Rethinking the Role of Structural Information: How It Enhances Code Representation Learning?

IJCNN 2024

Qiushi Sun[1,2], Nuo Chen[3], Jianing Wang[3], Xiaoli Li[1]

Rethinking the Role of Structural Information: How It Enhances Code Representation Learning?

Qiushi Sun[1,2], Nuo Chen[3], Jianing Wang[3], Xiaoli Li[1]

[1] Institute for Infocomm Research, A*STAR [2] National University of Singapore
[3] East China Normal University

Contact: qiushisun@u.nus.edu

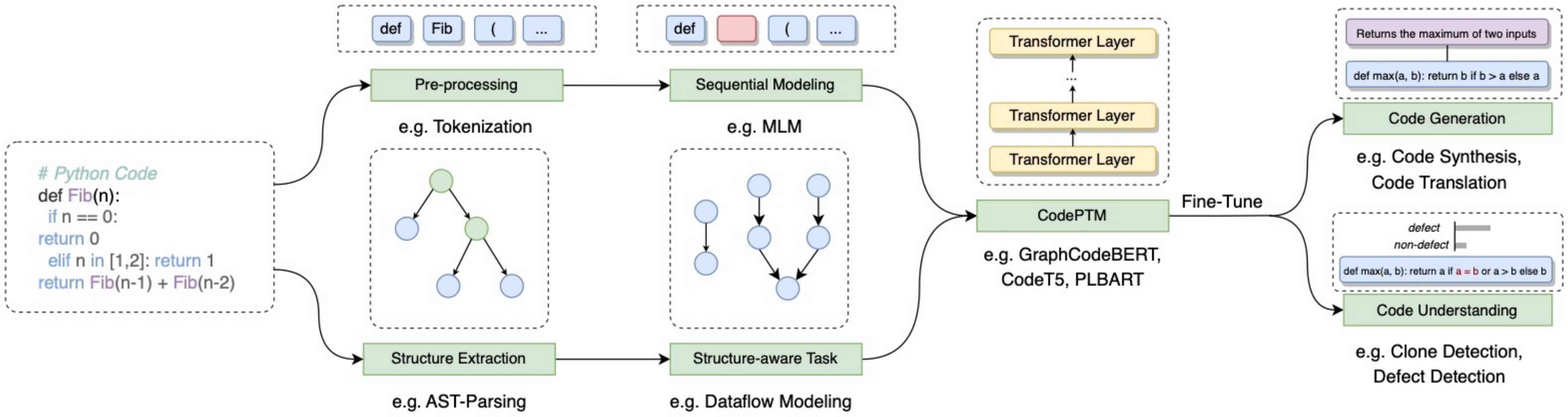
Background & Motivations

Background

- The success of Code Pre-trained Models on downstream tasks
  - Pre-trained language models have advanced the SOTA across NLP tasks.
  - The success of NL applications has led to their adaptation in code.
- Growing interest in leveraging code structures in training:
  - Code Tokens
  - Code Structures
- Recap: How are these CodePTMs built?

Background

- ■ Build & Use Code Pre-trained Models (CodePTMs)
  - ☐ Code snippets are tokenized & ASTs are also parsed
  - ☐ Modeling is conducted for both the code tokens and structure extracted
  - ☐ Will be available for downstream adaptation through fine-tuning



Typical Pipelines of training CodePTMs

CodePTMs

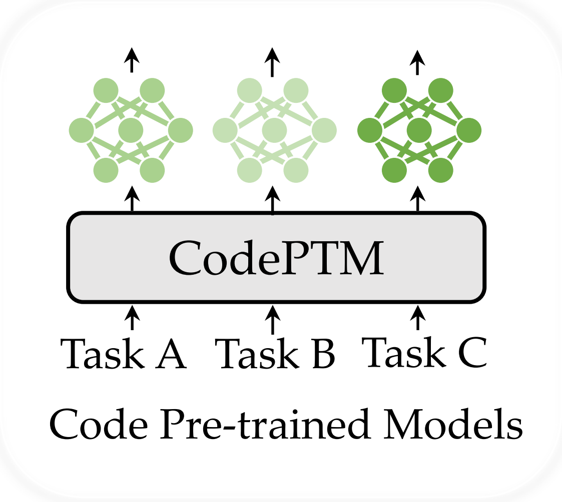
For pre-training, code structure matters!

Models	Inputs	Pre-training Tasks
RoBERTa	Natural Language (NL)	Masked Language Modeling (MLM)
CodeBERT	NL-PL Pairs	MLM+Replaced Token Detection (RTD)
GraphCodeBERT	NL-PL Pairs & AST	MLM+Edge Prediction+Node Alignment
UniXcoder	NL-PL Pairs & Flattened AST	MLM ULM (Unidirectional Language Modeling) Denoising Objective (DNS)

But, how about fine-tuning?

CodePTMs

For pre-training, code structure matters!



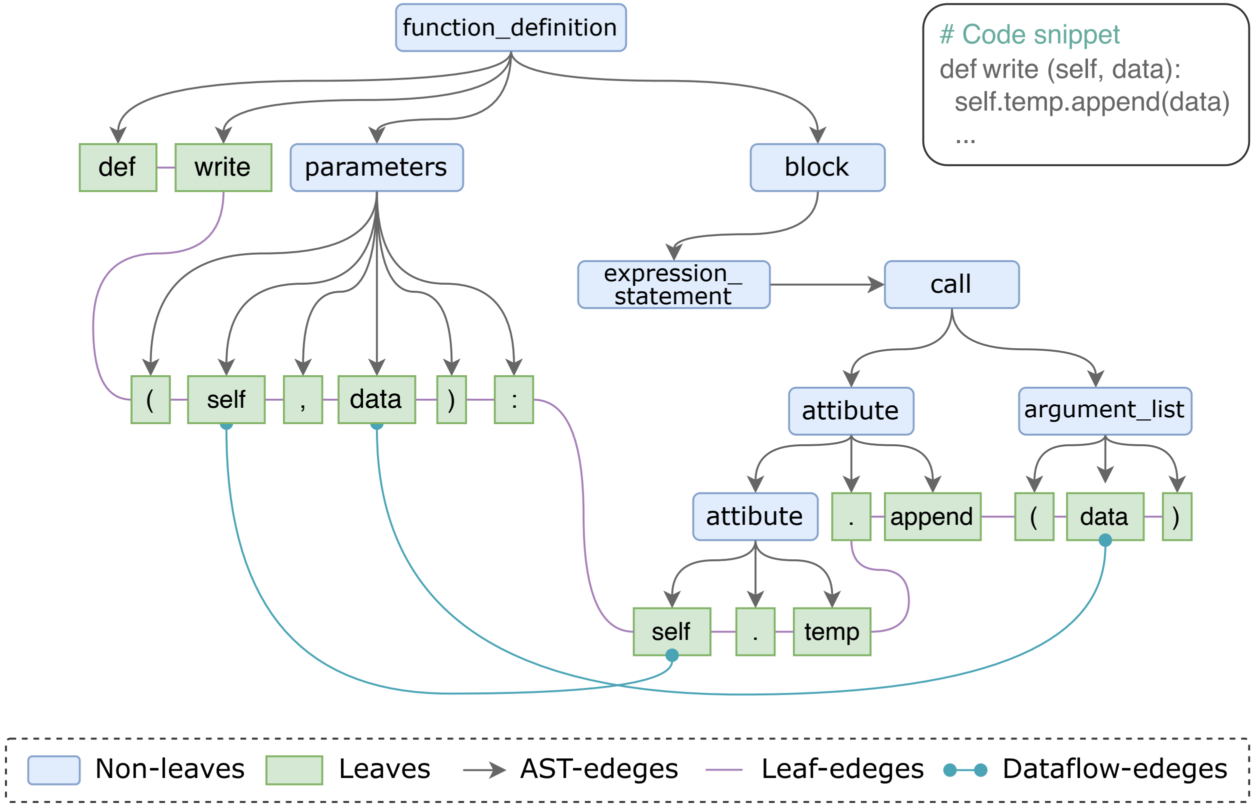
Core Research Question: How does code structural information impact task performance during the fine-tuning? How can it be utilized?

Rethinking the Role of Structural Information

Abstract Syntax Tree

- The most typical form: AST

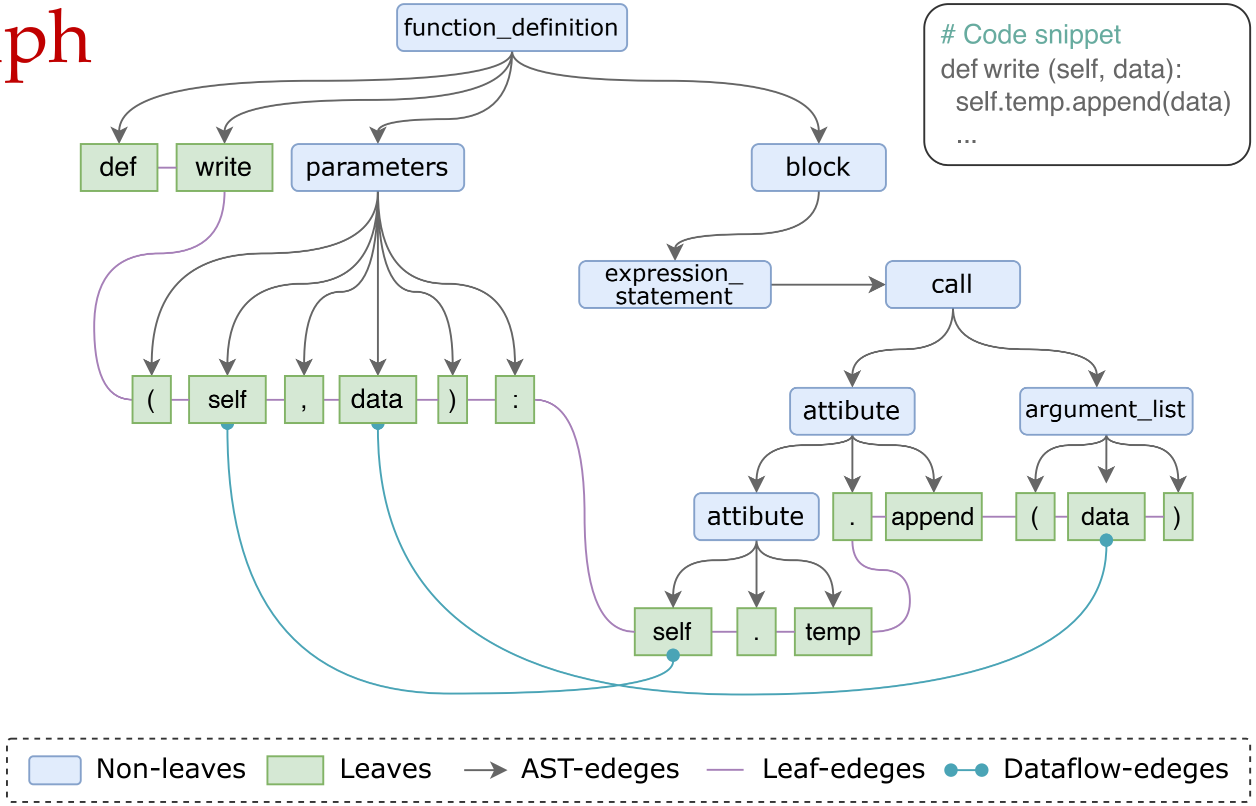
- ☐ Code structure plays an important role in this process.
- ☐ Providing signals beyond code tokens.



Raw AST is not Enough

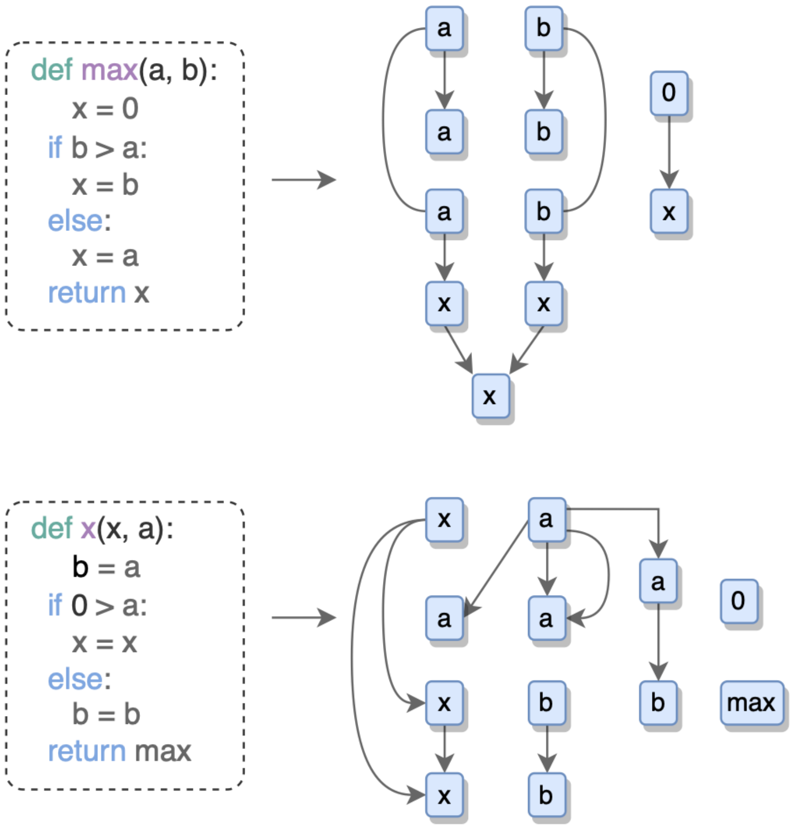
- Transforming AST into a Graph

- ☐ Connected data flow edges.
- ☐ Establish connections between adjacent leaf nodes to bolster the overall connectivity



Code Structural Information

From the perspective of perturbation



- Natural Language
  - Flexible and diverse structure and semantics.
  - Basic meaning and readability often maintained despite perturbations.
- Code
  - Perturbations can alter execution sequence.
  - Potential errors.
  - Can create incorrect dependencies.

Pilot Experiments

Pilot perturbation experiments

- Perturbation caused a significant impact!
- The effect is moderate for classification tasks but substantial for generation tasks.

Methods	Clone	Defect	Code Translation	
	F1	Acc	BLEU	EM
GraphCodeBERT				
Fine-Tuning	95.00	62.88	77.49	59.85
Fine-Tuning (Pert.)	94.75 $_{-0.25}$	61.86 $_{-1.02}$	59.04 $_{-18.45}$	47.15 $_{-12.7}$
PLBART				
Fine-Tuning	93.60	63.16	81.13	63.35
Fine-Tuning (Pert.)	93.99 $_{+0.33}$	62.48 $_{-0.68}$	74.96 $_{-6.17}$	57.85 $_{-5.50}$
CodeT5				
Fine-Tuning	95.00	65.78	81.63	65.85
Fine-Tuning (Pert.)	95.16 $_{+0.41}$	63.03 $_{-2.75}$	77.52 $_{-4.11}$	61.60 $_{-4.25}$
UniXcoder				
Fine-Tuning	91.36	62.34	76.59	63.45
Fine-Tuning (Pert.)	89.77 $_{-1.59}$	60.94 $_{-1.40}$	69.64 $_{-6.95}$	57.80 $_{-5.65}$

Code Structural Information

How can we verify it?

- Check if these samples rich in structural information can replicate the performance of CodePTMs.
- Let's first evaluate the structural information contained in the code.

Algorithm of Exemplar Selection

Algorithm 1 Exemplars Selection Process

Input:
Code snippets $\mathbf{C} = [c_1, c_2, \ldots, c_n]$
Empty exemplar list $\mathbf{E}$
Desired number of exemplars k
Output:
Updated exemplar list $\mathbf{E} = [e_1, e_2, \ldots, e_k]$
procedure SELECTEXEMPLARS($\mathbf{C}$, k)
  for $i = 1$ **to** n **do**
    $T \leftarrow$ ConvertToAST(c_i)
    $G \leftarrow$ AugmentToUAST(T)
    $ge \leftarrow$ ComputeGE(G)
    $l_i \leftarrow 1/ge$ ▷ Compute the inverse of global efficiency
    if length($\mathbf{E}$) < k **then**
      AddToExemplars($\mathbf{E}$, (l_i, c_i)) ▷ Add to exemplars if not full
    else if l_i > GetTopPriority($\mathbf{E}$).l **then**
      ReplaceTopPriority($\mathbf{E}$, (l_i, c_i)) ▷ Replace top priority if new one is higher
    end if
  end for
  return $\mathbf{E}$ ▷ Return the selected exemplars
end procedure

Global Efficiency: $E(G) = \frac{1}{N(N-1)} \sum_{i \neq j} d_{ij}$

- A quantitative metric for the structural information of the code.
- The communication efficiency between pairs of node
- Incorporate (potentially minimal) task-specific data for further training.

Experiments

Experiment Settings

CodePTMs

- GraphCodeBERT
- PLBART
- CodeT5
- UniXCoder

Tasks

- Code Understanding: Clone Detection & Defect Detection
- Code Generation: Code Translation & Code Summarization

Perturbation & Performance Recovery

Exemplars v.s. Random Examples

Tasks	Clone	Defect	Java to C#		C# to Java		Code Summarization
Metrics	F1	Accuracy	BLEU	EM	BLEU	EM	BLEU (Averaged)
Fine-Tuning with Perturbation							
GraphCodeBERT	94.75 -0.25	61.86 -1.02	62.75 -17.83	46.30 -13.10	55.33 -17.31	48.00 -18.80	17.59 -0.54
PLBART	93.99 +0.39	62.48 -0.40	76.60 -6.42	55.00 -9.60	73.31 -5.04	60.70 -4.30	17.84 -0.48
CodeT5	95.16 +0.16	63.03 -2.75	79.71 -4.32	60.40 -5.50	75.33 -4.54	62.80 -4.10	19.17 -0.38
UniXcoder	89.77 -1.59	60.94 -1.40	72.17 -6.78	57.40 -5.90	67.11 -7.11	58.20 -5.40	18.71 -0.52
Performance Recovery: Random Samples							
GraphCodeBERT	94.38 -0.37	60.91 -0.95	62.97 +0.22	50.10 +3.80	57.64 +2.31	48.30 +0.30	17.44 -0.15
PLBART	93.29 -0.70	61.16 -1.32	78.91 +2.31	56.40 +1.40	76.17 +2.86	61.70 +1.00	17.68 -0.16
CodeT5	95.46 -0.30	60.11 -2.92	80.11 +0.40	61.00 +0.60	74.34 -0.99	62.10 -0.70	19.04 -0.13
UniXcoder	89.40 -0.37	59.88 -1.06	72.92 +0.75	57.60 +0.20	67.19 +0.08	59.10 +0.90	18.88 +0.17
Performance Recovery: Exemplars							
GraphCodeBERT	94.36 -0.39	62.31 +0.45	70.10 +7.35	52.50 +6.20	59.15 +3.82	47.30 -0.70	17.66 +0.07
PLBART	94.10 +0.11	62.78 +0.30	79.91 +3.31	57.40 +2.40	76.54 +3.23	62.20 +1.50	17.94 +0.10
CodeT5	95.18 +0.02	63.20 +0.17	81.49 +1.78	62.60 +2.20	76.98 +1.65	64.20 +1.40	19.23 +0.06
UniXcoder	90.37 +0.60	61.15 +0.21	73.40 +1.23	58.80 +1.40	67.77 +0.66	59.10 +0.90	19.10 +0.39

Utilizing Code Structural Information

- How to utilize structural information in fine-tuning?

- Select code snippets rich in structural information for fine-tuning, instead of full fine-tuning.

Efficient Fine-Tuning

Methods	Java to C#		C# to Java		Code Summarization					
	BLEU	EM	BLEU	EM	Ruby	JavaScript	Go	Python	Java	PHP
10% Training Data (Random)										
GraphCodeBERT	79.81	59.60	75.00	59.00	11.99	15.06	18.43	19.15	18.57	25.22
PLBART	82.28	59.70	78.51	64.50	13.01	15.66	18.60	19.52	18.89	23.83
CodeT5	83.79	65.20	78.35	65.30	15.22	15.12	19.06	19.20	19.32	24.95
UniXcoder	78.09	62.10	74.67	63.60	14.59	16.12	18.71	19.62	20.31	25.84
10% Training Data (Exemplars)										
GraphCodeBERT	80.19 $_{+0.38}$	60.00 $_{+0.40}$	75.29 $_{+0.29}$	60.20 $_{+1.20}$	11.90 $_{-0.09}$	15.23 $_{+0.17}$	18.53 $_{+0.08}$	19.18 $_{+0.03}$	18.86 $_{+0.29}$	25.41 $_{+0.19}$
PLBART	82.72 $_{+0.44}$	60.90 $_{+1.20}$	78.88 $_{+0.37}$	64.20 $_{-0.30}$	13.47 $_{+0.46}$	16.57 $_{+0.91}$	19.03 $_{+0.20}$	19.55 $_{+0.03}$	19.09 $_{+0.20}$	23.90 $_{+0.07}$
CodeT5	84.40 $_{+0.61}$	65.70 $_{+0.50}$	79.17 $_{+0.82}$	65.90 $_{+0.60}$	15.48 $_{+0.26}$	16.22 $_{+1.10}$	19.57 $_{+0.51}$	19.96 $_{+0.76}$	20.38 $_{+1.06}$	26.09 $_{+1.14}$
UniXcoder	78.79 $_{+0.70}$	63.00 $_{+0.90}$	74.78 $_{+0.11}$	65.00 $_{+1.40}$	14.81 $_{+0.22}$	16.14 $_{+0.02}$	18.82 $_{+0.03}$	19.73 $_{+0.11}$	20.49 $_{+0.18}$	25.92 $_{+0.08}$
20% Training Data (Random)										
GraphCodeBERT	79.97	60.10	75.14	60.30	12.07	14.82	18.45	19.04	18.73	25.11
PLBART	81.91	60.30	78.17	63.70	13.12	15.33	18.83	19.45	18.83	23.45
CodeT5	84.01	65.00	78.34	64.00	15.23	16.01	19.44	19.91	20.38	25.51
UniXcoder	78.12	62.40	77.23	64.80	14.95	15.70	18.79	19.57	19.71	24.91
20% Training Data (Exemplars)										
GraphCodeBERT	80.20 $_{+0.23}$	60.50 $_{+0.40}$	74.85 $_{-0.29}$	59.60 $_{-0.70}$	11.87 $_{-0.20}$	15.56 $_{+0.74}$	18.66 $_{+0.21}$	19.19 $_{+0.15}$	18.66 $_{-0.07}$	25.35 $_{+0.24}$
PLBART	82.91 $_{+1.00}$	61.50 $_{+1.20}$	78.80 $_{+0.62}$	64.50 $_{+0.80}$	13.42 $_{+0.30}$	15.96 $_{+0.63}$	18.87 $_{+0.04}$	19.49 $_{+0.04}$	19.28 $_{+0.45}$	23.75 $_{+0.30}$
CodeT5	84.33 $_{+0.32}$	65.50 $_{+0.50}$	80.10 $_{+1.76}$	67.40 $_{+3.40}$	15.60 $_{+0.37}$	16.08 $_{+0.07}$	19.45 $_{+0.01}$	19.82 $_{-0.09}$	20.42 $_{+0.04}$	26.14 $_{+0.63}$
UniXcoder	78.97 $_{+0.85}$	63.20 $_{+0.80}$	77.30 $_{+0.07}$	65.10 $_{+0.30}$	14.99 $_{+0.04}$	16.05 $_{+0.35}$	18.84 $_{+0.05}$	19.77 $_{+0.20}$	20.14 $_{+0.43}$	25.80 $_{+0.89}$

More experiments can be found in our paper!

Analysis

Tasks	Code Translation (Avg.)			Code Summarization
Metrics	BLEU	EM	CodeBLEU	BLEU (Avg.)
CodeGen				
Full Fine-Tuning	80.46	65.70	81.42	20.57
10% Exemplars	77.07	64.05	79.25	19.95
20% Exemplars	80.07	65.80	81.46	20.35
30% Exemplars	**80.89**	**67.20**	**81.78**	**20.76**
CodeT5+				
Full Fine-Tuning	**83.97**	63.91	84.52	**21.75**
10% Exemplars	81.24	65.40	83.29	20.62
20% Exemplars	82.00	**66.60**	85.19	21.08
30% Exemplars	83.46	66.45	**85.61**	21.56

- Performance Comparison
  - Using 20% of dataset as exemplars nearly matches full fine-tuning performance
  - Using 30% of dataset as exemplars surpasses full fine-tuning on certain metrics

Analysis

Tasks	Code Translation (Avg.)			Code Summarization
Metrics	BLEU	EM	CodeBLEU	BLEU (Avg.)
CodeT5				
Raw AST	81.47	64.70	82.39	19.06
U-AST	**81.75**	**65.20**	**85.71**	**19.18**
UniXcoder				
Raw AST	77.79	63.42	80.98	19.22
U-AST	**78.05**	**64.50**	**83.94**	**19.31**

- Effectiveness of U-AST
  - Utilizes a variant of AST for computing global effectiveness
  - Solves the problem of poor connectivity among leaf nodes
- Performance Comparison: Significantly improve CodeBLEU

Conclusion

Main Contributions

(1) A collection of novel and practical methods for analyzing CodePTMs

(2) Comprehensive evaluations and analysis.

- Six CodePTMs perturbed and analyzed
- Four kind of tasks evaluated

(3) Research insights on using structural information during fine-tuning.

Thanks For Listening !

And more about Computational Intelligence in Software Engineering?

More works related to Computational Intelligence in SE

IEEE WCCI 2024 Yokohama Japan

上海人工智能实验室 Shanghai Artificial Intelligence Laboratory · NUS National University of Singapore · 香港大學自然語言處理實驗室 Natural Language Processing Group, The University of Hong Kong · NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE · DaSE SCHOOL OF DATA SCIENCE & ENGINEERING 数据科学与工程学院 · FUDAN UNIVERSITY NATURAL LANGUAGE PROCESSING · DeepMind · Institute for Infocomm Research I²R

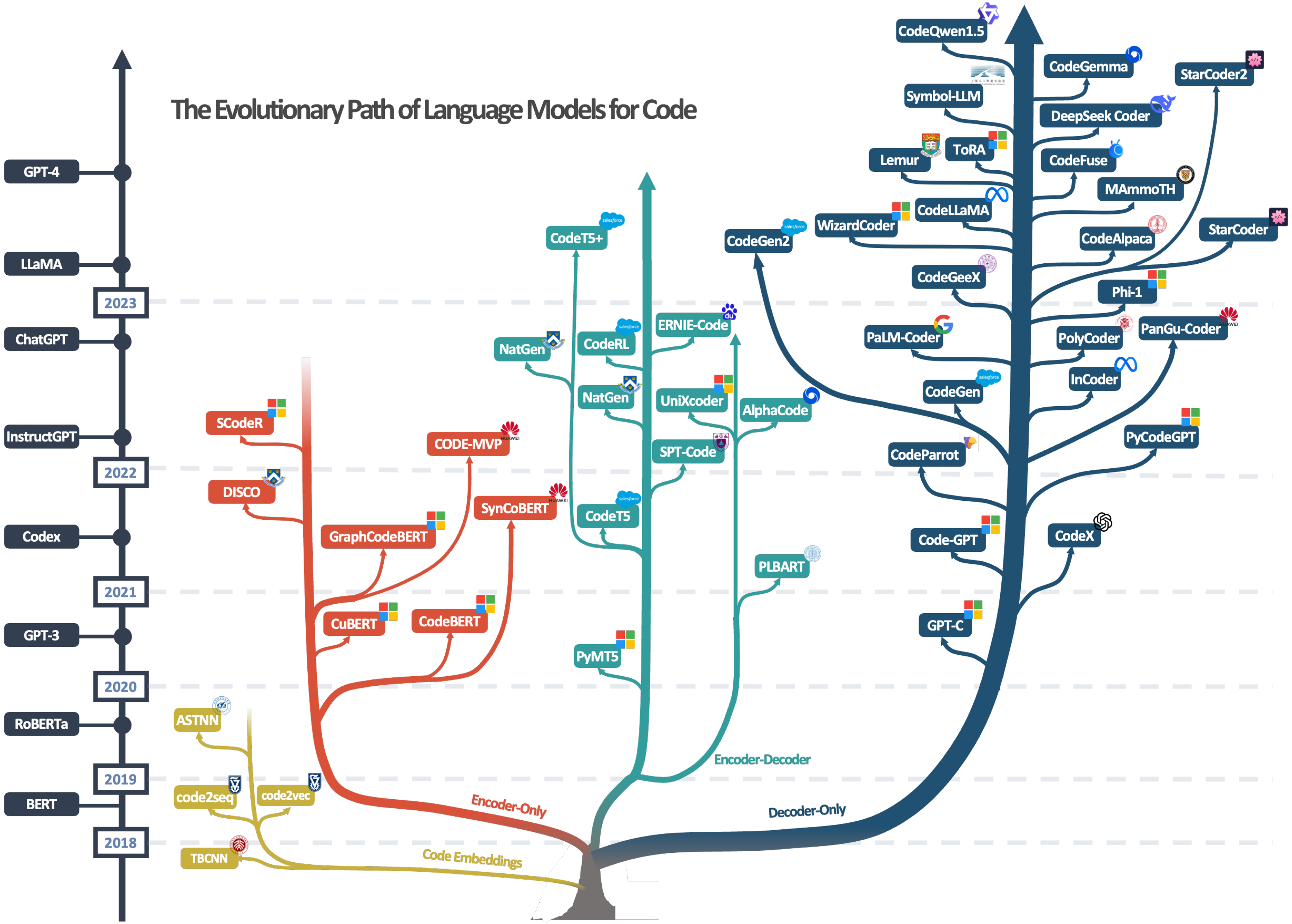
A Survey of Neural Code Intelligence: Paradigms, Advances and Beyond

Qiushi Sun, Zhirui Chen, Fangzhi Xu, Kanzhi Cheng, Chang Ma, Zhangyue Yin, Jianing Wang, Chengcheng Han, Renyu Zhu, Shuai Yuan, Qipeng Guo, Xipeng Qiu, Pengcheng Yin, Xiaoli Li, *Fellow, IEEE,* Fei Yuan, Lingpeng Kong, Xiang Li, and Zhiyong Wu

\> 50 models, > 20 categories of tasks, an extensive coverage of over 690 papers!

ArXiv: https://arxiv.org/abs/2403.14734

More works related to Computational Intelligence in SE



Project Repo: https://github.com/QiushiSun/NCISurvey