Proceedings of the 4th International Conference on Software Engineering and Machine Learning
DOI: 10.54254/2755-2721/2026.34759

Research on Automatic Generation and Optimization Methods for Large Model-Driven Agent Code

Jian Zhao

Department of Computer Science, Changzhi University, Changzhi, China, zhaojian@czc.edu.cn

Abstract. With the breakthrough of large language models in recent years, their strong ability in understanding, generating and reasoning have influenced the development of robotics. Robots still face challenges in complex dynamic environments, including ambiguous natural language instruction parsing, insufficient task allocation efficiency, and limited environmental adaptability. This paper is based on the National Key R&D Program of "Human-Robot Intelligent Fusion Technology". In office scenarios, it discuss and research on methods for robot natural language instruction parsing and intelligent decision-making in dynamic environments.

Keywords: Natural language commands, Service robots, Large language models, Human-computer interaction, Intelligent agents

1. Introduction

This paper deeply study dynamic task allocation methods for service robots under natural language instructions and methods for autonomously generating navigation tasks from conversations.It develop an active intelligent assistant driven by large language models.This enrich and improve the theoretical system of robot intelligent control.This paper introduce large language models into service robots [1]. It improve robots'ability to understand and process natural language. It also enhance robots'ability to perceive dynamic environmental changes . This paper use general large language models to effectively improve robots'understanding accuracy and semantic analysis ability of natural language instructions.It study dynamic task allocation mechanisms in multi-robot systems.This push forward the development of distributed intelligent system theory.It provide new ideas and solutions for solving dynamic task coordination and resource allocation problems in multi-agent cooperation.

2. Pre-trained large language models

Transformer is a neural network model based on self-attention mechanism.It solve the problem of long-distance dependency that traditional recurrent neural networks have. [2] Transformer mainly includes an encoder and a decoder. The encoder turn input sequences into hidden representations.The decoder turn hidden representations into outputs.The architecture is shown in Figure 1.

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2026.34759

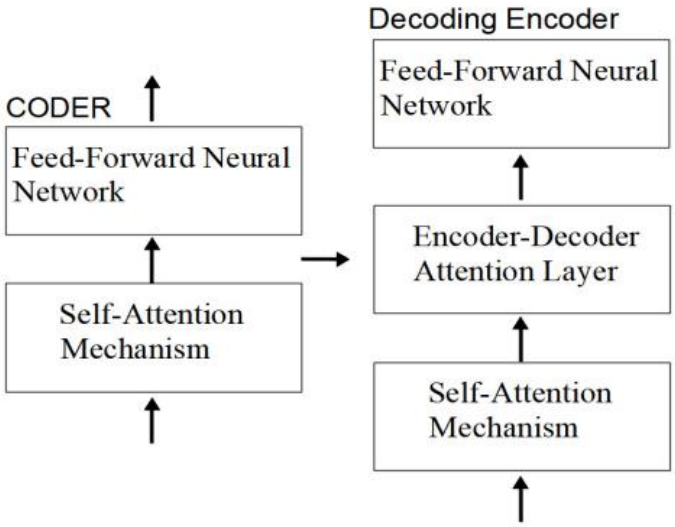

Figure 1. Transformer architecture

Transformer model mainly includes self-attention mechanism,positional encoding,encoder and decoder,multi-head attention,feed-forward neural network,and residual connection and layer normalization.Its core idea is to use attention mechanism for global information interaction across all positions in the input sequence. [3] This allow the model to capture long-distance dependency relationships within a sequenc. For the input sequence,each token is converted to a vector through an embedding layer,and positional encoding is added to keep sequence information.The calculation formula of self-attention mechanism is:

$$\text{Attention}\ (\text{Q,K,V}) = \text{Softmax}(\frac{\text{QK}^{\text{T}}}{\sqrt{\text{d}_{\text{x}}}})\text{V} \tag{1}$$

The calculation formulas corresponding to Q, K, and V are:

$$\begin{aligned} Q &= XW^{Q} \\ K &= XW^{K} \\ V &= XW^{V} \end{aligned} \tag{2}$$

3. Code generation-related technologies

The AST of a method is a tuple(N,T,X,S,δ,φ),where N is a set of non-terminal nodes,T is a set of terminal nodes,X is a set of values,S∈N is the root node,δ:N→(N∪T)*is a function that maps a non-terminal node to a list of its child nodes,*denotes the closure operation,andφ:T→X is a function that maps a terminal node to a related value.Except for the root node,each node appears only once in all child node lists.

DOI: 10.54254/2755-2721/2026.34759

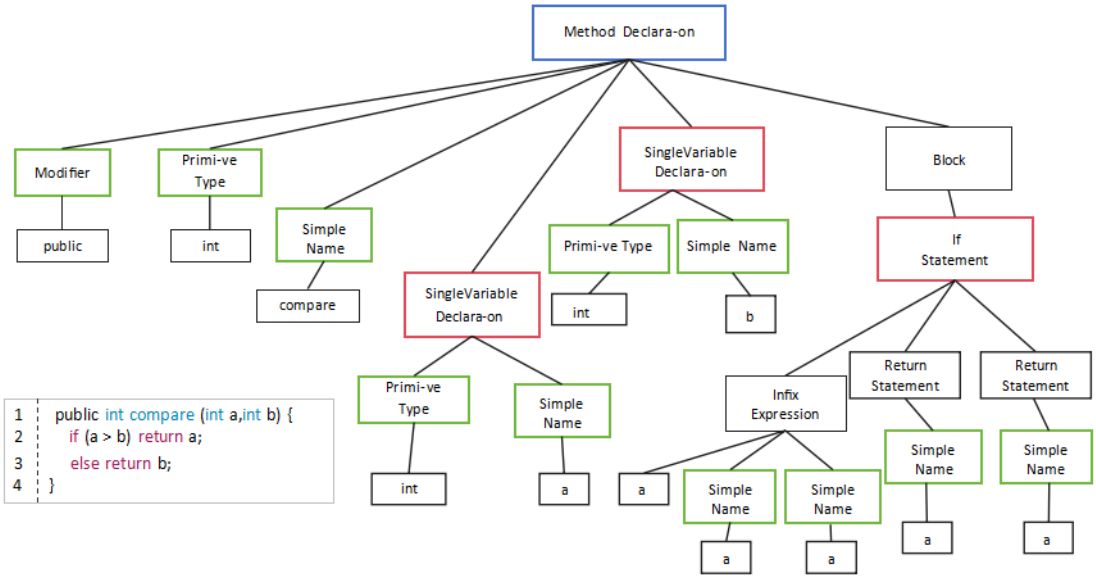

Figure 2. Source code snippet and an example of the AST parsed using JDT

During message propagation,each node collects information from its surrounding neighborhood.This process can be visualized as a tree-like structure,where the neighborhood around the target node is progressively expanded.For example,in a two-layer message passing model,the model first aggregates messages from the local graph neighbors of the target node [4, 5]. These messages are further passed and updated based on information aggregated from their respective neighborhoods.The calculation formula for message passing is shown below:

$$h_u^{k+1}=U^K(h_U^k,AGG^k(\{h_v^k,V_v\in N(u)\})) \tag{3}$$

Based on these aggregated neighborhood information,the U function first updates the message $h_u^{(k)}$,then combines it with the previous message $h_u^{(k+1)}$。

Sparse retrieval technology calculates similarity between queries and documents through lexical overlap. A representative algorithm is BM25. BM25 measures the relevance between documents and queries using term frequency and inverse document frequency. Specifically, a query as natural language input is initially divided into multiple tokens $\{t_1, t_2, \cdots, t_n\}$, and the relevance is calculated as:

$$score(q,d)=\Sigma_{i=1}^{n}(w_i R(t_i,d)) \tag{4}$$

4. Context Retrieval within projects based on tool calls

The lack of project context information is the main reason why large language models cannot accurately understand feedback and correctly fix code.Therefore,this section design a tool-enhanced large language model agent.It equip the agent with various code tools,such as getting class definitions.Through agent strategy,the agent can autonomously call tools according to error messages.It retrieve relevant context information within the project.This optimize code generation

DOI: 10.54254/2755-2721/2026.34759

results.Next,this section elaborate from three aspects:algorithm description,code analysis tools,and agent tool usage.

Table 1. Context-based retrieval optimization algorithm for projects based on tool calls

Algorithm: Context-based retrieval optimization algorithm for projects based on tool calls	
Input:Natural language function description r;Iteration round iteration_num;Compiler C,large language model agent M,tool execution commandεOutput:Optimized code y_{i+1}	
1	$y_1 = M(r)$
2	for i=1 to iteration_num do
3	$fb_i = C(y_i, test_case_i)$
4	if all_tests_pass(fb_i) then
5	return y_i
6	end if
7	$T_i = M(r,y_i,fb_i)$
8	$t_i = \varepsilon(T_i)$
9	$y_{i+1} = M(r,y_i,fb_i, t_i)$
10	end for
11	return y_{i+1}

Specifically, in the first generation stage, the large language model M makes a first guess based on the current function signature and function description info r to get a generation result. Because the info given at this point is limited, the model easily make wrong outputs, like using undefined names or data structures in the project. So, we use a compiler together with test cases to check if the current result is correct:

$$fb_i = C(y_i, test_case_i) \quad (5)$$

When the feedback information successfully passes all test cases or has reached the maximum iteration rounds we set, the iteration ends and the current prediction result yi is saved. Otherwise, the LLM will guide the optimization of the next round based on the feedback information fb_i.

5. Autonomous navigation

Autonomous navigation technology mainly covers global planning and local obstacle avoidance. In the global planning stage, the robot uses a pre-built environment map and calculates the optimal path from start to goal through graph search algorithms such as Dijkstra or A*. This path can be mathematically defined as:

$$J = \sum_{i=0}^{n-1} C(x_i, x_{i+1}) \quad (6)$$

DOI: 10.54254/2755-2721/2026.34759

Where $\{x_o, x_i, ..., x_n\}$ is the set of path points,and $C(x_i, x_{i+1})$ is the cost between adjacent nodes(such as Euclidean distance,energy consumption,or risk weight).

Local obstacle avoidance relies on real-time sensor data. It uses methods like Dynamic Window Approach to calculate safe motion commands that stay close to the globally planned path in real time. Its decision formula can be described as:

$$v^* = \arg\max_{v \in V}(\alpha.\text{heading}(v) + \beta.\text{velocity}(v) - \gamma.\text{obstacle}(v)) \tag{7}$$

CodeBLEU is an evaluation metric based on reference code matching. It considers both the syntactic accuracy and logical accuracy of generated code. Its calculation formula is:

$$\text{codeblue} = \alpha \times \text{NGram Match(BLEU)} + \beta \times \text{weighted NGram Match} \tag{8}$$

$$+ Y \times \text{syntaxMathching} + \delta \times \text{semanticDataflowMatch}$$

Where v=(v linear,v angular)is the velocity command;V is the feasible velocity set; α , β , γ are weight parameters that measure the requirements for target direction,motion speed,and obstacle avoidance respectively.The organic combination of global and local planning ensures that robots can pursue optimal paths while avoiding sudden obstacles in real time in complex,dynamic environments.

6. Conclusion

Generating executable code from natural language can greatly help developers work faster and cut software costs. This paper look at project-level code generation using large language models. It is harder than simple coding tasks because it need cross-file context. Current LLMs do not do well here. We study this from two sides: single-round generation and multi-round optimization. We find problems in both areas and propose solutions. Experiments show they work. By using project context inside and outside together with iterative strategies, our method make LLMs better at improving code through feedback.

Funding information

This work was supported by the School-level Scientific Research Project of Changzhi University under Grant No. XN0759 and the School-level Teaching Reform Project of Changzhi University under Grant No. JC202523.

References

[1] Zhang J , Liao J , Tang J , et al.The Power of Small LLMs: A Multi-Agent for Code Generation via Dynamic Precaution Tuning [J].IEEE Transactions on Software Engineering, 2026, 52(1): 86-99.DOI: 10.1109/TSE.2025.3632508.

[2] Awal A M , Rochan M , Roy K C .A metamorphic testing perspective on knowledge distillation for language models of code: Does the student deeply mimic the teacher? [J].The Journal of Systems & Software, 2026, 240112952-112952.DOI: 10.1016/J.JSS.2026.112952.

DOI: 10.54254/2755-2721/2026.34759

[3] Truong J , Nguyen V , Nguyen T T .An overview of evaluation and enhancement methods for code generation by large language models [J].Information and Software Technology, 2026, 197108185-108185.DOI: 10.1016/J.INFSOF.2026.108185.
[4] Wang Z , Ma J , Cai J , et al.Point2Brick: Automating building semantic model creation via multi-stage confidence-aware Large Language Model agents [J].Energy & Buildings, 2026, 362117516-117516.DOI: 10.1016/J.ENBUILD.2026.117516.
[5] Luo S, Kong X, Xie X, et al.Monitoring and Early Warning of the Cage-Rearing Broiler Farming Environment Based on an Inspection Robot [J].Animals, 2026, 16(9): 1417-1417.DOI: 10.3390/ANI16091417.