An Empirical Investigation into Learning Bug-Fixing Patches in the Wild via Neural Machine Translation

Michele Tufano, Cody Watson, Gabriele Bavota, Massimiliano Di Penta, Martin White, Denys Poshyvanyk
Can you translate buggy code into fixed code?
Can you translate buggy code into fixed code?

```java
public void addElement ( Element <> elem) {
    myList.add(elem);
}
```

```java
public void addElement ( Element <> elem) {
    if (!myList.contains(elem))
        myList.add(elem);
}
Why?

Automated Program Repair is (arguably) one of the most exciting research problem in SE.

//TODO - <insert example of bug which costed a lot of money>

//TODO - <sentence about testing and fixing being expensive>
How?

Via Neural Machine Translation by Learning from past mistakes (historical bug-fixes)
How?

Via Neural Machine Translation by Learning from past mistakes (historical bug-fixes)

~10M million Bug-fixing commits
Overview

1. Bug-fixes mining
2. Transformation Pairs extraction
3. Code Abstraction
4. NMT (Encoder-Decoder) training
Bug-Fixes Mining

Finally fixed that bug!

Archive

Regex for Comments

(“fix” or “solve”) AND
(“bug” or “issue” or “problem” or “error”)
Bug-Fixes Mining

Finally fixed that bug!

Archive

Regex for Comments

("fix" or "solve") AND
("bug" or "issue" or "problem" or "error")

Total Commits

10,056,052
Bug-Fixes Mining

Finally fixed that bug!

Archive

Regex for Comments

("fix" or "solve") AND
("bug" or "issue" or "problem" or "error")

<table>
<thead>
<tr>
<th>Total Commits</th>
<th>Java Commits</th>
</tr>
</thead>
<tbody>
<tr>
<td>10,056,052</td>
<td>787,178</td>
</tr>
</tbody>
</table>

No more than 5 files
Pair Extraction

Finally fixed that bug!

Buggy File

Fixed File
Pair Extraction

Finally fixed that bug!

GumTreeDiff
Method mapping
Semantic Anchors

Buggy File

Fixed File
Finally fixed that bug!
Finally fixed that bug!

GumTreeDiff

Method mapping
Semantic Anchors

Method-level
AST Diff

Buggy File

Translation

Fixed File

public void addElement ( element?: elem ) { 
  myList.add(elem), 
}

public void addElement ( element?: elem ) { 
  if (myList.contains(elem)) 
    myList.add(elem), 
}
Code Abstraction

Goal: reduce Vocabulary

Source Code

```java
public void addElement ( Element <?> elem) { if ( myList.size() > 0) { myList.add(elem); } }
```
Code Abstraction

Goal: reduce Vocabulary

Source Code

```java
public void addElement ( Element <?> elem) { if ( myList.size() > 0) { myList.add(elem); } }
```

Abstracted code

```java
public void ( <?> ) { if ( ) { ( ); } }
```

- Java Keywords and separators
Code Abstraction

Goal: reduce Vocabulary

Source Code

```java
public void addElement ( Element<?> elem) { if ( myList.size() > 0) { myList.add(elem); } }
```

Abstracted code

```java
public void (<?> ) { if ( .size() > 0) { .add(); } }
```

- Java Keywords and separators
- Idioms: frequent identifiers and literals (e.g. size, add, 0)
Code Abstraction

Goal: reduce Vocabulary

Source Code

```java
public void addElement ( Element<?, ?> elem) { if ( myList.size() > 0) { myList.add(elem); } }
```

Abstracted code

```java
public void METHOD_1 ( TYPE_1<?, ?> VAR_1) { if ( VAR_2.size() > 0) { VAR_2.add(VAR_1); } }
```

- Java Keywords and separators
- Idioms: frequent identifiers and literals (e.g. size, add, 0)
- IDs: replace identifiers and literals with typified IDs (e.g., METHOD, TYPE, VAR, INT, STRING, etc.)
Learning Fixes

Recurrent Neural Network (RNN) Encoder-Decoder

Encoder RNN

h₁  h₂  hₙ

Attention

Ci

Decoder RNN

Softmax

[Ci, Si]

S₁  S₂  ...  Sₘ

RNN Cell (GRU)

X₁  X₂  ...  <end>

abstract_b (buggy)

<start>  y₁  ...  yₘ

abstract_f (fixed)
Learning Fixes

Hyperparameters
10 configurations

RNN Cells
- LSMT
- GRU

Layers
- 1
- 2
- 4

Units
- 256
- 512

Embedding Size
- 256
- 512

Recurrent Neural Network (RNN) Encoder-Decoder

Encoder RNN
- $h_1$
- $h_2$
- $h_n$

Attention
- $C_i$

Decoder RNN
- Softmax
- $[C_i, S_i]$
- $S_1$
- $S_2$
- $S_m$

Input:
- $x_1$
- $x_2$
- $\text{<end>}$

Output:
- $\text{abstract}_b$ (buggy)
- $\text{abstract}_f$ (fixed)
Evaluation

Small Methods
No longer than 50 tokens

Dataset: 58,350 methods

- 80% Training
- 10% Validation
- 10% Test

No duplicates
Unique at source and abstracted code level
Buggy Code

```java
public void addElement ( Element <?> elem) { myList.add(elem); }
```
Buggy Code

```java
public void addElement ( Element <?> elem) { myList.add(elem); }
```

Abstracted Buggy Code

```java
public void METHOD_1 ( TYPE_1 <?> VAR_1) { VAR_2.add(VAR_1); }
```

Mapping

<table>
<thead>
<tr>
<th>ID</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>METHOD_1</td>
<td>addElement</td>
</tr>
<tr>
<td>TYPE_1</td>
<td>Element</td>
</tr>
<tr>
<td>VAR_1</td>
<td>elem</td>
</tr>
<tr>
<td>VAR_2</td>
<td>myList</td>
</tr>
</tbody>
</table>
Buggy Code

```java
public void addElement ( Element <?> elem) { myList.add(elem); }
```

Abstracted Buggy Code

```java
public void METHOD_1 ( TYPE_1 <?> VAR_1) { VAR_2.add(VAR_1); }
```

Neural Machine Translation

```
NMT
```

Mapping

<table>
<thead>
<tr>
<th>ID</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>METHOD_1</td>
<td>addElement</td>
</tr>
<tr>
<td>TYPE_1</td>
<td>Element</td>
</tr>
<tr>
<td>VAR_1</td>
<td>elem</td>
</tr>
<tr>
<td>VAR_2</td>
<td>myList</td>
</tr>
</tbody>
</table>
Buggy Code

```java
public void addElement ( Element<?> elem) { myList.add(elem); }
```

Abstracted Buggy Code

```java
public void METHOD_1 ( TYPE_1<?> VAR_1) { VAR_2.add(VAR_1); }
```

Neural Machine Translation

Abstracted Fixed Code

```java
public void METHOD_1 ( TYPE_1<?> VAR_1) { if (! VAR_2.contains(VAR_1)) VAR_2.add(VAR_1); }
```

Mapping

<table>
<thead>
<tr>
<th>ID</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>METHOD_1</td>
<td>addElement</td>
</tr>
<tr>
<td>TYPE_1</td>
<td>Element</td>
</tr>
<tr>
<td>VAR_1</td>
<td>elem</td>
</tr>
<tr>
<td>VAR_2</td>
<td>myList</td>
</tr>
</tbody>
</table>
Buggy Code

```java
public void addElement ( Element <?> elem) { myList.add(elem); }
```

Abstracted Buggy Code

```java
public void METHOD_1 ( TYPE_1 <?> VAR_1) { VAR_2.add(VAR_1); }
```

Neural Machine Translation

```
But can you generate real source code?
```

Abstracted Fixed Code

```java
public void METHOD_1 ( TYPE_1 <?> VAR_1) { if (! VAR_2.contains(VAR_1)) VAR_2.add(VAR_1); }
```

Mapping

<table>
<thead>
<tr>
<th>ID</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>METHOD_1</td>
<td>addElement</td>
</tr>
<tr>
<td>TYPE_1</td>
<td>Element</td>
</tr>
<tr>
<td>VAR_1</td>
<td>elem</td>
</tr>
<tr>
<td>VAR_2</td>
<td>myList</td>
</tr>
</tbody>
</table>
But can you generate real source code?

**YES!**

*Enlarged font for Reviewers*
Buggy Code

```java
public void addElement ( Element <?> elem) { myList.add(elem); }
```

Abstracted Buggy Code

```java
public void METHOD_1 ( TYPE_1 <?> VAR_1) { VAR_2.add(VAR_1); }
```

Neural Machine Translation

Abstracted Fixed Code

```java
public void METHOD_1 ( TYPE_1 <?> VAR_1) { if (! VAR_2.contains(VAR_1)) VAR_2.add(VAR_1); }
```

Fixed Code

```java
public void addElement ( Element <?> elem) { if (! myList.contains(elem)) myList.add(elem); }
```
Results

538 bug-fixes*

9.22% test set

*Unique at source and abstract code level.
*Never seen in training nor validation set.
Results

538 bug-fixes*
9.22% test set

*Unique at source and abstract code level.
*Never seen in training nor validation set.
Can you generate multiple candidate patches?

50 different translations?
Can you generate multiple candidate patches?

50 different translations?

With more candidates... can you fix 30-50% of the bugs?
Can you generate multiple candidate patches?

With more candidates... can you fix 30-50% of the bugs?

What type of AST operations?

50 different translations?
Can you generate multiple candidate patches?

With more candidates... can you fix 30-50% of the bugs?

What type of AST operations?

50 different translations?

Longer methods?
Can you generate multiple candidate patches?

Code?

What type of AST operations?

50 different translations?

With more candidates... can you fix 30-50% of the bugs?

DATA!!

Generation time?

Longer methods?
Can you generate multiple candidate patches?

50 different translations?

DATA!!

Journal submission coming very soon...

What type of AST operations?

Can you fix 30-50% of the bugs?

Patch generation time?
Thanks!

Questions?

Michele Tufano
@tufanomichele
http://www.cs.wm.edu/~mtufano/

An Empirical Investigation into Learning Bug-Fixing Patches in the Wild via Neural Machine Translation