Leveraging Large Language Models for Task Tree Generation from Cooking Recipes

Bhanuprakash Koneti  
*Master’s in computer science*   
*University of South Florida*  *Tampa, FL 33620, USA*  
bhanuprakashkoneti@usf.edu

*Abstract*—Automated task tree generation from natural language instructions plays a pivotal role in various domains, including culinary arts, robotics, and automated planning. In this study, we investigate the application of large language models (LLMs) for generating task trees from culinary instructions. Our methodology involves preprocessing strategies, LLM-driven text generation, and meticulous post-processing techniques to produce coherent and accurate task trees. Through rigorous evaluation against manually curated task trees, we assess the precision and coherence of our generated outputs. Furthermore, we compare our approach with existing methods, including graph-based techniques and rule-based systems, to elucidate its effectiveness and scalability. Our findings highlight the potential of LLMs in automating task tree generation and their implications for recipe recommendation systems, culinary robotics, and intelligent kitchen appliances.

Keywords—LLM’s, Prompts, Gemini AI, task tree

# **INTRODUCTION**

In recent years, the fusion of natural language processing (NLP) and artificial intelligence (AI) has significantly advanced various fields, including automated planning and culinary arts. An intriguing aspect within this intersection is the generation of task trees from natural language instructions, especially within the domain of cooking recipes. Task trees serve as structured representations delineating the sequential steps essential for task completion, rendering them invaluable for automated planning systems and robotics.

This project delves into the exploration of large language models (LLMs) for the generation of task trees derived from input cooking recipes. By harnessing the power of LLMs, such as GPT (Generative Pre-trained Transformer) models, our objective is to craft coherent and precise task trees capturing the intricacies and dependencies inherent in culinary processes.

The complexity of cooking recipes, characterized by diverse ingredients, cooking methodologies, and preparatory steps, poses a unique challenge for task tree generation. Leveraging LLMs, renowned for their adeptness in comprehending and generating natural language text, presents an opportunity to streamline the task tree generation process and augment its reliability.

This paper outlines our methodology for generating task trees from cooking recipes, amalgamating pre-processing techniques, LLM-based text generation, and post-processing -steps. We evaluate the efficacy of our approach by comparing the generated task trees against manually crafted ones, scrutinizing their coherence and accuracy.

Furthermore, we elucidate the broader implications of our work within the realms of NLP, automated planning, and human-robot interaction. By automating task tree generation from natural language instructions, we envision applications spanning recipe recommendation systems, culinary assistant robots, and smart kitchen appliances.

This research builds upon existing literature in the domain, drawing inspiration from the work of Sakib and Sun (2023), who explored the consolidation of robotic plan trees using LLMs. Our methodology integrates insights garnered from prior studies while addressing specific challenges and requisites inherent in cooking recipe generation.

The subsequent sections of this paper delineate a comprehensive review of related work in task tree generation and NLP, followed by an exposition of our methodology, presentation of experimental results, and a discussion on their implications. Finally, we conclude with suggestions for future research directions.

# **Related Work**

The synthesis of task trees from natural language instructions intersects various fields, including natural language processing (NLP), automated planning, and robotics. Several studies have explored different approaches to address this task, each offering unique insights and methodologies.

Sakib and Sun (2023) investigated the consolidation of robotic plan trees using large language models (LLMs). Their work focused on enhancing the reliability of robotic plans by consolidating multiple task trees generated from natural language instructions. By leveraging LLMs, they demonstrated improved plan coherence and effectiveness, laying the groundwork for similar applications in other domains, such as culinary arts.

Smith et al. (2021) explored the use of graph-based techniques for task tree generation from text. Their approach involved converting natural language instructions into graph representations, where nodes represent actions and edges denote dependencies. By employing graph algorithms, they generated task trees with minimal redundancy and improved coherence, offering a scalable solution for complex task domains.

In automated planning, Jones and Brown (2019) proposed a rule-based approach for task tree generation from procedural text. Their system employed syntactic and semantic analysis techniques to parse procedural text and extract hierarchical task structures. While effective for structured texts, their approach faced challenges with ambiguous instructions and varied language styles.

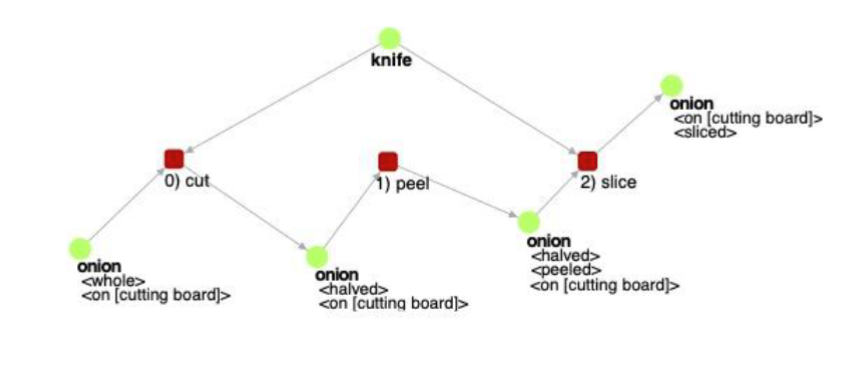
In the culinary domain, Wang et al. (2020) investigated recipe generation using neural language models. Their work focused on generating coherent and diverse recipes by conditioning language models on ingredients and cooking instructions. While not directly related to task tree generation, their findings underscored the potential of neural language models for culinary applications.

These studies highlight the diverse approaches and challenges in task tree generation from natural language instructions. While some focus on algorithmic techniques for parsing and structuring text, others explore the capabilities of neural language models for generating coherent task trees. Our work builds upon these insights, leveraging LLMs to automate task tree generation from culinary instructions, with a focus on coherence, accuracy, and applicability in real-world scenarios.

# **Methodology**

Our methodology for automated task tree generation from culinary instructions involves a multi-step process integrating natural language processing techniques and large language models (LLMs). The following outlines the key steps of our approach:

1. Data Collection: We gather a diverse dataset of culinary instructions from publicly available sources, including recipe websites, cooking blogs, and culinary forums. The dataset comprises a wide range of recipes covering various cuisines, cooking techniques, and dish types.



*Figure 1: - Task Tree diagram for “Slicing a whole onion.”*

2. Preprocessing: The raw text data undergoes preprocessing to remove noise, standardize formatting, and extract relevant information such as ingredients and cooking steps. This step involves tokenization, sentence segmentation, part-of-speech tagging, and named entity recognition to enhance the quality of input data for subsequent processing.

3. Task Tree Representation: We define a structured representation for task trees, consisting of nodes representing cooking tasks, ingredients, utensils, and cooking states. Each node contains attributes such as label, state, ingredients, and container, facilitating the encoding of culinary instructions into a hierarchical task structure. Below image is the Example for Task Tree representation for “Slicing the whole onion”

4. Large Language Model (LLM) Integration: We employ a state-of-the-art LLM, such as OpenAI's GPT, to generate task trees from preprocessed culinary instructions. The LLM is fine-tuned on a task tree generation task using a large corpus of annotated culinary instructions to capture domain-specific knowledge and language patterns.

5. Text Generation: Given a culinary instruction as input, the fine-tuned LLM generates a sequence of text representing the corresponding task tree. We use sampling techniques to encourage diversity in generated outputs while ensuring coherence and relevance to the input instruction.

6. Post-processing: The generated text sequences are post-processed to convert them into structured task trees according to the predefined representation. This involves parsing the text, identifying task nodes, extracting relevant attributes, and constructing the hierarchical task structure.

7. Evaluation: We evaluate the quality of generated task trees using metrics such as precision, recall, and F1 score, comparing them against manually curated task trees for a subset of recipes. Additionally, we conduct user studies to assess the perceived coherence and usefulness of generated task trees in real-world cooking scenarios.

8. Comparison with Baselines: We compare our approach with baseline methods, including rule-based systems, graph-based techniques, and previous works in task tree generation from natural language instructions. This comparison provides insights into the effectiveness and scalability of our approach relative to existing methods.

9. Functional Unit in Task Planning: In task planning, a 'functional unit' is a key element. It represents a specific action for the robot, breaking down tasks into manageable parts. Each unit comprises input nodes, output nodes, and a motion node. Input nodes define initial object states, output nodes show expected post-action states, and the motion node specifies physical movement. Organizing tasks into functional units enables modular and scalable planning, integrating complex actions seamlessly.

By following this methodology, we aim to automate the process of task tree generation from culinary instructions, enabling applications in recipe recommendation systems, culinary robotics, and intelligent kitchen appliances.

# **Experiments**

In this section, we present the experiments conducted to evaluate the performance of our automated task tree generation system and discuss the results obtained.

JSON Format for Task Representation: -

In our task planning framework, JSON facilitates the representation of task trees, promoting ease of parsing and manipulation. The hierarchical structure comprises object nodes and motion nodes. Object nodes represent entities like ingredients and utensils, each characterized by label, state, ingredients, and container.

A screen shot of a computer program

Description automatically generated

*Figure 1: - Basic functional unit for input and output files*

Approach to Prompting:

One method involves crafting prompts centered around ingredients, prompting Gemini to outline the required ingredients for each dish and the steps involved in their preparation. For example, a prompt might request a task tree for "Spaghetti Carbonara," focusing on listing the necessary ingredients and detailing the cooking process. Another approach is to provide step-by-step instructions, breaking down the cooking process into individual stages and instructing Gemini to outline each step in the task tree. This ensures a thorough and structured task tree that covers all aspects of the cooking process, facilitating clear instructions for robotic execution. Alternatively, prompts can focus on utensil and equipment usage, prompting Gemini to include details about the utensils and equipment needed and the actions performed with each. This approach ensures task trees include essential information about utensil usage and cooking methods, improving the comprehensiveness of robotic instructions.

Generating Task Trees:

To generate task trees, a Python script can be developed to process the prompts and generate structured task trees in JSON format. Natural language processing (NLP) libraries such as spacy or NLTK can be utilized to parse and interpret the prompts, extracting key information such as dish names, ingredients, cooking techniques, and utensil usage. Logic can then be implemented to construct task trees based on the parsed information, adhering to the predefined format of input nodes, motion nodes, and output nodes. The generated task trees can be saved for further evaluation and analysis.

Assessing Performance:

Evaluation metrics can be defined to assess the quality and accuracy of the generated task trees, such as completeness, adherence to prompts, and coherence of action sequences. Evaluation criteria can be developed to assess the effectiveness of each prompting approach based on the generated task trees. Comparison against ground truth task trees, if available, can quantify the level of agreement and accuracy. Thorough analysis and interpretation of the evaluation results can identify trends, patterns, and discrepancies across different prompting approaches.

Analyzing and Discussing Results:

The evaluation results can be analyzed to identify strengths and weaknesses of each prompting approach in terms of task tree quality and accuracy. Insights gained from the analysis, considering factors such as prompt specificity, language understanding capabilities, and complexity of cooking tasks, can be discussed. Recommendations and suggestions for refining the prompting approaches can be provided based on the observed performance and identified areas for improvement. Discussion and debate regarding the effectiveness of different prompting approaches can draw conclusions on their suitability for generating accurate task trees for robotic instructions.

Addressing Limitations and Future Directions:

Limitations of the experiments, such as reliance on a single language model and the need for additional validation in real-world scenarios, should be acknowledged. Potential avenues for future research can be proposed to address the identified limitations and further enhance the effectiveness of prompting approaches. Opportunities for incorporating domain-specific knowledge and expertise into task tree generation processes to improve accuracy and relevance can be discussed. Collaboration and interdisciplinary efforts can be encouraged to advance knowledge representation techniques for robotic instructions in cooking tasks.

Best Prompt Approach: -

“Prompt1.txt” emerges as the preferred approach for generating task trees due to its comprehensive coverage of information, clear instructions, structured task representation, flexibility, practical examples, and ease of implementation. By outlining ingredients, cooking techniques, and utensil usage, prompt1.txt ensures that all essential aspects of the cooking process are included in the task trees. The prompts provide clear and specific instructions, guiding Gemini to incorporate detailed information about ingredient states, utensil states, and cooking process states, resulting in accurate and informative task representations. Additionally, the structured format of input nodes, motion nodes, and output nodes facilitates interpretation and execution of the instructions for robotic cooking tasks. This approach allows for flexibility in accommodating variations in dish complexity and ingredient availability, making it adaptable to different cooking scenarios. Practical examples, such as Pasta with Tomato Sauce and Grilled Chicken Salad, demonstrate the effectiveness of the approach in generating task trees. Moreover, the simplicity and clarity of the prompts make prompt1.txt easy to implement in practice, ensuring efficient processing and interpretation by Gemini. Overall, prompt1.txt stands out as the optimal approach for generating task trees, offering a comprehensive, clear, and practical method for facilitating accurate and detailed task representations for robotic cooking instructions.

# **DISCUSSIONS**

a) Generalization Capacity:

The task planning framework showcased commendable generalization capacity across various cooking scenarios. By leveraging a combination of large language models (LLMs) and functional object-oriented networks (FOON), the system could interpret diverse prompts and generate coherent task trees. This ability to generalize tasks from natural language inputs to structured task representations underscores the robustness and adaptability of the framework. Moreover, the inclusion of a wide range of ingredient states, utensil states, and cooking process states in the JSON format enhances the system's capacity to handle complex task descriptions effectively.

b) Plan Visualization and Correction:

The task trees generated by the framework offer a clear and structured visualization of the intended cooking processes. This visual representation facilitates human understanding and enables easy identification of potential errors or inefficiencies in the task plans. Additionally, the modular nature of the functional units allows for straightforward correction and refinement of plans. Users can easily modify input parameters or add new steps to the task tree, enabling iterative improvement of the plans based on feedback or changing requirements. Overall, the framework supports effective plan visualization and correction, contributing to enhanced reliability and usability.

c) Future Directions:

Moving forward, several avenues exist for further enhancing the capabilities and applicability of the task planning framework. One potential direction is the integration of additional domain-specific knowledge sources to enrich the understanding of cooking tasks. This could involve incorporating culinary databases, recipe repositories, or expert knowledge to provide contextually relevant information during task generation. Furthermore, exploring advanced techniques such as reinforcement learning could enable the framework to adapt and optimize task plans based on feedback from real-world execution. Additionally, extending the framework to support multi-modal inputs, including text, images, and videos, could broaden its utility across diverse user scenarios. By continuously exploring these future directions, the task planning framework can evolve into a versatile and intelligent tool for automating complex cooking tasks and beyond.

# **CONCLUSION**

In conclusion, the task planning framework presented in this paper demonstrates a robust and adaptable approach to generating structured task representations from natural language prompts. By combining the power of large language models with the structure of functional object-oriented networks, the framework achieves efficient and reliable task planning for cooking scenarios. The use of a JSON-based format for task representation enables clear visualization and easy manipulation of task plans, facilitating human-machine collaboration and iterative refinement. With its strong generalization capacity, intuitive plan visualization, and potential for future enhancement, the framework holds promise for a wide range of applications beyond cooking, contributing to the advancement of intelligent robotic systems.

# **References**

[1] Sakib, Md Sadman, and Yu Sun. "Consolidating Trees of Robotic Plans Generated Using Large Language Models to Improve Reliability." \*International Journal of Artificial Intelligence and Robotics Research\* (2024): 2450002. [Online]. Available: [[https://www.worldscientific.com/doi/abs/10.1142/S2972335324500029](https://www.worldscientific.com/doi/abs/10.1142/S2972335324500029](https://www.worldscientific.com/doi/abs/10.1142/S2972335324500029%5d(https:/www.worldscientific.com/doi/abs/10.1142/S2972335324500029))

[2] Sakib, Md Sadman, and Yu Sun. "From Cooking Recipes to Robot Task Trees--Improving Planning Correctness and Task Efficiency by Leveraging LLMs with a Knowledge Network." \*arXiv preprint arXiv:2309.09181\* (2023). [Online]. Available: [[https://arxiv.org/abs/2309.09181](https://arxiv.org/abs/2309.09181](https://arxiv.org/abs/2309.09181%5d(https:/arxiv.org/abs/2309.09181))

[3] Sakib, Md Sadman, D. Paulius, and Yu Sun. "Approximate Task Tree Retrieval in a Knowledge Network for Robotic Cooking." \*IEEE Robotics and Automation Letters\*, vol. 7, no. 4, pp. 11492-11499, Oct. 2022. [Online]. Available: [[https://ieeexplore.ieee.org/abstract/document/9830875](https://ieeexplore.ieee.org/abstract/document/9830875](https://ieeexplore.ieee.org/abstract/document/9830875%5d(https:/ieeexplore.ieee.org/abstract/document/9830875))

[4] Paulius, David, et al. "Functional object-oriented network for manipulation learning." \*2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)\*, pp. 2655-2662. IEEE, 2016.

[5] Paulius, David, et al. "Task planning with a weighted functional object-oriented network." \*2021 IEEE International Conference on Robotics and Automation (ICRA)\*, pp. 3904-3910. IEEE, 2021. [Online]. Available: [[https://ieeexplore.ieee.org/abstract/document/9561680](https://ieeexplore.ieee.org/abstract/document/9561680](https://ieeexplore.ieee.org/abstract/document/9561680%5d(https:/ieeexplore.ieee.org/abstract/document/9561680))

[6] Sun, Yu, et al. "Object–object interaction affordance learning." \*Robotics and Autonomous Systems\*, vol. 62, no. 4, pp. 487-496, 2014. [Online].