An Approach for Human Error Root Cause Analysis based on Large Language Models

Theo Hytopoulos *hytopot@wwu.edu* · Tan Nguyen *nguye495@wwu.edu* · Kiley Schutte *schuttk2@wwu.edu* · Matthew Enertson *enertsm@wwu.edu* · Fuqun Huang *huangf2@wwu.edu*

WESTERN WASHINGTON UNIVERSITY

Abstract

Software defects often stem from systematic cognitive errors made by developers. In this work, we present a novel approach to detecting human error modes (HEMs) in source code using Large Language Models (LLMs). Our model leverages the Codesage LLM to perform multi-label classification, enabling it to detect multiple error types, such as rule encoding deficiencies and post-completion errors, within a single file. The model is trained using parameter-efficient fine-tuning with Low-Rank Adaptation (LoRA), and class-specific binary cross-entropy loss addresses class imbalance in the dataset. Our dataset includes student code submissions with real-world instances of human error, labeled according to a taxonomy grounded in cognitive psychology. Preliminary results indicate promising precision and recall in identifying error modes, with performance improving in models that support extended context. This work contributes to the growing effort to integrate human error theory into automated defect detection and provides a scalable foundation for building cognitively informed code analysis tools.

Results

- Achieved **44% micro average precision**, which indicates overall performance
- Achieved **46% macro average precision**, which indicates the ability to perform well on multiple error modes
- The results show that the model can learn cognitive patterns shared across different programmers

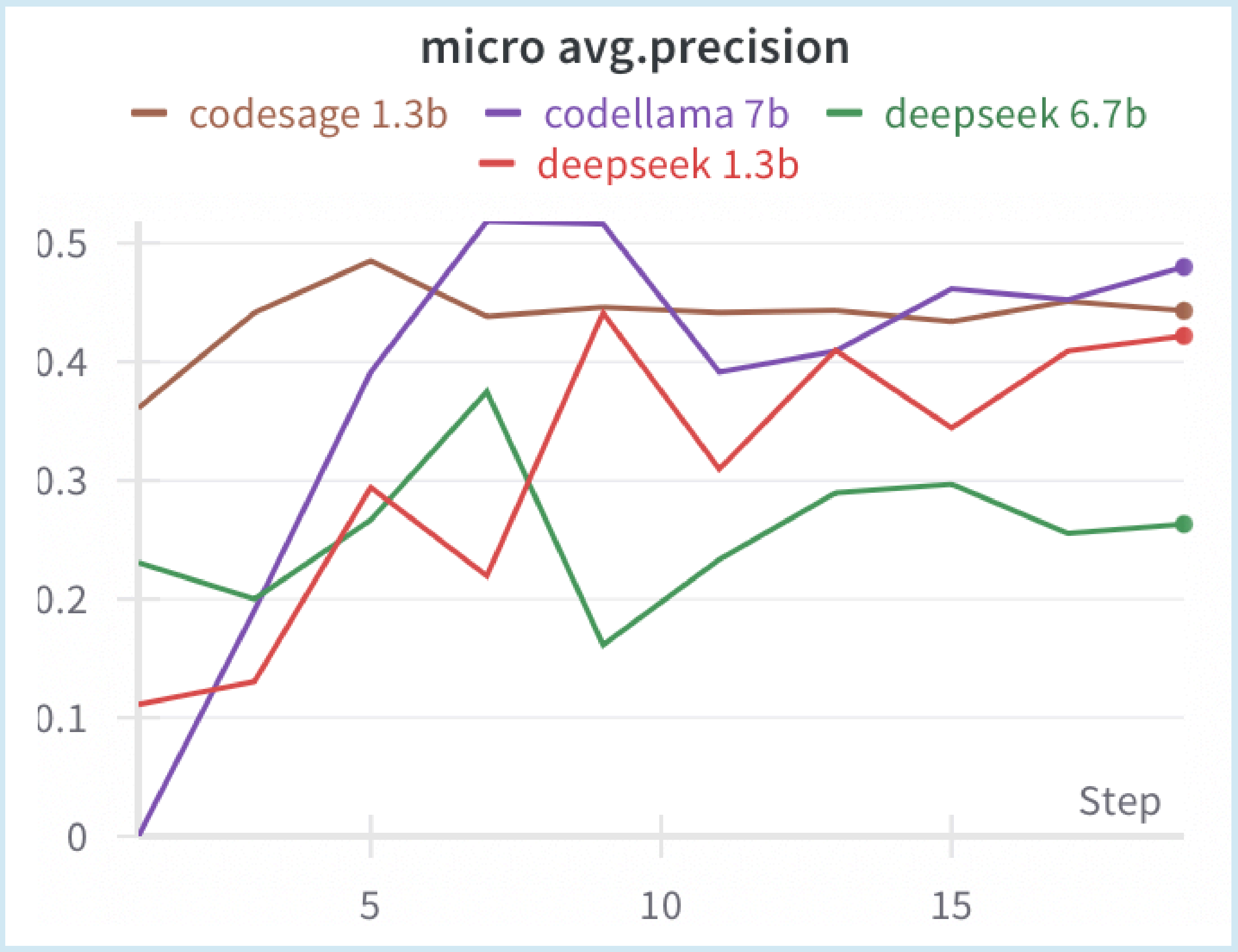

Figure 3: Comparison of the micro average precision of our model (codesage) to other popular models for error detection

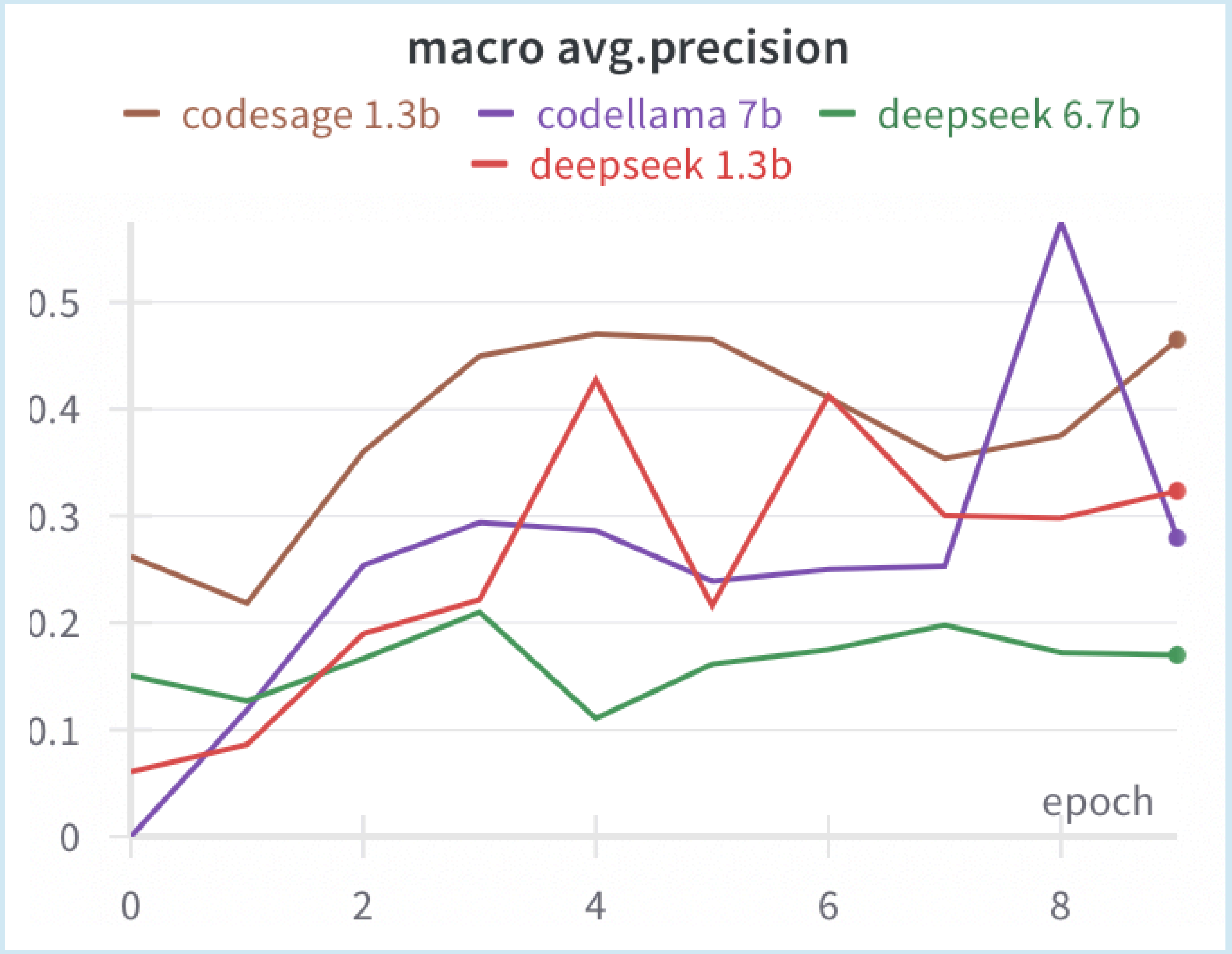

Figure 4: Comparison of the macro average precision of our model (codesage) to other popular models for error detection

Methodology

Our approach utilizes the Codesage LLM because of the model's large context window and parameter count, which can capture a broader range of contextual information necessary for detecting human error modes. The model is trained on a dataset containing labeled error-prone scenarios. Each error mode is assigned a binary classifier to support multilabel classification. Training is conducted using Low-Rank Adaptation (LoRA) for parameter-efficient fine-tuning, and quantization is applied to optimize performance for consumer-grade hardware. The model's effectiveness is validated against a held-out dataset, and performance is benchmarked against existing defect detection architectures.

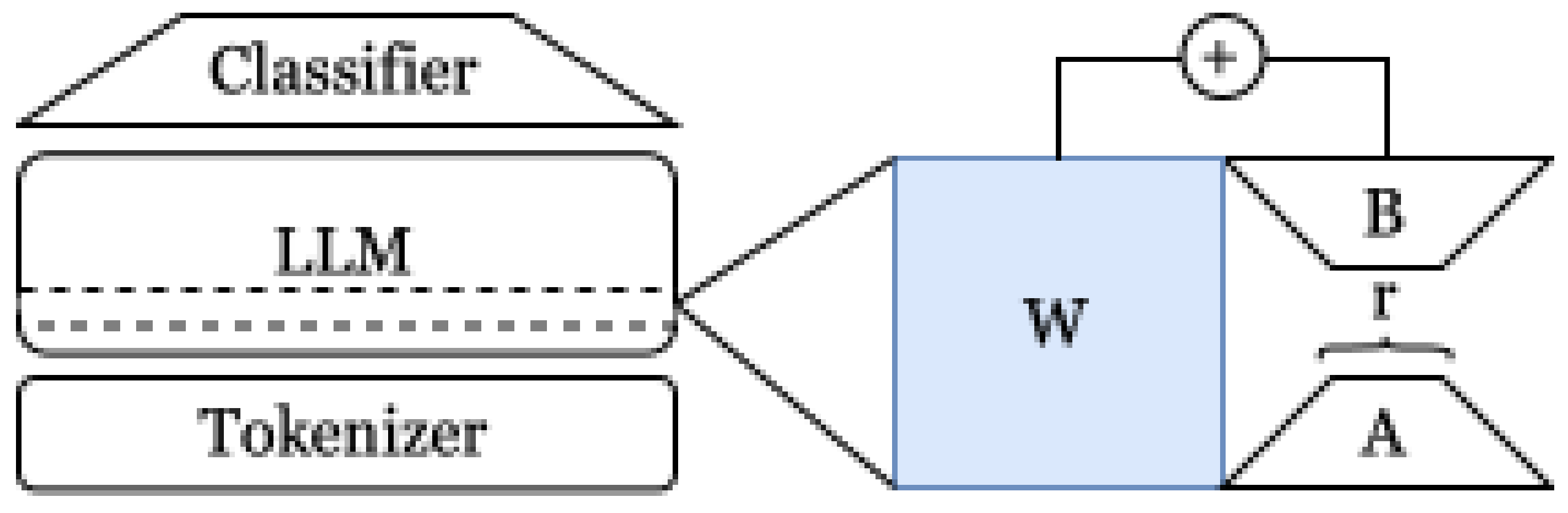

Figure 1: The model architecture with LoRA applied to each layer.

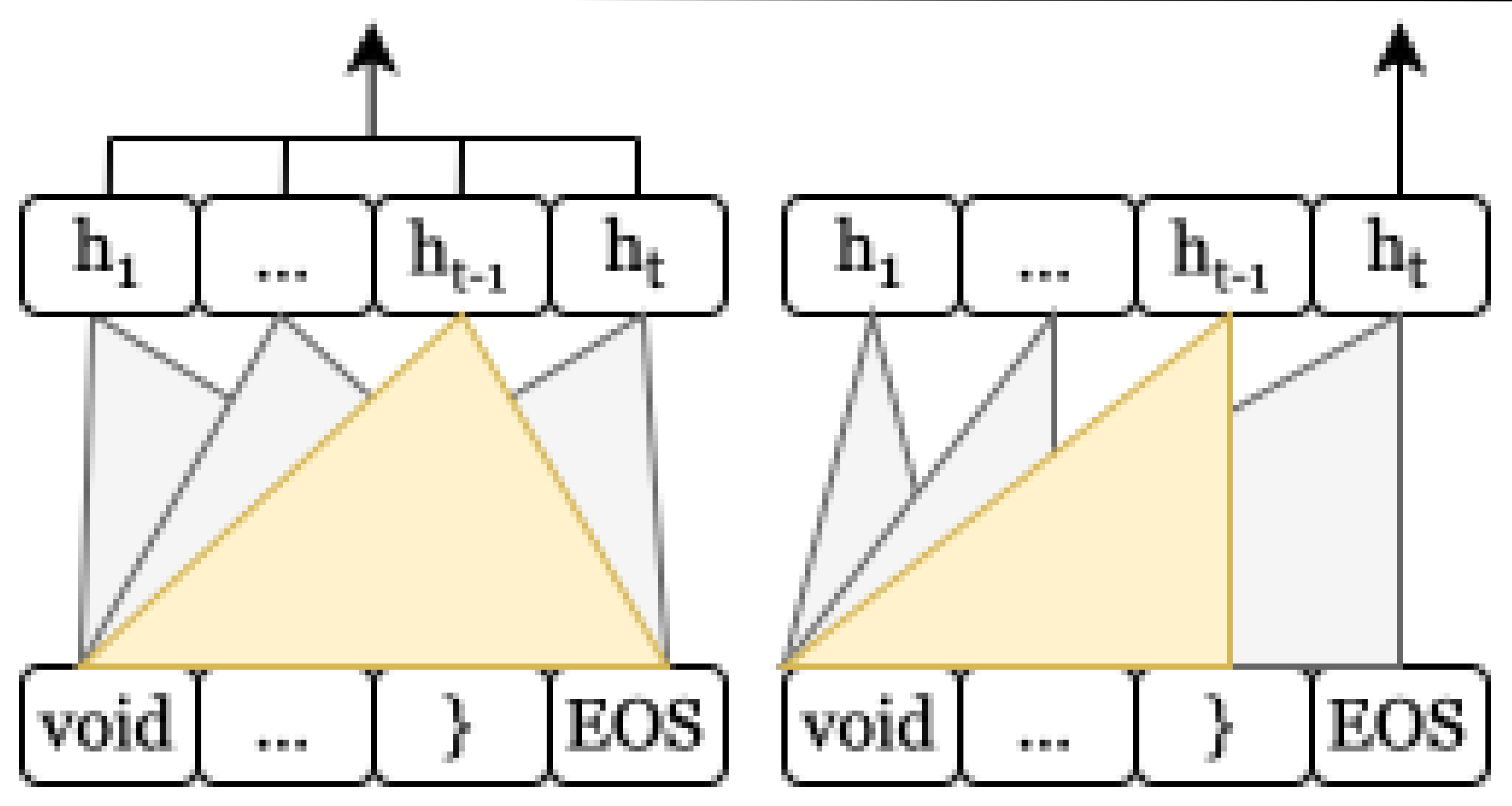

Figure 2: Shows the attention of the model. The left (ours) is bidirectional attention, meaning each token can "see" the entire sequence. The right is masked attention, meaning each token can only "see" the ones before it.

Key Findings

- Larger context windows of contemporary LLMs enable detection of multiple defects within a single code file, outperforming models with shorter context lengths.
- Multi-label classification improves real-world applicability by capturing interrelated human errors that co-occur in software defects.
- Preliminary evaluations show that our approach achieves promising results on multi-label defect detection, demonstrating higher precision and recall than the baseline.
- Further improvements include refining error mode categorization, testing alternative model architectures, and expanding the dataset to cover a wider range of human error scenarios.

Definitions

- **Defect**: An incorrect or misstep, process, or data definition in a computer program.
- **Human Error**: an erroneous human behavior that leads to a software defect.
- **Human Error Mode**: A particular pattern of erroneous behavior that recurs across different activities, due to the cognitive weakness shared by all humans.
- **Error-Prone Scenarios**: A set of conditions under which a HEM tends to occur.
- **Precision**: The quality of a positive prediction made by the model. Calculated as the number of true positives divided by the total number of positive predictions
- **Micro Avg. Precision**: A measure of overall prediction quality that accounts for all errors equally, regardless of their type. It calculates precision across all instances as if they belong to a single pool, giving more weight to common error types.
- **Macro Avg. Precision**: A measure of how well the model performs across each individual error type. It calculates precision for each error type separately, then averages them equally, highlighting the model's ability to detect rare or less frequent errors.