Code Representation Pre-trainig with Complements from Program Executions

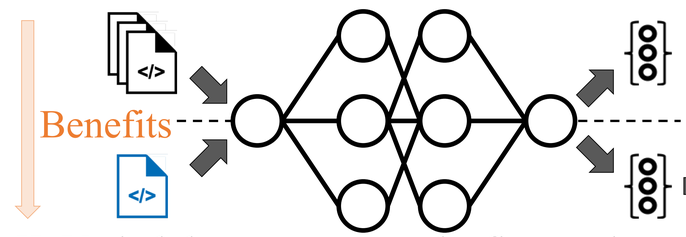
Jiabo Huang[1], Jianyu Zhao[1], Yuyang Rong[2], Yiwen Guo*[3], Yifeng He[2], Hao Chen[2]

[1]Tencent Security Big Data Lab, [2]UC Davis, [3]Indepdnent Researcher

{jiabohuang, yjjyzhao}@tencent.com, {PeterRong96, guoyiwen89}@gmail.com, {yfhe, chen}@ucdavis.edu

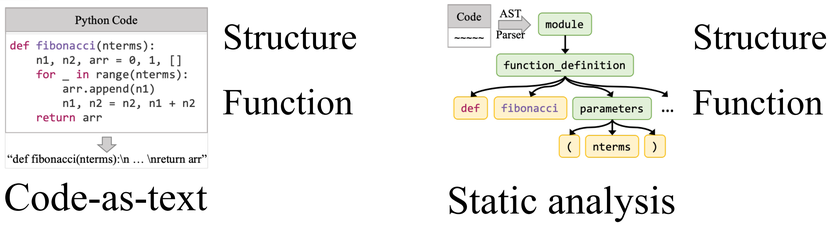
1 Problem Definition

I. *Unsupervised Representation Pre-training*



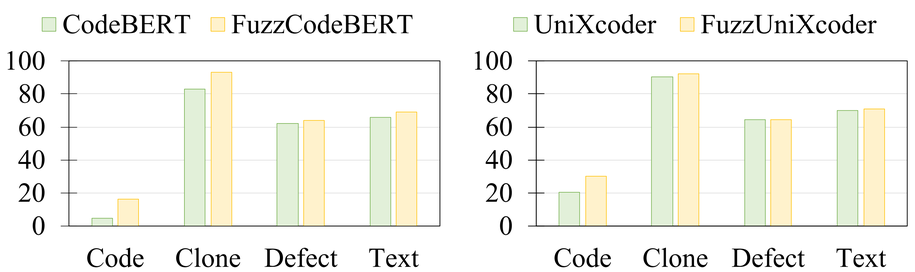
II. Varied down-stream tasks fine-tuning

2 Motivation

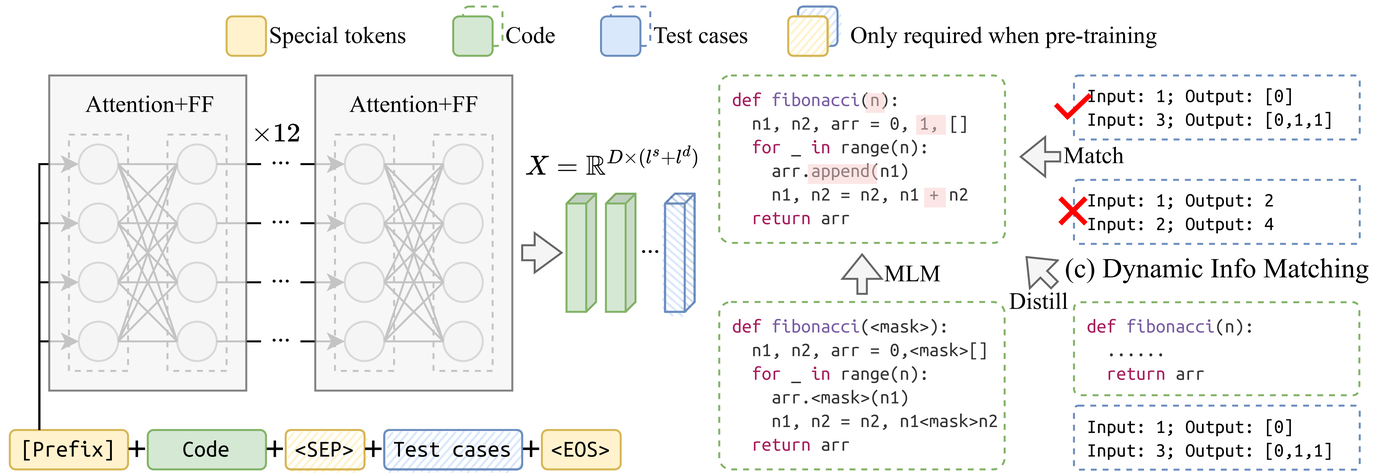


5 Experiments

- FuzzPretrain is beneficial to multiple code understanding tasks
- Dynamic information from test cases complements both code and its syntactic representations (AST)

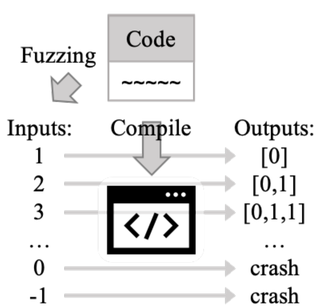


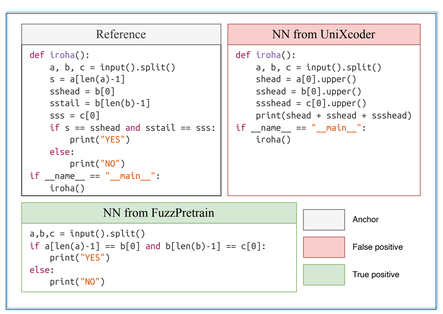
3 Overview of FuzzPretrain



(a) Model architecture (b) Static Info Modeling (d) Dynamic Info Distillation

4 Data collection





5 Static & Dynamic Information Modelling

- Static Information Modelling (SIM)

Masked tokens prediction on code (S)

$$\mathcal{L}_{SIM}(S) = -\sum_{m\in M}\log\left(p\left(m\middle|\tilde{X}^s\right)\right)$$

- Dynamic Information Modelling (DIM)

Matching code (S) with test cases (D)

$$\mathcal{L}_{DIM}(S,D) = \mathrm{BCE}(y, f_\phi(FC(\boldsymbol{x}^h))$$

- Dynamic Information Distillation (DID)

Distilling dynamic info from holistic representation $H = S \oplus D$ *to code (S)*

$$\mathcal{L}_{DID}(S, S\oplus D) = -\log\frac{g(\hat{x}^h, x^s)}{g(\hat{x}^h, x^s) + \sum_{x^-\in X^-} g(x^-, x^s)}$$