Perceptual Interpreter: A Human-Centric Metric for Code Readability

Tairan Wang
Supervisor: Prof. Earl T. Barr

February 25, 2025

Internship Proposal
June 2025 - August 2025

Abstract

In software engineering, code readability has long lacked a reliable, cognitively aligned quantitative metric. This research proposes a Perceptual Interpreter based on human short-term memory and abstraction capabilities. The interpreter simulates human code-reading behavior by maintaining a short-term memory model and ultimately providing a readability score. The simulation includes temporally and spatially ordered short-term memory, semantic-based code abstraction, and semantic annotation analysis. Moreover, Large Language Models (LLMs) have demonstrated strong performance in various software engineering tasks, including code generation, comprehension, bug fixing, and test generation. This research leverages LLMs' ability to infer code semantics for abstraction and analysis. Ultimately, we aim to develop a perceptual interpreter aligned with human expert cognition to assess code readability.

Relative Works and Background

Readable code has better locality and performs better.

Aims and Objectives

The early goal of this research is to rapidly develop an interpreter capable of recognizing and parsing explanatory content such as comments. It should include a basic short-term memory simulation and a readability scoring function.

The second phase aims to implement semantic similarity-based memory abstraction and comment analysis to further enhance the interpreter's capabilities.

The third phase focuses on refining the interpreter based on collected human expert feedback. This phase will gradually introduce additional parameters such as human long-term memory, forgetting, and recall mechanisms.

If time permits, after collecting human expert feedback, this research will attempt to train a neural network to weight LLM layer features based on readability scores. Under this weighting, the aggregated LLM layer features will form a new code readability metric, analogous to LPIPS in image similarity assessment.

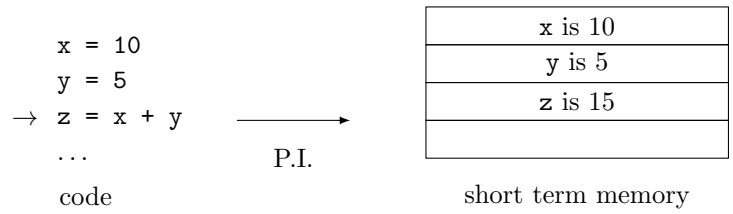
The minimum expected outcomes of this project are:

- A Perceptual Interpreter for a specific programming language, along with its dataset and experimental results, demonstrating its ability to accurately assess the readability of programs written in that language.
- A method for constructing or training a Perceptual Interpreter that can be generally applied to any programming language.
- A paper submitted to a top-ranked software engineering or machine learning conference.

Work Plan

[**Week 1-2**] Develop an interpreter capable of recognizing and parsing explanatory content such as comments. This interpreter should include a basic short-term memory simulation and a readability scoring function. Additionally, develop a refactoring mechanism that can controllably disrupt code structure without affecting program correctness or side effects, thereby altering readability.

At this stage, short-term memory will allocate a slot when encountering a new variable and release it when the variable goes out of scope or its left-time expires.



[**Week 3-4**] Implement memory abstraction based on semantic similarity. Unlike traditional interpreters that retain all environment variables, humans have an abstraction capability. For example, humans abstract functionally coherent code into functions and understand the code by remembering only the function's purpose rather than its implementation. Similarly, when reading logically coherent code, humans automatically abstract its content, such as "this code performs boundary checks and raises an error" or "this code retrieves user input and validates it."

When similar content exists in short-term memory, humans tend to merge abstractions. Therefore, the Perceptual Interpreter should periodically check for similar memory entries and merge them accordingly. This semantic similarity check can be performed using the embedding space of an LLM or by directly querying it with appropriate prompts.

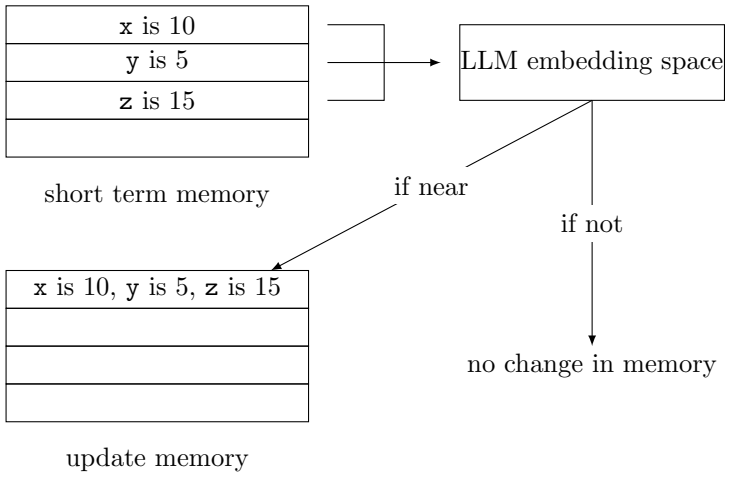


Figure 1: Demo

[**Week 5-6**]

[**Week 7**]

[**Week 8**]