Code Representation for Neural Networks and Applications

Tongyu Lu

SiDNN – 10.05.2022

Outline

Relevant Tasks

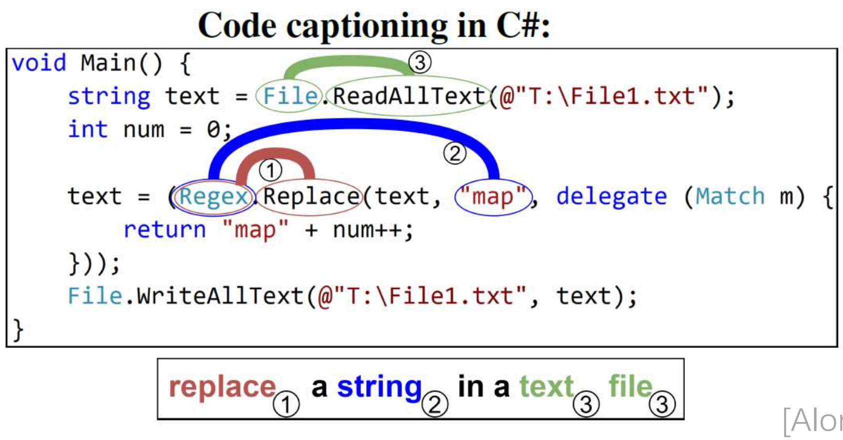
AST Code Representations

AST-based Models

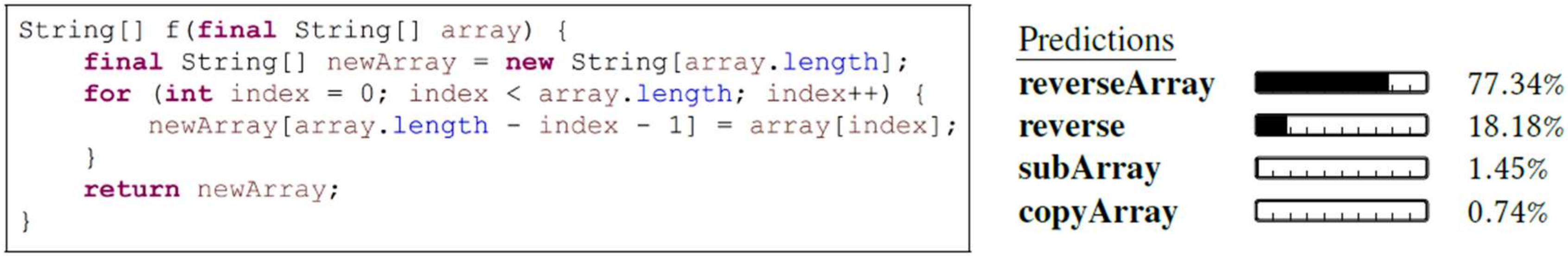
Possible Research Directions

Relevant Tasks

Task Category	Input	Output
Explanation Tasks (code captioning, code summary)	Code snippet	Natural language sequence



[Alon et al. 2018]



Relevant Tasks

Task Category	Input	Output
Explanation Tasks (code captioning, code summary)	Code snippet	Natural language sequence
Information Retrieval Tasks (identifier name search, code search)	Query String (e.g., key-word-to-find, code summary)	Relevant code (e.g., relevant identifiers, relevant code snippets)

A	≈B
size	getSize, length, getCount, getLength
active	isActive, setActive, getIsActive, enabled
done	end, stop, terminate
toJson	serialize, toJsonString, getJson, asJson,
run	execute, call, init, start

Figure from
[Alon et al. 2018]

```
swap two elements in the list
```

(a) Query q

```
void swapElementInList(List<Integer> list, int i, int j) {
    int element = list.get(i);
    list.set(i, list.get(j));
    list.set(j, element);
}
```

(b) Code Snippet s_1

```
void swapElementInList(List<Integer> list, int i, int j) {
    Collections.swap(list, i, j);
}
```

(c) Code Snippet s_2

2022, Sun et al., Code Search based on Context-aware Code Translation, https://arxiv.org/abs/2202.08029

Relevant Tasks

Task Category	Input	Output
Explanation Tasks (code captioning, code summary)	Code snippet	Natural language sequence
Information Retrieval Tasks (identifier name search, code search)	Query String (e.g., key-word-to-find, code summary)	Relevant code (e.g., relevant identifiers, relevant code snippets)
Generation Tasks (code completion, comment to code)	Code snippet (incomplete) or natural language	Code snippet (e.g., a single identifier, a code block)

```
class Operator(Employee):
  def __init__(self, name, employee_id):
    super(Operator, self).__init__(name, Rank.OPERATOR)
    self.employee_id = employee_id

  def _dispatch_call(self, call, employees):
    for employee in employees:
      employee.take_call(call)

  def record_path(self, base_name):
    return os.path.join(base_name, str(self.___?___))
```

Figure from [Li et al. 2017]

Code Representation

Central problem

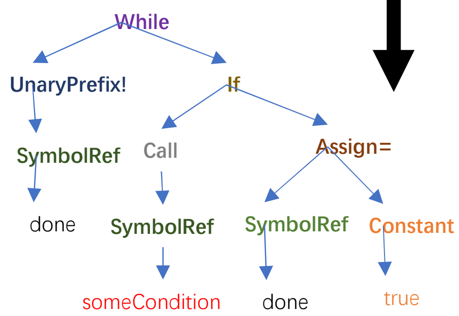
Question: how to feed code to neural networks?

Option 1: NLP approach

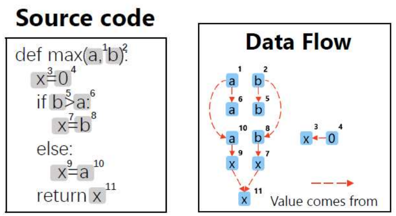
```
while (!done) {
    if (someCondition()) {
        done = true;
    }
}
```

[while, (, !, done,), {, \n, \t, if, (, someCondition, (,),), …]

Option 2: code as syntactic parse tree



Option 3: extract features



Code Representation

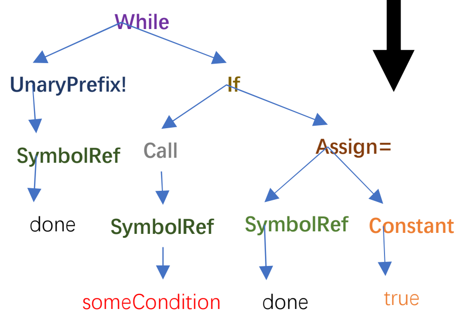
Central problem

Question: how to feed code to neural networks?

Option 1: NLP approach

```
while (!done) {
    if (someCondition()) {
        done = true;
    }
}
```

Option 2: code as syntactic parse tree



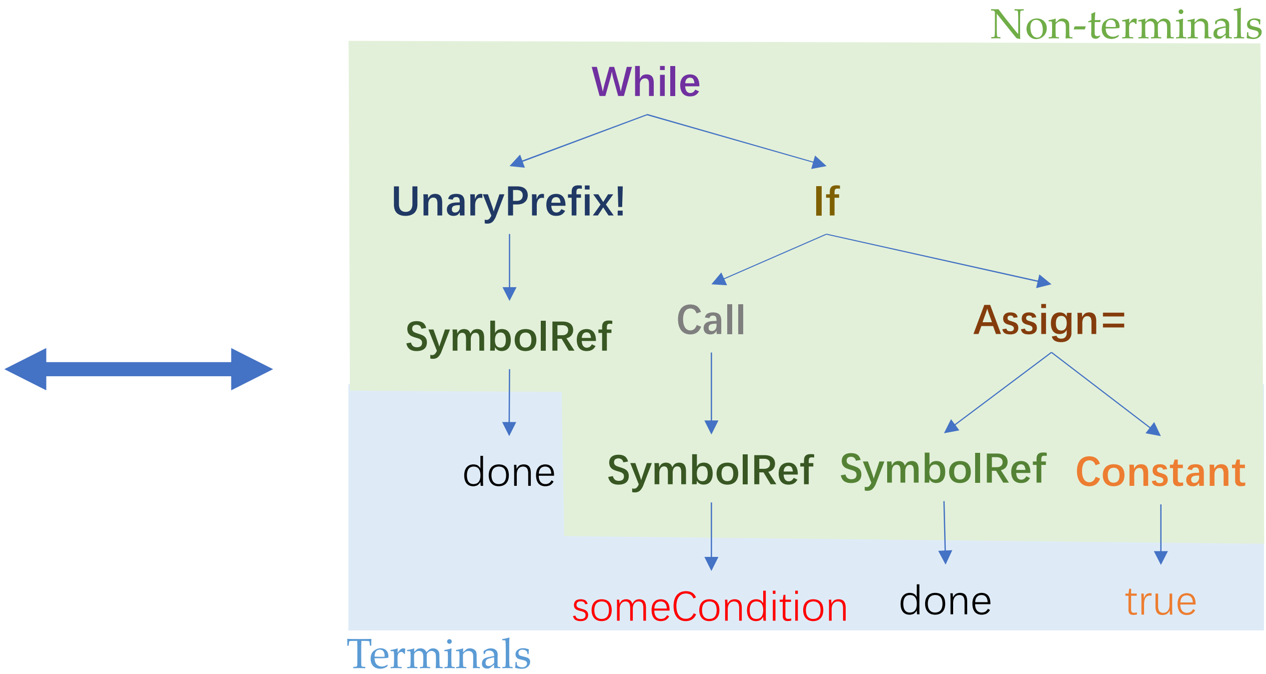
Option 3: extract features from parse tree

Code Representation AST

Abstract Syntax Tree (AST): parse tree for program codes

Example:

```
while (!done) {
    if (someCondition()) {
        done = true;
    }
}
```

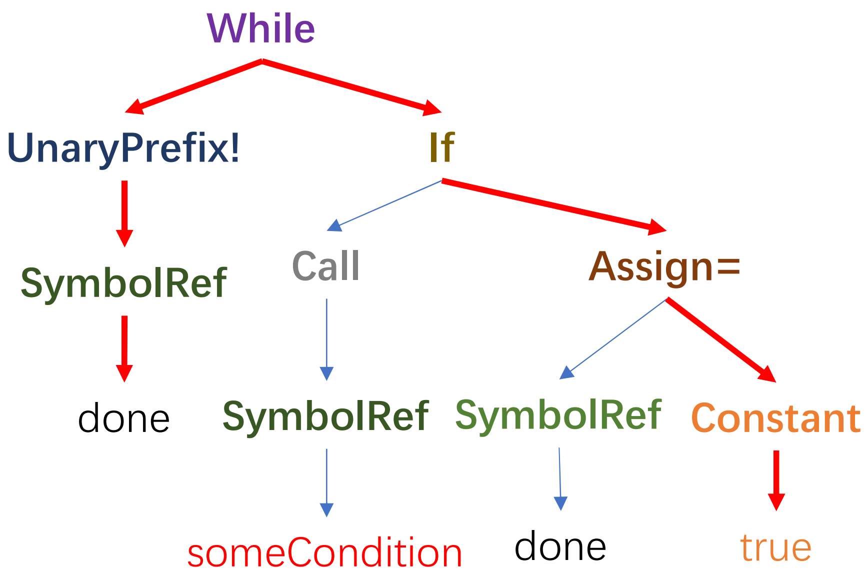


Code Representation

Bag of AST path contexts

- **How to feed parse tree to neural network?**

AST path example:



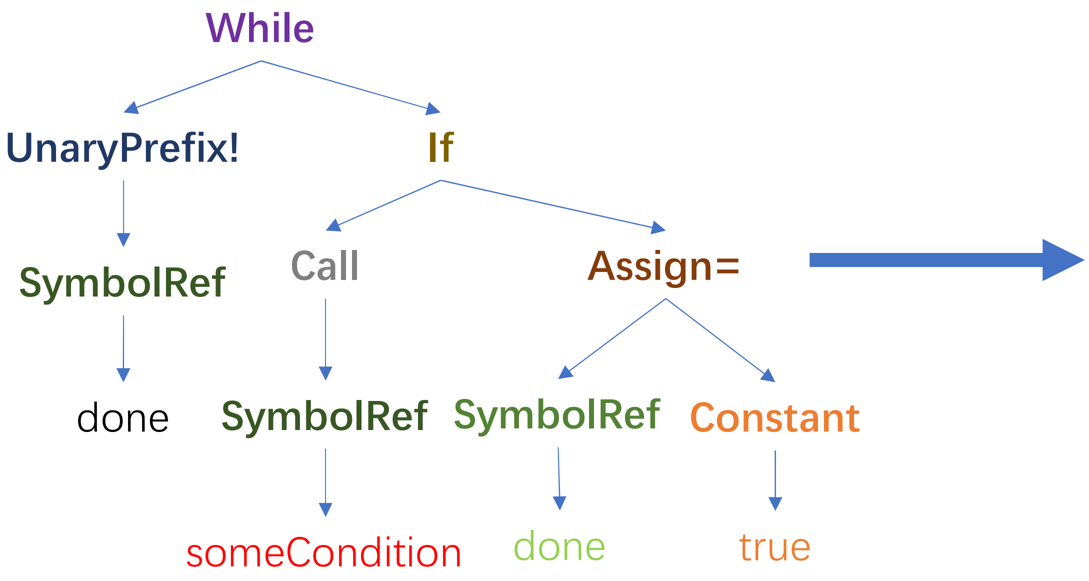
The red-marked path

$p = \langle$done, (**SymbolRef** $\uparrow$ **UnaryPrefix!** $\uparrow$ **While** $\downarrow$ **If** $\downarrow$ **Assign=** $\downarrow$ **Constant**), true$\rangle$

Code Representation

Bag of AST path contexts

- **How to feed parse tree to neural network?**



$$C = \left\{ \begin{array}{c} \langle \text{done}, p_1, \text{someCondition} \rangle, \\ \langle \text{done}, p_2, \text{done} \rangle, \\ \langle \text{done}, p_3, \text{true} \rangle, \\ \langle \text{someCondition}, p_4, \text{done} \rangle, \\ \langle \text{someCondition}, p_5, \text{true} \rangle, \\ \langle \text{done}, p_6, \text{true} \rangle \end{array} \right\}$$

Proposed in code2vec [Alon et al. 2018].

Code Representation

Bag of AST path contexts

Embedding for Bag of AST path contexts:

Basic idea: maintain 2 embedding vocabularies: $V_{\text{value}}, V_{\text{path}}$

$$C = \left\{ \begin{array}{c} \langle \text{done}, p_1, \text{someCondition} \rangle, \\ \langle \text{done}, p_2, \text{done} \rangle, \\ \langle \text{done}, p_3, \text{true} \rangle, \\ \langle \text{someCondition}, p_4, \text{done} \rangle, \\ \langle \text{someCondition}, p_5, \text{true} \rangle, \\ \langle \text{done}, p_6, \text{true} \rangle \end{array} \right\}$$

$$\text{Emb}(\langle x_s, p, x_t \rangle) = [V_{\text{value}}(x_s), V_{\text{path}}(p), V_{\text{value}}(x_t)]$$

Proposed as code2vec [Alon et al. 2018], further used in code2seq [Alon et al. 2019].

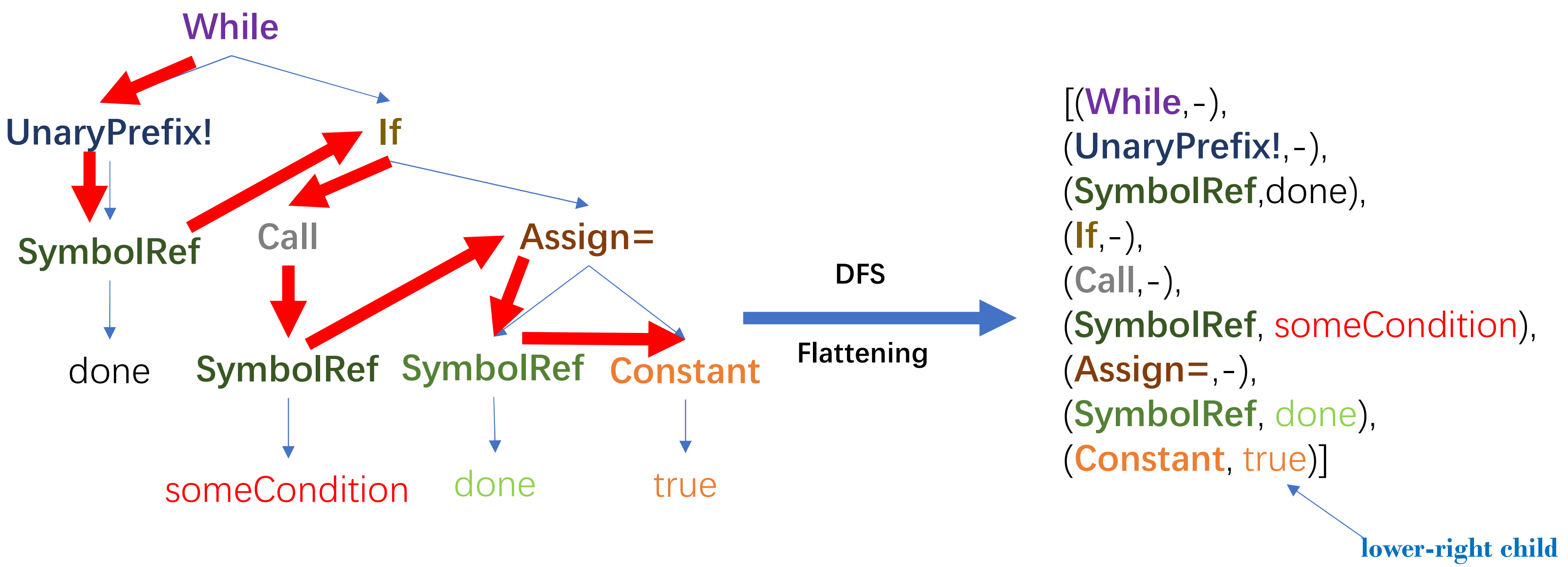
Afraid of large vocabulary size?

- Tokenize (e.g., `list_of_hash` = [`list, of, hash`])
- Use RNN encoder for paths [Alon et al. 2018]

Code Representation

AST as sequence of (non-terminal, terminal) pairs

- Another idea to feed parse tree to neural network



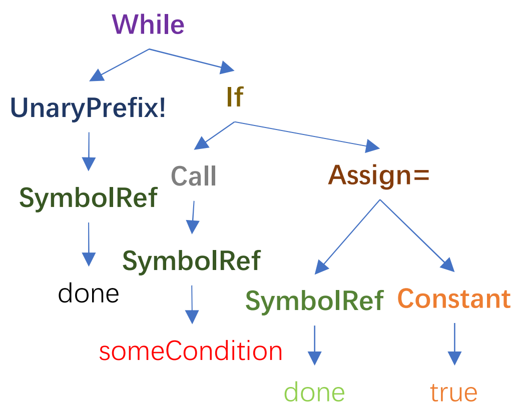
What is the benefit?

Tokenized AST is a suitable representation for code completion [Li et al. 2017]

Code Representation

AST graph

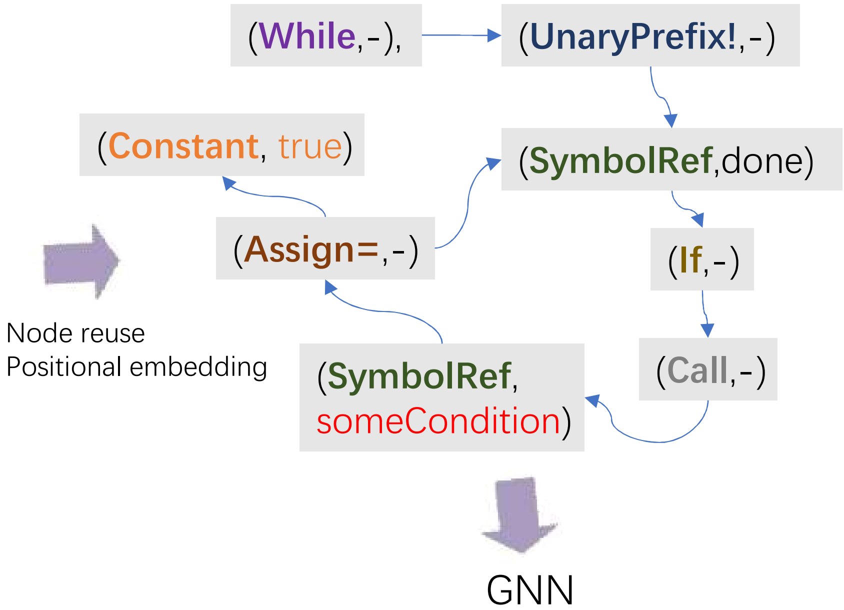
- **AST is a graph!**



[(**While**,-),
(**UnaryPrefix!**,-),
(**SymbolRef**,done),
(**If**,-),
(Call,-),
(**SymbolRef**, someCondition),
(**Assign=**,-),
(**SymbolRef**, done),
(**Constant**, true)]

Code Completion method by modeling flattened ASTs as Graphs

CCAG, [Wang et al. 2021]



Question: why not feed AST directly to GNN?

Reasoning of [Wang et al. 2021]: "in original AST, sequential information is missing"

Really?

Code Representation

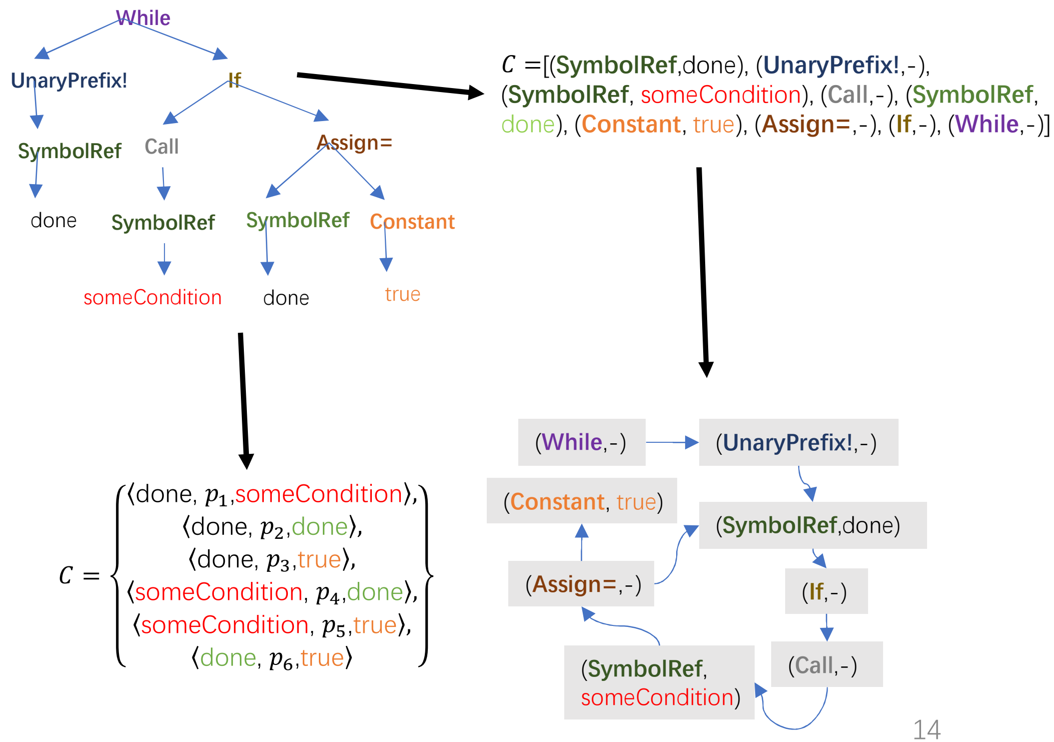
Summary

Level 1: natural-language-like representations

Level 2: AST (syntax-level representation)

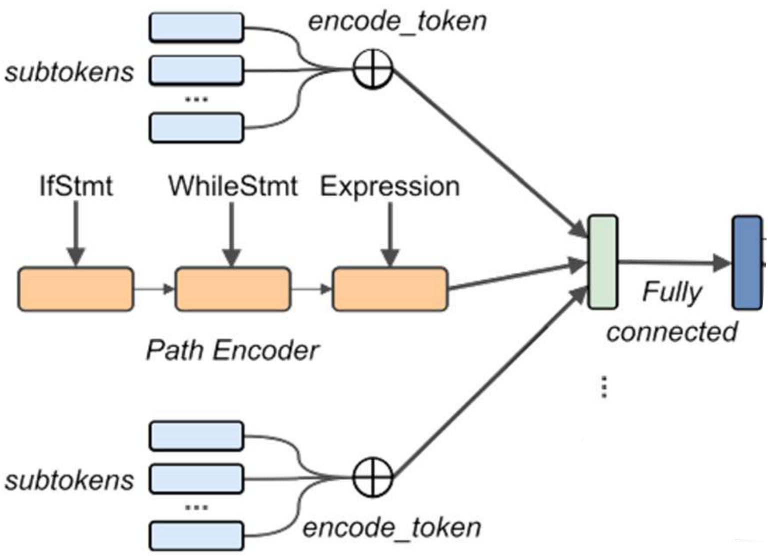
Level 3: extracted features (from AST)

Bag of AST paths,
Sequence of AST nodes (flattened AST),
AST graph



Models

Utilizing AST Paths



Embedding
z_i

Figure adapted from
code2seq [Alon et al. 2019]

Task Category	Input	Output
Explanation Tasks	Code snippet	Natural language sequence
Information Retrieval Tasks	Query String	Relevant code
Code completion	Code snippet	Code snippet

Models

Utilizing AST Paths – researches by Alon et al.

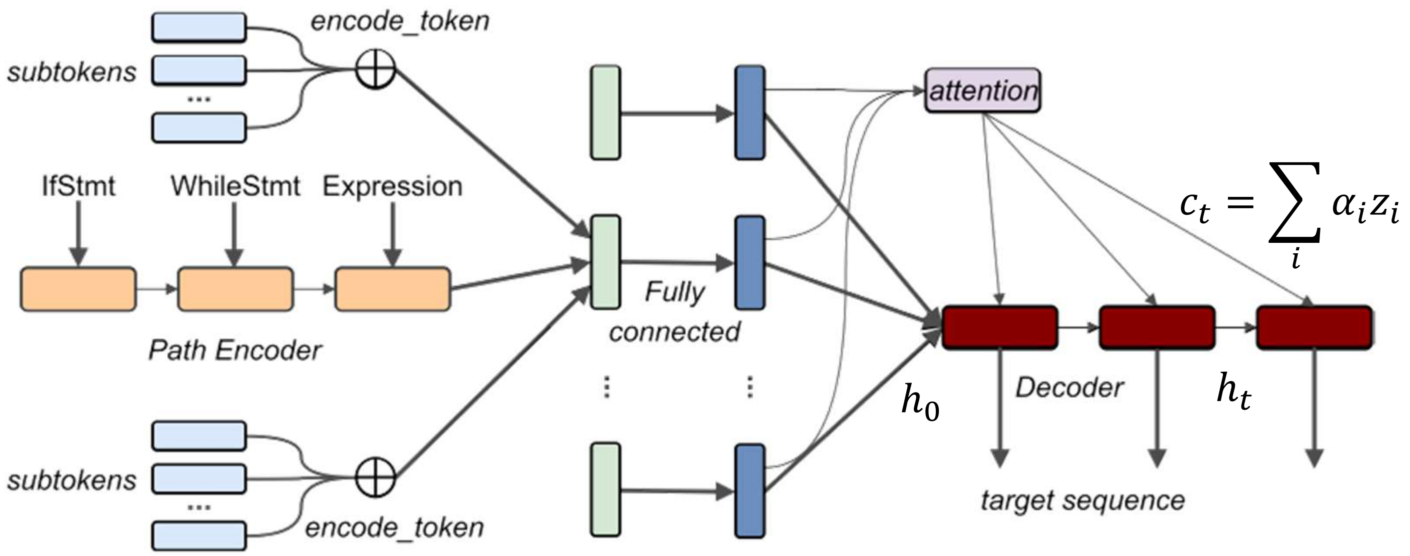


Figure adapted from code2seq [Alon et al. 2019]

Task Category	Input	Output
Explanation Tasks	Code snippet	Natural language sequence
Information Retrieval Tasks	Query String	Relevant code
Code completion	Code snippet	Code snippet

Models

Utilizing AST Paths – researches by Alon et al.

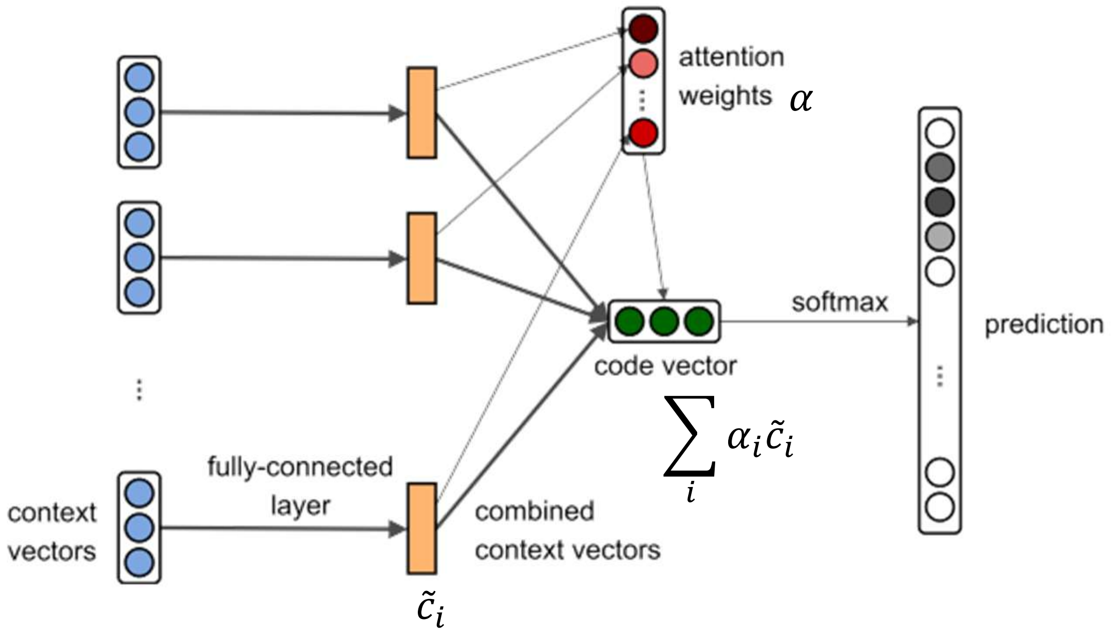


Figure adapted from
code2vec [Alon et al. 2018]

$\tilde{c}_i$ attention with what?

A global vector a maintained as a parameter:

$\alpha_i = \text{softmax}(\tilde{c}_i^T a)$

Task Category	Input	Output
Explanation Tasks	Code snippet	~~Natural language sequence~~ A single word
Information Retrieval Tasks	Query String	Relevant code
Code completion	Code snippet	Code snippet

Models

Utilizing AST Paths – researches by Alon et al.

DEMO: https://code2seq.org/

DEMO
https://code2vec.org/ -> “most similar”, “analogy”

Task Category	Input	Output
Explanation Tasks	Code snippet	Natural language sequence
Information Retrieval Tasks	Query String	Relevant code
Code completion	Code snippet	Code snippet

Models

Utilizing AST Paths – researches by Alon et al.

How to?

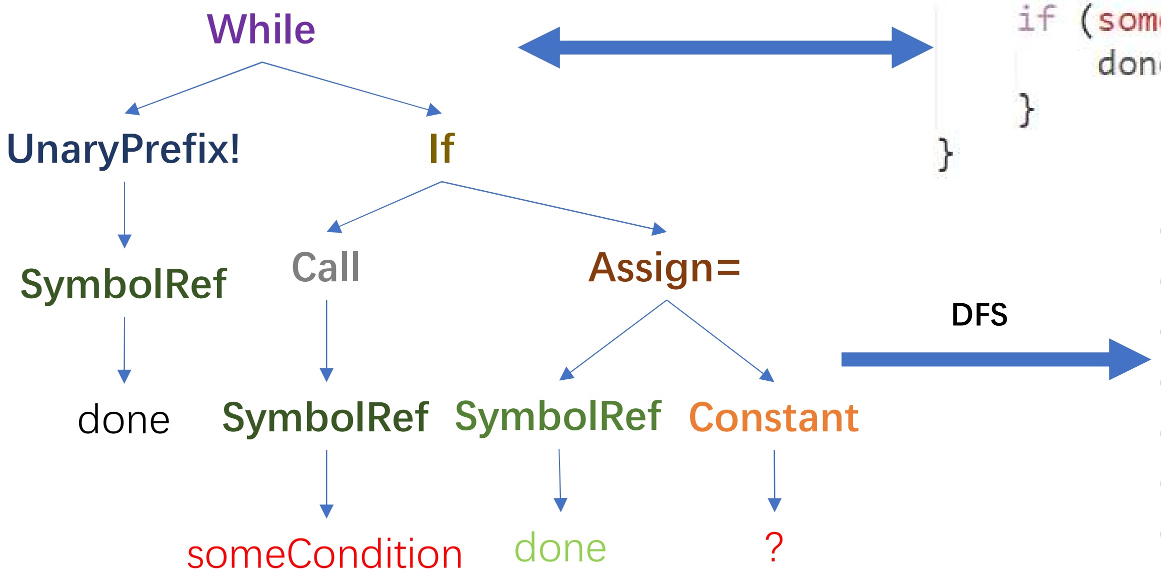
Task Category	**Input**	**Output**
Explanation Tasks	Code snippet	Natural language sequence
Information Retrieval Tasks	Query String	Relevant code
Code completion	Code snippet	Code snippet

Models

Single-token Code Completion Utilizing AST Token Sequences

Recall: converting code into AST token sequence

```
while (!done) {
    if (someCondition()) {
        done = _____;
    }
}
```



[(While,-),
(UnaryPrefix!,-),
(SymbolRef,done),
(If,-),
(Call,-),
(SymbolRef, someCondition),
(Assign=,-),
(SymbolRef, done),
(Constant, ?)]

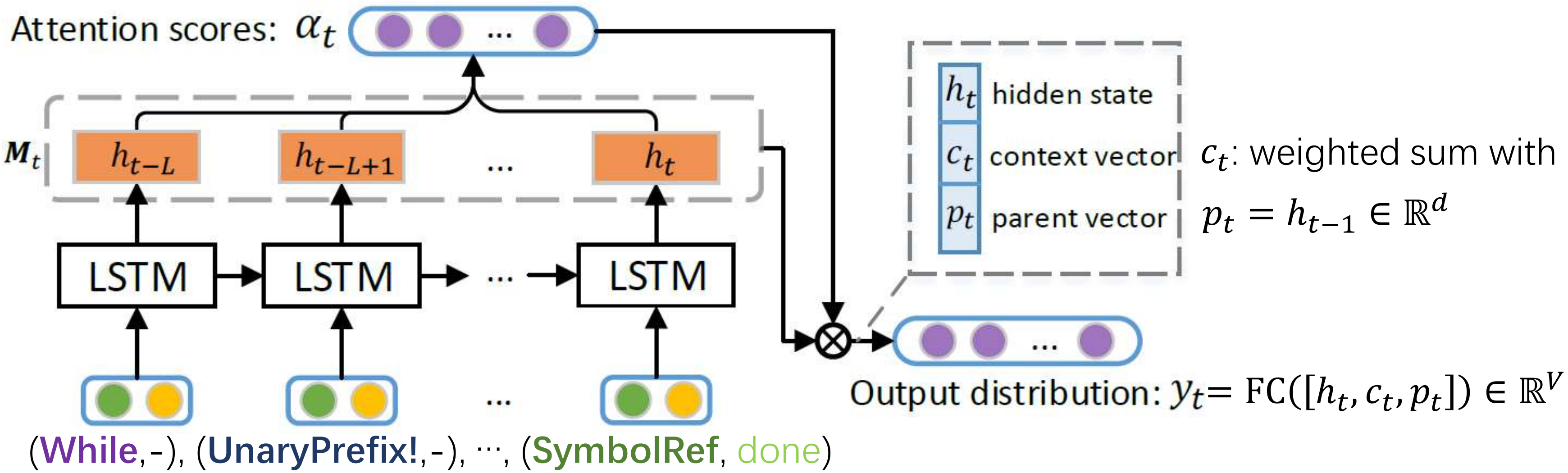
Single-token code completion:
Predict the last token (which is exactly the end of DFS).

Models

Single-token Code Completion Utilizing AST Token Sequences

Basic model: LSTM (with attention)

Figure from [Li et al. 2017]



c_t: weighted sum with attention scores

$p_t = h_{t-1} \in \mathbb{R}^d$

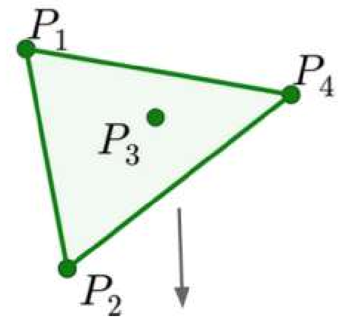
Question: what if the desired prediction is not in the vocabulary?

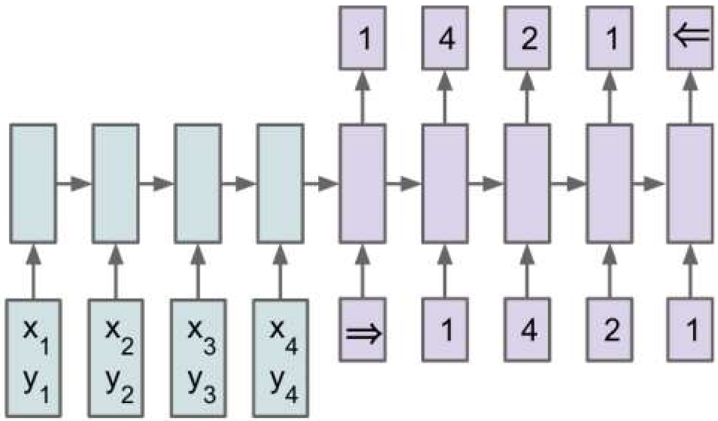
Pointer Network

Models

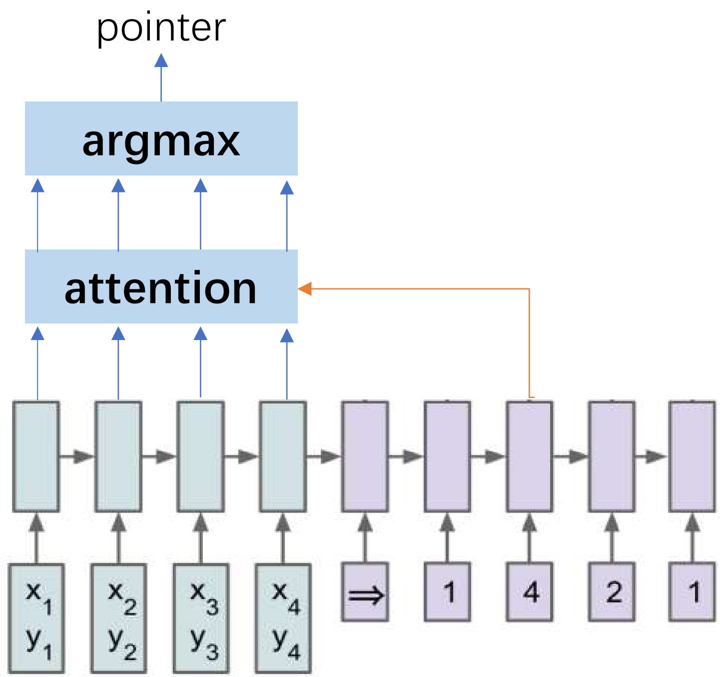
Pointer Network

2015, Vinyals et al., Pointer Network, https://arxiv.org/abs/1506.03134v2





Vanilla Seq2seq



Pointer network

Models

Single-token code completion utilizing AST Token Sequences

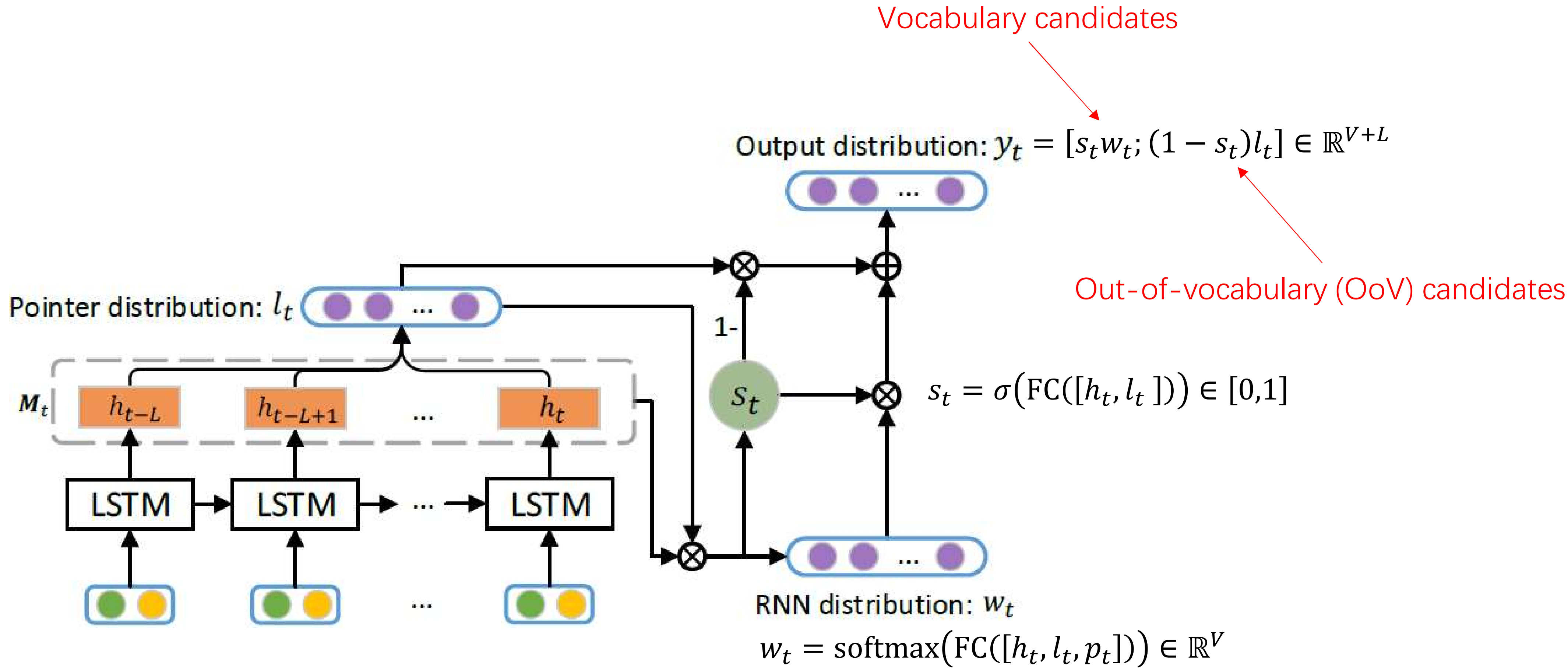


Figure from [Li et al. 2017]

Models

Single-token code completion utilizing AST Token Sequences

Example of OoV

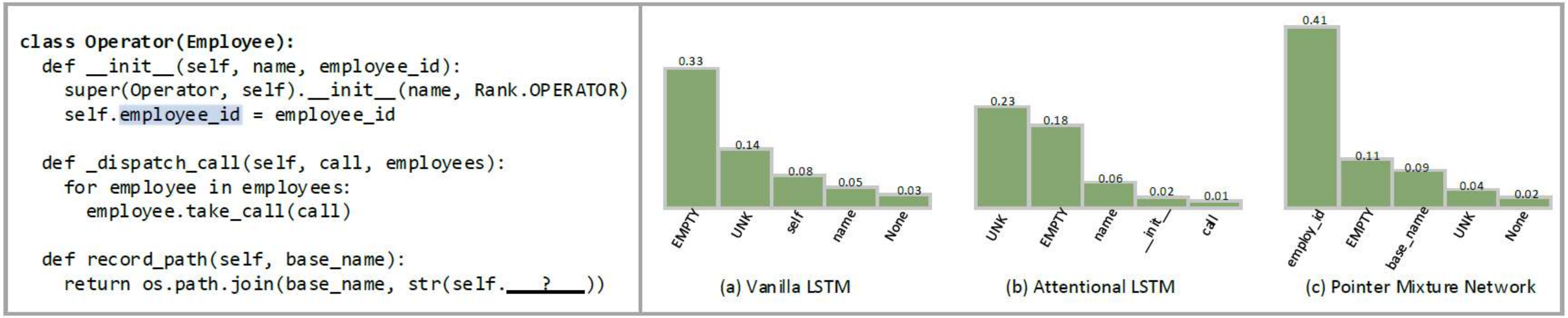


Figure from [Li et al. 2017]

Models

Single-token code completion using GNN

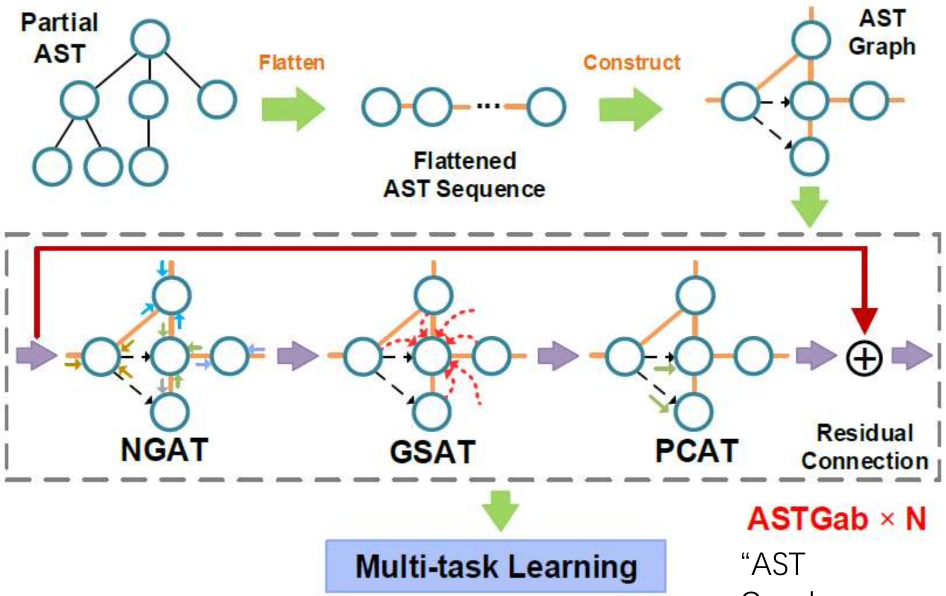


Figure from [Wang et al. 2021]

“AST Graph Attention Block”

Aggregation 1. Neighbor Graph Attention (NGAT)
Aggregation 2. Global Self-Attention (GSAT)
Aggregation 3. Parent-Child Attention (PCAT)
Aggregation 4. Residual connection

-> Get the final hidden state $\mathbf{H}$

Summarization:

$$s = \text{weighted\_pool}(\mathbf{H})$$
$$y^{(\text{nt})} = \text{FC}(s),$$
$$y^{(\text{t})} = \text{FC}(s)$$

Models

Code-block completion

Codex

Evaluating Large Language Models Trained on Code

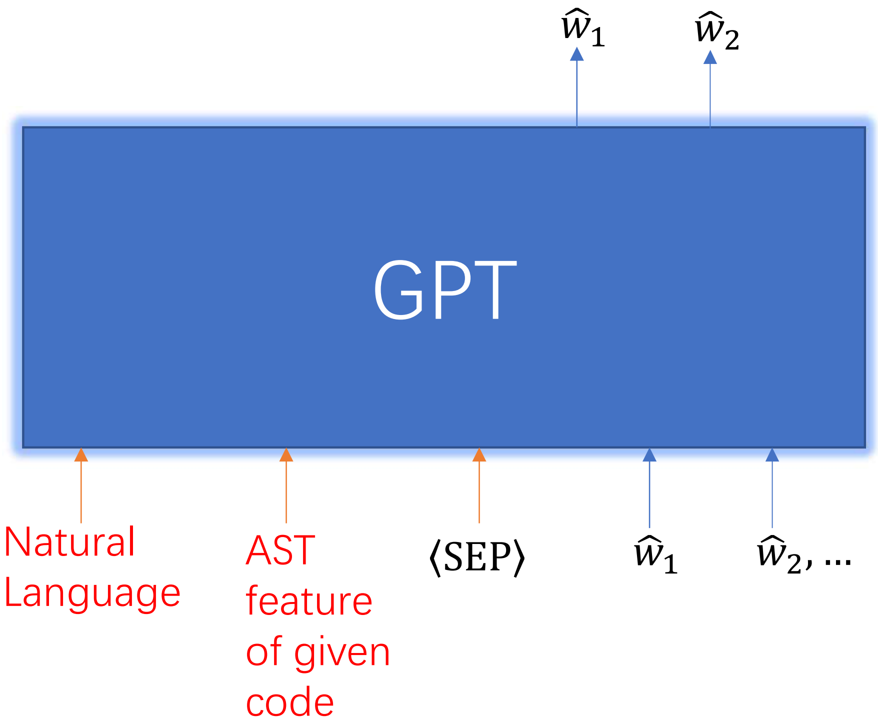
Mark Chen [*1] Jerry Tworek [*1] Heewoo Jun [*1] Qiming Yuan [*1] Henrique Ponde de Oliveira Pinto [*1]
Jared Kaplan [*2] Harri Edwards [1] Yuri Burda [1] Nicholas Joseph [2] Greg Brockman [1] Alex Ray [1] Raul Puri [1]
Gretchen Krueger [1] Michael Petrov [1] Heidy Khlaaf [3] Girish Sastry [1] Pamela Mishkin [1] Brooke Chan [1]
Scott Gray [1] Nick Ryder [1] Mikhail Pavlov [1] Alethea Power [1] Lukasz Kaiser [1] Mohammad Bavarian [1]
Clemens Winter [1] Philippe Tillet [1] Felipe Petroski Such [1] Dave Cummings [1] Matthias Plappert [1]
Fotios Chantzis [1] Elizabeth Barnes [1] Ariel Herbert-Voss [1] William Hebgen Guss [1] Alex Nichol [1] Alex Paino [1]
Nikolas Tezak [1] Jie Tang [1] Igor Babuschkin [1] Suchir Balaji [1] Shantanu Jain [1] William Saunders [1]
Christopher Hesse [1] Andrew N. Carr [1] Jan Leike [1] Josh Achiam [1] Vedant Misra [1] Evan Morikawa [1]
Alec Radford [1] Matthew Knight [1] Miles Brundage [1] Mira Murati [1] Katie Mayer [1] Peter Welinder [1]
Bob McGrew [1] Dario Amodei [2] Sam McCandlish [2] Ilya Sutskever [1] Wojciech Zaremba [1]

- Details?

https://copilot.github.com/

Models

Codex **Guesses on its inferencing**



Want-to-knows:

- **Representation of $\hat{w}_t$**
  - Problems of predicting natural-language-level tokens?
  - Problems of predicting AST token pairs?
- **OoV?**

Summary

Contents covered

- AST-based representations
- Code2Seq, Single-token code completion

Work	Code Representation	Task	Model
Code2vec [Alon et al. 2018]	AST Path Embedding	Code summary	Embedding + Attention + FC
Code2seq [Alon et al. 2019]	AST Path Embedding	Code captioning	Embedding + Attention + RNN decoder
[Li et al. 2017]	AST Token Sequence	Code completion	Pointer Mixture Network
CCAG [Wang et al. 2021]	AST Graph	Code completion	GNN
GraphCodeBERT [Guo et al. 2021]	Text + Code Text + Variable Flow	Universal	BERT + Downstream-specific Models
SynCoBERT [Wang et al. 2021]	Text + Code Text + AST Token Sequence	Universal	BERT + Downstream-specific Models
CodeBERT [Feng et al. 2020]	Text + Code Text	Universal	BERT + Downstream-specific Models

Tensors are universal

Possible Research Directions

A tentative list of relevant topics for research

GNN-related open questions and "combination of techniques" (which is not done yet)

- AST vs. flattened AST graph: does "sequential information" really matter?
- OoV (graph pointer neural network)
- More GNN architectures

Code-block completion

- How did Codex achieve this?
- Is it possible to generate code in a natural-language-like manner?
- How to generate AST using neural network?

More application scenarios

- e.g., code maintenance: given description (e.g., "plot the output") and modify the original code
- …

References

Uri Alon, Meital Zilberstein, Omer Levy, and Eran Yahav. 2018. A General Path-based Representation for Predicting Program Properties. 39th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 2018). ACM, New York, 404–419. https://doi.org/10.1145/3192366.3192412

Uri Alon, Meital Zilberstein, Omer Levy, and Eran Yahav. 2018. Code2Vec: Learning Distributed Representations of Code. Proc. ACM Program. Lang. http://doi.acm.org/10.1145/3290353

Uri Alon, Meital Zilberstein, Omer Levy, and Eran Yahav. 2019. Code2Seq: Generating Sequences from Structured Representations of Code. https://arxiv.org/abs/1808.01400

Jian Li, Yue Wang, Michael R Lyu, and Irwin King. 2017. Code completion with neural attention and pointer networks. https://arxiv.org/abs/1711.09573

Yanlin Wang and Hui Li. 2021. Code Completion by Modeling Flattened Abstract Syntax Trees as Graphs. https://arxiv.org/abs/2103.09499

Zhangyin Feng et al. 2020. CodeBERT: A Pre-Trained Model for Programming and Natural Languages. https://arxiv.org/abs/2002.08155

Daya Guo et al. 2021. GraphCodeBERT: Pre-training Code Representations with Data Flow. https://arxiv.org/abs/2009.08366

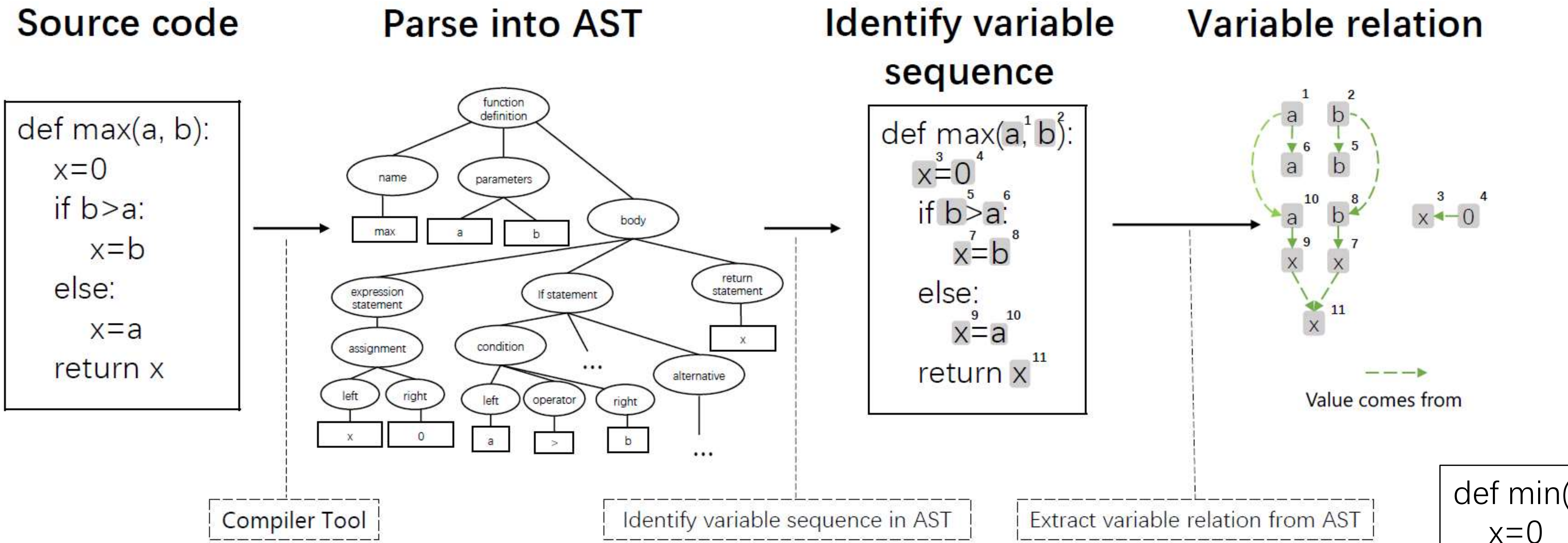
Xin Wang et al. 2021. SynCoBERT: Syntax-Guided Multi-Modal Contrastive Pre-Training for Code Representation. https://arxiv.org/abs/2108.04556

Back-up Contents

Code Representation

Variable flow

Proposed in GraphCodeBERT [Guo et al. 2021]



Figures adapted from [Guo et al. 2021], GraphCodeBERT

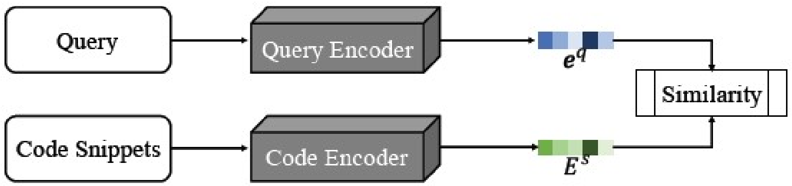
same variable flow

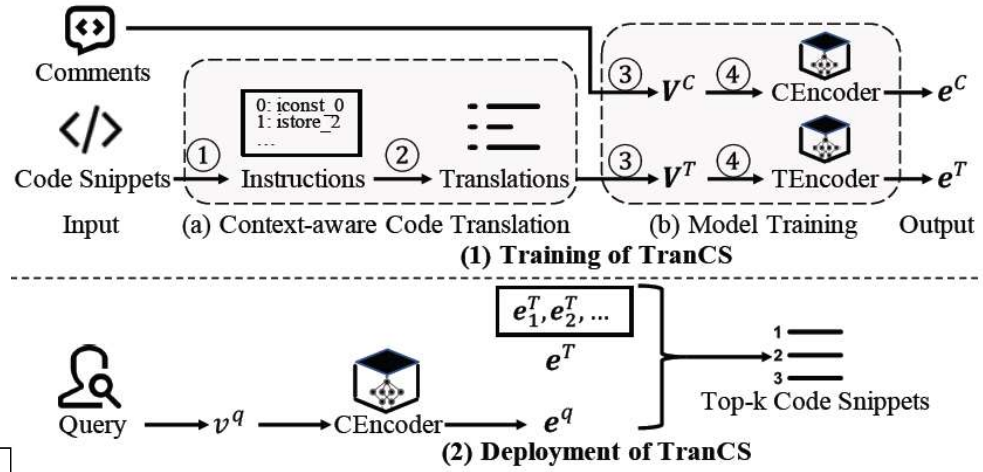
```
def min(a, b):
    x=0
    if b<a:
        x=b
    else:
        x=a
    return x
```

How does low-resolution feature help?

Code Search

Sun et al. 2022. Code Search based on Context-aware Code Translation. https://arxiv.org/abs/2202.08029





②

```java
// calculate the sum of an int array
public int calArraySum(int[] array) {
    int sum = 0;
    int i = 0;
    for (; i < array.length; i++) {
        sum = sum + array[i];
    }
    return sum;
}
```

(a) Code Snippet s_a

0: push int constant **0**.
1: store int **0** into local variable **sum**/**result**.
2: push int constant **0**.
3: store int **0** into local variable **i**/**index**.
4: load int **value** from local variable **i**/**index**.
5: load reference **array**/**array** from local variable **array**/**array**.
6: get length of array **array**/**array**.
7: if and only if int **value** is greater or equal to int **length** then go to **22**.
10: load int **value_1** from local variable **sum**/**result**.
11: load reference **array**/**array** from local variable **array**/**array**.
12: load int **value_2** from local variable **i**/**index**.
13: load int **value_3** from **array**/**array**[**value_2**].
14: int result is int **value_1** add int **value_3**; push result into **value_4**.
15: store int **value_4** into local variable **sum**/**result**.
16: increment local variable **i**/**index** by constant **1**.
19: goto **4**.
22: load int **value_5** from local variable **sum**/**result**.
23: return int **value_5** from method.

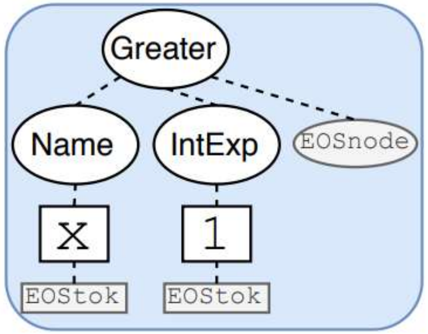
Code-block Completion

Anycode

2020, Alon et al., Structural Language Models of Code, https://arxiv.org/pdf/1910.00577.pdf

```
public static Path[] stat2Paths(
    FileStatus[] stats) {
  if (stats == null) return null;
  Path[] ret = new Path[stats.length];
  for (int i = 0; i < stats.length; ++i){
    ret[i] = [                    ];
  }
  return ret;
}
```

```
public static string Camelize(
    this string input)
{
    var word = input.Pascalize();
    return word.Length > 0 ?
        [                    ].ToLower()
            + word.Substring(1)
        : word;
}
```



Filling in the blank given a partial AST.

The output space in each generation step is determined by the previous token.

Generation ends when sampling EOS_token or EOS_node.

Datasets

For code completion:
JS (JS50K etc.),PY (PY50K etc.) Datasets: https://www.sri.inf.ethz.ch/research/plml

For code summary:
Java (Java Large etc.) Datasets: https://groups.inf.ed.ac.uk/cup/codeattention/
CodeNN C# dataset: https://github.com/sriniiyer/codenn/

For code search:
CodeSearchNet Data Corpus: https://github.com/github/CodeSearchNet#data-details

Model Performances

Single-token code completion
Metric: accuracy

	JS1k		JS10k		JS50k		PY1k		PY10k		PY50k	
	value	type	value	type	value	type	value	type	value	type	value	type
VanillaLSTM	53.19%	69.52%	58.04%	71.16%	59.70%	72.08%	49.99%	68.08%	52.67%	68.86%	53.66%	69.09%
ParentLSTM	56.45%	71.99%	61.54%	73.46%	63.39%	74.24%	52.57%	70.10%	55.87%	76.25%	56.93%	71.00%
PointerMixtureNet	56.49%	71.95%	62.33%	74.28%	64.14%	76.01%	52.98%	69.98%	56.91%	**76.94%**	57.22%	70.91%
Transformer	58.40%	**73.29%**	**63.93%**	**74.78%**	65.31%	75.89%	53.49%	70.63%	57.52%	71.45%	59.05%	71.91%
Transformer-XL	**59.23%**	72.11%	62.82%	74.09%	**66.41%**	**76.23%**	**55.13%**	**72.45%**	**58.21%**	73.19%	**60.00%**	**72.42%**
CCAG	**62.79% (6.01%)**	**75.72% (3.32%)**	**66.69% (4.32%)**	**78.55% (5.04%)**	**68.19% (2.68%)**	**80.14% (5.13%)**	**61.92% (12.32%)**	**76.71% (5.88%)**	**63.24% (8.64%)**	**80.90% (5.15%)**	**64.22% (7.03%)**	**75.31% (3.99%)**

Code Summary

Model	Full Test Set (413915 methods) Precision	Recall	F1
CNN+Attention [Allamanis et al. 2016]	-	-	-
LSTM+Attention [Iyer et al. 2016]	33.7	22.0	26.6
Paths+CRFs [Alon et al. 2018]	53.6	46.6	49.9
PathAttention (this work)	**63.1**	**54.4**	**58.4**

On Java dataset