Code Intelligence

Qiushi Sun
qiushisun.github.io
𝕏 @qiushi_sun

The Rise and Potential of Neural Code Intelligence

Large Language Models

We are quite familiar with them

Large Language Models: for Code

Code variants of LLMs

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

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

[arXiv: 2403.14734 [v5] Sun, 26 Jan 2025]

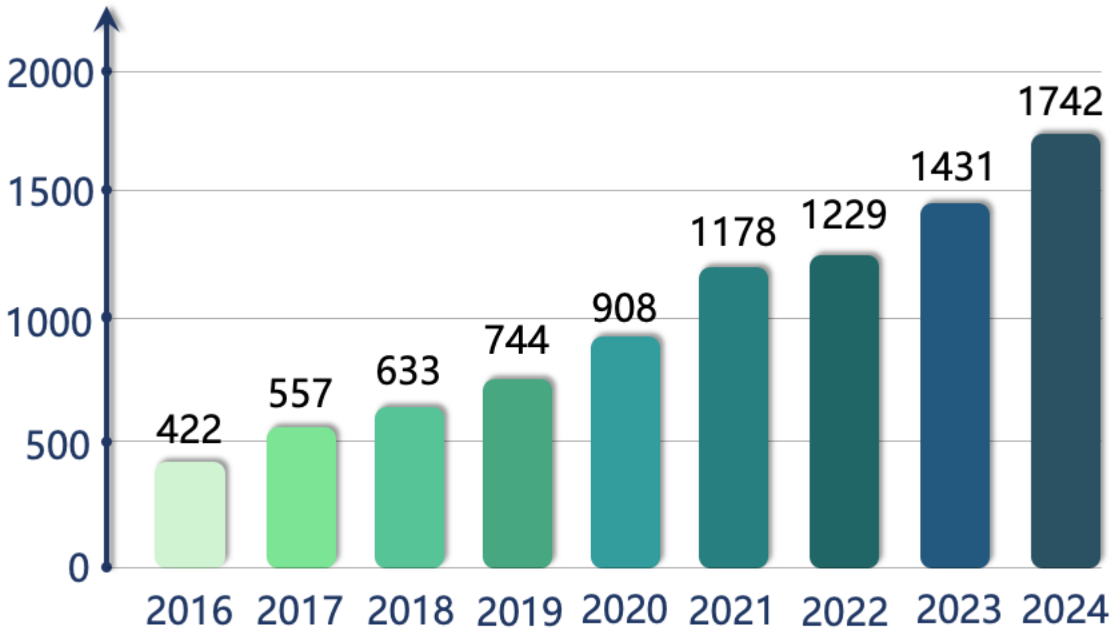
The Develop Timeline of CodeLMs

1. An increasing number of researchers are diving into

2. It is generally positively correlated with the development of LMs.

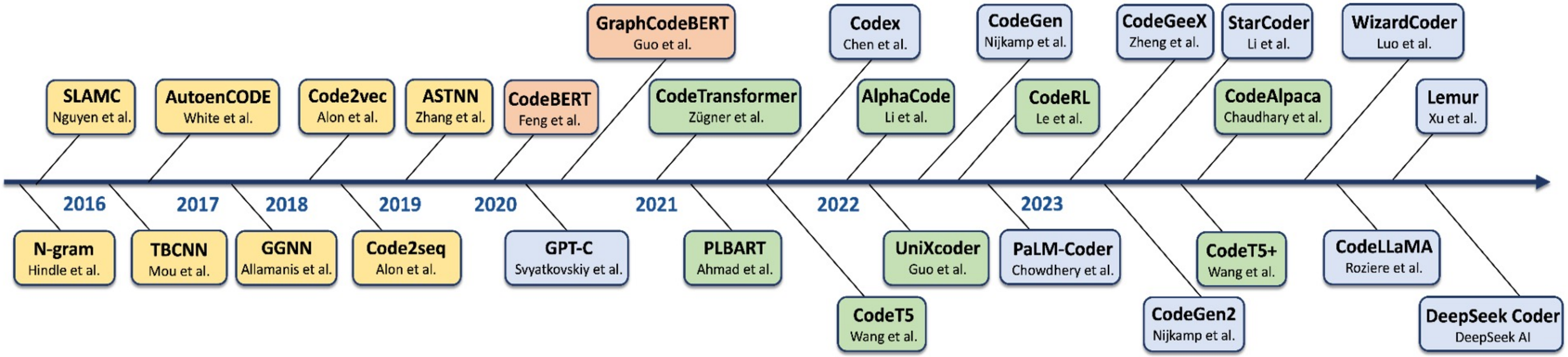
Papers are usually pubed at:

1. ML venues: NIPS, ICLR, ICML …
2. NLP venues: *ACL, COLM, …
3. SE venues: ICSE, ASE, ISSTA, …



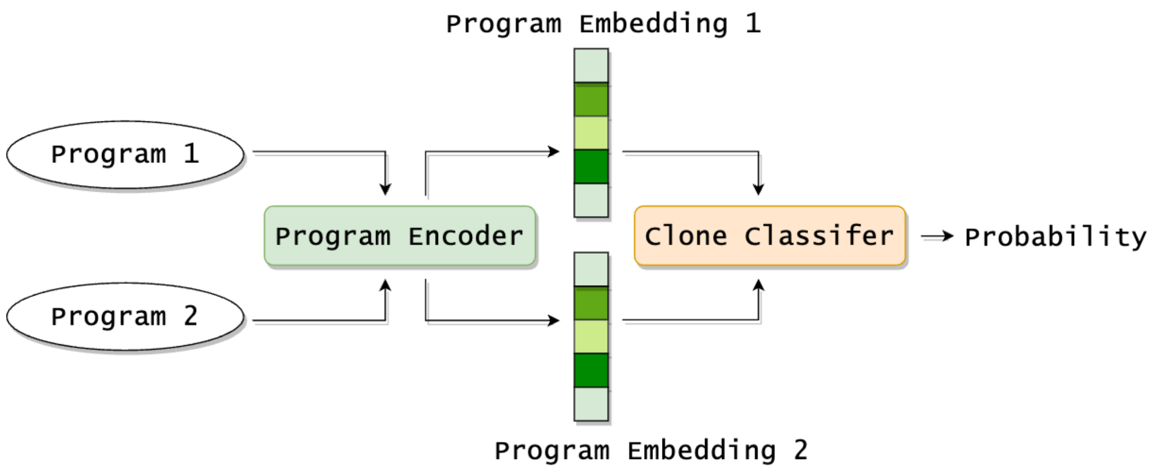
The Develop Timeline of CodeLMs

A story through CodeLMs.

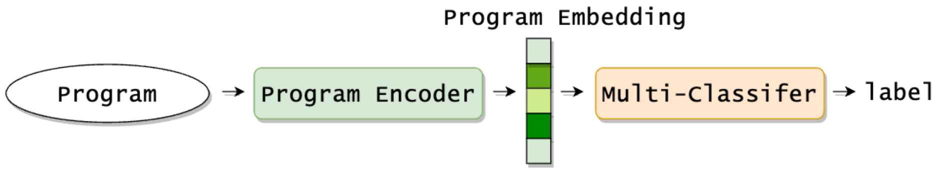


Code-Related Tasks

How it starts?



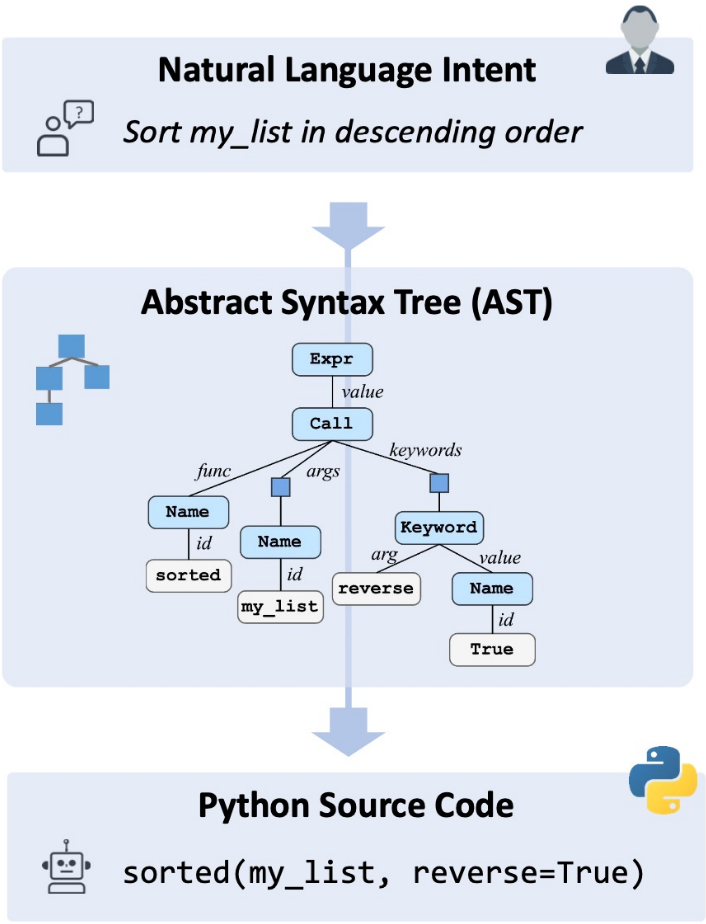
Clone Detection



Code Classification

Before LLMs, we were most focused on how to construct code representations.

Code Representation Learning



– Use Abstract Syntax Trees as general-purpose intermediate meaning representations

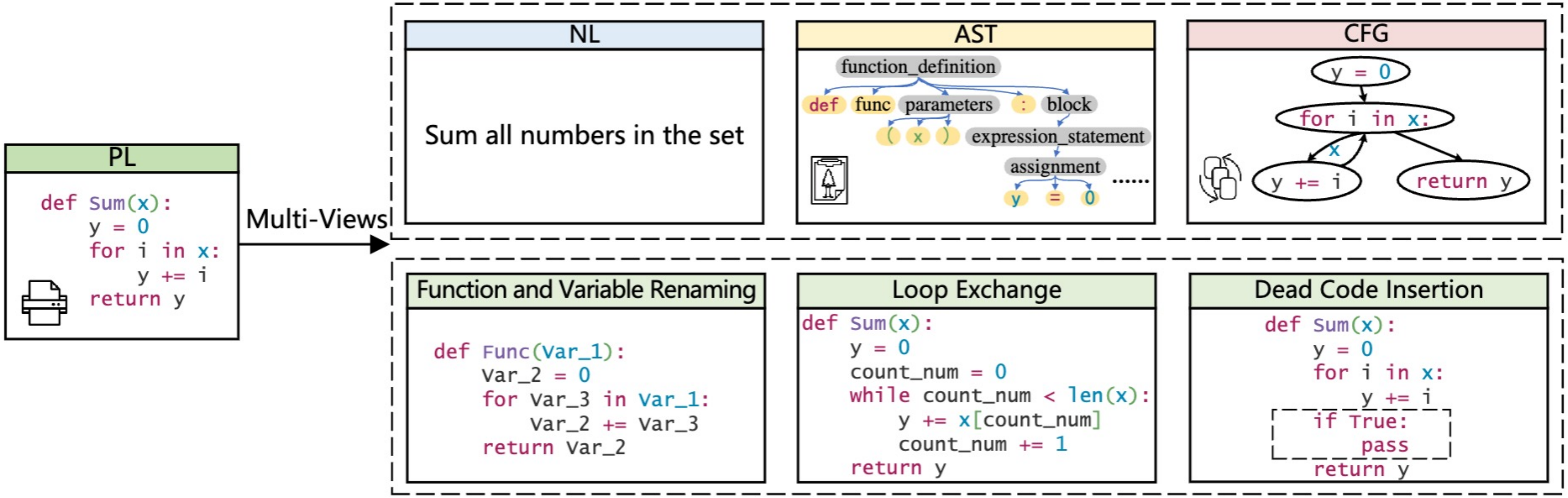
– $p_\theta(\; | \;)$ is a seq-to-tree model using program grammar as prior syntactic knowledge to constrain decoding space

– Deterministic transformation to source code

The pastoral era of code generation and understanding

Code-Related Tasks

How code differ from NL

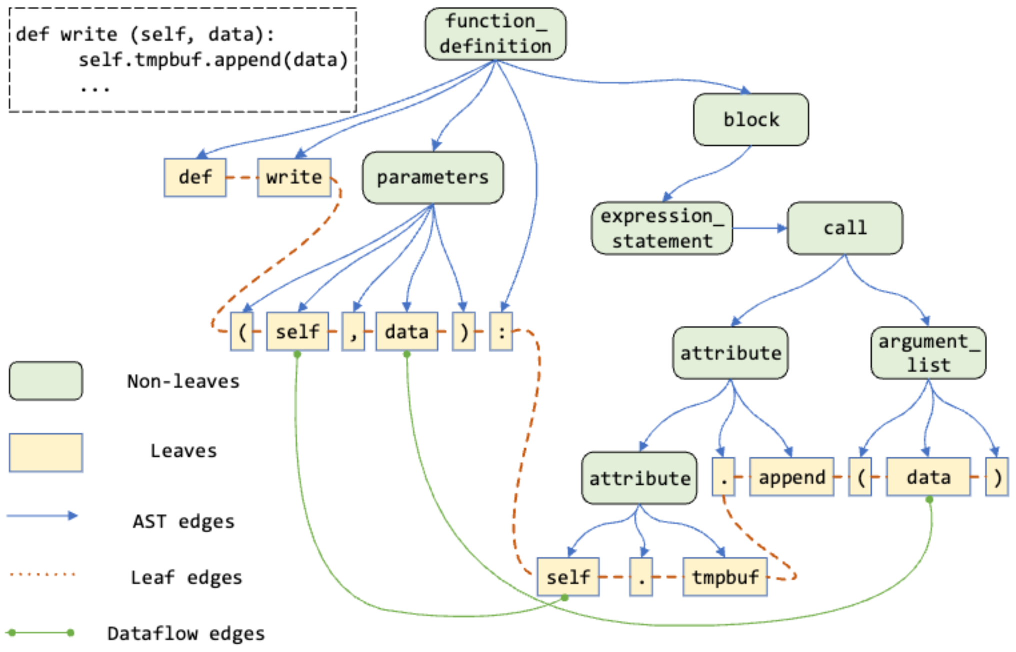


Multiple views of Source Code

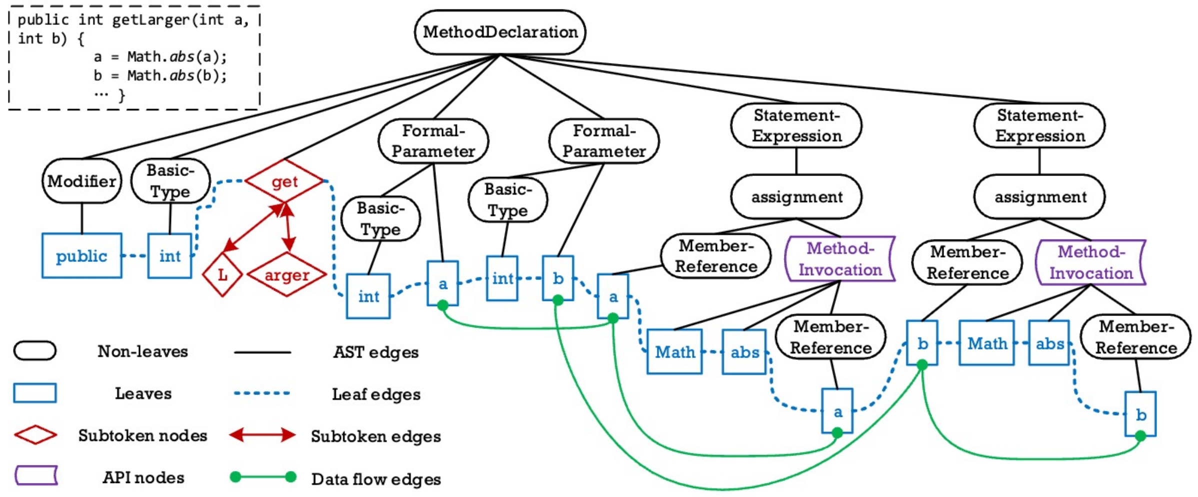
CODE-MVP: Learning to Represent Source Code from Multiple Views with Contrastive Pre-Training, NAACL 2022 Findings

Solving Code-Related Tasks

How code differ from NL



"Enhanced" ASTs

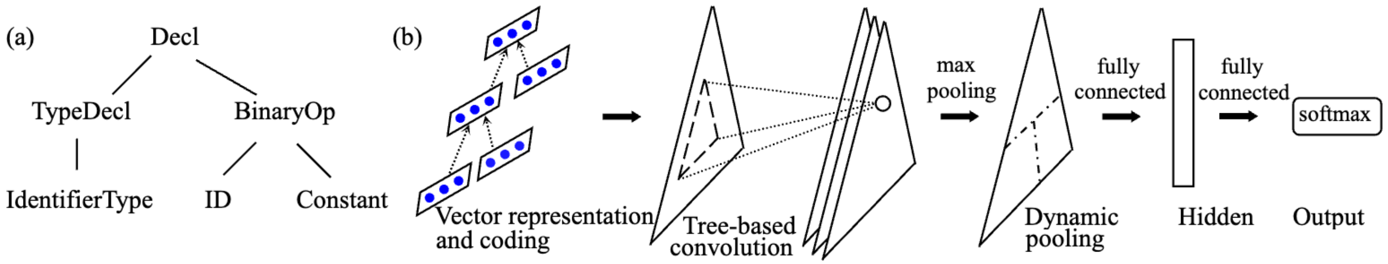


Another "Enhanced" ASTs

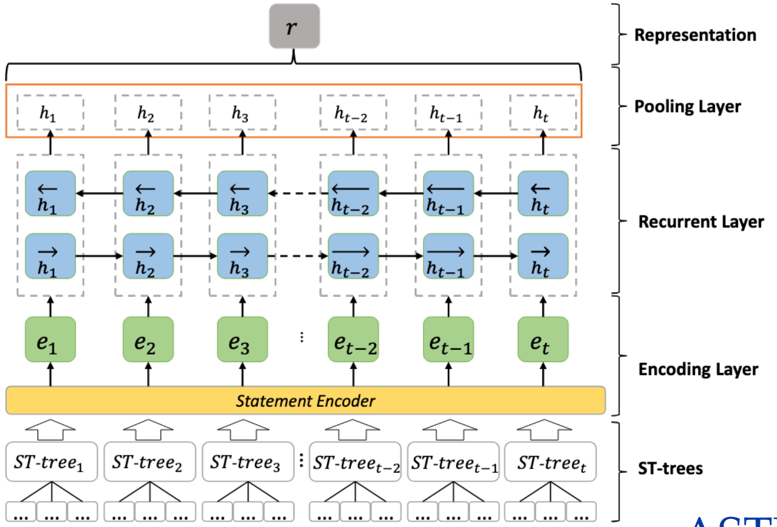
CAT-probing: A Metric-based Approach to Interpret How Pre-trained Models for Programming Language Attend Code Structure, EMNLP 2022 Findings

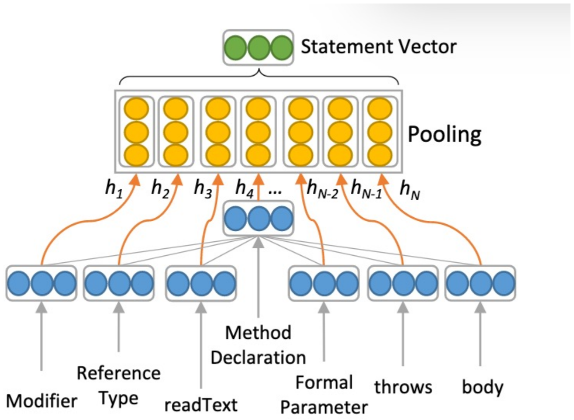
A Neural Network Architecture for Program Understanding Inspired by Human Behaviors, ACL 2022

Typical CodeLMs before Transformer



TBCNN





ASTNN

A Novel Neural Source Code Representation Based on Abstract Syntax Tree, ICSE 2019

Convolutional Neural Networks over Tree Structures for Programming Language Processing, IJCAI 2016

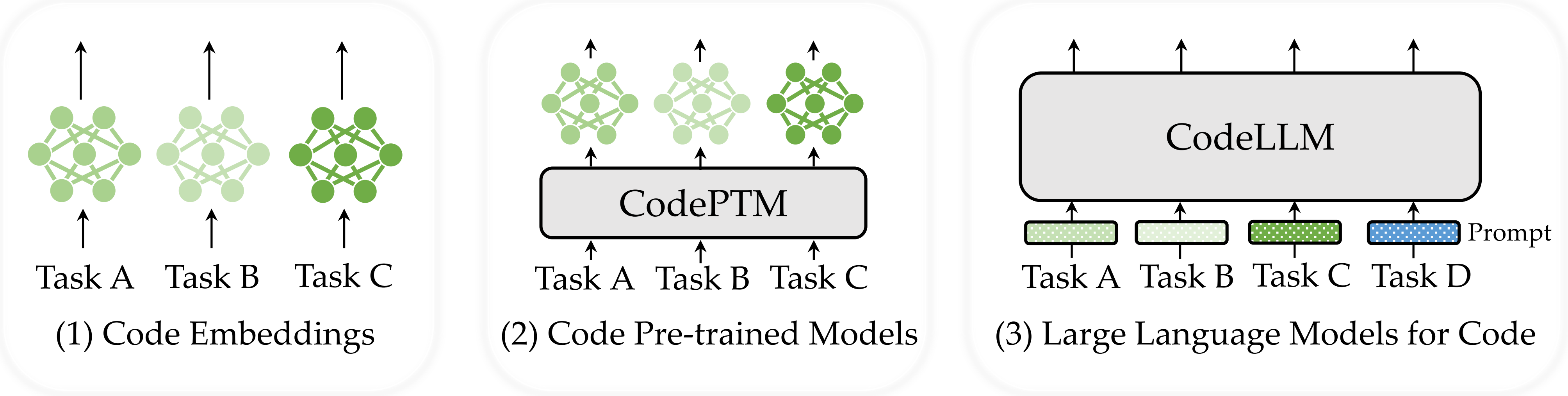
Solving Code-Related Tasks

And more tasks …

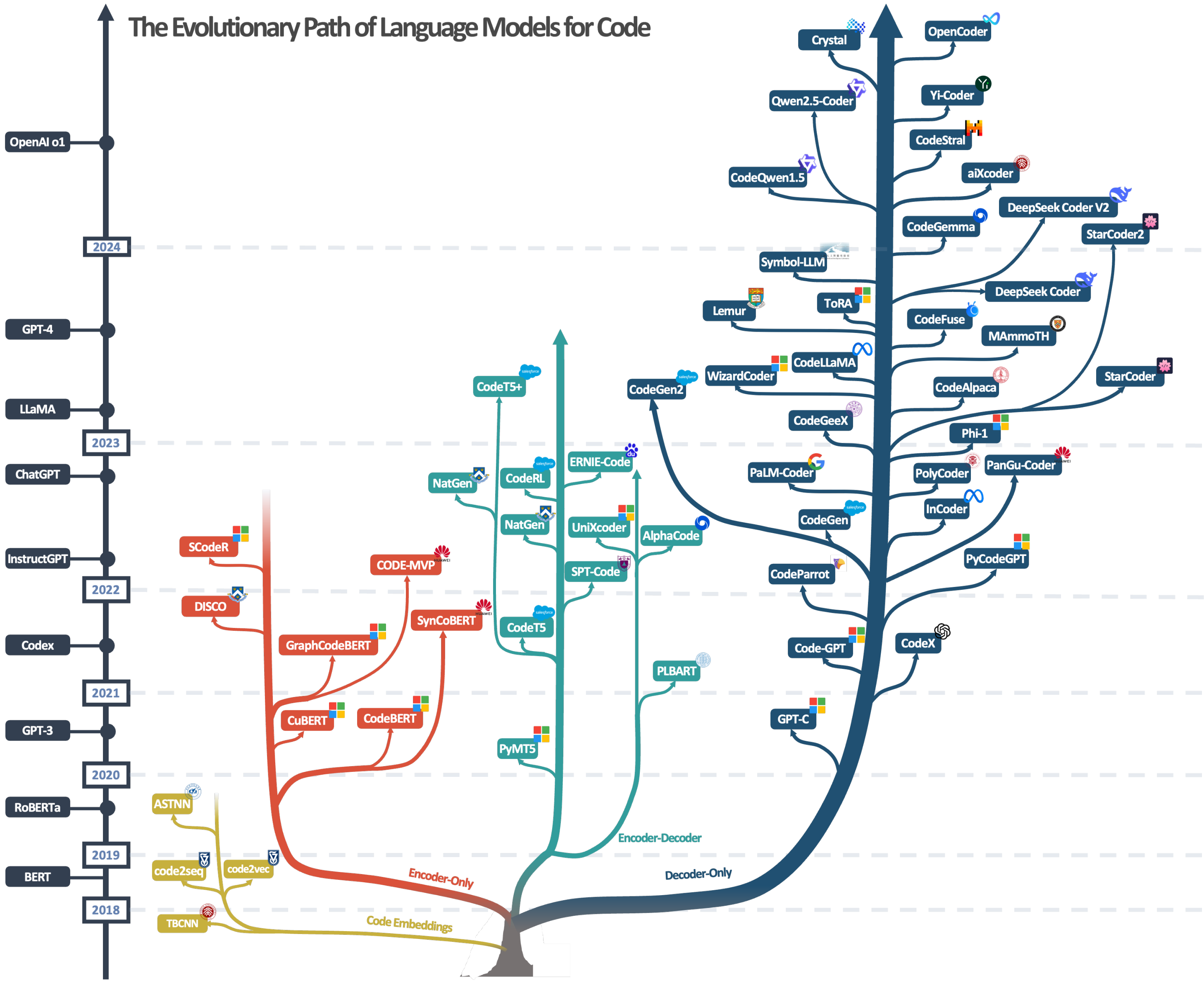
Task	Dataset	Date	# PLs.	Description
Clone Detection	POJ-104 [16] [link]	2014	2	a program classification dataset of 52K C/C++ programs
	BigCloneBench [108] [link]	2015	1	a clone detection dataset of eight million Java validated clones
	CLCDSA [109] [link]	2019	3	a cross-language clone dataset of more than 78K solutions
Defect Detection	Devign [78] [link]	2019	1	a dataset of vulnerable C functions
	CrossVul [110] [link]	2021	>40	a dataset of 13K/27K (vulnerable/non-vulnerable) files
	DiverseVul [111] [link]	2023	2	a dataset of 18K/330K (vulnerable/non-vulnerable) functions
Code Repair	Defects4J [link]	2014	1	a database of real Java bugs
	DeepFix [83] [link]	2017	1	a dataset of 7K erroneous C programs for 93 programming tasks
	QuixBugs [112] [link]	2017	2	a multilingual benchmark of similar buggy programs
Code Search	CodeSearchNet [113] [link]	2019	6	a dataset of 6M functions and natural language queries
	AdvTest [114] [link]	2021	1	a Python code search dataset filtered from CodeSearchNet
	WebQueryTest [114] [link]	2021	1	a testing set of Python code search of 1K query-code pairs
Code Translation	CodeTrans [114] [link]	2021	2	a C#/Java dataset collected from several repos
	CoST [115] [link]	2022	7	a dataset containing parallel data from 7 programming languages
	CodeTransOcean [116] [link]	2023	45	a large-scale comprehensive benchmark for code translation
Code Completion	GitHub Java Corpus [2] [link]	2013	1	a giga-token corpus of Java code from a wide variety of domains
	Py150 [117] [link]	2016	1	a corpus of Python programs from GitHub
	LCC [118] [link]	2023	3	a benchmark of code completion with long code context
Code Summarization	CODE-NN [119] [link]	2016	2	a dataset of (title, query) pairs from StackOverflow
	TL-CodeSum [120] [link]	2018	1	a dataset containing 69K pairs of (API sequence, code, summary)
	CodeSearchNet [113] [link]	2019	6	a dataset of 6M functions and natural language queries
GitHub	CommitGen [121] [link]	2017	4	a multilingual dataset collected from open source projects
	CommitBERT [122] [link]	2021	6	a multilingual dataset of code modification and commit messages
	SWE-bench [123] [link]	2023	1	a benchmark of 2K SE problems and corresponding PRs

Well, essentially, they all require task-specific modeling from scratch.

The Paradigm Shift of CodeLMs



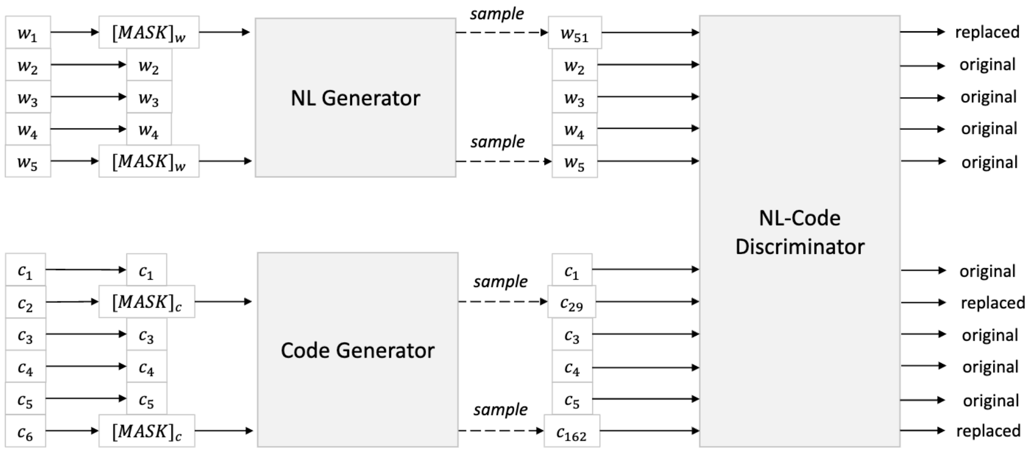
The evolution from the perspective of models



Code Pre-trained Models (CodePTMs)

CuBERT and CodeBERT

CodeBERT:
A Pre-Trained Model for Programming and Natural Languages



Learning and Evaluating Contextual Embedding of Source Code

CodeBERT: A code version of RoBERTa

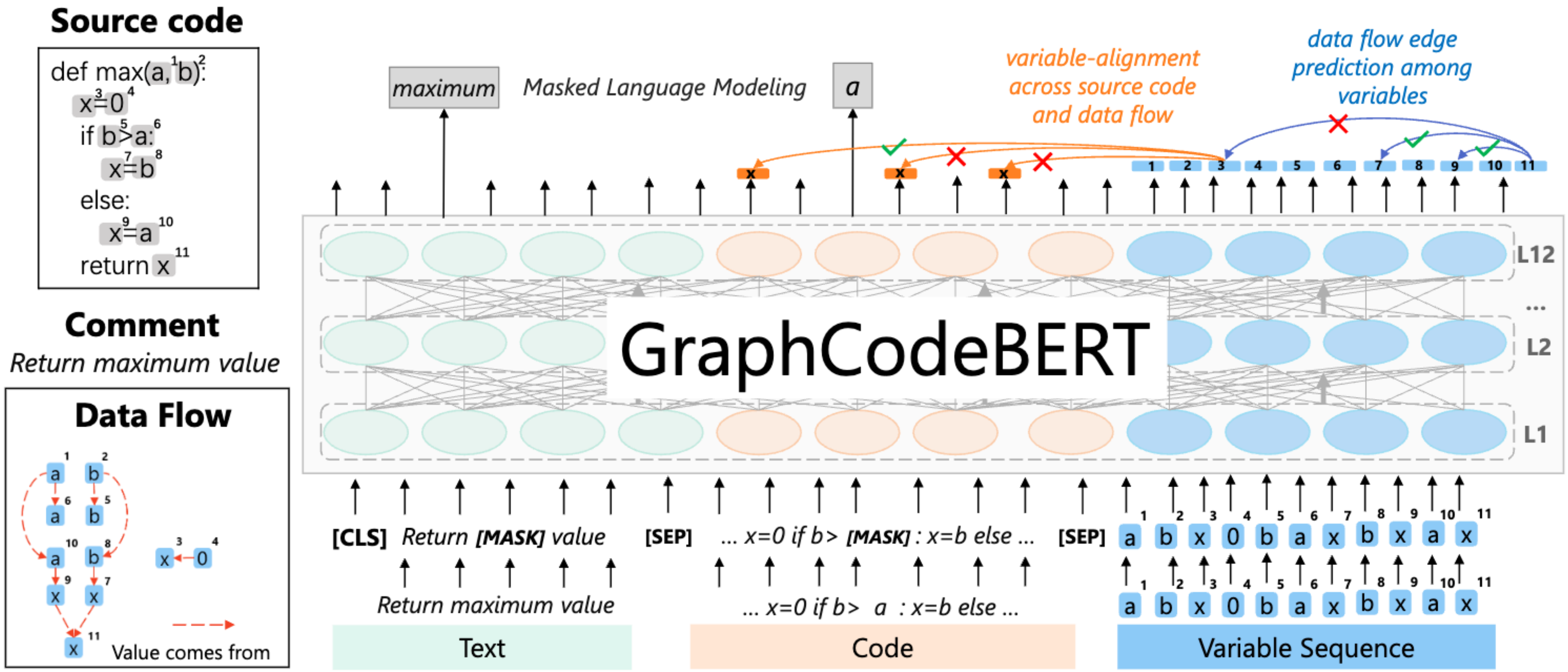
CuBERT: A code version of BERT

Learning and Evaluating Contextual Embedding of Source Code, ICML 2020

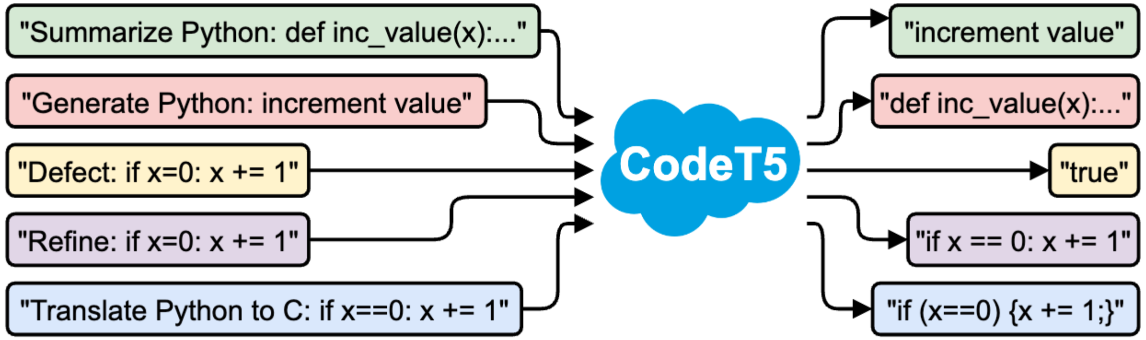
CodeBERT: A Pre-Trained Model for Programming and Natural Languages, EMNLP 2020 Findings

GraphCodeBERT

How about typical "BERT-Style" Training meets Code Structures



T5 and BART like CodePTMs



PLBART Encoder Input	PLBART Decoder Output
Is 0 the **[MASK]** Fibonacci **[MASK]** ? <En>	<En> Is 0 the **first** Fibonacci **number** ?
public static main (String args [ ]) { date = Date ( ) ; System . out . (String . format (" Current Date : % tc " ,)) ; } <java>	<java> public static **void** main (String args [ ]) { **Date** date = **new** Date ( ) ; System . out . **printf** (String . format (" Current Date : % tc " , **date**)) ; }
def addThreeNumbers (x , y , z) : NEW_LINE INDENT return **[MASK]** <python>	<python> def addThreeNumbers (x , y , z) : NEW_LINE INDENT return **x + y + z**

PLBART: Denoising Pre-training

1. Standard denoising training for T5 and BART models
2. Identifier types can be used for sequence labeling learning
3. Seq2seq learning unique to code
    - Deobfuscation
    - Naturalization
    - Mutual generation of code and comments

CodeT5: Identifier-aware Unified Pre-trained Encoder-Decoder Models for Code Understanding and Generation, EMNLP 2021

Unified Pre-training for Program Understanding and Generation, NAACL 2021

Some issues

1. When introducing code features, changes to the vocabulary, input format, or attention patterns often prevent generalization.

2. The generation capability is quite weak.

3. When adapting to downstream tasks, fine-tuning is typically required.

Architecture	Models	Struct.	Base	Strategy	Size
Encoder	CuBERT [179]	✗	-	MLM + NSP	340M
	CodeBERT [28]	✗	RoBERTa	MLM + RTD	125M
	GraphCodeBERT [182]	✓	CodeBERT	MLM + Edge Pred. + Node Align.	125M
	SynCoBERT [184]	✓	CodeBERT	MMLM + IP + TEP + MCL	125M
	CODE-MVP [185]	✓	GraphCodeBERT	FGTI + MCL + MMLM	125M
	SCodeR [187]	✓	UniXcoder	Soft-Labeled Contrastive Pre-training	125M
	DISCO [186]	✓	-	MLM + NT-MLM + CLR	110M
Enc-Dec	PLBART [30]	✗	-	Denoising Pre-training	140M/406M
	CodeT5 [29]	✓	-	MSP + IP + MIP + Bimodal Generation	60M/220M/770M
	PyMT5 [194]	✗	-	MSP	374M
	UniXcoder [195]	✓	-	MLM + ULM + MSP + MCL + CMG	125M
	NatGen [196]	✓	CodeT5	Code Naturalization	220M
	TreeBERT [192]	✓	-	TMLM + NOP	210M
	ERNIE-Code [197]	✗	mT5	SCLM + PTLM	560M
	CodeExecutor [198]	✗	UniXcoder	Code execution + Curriculum Learning	125M
	LongCoder [118]	✗	UniXcoder	CLM	150M
Decoder	GPT-C [199]	✗	-	CLM	366M
	CodeGPT [114]	✗	-	CLM	124M
	PyCodeGPT [200]	✗	GPT-Neo	CLM	110M

If you want to learn more …

Check out this post I wrote during my UG thesis in 2022!

知乎 | 首发于 Paper Reading

… 写文章

[代码表征] Code预训练语言模型学习指南（原理/分析/代码）

NaturalSelection
香港大学 计算机科学博士在读

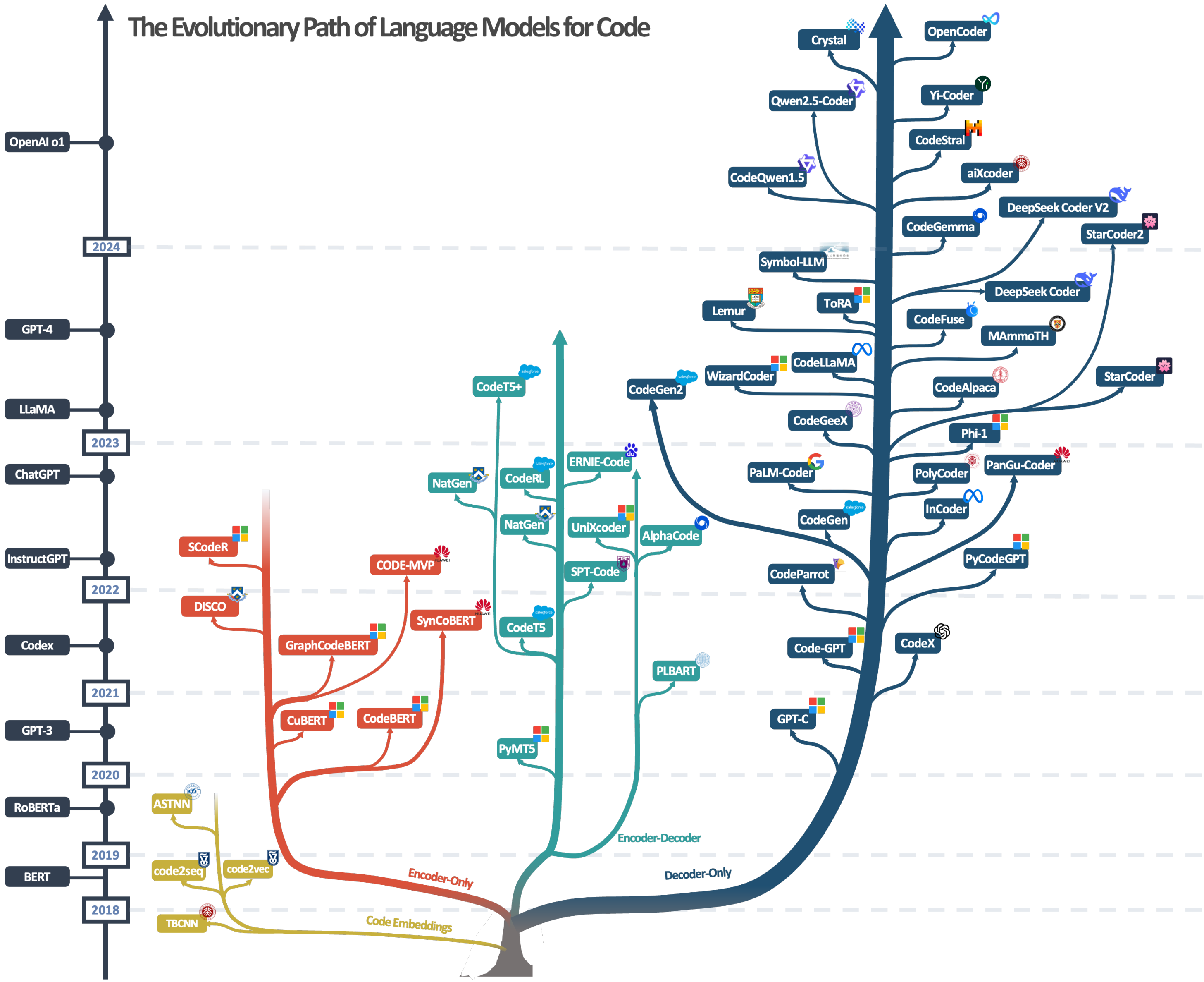
雨打蕉叶潇潇几夜、sonta 等 113 人赞同了该回答 ›

自从2020年CodeBERT开了代码表征预训练模型（本文称之为CodePTM）这个新坑后，在短短两年的时间内出现了若干个程序设计语言（Programming Language，称之为PL，与Natural Language，也就是NL对应）语言模型。它们的共同特点是大部分用于处理PL（或Source Code）

▲ 赞同 113 | 6 条评论 | 喜欢 | 收藏 | 设置

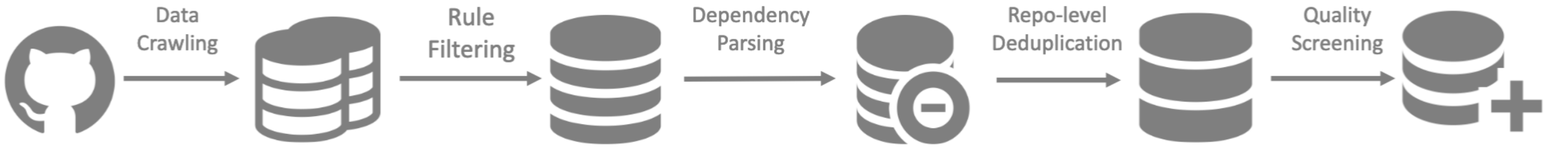
https://zhuanlan.zhihu.com/p/539929943

Large Language Models for Code



Large Language Models for Code

At early stage, training from scratch.



DeepSeek Coder V1

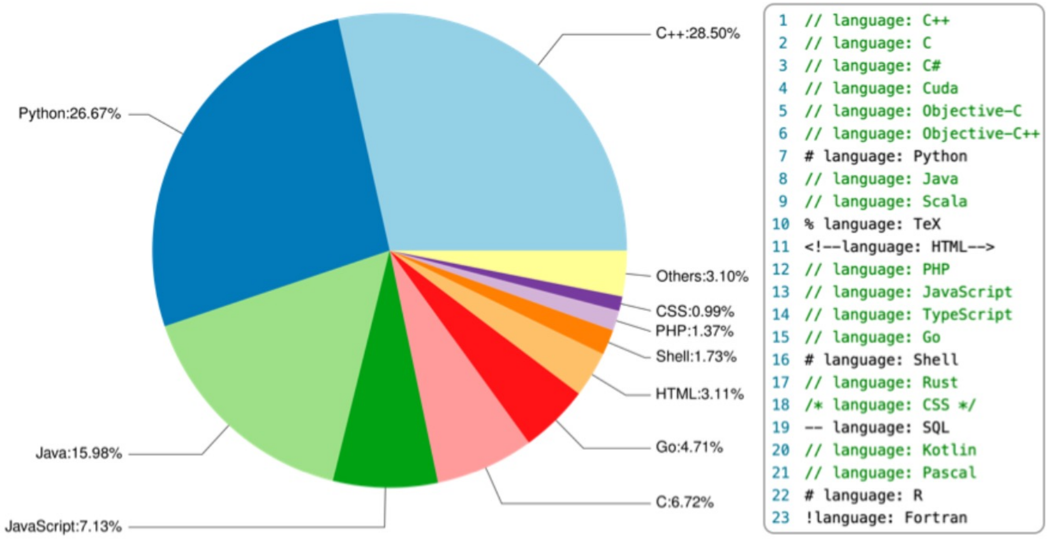
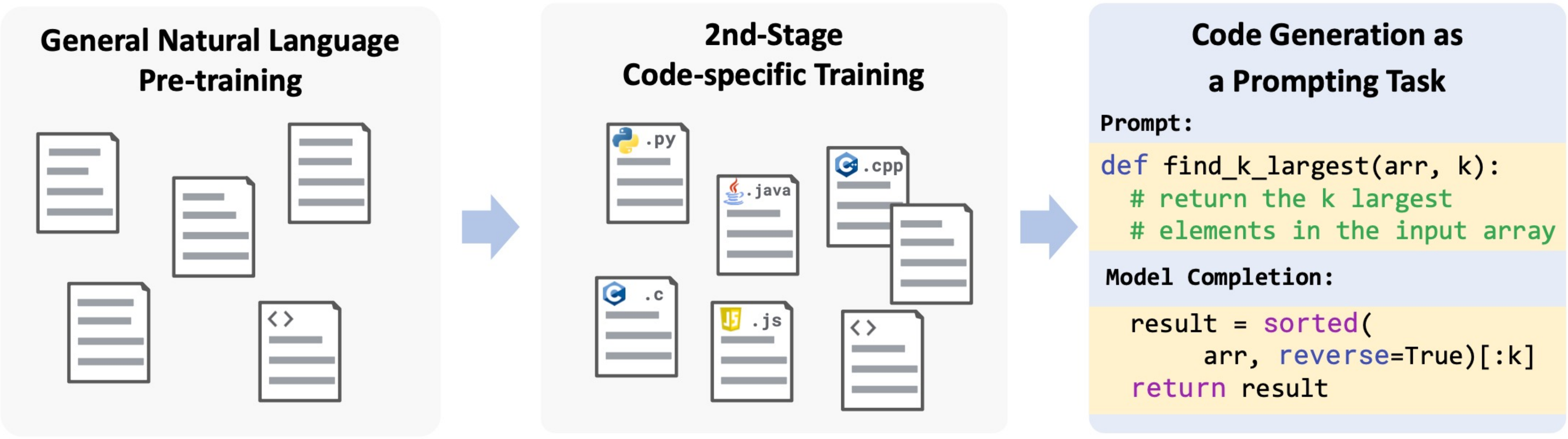


Figure 3: Language distribution and tags of CodeGeeX's data.

CodeGeeX V1

Large Language Models for Code

Gradually…



PaLM (540B Parameters)
50% social media conversations
30% filtered Web documents
5% Github Code (39B tokens)
(780B tokens in total)

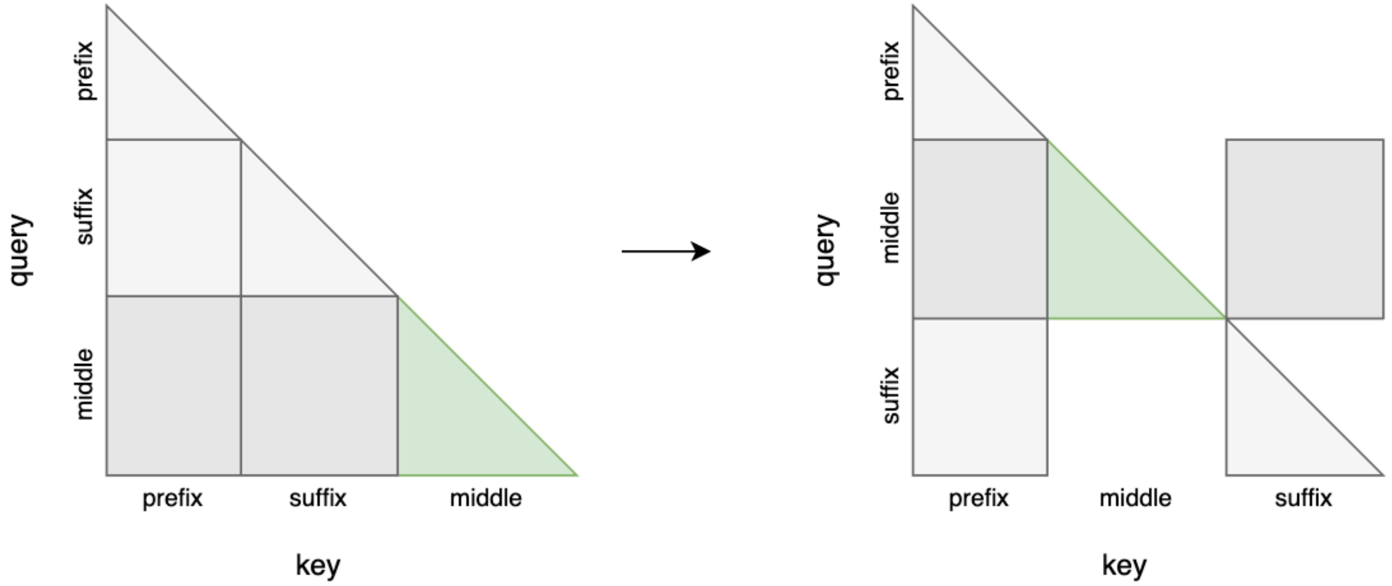
PaLM-Coder (based on PaLM 540B)
Additional 8B multilingual code tokens
(including 5B Python tokens)
Also mix with small % of NL data

Other more-recent Code LLMs:
- Code LLaMA Meta
- DeepSeek Coder

https://ai.googleblog.com/2022/04/pathways-language-model-palm-scaling-to.html

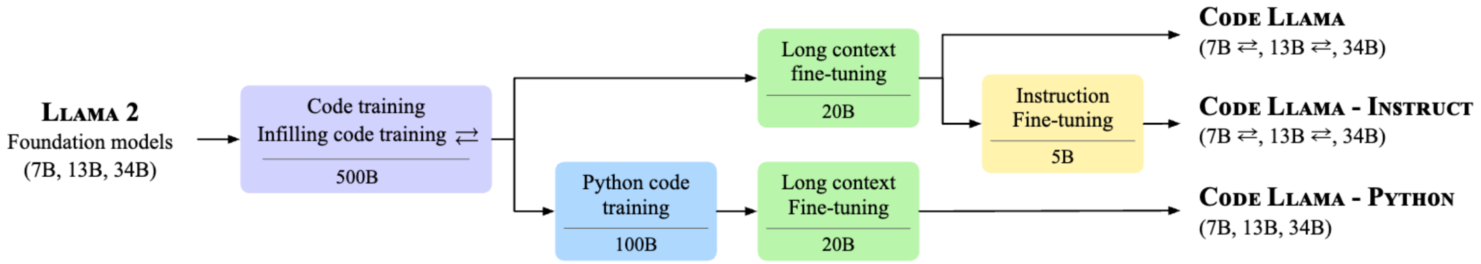
FIM Training for CodeLLMs

- Consistent with AR training, like GPT
- FIM (Fill-in-the-Middle) pretraining

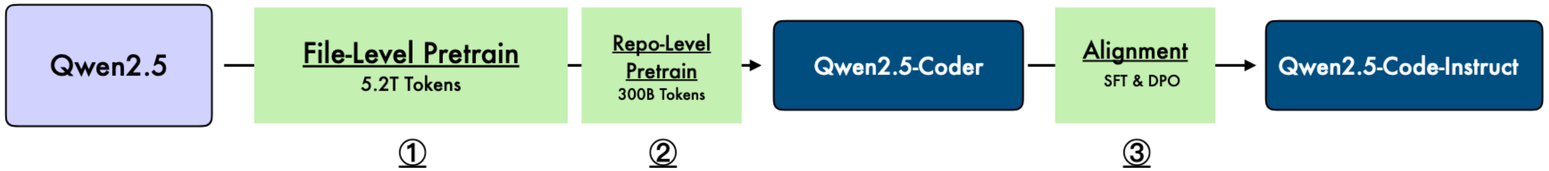


CodeGen2: Lessons for Training LLMs on Programming and Natural Languages
Efficient Training of Language Models to Fill in the Middle, 2022

Large Language Models for Code



CodeLLaMA



Qwen 2.5 Coder

And Codex, PaLM Coder, DeepSeekCoder V2 …

Evaluating Large Language Models Trained on Code
DeepSeek-Coder-V2: Breaking the Barrier of Closed-Source Models in Code Intelligence

Large Language Models for Code

Arch.	Model Name	Size	Base	Vocab.	Context	Training Objs.	Data Scale	Public
Enc-Dec	AlphaCode [275]	284M/1.1B/ 2.8B/8.7B/ 41.1B	-	8.0K	1536+768	MLM+CLM	354B/590B/ 826B/1250B/ 967B	✗
	CodeT5+ [276]	220M/770M/ 2B/6B/16B	CodeGen	50.0K	2048+2048	MSP+CLM+CL	51.5B	✓
Decoder / CodeLLMs	Codex [36]	2.5B/12B	-	50.3K	4K	CLM	100B/159GB	✓
	CodeParrot [288]	125M/1.5B	-	32.8K	1K	CLM	26B/50GB	✓
	PolyCoder [289]	160M/0.4B/2.7B	-	50.3K	2K	CLM	39B/254GB	✓
	CodeGen [277]	350M/2.7B/ 6.1B/16.1B	-	50.0K	2K	CLM	1.2T	✓
	PaLM-Coder [34]	8B/62B/540B	PaLM	256K	2K	CLM	7.75B	✗
	InCoder [94]	1.3B/6.7B	-	50.3K	2K	FIM	52B/159GB	✓
	PanGu-Coder [290]	317M/2.6B	-	42K	1K	CLM+MLM	387B/147GB	✗
	SantaCoder [291]	1.1B	-	49.2K	2K	FIM	236B/268GB	✓
	phi-1 [292]	350M/1.3B	-	50.0K	2K	CLM	7B	✓
	CodeGeeX [293]	13B	-	52.2K	2K	CLM	850B	✓
	CodeGen2 [286]	1B/3.7B/7B/16B	-	50.0K	2K	MLM+CLM	400B	✓
	StarCoder [279]	15.5B	-	49.2K	8K	FIM	1T/815GB	✓
	CodeAlpaca [294]	7B/13B	LLaMA	32.0K	4K	CLM	20K	✓
	WizardCoder [295]	1B/3B/7B/ 13B/15B/34B	StarCoder	32.0K	2K	CLM	78k	✓
	AquilaCode [296]	7B	Aquila	100.0K	2K	CLM	-	✓
	CodeGeeX2 [293]	6B	ChatGLM2	65.0K	8K	CLM	600B	✓
	CodeLLaMA [297]	7B/13B/34B/70B	LLaMA2	32.0K	4K	FIM	500B	✓
	ToRA-Code [298]	7B/13B/34B	CodeLLaMA	32.0K	2K	Min. NLL	223K	✓
	MAmmoTH-Coder [299]	7B/13B/34B	CodeLLaMA	32.0K	2K	CLM	260K	✓
	Code-Qwen [300]	7B/14B	Qwen	152.0K	8K	CLM	90B	✓
	CodeFuse [301]	1.3B/6.5B/ 13B/34B	Multiple	100.9K	4K	CLM	1.6TB	✓
	CodeShell [302]	7B	-	70.1K	8K	CLM	500B	✓
	Lemur [303]	70B	LLaMA2	32.0K	4K	CLM	90B	✓
	DeepSeekCoder [304]	1.3B/5.7B/ 6.7B/33B	-	32.0K	16K	FIM	2T	✓
	Symbol-LLM [305]	7B/13B	LLaMA2	32.0K	4K	CLM	2.25GB	✓
	Stable Code [306]	3B	-	50.3K	16K	FIM	1.3T	✓
	DeciCoder [307]	1B/6B	-	49.2K	2K	FIM	446B	✓
	StarCoder2 [280]	3B/7B/15B	-	49.2K	16K	FIM	900B/3TB	✓
	CodeGemma [308]	2B/7B	Gemma	250K	8K	FIM	1T	✓
	CodeStral [309]	22B	-	32.0K	32k	CLM+FIM	-	✓
	DeepSeekCoderV2 [310]	16B/236B	DeepSeekV2	100K	128K	CLM+FIM	10.2T	✓
	Crystal [311]	7B	-	32K	2K	CLM	1.4T	✓
	Yi-Coder [312]	1.5B/9B	Yi	64K	128K	CLM	2.4T	✓
	OpenCoder [313]	1.5B/8B	-	96.6K	8K	CLM	2.5T	✓

General-purpose Code Generation

HumanEval Doc-string2Code (Chen et al., 2021)

```python
def sum_odd_elements(lst):
  """given non-empty list of integers, return the
sum of all of the odd elements that are in even
positions

  Examples
  solution([5, 8, 7, 1]) ⇒ 12
  solution([3, 3, 3, 3, 3]) ⇒ 9
  solution([30, 13, 24, 321]) ⇒ 0
"""
return sum([
    lst[i] for i in range(0, len(lst))
    if i % 2 == 0 and list[i] % 2 == 1)
```

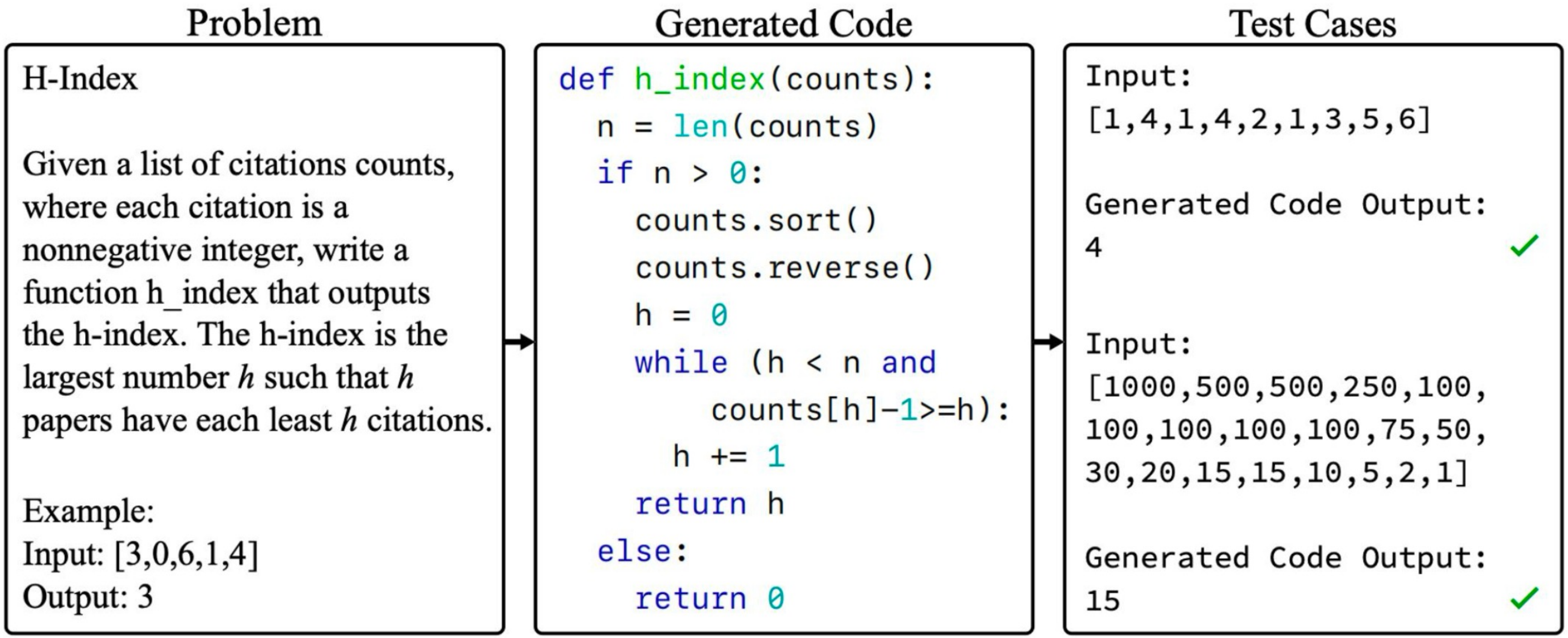
MBPP NL description + tests (Austin et al., 2021)

```python
Write a function to find the smallest missing
element in a sorted array. Your code should
satisfy these tests:

assert smallest_missing(
    [0, 1, 2, 3, 4, 5, 6], 0, 6) == 7
assert smallest_missing(
    [0, 1, 2, 6, 9, 11, 15], 0, 6) == 3
assert smallest_missing(
    [1, 2, 3, 4, 6, 9, 11, 15], 0, 7) == 0

def smallest_missing(arr, n, m):
  smallest = min(n, m)
  for i in range(n, m + 1):
    if arr[i] <= smallest:
      smallest += 1
  return smallest
```

Python Algorithmic Problems

(Example credit: talk by Pengcheng Yin)

Competition Level Programming



An example competition-level coding problem (figure from from Hendrycks et al. 2021)

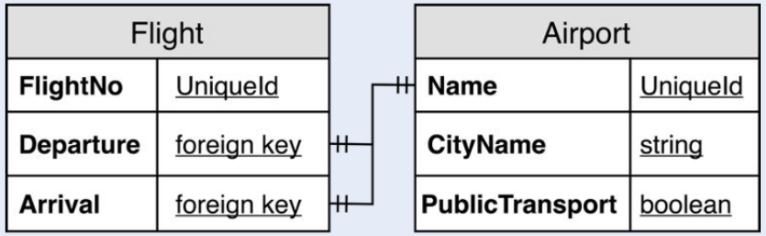
APPS and CodeContests

Code Generation to Domain-Specific Programs

Natural Language Questions with Database Schema

`Input Utterance`

Show me flights from Pittsburgh to SFO



SQL Query

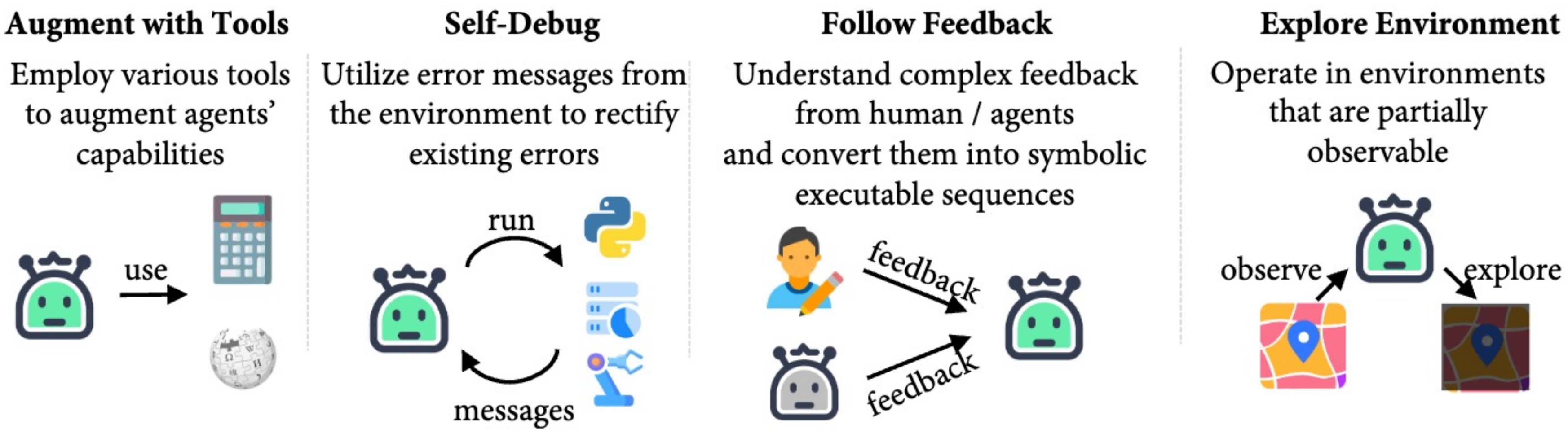
```
SELECT Flight.FlightNo
FROM Flight
JOIN Airport as DepAirport
ON
   Flight.Departure == DepAirport.Name
JOIN Airport as ArvAirport
ON
   Flight.Arrival == ArvAirport.Name
WHERE
    DepAirport.CityName == Pittsburgh
    AND
    ArvAirport.CityName == San_Francisco
```

Text-to-SQL

Large Language Models for Code

The coding ability has obviously become stronger

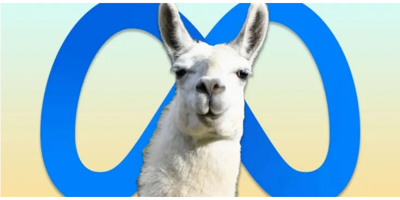
but pure code training often sacrifices other performance aspects, making the model impractical in the real world. For example:



Lemur: Harmonizing Natural Language and Code for Language Agents, ICLR 2024 Spotlight

Balancing Coding and NL

For Agentic Use



LLaMA2

Code-Centric Pre-training → 90B Tokens

Lemur

SFT → 300K Tokens

Lemur-Chat

For Efficiency

NL ← Phase1: SlimPajama

Crystal 7B

Phase2: 63% code from Stack → NL + Code

Phase2: 37% NL

Lemur: Harmonizing Natural Language and Code for Language Agents, ICLR 2024 Spotlight
CRYSTAL: Illuminating LLM Abilities on Language and Code, COLM 2024

CodeLLMs as Base

Early

CodeGen-16B-mono → Chinese Tokens Pre-training → **MOSS**

Recent

DeepSeek-Coder-Base-v1.5 7B → 120B Math Tokens / NL and Code Data → **DeepSeekMath 7B**

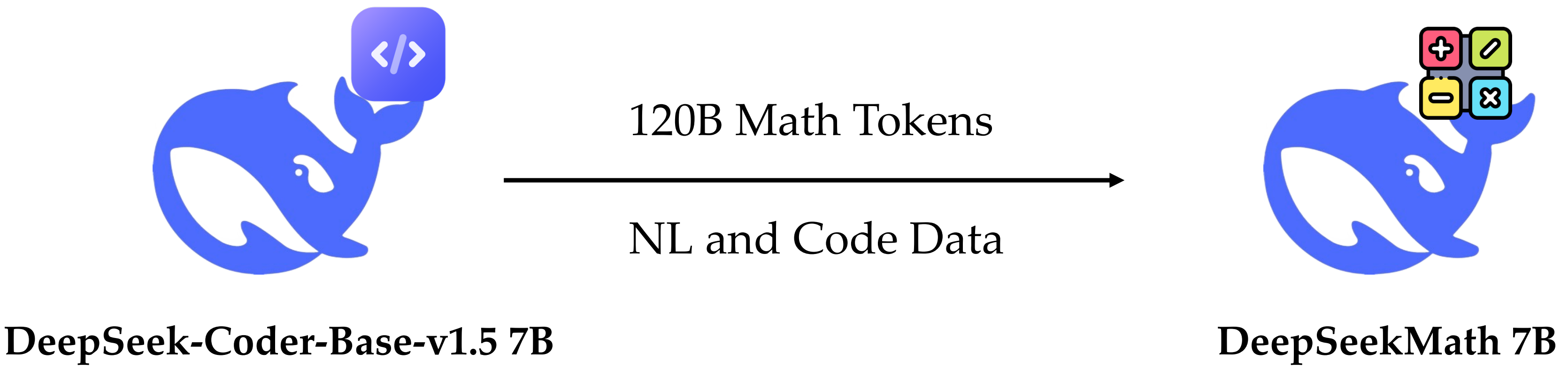
MOSS: An Open Conversational Large Language Model, Machine Intelligence Research
DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models

CodeLLMs as Base

Some findings

1. Code training or pre-training and tasks like math are largely not mutually exclusive; in fact, they often enhance each other.

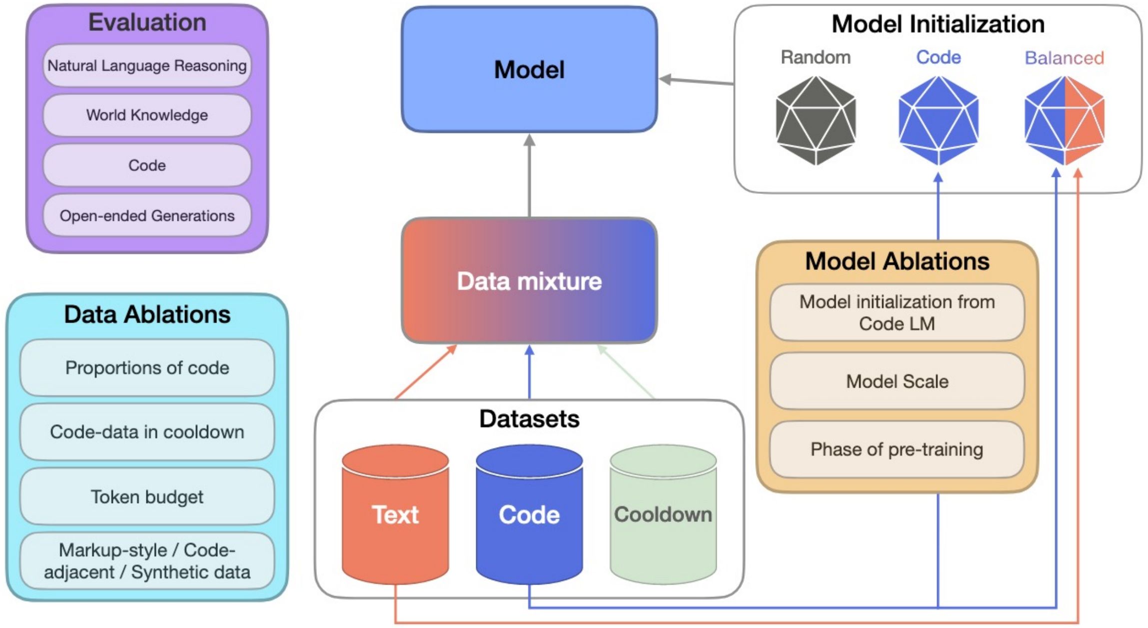
2. How to balance their proportions is very important.



CodeLLMs as Base

More findings, beyond math

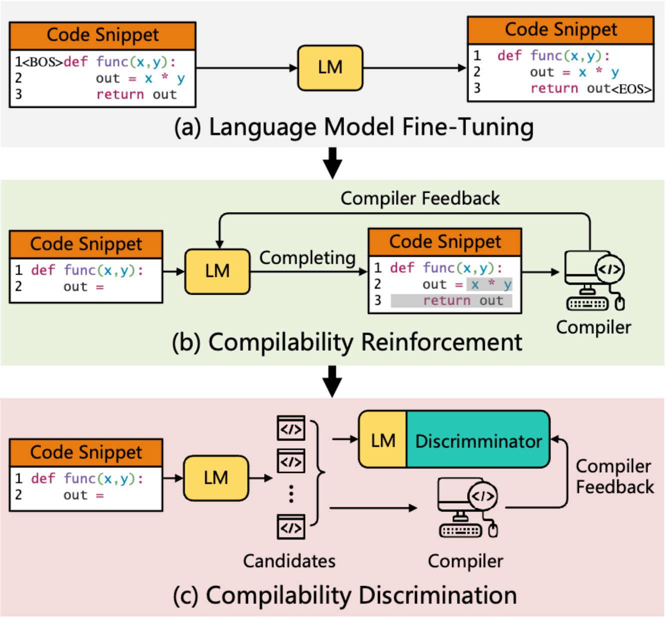
1. Non-code tasks, performance peaks on average when the code proportion is 25%.
2. Excessive code reduces world knowledge
3. Code performance improves linearly as the code proportion increases.

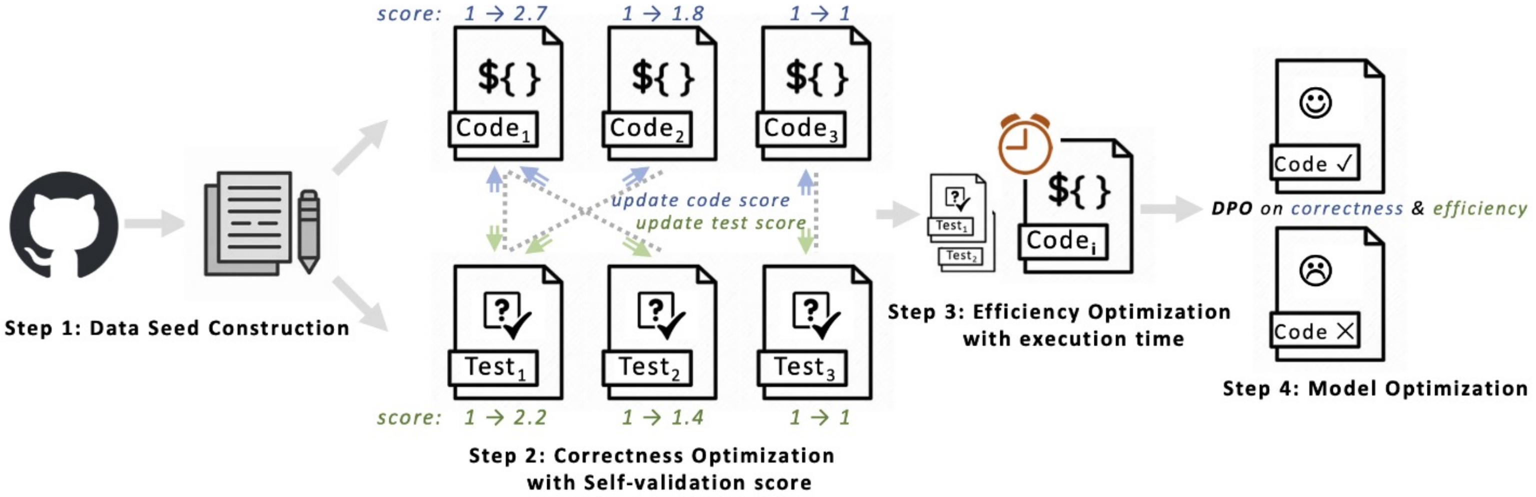


Preference Optimization + Compiler feedback

Insufficient Priority on Correctness: In ambiguous cases, CodeLLMs fail to prioritize the correct solution over an incorrect one.

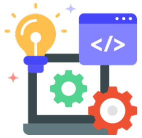
Runtime Efficiency: The generated code, while functionally correct, may have performance issues).





CodeDPO: Aligning Code Models with Self Generated and Verified Source Code
Compilable Neural Code Generation with Compiler Feedback

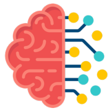
The applications of Code Intelligence



Software Engineering

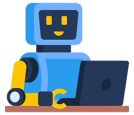
AI4Science

The Application of Code Intelligence



Reasoning

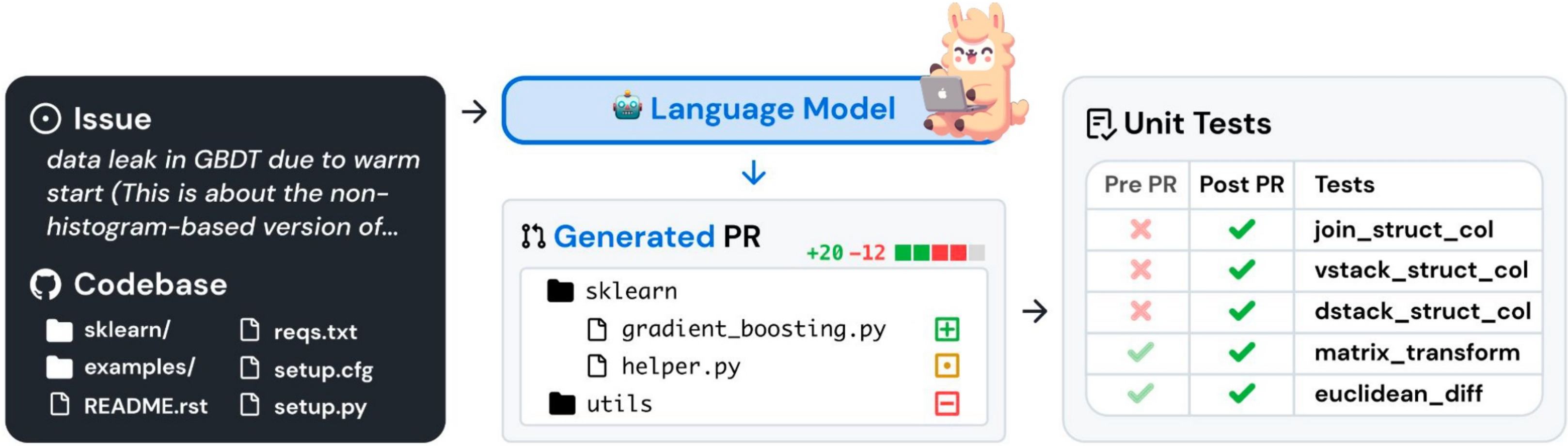
Agents



Applications: Software Engineering

Move beyond simple code generation

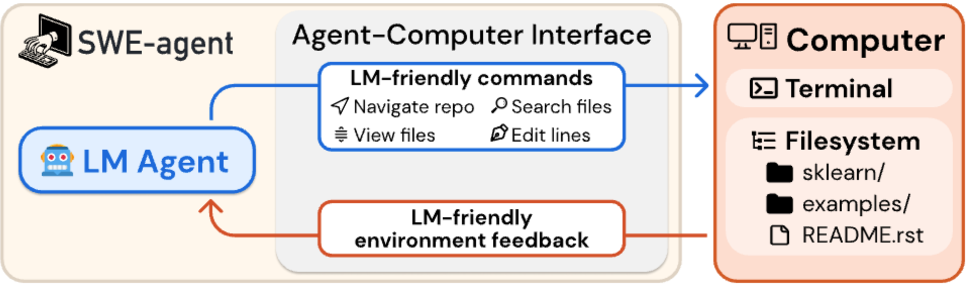
Real PRs from popular Python open-source repositories (e.g., Django, Flask, etc.), ultimately filtering out valid task instances. Each task instance corresponds to a GitHub Issue and its merged solution.



SWE-bench: Can Language Models Resolve Real-World GitHub Issues?, ICLR 2024

SWE-agent: Agent-Computer Interfaces Enable Automated Software Engineering, NIPS 2024

Applications: Software Engineering



An LM interacting with a computer through an agent-computer interface

SWE-agent lets your LM autonomously use tools to:

1. Fix issues in real GitHub repositories,
2. perform tasks on the web,
3. find cybersecurity vulnerabilities (by solving Capture The Flag challenges),
4. Custom Tasks

SWE-bench: Can Language Models Resolve Real-World GitHub Issues?, ICLR 2024

SWE-agent: Agent-Computer Interfaces Enable Automated Software Engineering, NIPS 2024

Applications: Software Engineering

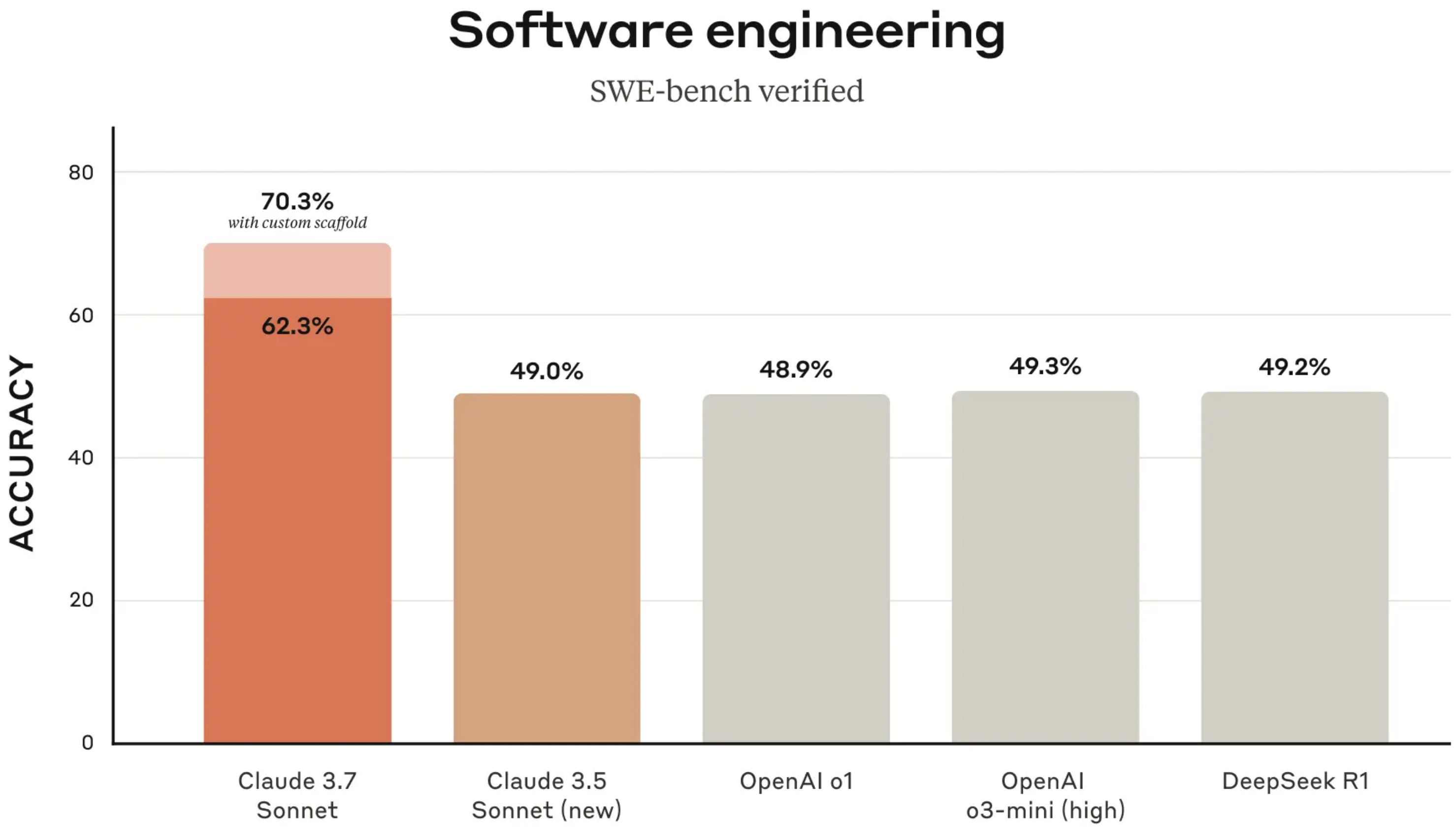
We are witnessing "Full-Stack" SE platforms

1. Modify code
2. Run commands
3. Browse the web
4. Call APIs
5. Even copy code snippets from StackOverflow.

OpenDevin / OpenHands

Applications: Software Engineering

More advanced models + frameworks Are "dominating" these benchmarks.



1. Claude 3.7 Sonnet
2. Scaffolding

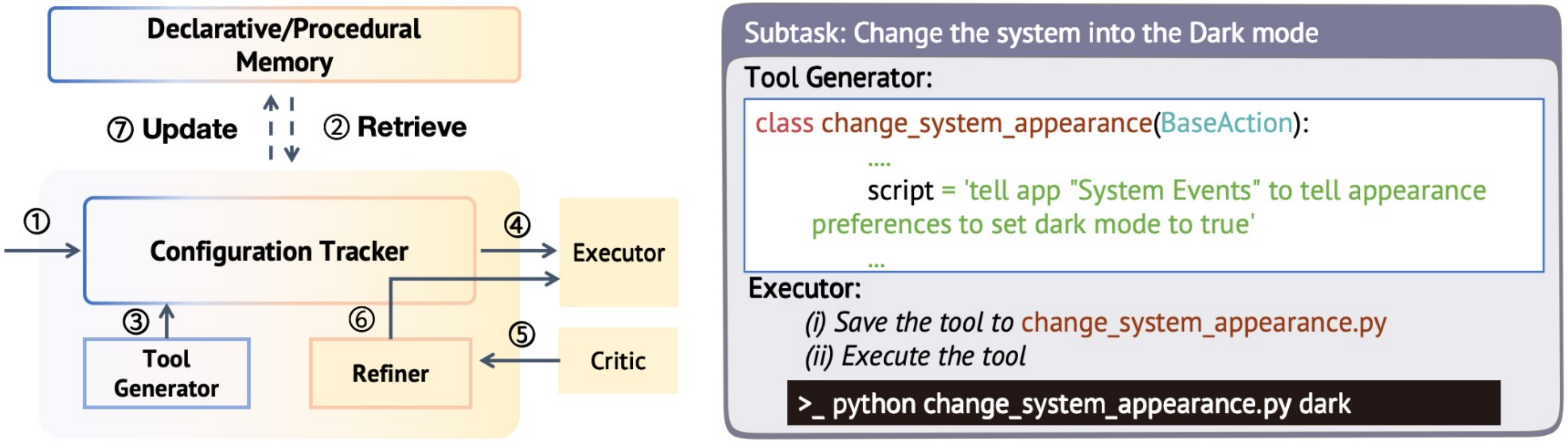
Agentic coding

Claude 3.7 Sonnet and Claude Code, 25 Feb, 2025

SWE-bench: Can Language Models Resolve Real-World GitHub Issues?

Applications: Agents

OS-Copilot: -> Code-based computer agents Framework



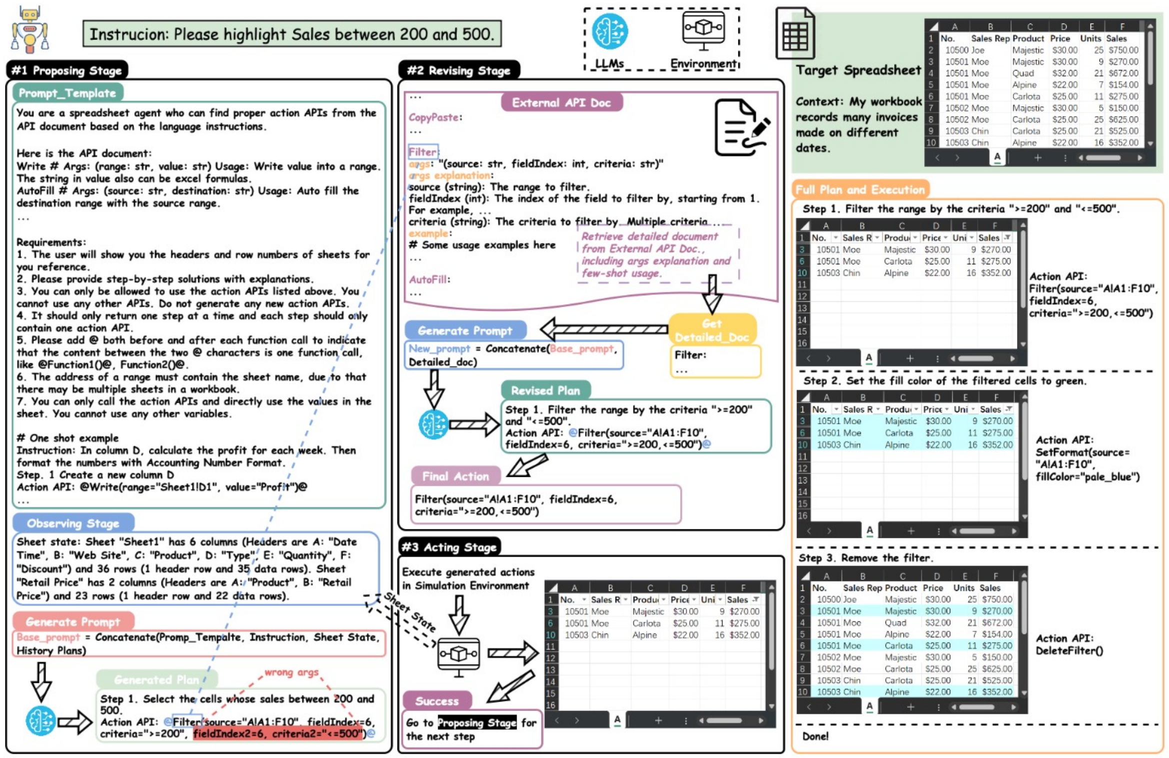
(a) Configurator (b) A running example

Applications: Agents

What kind of issues these agents can solve?

1. API-interface available
2. CLI, like “Apple Script”
3. Numerical Calculataions

Generate code + Invoke API -> Solve computer task



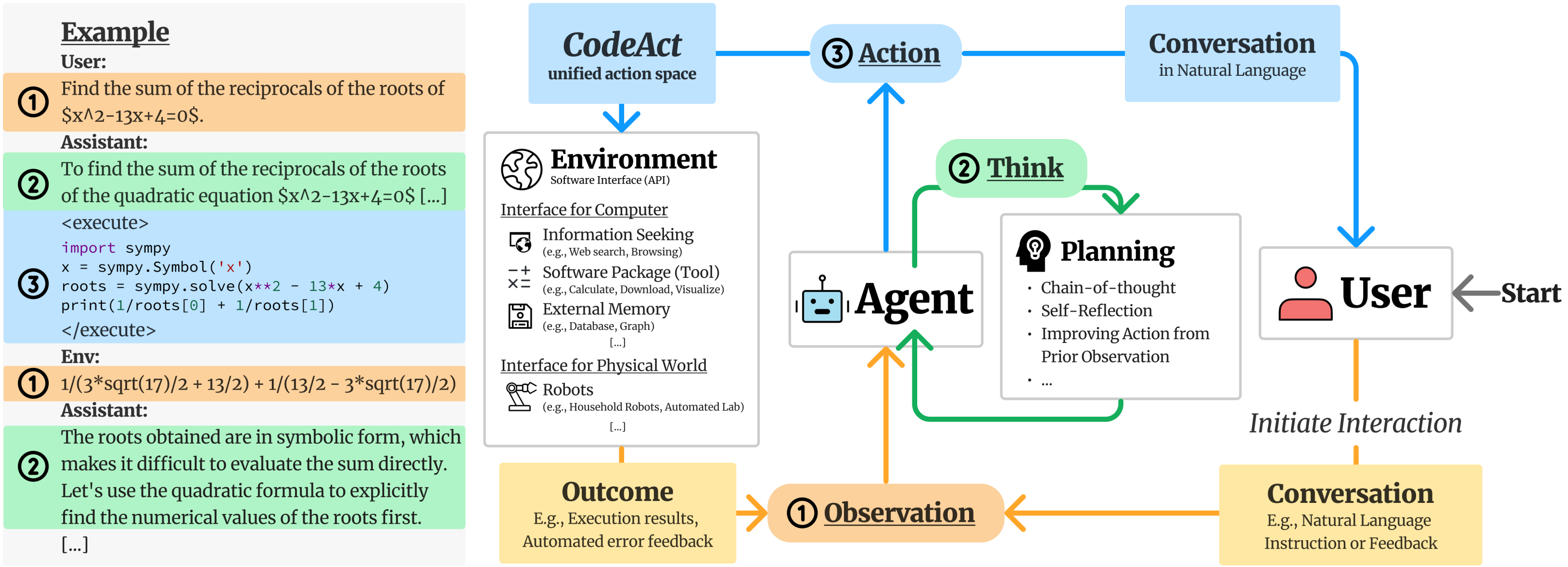
A case in SheetCopilot

OS-Copilot: Towards Generalist Computer Agents with Self-Improvement, LLM Agents Workshop @ ICLR 2024
SheetCopilot: Bringing Software Productivity to the Next Level through Large Language Models, NIPS 2023

Applications: Agents

Use executable code to consolidate LLM agents' actions into a unified action space

Integrated with a Python interpreter, execute code actions and dynamically revise prior actions or emit new actions upon new observations (e.g., code execution results) through multi-turn interactions

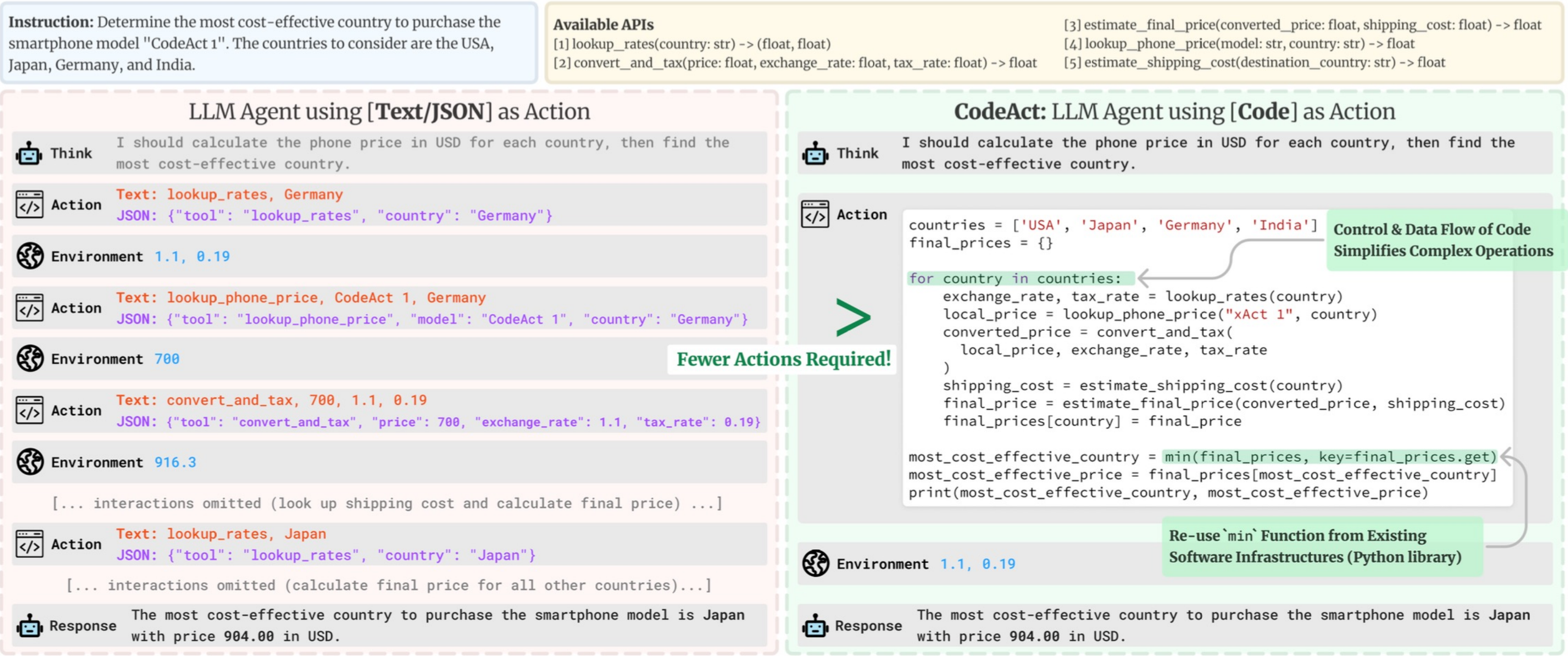


CodeACT

Executable Code Actions Elicit Better LLM Agents, LLM Agents Workshop @ ICLR 2024, Oral / ICML 2024

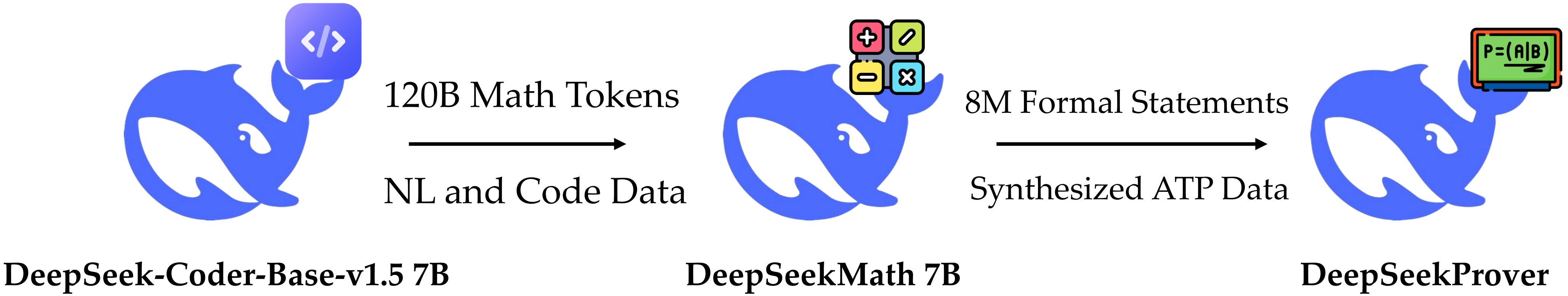
Applications: Agents

In short: Fewer actions, better efficiency



Applications: AI4Science

Models



Methods

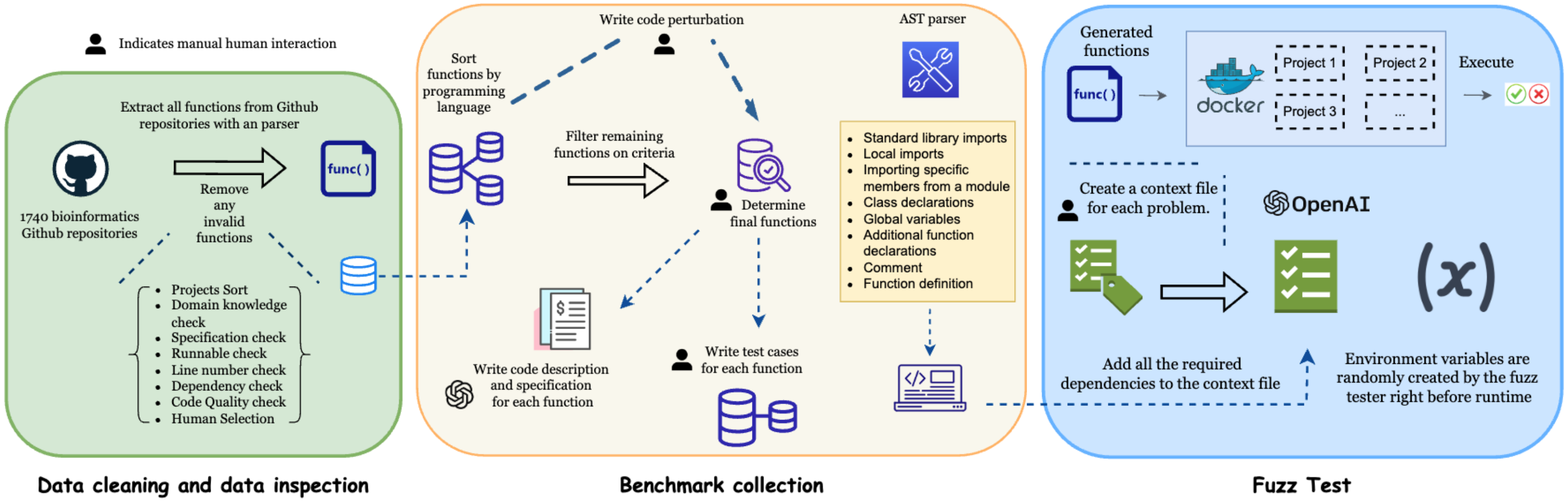
1. Biochemistry Discoveries.
2. Chemical programming languages -> automate the synthesis of chemical compounds.

DeepSeek-Prover: Advancing Theorem Proving in LLMs through Large-Scale Synthetic Data, 2024

Applications: AI4Science

Biochemistry Discoveries.

Processing Bioinformatics-related GitHub repositories



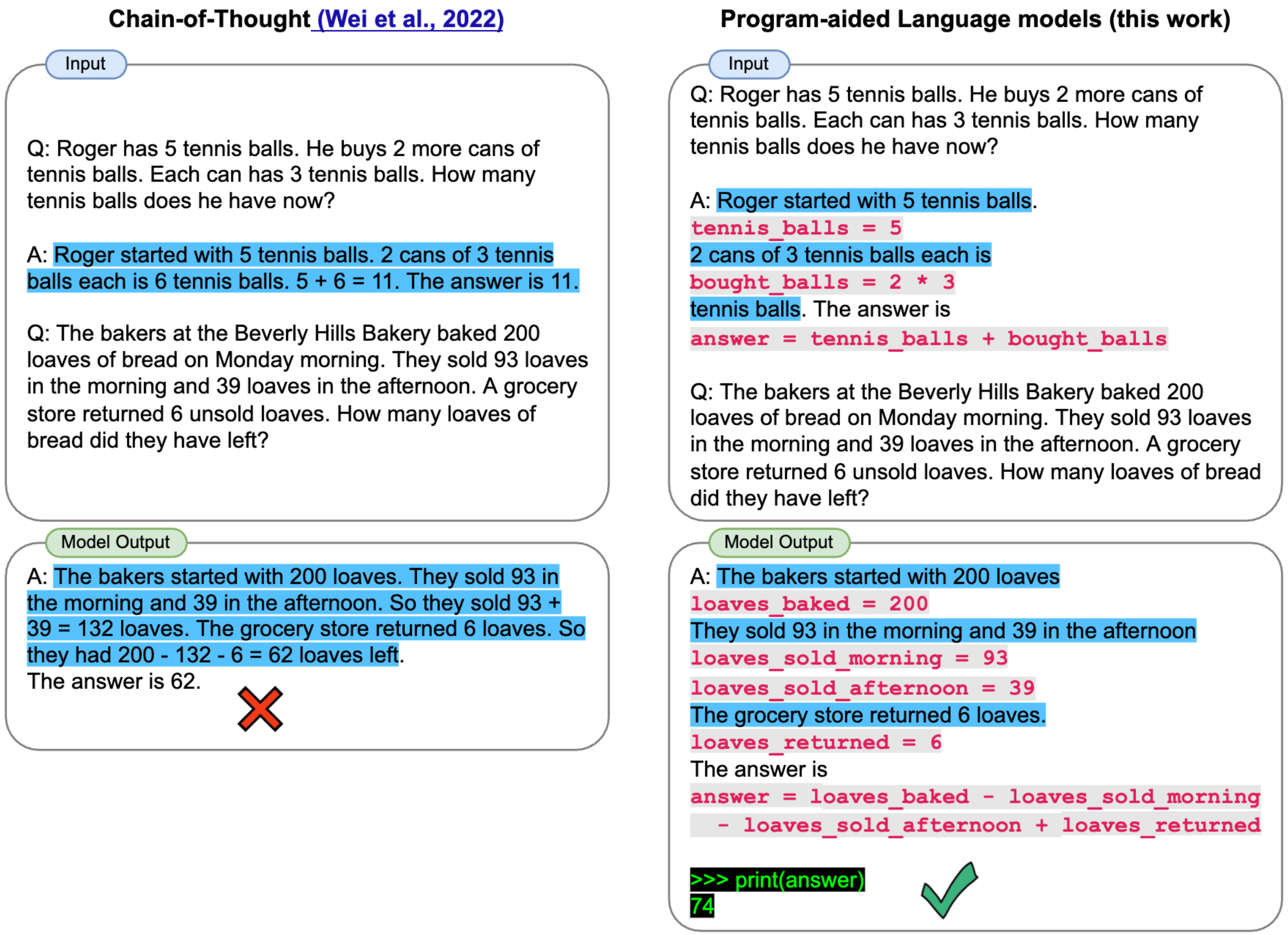
BioCoder: A Benchmark for Bioinformatics Code Generation with Large Language Models, Bioinformatics 2024

Applications: Reasoning

Program-aided language models

Coding instead of CoT

Offload “Computation” to a python interpreter



PAL: Program-aided Language Models, ICML 2023

Program of Thoughts Prompting: Disentangling Computation from Reasoning for Numerical Reasoning Tasks, TMLR (Oct, 2023)

Applications: Reasoning

Lot of techniques can be

1. Directly transferred to this scenario

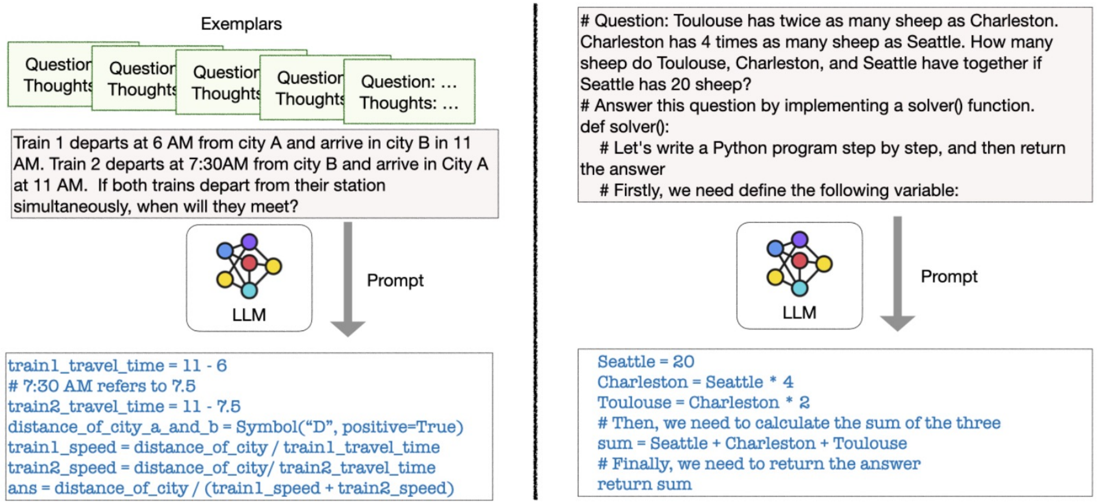
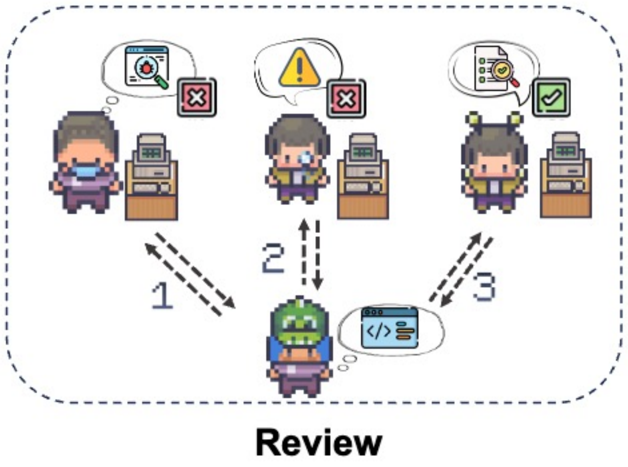


Figure 3: Left: Few-shot PoT prompting, Right: Zero-shot PoT prompting.

2. Combine together



LLMs like DeepSeekMath-Instruct leverage such math x code data in training

Program of Thoughts Prompting: Disentangling Computation from Reasoning for Numerical Reasoning Tasks, TMLR (Oct, 2023)
Corex: Pushing the Boundaries of Complex Reasoning through Multi-Model Collaboration, COLM 2024

Resources

🔗 Paper Collections / Tutorials 📚

- Language Models for Code 🤖
- Evaluations and Benchmarks 📊
- Preference Optimization 🍎
- Code Repair 🔧
- Reasoning with Code Synthesis 🧠
- Data Science 🔢
- Corpus containing Code Data 📚
- Code-Based Solutions for NLP Tasks 📝
- Code Empowered Agents 🤖
- Reinforcement Learning with CodeLMs 🎮
- Code Intelligence assists AI4Science 🧪
- Software Development 🛠
- Multilingual 🌍
- Multimodal Code Generation 🎨
- Awesome Slides, Talks and Blogs 👨‍🏫

Recent Work on Code Intelligence (Welcome PR) 📗

- CodeI/O: Condensing Reasoning Patterns via Code Input-Output Prediction 2025.2
- Competitive Programming with Large Reasoning Models 2025.2
- EpiCoder: Encompassing Diversity and Complexity in Code Generation 2025.1
- WarriorCoder: Learning from Expert Battles to Augment Code Large Language Models 2024.12
- FullStack Bench: Evaluating LLMs as Full Stack Coders 2024.12
- CodeDPO: Aligning Code Models with Self Generated and Verified Source Code 2024.11
- OpenCoder: The Open Cookbook for Top-Tier Code Large Language Models 2024.11
- Qwen2.5-Coder Series: Powerful, Diverse, Practical. 2024.11

https://github.com/QiushiSun/Awesome-Code-Intelligence

Thanks for listening

Contact: qiushisun@connect.hku.hk