	56/63	82/100

La tasa de éxito general del sistema alcanzó un 82 % tras la realización de 100 pruebas. En total, se identificaron 18 casos en los que no se ejecutaron la orden introducida, clasificados según su causa principal. En siete de ellos se debieron a instrucciones erróneas o imposibles de realizar que fueron repetidas por el usuario; en esos casos, el sistema dejó de solicitar aclaraciones y no generó BT. Otros dos casos ocurrieron cuando la lógica de la tarea expresada era ambigua o difícil de interpretar, lo que derivó en BTs que no cumplían con la intención original del usuario. Sin embargo, la mayor parte de las ocasiones (nueve en total) estuvo relacionado con problemas en la generación de código por parte del modelo de lenguaje. Estos incluyeron la creación de nodos repetidos, lógicas imposibles o errores de sintaxis dentro de los BTs, afectando directamente su ejecución.

5. Discusión

El sistema propuesto demuestra un enfoque novedoso para la generación y ejecución de árboles de comportamiento en robots, aprovechando un modelo de lenguaje de gran escala para la planificación de tareas en tiempo real. Esta integración del entendimiento del lenguaje natural y la planificación mediante BTs introduce nuevas perspectivas sobre el uso de BTs en robótica e inteligencia artificial.

El uso del lenguaje natural como entrada simplifica la utilización del sistema, especialmente para personas sin conocimientos técnicos, como personas mayores. Un prompt diseñado meticulosamente, basado en reglas estrictas y restringido a algunas de las acciones que el robot puede realizar, asegura una salida del LLM considerablemente fiable. La inclusión del módulo Clarificador cuando el sistema no entiende lo que el usuario quiere, incrementa la robustez del sistema.

Sin embargo, también presenta ciertos desafíos y áreas para futuras investigaciones. El rendimiento del sistema depende en gran medida de la calidad y precisión de las respuestas del LLM. Aunque el LLM se desempeña muy bien generando BTs para tareas bien estructuradas gracias al prompt cuidadosamente diseñado, el modelo puede producir planes erróneos o con fallos en su código. Además, la integración de un LLM como ChatGPT a través de la API para la planificación de tareas introduce cierta latencia, la cual conlleva tiempos de espera considerables hasta el inicio de la ejecución del BT para tareas complejas con numerosos nodos.

En este primer enfoque se utilizaron algunas de las acciones que el robot puede realizar pero, a medida que las tareas se vuelvan más complejas y se añadan más acciones, el tamaño y dificultad de la lógica de los BTs aumentarán, lo que podría dificultar su interpretación y ejecución, y requeriría un rediseño del prompt generador.

6. Conclusiones

En este artículo, presentamos un marco novedoso para la generación y ejecución de Árboles de Comportamiento (BTs) utilizando Modelos de Lenguaje de Gran Escala (LLMs) para la ejecución de tareas robóticas. Nuestro enfoque combina la interpretabilidad y modularidad de los BTs con las capacidades de razonamiento de los LLMs, permitiendo que los robots realicen tareas complejas y dinámicas con un alto grado de adaptabilidad. Gracias al módulo de Clarificación, el sistema es capaz de manejar instrucciones ambiguas o imposibles de realizar, lo que evita la ejecución de planes erróneos. La validación experimental demostró la robustez y flexibilidad del método propuesto en una variedad de escenarios, lo que resalta su potencial para aplicaciones reales en robótica.

El trabajo futuro se centrará en varias áreas clave. En primer lugar, se pretende integrar fuentes de entrada multimodales, como datos de sensores del hogar, para mejorar la conciencia situacional del robot. En segundo lugar, se explorarán técnicas para optimizar el proceso de generación de BTs, añadiendo más acciones que permitan ejecutar tareas más complejas y mejorando la escalabilidad. Además, se plantea la introducción de un módulo capaz de modificar el BT cuando este falle, de tal forma que se solucione el error acontecido en la programación para completar la instrucción del usuario. Finalmente, se planea instalar y evaluar el sistema en entornos reales, con un despliegue supervisado en múltiples hogares reales de personas mayores, a fin de verificar su rendimiento en condiciones reales.

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., Wu, F., Wu, Y., Swikir, A., Haddadin, S., 9 2024. Llm as bt-planner: Leveraging llms for behavior tree generation in robot task planning. arXiv.

Axelsson, N., Skantze, G., 2019. Modelling adaptive presentations in human-robot interaction using behaviour trees. SIGDIAL 2019 - 20th Annual Meeting of the Special Interest Group Discourse Dialogue - Proceedings of the Conference, 345–352.
DOI: 10.18653/V1/W19-5940

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

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

Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Chen, X., Choromanski, K., Ding, T., Driess, D., Dubey, A., Finn, C., Florence, P., Fu, C., Arenas, M. G., Gopalakrishnan, K., Han, K., Hausman, K., Herzog, A., Hsu, J., Ichter, B., Irpan, A., Joshi, N., Julian, R., Kalashnikov, D., Kuang, Y., Leal, I., Lee, L., Lee, T. W. E., Levine, S., Lu, Y., Michalewski, H., Mordatch, I., Pertsch, K., Rao, K., Reymann, K., Ryoo, M., Salazar, G., Sanketi, P., Sermanet, P., Singh, J., Singh, A., Soricut, R., Tran, H., Vanhoucke, V., Vuong, Q., Wahid, A., Welker, S., Wohlhart, P., Wu, J., Xia, F., Xiao, T., Xu, P., Xu, S., Yu, T., Zitkovich, B., 7 2023. Rt-2: Vision-language-action models transfer web knowledge to robotic control. Proceedings of Machine Learning Research 229.

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

Cooper, S., Lemaignan, S., 2022. Towards using behaviour trees for long-term social robot behaviour. ACM/IEEE International Conference on Human-Robot Interaction 2022-March, 737–741.
DOI: 10.1109/HRI53351.2022.9889662

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

Driess, D., Xia, F., Sajjadi, M. S., Lynch, C., Chowdhery, A., Ichter, B., Wahid, A., Tompson, J., Vuong, Q., Yu, T., Huang, W., Chebotar, Y., Sermanet, P., Duckworth, D., Levine, S., Vanhoucke, V., Hausman, K., Toussaint, M., Greff, K., Zeng, A., Mordatch, I., Florence, P., 3 2023. Palm-e: An embodied multimodal language model. Proceedings of Machine Learning Research 202, 8469–8488.

Hanschmann, L., Gnewuch, U., Maedche, A., 2024. Saleshat: A llm-based social robot for human-like sales conversations. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 14524 LNCS, 61–76.
DOI: 10.1007/978-3-031-54975-5_4

Iovino, M., Scukins, E., Styrud, J., Ögren, P., Smith, C., 8 2022. A survey of behavior trees in robotics and ai. Robotics and Autonomous Systems 154, 104096.
DOI: 10.1016/J.ROBOT.2022.104096

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

Izzo, R. A., Bardaro, G., Matteucci, M., 3 2024. Btgenbot: Behavior tree generation for robotic tasks with lightweight llms. arXiv.

Kalyan, K. S., 3 2024. A survey of gpt-3 family large language models including chatgpt and gpt-4. Natural Language Processing Journal 6, 100048.
DOI: 10.1016/j.nlp.2023.100048

Kim, C. Y., Lee, C. P., Mutlu, B., 3 2024. Understanding large-language model (llm)-powered human-robot interaction. ACM/IEEE International Conference on Human-Robot Interaction, 371–380.
DOI: 10.1145/3610977.3634966

Lykov, A., Tsetserukou, D., Lykov, A., Tsetserukou, D., 5 2023. Llm-brain: Ai-driven fast generation of robot behaviour tree based on large language model. arXiv, arXiv:2305.19352.
DOI: 10.48550/ARXIV.2305.19352

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., Corbato, C. H., Bonhof, S., Wisse, M., 4 2023. Active inference and behavior trees for reactive action planning and execution in robotics. IEEE Transactions on Robotics 39, 1050–1069.
DOI: 10.1109/TRO.2022.3226144

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/

Zhou, H., Lin, Y., Yan, L., Zhu, J., Min, H., 4 2024. Llm-bt: Performing robotic adaptive tasks based on large language models and behavior trees. Proceedings - IEEE International Conference on Robotics and Automation, 16655–16661.
DOI: 10.1109/ICRA57147.2024.10610183

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