Exploring Distributional Shifts in Large Language Models for Code Analysis

Shushan Arakelyan , Rocktim Jyoti Das, Yi Mao, Xiang Ren

University of Southern California, IIT Delhi, Microsoft Azure AI (correspondence to: shushana@usc.edu)

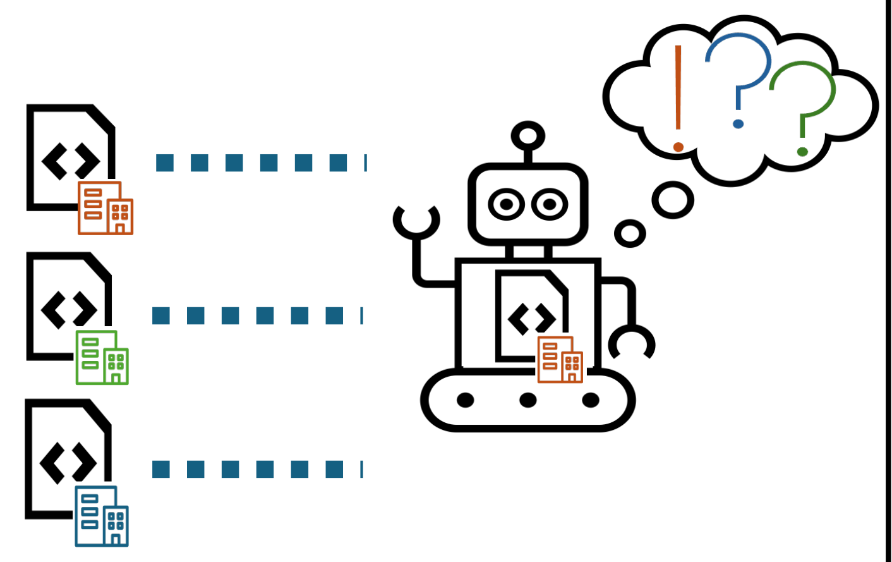
TL; DR

We study how three large language models with code capabilities - CodeT5, Codex, and ChatGPT - generalize to out-of-domain data on two applications **code summarization** and **code generation.** We establish that **all models are subject to distribution shift** naturally occurring in software. Finally, we show how different **domain adaptation techniques** handle such distribution shift.

Introduction

Motivation

Researchers using machine learning approaches for tackling issues in software engineering or cyber security have noticed that statistical learning systems trained on one project **do not generalize well to new projects**.



Studies looked at extensive lists of possible culprits that would predict whether or not a machine learning model built for one software system would generalize to a new software system.

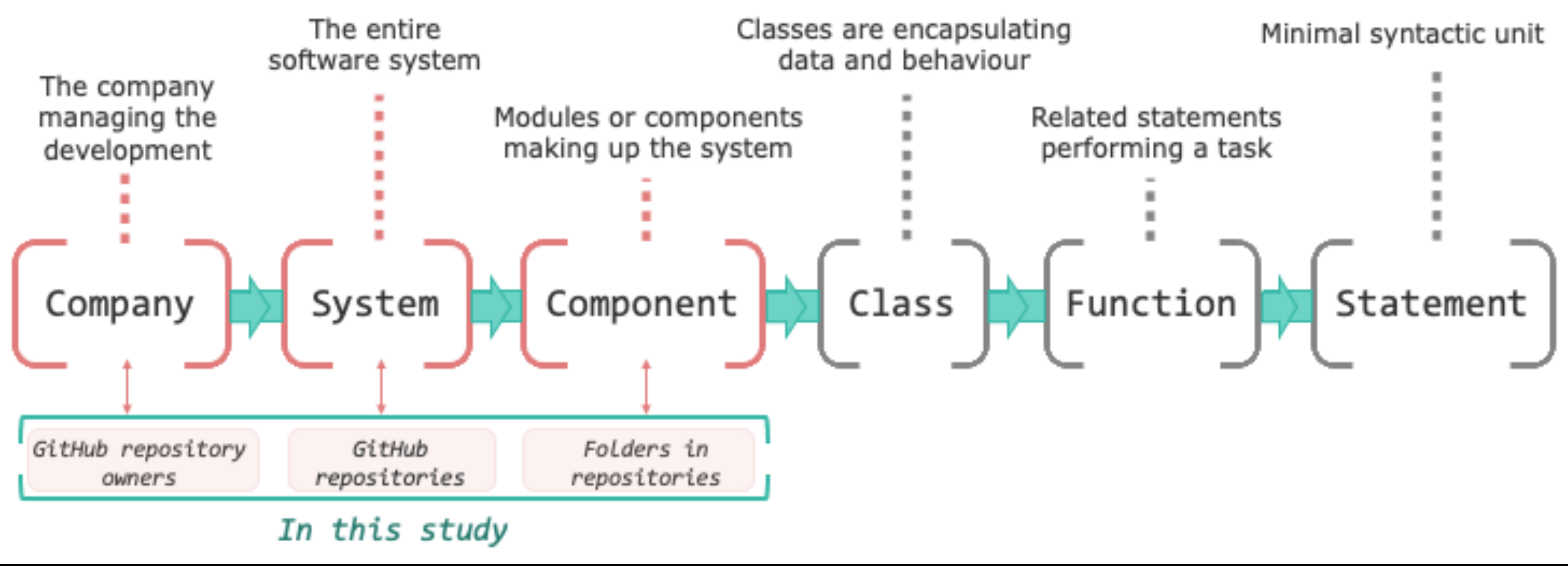
"What characteristics guarantee transferability?"

Open source	Yes/No
Global development	Yes/No
Code reviews	Yes/No
Static checkers	Yes/No
...	

Today, with the advances of LLMs, we are dealing with much **larger models**, trainer on much **more data**, that are **being deployed** at break-neck speed. However, whether the models **struggle** with unseen software systems after deployment remains unclear.

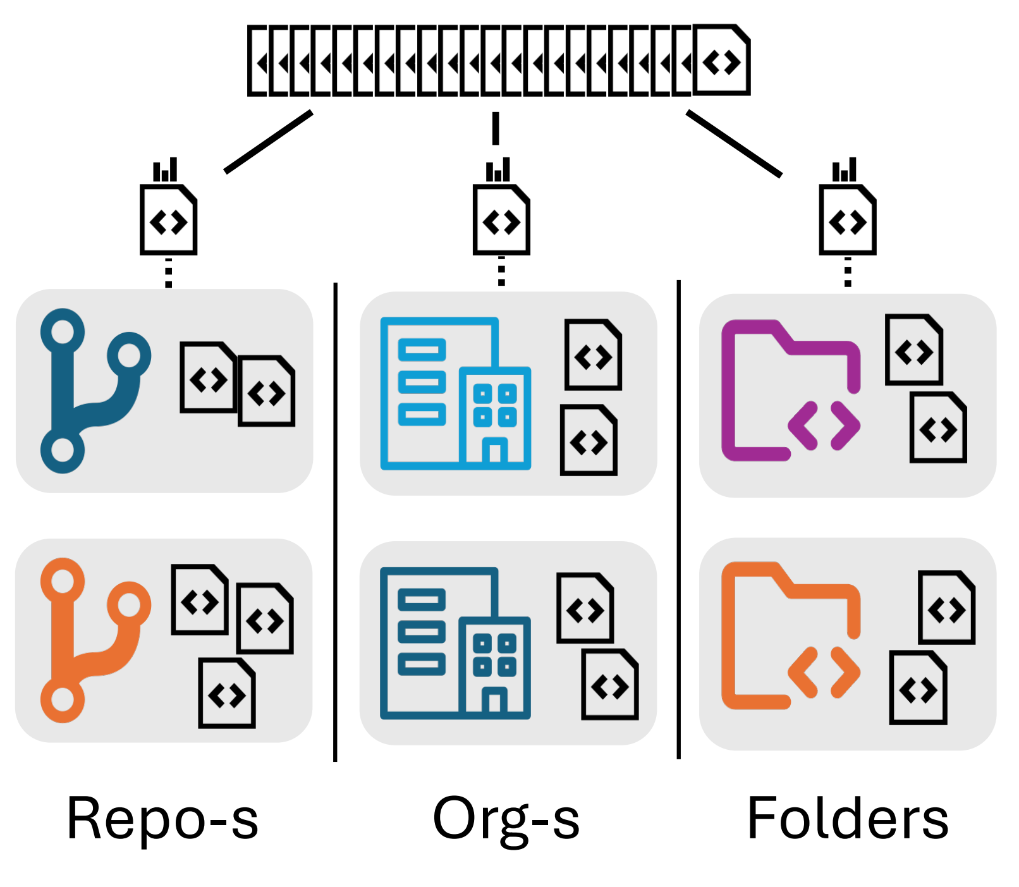
In this work

We study the challenges of distribution shifts that are stemming from the **hierarchical nature of software** data.


The company managing the development
The entire software system
Modules or components making up the system
Classes are encapsulating data and behaviour
Related statements performing a task
Minimal syntactic unit
Company
System
Component
Class
Function
Statement
GitHub repository owners
GitHub repositories
Folders in repositories
In this study

Analysis Setup

We used **CodeSearchNet dataset**'s **JavaScript** portion for our study. After grouping data by domain, we filter the domains to eliminate those containing less than 96 samples.



	Train	Train ≥ 96	Test ≥ 96
org	9737	195	8
repos	15858	147	15
folders	25268	100	10

We performed the analysis for two popular code tasks
- **code summarization**, which is evaluated with **BLEU** metric
- **code generation**, which is evaluated with **CodeBLEU** metric

*See the paper for results with **RougeL, CodeBERTScore** and **ChrF**.*

We separate all domains into a set of **training** and **target** domains, where the latter is unseen during training. For domain adaptation, we use some examples from a target domain and evaluate models on unseen examples from the same target domain.

*Follow the **QR code** at the top for access to **code, data, individual experimental results and model outputs**.*

Analysis

How do code models perform on new domains?

We test the capacity for generalization to new domains by comparing the performance of the models that have been adapted to the new domain (**ID**) to those that only encountered out-of-domain data (**OOD**).
For CodeT5 we use few-shot finetuning or PE finetuning for adaptation, whereas for Codex and ChatGPT we use ICL.

Code summarization	folder			repo			org		
	8-shot	16-shot	32-shot	8-shot	16-shot	32-shot	8-shot	16-shot	32-shot
CodeT5 FT ID	14.39	16.06	18.31	12.68	14.73	16.82	13.14	16.35	17.65
CodeT5 LoRA ID	16.57	19.07	20.93	15.22	17.14	21.20	15.61	18.56	20.87
CodeT5 FT random	3.58	4.30	5.02	4.35	4.70	5.79	4.53	5.47	6.27
CodeT5 LoRA random	3.69	4.37	4.92	4.70	5.56	5.92	5.27	5.53	6.26

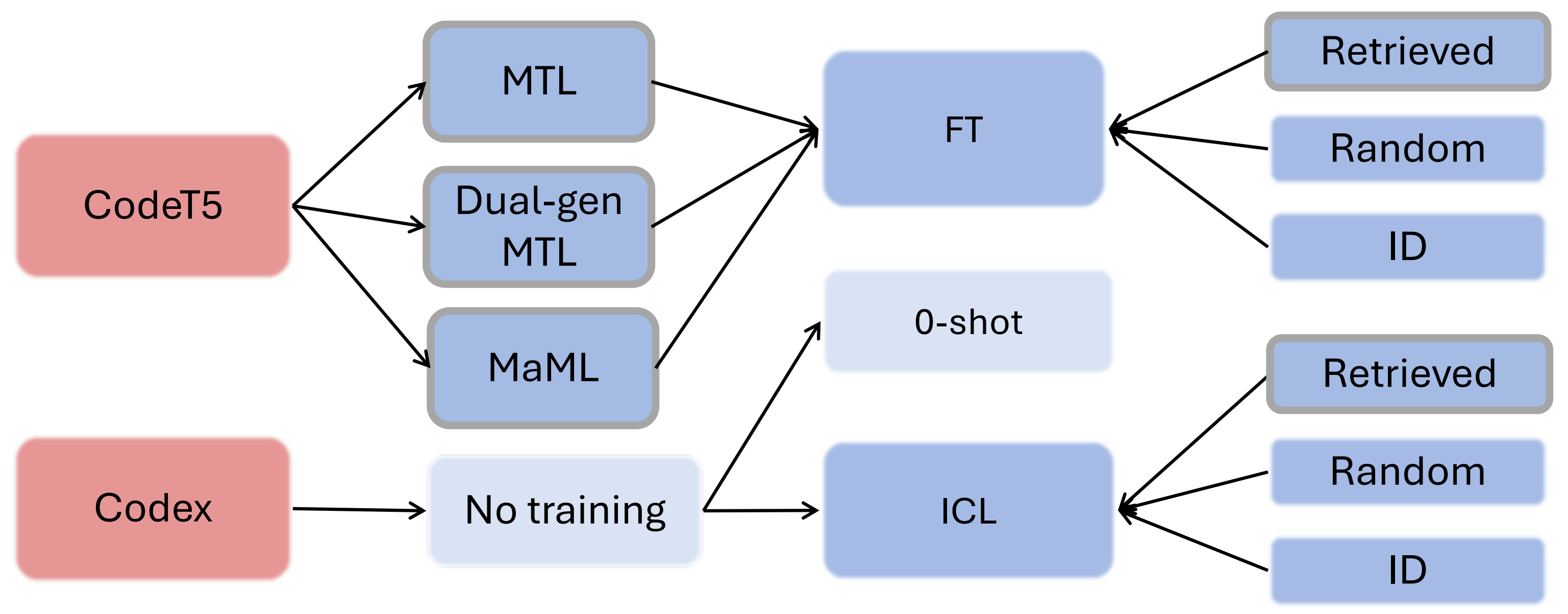
Code generation	folder			repo			org		
	8-shot	16-shot	32-shot	8-shot	16-shot	32-shot	8-shot	16-shot	32-shot
CodeT5 FT ID	14.67	15.22	16.13	16.15	17.42	18.62	14.54	15.34	16.43
CodeT5 LoRA ID	14.14	15.06	16.36	16.23	17.45	18.96	14.17	15.30	16.62
CodeT5 FT random	15.23	14.94	15.15	14.19	14.14	14.67	13.39	13.43	14.44
CodeT5 LoRA random	14.45	14.29	15.37	14.29	13.74	15.04	13.76	13.85	14.81

Code summarization		folder	repo	org
Codex	instr. only *(0-shot)*	1.55	1.52	1.61
	ICL random *(8-shot)*	7.17	6.84	6.73
	ICL ID *(8-shot)*	20.34	19.00	20.72
ChatGPT	instr. only *(0-shot)*	5.74	5.48	4.63
	ICL random *(8-shot)*	5.47	6.58	6.48
	ICL ID *(8-shot)*	7.47	9.15	7.54
Code generation		**folder**	**repo**	**org**
Codex	instr. only *(0-shot)*	5.49	5.72	5.77
	ICL random *(8-shot)*	16.82	17.47	16.82
	ICL ID *(8-shot)*	25.73	24.64	23.87
ChatGPT	instr. only *(0-shot)*	8.45	8.39	8.04
	ICL random *(8-shot)*	12.95	13.19	12.70
	ICL ID *(8-shot)*	15.17	15.81	15.55

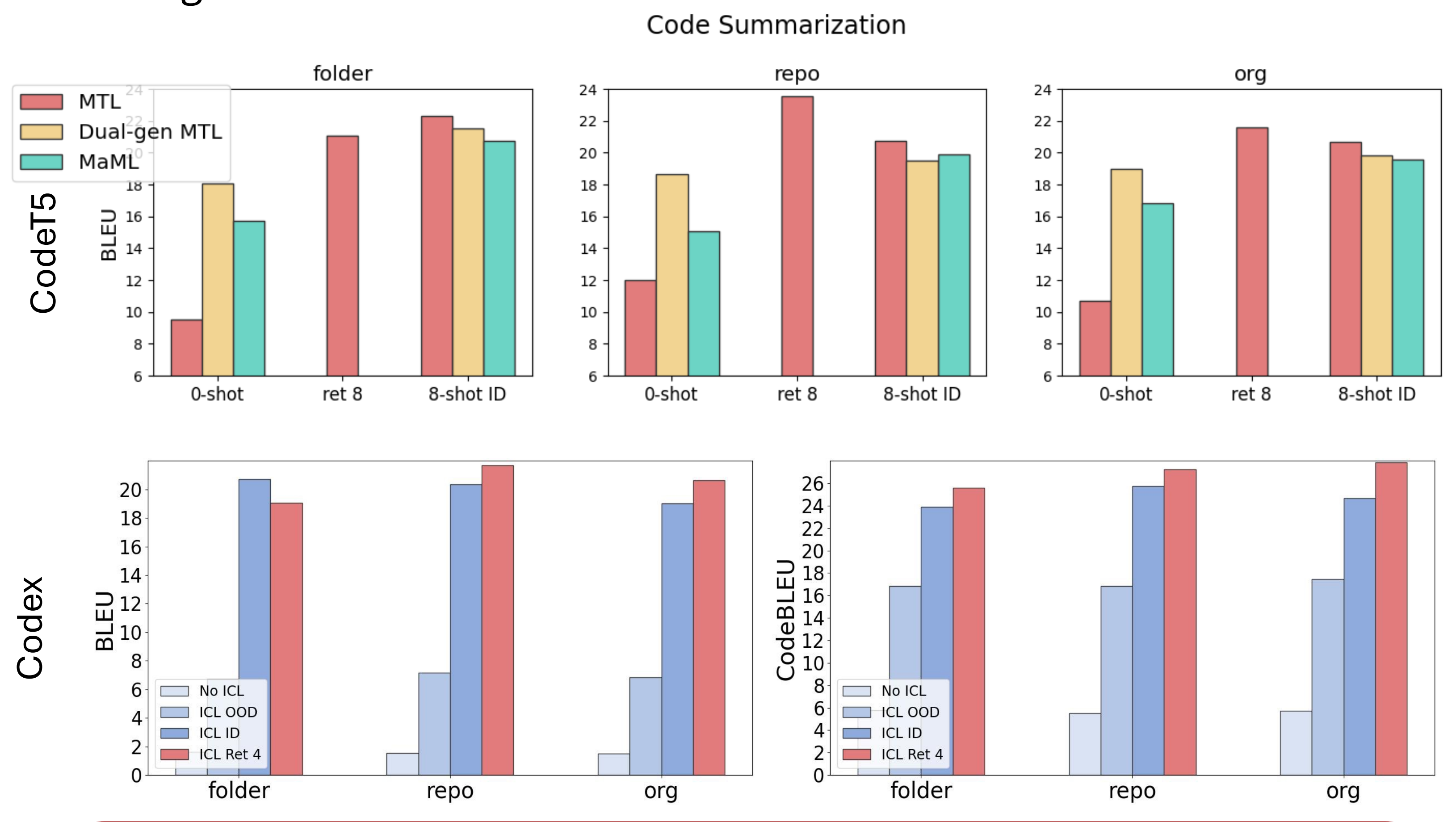
Splits naturally occurring in software present distributional shift challenge

How to get better out-of-domain generalization?

There are many reasons why adapting with labelled data might impractical, so we consider other approaches that would **not require labelled data** and could potentially close the performance gap.



For **retrieved supervision** we used 4/8/32 most similar examples from training data, combined and deduplicated for CodeT5, or used as demonstrations for Codex in ICL.
*See the paper for more experimental results, as well results on a challenge scenario with a new dataset – **the Vault**.*



- **Domain adaptation can be effective with a very small amount of labelled/unlabelled data**
- **Retrieving examples for supervision is effective for combating distribution shift**