DEEP LEARNING FOR CODE UNDERSTANDING AND GENERATION

CHALLENGES & OPPORTUNITIES

Chandan K. Reddy

Dept. of Computer Science

Virginia Tech

http://www.cs.vt.edu/~reddy

Virginia Tech 1872 *Invent the Future*

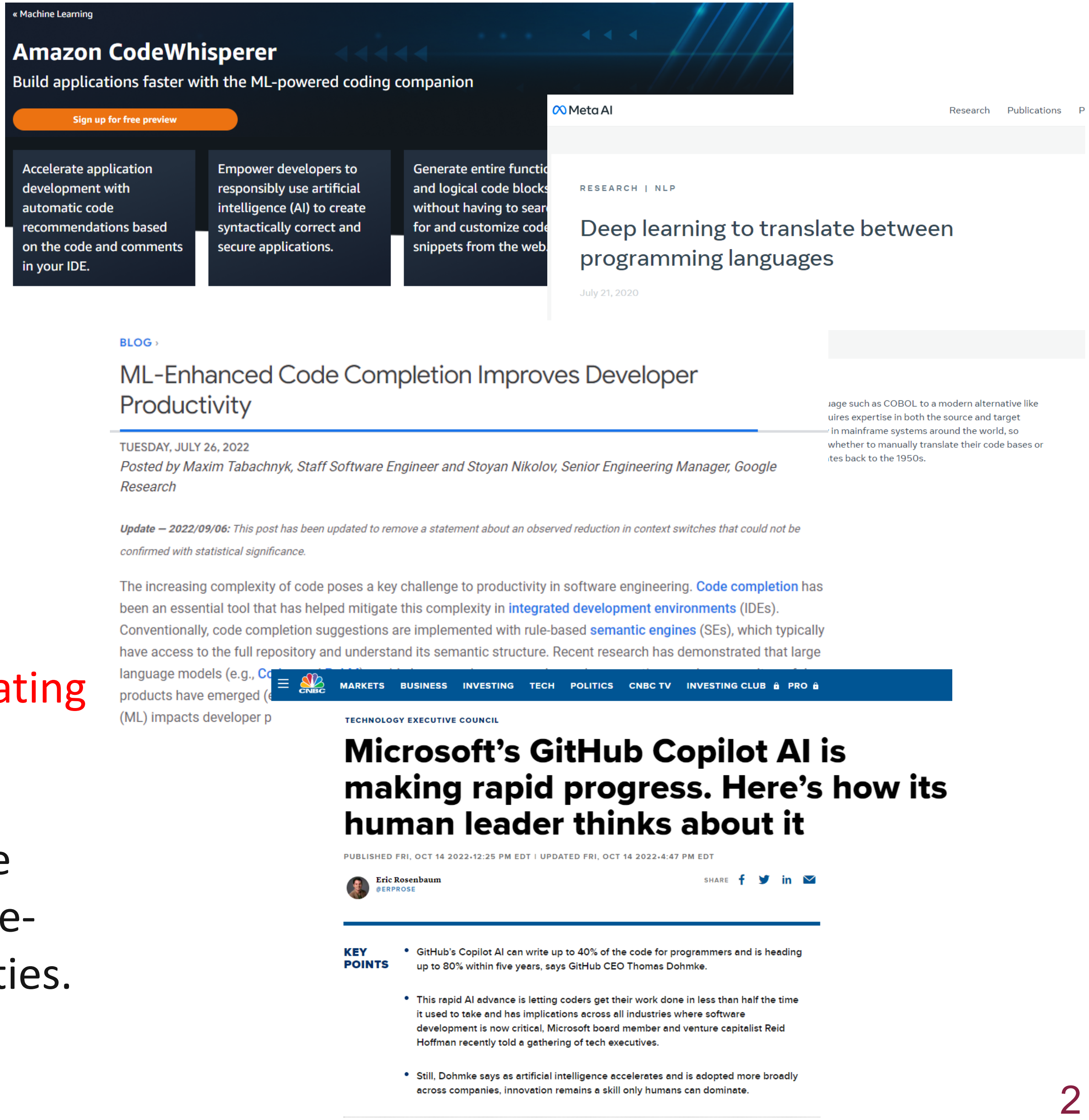
AI for Software Engineering

Common tasks in Software Engineering:

- Write code from a specification
- Translate code
- Fixing bugs

Can AI help in supporting humans and/or automating some of these tasks?

- Yes! The availability of large open-source code repositories and the feasibility of training large-scale deep models, provides exciting possibilities.



AI Assisted Code Tasks

https://github.com/microsoft/CodeXGLUE

Category	Task	Dataset Name	Language	Train/Dev/Test Size	Baselines	Task definition
Code-Code	Clone Detection	BigCloneBench	Java	900K/416K/416K	CodeBERT	Predict semantic equivalence for a pair of codes.
		POJ-104	C/C++	32K/8K/12K		Retrieve semantically similar codes.
	Defect Detection	Devign	C	21k/2.7k/2.7k		Identify whether a function is vulnerable.
	Cloze Test	CT-all	Python, Java, PHP, JavaScript, Ruby, Go	-/-/176k		Tokens to be predicted come from the entire vocab.
		CT-max/min	Python, Java, PHP, JavaScript, Ruby, Go	-/-/2.6k		Tokens to be predicted come from {max, min}.
	Code Completion	PY150	Python	100k/5k/50k	CodeGPT	Predict following tokens given contexts of codes.
		GitHub Java Corpus	Java	13k/7k/8k		
	Code Repair	Bugs2Fix	Java	98K/12K/12K	Encoder-Decoder	Automatically refine codes by fixing bugs.
	Code Translation	CodeTrans	Java-C#	10K/0.5K/1K		Translate the codes from one programming language to another programming language.
Text-Code	NL Code Search	CodeSearchNet, AdvTest	Python	251K/9.6K/19K	CodeBERT	Given a natural language query as input, find semantically similar codes.
		CodeSearchNet, WebQueryTest	Python	251K/9.6K/1k		Given a pair of natural language and code, predict whether they are relevant or not.
	Text-to-Code Generation	CONCODE	Java	100K/2K/2K	CodeGPT	Given a natural language docstring/comment as input, generate a code.
Code-Text	Code Summarization	CodeSearchNet	Python, Java, PHP, JavaScript, Ruby, Go	908K/45K/53K	Encoder-Decoder	Given a code, generate its natural language docstring/comment.
Text-Text	Documentation Translation	Microsoft Docs	English-Latvian/Danish/Norwegian/Chinese	156K/4K/4K		Translate code documentation between human languages (e.g. En-Zh), intended to test low-resource multi-lingual translation.

A benchmark for multiple code understanding and generation tasks.

StructCoder on the CodeXGLUE leaderboard!

https://microsoft.github.io/CodeXGLUE/

Code Translation (Code-Code)

				Java to C#			C# to Java		
Rank	Model	Organization	Date	BLEU	Acc(%)	CodeBLEU	BLEU	Acc(%)	CodeBLEU
1	StructCoder	Virginia Tech	2022-06-02	85.02	66.60	88.42	80.66	67.70	86.03
2	PLNMT-sys0	Microsoft DevDiv...	2022-05-09	83.37	64.60	87.38	80.91	66.80	85.87
3	PLBART	UCLA & Columbi...	2021-04-02	83.02	64.60	87.92	78.35	65.00	85.27
4	CodePALM	Microsoft DevDiv...	2021-08-27	83.26	65.50	86.37	78.94	65.20	83.74
5	CodeBERT	CodeXGLUE Team	2020-08-30	79.92	59.00	85.10	72.14	58.80	79.41
6	RoBERTa(code)								

Code Generation (Text-Code)

				Text2Code Generation		
Rank	Model	Organization	Date	EM	BLEU	CodeBLEU
1	StructCoder	Virginia Tech	2022-05-30	22.35	40.91	44.76
2	JaCoText	Novelis.io	2021-12-07	22.15	39.07	41.53
3	CoTexT	Case Western R...	2021-04-23	20.1	37.4	40.14
4	Text2Java-T5	Novelis.io	2021-09-29	21.45	37.46	39.94
5	PLBART	UCLA & Columbi...	2021-04-02	18.75	36.69	38.52
6	CodeGPT-adapted	CodeXGLUE Team	2020-08-30	20.1	32.79	35.98

Code Generation

Code generation is the problem of generating code given a source code that is either imperfect or in a different language, or generating code from a natural language description.

Given that the goal here is to read a sequence and generate a sequence, several NLP techniques have been proposed to solve this problem.

```
#include <stdio.h>

int add1 ( int a ) {
  int s = a + 1 ;
  return s ;
}

int main () {
  int s = add1(8);
  printf (“%d”, s);
}
```

```
Write a function add1()
to increment a number,
and test it with
add1(8).
```

Translate from C to Python

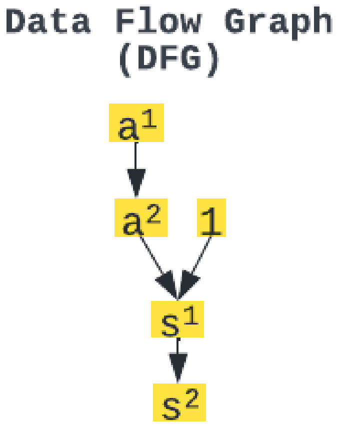
Generate python code from description.

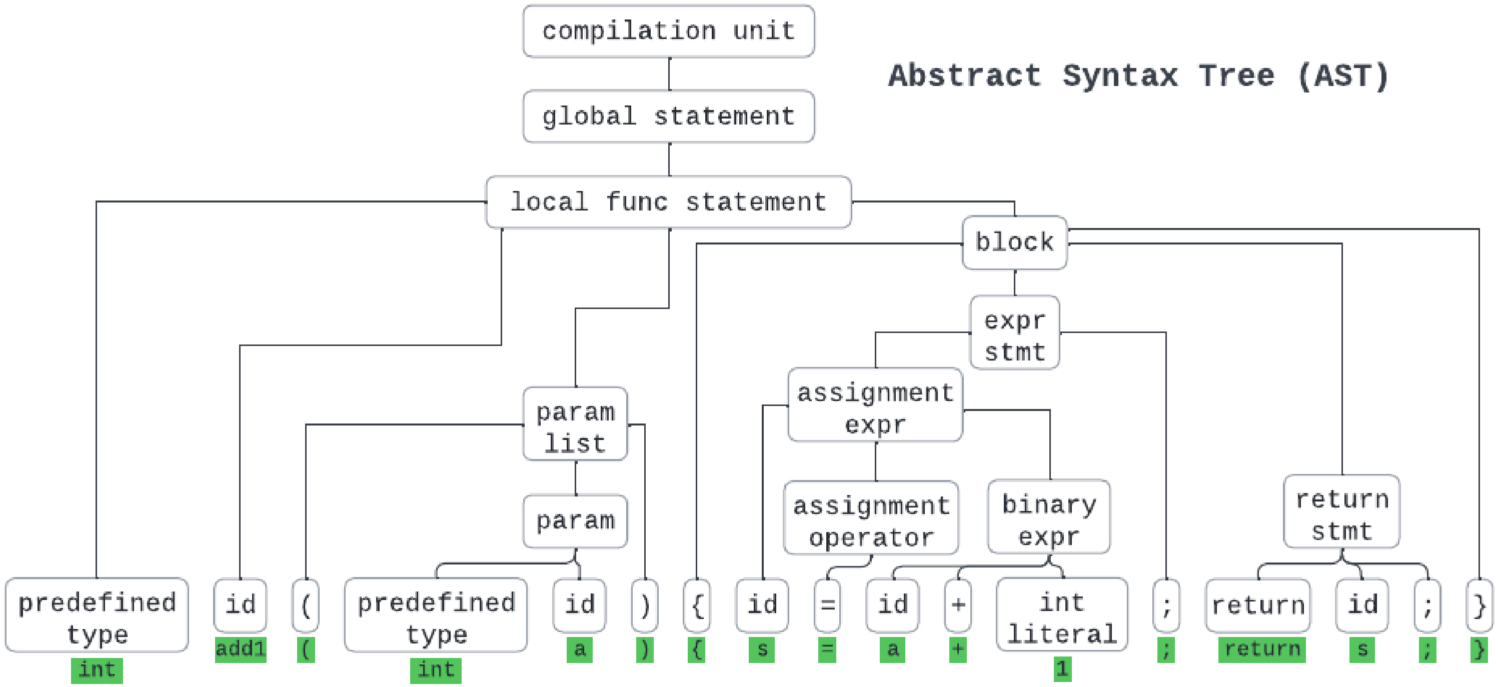
```
def add1(a):
    s = a + 1
    return s

s = add1(8)
print (s)
```

Code is not Just a Sequence of Tokens !

- Can we improve syntactic and semantic correctness of generated codes?
- Can we encourage the model to preserve target code structure?
    - StructCoder does this using target AST and DFG preserving auxiliary tasks.

```
int add1 ( int a ) {
    s = a + 1 ;
    return s ;
}
```





An **AST** is a tree-like structure used to represent the syntactic structure of a program. It is a graph representation of source code primarily used by compilers to read code and generate the target binaries.

DFG shows the data flow among variables in the code.

Existing Approaches

Model	Encoder-only pretraining	Encoder-Decoder pretraining	Encoder structure-awareness	Decoder structure-awareness
CodeGPT				
CodeBERT	MLM, RTD	-	-	-
GraphCodeBERT	MLM, EP, NA	-	DFG	-
Transcoder	MLM	DAE, BT	-	-
PLBART	-	DAE	-	-
DOBF	-	DOBF	-	-
CodeT5	IT	MSP, MIP, NL-PL dual generation	Identifiers	Identifiers
StructCoder (ours)		structure-based DAE, NL-PL dual generation	AST, DFG	AST, DFG

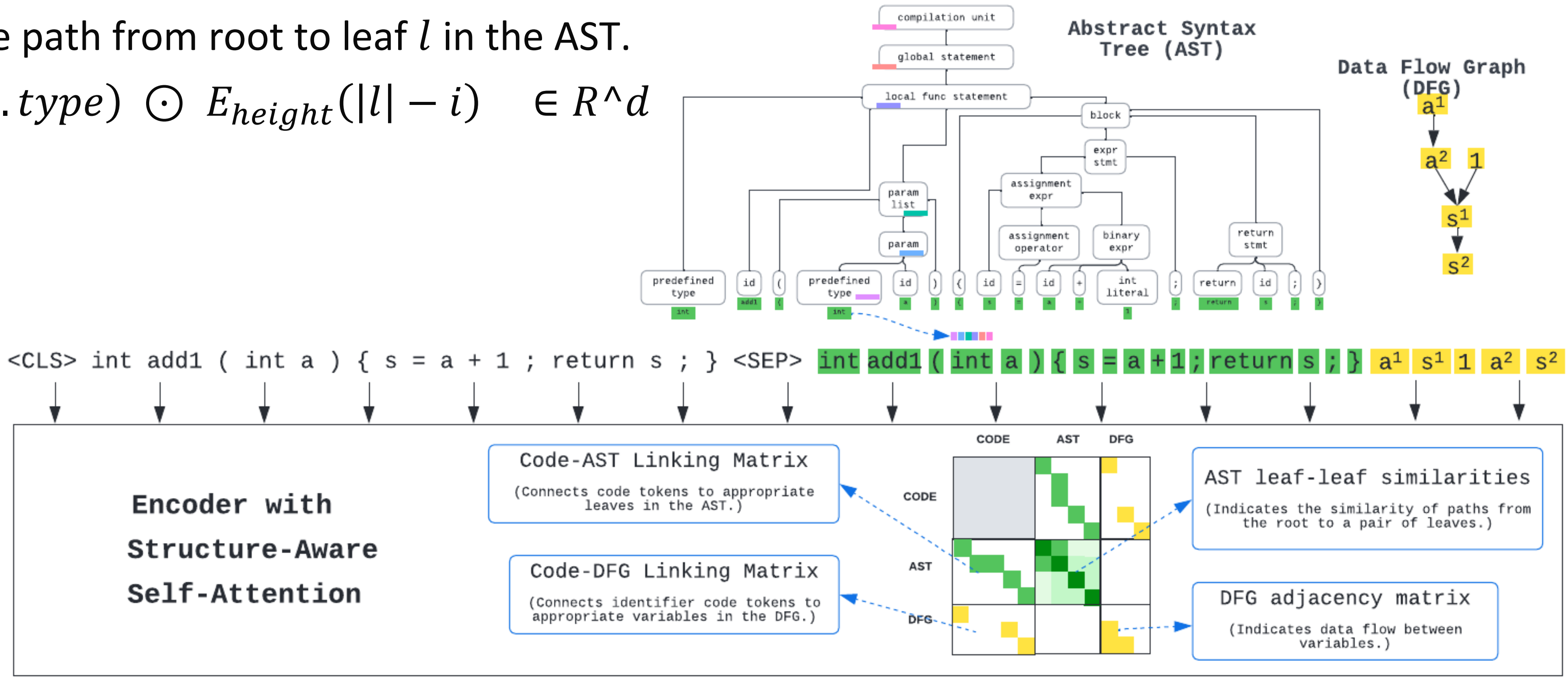
Table: A summary of the recent pre-trained models for code generation. (Abbreviations: DFG: Data Flow Graph, MLM: Masked Language Modeling, DAE: Denoising Autoencoding, RTD: Replaced Token Detection, BT: Back Translation, EP: DFG Edge Prediction, NA: Alignment prediction between code tokens and DFG nodes, DOBF: Deobfuscation, IT: Identifier Tagging, MSP: Masked Span Prediction, MIP: Masked Identifier Prediction.)

- Unlike existing models, StructCoder models code structure in both encoder and decoder by incorporating both AST and DFG.
- Though some existing works modeled AST or DFG in the encoder, ***none of the state-of-the-art pretrained code models utilize code structure in the decoder***, *which is crucial for code generation.*

S. Tipirneni, M. Zhu, and C. K. Reddy. "StructCoder: Structure-Aware Transformer for Code Generation." arXiv 2022.

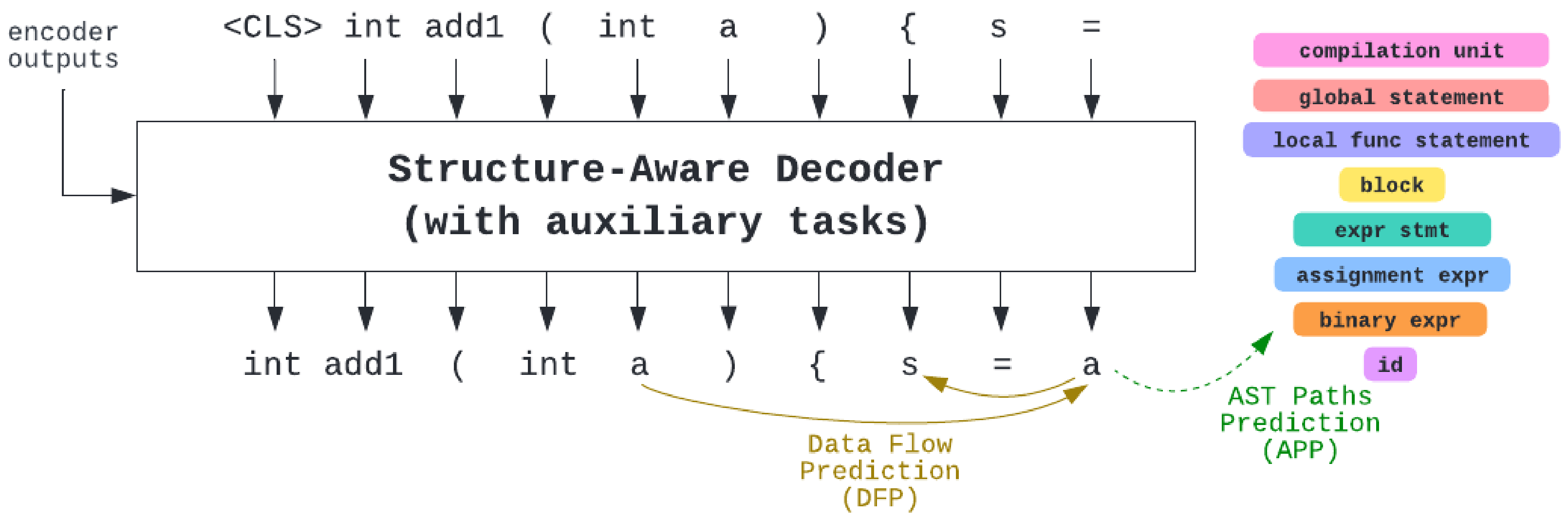
StructCoder - Encoder

- Input tokens contain AST leaves and DFG variables in addition to source code.
- Embedding AST leaves:
  - Let $(r_1, \ldots, r_{|l|})$ be the path from root to leaf l in the AST.
  - $E(l) = \sum_{i=1}^{|l|} E_{type}(r_i.type) \odot E_{height}(|l| - i) \quad \in R\text{^}d$



Incorporating such structural information can be model-agnostic, i.e., we can choose our favorite encoder-decoder model (such as a SOTA CodeT5 model)

StructCoder - Decoder



Along with predicting the next token, the decoder also performs these auxiliary tasks:

1. Data Flow Prediction: predict the DFG edges incident on this token.
2. AST Paths Prediction: predict the node types on the root-leaf path to the leaf containing this token in the AST.

Hypothesis: The auxiliary tasks encourage the decoder to generate correct code. In this example, if the decoder performs auxiliary tasks correctly, it knows that the next token is an identifier that gets its value from the function argument ‘a’ and provides its value to the variable `s’.

Results on Code Translation

	Java-C#			C#-Java		
	BLEU	xMatch	CodeBLEU	BLEU	xMatch	CodeBLEU
Naive Copy	18.54	0.00	42.20	18.69	0.00	34.94
Transformer	55.84	33.00	63.74	50.47	37.90	61.59
RoBERTa (code)	77.46	56.10	83.07	71.99	57.90	80.18
CodeBERT	79.92	59.00	85.10	72.14	58.80	79.41
GraphCodeBERT	80.58	59.40	-	72.64	58.80	-
PLBART	83.02	64.60	87.92	78.35	65.00	85.27
CodeT5*	83.88	64.70	87.38	79.71	67.50	85.51
StructCoder	**85.03**	**66.60**	**88.41**	**80.73**	**67.70**	**86.10**

Results on code translation tasks from CodeXGLUE benchmark.

Ablation Study

Enabled	xMatch		BLEU		Weighted BLEU		AST match		Data Flow match		CodeBLEU	
	J-C	C-J	J-C	C-J	J-C	C-J	J-C	C-J	J-C	C-J	J-C	C-J
No structure (baseline)	43.90	40.20	62.30	53.20	63.60	54.56	78.82	75.40	73.79	64.20	69.62	61.84
DFG (i/p)	47.20	27.10	65.59	41.64	66.72	43.20	80.04	70.19	75.66	58.63	72.00	53.41
DFG (o/p)	48.10	43.10	64.87	56.64	66.12	57.90	79.88	77.24	75.26	66.52	71.53	64.57
AST (i/p)	51.10	45.90	69.92	59.25	70.93	60.30	82.89	79.12	77.97	68.31	75.42	66.74
AST (o/p)	46.00	49.50	64.16	63.70	65.34	64.79	80.02	81.84	75.45	72.89	71.24	70.80
DFG (i/p,o/p), AST (i/p,o/p)	51.20	51.20	70.86	66.12	71.82	66.99	83.87	83.79	79.41	74.30	76.49	72.80
DFG(i/p,o/p), AST(i/p,o/p), & structure-based DAE pt	**53.80**	**55.10**	**76.86**	**73.53**	**78.07**	**74.41**	**87.07**	**87.30**	**85.00**	**83.80**	**81.75**	**79.76**

Results on Java-C# (J-C) and C#-Java (C-J) translation by adding the proposed structure-based components to a smaller T5 model. The best results are in bold and the second best are underlined. ('i/p' and 'o/p' indicate whether the structure was included in the encoder and decoder, respectively.)

Case Study – Java-C# Translation

CodeT5	StructCoder

CodeT5

```
public override bool Eat(Row @in, int remap){
     int sum = 0;
     for (IEnumerator<Cell> i =
@in.cells.Values.GetEnumerator(); i.MoveNext();){
          sum += c.cnt;
          if (remap[c.@ref] == 0){
               c.@ref = -1;
          }
     }
     }
     int frame = sum / 10;
     bool live = false;
     for (IEnumerator<Cell> i =
@in.cells.GetEnumerator(); i.MoveNext();){
          if (c.cnt < frame && c.cmd >= 0){
               c.cnt = 0; c.cmd = -1;
          }
          if (c.cmd >= 0 || c.@ref >= 0){
               live |= true;
          }
     }
     return !live;
}
```

StructCoder

```
public bool Eat(Row @in, int[] remap){
     int sum = 0;
     foreach (Cell c in @in.cells.Values){
          sum += c.cnt;
          if (c.@ref >= 0){
               if (remap[c.@ref] == 0){
                    c.@ref = -1;
               }
          }
     }
     int frame = sum / 10;
     bool live = false;
     foreach (Cell c in @in.cells.Values){
          if (c.cnt < frame && c.cmd >= 0){
               c.cnt = 0; c.cmd = -1;
          }
          if (c.cmd >= 0 || c.@ref >= 0){
               live |= true;
          }
     }
     return !live;
}
```

Case study: *An example from Java-C# translation task where StructCoder is able to accurately predict the target code while CodeT5 fails. Red text indicates errors made by CodeT5 and blue text indicates correctly predicted code by StructCoder where baseline generates errors. The blue arrows show some of the correctly predicted data flow edges relevant to the colored text. StructCoder correctly generates the for loops by defining variable 'c' and the model predicts most of the DFG edges incident on the variable 'c' inside these for loops and also in the first 'if' statement.*

PPOCoder - Code Generation using Deep Reinforcement Learning

Goal: Improving the quality of codes generated from pre-trained models

- **Proposed Idea:** Designing a deep reinforcement learning fine-tuning framework which can incorporate the compiler/execution feedback (i.e., syntactic or functional correctness) as the external source of knowledge in the model optimization.

- We develop **a new reward function** based on the **discrete compiler feedback** (compilation or unit test signal when available) and the **syntactic and semantic matching scores** between the AST sub-trees and DFG edges of the sampled generations and the correct targets.

Syntactic Correctness:

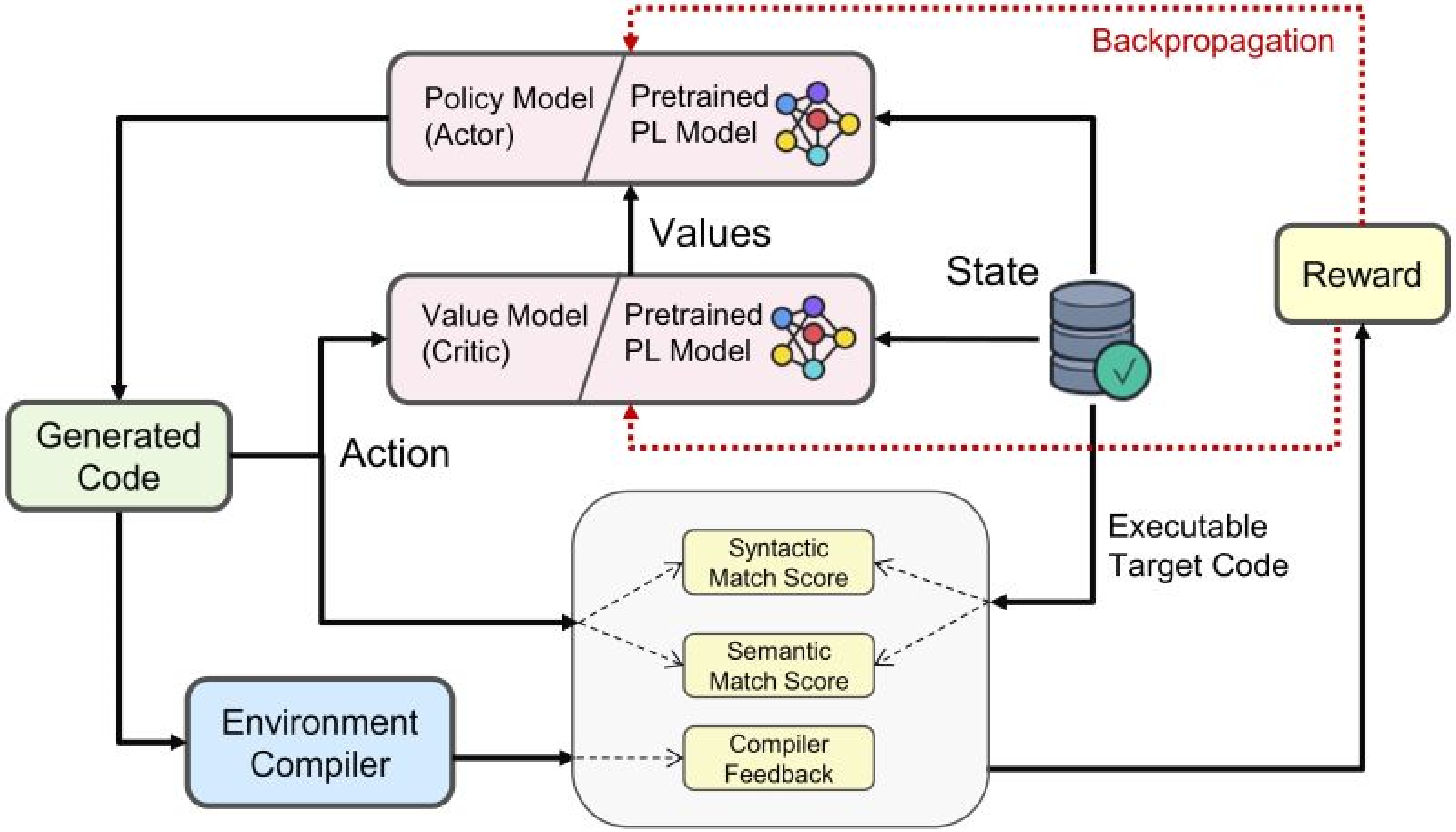
$$R_{cs}(\hat{y}) = \begin{cases} +1, \text{if } \hat{y} \text{ passed compilation test} \\ -1, \text{otherwise} \end{cases}$$

Functional Correctness:

$$R_{cs}(\hat{y}) = \begin{cases} +1 \ , & \text{if } \hat{y} \text{ passed all unit tests} \\ -0.3, & \text{if } \hat{y} \text{ failed any unit test} \\ -0.6, & \text{if } \hat{y} \text{ received RunTime error} \\ -1 \ , & \text{if } \hat{y} \text{ received Compile error} \end{cases}$$

This can provide a more **stable** and **generalizable** model optimization that is **less sensitive to new environments (i.e., tasks, PLs, or datasets).**

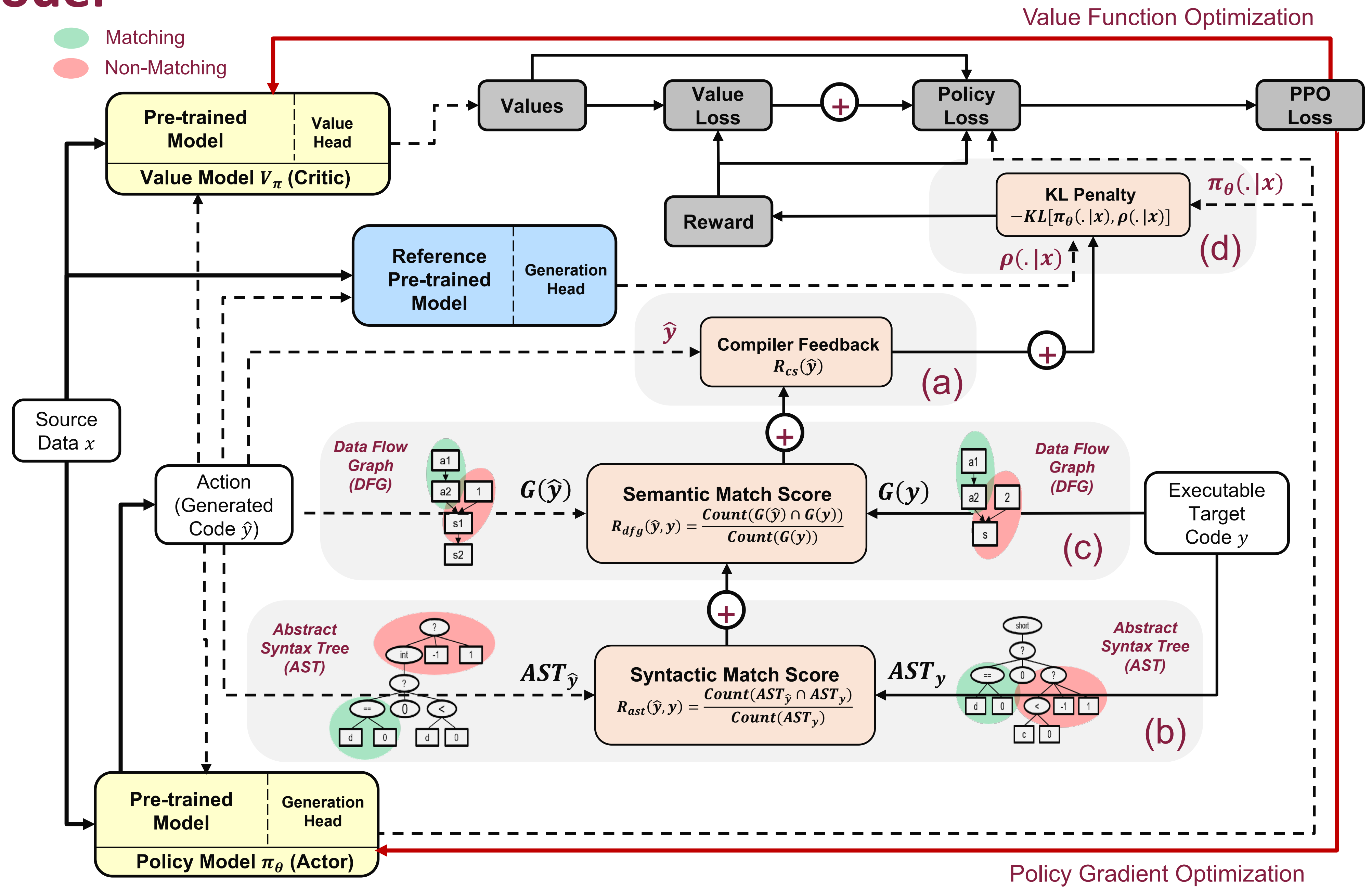
PPOCoder – Block Diagram



For Code Generation Tasks, we can have **Computer Feedback** instead of Human Feedback.
→ **Instead of RLHF, we have RLCF.**

P. Shojaee, A. Jain, S. Tipirneni, and C. K. Reddy, "Execution-based Code Generation using Deep Reinforcement Learning", arXiv 2023 (under review).

PPOCoder



Matching
Non-Matching
Value Function Optimization
Pre-trained Model
Value Head
Value Model V_π (Critic)
Values
Value Loss
Policy Loss
PPO Loss
Reward
KL Penalty
$-KL[\pi_\theta(.|x), \rho(.|x)]$
$\pi_\theta(.|x)$
$\rho(.|x)$
(d)
Reference Pre-trained Model
Generation Head
$\hat{y}$
Compiler Feedback
$R_{cs}(\hat{y})$
(a)
Source Data x
Action (Generated Code $\hat{y}$)
Data Flow Graph (DFG)
$G(\hat{y})$
Semantic Match Score
$R_{dfg}(\hat{y}, y) = \frac{Count(G(\hat{y}) \cap G(y))}{Count(G(y))}$
$G(y)$
Data Flow Graph (DFG)
(c)
Executable Target Code y
Abstract Syntax Tree (AST)
$AST_{\hat{y}}$
Syntactic Match Score
$R_{ast}(\hat{y}, y) = \frac{Count(AST_{\hat{y}} \cap AST_y)}{Count(AST_y)}$
AST_y
Abstract Syntax Tree (AST)
(b)
Pre-trained Model
Generation Head
Policy Model π_θ (Actor)
Policy Gradient Optimization

Experimental Results

Code Completion: Results on the code completion task for completing the last 25 masked tokens.

Model	*xMatch*	*Edit Sim*	*Comp Rate*
BiLSTM	20.74	55.32	36.34
Transformer	38.91	61.47	40.22
GPT-2	40.13	63.02	43.26
CodeGPT	41.98	64.47	46.84
CodeT5	42.61	68.54	52.14
PPOCoder + CodeT5	**42.63**	**69.22**	**97.68**

Code Translation: Performance comparison of PPOCoder and baselines on XLCoST. The column and row language headers represent the translation target languages. These values are a weighted average scores over six different source languages. The best results are shown in **bold** font.

Model	C++		Java		Python		C#		PHP		C	
	CodeBLEU	*CompRate*	*CodeBLEU*	*CompRate*	*CodeBLEU*	*CompRate*	*CodeBLEU*	*CompRate*	*CodeBLEU*	*CompRate*	*CodeBLEU*	*CompRate*
Naive Copy	38.68	12.82	50.39	16.38	38.93	13.26	50.83	6.16	29.88	7.77	53.83	2.56
CodeBERT	45.34	23.38	51.28	27.89	45.07	27.62	57.63	11.30	46.53	14.67	23.69	12.06
PLBART	66.03	46.42	65.23	35.67	62.93	46.66	70.61	31.29	69.05	60.85	47.24	16.36
CodeT5	71.92	62.46	73.18	64.73	**73.24**	69.19	75.29	63.51	79.21	78.35	**71.42**	42.70
PPOCoder + CodeT5	**72.11**	**72.38**	**73.22**	**86.95**	72.67	**90.81**	**75.86**	**76.71**	**79.96**	**85.81**	70.92	**48.82**

Results on Program Synthesis

Model	Size	State	*pass@80*
GPT	224M	fine-tuned	7.2
GPT	422M	fine-tuned	12.6
GPT	1B	fine-tuned	22.4
GPT	4B	fine-tuned	33.0
GPT	8B	fine-tuned	40.6
GPT	68B	fine-tuned	53.6
GPT	137B	fine-tuned	61.4
CodeT5	60M	fine-tuned	19.2
CodeT5	220M	fine-tuned	24.0
CodeT5	770M	fine-tuned	32.4
CodeRL+CodeT5	770M	zero-shot	63.0
PPOCoder +CodeT5	770M	zero-shot	**68.2**

Results of the zero-shot transferability on MBPP. Both zero-shot models are finetuned on APPS and evaluated on MBPP in the zero-shot setting.

Program Translation

➔ Converting source code from one programming language to another

Program Translation

Java Program

```java
class GFG {
  static void printPairs(int arr[], int
n, int sum){
    for (int i = 0; i < n; i++)
      for (int j = i + 1; j < n; j++)
        if (arr[i] + arr[j] == sum)
          System.out.println(arr[i] + ",
" + arr[j]);
  }
  public static void main(String[] arg){
    int arr[] = { 1, 5, 7, -1, 5 };
    int n = arr.length;
    int sum = 6;
    printPairs(arr, n, sum);
  }
}
```

Python Program

```python
def printPairs(arr, n, sum):
    for i in range(0, n ):
        for j in range(i + 1, n ):
            if (arr[i] + arr[j] == sum):
                print(arr[i], ", ",
arr[j], sep = "")


arr = [1, 5, 7, -1, 5]
n = len(arr)
sum = 6
printPairs(arr, n, sum)
```

Manual/Rule-based program translation:

➔ Requires expertise in both source and target programming languages

➔ Requires significant amount of time and resources depending on the scale of the code base

Available Code Translation Datasets

Java Program

```java
public static void main(String[] args) {
    FastScanner fs=new FastScanner();
    int T=1;
    for (int tt=0; tt<T; tt++) {
        int n=fs.nextInt();
        char[] a=fs.next().toCharArray();
        ArrayList<Integer>ws=new ArrayList<>(),
              rs=new ArrayList<>();
        for(int i=0;i<a.length;i++){
            if(a[i]=='W'){
                ws.add(i);
            }
            else{
                rs.add(i);
            }
        }
        int wInd=0,rInd=rs.size()-1;
        int count=0;
        while( wInd<ws.size()&& rInd>=0 &&
ws.get(wInd)<rs.get(rInd)){
            count++;
            wInd++;
            rInd--;
        }
        System.out.println(count);
    }

}
```

C++ Program

```cpp
#include <bits/stdc++.h>
using namespace std;
int main() {
  int n,r=0,a=0; cin>>n;
  char c[n];
  for(int i=0;i<n;i++){
    cin>>c[i];
    if(c[i]=='R') r++;
  }
  for(int i=0;i<r;i++){
    if(c[i]=='W') a++;
  }
  cout<<a;
}
```

- ➔ Submitted solutions to online code challenges
- ➔ Significant distribution discrepancy across different languages

Problem Description:
Given an input string, find the minimal number of steps to ensure W is not on the immediate left of R. You can swap any two characters, or flip R to W and vice versa.

Input: WRWWRWRR
Output: 3
Swap: WRWWRWRR; Flip (twice): RRWWWWRR; Result: RRWWWWWW

Source: https://atcoder.jp/contests/abc174/tasks/abc174_d

Our XLCOST Dataset

A **C**ross-**l**ingual **Co**de **S**nippet **T**ranslation (XLCoST) dataset https://www.geeksforgeeks.org/

- ➔ Parallel at both program and snippet level
  - ◆ Snippets are aligned by comments
- ➔ 7 common programming languages
  - ◆ C++, Java, Python, C#, Javascript, PHP, C
  - ◆ 42 languages pairs for Translation

- ➔ Similar distribution of source and target languages
  - ◆ Similar length, vocabulary and style
- ➔ Manually verified for misalignment and other errors
  - ◆ Data quality ensured

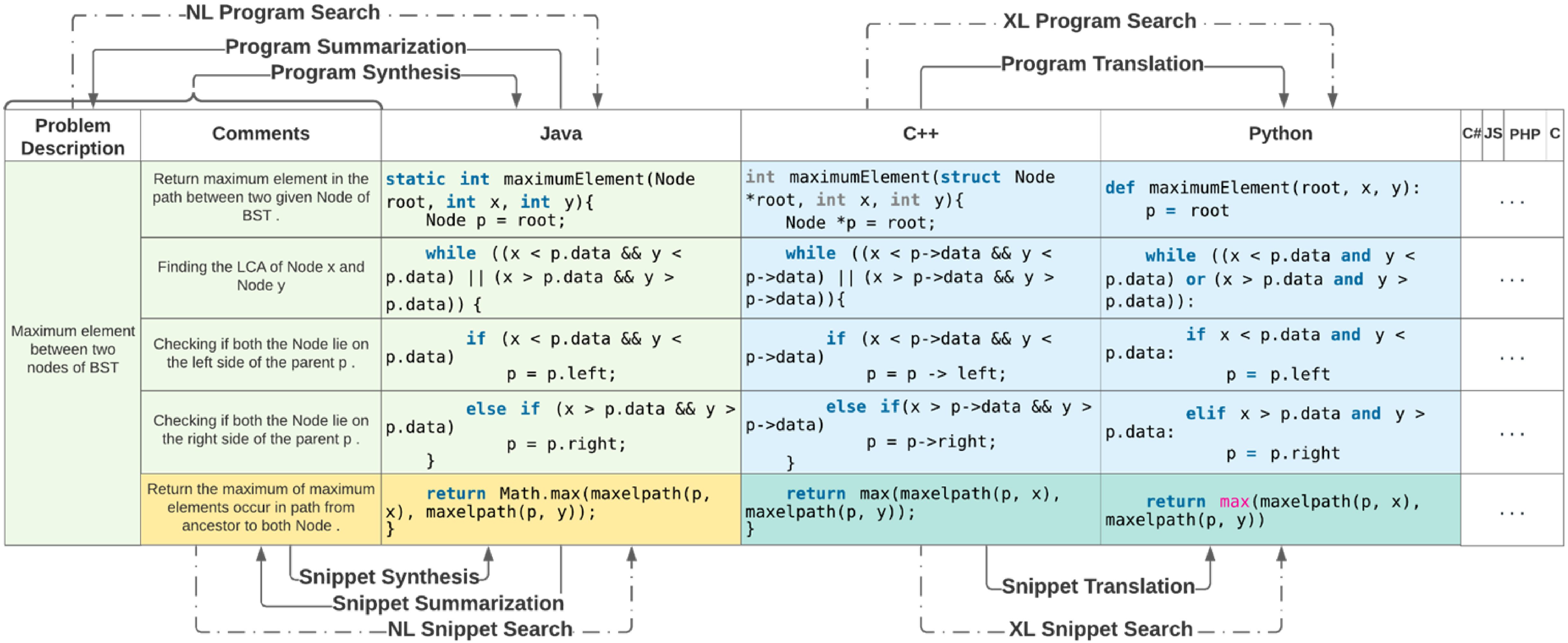
<table>
<tr><th>Java</th><th>Python</th><th>PHP</th><th>C</th></tr>
<tr>
<td>import java.io.*;
class GFG {
    // Function to check whether a number is divisible by 7
    static boolean isDivisibleBy7(int num) {</td>
<td># Function to check whether a number is divisible by 7
def isDivisibleBy7(num) :</td>
<td><?php
// Function to check whether a number is divisible by 7
function isDivisibleBy7($num){</td>
<td>#include <stdio.h>
// Function to check whether a number is divisible by 7
int isDivisibleBy7(int num) {</td>
</tr>
<tr>
<td>// If number is negative,
// make it positive
if(num < 0)
    return isDivisibleBy7(-num);</td>
<td># If number is negative
make it positive
if num < 0 :
    return isDivisibleBy7(-num)</td>
<td>// If number is negative,
// make it positive
if($num < 0)
    return isDivisibleBy7(-$num);</td>
<td>// If number is negative,
// make it positive
if(num < 0)
    return isDivisibleBy7(-num);</td>
</tr>
<tr>
<td>// Base cases
if(num == 0 || num == 7)
    return true;
if(num < 10)
    return false;</td>
<td># Base cases
if(num == 0 or num == 7) :
    return True
if(num < 10) :
    return False</td>
<td>// Base cases
if($num == 0 || $num == 7)
    return 1;
if($num < 10)
    return 0;</td>
<td>// Base cases
if(num == 0 || num == 7)
    return 1;
if(num < 10)
    return 0;</td>
</tr>
<tr>
<td>// Recur for (num / 10 - 2 * num % 10)
return isDivisibleBy7(...</td>
<td># Recur for (num / 10 - 2 * num % 10)
return isDivisibleBy7(...</td>
<td>// Recur for (num / 10 - 2 * num % 10)
return isDivisibleBy7(...</td>
<td>// Recur for (num / 10 - 2 * num % 10)
return isDivisibleBy7(...</td>
</tr>
<tr><td>...</td><td>...</td><td>...</td><td>...</td></tr>
</table>

Challenges with Available Code Translation Datasets

- Huge amount of code data from open source repos, but unlabelled
  - GitHub, billions of programs in all possible programming languages
- Labelled data are very small in size
  - CoST only has around 70 programs for testing and 50 programs for validation
- Labelled data covers very limited languages
  - CodeXGLUE translation, only Java-C#
- Quality of the data are generally unverified
  - Many of the available programs are crowd-sourced

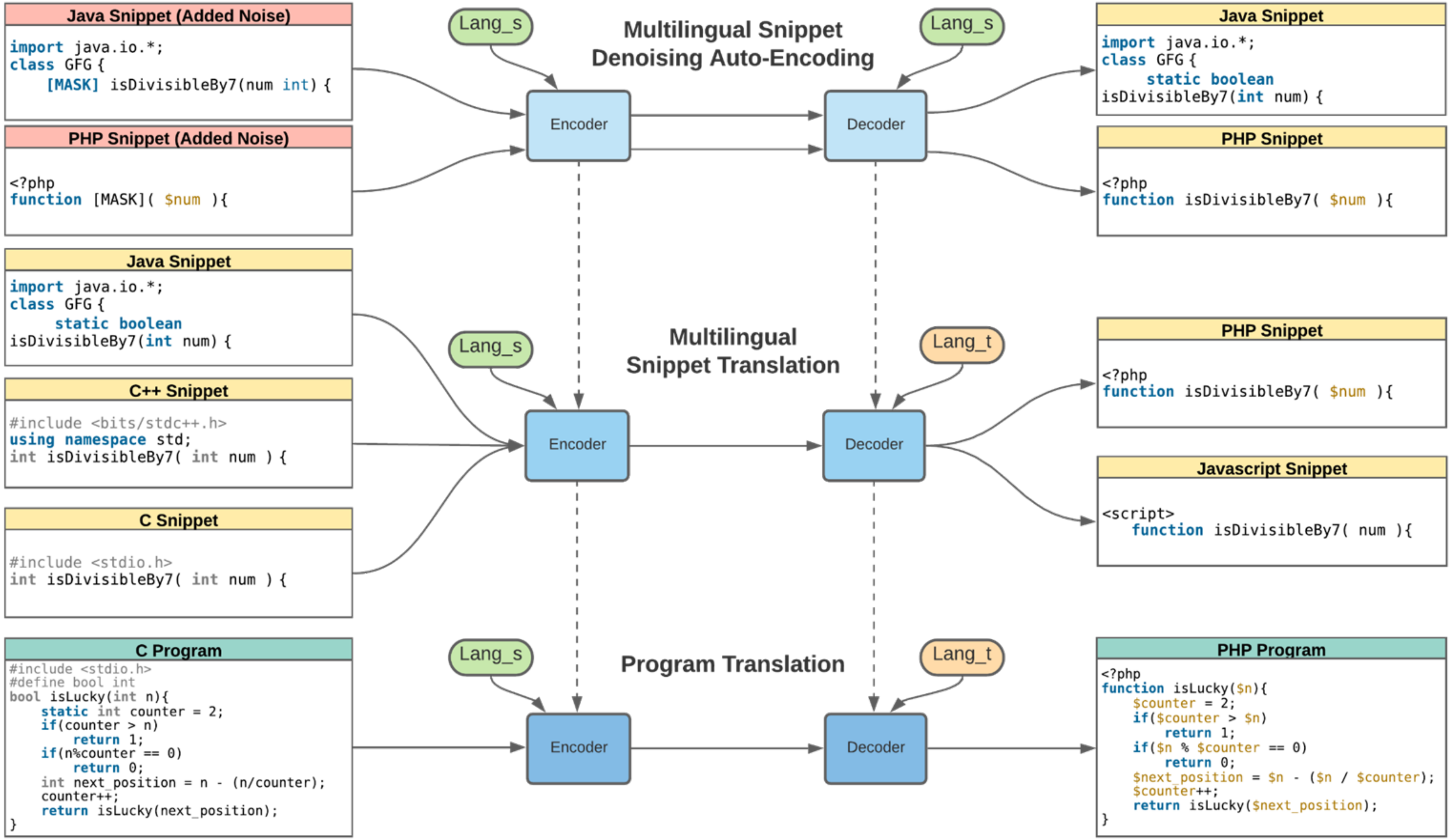
Dataset	Alignment	Task	Labelling	Size	Languages
CodeNet	Program	Multiple	Solutions to the same problem	13.9M*	55 programming languages
AVATAR	Program	Translation	Solutions to the same problem	57,414	Java, Py
CodeXGLUE	Method	Multiple	Matching function names	11,800	Java, C#
CoST	Snippet	Translation	Matching code comments	132,046	C++, Java, Py, C#, JS, PHP, C
XLCoST	Snippet	Multiple	Matching code comments	1,002,296	C++, Java, Py, C#, JS, PHP, C, English

XLCoST - Data and Tasks



M. Zhu, A. Jain, K. Suresh, R. Ravindran, S. Tipirneni, and C. K. Reddy, XLCoST: A Benchmark Dataset for Cross-lingual Code Intelligence. arXiv 2022.

Multilingual Snippet Training (MuST)



M. Zhu, K. Suresh, and C. K. Reddy. "Multilingual Code Snippets Training for Program Translation." AAAI 2022.

Generated Outputs

Problem Description:
Count the number of 1s in the binary of form of the given integer.

DOBF: outputs a different function (that returns the weighted sum of an array)

CodeBERT: infinite loop; undefined variable; incorrect logic

Transformer: totally incorrect logic

Java to PHP

Input

```
import java.io.*;
class countSetBits {
  static int countSetBits(int n) {
    int count = 0;
    while (n > 0) {
      count += n & 1;
      n >>= 1;}
    return count;}
  public static void main(String args[])
  {
    int i = 9;
    System.out.println(countSetBits(i));
  }}
```

Ground Truth

```
<?php
function countSetBits($ n) {
  $ count = 0;
  while ($ n) {
    $ count += $ n & 1;
    $ n >>= 1;
  }
  return $ count;
}
$ i = 9;
echo countSetBits($ i); ? >
```

MuST-PT

```
<?php
function countSetBits($ n) {
  $ count = 0;
  while ($ n > 0) {
    $ count += $ n & 1;
    $ n >>= 1;
  }
  return $ count;
}
$ n = 100;
echo countSetBits($ n); ? >
```

DOBF

```
<?php
function maxSum($ arr, $ n) {
  sort($ arr);
  $ sum = 0;
  for ($ i = 0; $ i < $ n; $ i++) $ sum
      += ($ arr[$ i] * $ i);
  return $ sum;
}
$ arr = array(3, 5, 6, 1);
$ n = count($ arr);
echo maxSum($ arr, $ n); ? >
```

CodeBERT

```
<?php
function countSetBits($ n) {
  $ count = 0;
  while ($ n > 0) {
    $ count += 1;
    $ i++;
  }
  return $ count;
}
$ n = 5;
$ n = count($ n);
echo countSetBits($ n); ? >
```

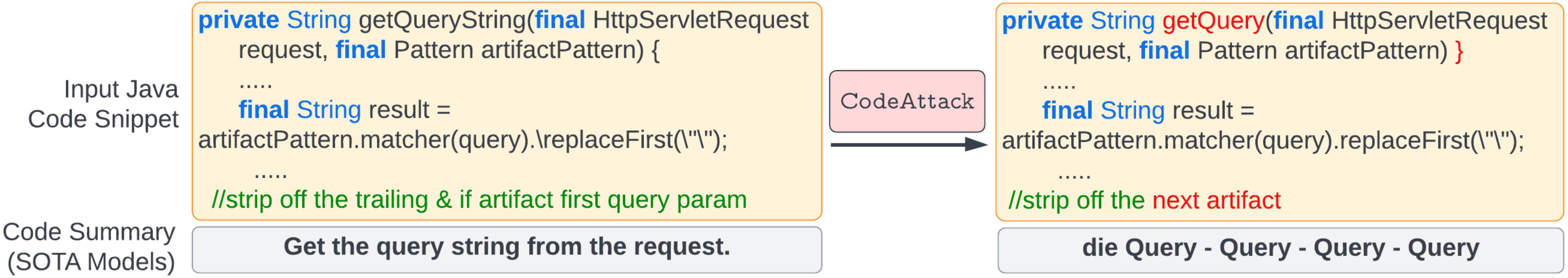
Transformer

```
<?php
function countSetBits($ n) {
  $ count = 0;
  while ($ n) {
    $ count = 0;
    $ n = 0;
    while ($ p != 0) {
      $ count += ($ n + 1);
    }
    return $ count;
  }
$ n = 0;
echo countSetBits($ n); ? >
```

CodeAttack: Code-based Adversarial Attacks

- A simple yet effective black-box attack model for generating adversarial samples.
- Detect the vulnerabilities of pre-trained Code PL models to adversarial attacks.
- Transferable across different downstream tasks and different programming language tasks.

CodeAttack makes a small modification (in red) which changes the output significantly.



A. Jha and C. K. Reddy. "CodeAttack: Code-based Adversarial Attacks for Pre-Trained Programming Language Models." AAAI 2023.

CodeAttack – Threat Model

- Adversary's Capabilities
  - Character-level / Token-level perturbations
  - Perturb only a small number of tokens/characters
  - High similarity between the perturbed (X_{adv}) the original (X) code
- Adversary's Knowledge
  - Black-box access – no access to model parameters, model architectures, gradients
  - Access to output logits for supervision
- Adversary's Goal
  - Degrade the quality of the generated output sequence.
  - Objective function: $\Delta_{atk} = \text{argmax}_{\delta} [Q(F(X)) - Q(F(X_{adv}))]$
  - $Q(.)$ measures the quality; F is the given pre-trained model

Code-specific constraints for code consistency and for **limiting** the **search space** for efficient attacks.

Performance Results

- ➔ Downstream Task and Languages
  - Code Translation, Code Repair, Code Summarization
  - C#, Java, Python, PHP
- ➔ Victim Models
  - CodeT5, CodeBERT, GraphCodeBERT, RoBERTa
- ➔ Baseline Models
  - TextFooler, BERT-Attack

Original Code	TextFooler	BERT-Attack	CodeAttack

`public override void WriteByte(`
`byte b) {`
`if (outerInstance.upto ==`
`outerInstance.blockSize)`
`{... }}`
	`audiences revoked canceling`
	`WriteByte(byte b) {`
	`if (outerInstance.upto ==`
	`outerInstance.blockSize)`
	`{.... }}`
		`public override void ; . b) {`
		`if (outerInstance.upto ==`
		`outerInstance.blockSize)`
		`{... }}`
			`public override void WriteByte(`
			`bytes b) {`
			`if (outerInstance.upto ==`
			`outerInstance.blockSize)`
			`{... }}`

$\text{CodeBLEU}_{\text{before}}$:100	Δ_{drop}:5.74; CodeBLEU_q: 63.28	Δ_{drop}:27.26; CodeBLEU_q:49.87	Δ_{drop}:20.04; CodeBLEU_q: 91.69

Qualitative Results: Code Translation for C#-Java tasks

Generates adversarial samples that are **efficient, effective, imperceptible, fluent,** and **consistent.**

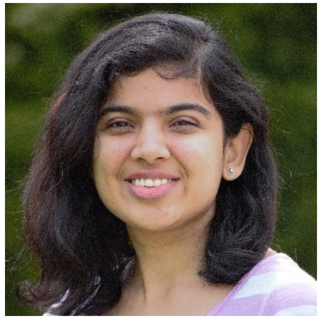
Conclusion & Future Directions

- **StructCoder** improves code generation by introducing two structure-preserving tasks for the decoder. Incorporating AST and DFG code structure constraints can improve the syntax and semantics of the generated code.
- **PPOCoder** - Reinforcement learning can aid in developing codes of high quality by incorporating various feedbacks – which will compile and pass unit test cases along with syntactic and functional correctness.
- Data quality is extremely important and can significantly help in reducing the size of the massive deep learning architectures. We released a code snippet level translation dataset **XLCOST**.
- Developed **CodeAttack**, a black-box adversarial attack model to detect vulnerabilities of the SOTA Code pre-trained LMs by finding the most vulnerable tokens to identify contextualized substitutes subject to code-specific constraints.

- How well do these models work on low-resource programming languages (legacy codes)?
- Identify vulnerabilities through structure-preserving attacks that will allow the code to compile and execute.
- Can we build defense mechanisms against such attacks and make these models robust?

Acknowledgements

Graduate Students and Collaborators

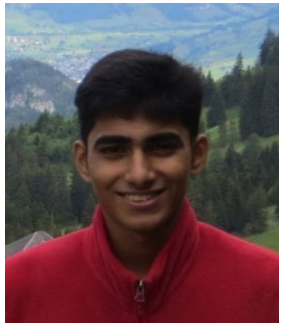


Sindhu Tipirneni

Parshin Shojaee

Aneesh Jain

Akshita Jha



Karthik Suresh

Ming Zhu

Funding Agencies

NSF

NIH

DEPARTMENT OF ENERGY · UNITED STATES OF AMERICA

aws

BlueCross BlueShield

Questions and Comments

Feel free to email questions or suggestions to

reddy@cs.vt.edu

http://www.cs.vt.edu/~reddy/

https://github.com/reddy-lab-code-research