Faculty Of Engineering and Natural Sciences
Department of Electrical and Electronic Engineering
April 24, 2025

Integrating Large Language Models for Robot Task Planning

Almira Demirkıran 200202150

E-mail: almira.demirkiran@std.antalya.edu.tr

Mehmet Tosun 190202009

E-mail: mehmet.tosun@std.antalya.edu.tr

Abdallah Rasthy 200201125

E-mail: ali.rashty@std.antalya.edu.tr

Rustam Akhmedov 210201113

E-mail: rustam.akhmedov@std.antalya.edu.tr

Contents

Large Language Models (LLMs), and particularly those that employ Transformer architecture, have significantly improved the ability of machines to understand and generate human language. Their incorporation into robot systems presents with it the potential for machines to execute complex task planning from natural language inputs. The present weekly report gives a technical overview of the basic mechanisms in LLMs and how they get applied in robot task interpretation and planning. The focus is on learning about the attention mechanism, LLM architecture, and how they work internally through Transformer blocks, and how these capabilities can be applied in robotics.

2. Attention Mechanism

The attention mechanism, the foundation of transformer models, cannot be done without in LLMs' comprehension of contexts. We studied the "Scaled Dot-Product Attention" mechanism this week, which transforms input tokens to query (Q), key (K), and value (V) matrices to get their weights. The attention score is computed as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

where (d_k) is the size of the key vector. Multi-Head Attention allows parallel learning of many contexts, e.g., between "kitchen" and "glass" in the instruction "pick up the glass in the kitchen." A Python implementation was devised to try out this mechanism:

Code Example:

```
import torch
import torch.nn as nn

class Attention(nn.Module):
    def __init__(self, dim):
        super(Attention, self).__init__()
        self.scale = dim ** -0.5

    def forward(self, query, key, value):
        scores = torch.matmul(query, key.transpose(-2, -1)) * self.scale
        weights = torch.softmax(scores, dim=-1)
        return torch.matmul(weights, value)

dim = 64
attention = Attention(dim)
query = torch.rand(1, 10, dim)
key = torch.rand(1, 10, dim)
value = torch.rand(1, 10, dim)
output = attention(query, key, value)
print(output.shape)  # Output: torch.Size([1, 10, 64])
```

This code simulated the attention mechanism's role in processing instructions, demonstrating its foundation for robots' understanding of complex commands.

3. Main Stages of LLM Structure and Block Diagram

The LLM structure comprises four main stages:

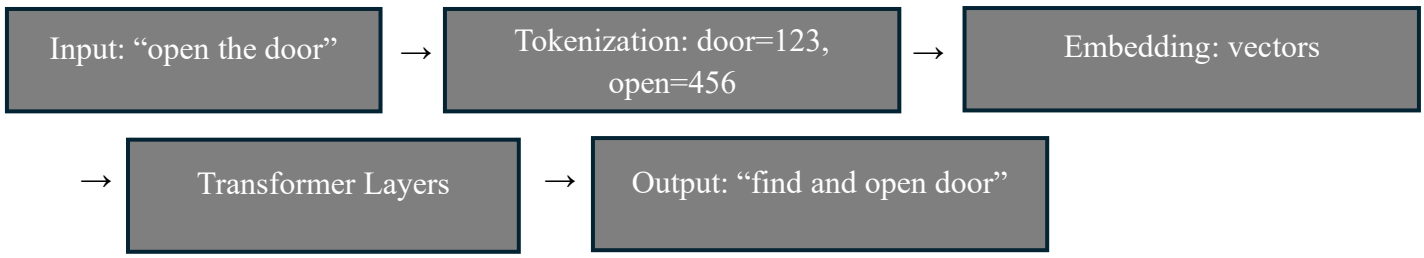
1.Tokenization: Input text is split into tokens (e.g., "open the door" → [open, door]).

2.Embedding Layer: Tokens are converted into vectors representing their meanings.

3.Transformer Layers: Attention mechanisms and feed-forward networks learn contextual relationships.

4. Output Layer: A task plan or response is generated.

These stages were depicted in a block diagram for the command "open the door":



4. How Transformer Architecture Realizes LLM Objectives

Transformer architecture is a foundation framework for LLMs to read natural language instructions and translate them into robot task plans. The architecture serves to fulfill LLM objectives, such as contextual learning and modeling long-range dependencies. Through its attention mechanism and stacked layer design, transformers enable robots to understand complex context in instructions and generate appropriate action sequences.

Multi-Head Attention is the core architectural component of the transformer that contrasts the relations of all tokens in the input sequence and infuses those that are relevant context-wise. The attention mechanism is computationally the calculation of query, key, and value vector interactions to find contextual dependencies among tokens. Multi-Head Attention extends this to perform multiple parallel computations of attention, which allows the model to capture a variety of contextual relations simultaneously. This is particularly helpful in encoding the nuances of complex instructions.

Transformer architecture consists of encoder layers and decoder layers. Encoder layers encode the context of an input instruction, and decoder layers convert this context into a task plan or sequence of actions. Each layer blends the attention mechanism with feed-forward neural networks (FFNNs) and layer normalization, recursively enhancing input token contextual representations. This design enables appropriate modeling of long-range dependencies, i.e., connections between distant tokens in the input sequence, enabling full contextual understanding.

The primary goal of LLMs for robotic task planning is the interpretation of natural language instructions to generate meaningful and executable action sequences. The transformer model achieves this by analyzing the semantic structure of instructions and generating task plans consistent with the environmental context. The attention mechanism emphasizes relevant tokens (e.g., action or object words) within the instruction, delineating the sequence and manner of actions the robot should perform. Further, the hierarchical structure of transformers facilitates multi-level instruction processing—syntactic, semantic, and pragmatic—enabling robots to interpret instructions accurately.

Contextual learning abilities of transformer architectures were discussed this week, particularly models such as GPT-2, which take advantage of a 12-layer model and a large count of parameters. These models facilitate the intricacy of contextual analysis, converting instructions into vector forms that are actionable through task planning. The abilities of the transformer made it possible for robots to correctly understand natural language instructions and perform relevant actions within the physical environment.

5. Robot Instruction Interpretation and Task Planning Process

The robot instruction interpretation process involves:

1.Instruction Input: A natural language instruction is given.

2.Semantic Analysis: The LLM tokenizes the instruction and derives context with the help of transformer layers.

3.Task Planning: The instruction is translated into a hierarchical action sequence:

- Objects and actions are identified.
- An action sequence is generated (e.g., find object, grasp object, move object).

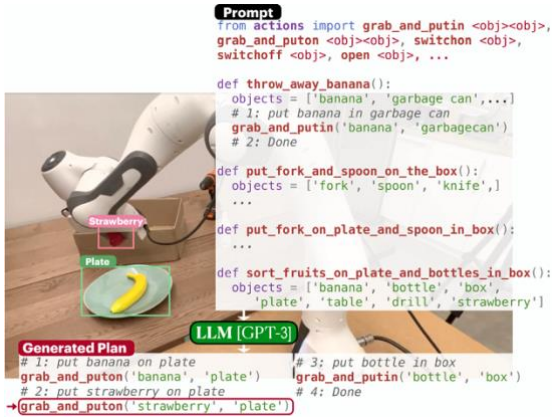


Figure 1: A robot executing the instruction "sort_fruits_on_plate_and_bottles_in_box()" and following through with the action plan. The figure illustrates the processing of the instruction at the code level and the resulting action sequence (e.g., "put banana on plate," "put strawberry on plate").

4.Action Execution: The plan is executed by the robot in the physical world.

This process ensured proper interpretation and optimization of task planning. Optimization methods for generalization of the process to complex instructions were discussed [4].

6. Task Planning Types and Contribution of LLM

Robot task planning is categorized as:

- **Hierarchical Planning:** Decomposition of tasks into subtasks.
- **State-Based Planning:** Plans based on environmental states.
- **Learning-Based Planning:** LLMs apply large datasets to generate flexible plans.

LLMs work best in learning-based planning, and transformer attention mechanisms aid context-sensitive instruction processing and adaptive plan generation.

7. Conclusion

The week witnessed significant advancements in mathematically modeling the attention mechanism, visualizing LLM architecture, exploring the transformers' role in task planning, and developing processes for instruction interpretation. Python scripts were employed to experiment with attention mechanisms and instruction processing, block diagrams were established, and processes were mapped. Next week, we experiment with these innovations using complex instructions and real robotic scenarios.

References

1. Vaswani, A., et al. (2017). “Attention is All You Need.” Advances in Neural Information Processing Systems.

2. Devlin, J., et al. (2018). “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.” arXiv preprint arXiv:1810.04805.

3. Brown, T. B., et al. (2020). “Language Models are Few-Shot Learners.” Advances in Neural Information Processing Systems.

4. Radford, A., et al. (2019). “Language Models are Unsupervised Multitask Learners.” OpenAI.