Programming Language Processing

Learning To Edit Code

Saikat Chakraborty
Ph.D. Candidate
Department of Computer Science
Columbia University.

CS@CU COMPUTER SCIENCE

Automatic Code Editing

Adding Feature

```
public String removeComment(String leftOver) {
-   if (hasBlockComment(leftOver)){
+   while (hasBlockComment(leftOver)){
        leftOver = removeBlockComment(leftOver);
    }
-   if (hasLineComment(leftOver)){
+   while (hasLineComment(leftOver)){
        leftOver = removeLineComment(leftOver);
    }
    return leftOver;
}
```

Bug-fixing

```
public abstract void removeSessionCookies()
                                    throws Exception { ;
-     throw new MustOverrideException();
- }
```

Refactoring

```
void visit(JSession *session, Timer t) throws Exception {
    if (*session != null && t.getTime() > *session.getStartTime()){
        visit((JNode) *session, t);
    }
    else {
        visit(new JNode(), new Timer());
    }
}
```

Code Edits Are Repetitive [Meng *et. al.* 2011, 2013], [Ray *et. al.* 2015]

```
    ...
-   if (super.equals(object)){
+   if (this == object){
        ...
    }
    ...
```

```
    ...
-   return super.equals(object);
+   return this == object;
```

```
    ...
-   if (this.equals(object)){
+   if (this == object){
        ...
    }
    ...
```

Is it possible to
Automate
Such Code Edits?

Automating Code Edits - Template/Search based

Edit Template

```
      ...
-     return super.equals(object);
+     return this == object;
      ...
```

Match Found

```
public Model copy(Model instance){
    ...
    instance.notify();
    if (super.equals(instance) && !instance.isEmpty()){
        return instance.clone();
    }
    ...
}
```

Automating Code Edits - Template/Search based

Edit Template

```
    ...
-   return super.equals(object);
+   return this == object;
    ...
```

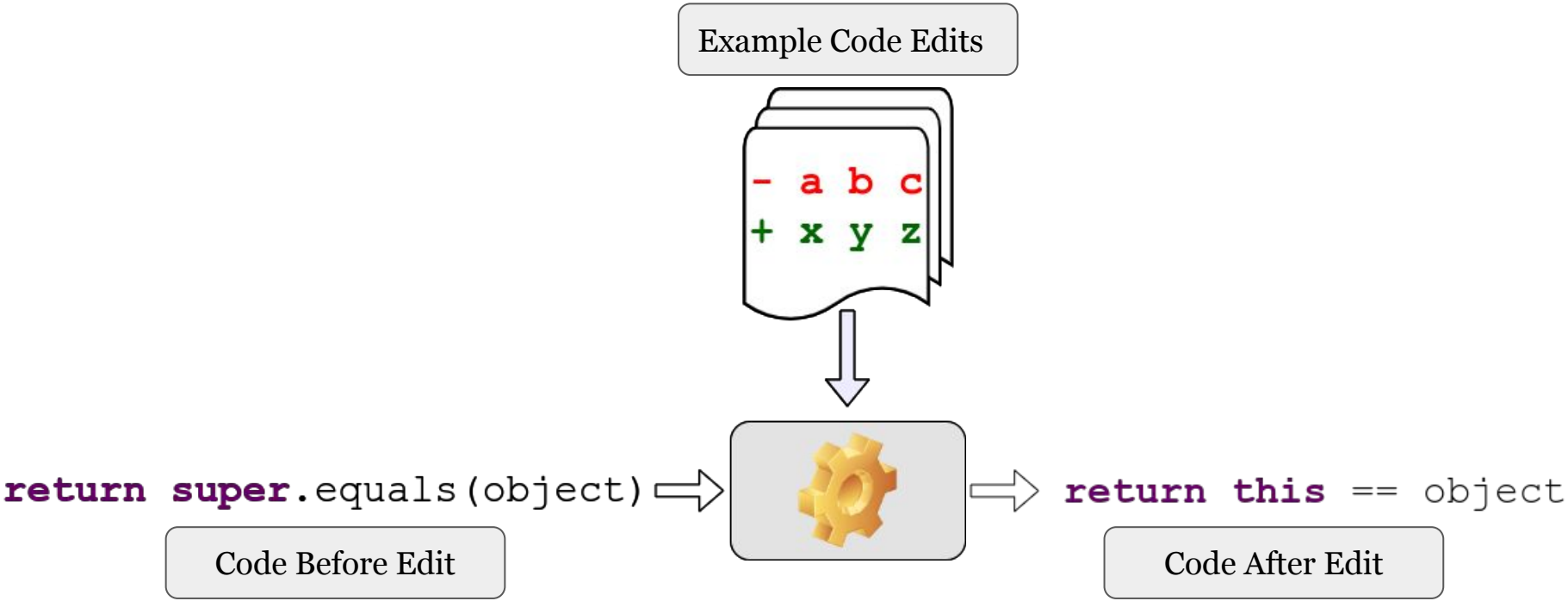
Patch Applied

```
public Model copy(Model instance){
    ...
    instance.notify();
    if (this == instance && !instance.isEmpty()){
        return instance.clone();
    }
    ...
}
```

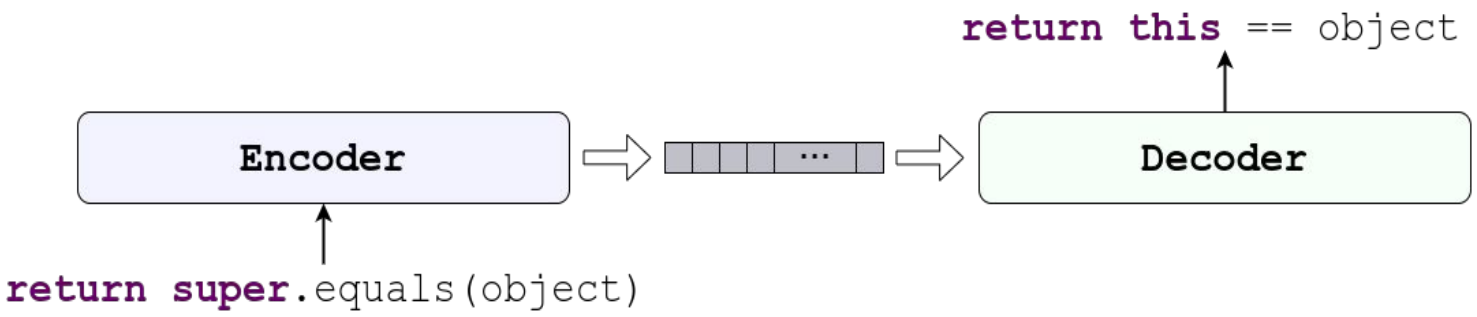
Problems

- **Too many templates** to write.
- Bottleneck : **Template Matching** / **Code Search**

Code Editing Task - Learning Based Solution [Tufano *et. al.* 2019]

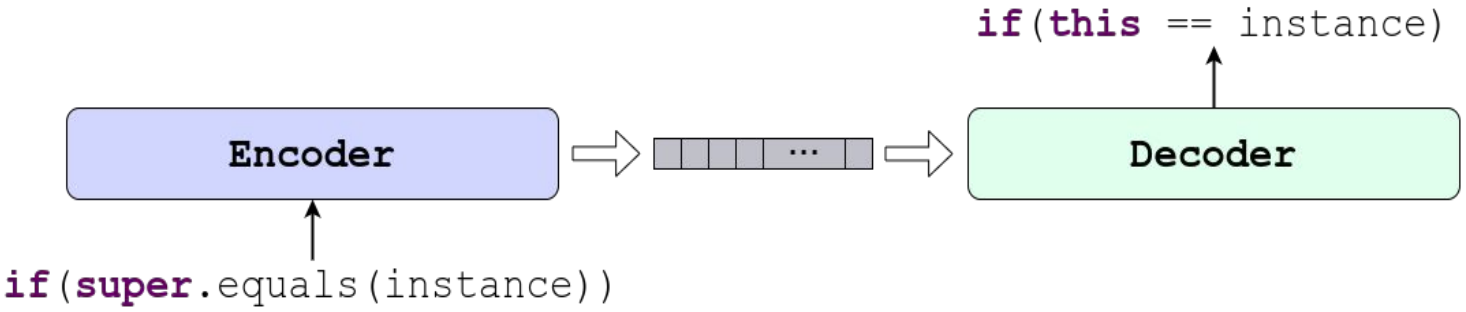


Code Editing Task - Learning Based Solution



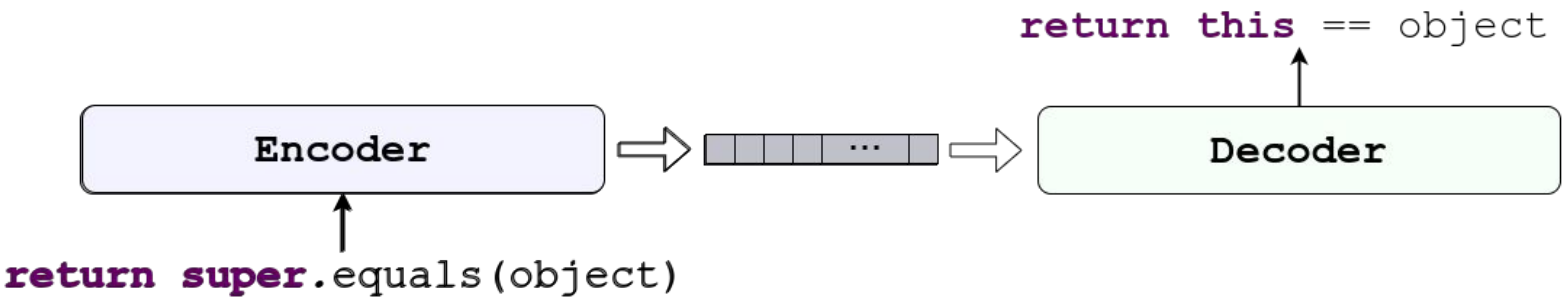
Encoder and **Decoder** learns
Code Edit Patterns from example edits.

Code Editing Task - Learning Based Solution



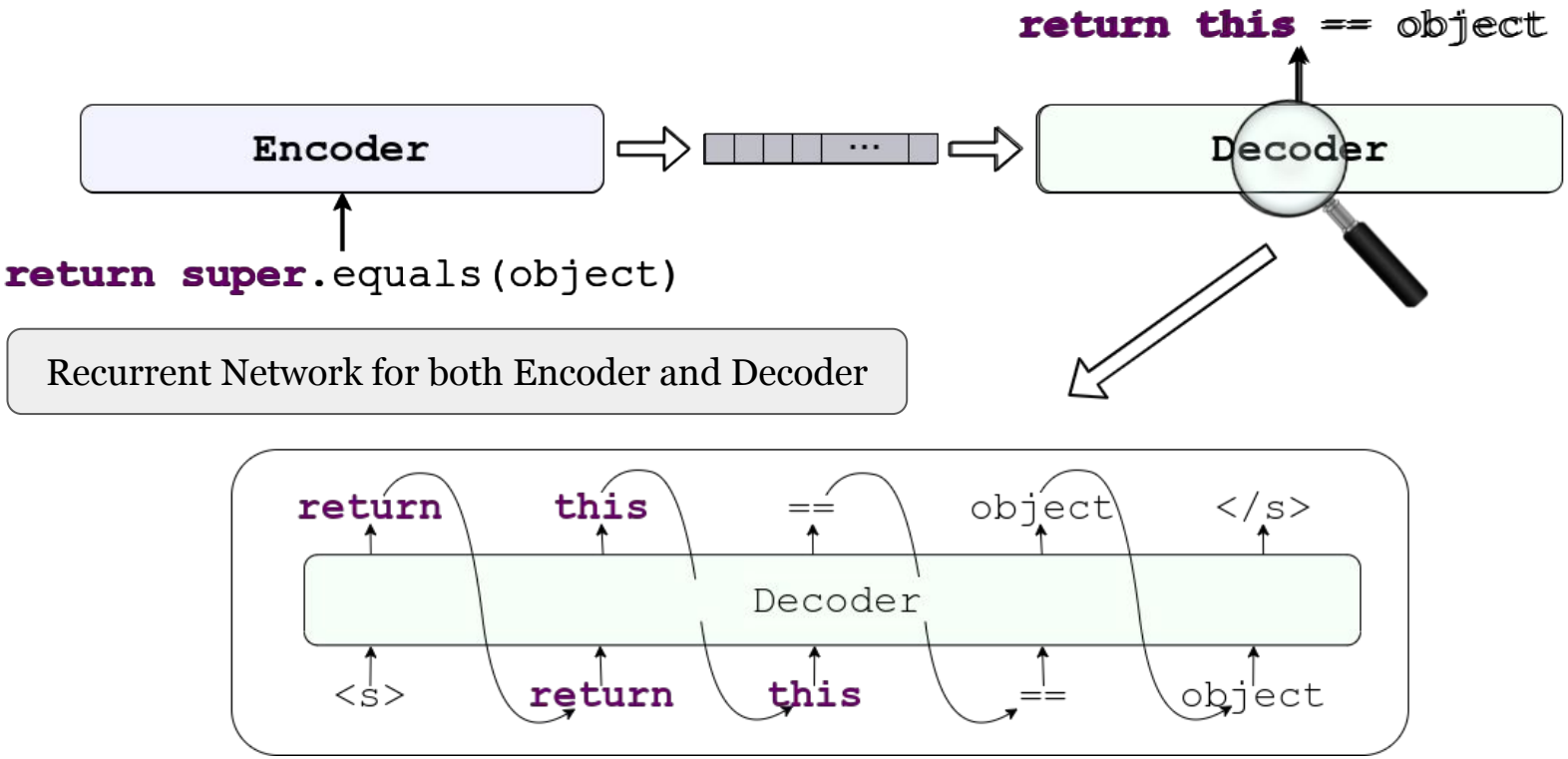
Encoder and **Decoder** learns to
Apply the Pattern in **Similar Context**.

Encoder-Decoder

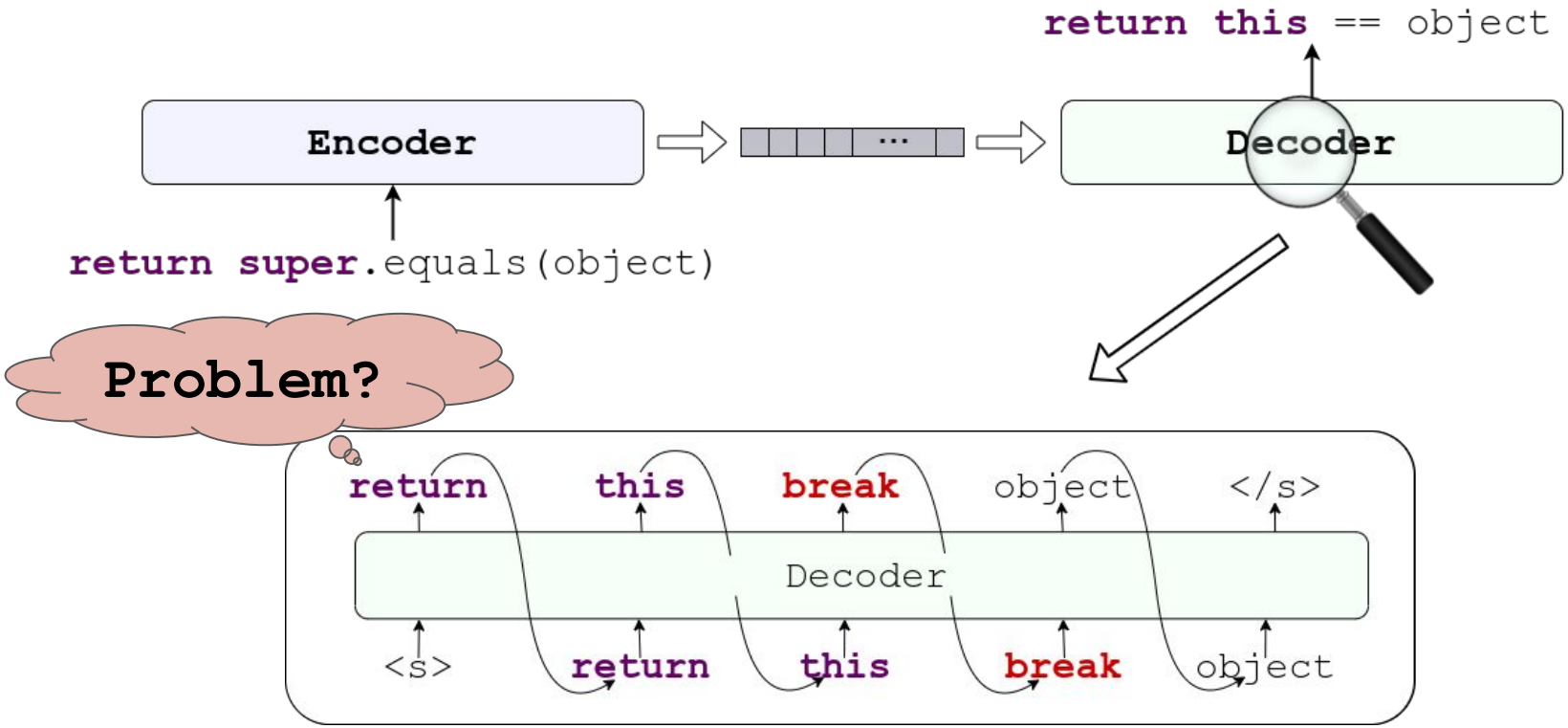


Decoder generates the edited code.

Code as sequence of tokens [Tufano *et. al.* 2018, 2019]



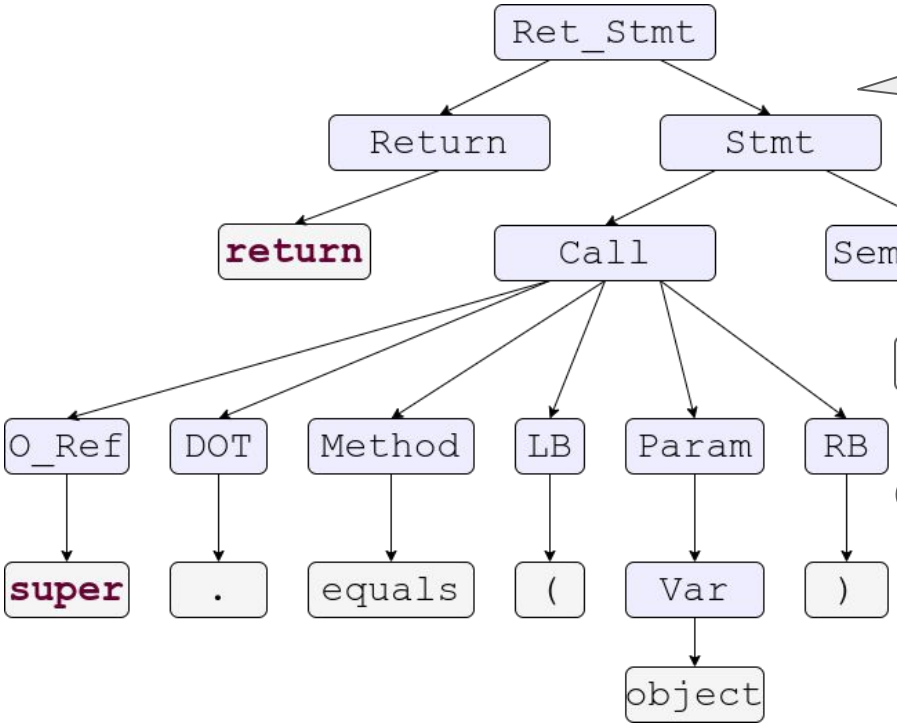
Code as sequence of tokens [Tufano *et. al.* 2018, 2019]



Desired Properties of Encoder and Decoder

Properties of Source Code.

```
return super.equals(object);
```



Code are **Highly Structured.**

Invisible properties.
e.g.
Non-Terminals

Properties of Source Code.

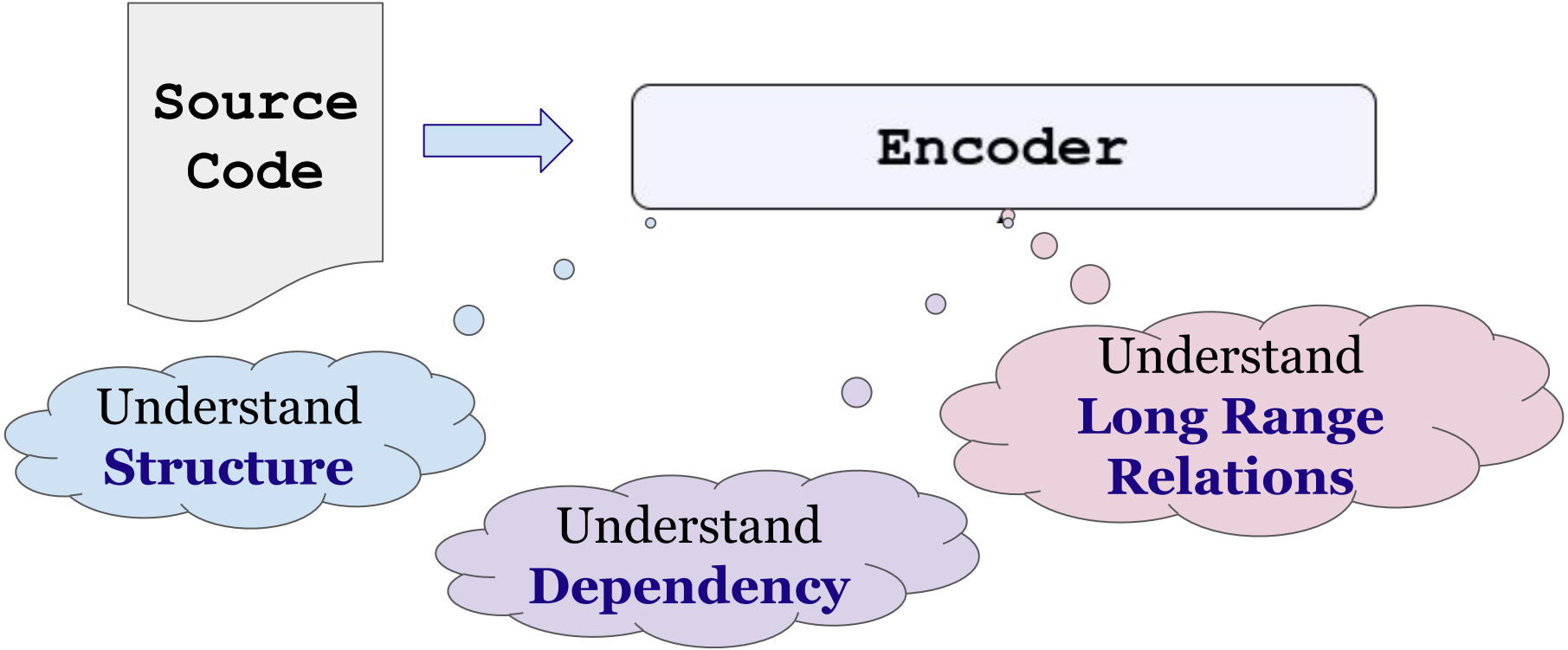
```
1.  public boolean checkEqual(
2.                  Object inst, MyClass object){
3.      MyClass tmp = new MyClass();
4.      if (inst == null) {
5.          MyClass tmp2 = new MyClass();
...
20.     }
21.     return super.equals(object);
22. }
```

Control

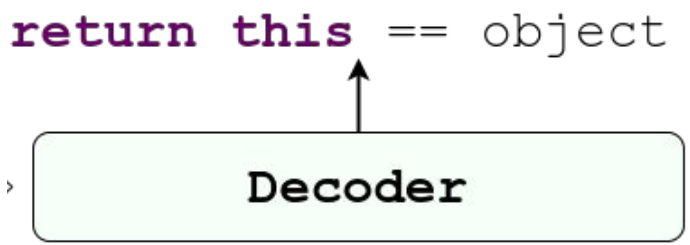
Data

Long Range Relation

Desired Properties of the Encoder



Desired Properties of the Decoder



1. Syntactic correctness.
2. Contextual correctness.

Syntactically Incorrect

```
boolean f (Object target)  {
    for(Object elem : if.elements) {
        if (elem.equals(target)) {
            return true;
        }
    }
    return false;
}
```

Contextually Incorrect

```
boolean f (Object target)  {
    for(Object elem : this.elements) {
        if (elem.equals(f)) {
            return null;
        }
    }
    return false;
}
```

1 Enforcing the Syntactic Correctness

CODIT: Code Editing With Tree Based Neural Models
TSE 2020

2 Learning the Contextual Correctness

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

3 Multi-Modal Code Editing

Work in progress

CODIT: Code Editing With Tree Based Neural Models

TSE - 2020

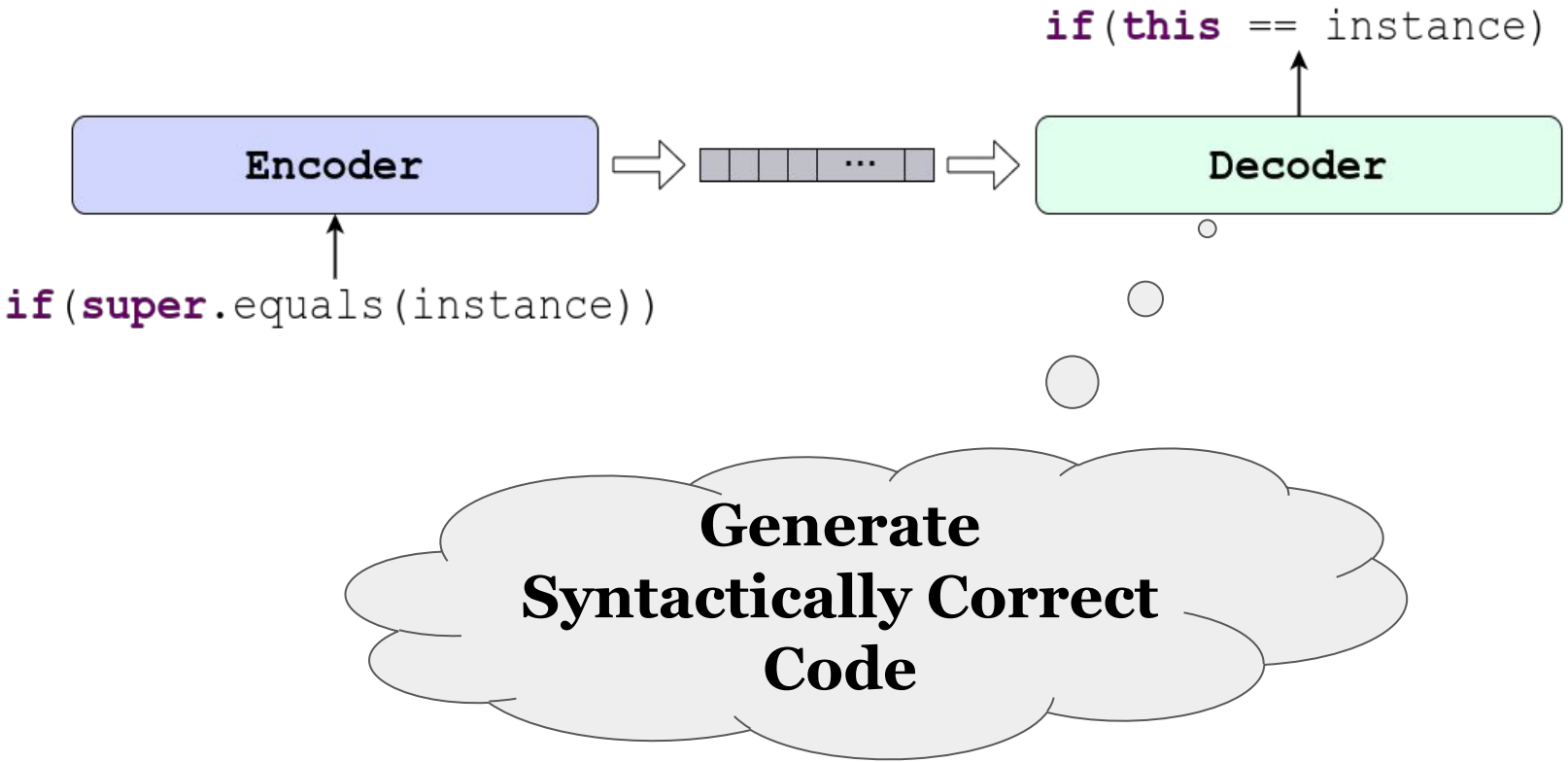
Findings

Generation of Syntax Tree instead of code Guarantees Syntactic Correctness.

Contribution

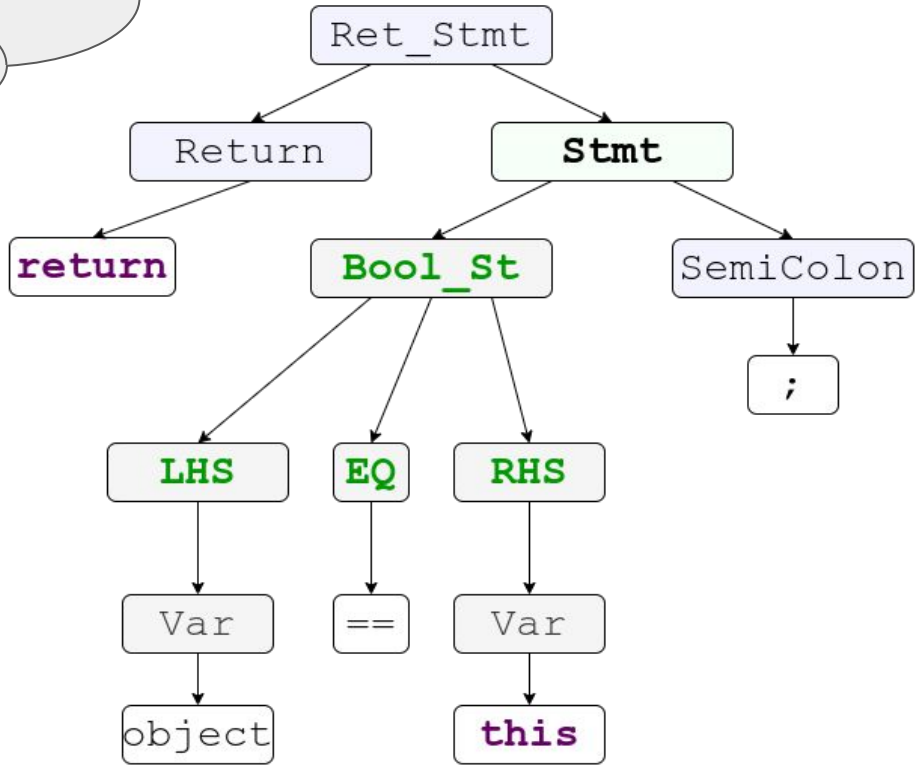
Tree/Grammar Based Model for Automatic Code Editing.

Is it possible to **Edit Code** with **Guaranteed Syntactic Correctness**?

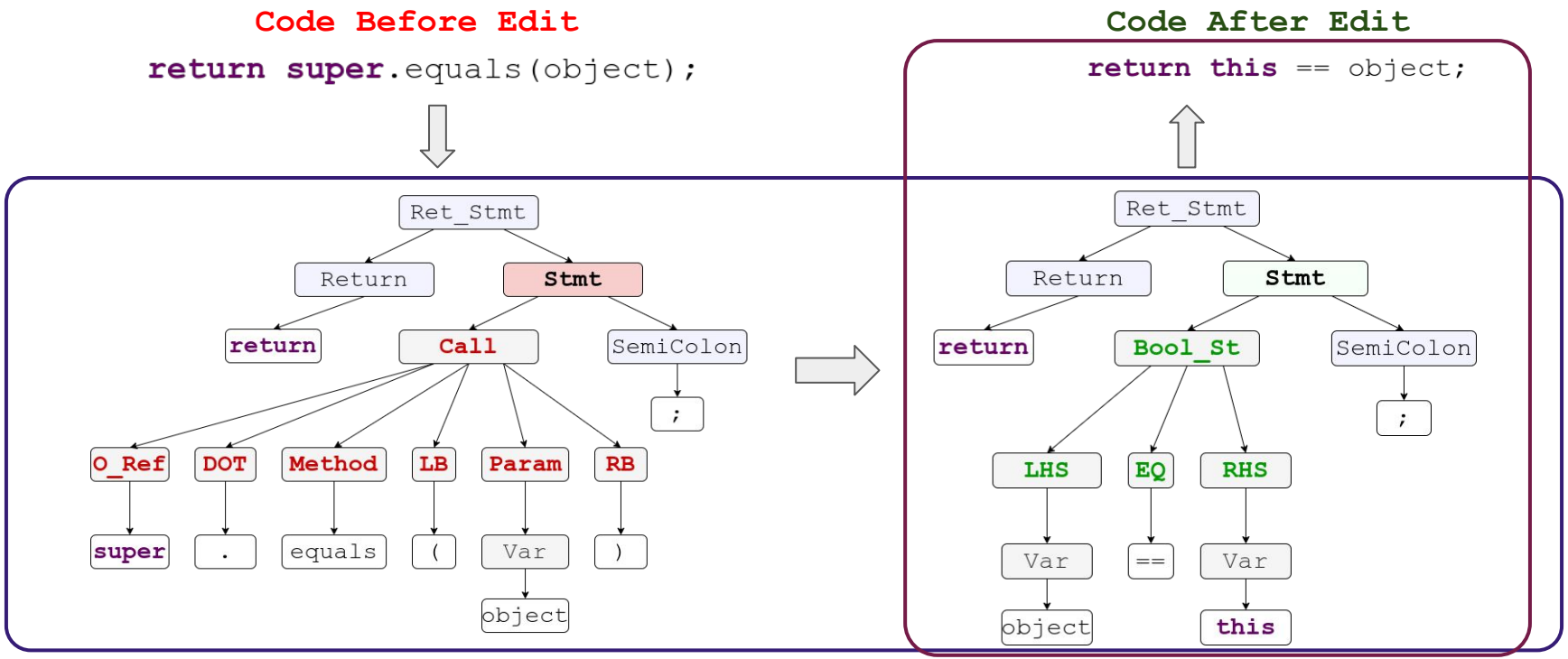

if(this == instance)
Encoder
...
Decoder
if(super.equals(instance))
Generate
Syntactically Correct
Code

Generate Syntactically Correct Code

```
return this == object;
```



CODIT: Code Editing With Tree Based Neural Models

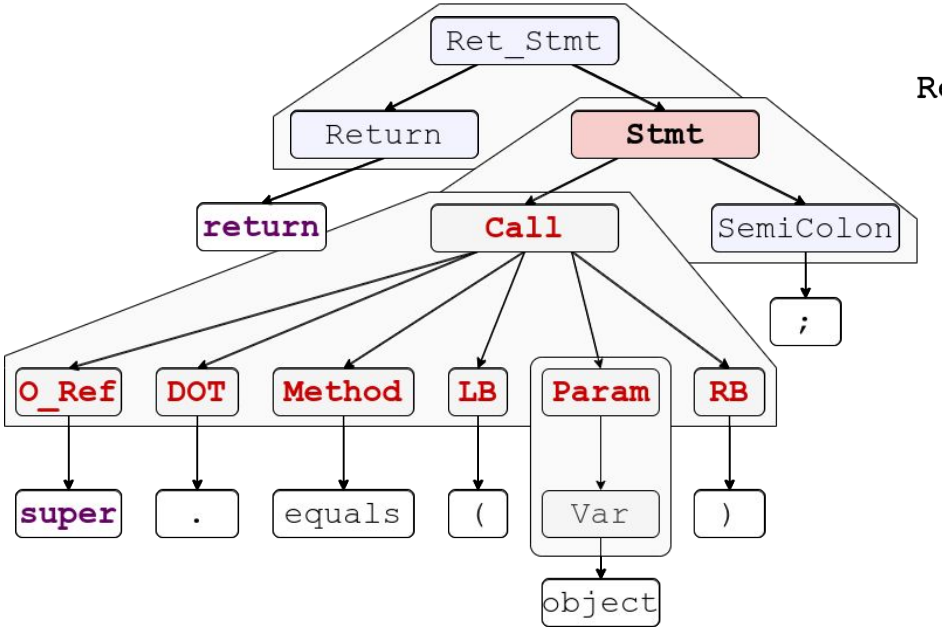


CODIT Step 1

Tree Translation

CODIT: Code Editing With Tree Based Neural Models

```
return super.equals(object);
```



Ret_Stmt → Return Stmt

Stmt → Call SemiColon

Call → O_Ref DOT Method LB Param RB

Param → Var

CODIT: Code Editing With Tree Based Neural Models

Rules sequence of Syntax Tree before edit

```
Ret_Stmt ──→ Return Stmt
  Stmt   ──→ Call Semicolon
  Call   ──→ O_Ref DOT Method
                LB Param RB
 Param   ──→ Var
```

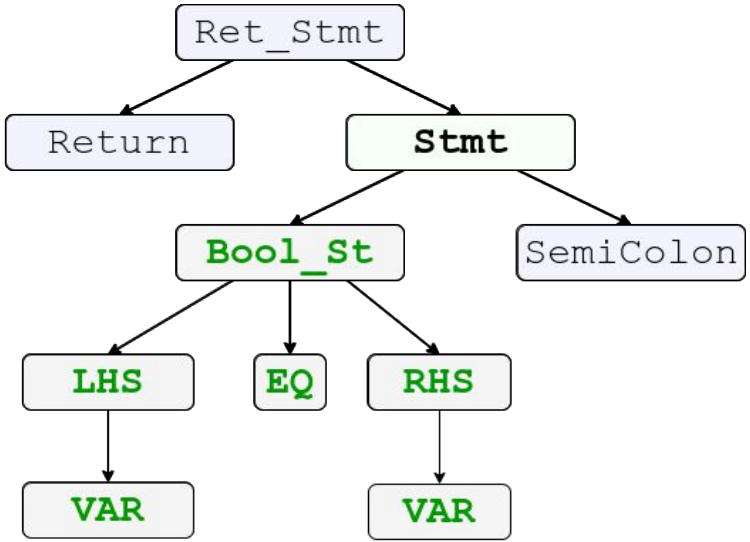
⇨

Rules sequence of Syntax Tree after edit

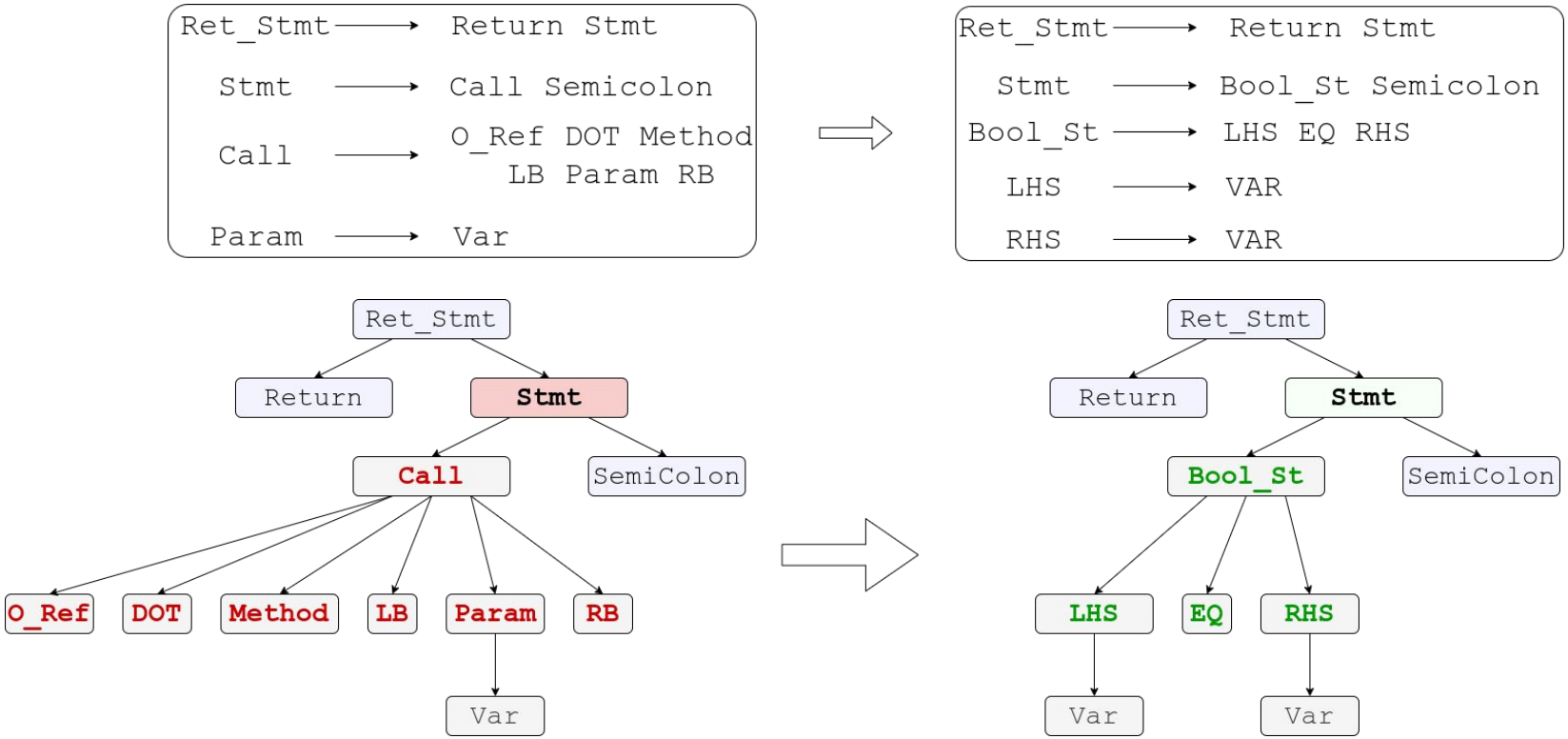
```
Ret_Stmt ──→ Return Stmt
  Stmt   ──→ Bool_St Semicolon
Bool_St  ──→ LHS EQ RHS
  LHS    ──→ VAR
  RHS    ──→ VAR
```

CODIT: Code Editing With Tree Based Neural Models

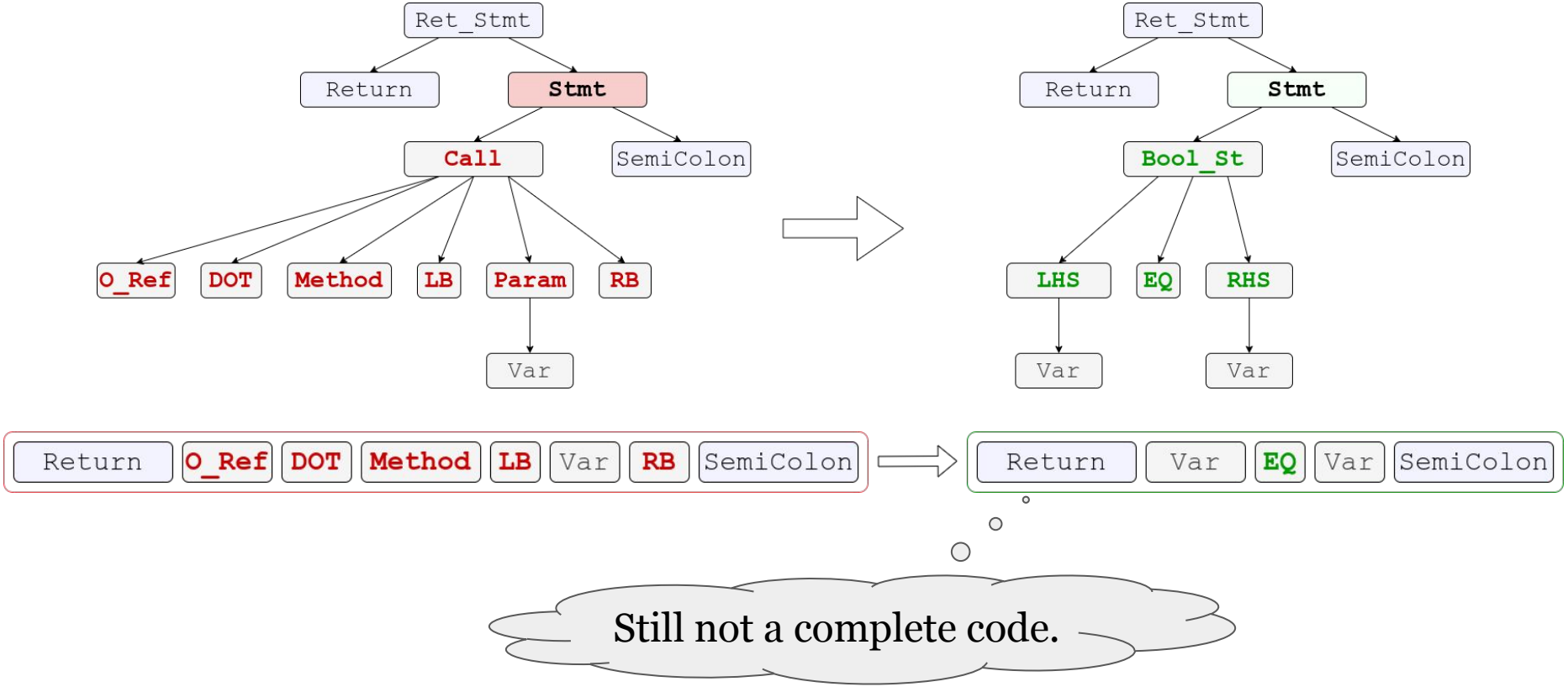
```
Ret_Stmt ──→ Return Stmt
  Stmt   ──→ Bool_St Semicolon
Bool_St  ──→ LHS EQ RHS
  LHS    ──→ VAR
  RHS    ──→ VAR
```



CODIT: Code Editing With Tree Based Neural Models



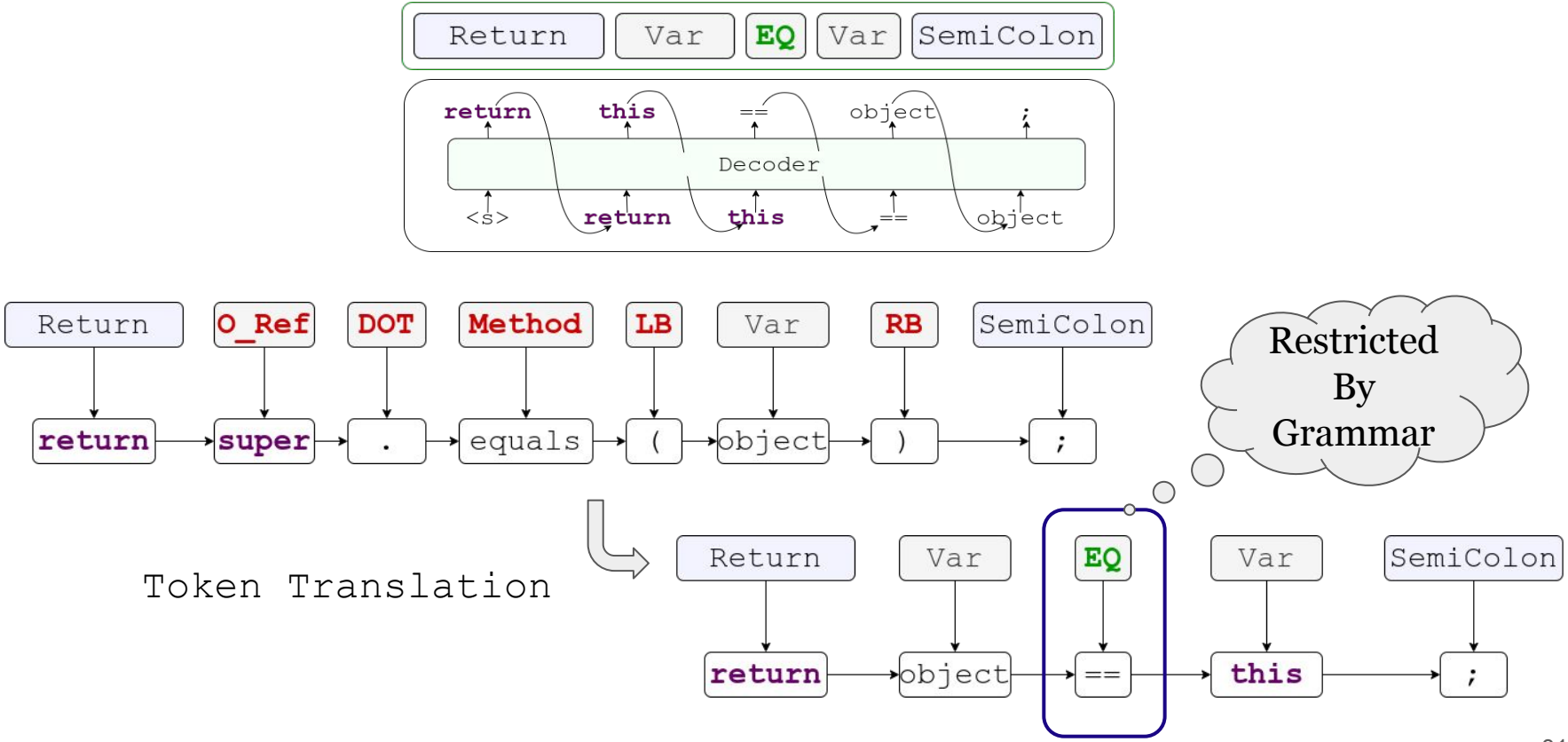
CODIT: Code Editing With Tree Based Neural Models



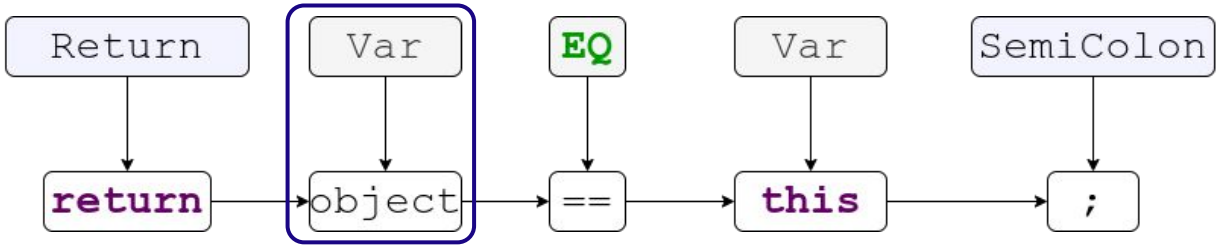
CODIT Step 2

Token Generation

CODIT: Code Editing With Tree Based Neural Models



CODIT: Code Editing With Tree Based Neural Models




```
public boolean checkEqual(
            Object inst, MyClass object){
    MyClass tmp = new MyClass();
    if (inst == null) {
        MyClass tmp2 = new MyClass();
        ...
    }
    return super.equals(object);
}
```

Edit Location

Reachable Variables in Edit Location:

{inst, object, tmp}

CODIT

Evaluation

CODIT: Code Editing With Tree Based Neural Models

Study Subjects

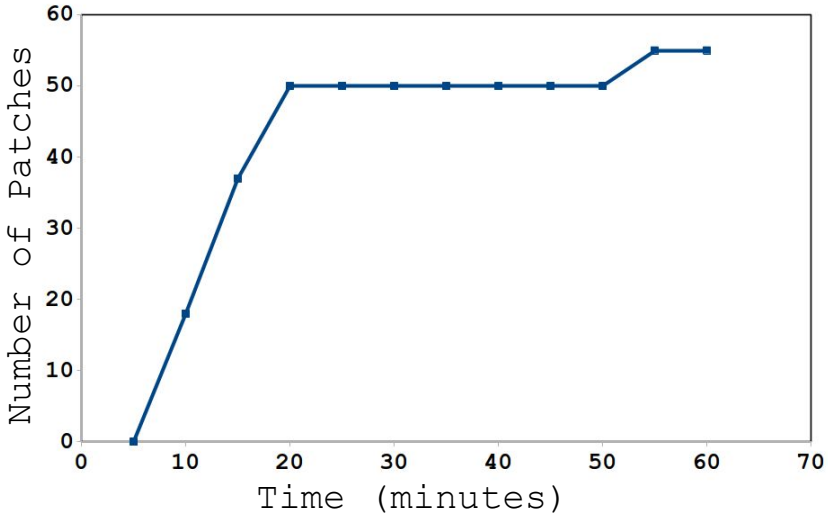
Data Set	**Number of Projects**	**Number of Edit Examples**	**Code Fragment Size**	
			Number of Tokens	**Number of Nodes**
Generic Code Edit from Github	48	32,473	Max - 38 Avg - 15	Max - 47 Avg - 20
Pull Request Edits [Tufano *et. al.* 2019]	3	5546	Max - 34 Avg - 17	Max - 47 Avg - 23

Results (Accuracy in top 5 edited code)

Method		Generic Code Edits	Pull Request Edit
Sequence Based	LSTM-Seq2Seq	3.77%	11.26%
	Tufano et. al.	6.57%	23.65%
	SequenceR	9.76%	26.43%
Tree Based	Tree2Seq	11.04%	23.49%
	CODIT	**15.94%**	**28.87%**

Application - Automatic Program repair

CODIT fixes **15 bugs completely** and **10 bugs partially**, out of 80 bugs in **Defects4j**.



Take Away Point

Generating Syntax Tree Guarantees Syntactic Correctness of Edited Code.

Contextual Correctness: Is it there?

```
boolean checkEqual (Object target)  {
    for(Object elem : this.elements) {
        if (elem.equals(checkEqual)) {
            return true;
        }
    }
    return false;
}
```

Wrong Identifier

```
boolean checkEmpty (Object target){
    if(target == null){
        return null;
    }
    return false;
}
```

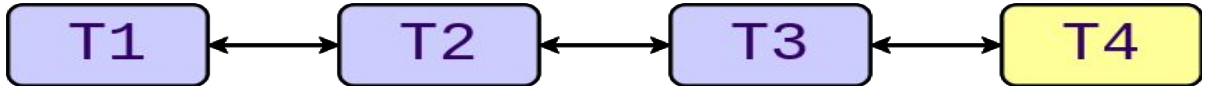
Wrong Return Type

```
public void setPredecessor(Message m) {
    ...
    Message sent = new Message();
    sent.setTo(m.orgin);
-   sendMessage(m);
+   return sendMessage(m);
}
```

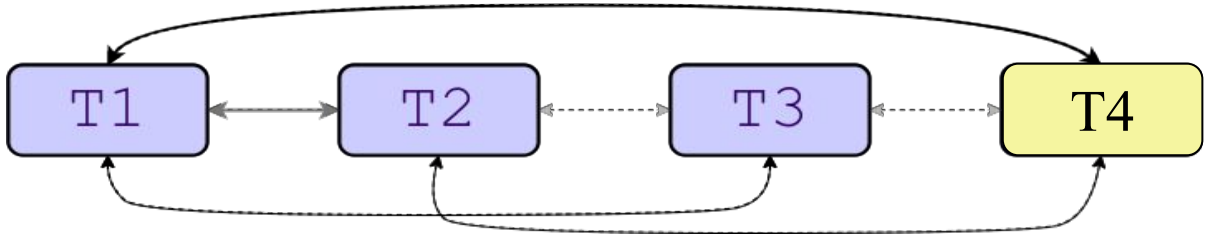
Wrong return type assumption

Recurrent Model Vs. Transformer Model

1. Recurrent Model



2. Transformer Model



Is it possible to learn
Contextual Correctness?

Unified Pre-training for Program Understanding and Generation (PLBART)

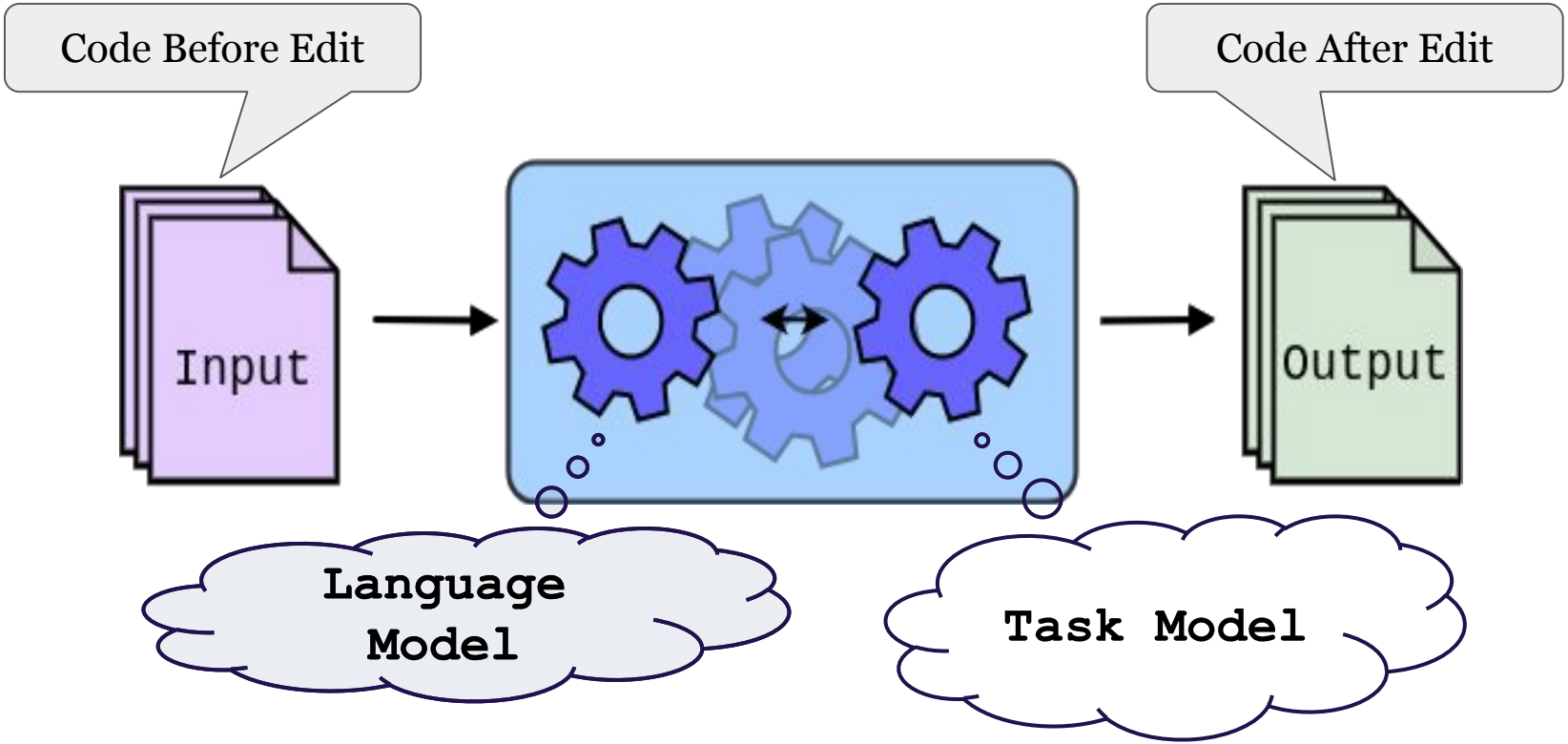
NAACL - 2021

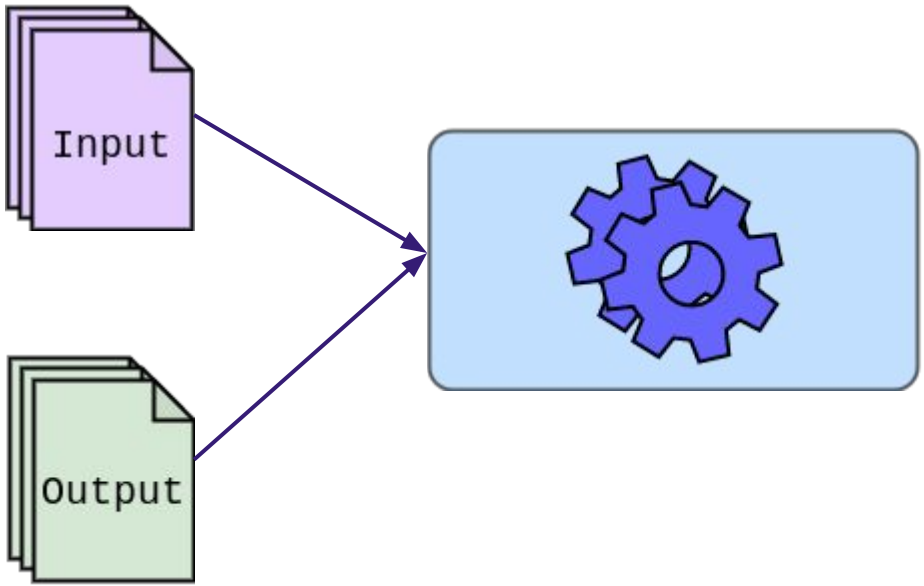
Findings

Contextual Correctness can be learned from large scale dataset.

Contribution

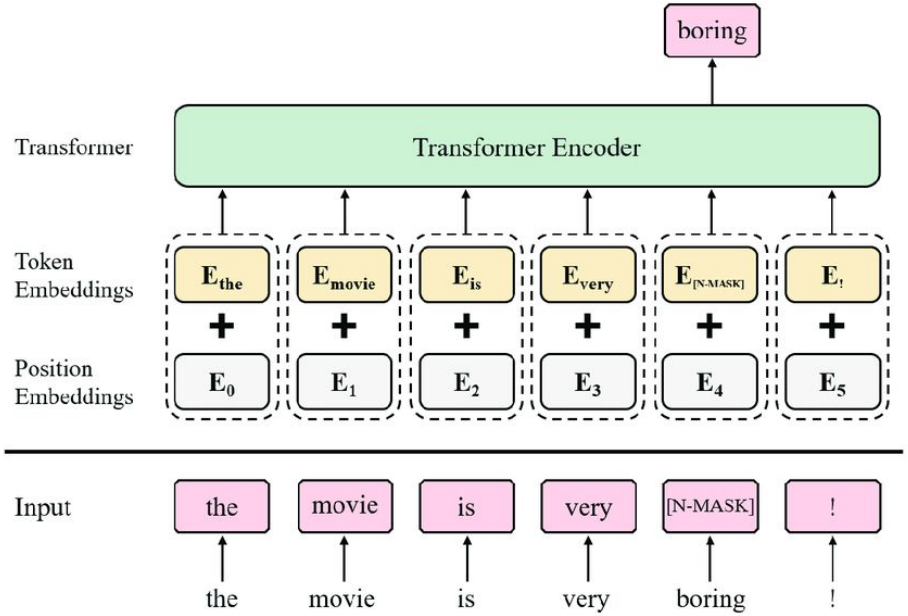
Developed large scale pre-trained models for different SE tasks.


Code Before Edit
Code After Edit
Input
Output
Language Model
Task Model



Pretraining - Embed the knowledge of **input and output** into the model.

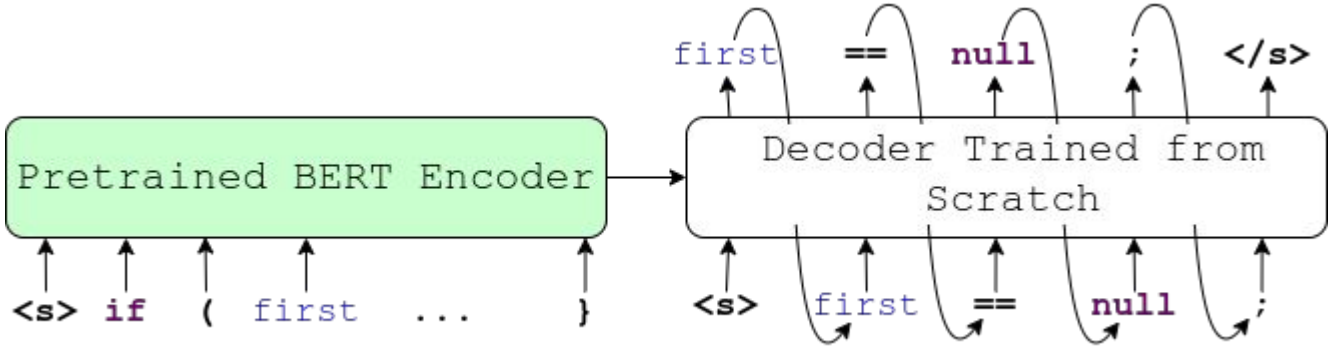
BERT - Pretrained Transformer Encoder



Pre-training:
Task agnostic Masked Language Model.

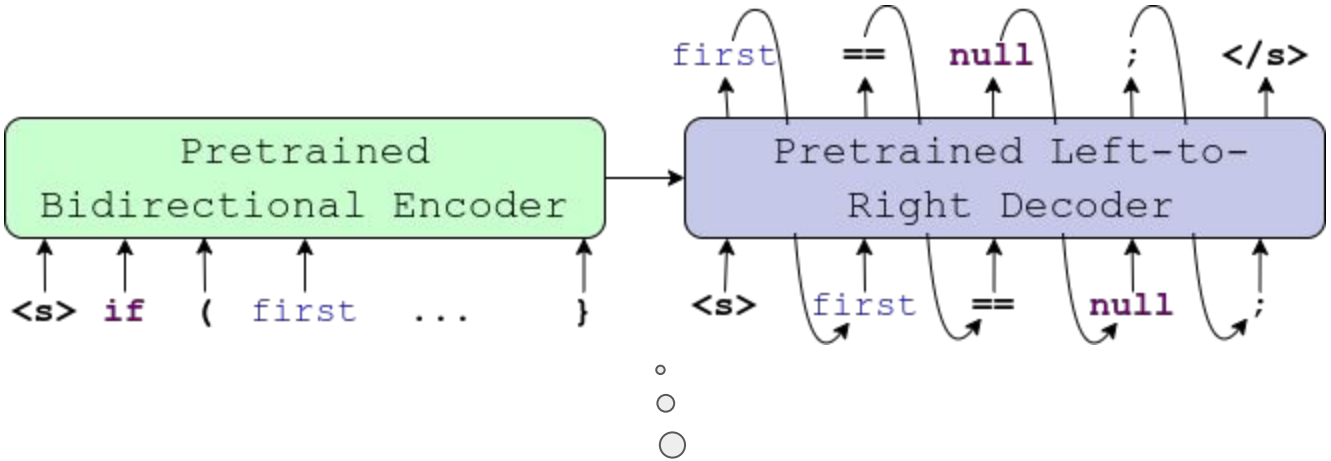
1. CuBERT - Kanade *et. al.* 2020.
2. CodeBERT - Feng *et. al.* 2020.
3. GraphCodeBERT - Guo *et. al.* 2021

SE usage: Better **Suitable** for **Understanding Code**.



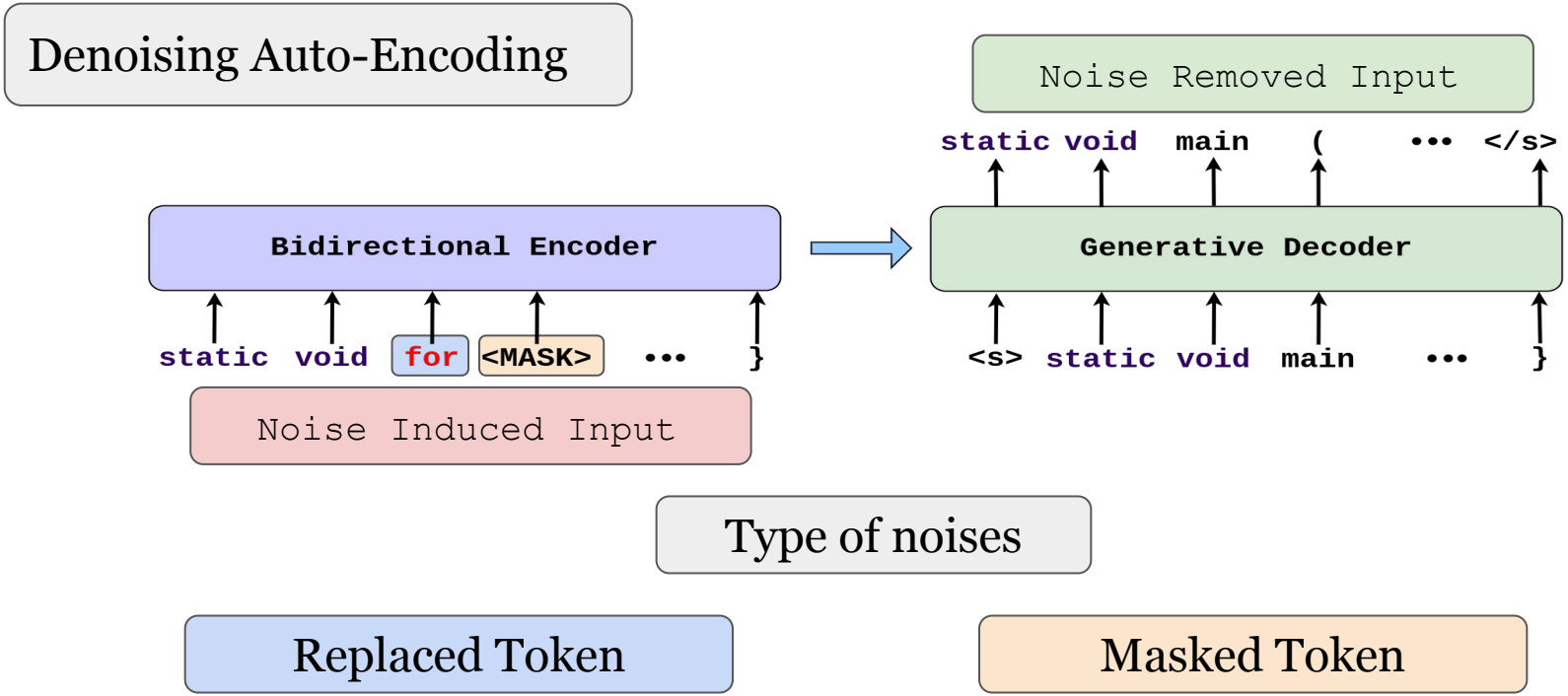
Knowledge about **generation** is **not** embedded in Decoder.

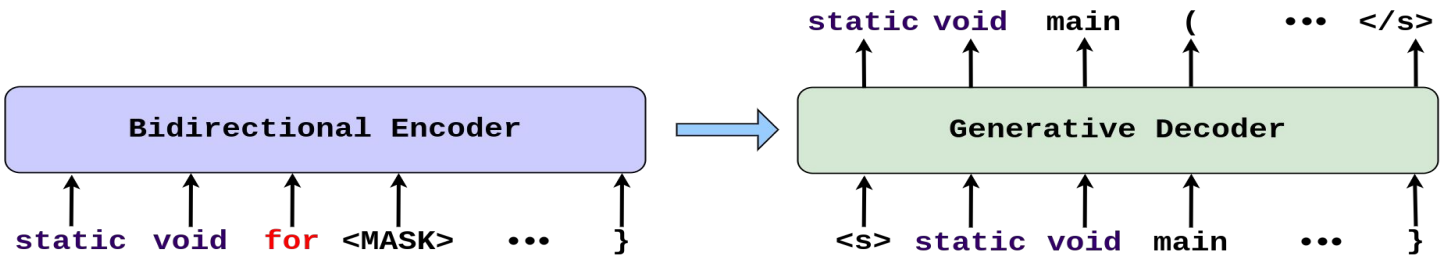
PLBART - Pretraining both encoder and decoder.

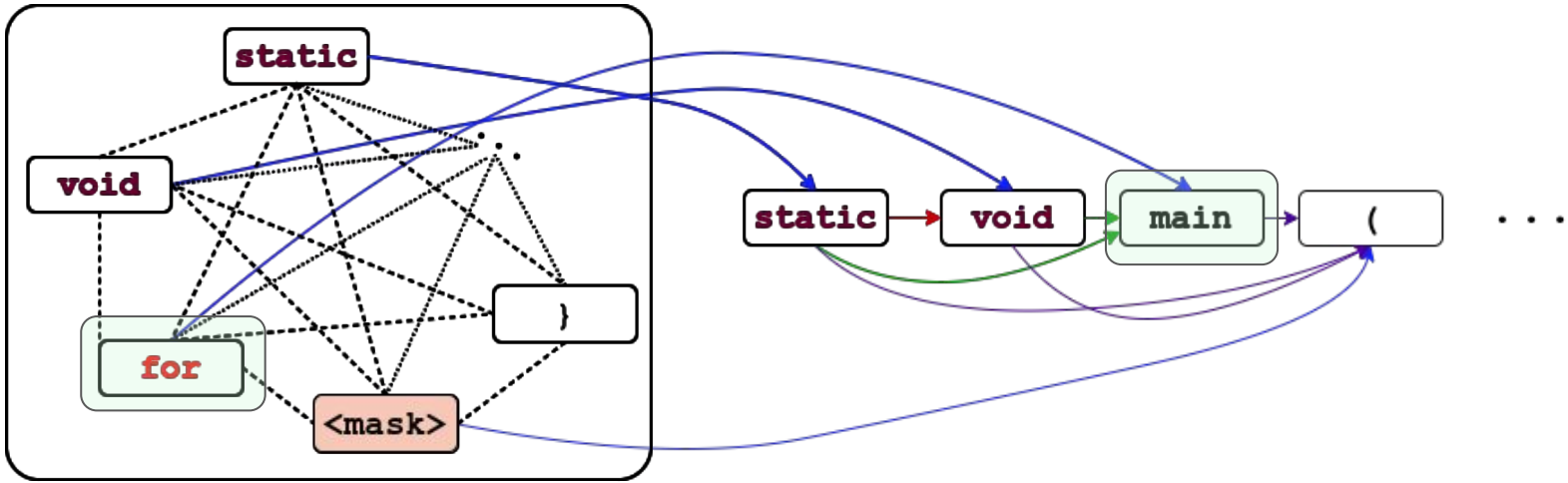


How do we train PLBART?

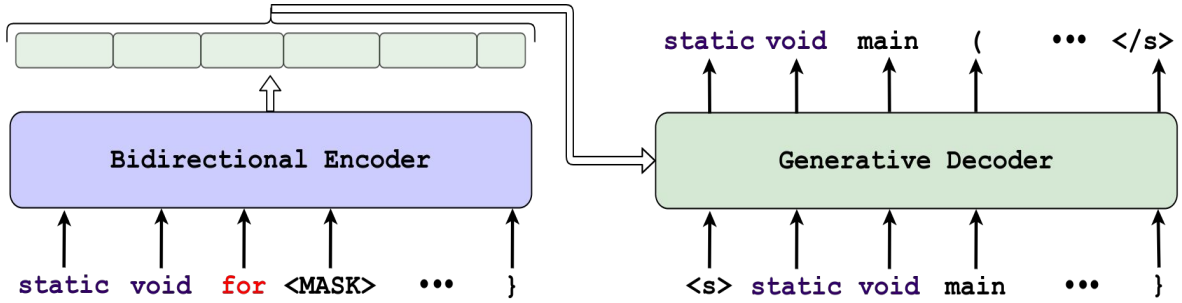
Unified Pre-training for Program Understanding and Generation




static void main (••• </s>
Bidirectional Encoder
Generative Decoder
static void for <MASK> ••• }
<s> static void main ••• }


static
void
}
for
<mask>
static
void
main
(
. . .

PLBART - what is it learning?



Encoder

- Learns the input structure implicitly.
- Learns long range dependencies.
- Learns to generate representations based on these features.

Decoder

- Learns to generate correct code
- Learn to reason about any missing and misplaced token in input.

PLBART

Evaluation

Downstream Tasks

Task	Dataset	Language	Train	Valid	Test
Summarizaion	Husain et al. (2019)	Ruby	24,927	1,400	1,261
		Javascript	58,025	3,885	3,291
		Go	167,288	7,325	8,122
		Python	251,820	13,914	14,918
		Java	164,923	5,183	10,955
		PHP	241,241	12,982	14,014
Generation	Iyer et al. (2018)	NL to Java	100,000	2,000	2,000
Translation	Code-Code (Lu et al., 2021)	Java to C#	10,300	500	1,000
		C# to Java	10,300	500	1,000
	Program Repair (Tufano et al., 2019)	Java_{small}	46,680	5,835	5,835
		Java_{medium}	52,364	6,545	6,545
Classification	Vulnerability Detection (Zhou et al., 2019)	C/C++	21,854	2,732	2,732
	Clone Detection (Wang et al., 2020)	Java	100,000	10,000	415,416

PLBART - Code Translation Result

Dataset: CodeXGLUE [Lu *et. al.* 2021]
Metric: EM (exact match %), **BLEU-4**(max 100), **CodeBLEU**(max 100)

CodeBLEU =
0.25 * token_match +
0.25 * keywords_match +
0.25 * syntax_match +
0.25 * dataflow_match

Methods	Java to C#			C# to Java		
	BLEU	EM	CodeBLEU	BLEU	EM	CodeBLEU
Naive Copy	18.54	0	34.20	18.69	0	43.04
PBSMT	43.53	12.50	42.71	40.06	16.10	43.48
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**

Code Editing (Bug Fixing) Result

Dataset: Tufano *et. al.* 2019
Metric: EM (exact match %), **BLEU-4**(max 100), **CodeBLEU**(max 100)

CodeBLEU =
0.25 * token_match +
0.25 * keywords_match +
0.25 * syntax_match +
0.25 * dataflow_match

Methods	Small		Medium	
	EM	BLEU	EM	BLEU
Naive Copy	0	78.06	0	90.91
Seq2Seq	10.00	76.76	2.50	72.08
Transformer	14.70	77.21	3.70	89.25
CodeBERT	16.40	77.42	5.16	**91.07**
GraphCodeBERT	17.30	**80.58**	**9.10**	72.64
PLBART	**19.21**	77.02	8.98	88.50

Code Translation Example

Input Code : C#

```
1  public int GetCells(){
2      int size = 0;
3      foreach (char c in cells.Keys){
4          Cell e = At(c);
5          if (e.cmd >= 0 || e.@ref >= 0){
6              size++;
7          }
8      }
9      return size;
10 }
```

Generated Code : Java

```
1  public int getCells() {
2      Iterator<Character> i =
3        cells.keySet().iterator();
4      int size = 0;
5      for (; i.hasNext();) {
6          Character c = i.next();
7          Cell e = at(c);
8          if (e.cmd >= 0 || e.ref >= 0){
9              size++;
10         }
11     }
12     return size;
13 }
```

Take Away Point

Large scale model and pre-training for code generation help generate contextually correct code.

What's Next?

PLBART "**Learns**" Contextual Correctness.
Does not "**Enforce**" like CODIT does.
Further research on enforcing Contextual Correctness.

Thanks

Questions?