Towards Data Augmentation for Supervised Code Translation

Binger Chen[1*], Jacek Golebiowski[2], and Ziawasch Abedjan[3]
1 Technische Universität Berlin
2 Amazon AWS
3 Leibniz Universität Hannover & L3S Research Center
* chen@tu-berlin.de

Problem

Supervised Code Translation
- High accuracy ☺
- Lack of quality training data ☹

Current Mehtods

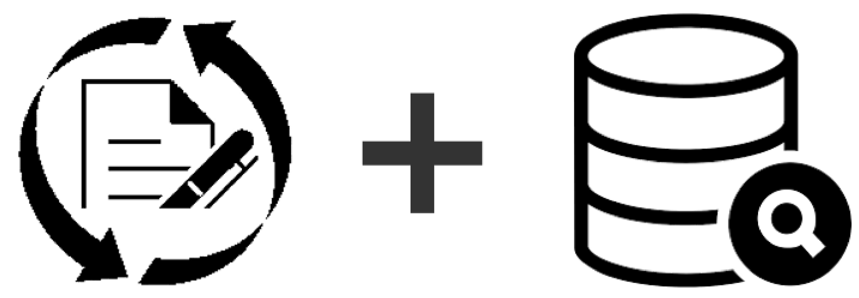
Data augmentation for *natural languages*

Data augmentation for *bug fixing*

Code Translation

Our Solution

Adapted rules + Retrieving from big data



Method 1: Rule-based data augmentation

Goal: Enhance code translation pairs through strategic transformations, ensuring functional and semantic integrity.

Transformation Rules:

1. **Reverse**: Adjusting logic in conditional statements and making corresponding changes in the target language code.

   Example:

   ```
   // original code
   if (x == 5)

   // transformed code
   if (x != 5)
   ```

2. **Merge**: Combining code at the control structure or program level, using logical operators or concatenation.

   Example:

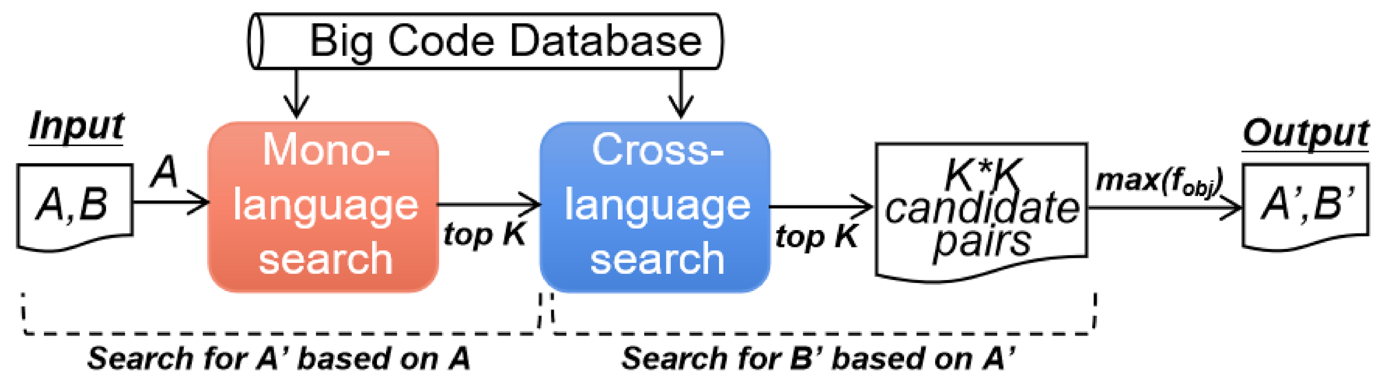
   ```
   // original code
   if (x > 5) { i ++; }
   if (y < 10) { System.out.println("pass"); }

   // transformed code
   if (x > 5 && y < 10) {
       i ++;
       System.out.println("pass");
   }
   ```

3. **Split**: Separating combined conditional statements.
4. **All**: A composite application of the above rules.

Application: These methods, derived from bug-fixing techniques, apply reversible changes and adapt to a range of programming languages, preserving logical semantics between source and target codes.

Method 2: Retrieval-based data augmentation (CTAug)

Workflow of CTAug:



Goal: Discover new and unseen code pairs in Big Code repositories to augment training data for code translation.
Method: Utilizes cross-language and mono-language retrieval methods for identifying potential translations and similar code snippets.
Process Overview:

1. Mono-Language Search: Finds top-k code snippets similar to the given code.
2. Cross-Language Search: Identifies potential translations, creating a $K \times K$ candidate set.

Optimization Method:
Proposes a sophisticated selection method ensuring similarity in multiple dimensions.

- *Selection Criteria:* Considering the original training pair (A, B) and a sample (A', B') from the $K \times K$ candidate set:
  1. $f_1(A', B') = S(A', B')$: Similarity within new code pairs.
  2. $f_2(A', B') = S(A', A) + S(B', B)$: Similarity between the new pair and original code context.
  3. $f_3(A', B') = -|S(A', A) - S(B', B)|$: Balancing similarity measures to ensure relevance and diversity.
- *Optimization Goal:* Formalizes task as a multi-objective optimization problem.
- *Objective function* to maximize selection criteria:

$$\mathbf{max} f_{obj} = \alpha \mathbf{max} f_1(A', B') + \beta \mathbf{max} f_2(A', B') + \gamma \mathbf{max} f_3(A', B')$$

Experiment

Datasets & Methodology:

- Utilized two key datasets: original Tree2tree translation datasets [1] and a "Big Code" database from GitHub's Public Git Archive.

Tree2tree Dataset	Lucene	POI	IText	JGit	JTS	ANTLR
# of data pairs	5516	3153	3079	2780	2003	465

- We focused on translating Java to C#, comparing data-augmentation strategies against NLP baselines using the Tree2tree model.
- We used RPT [2] as mono-language translation system and BigPT [3] as cross-language translation system.

Baselines:

- Compared our data-augmentation strategies against NLP baselines due to the lack of existing methods in supervised code translation.
- Baselines include Word Masking and Back-translation, implemented with adjustments for code data.

Metrics:

- Program Accuracy (PA): Percentage of predicted translations matching the ground truth, emphasizing semantic equivalence.
- Token Accuracy (TA): Matches the percentage of tokens to the ground truth, offering insights into textual alignment.
- CodeBLEU: Adapts BLEU score for code translation, evaluating syntax and semantics through n-gram, AST, and data-flow matching.

Results

Dataset	w/o Aug	CTAug	NLP Methods		Rule-based methods			
			WM	*BT*	*Reverse*	*Merge*	*Split*	*All*
Metric: Program Accuracy								
Lucene	72.8%	**85.8%**	63.1%	70.2%	72.7%	72.0%	72.8%	75.9%
POI	72.2%	**86.1%**	62.4%	69.9%	73.6%	70.3%	72.5%	75.1%
IText	67.5%	**83.2%**	61.5%	66.3%	68.9%	66.9%	67.8%	72.8%
JGit	68.7%	**81.7%**	59.3%	65.4%	70.3%	70.2%	69.1%	72.8%
JTS	68.2%	**82.3%**	61.2%	67.7%	70.9%	69.8%	69.6%	73.3%
ANTLR	31.9%	**66.2%**	25.3%	31.9%	33.6%	36.7%	32.7%	40.1%
Metric: Token Accuracy								
Lucene	85.3%	**92.3%**	80.3%	87.7%	88.3%	87.1%	88.9%	90.1%
POI	84.8%	**94.8%**	81.1%	78.2%	87.2%	85.2%	87.3%	91.7%
IText	80.3%	**93.3%**	77.2%	83.1%	81.3%	79.8%	79.2%	85.6%
JGit	81.7%	**88.9%**	73.7%	82.3%	84.2%	82.5%	82.5%	88.2%
JTS	82.1%	**90.2%**	72.1%	82.8%	83.5%	81.2%	83.6%	89.7%
ANTLR	70.2%	**80.1%**	57.2%	69.2%	73.8%	75.7%	74.9%	78.3%
Metric: CodeBLEU								
Lucene	87.6%	**95.3%**	84.4%	91.2%	92.8%	89.5%	92.6%	94.2%
POI	88.6%	**96.7%**	85.4%	81.7%	90.0%	88.9%	91.3%	94.5%
IText	84.8%	**96.5%**	81.3%	87.8%	85.6%	83.5%	84.3%	89.2%
JGit	85.1%	**92.4%**	78.9%	86.3%	88.7%	87.2%	85.9%	91.6%
JTS	86.7%	**93.6%**	75.7%	87.1%	88.0%	84.1%	85.8%	93.1%
ANTLR	75.2%	**83.9%**	62.2%	73.3%	78.1%	80.1%	77.6%	82.8%

- Retrieval-Based Method: Enhanced accuracy significantly, demonstrating robust improvement over traditional methods.
- Rule-Based Augmentation: Showed modest enhancements, indicating limitations in diversifying training data.
- NLP Baselines: Underperformed, highlighting the unique challenges of code translation.
- Metrics: Revealed the retrieval-based method's superiority in semantic and syntactic alignment, as evidenced by improved Program Accuracy and CodeBLEU scores.

Example of the results

Input: Java code snippet:

```
public class Example {
  public static void main(String[] args) {
    int x = 5;
    System.out.println(x);
  }
}
```

Output: C# translation of the input code snippet:

```
using System;

public class Example {
  public static void Main() {
    int y = 5;
    Console.WriteLine(y);
  }
}
```

Conclusion & Future work

1. Adapted rules for code translation,
2. A retrieval-based method with optimization strategy,
3. Retrieval-based method is effective and outperforms other methods.

Future work:

- Combine augmentation methods,
- Apply to more languages and models,
- Apply to more tasks, eg. stylization and summary.

References

[1] X Chen, C Liu, D Song: Tree-to-tree Neural Networks for Program Translation. NeurIPS 2018
[2] B Chen, Z Abedjan: RPT: Effective and Efficient Retrieval of Program Translations from Big Code. ICSE (Companion Volume) 2021
[3] B Chen, Z Abedjan: Interactive Cross-language Code Retrieval with Auto-Encoders. ASE 2021