Learning to Detect Vulnerable Code Using Differentiable Line Probability of Large Code Models

Minjae Gwon and Sangdon Park.
Research Project II, Dept. of Computer Science and Engineering, POSTECH, Korea.

Motivation

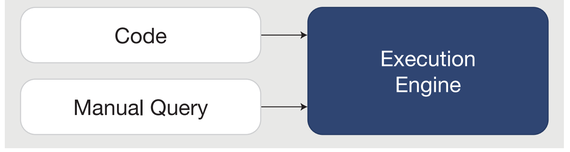
Problem Description

Mission. Automate vulnerability detection to reduce manual effort.

- ***Automation.*** Reduce manual efforts in detection.
- ***Efficiency.*** Quickly identify vulnerabilities in code.
- ***Accuracy.*** Minimize false positives and false negatives.
- ***Scalability.*** Easily intergrated with multiple languages.

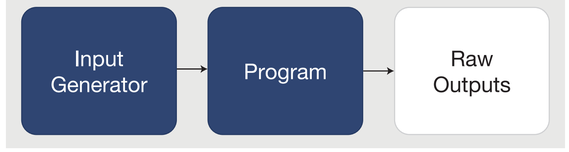
Related Works

<u>1. Static Analysis</u>



- Requires significant manual effort for rule definition.

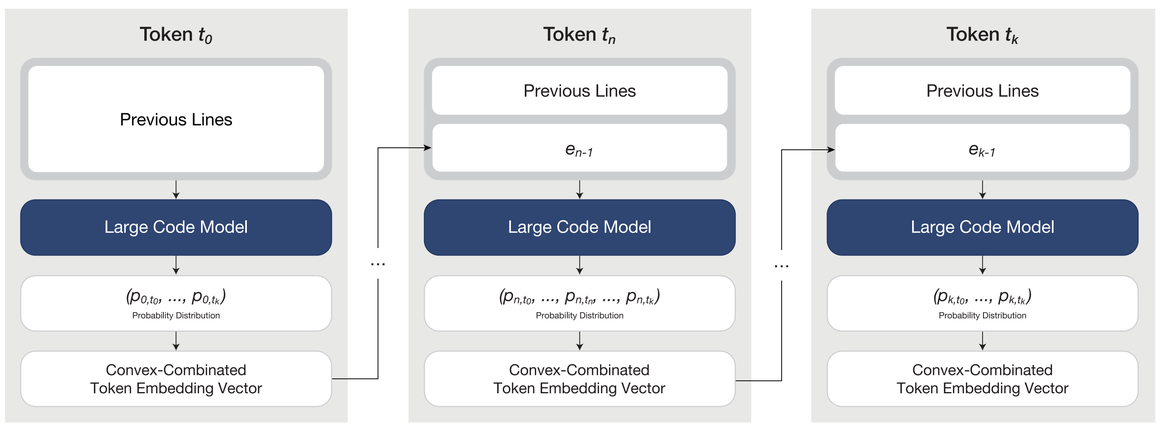
<u>2. Fuzzing</u>



- Requires lots of time and manual effort for defining inputs and analyzing outputs.

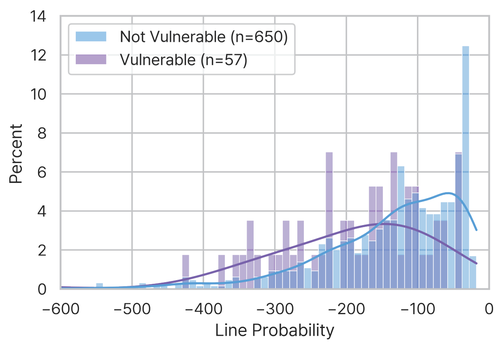
Method

Line Probability

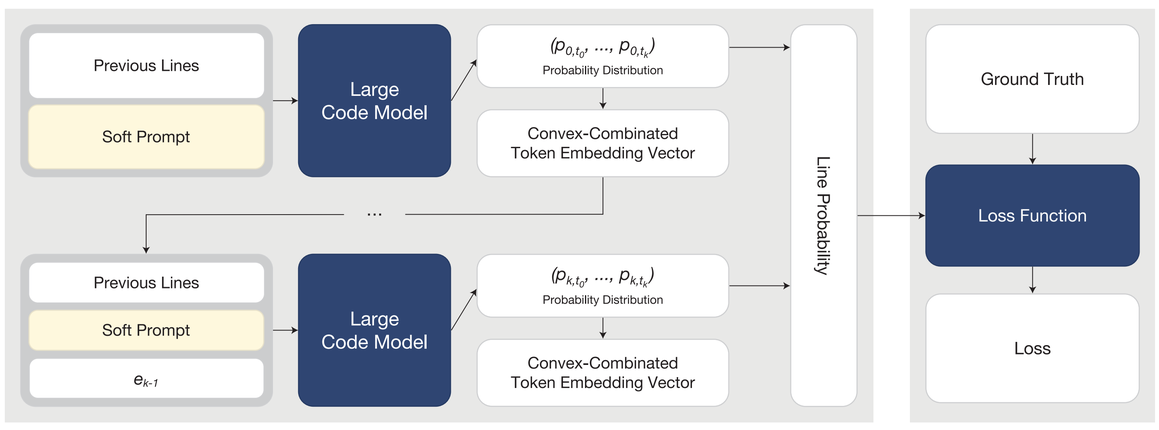


Why? Empirically useful in discriminating vulnerable lines from benign lines.
Objective 1. Must quantify vulnerability likelihood at the line level.
Objective 2. Must be differentiable for accurate training.

- *Line probability* is the sum of token probabilities which indicates the vulnerability of a code line.
- *Token probability* is derived from a distribution of large code model outputs, where the input is a convex combination of prior distributions and some previous code lines.

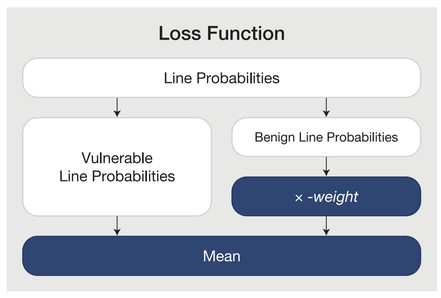


Prompt Tuning



Why? Use a small number of parameters to effectively tune the code model's performance.
Objective 1. Must optimize the performance of line probability.
Objective 2. Must be tuned in terms of line probabilities.

- *Prompt tuning* is a mechanism for learning *soft prompts*, enabling models to perform specific down-stream tasks.[1]
- *Large Code Model*, the tuned version of large language models, detects the possibility of vulnerability by line probabilities.



Result

Model and Dataset

Base Model	Code Llama 7B
Vulnerable Lines	336
Benign Lines	3,207
Validation Split	0.2

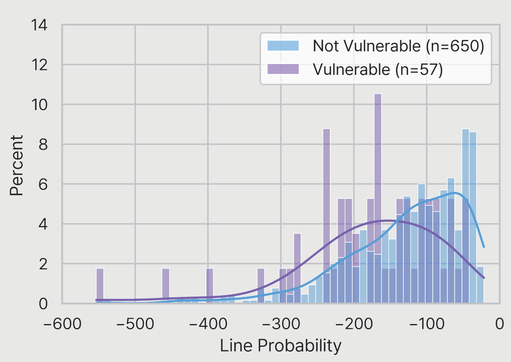
Parameters

Soft Prompt Length	64
Epochs	8
Batch Size	16
Learning Rate	0.0005

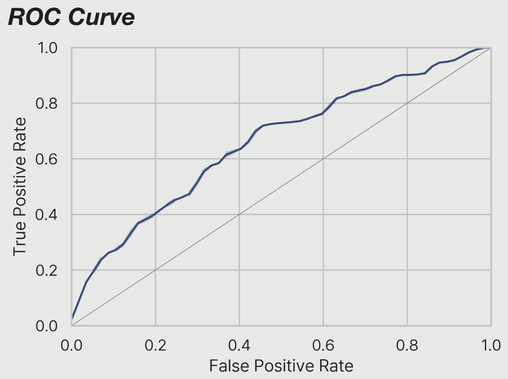
Analysis

Observation 1. *Line probability* emergence as a potent metric for vulnerability detection.
Observation 2. *Prompt tuning* makes the model more sensitive to vulnerable lines.

Distribution

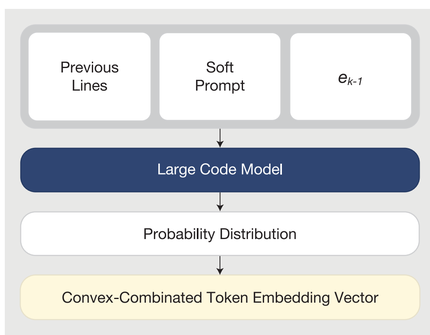


ROC Curve



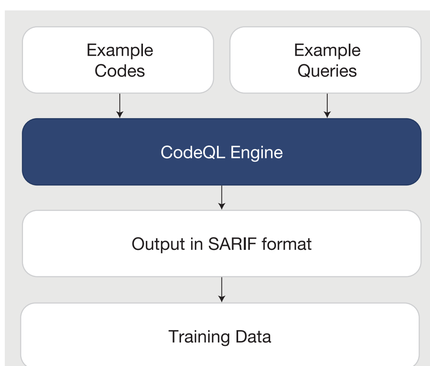
Discussion

<u>1. High Cost of Calculating Line Probabilities</u>



- Convex-combinated token embedding vector is constructed from all-available tokens of a base model.
- A line probablity should be calculated sequentially.
- These two reasons make the calculation very slow.
- Exploring ways to calculate line probabilities in parallel could be beneficial.

<u>2. Lack of Data for Training and Testing</u>



- Our dataset is constructed from CodeQL's public repository and sampled DARPA's challenges.
- Some data did not fit well during tuning.
- The size of dataset was insufficient for tuning.
- Explore self-supervised learning methods that do not require labeled data but need to design auxiliary tasks useful for vulnerability findings.

[1] B. Lester, R. Al-Rfou, N. Constant, "The Power of Scale for Parameter-Efficient Prompt Tuning," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021, pp. 3045–3059.