TransCoder

.::. LREC-COLING 2024

Qiushi Sun

qiushisun@u.nus.edu

May 2, 2024

Introduction

- LREC-COLING 2024

Paper: TransCoder: Towards Unified Transferable Code Representation Learning Inspired by Human Skills

Authors: Qiushi Sun, Nuo Chen, Jianing Wang, Xiang Li, Ming Gao

Overview

Backgrounds

- Code Pre-trained Models

- Pre-trained language models have advanced the SOTA across various NLP tasks.
- The success of NL applications has led to their adaptation in code.

Backgrounds

- Code Pre-trained Models

- Pre-trained language models have advanced the SOTA across various NLP tasks.
- The success of NL applications has led to their adaptation in code.

However, for applications, fine-tuning these models is task/language-specific.

Backgrounds

- Code Pre-trained Models

- Pre-trained language models have advanced the SOTA across various NLP tasks.
- The success of NL applications has led to their adaptation in code.

However, for applications, fine-tuning these models is task/language-specific.

Question: Can we integrate code knowledge from different tasks/languages?

Backgrounds

- How do humans learn to program?

Recap: How did we learn multiple PLs and become proficient in using them?

Generally, we:

- Master one PL by writing code, debugging, and adding comments.

Backgrounds

- How do humans learn to program?

Recap: How did we learn multiple PLs and become proficient in using them?

Generally, we:

- Master one PL by writing code, debugging, and adding comments.
- Then, learn additional languages.

Backgrounds

- How do humans learn to program?

Recap: How did we learn multiple PLs and become proficient in using them?

Backgrounds

- How do humans learn to program?

Recap: How did we learn multiple PLs and become proficient in using them?

It's the same for models! Code representation learning can be divided into two dimensions: Task and Language, which share similar principles and underlying logic.

Therefore, enabling models to perform continual learning in different scenarios, encouraging knowledge sharing, is very promising.

TransCoder - Learning Like Human Programmers

- Cross-Task Learning

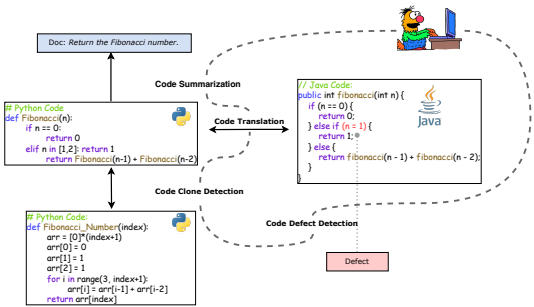


Fig: Cross-task code representation learning

- The knowledge acquired in pre-training phase has a gap between the downstream tasks.
- A huge difference in the amount of data between different tasks.
- Due to the catastrophic forgetting, training results for each task cannot be reused.

However, performing multiple code-related tasks (e.g., debugging, writing docs) should not cancel each other out but rather reinforce each other!

TransCoder - Learning Like Human Programmers

- Cross-Language Learning

Taking the CodeSearchNet[1] as an example.

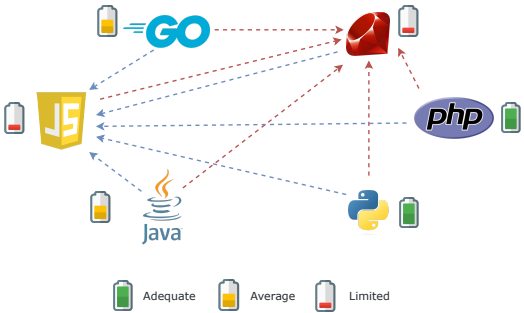


Fig: Cross-lang code representation learning

- Tuning and saving separate model instances for each PL is costly.
- A significant imbalance between different PLs
- Collecting and pre-processing code data are quite tedious.

But ... Different PLs share similar programming principles!

[1] CodeSearchNet challenge: Evaluating the state of semantic code search, arXiv:1909.09436, 2019

TransCoder
- Knowledge-acquisition

Our solution: Acquiring cross-task and cross-language "knowledge" through continual learning among all prevalent sources.

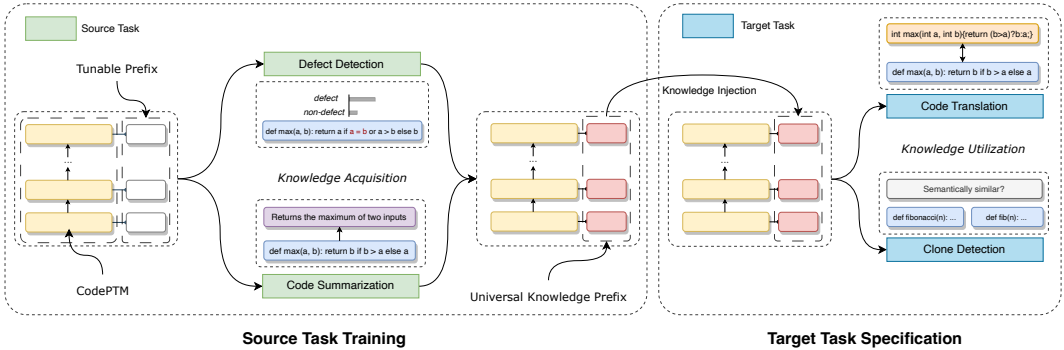
TransCoder
- Knowledge-acquisition

Our solution: Acquiring cross-task and cross-language "knowledge" through continual learning among all prevalent sources.

So... what is the carrier of knowledge? **A transferable knowledge prefix!**

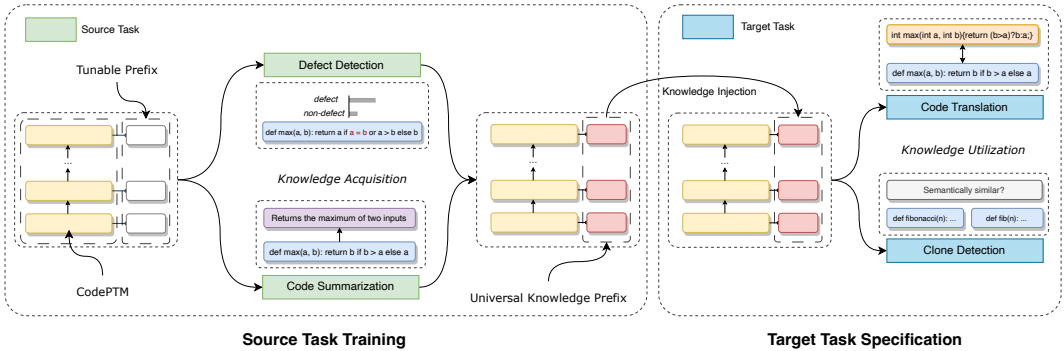
TransCoder - The Knowledge Prefix

- How code-related knowledge are passed down?



TransCoder - The Knowledge Prefix

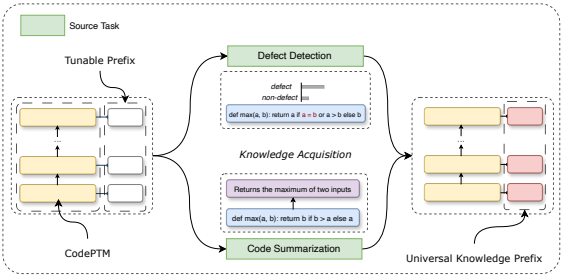
- How code-related knowledge are passed down?



The prefix will first acquire cross-task/language knowledge from sources, and then apply it to unseen task/language through prefix concatenation.

TransCoder - Pipeline

- Source Task Training



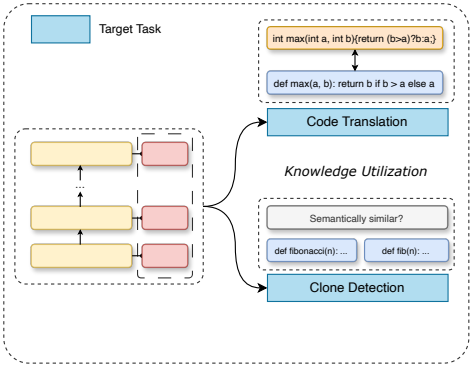
Source Task Training

- Add a set of prefixes to a CodePTM.
- It will continuously learn across several tasks or programming languages alongside the model.
- The sample sizes for each learning phase are kept similar.

Now, we have a set of prefixes that have "absorbed" a vast amount of code knowledge.

TransCoder - Pipeline

- Target Task Specification



Target Task Specification

- Utilize existing prefixes as initialization for the next stage.
- Incorporate (potentially minimal) task-specific data for further training.

The learned knowledge from the source tasks can be freely combined and applied to downstream tasks.

TransCoder - Performance

- Cross-Task Learning

Presenting experimental results using PLBART and CodeT5 as backbones.

Methods	CLS2Trans		Sum2Trans		CLS2Sum	Trans2Sum	Sum2CLS		Trans2CLS	
	BLEU	EM	BLEU	EM	BLEU	BLEU	Clone F1	Defect Acc	Clone F1	Defect Acc
CodeT5										
Fine-Tuning	**81.63**	65.80	81.63	65.80	19.56	19.56	**94.97**	64.35	94.97	64.35
TransCoder	81.43	**67.00**	**82.12**	**68.20**	**20.39**	**19.77**	93.70	**66.58**	**95.39**	**66.36**
PLBART										
Fine-Tuning	**78.17**	**62.70**	**78.17**	**62.70**	17.93	17.93	**92.85**	62.27	92.85	62.27
TransCoder	71.00	58.40	69.50	51.00	**18.62**	**18.25**	92.28	**64.58**	**92.91**	**64.98**

Table: The performance on the code cross-task learning

TransCoder - Performance

- Cross-Language Learning

Settings	Ruby	JavaScript	Go	Python	Java	PHP	Overall
CodeT5							
Fine-Tuning	15.24	16.21	19.53	19.90	20.34	26.12	19.56
TransCoder	**16.88**	**18.45**	**20.40**	**20.17**	**21.28**	**27.28**	**20.74**
PLBART							
Fine-Tuning	13.97	14.13	18.10	**19.33**	18.50	**23.56**	17.93
TransCoder	**15.32**	**15.00**	**18.67**	19.27	**19.44**	23.52	**18.54**

Table: Comparison between cross-language learning by TransCoder and full fine-tuning

TransCoder - Analysis

- Ablation Studies

Methods	Sum2CLS		Trans2CLS	
	Clone F1	Defect Acc	Clone F1	Defect Acc
CodeT5				
Random Knowl.	92.38	60.76	93.68	60.29
Universal Knowl.	**93.70**	**66.58**	**95.39**	**66.36**
PLBART				
Random Knowl.	90.96	61.02	91.15	61.64
Universal Knowl.	**92.28**	**64.58**	**92.91**	**64.98**

Table: Ablation study of universal code-related knowledge.

- Compared with randomly initialized prefixes.
- Significant performance improvements from TransCoder.

TransCoder - Analysis

- Low Resource Learning

Settings	Ruby	JavaScript	Go	Python	Java	PHP	Overall
5% Data							
Fine-Tuning	13.96	14.68	18.04	18.30	18.74	23.42	17.86
TransCoder	14.21	15.14	19.16	19.36	18.23	23.68	18.30
10% Data							
Fine-Tuning	15.22	15.12	19.06	19.20	19.32	24.95	18.81
TransCoder	16.05	16.63	20.21	20.07	20.48	26.20	19.94
20% Data							
Fine-Tuning	15.23	16.01	19.44	19.91	20.38	25.51	19.41
TransCoder	16.11	17.25	20.28	20.11	20.73	26.96	20.24

Table: Varying degrees of low-resource cross-language learning by TransCoder, using code summarization and CodeT5 backbone for evaluation.

The End

Thank You!