ICLR

Learning Structural Edits via Incremental Tree Transformations

Ziyu Yao†, Frank F. Xu‡, Pengcheng Yin‡, Huan Sun†, Graham Neubig‡

†The Ohio State University ‡Carnegie Mellon University

ICLR 2021

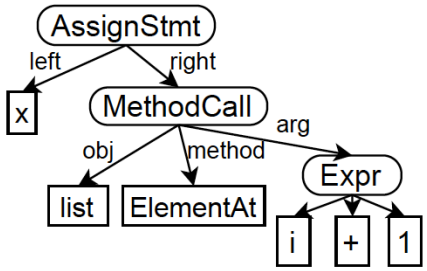
THE OHIO STATE UNIVERSITY

Carnegie Mellon University
Language Technologies Institute

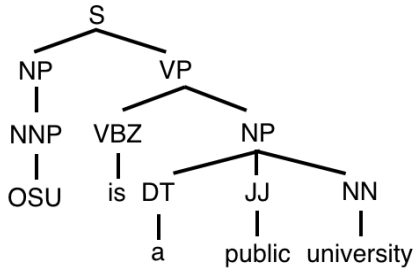
NEULAB

Motivation

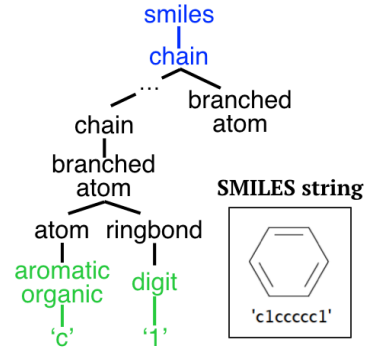
- The prevalence of tree-structured data



Abstract Syntax Trees (AST) of computer programs



Syntactic parse trees of sentences



Parse trees of molecules
(James+ 2015; Kusner+ 2017)

The need for **editing** tree-structured data

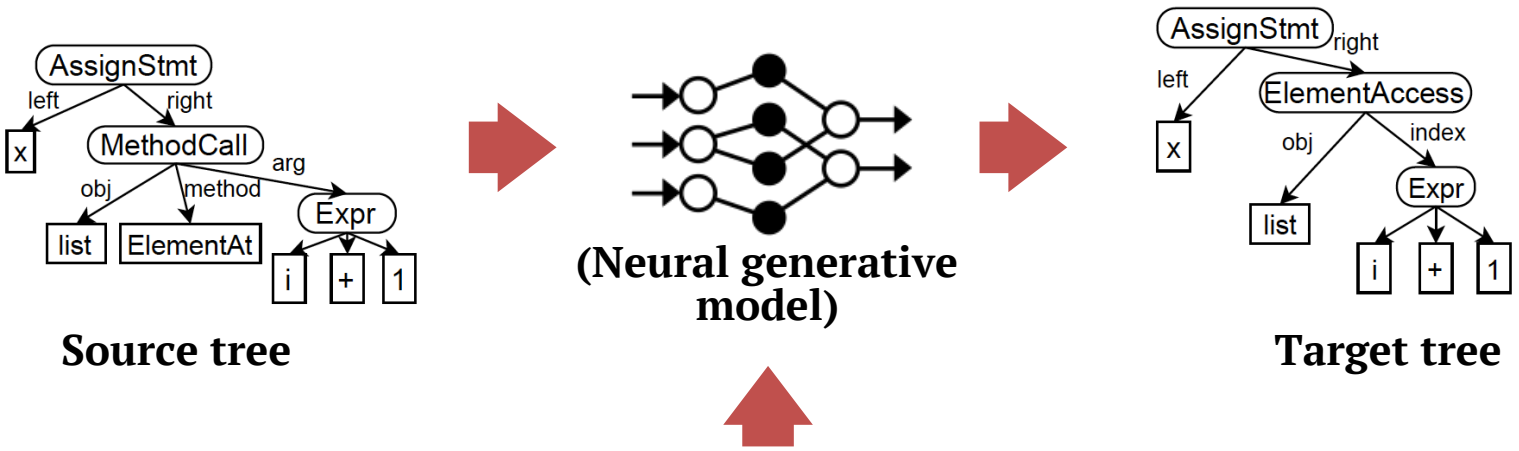
e.g., code refactoring, bug repair, edit-based code generation

e.g., syntax-based sentence simplification

e.g., drug molecule discovery and design

Learning to Edit Tree-Structured Data

- Problem formulation

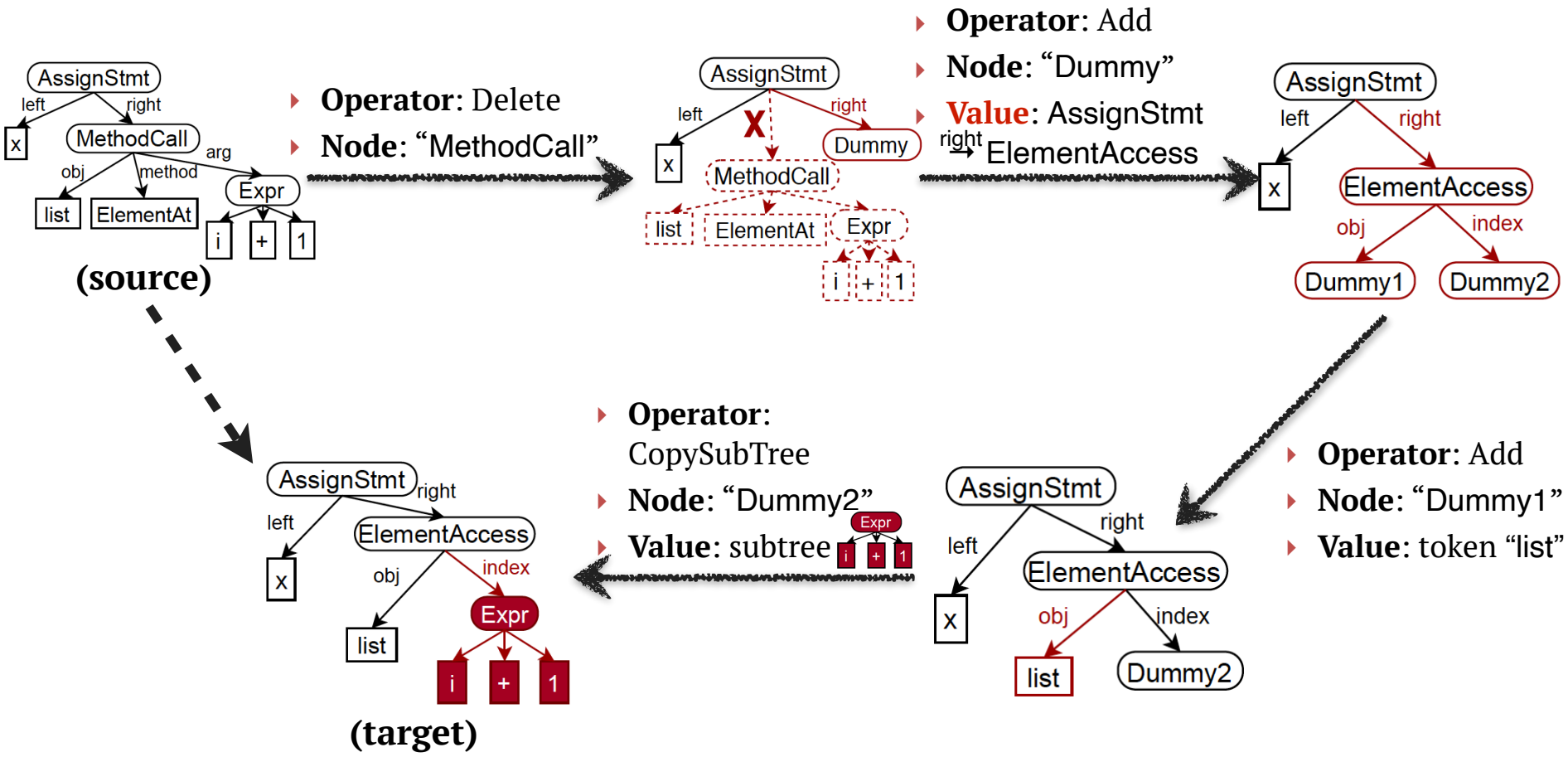


This work: **A general-purpose model for tree editing**

- Generating *incremental tree edit actions* => capable of iterative tree edit
  - vs. generating *the edited tree* (Yin+ 2019; Chakraborty+ 2020)
- Language-agnostic & grammar-valid
  - w/ Abstract Syntax Description Language (ASDL; Wang+ 1997)

Structural Edit via Incremental Tree Transformations

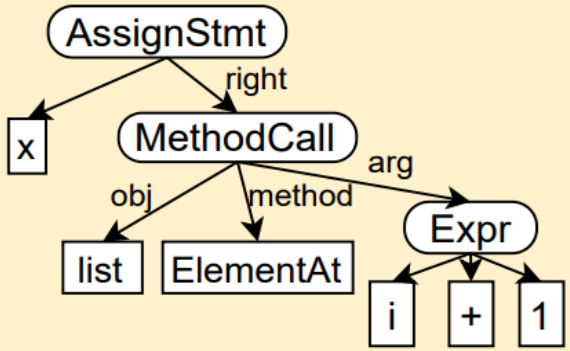
(Using editing the AST of a computer program as a canonical example)



Note: Operator "Stop" is omitted

Structural Edit via Incremental Tree Transformations

Apply edit & transform tree, $t \leftarrow t + 1$



Current tree C_t

+

Edit specification Δ

state s_t → edit action a_t

- **Operator**: [Delete, Add, CopySubTree, Stop]
- **Node**: e.g., "MethodCall" (edit location)
- **Value**: new node to add or subtree from C_1 to copy

$$p(a_{1:T}|\Delta, C_1) = p(a_1|\Delta, C_1)p(a_2|\Delta, C_{1:2}) \cdots p(a_T|\Delta, C_{1:T})$$

Experiment

Evaluation setting: $\langle \Delta = \langle C'_{src}, C'_{tgt} \rangle, C_{src} \rangle \rightarrow C_{tgt}$

(**Δ**: edit specification)

e.g.,

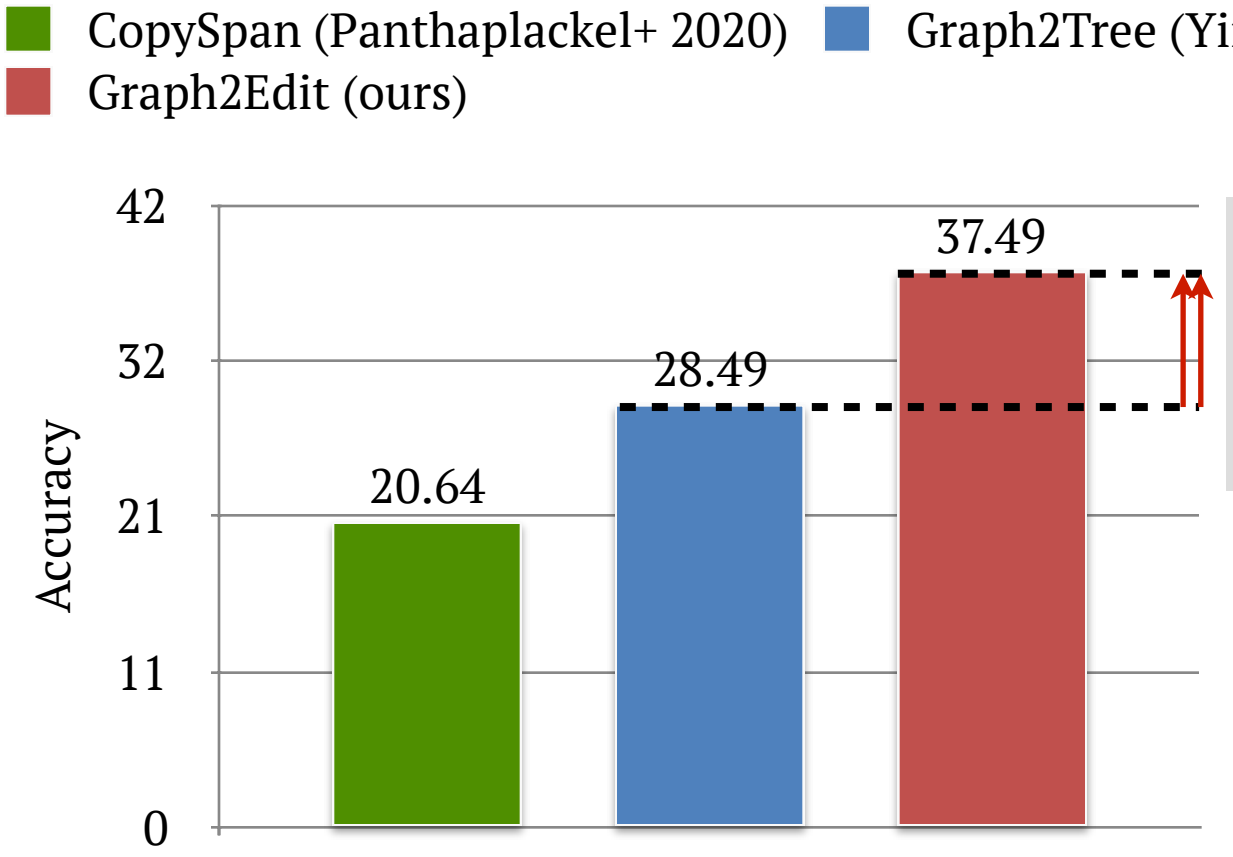
$\langle$ `string VAR0 = VAR1.ElementAt((int)VAR2.id);` , `string VAR0 = VAR1[(int)VAR2.id];` $\rangle$

```
x = list
.ElementAt(i+1)
```

```
x=list[i+1]
```

Experiment

Evaluation setting: $\langle \Delta = \langle C'_{src}, C'_{tgt} \rangle, C_{src} \rangle \rightarrow C_{tgt}$

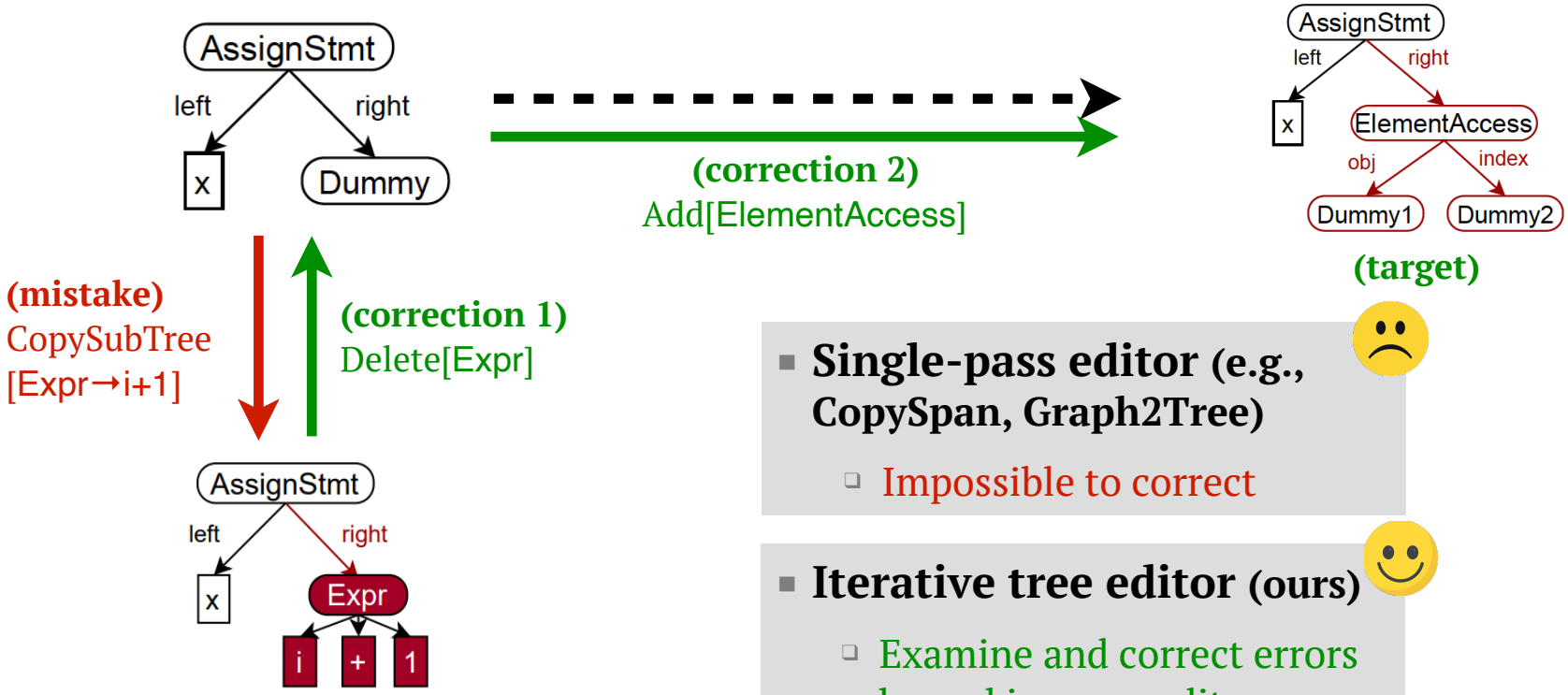


Graph2Edit:
More capable of capturing the generalizable edit intent

w/ Sequential edit encoder; on Fixers dataset (Yin+ 2019)

Robust Tree Edit with Error Correction

- What if the model made a *wrong* edit in *inference time*?



- **Single-pass editor (e.g., CopySpan, Graph2Tree)**
  - Impossible to correct
- **Iterative tree editor (ours)**
  - Examine and correct errors by making new edits

Our contributions

- Training iterative tree editors to correct mistakes by **imitation learning**
- **Take-away**: continuous structural demonstrations are preferable

Summary

- Contributions: How to edit tree-structured data?
  - `Graph2Edit`: A general-purpose neural model for editing tree-structured data iteratively
  - Robust tree editing via imitation learning

  (More in paper:)
  - `TreeDiff`: a novel edit encoder for learning representations of edit specifications

- Code and data available on GitHub: https://github.com/neulab/incremental_tree_edit
- See you at **Poster Session 2 on May 3, 9-11AM PDT**!