The Impact of Prompt Engineering on LLM-Synthesized Behavior Trees

Nurun Naher[0000−0001−7974−6742] and Gita Sukthankar[0000−0002−6863−6609]

University of Central Florida, Orlando, FL USA

Abstract. Behavior modeling is essential for building realistic training and simulation environments, covering everything from individual actions to complex team behaviors. This paper investigates using large language models (LLMs) to streamline the creation of behavior trees, a popular method for controlling AI character actions. We demonstrate that by using specific prompting techniques, we can ensure the LLM generates syntactically valid behavior trees, even when using higher temperature settings to encourage creativity. We also examine how different prompts influence the action diversity and depth of the generated behavior trees. Ultimately, our work aims to leverage LLMs to create a more efficient and scalable process for developing complex AI behaviors, thereby enhancing the realism of training simulations.

Keywords: behavior trees · training simulations · LLMs

1 Introduction

Recent advances in large language models (LLMs) have shown promise in streamlining the development of complex game AI behaviors, offering a potential breakthrough in generating behavior tree libraries that accurately model military tactics, techniques, and procedures. Behavior trees [4] are a valuable framework for modeling doctrinal behavior, and integrating LLMs with behavior trees has made generating complex AI behaviors more efficient and accessible. These models bridge the gap between expert-driven authoring and user-friendly AI generation, reducing authoring complexity and costs.

This paper describes a system that uses an LLM to generate behavior trees. To do this, it leverages information from training manuals with a technique called Retrieval Augmented Generation (RAG) [7]. The system also uses a fine-tuned adapter, created with LoRA (Low Rank Adaptation) [3], to specialize in generating these behavior trees.

Behavior Trees (BTs) have been widely adopted in games and simulations for controlling non-player characters (NPCs) and other intelligent agents; they have many advantages as a programming formalism including strong support for task hierarchies, sequences, fallbacks, and reactivity. Originally, programmers of game AI commonly used finite state automata (FSA) to script the behaviors of non-player characters. However, the transition logic of FSAs is distributed across states, making them unwieldy to understand and maintain. In contrast, behavior

trees possess a more modular structure in which the leaves are execution nodes, and non-leaf nodes govern the tree's control flow. Figure 1 shows an example behavior tree generated by our system. We compared a simple, generic prompt with

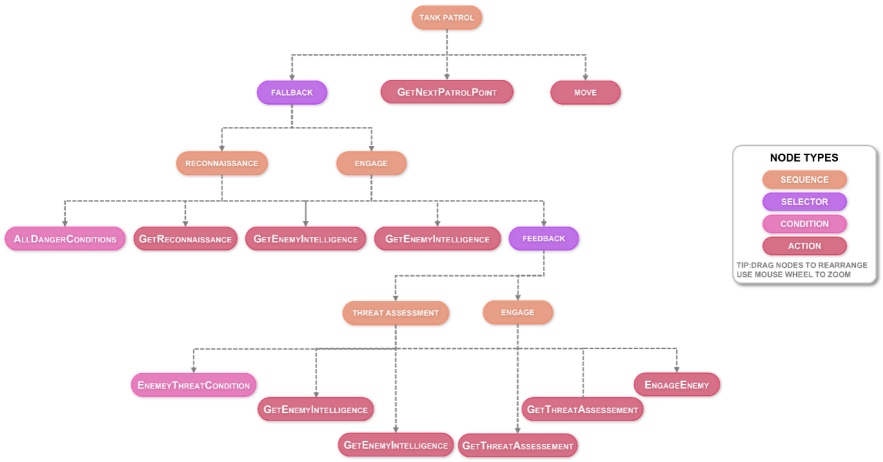

Fig. 1: An example behavior tree generated by our system with user prompt "Create a behavior tree for a tank patrol mission that includes reconnaissance, threat assessment, and engagement protocols."

a more specific one that included details about a behavior tree's structure. Our evaluation, which was conducted at various LLM temperature settings, showed that the specific prompt consistently created deeper, more complex behavior trees with fewer errors. While the generic prompt produced a wider variety of action types at lower temperatures, it also resulted in more syntactical errors. This next section provides an overview on related work on the application of LLMs to behavior tree generation and military planning.

2 Related Work

2.1 LLMs for Behavior Tree Generation

LLMs can greatly simplify the behavior tree authoring process, since unlike the other machine learning approaches, they don't require example demonstrations. Recent research [1,9,10,8] has demonstrated that LLMs can be effective at generating behavior trees for robotic applications. The general methodology for training a large language model (LLM) to generate behavior trees is as follows: 1) select an instruction-following or code generation LLM, 2) harvest a dataset of behavior trees from online code repositories, 3) use parameter-efficient fine tuning (e.g., LoRA) to learn a specific adapter, 4) experiment with different prompting techniques, 5) evaluate the generated behavior trees using metrics like

syntactic correctness. Li et al. (2024) [8] provide an excellent high-level overview of this process, along with proposing a method for generating synthetic behavior tree data for fine-tuning using Monte Carlo Tree Search.

Our research builds on the work of Izzo et al. (2024) [5] who published an open source implementation of their robotic behavior tree generation system, BTGenBot. They created several different fine-tuned versions of Llama using a dataset of 600 BTs paired with natural language descriptions. Both the LlamaChat and CodeLlama versions are successful at the simpler robot tasks, assuming that one shot prompting is used and the behavior trees are corrected post-generation using basic static analysis. This research demonstrates that fine-tuning LLMs specifically for BT tasks can significantly improve the effectiveness of the generated behaviors.

2.2 LLMs and Military Doctrine

While machine learning excels at creating optimized behaviors, less attention has been given to developing behaviors that align with military doctrine. LLMs are a promising direction for this task because of their ability to directly "read" warfighter manuals and generate instructions based on what they've read. Goecks and Waytowich (2024) [2] explored the usage of LLMs to expedite the development of Courses of Action (COAs) for military operations. By incorporating domain expertise through in-context learning, COA-GPT allows commanders to input mission-specific details and receive rapidly generated COAs. The system also enables real-time refinement of COAs based on human feedback, enhancing both adaptability and alignment with the commander's objectives. COA-GPT uses the GPT-4 model to generate multiple COA options based on a detailed prompt. These options were then refined through human feedback and tested for strategic efficacy in StarCraft II measuring performance through reward scores, casualties, and response times. In a military-relevant scenario, COA-GPT demonstrated its ability to generate COAs more effectively than expert humans and state-of-the-art reinforcement learning algorithms. The system's real-time adaptability to battlefield dynamics highlights its potential to revolutionize military decision-making processes.

SC-Phi2 [6] relies on extensive fine-tuning to generate StarCraft build orders. In the first stage of fine tuning, the Phi-2 LLM is trained on a question-answer text dataset to correctly answer questions about the StarCraft 2 game. In the second phase, the model is trained to predict build orders from StarCraft game play. SC-Phi2 also leverages visual data of the map to construct a dynamic prompt that includes the current game state in the context provided to the LLM for build order prediction. In contrast, our system uses human-readable documents from the Army sources and FEMA rather than training data. We are investigating solutions to BT generation that build on a common architecture but can be fed appropriate doctrine for the specific simulation domain, and tuned using BTs related to that domain. A common solution could be applied to air or ground for crew, entity semi-automated forces, and constructive simulations.

3 Method

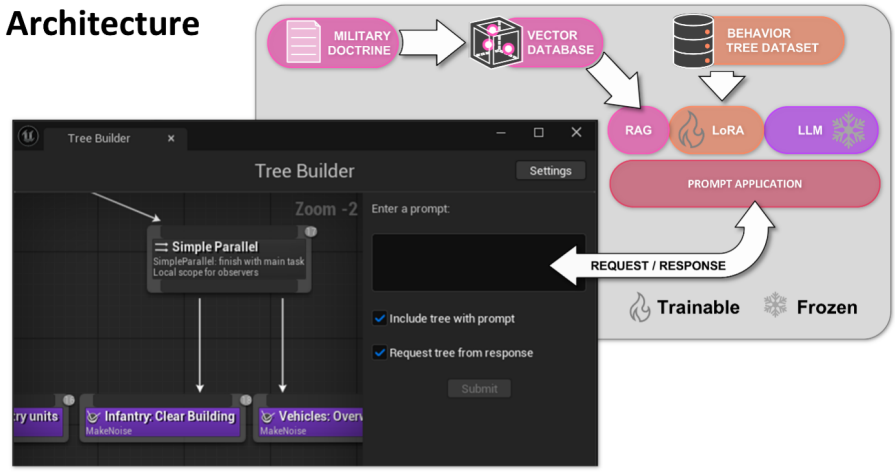

Fig. 2: Our LLM-based behavior tree generation system. Unlike previous work, our system uses RAG to directly leverage existing publicly available military documents.

We have developed a system that leverages LLMs coupled with LoRA adapters to generate both military and emergency management behavior trees, combining doctrinal knowledge with AI to create sophisticated tactical decision-making structures. Figure 2 shows the system architecture. Our base LLM is llama 3.3-70B-instruct, which we run locally on our GPU cluster. We fine-tuned a set of adapters using the BTGenBot dataset [5] to customize the model to output syntactically correct behavior trees. Due to the lack of training examples on military and emergency management doctrine, we provide the model with a vector database containing publicly available training manuals. This serves as the basis for our Retrieval-Augmented Generation (RAG) pipeline, which is crucial for incorporating military doctrine into our behavior tree generation. For each scenario, the system:

1. Retrieves relevant documents from the vector database based on the user prompt
2. Constructs an enhanced prompt by combining the system prompt, context retrieved from the doctrine database, and the user prompt
3. Generates a behavior tree using the LLM model with the fine tuned adapters
4. Performs a syntactic check on the behavior tree

The LLM-generated behavior trees must be manually verified by the human programmer to ensure that the LLM did not hallucinate states and actions that don't exist within the training simulation.

4 Results

This paper evaluates the performance of different prompting strategies at generating syntactically correct behavior trees at higher temperature settings. In LLM API calls, temperature is a critical parameter that controls the creativity of the generated text. By setting the temperature to zero, it forces the model to act as a greedy decoder that chooses the word with the highest predicted probability.

RQ1: How does raising the LLM temperature affect behavior tree generation?

RQ2: Can using a more detailed prompt mitigate syntactic problems caused by higher temperature settings?

Generic Prompt

[INST] «SYS» You are an emergency response planning AI. Generate a behavior tree in strictly valid XML format that models an emergency response plan. The tree should represent actions and decisions typically required in crisis management. «/SYS»
Use the following context about emergency response behavior trees:
{context}
Now, based on this context, generate a behavior tree in XML format for:
{prompt}
The output should be a valid XML tree. [/INST]

Our prompt is composed of a standard system prompt that describes the domain, a user-provided prompt, additional context added from the RAG document retrieval, and some final output instructions. We evaluated the performance of a *generic* prompt that includes less information about the correct BT syntax vs. a *specific* prompt that contains additional specifications about the BT. Note that the LLM plus LoRA adapter can generate syntactically correct BTs even without special prompting, since it was fine-tuned on a dataset of 600 BTs.

We evaluate the complexity of the behavior trees by measuring the depth and also the diversity of actions contained within the tree. Figure 3 shows our analysis of how LLM temperature affects depth and action diversity of the generated behavior trees. Temperature did not have an effect on the depth of the generated behavior tree; however using a more specific prompt with additional specifications results in deeper trees, at all but the lowest temperature setting. Interestingly, increasing the temperature decreased action diversity; at lower temperature settings using the *generic* prompt increased action diversity. Thus we conclude that increasing the LLM temperature is not necessary for improving the complexity or action diversity of specific behavior trees. However, in cases where the aim is to generate multiple different behavior trees for the same scenario, increasing the LLM temperature is critical. Figure 4 (left) shows the effect of temperature on the Jaccard coefficient over the entire set of BTs, generated using the same prompt. The Jaccard coefficient of two sets, A and B, is defined as $J(A,B) = \frac{A \cap B}{A \cup B}$; it is a statistical measure commonly used to gauge similarity

between two sets. Unsurprisingly, as the temperature decreases, the LLM is more likely to generate the same BT in response to the same prompt.

Finally we attempted to characterize the most common types of syntactic errors. The *specific* prompt was very successful in eliminating syntactic errors. Figure 4 (right) shows the distribution of errors in BTs generated using the *generic* prompt. High temperatures results in incorrect XML wrappers, with mismatched tags surrounding the BT. Lower temperatures seemed to occasionally result in the inclusion of plain non-XML text. Mid-range temperatures were more likely to generate smaller, less easily categorized errors.

Specific Prompt

[INST] «SYS»
You are an emergency response planning AI. Your role is to help emergency management teams develop behavior trees (BTs) for handling emergency scenarios such as fires, earthquakes, hazardous materials spills, mass casualty incidents, or search and rescue operations. You generate behavior trees that structure response actions, assign responsibilities, and ensure safe and effective crisis management.«/SYS»
Use the following context about emergency response behavior trees:
{context}
Now based on this context generate a behavior tree in strictly valid XML format for:
{prompt}
- Always contain a ‘<BehaviorTree ID=M̈ainTree>̈‘ tag inside the root
- Contain valid nested nodes such as ‘<Sequence>‘,‘<Fallback>‘, or custom actions.
- Use well-formed and closed XML tags.
- Never include any explanation, description, or text outside the XML.
- Output only the XML and nothing else.[/INST]

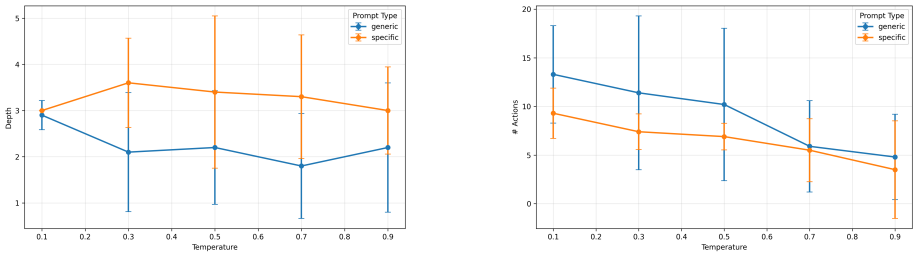

Fig. 3: The effect of temperature and prompt on generated BT depth (left) and action diversity (right). Mean and standard deviation are reported over ten runs.

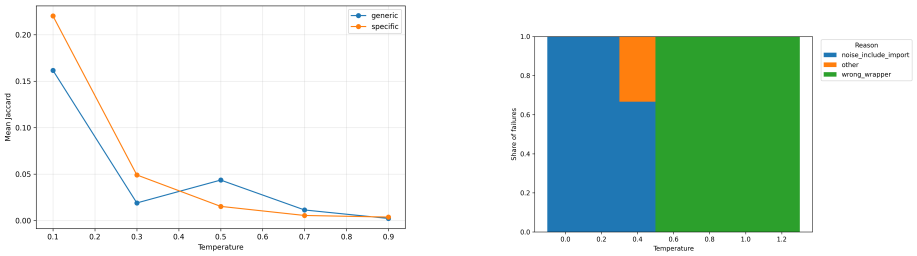

Fig. 4: The effect of temperature and prompt on Jaccard coefficient (left); Distribution of syntactic errors for the *generic* prompt at different temperature levels (right). (The *specific* prompt only produced a single syntactic error.)

5 Conclusion

Authoring the behaviors required to create a rich training environment populated by computer-generated forces is a labor-intensive programming task. The integration of LLMs with behavior trees has made the process of generating complex AI behaviors more efficient and accessible. These models help bridge the gap between expert-driven authoring and user-friendly AI generation by reducing authoring complexity and costs.

This paper analyzed how an LLM's temperature setting affects the complexity, diversity, and syntactic correctness of generated behavior trees (BTs). LLM temperature does not affect the depth of the generated behavior trees. However, using a more specific prompt consistently led to deeper trees, except at the lowest temperature setting. Counterintuitively, increasing the temperature decreased the diversity of actions within a single BT. However, increasing the temperature is crucial when the goal is to generate multiple, different behavior trees for the same scenario. Using a more detailed prompt that includes more information about legal BT structure was effective at eliminating syntactic errors. Our findings suggest that a lower temperature, combined with a more detailed prompt, produces syntactically valid BTs, while maintaining a complex tree structure.

Despite these advancements, challenges remain, such as the paucity of training data and the need for domain-specific expertise for fine-tuning complex behaviors. An LLM cannot fully generate behavior trees from military training manuals without human intervention or example execution traces. This is because, despite detailed action descriptions, the manuals frequently omit specifics about the crucial numeric environmental variables necessary for programming condition nodes. Consequently, human input or example execution traces are required to complement the high-level material provided by the manual. However, the potential to greatly reduce authoring time and costs balances these limitations. Our long-term goal is for such a backend service to provide response BTs to a front-end application, such as the Unreal Engine (UE) Editor, for integration and testing in the simulation.

Acknowledgments. This research was funded by LMCO. Paul J. Foster was instrumental in shaping our research plan. We thank Justin Gifford, Steven Grady, Abhinav Kotta, Ohm Patel, and David Umanzor for their work on the BT generation architecture.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Cao, Y., Lee, C.G.: Behavior-tree embeddings for robot task-level knowledge. In: 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 12074–12080. IEEE (2022)
2. Goecks, V.G., Waytowich, N.: COA-GPT: Generative pre-trained transformers for accelerated course of action development in military operations. In: 2024 International Conference on Military Communication and Information Systems (ICMCIS). pp. 01–10. IEEE (2024)
3. Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W.: LoRA: Low-rank adaptation of large language models. arXiv preprint arXiv:2106.09685 (2021)
4. Iovino, M., Scukins, E., Styrud, J., Ögren, P., Smith, C.: A survey of behavior trees in robotics and AI. Robotics and Autonomous Systems **154**, 104096 (2022)
5. Izzo, R.A., Bardaro, G., Matteucci, M.: BTGenBot: Behavior tree generation for robotic tasks with lightweight LLMs. arXiv preprint arXiv:2403.12761 (2024)
6. Khan, M.J., Sukthankar, G.: SC-Phi2: A fine-tuned small language model for StarCraft II build order prediction. Artificial Intelligence **5**(4), 2338–2352 (Nov 2024)
7. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.t., Rocktäschel, T., et al.: Retrieval-augmented generation for knowledge-intensive NLP tasks. Advances in Neural Information Processing Systems **33**, 9459–9474 (2020)
8. Li, F., Wang, X., Li, B., Wu, Y., Wang, Y., Yi, X.: A study on training and developing large language models for behavior tree generation. arXiv preprint arXiv:2401.08089 (2024)
9. Lykov, A., Dronova, M., Naglov, N., Litvinov, M., Satsevich, S., Bazhenov, A., Berman, V., Shcherbak, A., Tsetserukou, D.: LLM-MARS: Large language model for behavior tree generation and nlp-enhanced dialogue in multi-agent robot systems. arXiv preprint arXiv:2312.09348 (2023)
10. Zhou, H., Lin, Y., Yan, L., Zhu, J., Min, H.: LLM-BT: Performing robotic adaptive tasks based on large language models and behavior trees. arXiv preprint arXiv:2404.05134 (2024)