IntentGuide: Neuro-Symbolic Integration for Clarifying Human Intents by Rewriting Free-Form Sentences

Nikita Kiran Yeole
Computer Science
Virginia Tech, Blacksburg, USA
nikitay@vt.edu

Michael S. Hsiao
Electrical and Computer Engineering
Virginia Tech, Blacksburg, USA
hsiao@vt.edu

***Abstract*—This paper presents "IntentGuide," a neuro-symbolic integration framework to enhance the clarity and executability of human intentions expressed in free-form sentences. IntentGuide effectively integrates the rule-based error detection capabilities of symbolic AI with the powerful adaptive learning abilities of Large Language Model (LLM) to convert ambiguous and/or complex sentences into clear, machine-understandable English instructions. The empirical evaluation of IntentGuide, performed on natural language sentences written by middle school students for designing video games, reveals a significant improvement in error correction and code generation abilities compared to previous approaches, attaining an accuracy rate of 90%.**

***Index Terms*—Natural language processing, Natural language programming, neuro-symbolic**

I. Introduction

Fundamentally, Natural Language Processing (NLP) aims to parse, process, and understand human language, complete with all of its nuances, for applications ranging from question answering to code generation. The primary challenge still stands, though: how can we enable computers to process poorly written or difficult sentences in a meaningful way? In addition, can the system provide feedback on why it chose to interpret the sentence in a particular manner? Such feedback can also serve to help the user adjust their original sentence if the original sentence was in fact misinterpreted. Transforming vague, ambiguous, and possibly illogical free-form phrases and sentences into precise, machine-understandable statements is a crucial component of this problem [1].

The core issue lies in the inherent properties of natural language (NL): ambiguity, incompleteness, and verbosity of the NL text. For example, in terms of ambiguity, a single phrase can have multiple meanings depending on the context, the user's intent, etc [2]. As a result, potential errors in conveying that intent or omissions in the detailed instructions needed by a compiler may render the generated code inaccurate or incomplete. This does not fit well with the unambiguous nature required by traditional compilers.

To overcome these challenges, NL instructions must often be decomposed into a sequence of simpler, unambiguous statements that a machine can understand and process. This raises the question: is it possible to rewrite free-form sentences such that the machine adapts more efficiently to them?

In this paper, we introduce IntentGuide, a novel approach designed to refine and clarify user prompts before they are converted to code. IntentGuide utilizes a neuro-symbolic framework. This framework processes free-form NL inputs, systematically transforming them into a sequence of unambiguous, clear NL sentences. This transformation not only clarifies the user's original intentions but also enhances the fidelity of the resulting code generation. Nevertheless, simply asking the AI to rewrite a sentence into a complete, unambiguous and concise sentence is quite a challenge [3] [4].

In order to correctly rewrite the original sentence, it is essential to know what phrases are problematic. This is where we combine the salient features of the two AI engines. The first symbolic engine checks for any violation to the constraints required for clear, precise and unambiguous sentences. The result of this step are messages that detail distinct information that will direct the rewriting of free-form sentences into unambiguous ones in the second step with the LLM. Through this integration, our system enhances its capacity to identify NL ambiguities, errors, and conflicting concepts. In addition, it offers a way to re-write the original sentences into a sequence of one or more unambiguous sentences [5] [6].

We applied IntentGuide to create games on an educational platform called GameChangineer. Users can design games on this platform, which uses simple English sentences as its operating language [7] [8] [9]. The results of IntentGuide, performed on NL description written by middle school students for video games, reveals a significant improvement in error correction and code generation abilities compared to previous approaches, attaining an accuracy rate of 90%.

II. Methodology

The symbolic AI is embedded within GameChangineer to check for adherence to rules and violation against any constraints. Next, error correction is conducted via an LLM. These two separate phases of the proposed system's operation

supported in part by NSF grant 2101021

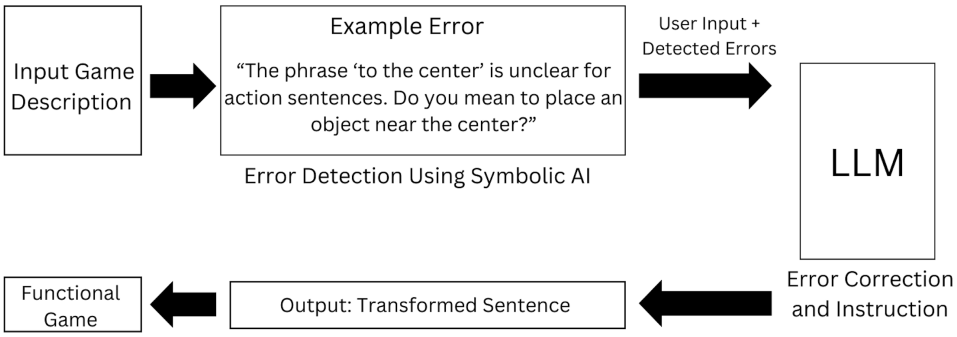

Fig. 1. Process flow of the Methodology

form the end-to-end rewriting of problematic sentences. Figure 1 shows the flow of our approach.

The first phase of IntentGuide manages the identification and classification of errors in the students' NL inputs. This phase employs symbolic AI techniques that are designed to work with the predefined grammatical, semantic, and logical rules to thoroughly analyze user inputs. The errors are categorized into several types, similar to the errors identified in the research [10]. The 4 types of errors are:

1) Syntax Errors: grammatical and structural errors.
2) Semantic Errors: ambiguous or improperly used words, leading to confusion about the intended meaning.
3) Logical Errors: inconsistencies or contradictions within the logic presented in the user input.
4) Lexical Errors: use of unknown, incorrect, or misspelled words.

Phase 2: Error Correction and Instruction Using LLM (GPT-4)

The feedback generated from the symbolic AI system are given to the second phase using OpenAI's GPT-4. We use OpenAI's GPT-4 API with a fixed temperature setting of 0 for consistent output. The API interactions are managed through Python scripts that format the input sentences and parse the output for further processing. This stage not only fixes the errors found by the Symbolic AI but also instructs users by providing step-by-step guidance on how to improve their inputs. This is how this phase goes:

1) Integration with GPT-4: The original sentences are sent into GPT-4 via the OpenAI's API and the Python programming language. The temperature parameter of the model is set to 0, which encourages deterministic and consistent output. This setting is crucial for generating precise and unambiguous corrections without variability in the responses, ensuring the integrity of the instruction.
2) Sentence Rewriting: This step does not involve the direct use of feedback from the symbolic AI in the initial rewriting process. Instead, GPT-4 focuses on rewriting the sentences using a series of predefined prompts. This step makes sure the model improves the sentences according to linguistic best practices instead of concentrating only on the errors that the symbolic AI logs. This step is essential because, despite its effectiveness, symbolic AI has limitations and may not capture all nuances or contextual meanings, potentially missing subtle yet significant aspects of language usage.
3) Feedback Integration and Further Rewriting: Following the first round of rewriting, the procedure entails a crucial integration of the GPT-4-rewritten phrases with the feedback from the symbolic AI. The feedback (including syntactic, semantic and logical errors), along with the original and initially rewritten sentences, are passed into GPT-4. This comprehensive input allows the LLM to understand both the initial sentence and prior attempts to correct it. After that, the LLM carefully examines the rewritten sentence in light of the particular errors identified. This focused strategy makes sure that the LLM's output corrects any error that the symbolic AI found. The LLM rewrites the sentence if it finds that the rewritten version does not sufficiently address the identified errors or if it introduces new ambiguities. This iteration specifically aims to resolve the errors, refining the sentence into a clearer and more precise version.
4) Educational Feedback and Lesson Generation: The final step of the process focuses on educational enhancement and user learning, where IntentGuide generates a step by step lesson based on the transformations from original user input to a sequence of unambiguous and clear sentences. The lesson focuses on every particular modification made to fix the errors that the symbolic AI found. These lessons are designed to educate the users about the principles of clear and effective communication in programming. By understanding the corrections made, students learn how to avoid common mistakes and improve their natural language programming skills.

For further details on the prompts and dataset used in this phase, please refer to our GitHub repository.

Let us look at the following free-form sentence: *"If rabbit collides with the edge they move to the center."*

By its nature, the strength of symbolic AI is in uncovering violations to a set of rules or constraints. The symbolic AI in

Phase 1, thus, generates the following feedback:

- *the phrase 'to the center' is unclear for action sentences. Do you mean to place an object near the center? And then have the character move to the object at the middle of the canvas?*
- *Cannot move in an unspecified direction based on reaching any border. Reversing direction is interpreted.*
- *Do you mean 'When [obj1] sees [obj2], [obj1] chases [obj2].', in which [obj2] is placed in the middle of the canvas?*

By giving the above feedback and the originial sentence to the LLM in Phase 2, we obtain the resulting final sequence of sentences:

Rewritten Sentence: *"There is an invisible diamond in the center. When the rabbit collides with the border, it becomes alert. When the rabbit is alert, it moves towards the diamond."*

The original sentence contains ambiguity by using the phrase "move to the center". Diverse interpretations of the term "center" and varied implementations of the process of "moving towards it" could result in misunderstanding or inaccuracies in execution. The introduction of an "invisible diamond" at the center of the canvas provides a concrete target for the rabbit's movement. This aligns with the suggestion to clearly define a destination or object to interact with. Introducing the state of being "alert" after colliding with the border breaks down the rabbit's response into two stages. First, the rabbit becomes alert due to the collision, and second, this state triggers the movement towards the center Which is now defined by the invisible diamond. "alert" acts as an intermediate attribute here. This decomposition helps in programming by providing clear conditions and states that can be easily monitored and triggered. The original sentence was vague about how the rabbit should move towards the center after colliding with any edge. By specifying that the rabbit moves towards a predefined point, the revised sentence eliminates ambiguity about the direction and endpoint of the movement. This transformation is beneficial for programming where precise and unambiguous instructions are crucial for consistent and error-free execution.

IntentGuide not only resolves immediate issues but also provides focused educational feedback that helps users enhance their natural language input. As GPT-4 processes the original sentences and the subsequent iterations, it inherently considers the specific context and nuances of each user's input. This means that the system's responses are naturally tailored to address the unique needs and errors present in each case. Students are more likely to be engaged in personalized interactions as they see direct improvements and relevance to their own work.This can inspire them to focus more intently and exert more effort.

III. Evaluation and Discussion

In this section, we evaluate the performance of the proposed IntentGuide with a dataset of 1,000 sentences, similar to the dataset in previous research, utilizing identical metrics for evaluation [10]. In contrast to the earlier approach, Intentguide uses a neuro-symbolic approach improving both the correction of errors and the instructive feedback provided to users. This dataset includes sentences that fit into five categories that are comparable to those found in earlier research:

1) Grammar/Typos: Phrases that contain faults in grammar or spelling that could cause misunderstandings or incorrect code translations.
2) Ambiguous sentences: phrases with ambiguous references or meanings, including pronoun references.
3) Unrealizable actions: Sentences describing actions that are impractical to translate into computer logic.
4) Excessively Complex/Descriptive: Phrases that are difficult to convert into executable code because they contain too many details or complicated structures.
5) Non-problematic Sentences: Sentences that are already clear and straightforward, serving to check whether the system introduces errors into already correct sentences. These sentences are successfully translated into a functional game by GameChangineer platform.

The fifth category, "Non-problematic Sentences," was added primarily as a benchmark, giving a standard by which to evaluate how effectively the AI system handles well-formed inputs and whether the AI intervention may unintentionally produce errors. The evaluation involved processing these sentences through our IntentGuide system, which integrates error detection using symbolic AI and error correction and instruction using a LLM. The rewritten sentences were tested on the GameChangineer platform, which offered an automated compatibility score for each sentence based on its adherence to expected input forms [7] [8]. Using the same dataset, the efficacy of IntentGuide was analyzed. The primary metrics for evaluation were:

1) Success Rate: The percentage of sentences within each category that were correctly rewritten to meet the platform's standards for executable code. A sentence was considered successfully transformed if it achieved an accuracy rate of 90% or higher on GameChangineer.
2) Semantic Integrity: This was assessed manually to ensure that the rewritten sentences maintained the original context and intended meaning of the inputs.

Our results, which are compiled in Table I, are compared with those of the earlier research, in which no error messages from the Symbolic engine were provided to the LLM. The success rates in each of the category of previous study serve as a benchmark for evaluating IntentGuide's effectiveness. According to the comparative evaluation, IntentGuide has significantly outperformed the prior methodology in every category. The "Grammar/Typos" and "Ambiguous" categories have significantly higher success rates, which is indicative of improved error detection and correction abilities. IntentGuide's sophisticated ability to reframe and simplify complex instructions into executable game logic is demonstrated by the rise in the "Overly Complex/Descriptive" category. The higher success rate in the "Non-problematic" category (100% vs. 94%) is particularly significant since it highlights how precisely IntentGuide handles inputs that are already accurate

TABLE I
SENTENCE CATEGORIZATION RESULTS

Category	Sentence proportion	Previous work Success Rate	IntentGuide Success Rate
Grammar/Typing mistakes	22%	75.6%	98%
Ambiguous Sentences	18%	57.7%	84%
Unrealizable Actions	11.6%	50%	69.52%
Overly Complex/Descriptive	28.4%	59%	89%
Non-problematic sentences	20%	94%	100%
Total Success Rate	100%	68.4%	90.02%

without introducing additional errors—a crucial component for user confidence and system dependability. The contribution of symbolic AI was significant in identifying grammatical and logical errors, while GPT-4 excelled in refining semantic content and enhancing sentence clarity.

To provide a detailed illustration of how IntentGuide transforms user input, consider the following example. The original sentence highlights the complexities of translating free-form natural language into executable code. Below, we outline the system's error detection and rewriting process, illustrating how IntentGuide enhances user prompts for clarity and precision.
Original Input Sentence: *"The cheese reappears in 3 seconds when the cheese is silver"*
Phase 1: Error Messages Generated by the Symbolic AI:

- Standard Form: The sentence is not written in standard form (antecedent precede the consequent). Example: "When the rabbit touches a diamond, it inserts a carrot."
- Action Context: The word "reappears" is interpreted as "insert," meaning a character is needed to insert another.
- Unexplained State Change: The condition "cheese is silver" isn't defined by any prior description in the game.
- Time Association: Time should not be associated with insertion directly. Instead, it should be linked to states or attributes.
- Multiple Objects Issue: The number "3" is out of place, as it implies multiple insertions over time.
- Redundant Action: A cheese shouldn't insert another cheese since they're the same type.

Phase 2: The first iteration of rewriting led to the following output: *"There is a cheese. If the cheese is silver, it reappears in 3 seconds."*
However, this version still contained the same errors as the original sentence. The problems with associating time and action remain, and the mechanism for the cheese becoming silver is unexplained. IntentGuide proceeds with further analysis to rectify these errors:
Introducing an Intermediate Attribute: This step ensures that the cheese goes through an intermediate state, adding precision to the timing mechanism.
Defining Duration and Action: By specifying the state and duration of the cheese's intermediate attribute, the rewritten sentences become clearer.

Final transformed sequence of sentences: *"When the cheese touches the alien, the cheese becomes silver. When the cheese is silver, the alien becomes happy for 3 seconds. When the alien is happy, the alien inserts a cheese."*

The errors identified in the original sentence have been used to help rewrite the sentence. This example also demonstrates how IntentGuide refines complex ambiguous inputs into a sequence of simpler, executable statements. The error analysis and transformation strategies ensure that the rewritten sentences are clearer, logically consistent, and properly formatted, aligning with the intended game logic.

IV. CONCLUSION

The neuro-symbolic methodology of IntentGuide facilitates a rewriting system that detects errors while simultaneously providing extensive feedback. IntentGuide demonstrates a novel integration of neuro-symbolic framework for natural language processing, specifically designed for programming and educational technology. This system converts ambiguous, free-form natural language instructions into unambiguous, executable commands appropriate for code generation with more than 90% accuracy.

REFERENCES

[1] S. Harnad, "The symbol grounding problem," *Physica D: Nonlinear Phenomena*, vol. 42, no. 1, pp. 335–346, 1990. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0167278990900876

[2] D. Jurafsky and J. H. Martin, *Speech and Language Processing*, 2nd ed. Prentice Hall, 2009.

[3] T. R. Besold et al., "Neural-symbolic learning and reasoning: A survey and interpretation," *ArXiv*, vol. abs/1711.03902, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:1755720

[4] A. S. d. Garcez, M. Gori, L. Lamb, L. Serafini, M. Spranger, and S. Tran, "Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning," *FLAP*, vol. 6, pp. 611–632, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:155092677

[5] G. Marcus, "The Next Decade in AI: Four Steps Towards Robust Artificial Intelligence," *arXiv e-prints*, p. arXiv:2002.06177, 02 2020.

[6] E. Davis and G. Marcus, "Commonsense reasoning and commonsense knowledge in artificial intelligence," *Commun. ACM*, vol. 58, no. 9, p. 92–103, 08 2015. [Online]. Available: https://doi.org/10.1145/2701413

[7] M. S. Hsiao, "Automated program synthesis from object-oriented natural language for computer games," in *Proceedings of the Controlled Natural Language Conference*, August 2018.

[8] ——, "Multi-phase context vectors for generating feedback for natural-language based programming," in *Controlled Natural Language*, September 2021.

[9] ——, "Automated program synthesis from natural language for domain specific computing applications," Patent 10 843 080, November, 2020.

[10] N. K. Yeole and M. S. Hsiao, "Bridging natural language and code by transforming free-form sentences into sequence of unambiguous sentences with large language model," in *conference*. IARIA, 2024, pp. 4–10, retrieved: September, 2024.