Procesamiento del Lenguaje Natural, Revista nº 75, septiembre de 2025, pp. 237-250 recibido 06-04-2025 revisado 05-05-2025 aceptado 19-05-2025

Assessing Small Language Models for Translating Spanish Instructions into Behavior Trees

Evaluación de Modelos de Lenguaje Pequeños para la Conversión de Instrucciones en Español a Árboles de Comportamiento

Aitzol Saizar, Ander Corral,
Maddalen Lopez de Lacalle, Gorka Urbizu, Xabier Saralegi
Orai NLP Technologies
{a.corral,m.lopezdelacalle}@orai.eus

Abstract: Behavior Trees (BTs) are a widely used framework for decision-making in robotics due to their modularity, hierarchical organization, and reactivity. Large Language Models (LLMs), known for their strong capabilities in natural language understanding and structured generation, offer new opportunities for automating BT construction from natural language instructions. While prior work has explored LLM-based BT generation, most research has focused exclusively on English, limiting accessibility for non-English-speaking communities. This paper investigates the ability of LLMs—particularly sub-10B Small Language Models (SMLs)—to generate BTs from natural language, with a specific focus on Spanish. We evaluate the impact of learning paradigms (zero-shot, few-shot, and fine-tuning), model size, and language variation using NL2BT-bi, a novel bilingual dataset spanning 23 robotics domains. Our results demonstrate that fine-tuned SMLs can achieve performance on par with the best few-shot configurations of much larger 70B LLMs, suggesting that SMLs can be both effective and efficient for multilingual BT generation.
Keywords: NL2CODE, Behavior Trees, SLM, Robotics.

Resumen: Los Árboles de Comportamiento (ACs) son un marco ampliamente utilizado en robótica, gracias a su modularidad, estructura jerárquica y capacidad de respuesta. Los Modelos de Lenguaje de Gran Tamaño (LLMs), conocidos por su habilidad en la comprensión del lenguaje natural y la generación estructurada, abren nuevas oportunidades para automatizar la construcción de ACs a partir de instrucciones en lenguaje natural. Aunque investigaciones previas han explorado la generación de ACs mediante LLMs, la mayoría se ha centrado exclusivamente en el inglés, limitando su accesibilidad para comunidades que hablan otros idiomas. Este trabajo analiza la capacidad de los LLMs—en particular los Modelos de Lenguaje Pequeños (SMLs) de menos de 10B parámetros—para generar ACs a partir de instrucciones en español. Evaluamos el impacto de distintos paradigmas de aprendizaje (zero-shot, few-shot y ajuste fino), del tamaño del modelo y de la variación lingüística utilizando NL2BT-bi, un nuevo dataset bilingüe que abarca 23 dominios del ámbito de la robótica. Nuestros resultados demuestran que los SMLs de ajuste fino pueden alcanzar un rendimiento comparable al de las mejores configuraciones few-shot de LLMs de 70B, lo que sugiere que los SMLs pueden ser eficaces y eficientes para la generación multilingüe de BTs.
Palabras clave: NL2CODE, Árboles de comportamiento, SLM, Robótica.

1 Introduction

Behavior Trees (BTs) have become a widely adopted decision-making framework in robotics due to their modularity, hierarchical structure, and reactive properties (Colledanchise and Ögren, 2018; Iovino et al., 2022). Compared to traditional finite-state machines, BTs offer a more structured, scalable, and reusable approach for defining complex robotic behaviors, making them par-

ISSN 1135-5948 DOI 10.26342/2025-75-18

ticularly valuable for autonomous systems. However, manually designing effective BTs requires substantial domain expertise, which can be both tedious and time-consuming, especially for complex tasks.

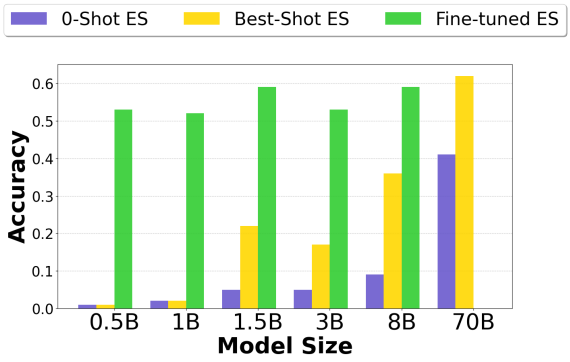

Figure 1: Accuracy on the NL2BT-bi Spanish test set for different model sizes across evaluation setups. Each group of bars represents a model size, with the first bar showing zero-shot results, the second showing the best few-shot results, and the third showing fine-tuned model results.

Large Language Models (LLMs) exhibit remarkable capabilities in natural language understanding, reasoning, and structured output generation, making them promising candidates for natural language to code (NL2Code) use cases (Chen et al., 2021; Nijkamp et al., 2023; Yu et al., 2024). Recent research on automated BT generation has also explored the use of LLMs (Ao et al., 2024; Izzo, Bardaro, and Matteucci, 2024; Zhou et al., 2024). By leveraging in-context learning and fine-tuning techniques, LLMs can generate BT representations from task descriptions, reducing reliance on expert knowledge and streamlining the development process.

Most research on automatic BT generation has focused on English, limiting accessibility for the Spanish-speaking robotics community. Given that Spanish is one of the most widely spoken languages, addressing this gap is crucial for broader adoption and linguistic inclusion in robotics.

This study evaluates the capability of LLMs to generate BT representations from natural language instructions, with a particular focus on Spanish and Small Language Models (SLM). We investigate how factors such as model size, language, in-context learning strategies, and fine-tuning influence BT generation performance.

Through this research, we aim to establish a foundational understanding of BT generation from Spanish instructions, contributing to the broader adoption of LLM-based BT methodologies in robotics across languages beyond English, with particular attention to sub-10B SMLs due to their advantages, such as on-premise deployment and lower computational requirements. Another key contribution of this work is NL2BT-bi[1], a novel dataset for the task of translating natural language instructions into BTs. It features examples in both English and Spanish, spanning 23 distinct domains.

The research questions addressed in this paper are as follows:

- **RQ1**: Can Language Models (LM) generate BTs from natural language using zero-shot prompting, and does in-context learning (few-shot prompting) improve the quality of results?
- **RQ2**: Does fine-tuning an SLM for BT generation provide improvements over in-context learning?
- **RQ3**: What is the impact of model size on BT generation quality, computational efficiency, and resource consumption? Are SMLs competitive?
- **RQ4**: Is there a notable difference in the task results between Spanish and English?

The results shown in Figure 1 offer a comparative overview of accuracy on the NL2BT-bi Spanish test set across various model sizes and learning paradigms (zero-shot, few-shot, and fine-tuning). They highlight how performance is influenced not only by model scale but also by the adaptation strategy. Notably, fine-tuned SMLs achieve accuracy comparable to the best few-shot configurations of 70B LLMs.

The structure of this paper is as follows: Section 2 reviews related work. Section 3 describes the experimental setup, including models, datasets, and in-context-learning and fine-tuning strategies. Section 4 presents our results and analysis. Finally, Section 5 summarizes key findings and provides insights for future research.

[1] huggingface.co/datasets/orai-nlp/NL2BT-bi

2 Related Work

2.1 Natural Language-To-Code

Beyond traditional NLP tasks, LLMs have been applied to code generation, a field often referred to as NL2Code. Code generation models such as OpenAI Codex (Chen et al., 2021) and StarCoder (li2, 2023) have demonstrated the ability to produce executable code from textual descriptions, significantly reducing human effort in programming. CodeT5 (Wang et al., 2021), an identifier-aware pretrained model, enhances NL2Code capabilities by improving semantic understanding and syntactic correctness in generated code. These advancements have streamlined software development workflows and opened new possibilities for automating structured code synthesis.

Several new benchmarks and models further expanded the NL2Code landscape. The benchmarks TACO (Li et al., 2023), EvalPlus (Liu et al., 2023) and CoderEval (Yu et al., 2024) emphasize pragmatic and executable code generation, while CodeAgent (Zhang et al., 2024) evaluates LLMs as autonomous agents capable of tool-augmented code generation. Newer instruction-tuned models for coding, such as WizardCoder (Luo et al., 2024) and DeepSeekCoder-V2 (Zhu et al., 2024), and the latest flagship reasoning models, such as DeepSeek-R1 (Guo et al., 2025), have shown state-of-the-art results on function-level and task-level code synthesis.

In robotics, NL2Code techniques play a crucial role in enabling robots to autonomously learn and execute behavior policies without manually defined rule sets (Sun et al., 2024). By generating interpretable and executable robotic control programs, LLMs bridge the gap between natural language instructions and machine-executable tasks, as demonstrated by Liang et al. (2023), which leverages LLMs to generate structured code for robotic control, reducing reliance on expert programmers.

Several works have explored LLM-driven code generation for robotic behavior. Driess et al. (2023) introduced PaLM-E, a multimodal model capable of interpreting language and sensor data to guide robotic actions. Similarly, Brohan et al. (2023) proposed Robotics Transformer 2, a model designed for large-scale robotic control. Boston Dynamics has also investigated LLM-driven behavior generation for its quadrupedal robot, Spot, demonstrating real-world applications of LLMs in autonomous robotic tasks (Petkauskas, 2023).

2.2 Behaviour Tree Generation

BTs are widely used for robotic decision-making due to their modular and hierarchical structure. However, manually designing BTs requires substantial domain expertise, making automatic BT generation a promising area of research.

Prior studies have explored the integration of LLMs for BT generation, yet most existing approaches focus on structured templates rather than open-ended tree construction. For instance, Cao and Lee (2023) proposed BT-GPT, which employs GPT-based models to generate robotic behaviors but is constrained by predefined templates, limiting adaptability to diverse and complex task descriptions. Similarly, Izzo, Bardaro, and Matteucci (2024) introduced BTGenBot, a lightweight LLM fine-tuned on robotic datasets for task-specific BT generation, optimizing computational efficiency but lacking generalization beyond the training domain.

Recent works have attempted to move beyond templated approaches by exploring methods that enable LLMs to construct novel BT structures autonomously. Wang et al. (2023) investigated self-instruct learning for open-ended BT generation, allowing LLMs to create BTs without relying on rigid templates. Additionally, Lykov and Tsetserukou (2024) developed the LLM-BRAIn dataset, a structured reasoning benchmark designed to evaluate LLM performance on BT generation and related tasks.

Our work extends these efforts by systematically evaluating LLM-based BT generation in Spanish, an area that has been largely overlooked in previous studies. Unlike BT-GPT (Cao and Lee, 2023) and BTGenBot (Izzo, Bardaro, and Matteucci, 2024), which rely on explicit fine-tuning or pre-defined structures, we analyze zero-shot, few-shot, and fine-tuned approaches to determine the extent to which LLMs -particularly SMLs- can generalize BT generation in Spanish and English context.

3 Experimental Setup

3.1 Models

For our experiments, we selected two families of language models: LLaMA3 and Qwen2.5. These models were originally pretrained on multilingual corpora encompassing English, Spanish, and code, although the extent of exposure to each language and modality varies between families. This multilingual and multimodal pretraining enables the models—each to differing degrees—to comprehend and generate both natural language (in English and Spanish) and code.

As our focus were SLMs, we selected several sub-10B models from both families for our experiments, complemented by a 70B LLM for comparative analysis. Specifically, the models used in this work are:

- **LLaMA models**: LLaMA3.2-1B, LLaMA3.2-3B, LLaMA3.1-8B and LLaMA3.1-70B (Grattafiori et al., 2024).
- **Qwen models**: Qwen2.5-0.5B and Qwen2.5-1.5B (Yang et al., 2025).

3.2 Datasets

Due to the lack of high-quality, publicly available datasets for the Natural Language to Behavior Tree (NL2BT) task, we developed a synthetic bilingual dataset specifically designed to address this gap. Existing resources such as BTGenBot (Izzo, Bardaro, and Matteucci, 2024), with only 600 examples, are insufficient for fine-tuning or robust evaluation, while LLM-BRAIn (Lykov and Tsetserukou, 2024) is automatically generated by an outdated language model[2] and contains numerous observable errors. Moreover, both datasets are limited in domain coverage and lack the linguistic and situational diversity required for realistic and generalizable NL2BT applications. To overcome these limitations, our dataset spans multiple domains—including home automation, warehouse logistics, and autonomous transport—and is crafted to ensure both linguistic quality and task relevance in English and Spanish. To construct the dataset, we leveraged the strong instruction-following capabilities of GPT-4o, whose suitability for the task was manually validated prior to large-scale generation. This validation involved analyzing a random sample of 100 training examples, of which 94 were deemed correct, confirming the model's reliability in producing diverse and structurally rich examples.

The data generation process was as follows:

Domain Definition. We first defined a list of 23 domains relevant to robotic applications (see Table 8 in Appendix B for the full list of domains and corresponding example instructions). These cover a broad range of scenarios such as Home Automation and Personal Assistants, Warehouse Automation, Robotic Lab Assistance, Construction and Infrastructure, among others.

Use Case Specification. For each domain, we designed three representative use cases to ensure scenario diversity. For instance, within the Warehouse Automation domain, the selected use cases include:

- Inventory Management Robot: Scans barcodes, restocks shelves, and organizes inventory based on demand or layout changes.
- Package Sorting Robot: Identifies packages, sorts them by destination, and ensures efficient routing for delivery.
- Automated Forklift: A robot that navigates through the warehouse to move pallets, avoiding obstacles and following a structured path.

Instruction Generation. For each use case, we generated 100 diverse natural language instructions using GPT-4o, following strict guidelines to ensure quality, diversity, and task relevance. The instructions follow a professional, engineering-driven tone. They are action-oriented, with direct, explicit commands. Each instruction includes decision-making logic and task sequences.

Behavior Tree Generation. For each instruction, we prompted GPT-4o to generate the corresponding BT in XML format, capturing the logic and flow needed for a robot to autonomously execute the task. The prompting template enforced valid XML output, defined node types (e.g., `Sequence`, `Fallback`, `Action`, `Condition`), and included design constraints to ensure clarity, modularity, and correct control flow representation.

Spanish Translation. Finally, we translated the English instructions into Spanish

[2] `text-davinci-003`

Instruction (EN):

```
If an endangered species is detected within 20 meters, initiate
identification process; check current health status by analyzing
thermal and movement patterns; update species tracking logs.
```

Instruction (ES):

```
Si se detecta una especie en peligro de extinción dentro de 20 metros,
inicie el proceso de identificación; verifique el estado de salud actual
analizando los patrones térmicos y de movimiento; actualice
los registros de seguimiento de especies.
```

Behaviour Tree (XML):

```
<BehaviourTree>
    <Selector ID="EndangeredSpeciesDetection">
        <Sequence ID="CheckProximityAndAct">
            <Condition ID="IsEndangeredSpeciesDetected" range="20"/>
            <Sequence ID="IdentificationAndHealthCheck">
                <Action ID="IdentifySpecies"/>
                <Parallel ID="HealthStatusCheck" success_threshold="1">
                    <Action ID="AnalyzeThermalPatterns"/>
                    <Action ID="AnalyzeMovementPatterns"/>
                </Parallel>
                <Action ID="UpdateTrackingLogs"/>
            </Sequence>
        </Sequence>
    </Selector>
</BehaviourTree>
```

Figure 2: Sample from NL2BT-bi dataset showing bilingual instructions (English and Spanish) and the corresponding BT (XML). Other metadata such as domain, use case description, available nodes, and node lists are omitted in the image.

using GPT-4o, ensuring linguistic consistency while preserving the technical precision and structure of the original instructions[3].

Each sample in the resulting dataset includes a Spanish natural language instruction (alongside its corresponding English instruction) and a corresponding BT composed of control flow and leaf nodes. All BT node names are kept in English, following common programming conventions.

The final dataset consists of 6,135 samples covering 23 diverse domains and use cases. For validation and testing, we selected disjoint subsets of domains and their corresponding use cases (see Table 8 in Appendix B for details on domain splits). The validation set contains 900 samples drawn from three domains, each featuring three distinct use cases. Similarly, the test set includes three different domains and their associated use cases, with 15 randomly selected samples per use case, totaling 135 test instances. Among these, 5 were randomly chosen to serve as few-shot examples. The remaining 5,100 samples constitute the training set. Figure 2 presents an example from the dataset, showcasing a bilingual instruction (EN/ES) alongside its corresponding Behavior Tree (BT) in XML format.

[3]According to (Intento and e2f, 2024), GPT-4 achieved top COMET scores, comparable to those of DeepL and Google Translate. Its translation fluency was rated among the highest—alongside DeepL—and its contextual understanding outperformed most traditional machine translation systems.

3.3 In-Context Learning Strategies for NL2Code

We evaluated the effectiveness of LMs in generating BTs from instructions formulated in natural language employing different in-context learning strategies. We examined both zero-shot and few-shot learning approaches to assess their ability to produce structured and executable BT representations. These strategies are particularly relevant in real-world applications where predefined datasets for fine-tuning are not available, and models have to efficiently generalize to unseen tasks. In robotics, for instance, users often provide high-level natural language instructions without the possibility of extensive training, making zero-shot and few-shot learning highly practical.

For the **zero-shot approach**, we designed a prompt (see Table 7) in Appendix A that included a concise instruction directing the LM to generate an XML-formatted BT, strictly adhering to the available nodes. The input consisted of a natural language instruction, and the model was expected to produce a valid BT.

For the **few-shot prompting approach**, we provided the LM with multiple examples for the task (ranging from 1 to 5) directly inside the prompt. While Izzo, Bardaro, and Matteucci (2024) employed a one-shot approach, our study systematically explored the impact of increasing the number of demonstrations. To the best of our knowledge, there was no prior systematic study on applying few-shot in-context learning for BT generation, particularly in Spanish.

The effectiveness of both zero-shot and few-shot prompting strategies was evaluated through a series of experiments, detailed in Sections 4.1 and 4.2, respectively.

3.4 Supervised Fine-Tuning (SFT)

Supervised fine-tuning (SFT) has proven effective for structured task generalization in prior work (Lykov and Tsetserukou, 2024; Izzo, Bardaro, and Matteucci, 2024). Thus, we fine-tuned each model under 10B parameters on the NL2BT-bi dataset for up to 3 epochs. The training was conducted on the training split of the dataset, while the development split was used to select the best-performing checkpoint based on validation loss. We used a learning rate of 1.0×10^{-4} and a batch size of 128. Due to the high compu-

tational costs, we did not fine-tune Llama3-70B. In addition, maintaining a 70B fine-tuned model on-premise for a single task is impractical even in industrial settings.

The SFT experiments were carried out on NVIDIA A100 80GB GPUs (1-4 GPUs). For efficient training we leveraged DeepSpeed ZeRO (Rajbhandari et al., 2020) and Accelerate (Gugger et al., 2022) from the Hugging Face Transformers library (Wolf et al., 2020).

4 Experiments

In this section, we present the results of our experiments, structured as follows. First, we analyze zero-shot performance (Section 4.1), where models generate BT without prior examples. Then, we evaluate the impact of few-shot in-context learning strategy (Section 4.2), investigating how performance evolves as the number of provided examples increases. We then compare fine-tuned models with in-context learning approaches (Section 4.3), assessing the relative effectiveness of each method and the influence of model size. Finally, we analyze the computational efficiency and resource consumption of the different language models (Section 4.4).

All experiments are evaluated using an LLM-as-a-judge framework (Zheng et al., 2023), with GPT-4o (Hurst et al., 2024) used to assess output quality. A structured prompt (detailed in Appendix C) guides the LLM to verify whether the generated BT correctly reflects the input instruction and adheres to the allowed node types. The LLM provides a binary judgment (Valid or Invalid) along with a concise technical explanation. Accuracy is reported as the percentage of BTs classified as Valid. This evaluation strategy enables semantic and structural validation that traditional code metrics like CodeBLEU (Ren and others, 2020) cannot capture, while also avoiding the need for costly and time-consuming manual evaluation. To ensure reliability, we measured the correlation between GPT-4o's judgments and human annotations on a random subset of 100 examples, achieving 80% agreement.

4.1 Zero-Shot Results

Table 1 presents the accuracy results for the zero-shot setting (Section 3.3) across the Qwen and LLaMA models described in Section 3.1, evaluating their performance on the task using both English and Spanish commands.

Name	Size	ES	EN
Qwen2.5	0.5B	0.8	0.0
	1.5B	5.4	5.4
LLaMA3	1B	1.5	0.8
	3B	4.6	2.7
	8B	9.2	9.2
	70B	**41.7**	**38.5**

Table 1: Zero-shot accuracy on the NL2BT-bi test set across Qwen and LLaMA models for Spanish (ES) and English (EN).

Overall, most models show limited generalization capabilities in zero-shot BT generation with accuracy scores far from the optimum. However, larger models show a clear advantage in this setup. **LLaMA3-70B** achieves the highest zero-shot accuracy in both English (38.5) and Spanish (41.7), substantially outperforming smaller models.

In contrast, SLMs like **Qwen2.5-0.5B** and **LLaMA3-1B** achieve very low accuracy, suggesting that without demonstrations, they struggle to infer structured outputs effectively.

4.2 Few-Shot Results

Tables 2 and 3 summarize the results of the in-context learning approach for English and Spanish, respectively. We report the accuracy scores for varying numbers of examples in the prompt, ranging from 1-shot to 5-shot.

Name	Size	1	2	3	4	5
Qwen2.5	0.5B	1.5	8.5	3.9	**9.2**	8.5
	1.5B	5.4	16.9	17.7	20.0	**21.5**
LLaMA3	1B	2.3	6.9	**12.3**	10.0	6.2
	3B	6.2	7.7	10.0	**16.9**	15.4
	8B	19.2	23.1	30.8	**36.2**	34.6
	70B	40.8	45.4	50.8	56.2	**61.5**

Table 2: Accuracy on the NL2BT-bi (Spanish) test set in the few-shot setting, comparing Qwen and LLaMA models with 1 to 5 in-context examples.

Overall, we observe a clear upward trend in performance as the number of in-context examples increases, particularly for larger models. For instance, **LLaMA3-8B** improves from 19.2 (1-shot) to 34.6 (5-shot) in Spanish, and from 12.3 to 48 in English. This

Name	Size	1	2	3	4	5
Qwen2.5	0.5B	0.8	5.4	**8.5**	4.6	4.6
	1.5B	4.6	13.9	17.7	18.5	**19.2**
LLaMA3	1B	4.6	3.1	**6.2**	4.6	**6.2**
	3B	1.5	3.9	10.0	12.3	**15.4**
	8B	12.3	20.8	25.4	32.3	**48.0**
	70B	40.8	44.6	50.8	**63.1**	53.9

Table 3: Accuracy on the NL2BT-bi (English) test set in the few-shot setting, comparing Qwen and LLaMA models with 1 to 5 in-context examples.

indicates that model scale and demonstration count both play critical roles in enabling the model to better understand and generate correct BTs.

LLaMA3-70B consistently outperforms smaller counterparts across all settings and languages, with peak performance achieved at 5-shot in Spanish and 4-shot in English.

Smaller models such as **Qwen2.5-0.5B** and **LLaMA3-1B** show only marginal improvements with additional examples, rarely exceeding 10-12 accuracy. This suggests that low-capacity models struggle to benefit from In-context learning alone in structured tasks like BT generation.

These results highlight the importance of both scale and demonstration design in few-shot prompting. While few-shot in-context learning improves performance, notable differences remain—especially for smaller models—motivating the use of fine-tuning for further gains.

4.3 Fine-Tuning Results

In this section, we present the results obtained from the fine-tuned models, trained as described in Section 3.4. Fine-tuning allows models to internalize structural patterns beyond short-context prompting, potentially yielding higher accuracy scores.

Fine-tuning leads to substantial improvements in accuracy for all models compared to their few-shot counterparts. For instance, Qwen2.5-0.5B, which reached at most 9.2 accuracy in the few-shot setting, achieves over 50 accuracy post fine-tuning in both languages. Similarly, LLaMA3-1B improves from 6.2 (few-shot) to 52.3 (fine-tuned), indicating that even small models can learn structured output formats when provided with supervised training data.

Name	Size	ES	EN
Qwen2.5	0.5B	53.1	51.5
	1.5B	59.2	53.9
LLaMA3	1B	52.3	52.3
	3B	53.1	53.9
	8B	**59.2**	**63.1**

Table 4: Accuracy on the NL2BT-bi test set (Spanish and English) using fine-tuned Qwen and LLaMA models under 10B parameters.

LLaMA3-8B achieves the highest fine-tuned performance with 63.1 accuracy in English and 59.2 in Spanish, matching the result achieved by Qwen2.5-1.5B in Spanish. Notably, the performance gap between models of different sizes is less pronounced after fine-tuning than in the few-shot setting, suggesting diminishing returns with scale once sufficient supervision is available.

These findings confirm that fine-tuning is a more effective and efficient strategy than few-shot prompting for sub-10B models in BT generation tasks. In Spanish, small models (e.g., LLaMA3-8B and Qwen2.5-1.5B) achieve an accuracy of 59.2, while the best few-shot result from LLaMA3-70B reaches only slightly higher at 61.5. In English, fine-tuned LLaMA3-8B even matches the few-shot performance of LLaMA3-70B. This relatively small gap highlights that fine-tuning enables smaller models to approach, and in some cases match, the performance of much larger ones—making it a practical and resource-efficient alternative when working under compute or deployment constraints.

4.4 Computational Efficiency and Consumption

Beyond evaluating the accuracy of BT generation, we analyze the computational efficiency of the different SLMs and environmental impact (Samsi et al., 2023; Rillig et al., 2023). These factors are crucial for selecting models suitable for real-time robotics applications and large-scale deployment. We compare two strategies:

- **4-shot prompting**: The model receives four example demonstrations in the prompt before generating a response. This configuration yields optimal performance across most models (see Section 4.2).

Model	Size	Accuracy	TTFT	Runtime	GPU	Energy	Emissions
Qwen2.5	0.5B	9.2	5.1 (34%)	8.79 (48%)	1575 (90%)	0.11 (71%)	19.7 (71%)
	1.5B	20.0	5.2 (32%)	8.65 (49%)	3807 (76%)	0.13 (68%)	21.7 (68%)
LLaMA3	1B	10.0	5.2 (32%)	3.72 (78%)	3019 (81%)	0.05 (86%)	9.5 (86%)
	3B	16.9	7.8 (-1%)	12.62 (26%)	6687 (58%)	0.22 (43%)	40.0 (43%)
	8B	36.2	7.7	16.99	15919	0.39	67.5

Table 5: Accuracy and computational efficiency for the 4-shot setting. TTFT is measured in milliseconds, Runtime in seconds, GPU memory in MB, Energy in kWh, and Emissions in g CO_2. Percentages indicate improvement over the LLaMA3 8B.

Model	Size	Accuracy	TTFT	Runtime	GPU	Energy	Emissions
Qwen2.5	0.5B	53.1	4.5 (17%)	4.52 (72%)	1340 (92%)	0.09 (76%)	15.0 (76%)
	1.5B	59.2	4.7 (13%)	12.24 (23%)	3500 (78%)	0.26 (29%)	44.7 (29%)
LLaMA3	1B	52.3	5.8 (-7%)	8.41 (47%)	2757 (83%)	0.17 (53%)	30.0 (52%)
	3B	53.9	5.2 (4%)	9.72 (39%)	6696 (58%)	0.21 (43%)	36.0 (43%)
	8B	63.1	5.4	15.89	15934	0.36	63.0

Table 6: Accuracy and computational efficiency for the fine-tuned models (0-shot setting). TTFT is measured in milliseconds, Runtime in seconds, GPU memory in MB, Energy in kWh, and Emissions in g CO_2. Percentages indicate improvement over the LLaMA3 8B.

- **Fine-tuning**: The model is fine-tuned on the target task, enabling it to generate responses without in-context examples (see Section 4.3).

Although smaller models are capable of running efficiently on widely available, lower-powered GPUs, all experiments in this analysis were performed on a standardized high-performance system (NVIDIA A100 SXM4 GPU with 80GB memory). This consistent hardware setup eliminates variability caused by hardware differences, ensuring that performance comparisons reflect only differences in model architecture and evaluation strategy, and providing a fair and controlled basis for comparing models of varying sizes.

To evaluate inference **efficiency**, we use the llm-perf-benchmark library (Ruman and contributors, 2024), which provides standardized tools for measuring SLM performance. Specifically, we report:

- **Time to First Token (TTFT)**: The time (in milliseconds) from issuing a query to receiving the first token.
- **Model Inference Runtime**: The time (in seconds) taken to generate the full output.

Metrics were measured as the average over 100 input samples from the NL2BT-bi dataset (Spanish test set). For each run, we recorded TTFT and total generation time.

Additionally, we evaluated resource **consumption** in terms of GPU memory usage, energy consumption (kWh), and carbon emissions (g CO_2), using CodeCarbon (Courty et al., 2024)—a widely adopted tool for estimating power usage in machine learning workloads. These metrics are essential for assessing the environmental impact associated with deploying large-scale models.

Table 5 presents results for the 4-shot configuration, while Table 6 shows results for fine-tuned models evaluated in a zero-shot setting. In both tables, we include accuracy, TTFT, TGT, GPU memory consumption, energy usage, and emissions. Percentage improvements relative to LLaMA3-8B are shown in parentheses.

To identify the best model configuration, we compare 4-shot prompting and fine-tuned (0-shot) models. Fine-tuned models consistently outperform 4-shot models in accuracy and offer similar runtimes, making them the preferred choice.

Among fine-tuned models, Qwen2.5-0.5B stands out for its efficiency—lowest runtime (4.52s), energy (0.09 kWh), and emissions (15.0 g CO_2)—with a solid 53.1 accuracy.

Qwen2.5-1.5B achieves the highest accuracy 59.2 while remaining efficient, reducing

energy use and emissions by 29% compared to LLaMA3-8B. LLaMA3-3B also performs well (53.9 accuracy), with 43% energy savings compared to LLaMA3-8B, but it has a longer runtime and higher memory usage. In contrast, LLaMA3-1B achieves nearly the same accuracy (52.3) with 38% energy savings, and Qwen2.5-0.5B matches the accuracy (53.1) while offering the highest efficiency—saving 76% energy.

5 Conclusions

Our results show that zero-shot prompting is insufficient for accurate BT generation from Spanish instructions, particularly for small models. Few-shot prompting improves performance, especially for larger models, but its impact is limited for models under 1B parameters.

Supervised fine-tuning consistently outperforms prompting strategies across all sub-10B sizes, enabling even small models like Qwen2.5-0.5B and LLaMA3-1B to exceed 50% accuracy. This confirms fine-tuning as the most effective method for Spanish to BT generation task.

While model size plays a key role in zero- and few-shot settings, fine-tuned small models narrow the performance gap with larger ones. They also offer clear advantages for deployment in computational efficiency and sustainability.

Finally, we found that SLMs are capable of generating high-quality BTs from natural text in both English and Spanish. The overall performance trend was consistent across both languages. This supports the viability of BT generation from Spanish instructions and highlights the importance of expanding research and dataset coverage beyond English.

Acknowledgments

This work has been partially funded by the Basque Government (IKASPROD project, grant no. KK-2024/00050). Model fine-tunings were conducted using the Hyperion system at the Donostia International Physics Center (DIPC).

References

2023. StarCoder: May the source be with you! *Transactions on machine learning research*.

Ao, J., F. Wu, Y. Wu, A. Swikir, and S. Haddadin. 2024. LLM-as-BT-Planner: Leveraging LLMs for behavior tree generation in robot task planning. *arXiv preprint arXiv:2409.10444*.

Brohan, A., N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, et al. 2023. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*.

Cao, X. and K. Y. Lee. 2023. BT-GPT: Behavior tree generation using large language models. *arXiv preprint arXiv:2305.12345*.

Chen, M., J. Tworek, H. Jun, Q. Yuan, H. P. D. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Colledanchise, M. and P. Ögren. 2018. *Behavior trees in robotics and AI: An introduction*. CRC Press.

Courty, B., V. Schmidt, S. Luccioni, Goyal-Kamal, MarionCoutarel, and B. Feld. 2024. MLCO2/Codecarbon: v2.4.1. may.

Driess, D., F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, et al. 2023. PaLM-E: an embodied multimodal language model. In *Proceedings of the 40th International Conference on Machine Learning*, pages 8469–8488.

Grattafiori, A., A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan, et al. 2024. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Gugger, S., L. Debut, T. Wolf, P. Schmid, Z. Mueller, S. Mangrulkar, M. Sun, and B. Bossan. 2022. Accelerate: Training and inference at scale made simple, efficient and adaptable. `https://github.com/huggingface/accelerate`.

Guo, D., D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, et al. 2025. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Hurst, A., A. Lerer, A. P. Goucher, A. Perelman, A. Ramesh, A. Clark, A. Ostrow,

A. Welihinda, A. Hayes, A. Radford, et al. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Intento and e2f. 2024. The state of machine translation 2024. `https://inten.to/machine-translation-report-2024/`. Accessed: 2025-05-29.

Iovino, M., E. Scukins, J. Styrud, P. Ögren, and C. Smith. 2022. A survey of behavior trees in robotics and AI. *Robotics and Autonomous Systems*, 154:104096.

Izzo, R. A., G. Bardaro, and M. Matteucci. 2024. BTGenBot: Behavior tree generation for robotic tasks with lightweight LLMs. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9684–9690. IEEE.

Li, R., J. Fu, B.-W. Zhang, T. Huang, Z. Sun, C. Lyu, G. Liu, Z. Jin, and G. Li. 2023. TACO: Topics in Algorithmic COde generation dataset. *arXiv preprint arXiv:2312.14852*.

Liang, J., W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng. 2023. Code as policies: Language model programs for embodied control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9493–9500. IEEE.

Liu, J., C. S. Xia, Y. Wang, and L. Zhang. 2023. Is your code generated by ChatGPT really correct? rigorous evaluation of large language models for code generation. *Advances in Neural Information Processing Systems*, 36:21558–21572.

Luo, Z., C. Xu, P. Zhao, Q. Sun, X. Geng, W. Hu, C. Tao, J. Ma, Q. Lin, and D. Jiang. 2024. WizardCoder: Empowering code large language models with evol-instruct. In *The Twelfth International Conference on Learning Representations*.

Lykov, A. and D. Tsetserukou. 2024. LLM-BRAIn: AI-driven fast generation of robot behaviour tree based on large language model. In *2024 2nd International Conference on Foundation and Large Language Models (FLLM)*, pages 392–397. IEEE.

Nijkamp, E., B. Pang, H. Hayashi, L. Tu, H. Wang, Y. Zhou, S. Savarese, and C. Xiong. 2023. CodeGen: An open large language model for code with multi-turn program synthesis. In *The Eleventh International Conference on Learning Representations*.

Petkauskas, V. 2023. Boston Dynamics' Spot robot dog gets a ChatGPT brain. *TechRadar*.

Rajbhandari, S., J. Rasley, O. Ruwase, and Y. He. 2020. ZeRO: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16. IEEE.

Ren, S. et al. 2020. CodeBLEU: A method for evaluating code generation. *arXiv preprint arXiv:2009.10297*.

Rillig, M. C., M. Ågerstrand, M. Bi, K. A. Gould, and U. Sauerland. 2023. Risks and benefits of large language models for the environment. *Environmental science & technology*, 57(9):3464–3466.

Ruman and contributors. 2024. LLM-perf-benchmark: A library for measuring LLM inference performance. `https://github.com/rumanxyz/llm-perf-benchmark`. Accessed: 2025-03-17.

Samsi, S., D. Zhao, J. McDonald, B. Li, A. Michaleas, M. Jones, W. Bergeron, J. Kepner, D. Tiwari, and V. Gadepally. 2023. From words to watts: Benchmarking the energy costs of large language model inference. In *2023 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–9. IEEE.

Sun, Q., Z. Chen, F. Xu, K. Cheng, C. Ma, Z. Yin, J. Wang, C. Han, R. Zhu, S. Yuan, et al. 2024. A survey of neural code intelligence: Paradigms, advances and beyond. *CoRR*.

Wang, Y., Y. Kordi, S. Mishra, A. Liu, N. A. Smith, D. Khashabi, and H. Hajishirzi. 2023. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13484–13508.

Wang, Y., W. Wang, S. Joty, and S. C. Hoi. 2021. CodeT5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation.

In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8696–8708.

Wolf, T., L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45.

Yang, A., B. Yu, C. Li, D. Liu, F. Huang, H. Huang, J. Jiang, J. Tu, J. Zhang, J. Zhou, et al. 2025. Qwen2. 5-1m technical report. *arXiv preprint arXiv:2501.15383*.

Yu, H., B. Shen, D. Ran, J. Zhang, Q. Zhang, Y. Ma, G. Liang, Y. Li, Q. Wang, and T. Xie. 2024. CoderEval: A benchmark of pragmatic code generation with generative pre-trained models. In *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering*, pages 1–12.

Zhang, K., J. Li, G. Li, X. Shi, and Z. Jin. 2024. CodeAgent: Enhancing code generation with tool-integrated agent systems for real-world repo-level coding challenges. In L.-W. Ku, A. Martins, and V. Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13643–13658, Bangkok, Thailand, August. Association for Computational Linguistics.

Zheng, L., W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. Xing, et al. 2023. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. *Advances in Neural Information Processing Systems*, 36:46595–46623.

Zhou, H., Y. Lin, L. Yan, J. Zhu, and H. Min. 2024. LLM-BT: Performing robotic adaptive tasks based on large language models and behavior trees. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 16655–16661. IEEE.

Zhu, Q., D. Guo, Z. Shao, D. Yang, P. Wang, R. Xu, Y. Wu, Y. Li, H. Gao, S. Ma, et al. 2024. DeepSeek-Coder-V2: Breaking the barrier of closed-source models in code intelligence. *arXiv preprint arXiv:2406.11931*.

A Appendix A: Example of Zero-Shot prompt

Table 7 illustrates the complete zero-shot prompt, including a task description in Spanish, as used in our multilingual experimental setup. The prompt includes a concise description of the task, the required output format, an explanation of the different types of BT nodes, and a list of available node templates for action and condition nodes. Additionally, it provides design guidelines to ensure clarity, modularity, and consistency in the generated XML structure. The goal is to assess the model's ability to understand and execute the task solely based on this instruction.

B Appendix B: Domains and Instruction Examples

Table 8 provides a detailed overview of the application domains included in the NL2BT-bi dataset. For each domain, we present a representative example of a natural language instruction that highlights the type of robotic behavior expected in that context.

C Appendix C: Evaluation Prompt for BT Validity

Table 9 presents the structured prompt used to assess the validity of each generated Behavior Tree. The prompt provides clear validation rules and an expected output format, ensuring consistency and objectivity across evaluations.

```
You are a skilled robotics engineer. Your task is to generate a Behavior Tree (BT)
in XML format that represents the logic and flow needed for a robot to perform the
given task. The BT should support autonomous execution, including conditions,
sequences, and decision-making.

Input: A single, clear natural language instruction describing a specific robotic
task or sequence.

Output Format: Provide only the XML Behavior Tree.

Node Types:
1. Control Flow Nodes:
    Sequence: Executes children in order, stopping on failure.
    Fallback (Selector): Executes children in order, stopping on success.
    Parallel: Executes all children simultaneously, returns
              success/failure based on criteria.
    Decorator: Modifies child nodes (e.g., Inverter, Repeater, Timeout).
2.Leaf Nodes:
    Action: Performs tasks (e.g., MoveTo(location), Pick(item)).
    Condition: Checks conditions (e.g., IsObjectInRoom).

Design Guidelines:
    Use meaningful names for actions and conditions by using the ID attribute
    Ensure the XML reflects the conditional and sequential logic of the instruction.
    Keep the tree modular and easy to understand.
    The root element is <BehaviourTree>
    Ports are configured using attributes. For example the action SaySomething
    requires the input port message="Hello".

Avoid:
    No comments or descriptions in the XML.

Generate the XML Behavior Tree for this instruction: Mientras transporta paquetes al
área de entrega, supervise continuamente los niveles de batería; si la batería cae
por debajo del 20%, pause las tareas, regrese a la estación de carga y reanude las
operaciones una vez que esté adecuadamente cargada. List of node availables:
['<Action ID="MoveTo" location="PackagePickupArea"/>', '<Action ID="Pick" item="Package"/>',
'<Action ID="MoveTo" location="DeliveryBay"/>', '<Action ID="Drop" item="Package"/>',
'<Action ID="MoveTo" location="PackagePickupArea"/>',
'<Condition ID="IsBatteryLow" level="20"/>', '<Action ID="PauseTasks"/>',
'<Action ID="MoveTo" location="ChargingStation"/>',
'<Condition ID="IsAdequatelyCharged" level="80"/>', '<Action ID="ResumeOperations"/>']

XML:
```

Table 7: Zero-Shot prompt (Instruction in Spanish) used for Behavior Tree generation.

Domain	Example of Instruction
Train split	
Autonomous Construction Robots	After completing each foundation segment, perform dimensional accuracy check using infrared sensors; if deviation is detected, make necessary adjustments and record changes.
Construction and Infrastructure	If the weather conditions are adverse, such as rain or extreme temperatures, pause the operations and enter standby mode until conditions improve.
Consumer Electronics and IoT	If the doorbell is pressed and the time is between 9 PM and 7 AM, chime the internal signal at reduced volume, activate the front camera, and verify the identity of the visitor before unlocking the door.
Education and Training	Identify the difficulty level of the upcoming task based on the student's progress; if a task matches the proficiency level, proceed with task initiation; if too challenging, adjust the task accordingly.
Entertainment and Interactive Robots	If rain is detected, notify all nearby visitors of the weather condition, suggest indoor attractions, and provide directions to the nearest sheltered area.
Environmental Monitoring and Protection	Initiate daily maintenance protocol after completing all tasks; perform comprehensive system diagnostics including sensor calibration and motor inspection; if any discrepancies are found, initiate correction sequence and log maintenance activity before shutting down.
Hospitality and Food Service	If a new customer is detected at a table, greet the customer, wait for a response, and then prompt them for their order.
Hospitality and Guest Experience	Detect and confirm the door's open status using the onboard camera sensor; if the door remains closed for more than 30 seconds, notify the monitoring system for guest assistance.
Media Production and Filmmaking	Attach the camera securely to the tripod; confirm the camera mount is locked; initiate a test shake to ensure stability; if instability is detected, readjust and perform the test again.
Military and Defense	In case of detection of human presence within 10 meters, switch to alert mode, halt all operations, and broadcast warning to surrounding personnel.
Retail and Customer Service	Initiate scanning mode by directing sensor towards checkout item; if item is detected, commence barcode scan; upon successful scan, register item to digital cart and display total on screen.
Retail and Supply Chain	Check for replenishment approval on the management console; once approved, gather necessary transportation tools; move to the supply area to collect stock.
Robotic Bartenders and Waitstaff	Navigate to the kitchen area after receiving a new order; if the order is ready, collect the food tray carefully, ensuring no items are missing.
Robotic Lab Assistance	After validating an entry, transcribe the data into the designated fields of the lab database application; confirm data alignment with existing records; if alignment is incorrect, append an error message to the daily report log.
Security and Surveillance	Every 30 minutes, initiate a patrol routine by navigating the perimeter in a clockwise direction, avoiding obstacles, and checking for anomalies.
Space Exploration	Every communication cycle, check signal strength and integrity; adjust antenna alignment to enhance transmission quality, and confirm successful data transfer before terminating the cycle.
Tourism and Adventure	Engage visual recognition system to identify potential artifacts; if artifact is detected, halt movement and initiate detailed imaging process for documentation.
Validation split	
Agriculture	If the robot's weed bin is full, navigate to the disposal unit, empty the bin, and resume tasks from the last weed detection point.
Autonomous Vehicles	If the robot encounters a steep incline, automatically switch to low gear mode, engage additional traction controls, and proceed at half speed until the grade levels out.
Search and Rescue	Upon detecting a human presence, utilize auditory sensors to verify distress calls; if a distress call is confirmed, send an immediate alert to the command center and proceed with task 5.
Test split	
Healthcare and Hospital Assistance	Navigate to the pharmacy storage area; scan for required medication; if medication is not located, send an alert to the pharmacy staff.
Home Automation and Personal Assistants	If in the bedroom and the bed is unmade, automatically make the bed; subsequently, proceed with surface dusting and completion confirmation before exiting.
Warehouse Automation	Calculate the optimal path to the destination using stored warehouse map data; while navigating, constantly check for path deviations and recalibrate as needed.

Table 8: Overview of the domains included in the NL2BT-bi dataset, with example natural language instructions corresponding to each domain.

```
You are a senior robotics engineer. Your task is to determine whether the given Behavior Tree
(BT) is valid with respect to the exact content of the instruction and the list of available
leaf nodes.

Instruction: {{instruction}}

List of available nodes: {{tree_nodes|join('\n')}}

Behavior Tree (XML): {{behaviour_tree}}

Validation Rules:
- The BT must faithfully represent only the logic explicitly described in the instruction.
- Do not evaluate or assume any behavior not mentioned in the instruction.
- All leaf nodes (i.e., Action and Condition nodes) used in the BT must come only from the
provided list of available node names.
- Ignore whether the BT could be improved. Focus only on fidelity to the instruction and
compliance with the available node list.

Output Format: Return a JSON object with binary validity and a short explanation:

‘‘‘json
{
  "result": "Valid" | "Invalid",
  "explanation": "<brief justification based only on the instruction and node list>"
}‘‘‘
```

Table 9: Prompt used to evaluate the correctness of a generated Behavior Tree (BT).