SimSCOOD: Systematic Analysis of Out-of-Distribution Generalization in Fine-tuned Source Code Models

Hossein Hajipour[1], Ning Yu[2], Cristian-Alexandru Staicu[1], Mario Fritz[1]

[1]CISPA Helmholtz Center for Information Security, [2]Netflix Eyeline Studios

Introduction

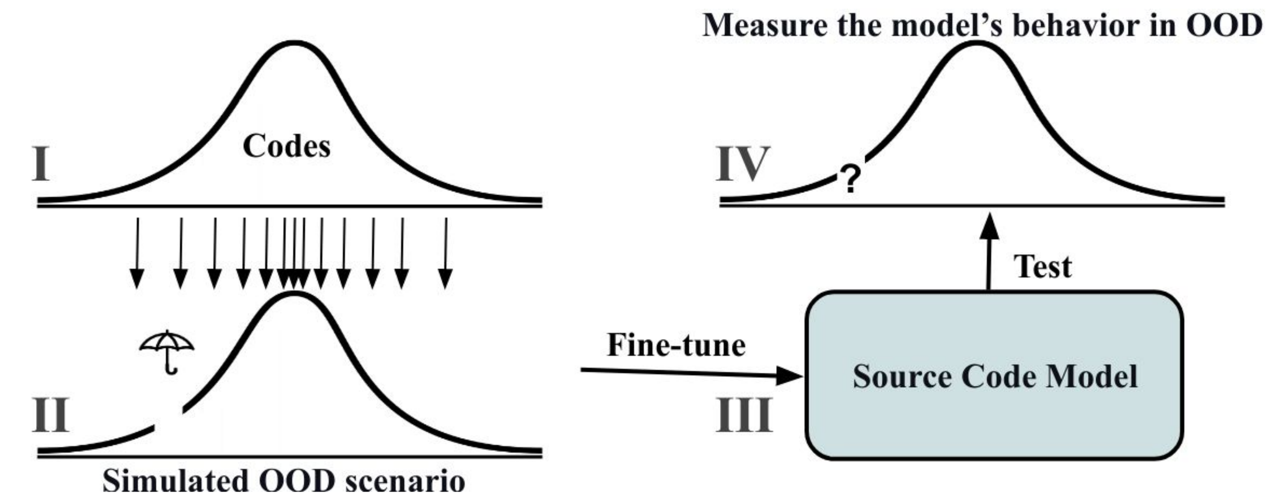
- **Generative models** suffer from finite size of data.
- Programs have **complex compositional nature**.
    - Given a complex enough grammar we can have infinite number of potential programs.
- Despite having access to the large code datasets to pre-train these models, it remains challenging in practice to **fully cover the code distribution**, specifically in **fine-tuning datasets**.
- It is crucial to analyze the behavior of these models in different scenarios
    - Beyond the traditional train/test splits.

Contributions

- We propose a **systematic approach** to simulate various **OOD scenarios** by **masking out sub-regions** of source code distribution along the **length**, **syntax**, and **semantics** dimensions.
- We find that the performance of the fine-tuned models can significantly deteriorate in various **OOD scenarios** despite the model **encountering similar examples** during the **pre-training phase**.
- Our systematic analysis shows that, while **full fine-tuning** and **LoRA fine-tuning** perform comparably on **in-distribution** code data, **LoRA fine-tuning** demonstrates significantly **better performance** on **OOD data.**

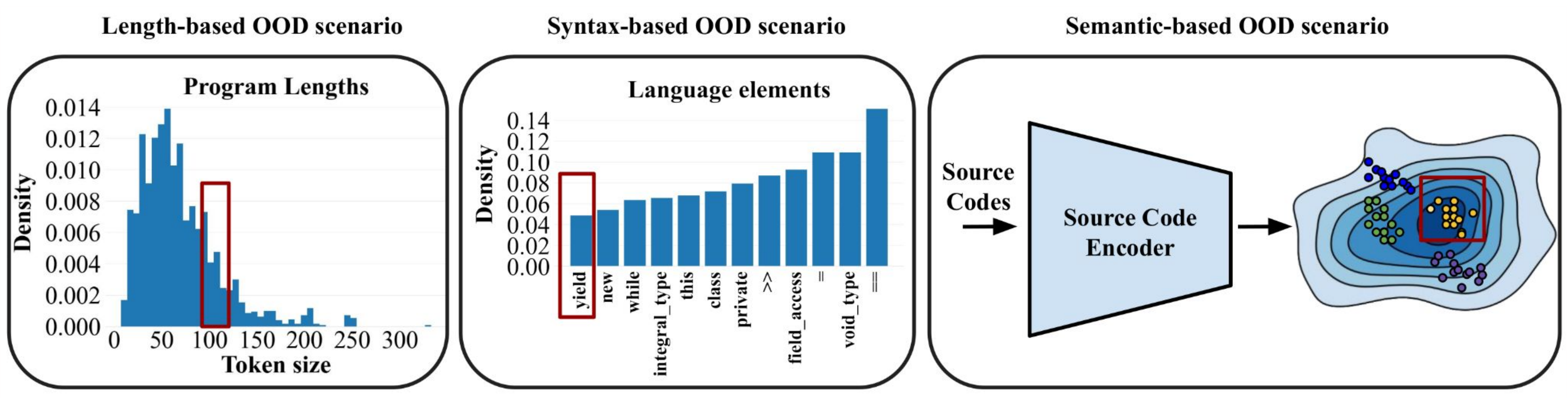
SimSCOOD: Simulation of Source Code Out-of-Distribution Scenarios

Overview of SimSCOOD



I. Original code distribution along a dimension.
II. OOD simulation by masking out a sub-region.
III. Model fine-tuning.
IV. Evaluation on OOD data.

Out-of-distribution scenarios



- To simulate **length-based** scenarios, we use the histogram of program token sizes to represent the distribution of a given dataset.
- We use the **histogram of language elements** to model the **syntax** distribution of a given source code dataset.
- We employ a **pretrained model** to cluster programs within the continuous space and simulate the **semantic-based scenarios**.

Experiments

Setup

Tasks:
- Text-to-Code
- Code refinement

Models:
- GraphCodeBERT [1]
- CodeT5 [2], CodeT5+ [3]
- Code Llama [4]

Fine-tuning:
- Full fine-tuning
- LoRA fine-tuning [5]

How Do Fine-tuned Models Generalize?

- Overall results of the model performance for different scenarios in **text-to-code** task.
    - The results provide the **relative exact match to the 100%** baseline for different **OOD** and **few-data** regime scenarios.
    - **FT** denotes **full fine-tuning**, and **LoRA** refers to the **LoRA fine-tuning** method.

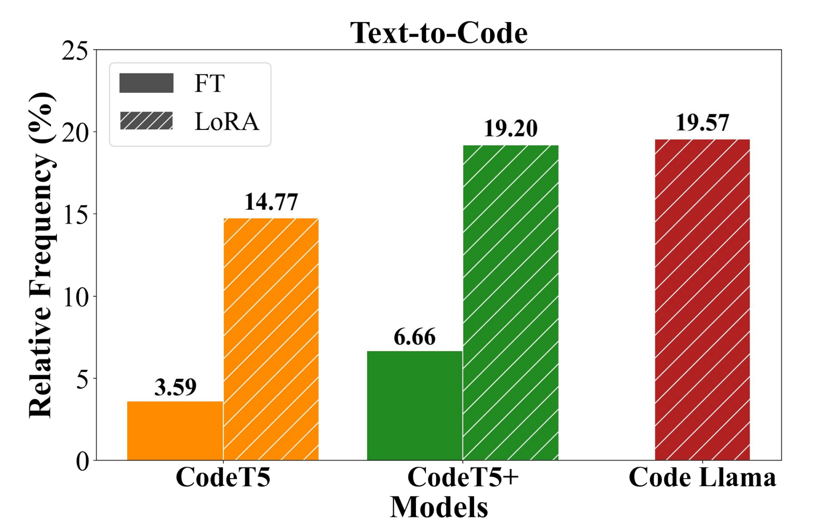
Models		Length Inter		Length Extra		Syntax		Semantic	
		FT	LoRA	FT	LoRA	FT	LoRA	FT	LoRA
CodeT5	OOD	53.92%	66.91%	0.00%	24.99%	16.46%	34.81%	31.90%	51.42%
	Few	86.56%	103.79%	28.56%	55.0%	93.90%	100.0%	37.56%	72.43%
CodeT5+	OOD	49.65%	70.94%	5.0%	26.09%	47.95%	68.97%	39.69%	55.71%
	Few	76.40%	96.36%	77.38%	101.72%	67.21%	78.54%	66.04%	83.68%
Code Llama	OOD	-	71.75%	-	23.57%	-	64.81%	-	56.72%
	Few	-	94.08%	-	63.21%	-	86.08%	-	84.74%

How fine-tuned models perform on the full dataset?

- **Exact match** results of the fine-tuned models using the full fine-tuning dataset.

Models	Text-to-Code		Refinement	
	FT	LoRA	FT	LoRA
GCBERT	-	-	10.74	11.38
CodeT5	22.15	21.65	14.43	14.53
CodeT5+	24.95	24.70	15.18	15.29
Code Llama	-	27.65	-	19.19

Can fine-tuned LLMs generate unseen language elements?



Key Findings

- Performance of fine-tuned models, can significantly deteriorate in OOD scenarios, even when the models have seen similar code samples during pre-training.
- While full fine-tuning and LoRA fine-tuning methods show comparable results over in-distribution data, LoRA fine-tuning significantly outperforms full fine-tuning in OOD scenarios.
- By incorporating a small amount of relevant data into the fine-tuning set, models can achieve substantial performance enhancements.

CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

EYELINE STUDIOS POWERED BY NETFLIX

References

[1] Guo, Daya, et al. "Graphcodebert: Pre-training code representations with data flow." ICLR 21.
[2] Wang, Yue, et al. "Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation." EMNLP 21.
[3] Wang, Yue, et al. "Codet5+: Open code large language models for code understanding and generation." EMNLP 23.
[4] Roziere, Baptiste, et al. "Code llama: Open foundation models for code." arXiv 23.
[5] Hu, Edward J., et al. "Lora: Low-rank adaptation of large language models." ICLR 22.