Contextual Cues for Deep Learning Models of Code

Disha Shrivastava

Université de Montréal

Mila

Outline

On-the-Fly Adaptation of Source Code Models
(CAP Workshop, NeurIPS 2020)


On-the-Fly Adaptation of Source Code Models

Disha Shrivastava [1,2] Hugo Larochelle [2,1,3] Daniel Tarlow [2,4]

RepoFusion: Training Code Models to Understand Your Repository
(under review at NeurIPS 2023)


RepoFusion: Training Code Models to Understand Your Repository

Disha Shrivastava, Mila, Université de Montréal; Denis Kocetkov, ServiceNow Research; Harm de Vries, ServiceNow Research; Dzmitry Bahdanau, ServiceNow Research, Mila, McGill University, Canada CIFAR AI Chair; Torsten Scholak, ServiceNow Research

Introduction → Article 1 → Article 2 → Article 3 → Article 4 → Final Thoughts


Learning to Combine Per-Example Solutions for Neural Program Synthesis

Disha Shrivastava, Mila, Université de Montréal, Google Research; Hugo Larochelle, Mila, Université de Montréal, Google Research, CIFAR Fellow; Daniel Tarlow, Mila, McGill University, Google Research


Repository-Level Prompt Generation for Large Language Models of Code

Disha Shrivastava [1,2] Hugo Larochelle [1,2,3,4] Daniel Tarlow [1,5,3]

Learning to Combine Per-Example Solutions for Neural Program Synthesis
(NeurIPS 2021)

Repository Level Prompt Generation for Large Language Models of Code
(ICML 2023)

Outline

On-the-Fly Adaptation of Source Code Models
(CAP Workshop, NeurIPS 2020)

RepoFusion: Training Code Models to Understand Your Repository
(under review at NeurIPS 2023)

Introduction → **Article 1** → **Article 2** → **Article 3** → **Article 4** → **Final Thoughts**

Learning to Combine Per-Example Solutions for Neural Program Synthesis
(NeurIPS 2021)

Repository Level Prompt Generation for Large Language Models of Code
(ICML 2023)

Deep Learning for Code

- Source Code: Exciting application domain for deep learning methods.

```
# Given a string representing musical
notes in a special ASCII format,
# parse this string and compute a total
length of the music in beats.
# Legend:
#   'o' - whole note, last four beats
#   'o|' - half note, last two beats
#   '.|' - quarter note, last one beat

notes = 'o o| .| o| o| .| .| .| .| o o'
```

Code Generation

```
# 1. Split the string into a list of
notes.
# 2. Iterate over the list and compute the
total length of the music.
# 3. Print the result.

# 1.
notes = notes.split()

# 2.
total_length = 0
For note in notes:
    if note == 'o':
         total_length += 4
    if note == 'o|':
         total_length += 2
     if note == '.|':
         total_length += 1

# 3.
print(total_length)
```

https://ai.googleblog.com/2022/04/pathways-language-model-palm-scaling-to.html

Deep Learning for Code

- Source Code: Exciting application domain for deep learning methods.

```
# Given a string representing musical notes in a special ASCII format,
# parse this string and compute a total length of the music in beats.
# Legend:
#   'o' - whole note, last four beats
#   'o|' - half note, last two beats
#   '.|' - quarter note, last one beat

notes = 'o o| .| o| o| .| .| .| .| o o'
```

```
# 1. Split the string into a list of notes.
# 2. Iterate over the list and compute the total length of the music.
# 3. Print the result.

# 1.
notes = notes.split()

# 2.
total_length = 0
For note in notes:
    if note == 'o':
        total_length += 4
    if note == 'o|':
        total_length += 2
    if note == '.|':
        total_length += 1

# 3.
print(total_length)
```

Code Generation

```
#include <stdio.h>
int main() {
    int a[i];
    int i;
    scanf("%d", &n);
    scanf("%d", &a[i]);
    int count = 0;
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            if (a[i] == a[j])
                count = count + 1;
        }
    }
    if (count == a[i])
        printf("Yes");
    else
        printf("No");
    return 0;
}
```

```
#include <stdio.h>
int main() {
    int a[100], i, n, count = 0;

    scanf("%d", &n);
    for (i = 0; i < n; i++)
        scanf("%d", &a[i]);
    for (i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            if (a[i] == a[j])
                count = count + 1;
        }
    }
    if (count == n)
        printf("Yes");
    else
        printf("No");
    return 0;
}
```

Code Repair

https://ai.googleblog.com/2022/04/pathways-language-model-palm-scaling-to.html

Deep Learning for Code

- Source Code: Exciting application domain for deep learning methods.

```
# Given a string representing musical
notes in a special ASCII format,
# parse this string and compute a total
length of the music in beats.
# Legend:
#   'o' - whole note, last four beats
#   'o|' - half note, last two beats
#   '.|' - quarter note, last one beat

notes = 'o o| .| o| o| .| .| .| .| o o'
```

```
# 1. Split the string into a list of
notes.
# 2. Iterate over the list and compute the
total length of the music.
# 3. Print the result.

# 1.
notes = notes.split()

# 2.
total_length = 0
For note in notes:
    if note == 'o':
        total_length += 4
    if note == 'o|':
        total_length += 2
     if note == '.|':
        total_length += 1

# 3.
print(total_length)
```

Code Generation

```
#include <stdio.h>
int main() {
    int a[i];
    int i;
    scanf("%d", &n);
    scanf("%d", &a[i]);
    int count = 0;
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            if (a[i] == a[j])
                count = count + 1;
        }
    }
    if (count == a[i])
        printf("Yes");
    else
        printf("No");
    return 0;
}
```

```
#include <stdio.h>
int main() {
    int a[100], i, n, count = 0;

    scanf("%d", &n);
    for (i = 0; i < n; i++)
        scanf("%d", &a[i]);
    for (i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            if (a[i] == a[j])
                count = count + 1;
        }
    }
    if (count == n)
        printf("Yes");
    else
        printf("No");
    return 0;
}
```

Code Repair

```
// Translate from C to Python
int add_one ( int x ){
 int m = 1;
 while ( x & m ) {
   x = x ^ m;
   m <<= 1;
 }
 x = x ^ m;
 return x; }
```

```
def add_one(x: int):
  m = 1
  while (x & m):
    x = (x ^ m)
    m <<= 1
  x = (x ^ m)
  return x
```

Code Translation

https://ai.googleblog.com/2022/04/pathways-language-model-palm-scaling-to.html

Deep Learning for Code

- Source Code: Exciting application domain for deep learning methods.

```
# Given a string representing musical
notes in a special ASCII format,
# parse this string and compute a total
length of the music in beats.
# Legend:
#    'o' - whole note, last four beats
#    'o|' - half note, last two beats
#    '.|' - quarter note, last one beat

notes = 'o o| .| o| o| .| .| .| .| o o'
```

```
# 1. Split the string into a list of
notes.
# 2. Iterate over the list and compute the
total length of the music.
# 3. Print the result.

# 1.
notes = notes.split()

# 2.
total_length = 0
For note in notes:
    if note == 'o':
        total_length += 4
    if note == 'o|':
        total_length += 2
    if note == '.|':
```

Code Generation

```
#include <stdio.h>
int main() {
    int a[i];
    int i;
    scanf("%d", &n);
    scanf("%d", &a[i]);
    int count = 0;
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            if (a[i] == a[j])
                count = count + 1;
        }
    }
    if (count == a[i])
        printf("Yes");
    else
        printf("No");
    return 0;
}
```

```
#include <stdio.h>
int main() {
    int a[100], i, n, count = 0;

    scanf("%d", &n);
    for (i = 0; i < n; i++)
        scanf("%d", &a[i]);
    for (i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            if (a[i] == a[j])
                count = count + 1;
        }
    }
    if (count == n)
        printf("Yes");
    else
        printf("No");
    return 0;
}
```

Code Repair

Slow Version(Runtime=4.18s)

```
t = int(input())
for i in range(1, t+1):
  n, b = [int(s) for s in input().split(" ")]
  houses = [int(s) for s in input().split(" ")]
  houses.sort()

  result = 0
  for h in houses:
      if b >= h:
          result += 1
          b -= h
  print("Case #{}: {}".format(i, result))
```

Algorithmic Difference: Using a heap instead of pre-sorting the list once allows early termination of the main loop.

Fast Version(Runtime=2.52s: ~1.66x Speedup)

```
import heapq

# input() reads a string with a line of input, stripping the '\n'(newline) at the end.
# This is all you need for most Kickstart problems.
#all_data = []
t = int(input())  # read a line with a single integer
for i in range(1, t + 1):
  n, b = [int(s) for s in input().split(" ")]  # read a list of integers, 2 in this case
  prices = [int(s) for s in input().split(" ")]
  #all_data.append([n,b,prices])
  heapq.heapify(prices)
  houses = 0
  while prices and b > 0:
      new_house = heapq.heappop(prices)
      b -= new_house
      if b >= 0:
          houses += 1
  print("Case #{}: {}".format(i, houses))
```

Code Analysis

```
// Translate from C to Python
int add_one ( int x ){
 int m = 1;
 while ( x & m ) {
   x = x ^ m;
   m <<= 1;
 }
 x = x ^ m;
 return x; }
```

```
def add_one(x: int):
  m = 1
  while (x & m):
    x = (x ^ m)
    m <<= 1
  x = (x ^ m)
  return x
```

Code Translation

“Learning to Improve Code Efficiency”. Chen, Binghong, et al. (2022)

https://ai.googleblog.com/2022/04/pathways-language-model-palm-scaling-to.html

Deep Learning for Code

- Source Code: Exciting application domain for deep learning methods.

“Learning to Improve Code Efficiency”. Chen, Binghong, et al. (2022)

https://denigma.app/

https://ai.googleblog.com/2022/04/pathways-language-model-palm-scaling-to.html

Deep Learning for Code

- Source Code: Exciting application domain for deep learning methods.
- Motivation

Helping Non-Programmers

Enable non-expert users to solve problems in an automated fashion.

- Programming requires technical skills.
- Generate programs from user's intent expressed in forms that are natural to them such as NL.

Deep Learning for Code

- Source Code: Exciting application domain for deep learning methods.
- Motivation

Helping Non-Programmers

Enable non-expert users to solve problems in an automated fashion.

- Programming requires technical skills.
- Generate programs from user's intent expressed in forms that are natural to them such as NL.

Helping Programmers

Boost productivity of software developers.

- Divert attention from mundane tasks.
- Focus on tasks that require creative thinking.
- Code completion to avoid typing boilerplate code.

Deep Learning for Code

- Source Code: Exciting application domain for deep learning methods.
- Motivation

Helping Non-Programmers

Enable non-expert users to solve problems in an automated fashion.

- Programming requires technical skills.
- Generate programs from user's intent expressed in forms that are natural to them such as NL.

Helping Programmers

Boost productivity of software developers.

- Divert attention from mundane tasks.
- Focus on tasks that require creative thinking.
- Code completion to avoid typing boilerplate code.

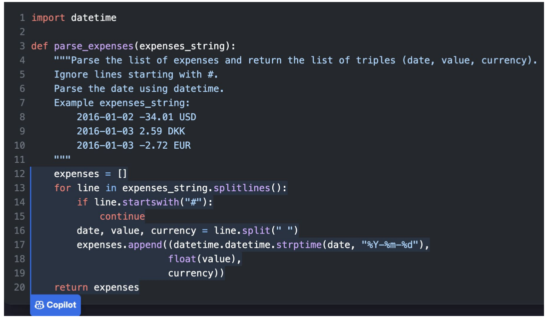
Advancing ML Research

Several challenges with modeling source code.

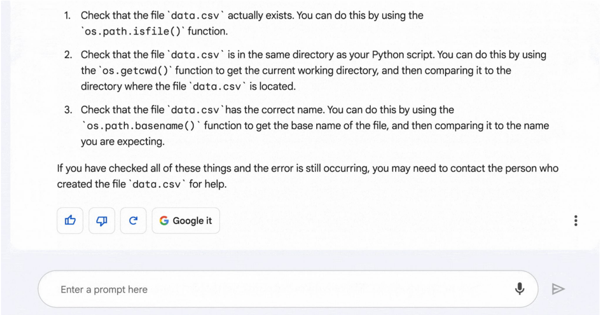
- Rigid Syntax
- Structure
- Discrete nature
- Multiple symbolic representation forms
- Continuously evolving

Deep Learning for Code

- Source Code: Exciting application domain for deep learning methods.
- Motivation
- LLMs of code have potential for huge impact.



GitHub Copilot



Bard

tabnine

Already part of consumer-facing products

Central Theme

Effectively Harness Contextual Cues

Identify and **select relevant** contextual cues from a given task.

Leverage these contextual cues **effectively** in deep learning models of code.

Central Theme

Effectively Harness Contextual Cues

Identify and **select** **relevant** contextual cues from a given task.

Leverage these contextual cues **effectively** in deep learning models of code.

Improves Generalization

- Adding information that the model wouldn't normally have access to.
- Directing model's attention to specific information.

More Context-Aware Predictions

- Adapt to unseen tasks
- Improve performance on existing tasks.

Our General Framework

Given

X = Input Context *(code in the current file before the cursor)*
Y = Actual Target *(tokens following the cursor till the end of the line)*
W = Context Meta-information *(content in other files in the repository)*

Goal: Effectively harness contextual cues based on **X** and **W** such that the predicted target $\hat{\mathbf{Y}}$ is close to the actual target **Y**.

Our General Framework

Given

X = Input Context *(code in the current file before the cursor)*
Y = Actual Target *(tokens following the cursor till the end of the line)*
W = Context Meta-information *(content in other files in the repository)*

Goal: Effectively harness contextual cues based on **X** and **W** such that the predicted target $\hat{\mathbf{Y}}$ is close to the actual target **Y**.

Context Enhancement

$$\mathbf{Z} = \mathit{Enhance}\,(\,\mathbf{X}\,,\ \mathbf{W}\,)$$

Support Context
(method names and bodies from the imported file)

Our General Framework

Given

X = Input Context *(code in the current file before the cursor)*
Y = Actual Target *(tokens following the cursor till the end of the line)*
W = Context Meta-information *(content in other files in the repository)*

Goal: Effectively harness contextual cues based on **X** and **W** such that the predicted target $\hat{\mathbf{Y}}$ is close to the actual target **Y**.

Context Enhancement

$$\mathbf{Z} = \textit{Enhance}\,(\,\mathbf{X}\,,\ \mathbf{W}\,)$$

Prediction using the Enhanced Context

$$\hat{\mathbf{Y}} = \textit{Predict}\,(\,\mathbf{X}\,,\mathbf{Z}\,)$$

Support Context
(method names and bodies from the imported file)

Predicted Target
(tokens generated by the model)

Our General Framework

Given

- **X** = Input Context *(code in the current file before the cursor)*
- **Y** = Actual Target *(tokens following the cursor till the end of the line)*
- **W** = Context Meta-information *(content in other files in the repository)*

Goal: Effectively harness contextual cues based on **X** and **W** such that the predicted target $\hat{\mathbf{Y}}$ is close to the actual target **Y**.

Context Enhancement

$\mathbf{Z} = \mathit{Enhance}(\mathbf{X}, \mathbf{W})$

Prediction using the Enhanced Context

$\hat{\mathbf{Y}} = \mathit{Predict}(\mathbf{X}, \mathbf{Z})$

Support Context
(method names and bodies from the imported file)

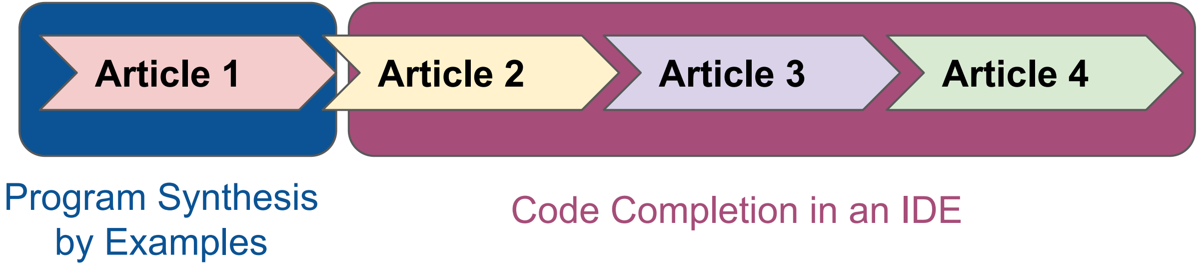
Without Context Enhancement $\hat{\mathbf{Y}} = Q(\mathbf{X})$

Predicted Target
(tokens generated by the model)

Thesis Overview

All articles in this thesis are based on our general **Enhance-Predict framework**.

- We propose novel approaches for Enhance and Predict stages.
- We focus on two main tasks.



Outline

On-the-Fly Adaptation of Source Code Models
(CAP Workshop, NeurIPS 2020)

RepoFusion: Training Code Models to Understand Your Repository
(under review at NeurIPS 2023)

Introduction → **Article 1** → **Article 2** → **Article 3** → **Article 4** → **Final Thoughts**

Learning to Combine Per-Example Solutions for Neural Program Synthesis
(NeurIPS 2021)

Repository Level Prompt Generation for Large Language Models of Code
(ICML 2023)

Learning to Combine Per-Example Solutions for Neural Program Synthesis

NeurIPS 2021

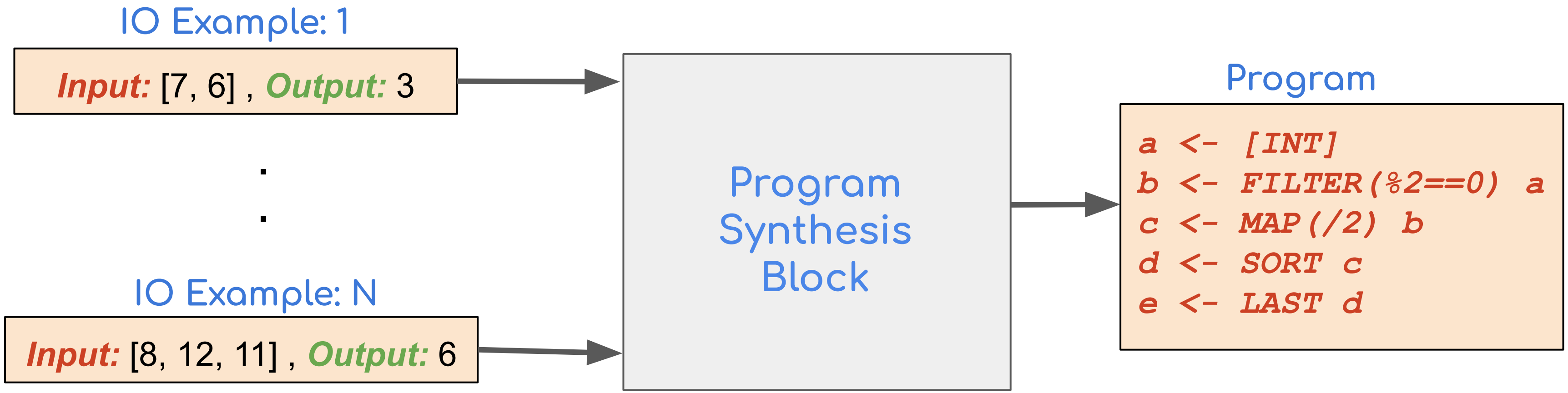
Université de Montréal

Mila

Google Research

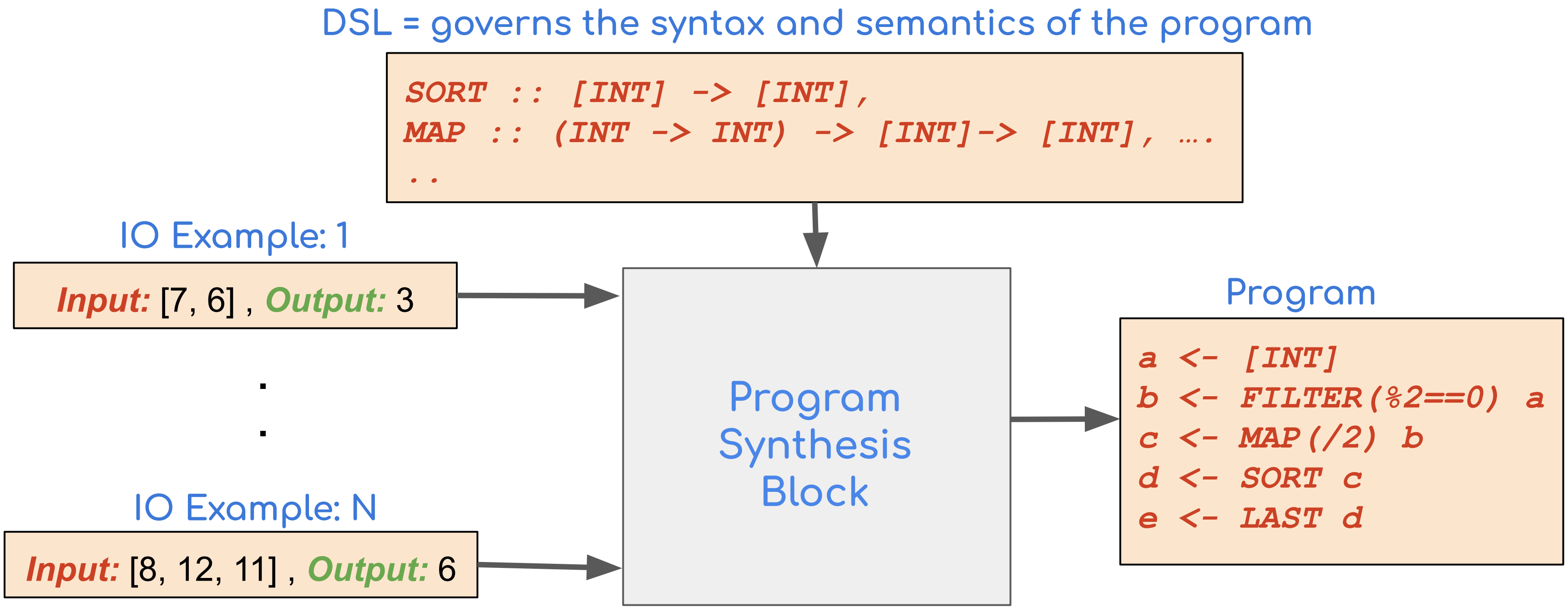
Code, data and trained checkpoints: https://github.com/shrivastavadisha/N-PEPS

Task: Program Synthesis by Examples



- Given a set of N IO examples, find a program that satisfies those examples.

Task: Program Synthesis by Examples



- Given a set of N IO examples, find a program that satisfies those examples.
- Given a ***timeout*** value to be practically meaningful.

Neural Per-Example Program Synthesis (N-PEPS)

#1. [154, -252, -228, -85, -136], [109, 65, -3, 71, 189] -> [ ]
#2: [-113, 240, -59, 66], [-197, 150] -> [-240, -66]
#3: [-7, 106, -138], [225, 97, 17] -> [ ]
#4: [-140, -51, 155, 74, -21], [35, 82, -103]-> [-155, -74]
#5: [87, -115, 52], [177, 193, -17] -> [-52]

p_g

```
   a <- LIST
   b <- LIST
1: c <- COUNT (>0) b
2: d <- DROP c a
3: e <- MAP (*-1) d
4: f <- FILTER (<0) e
```

Global Program Synthesis (**GPS**)

- Find **global solution** p_g that satisfies all IO examples ***simultaneously***
- Can be hard

Neural Per-Example Program Synthesis (N-PEPS)

#1. [154, -252, -228, -85, -136], [109, 65, -3, 71, 189] -> [ ]
#2: [-113, 240, -59, 66], [-197, 150] -> [-240, -66]
#3: [-7, 106, -138], [225, 97, 17] -> [ ]
#4: [-140, -51, 155, 74, -21], [35, 82, -103]-> [-155, -74]
#5: [87, -115, 52], [177, 193

p_g

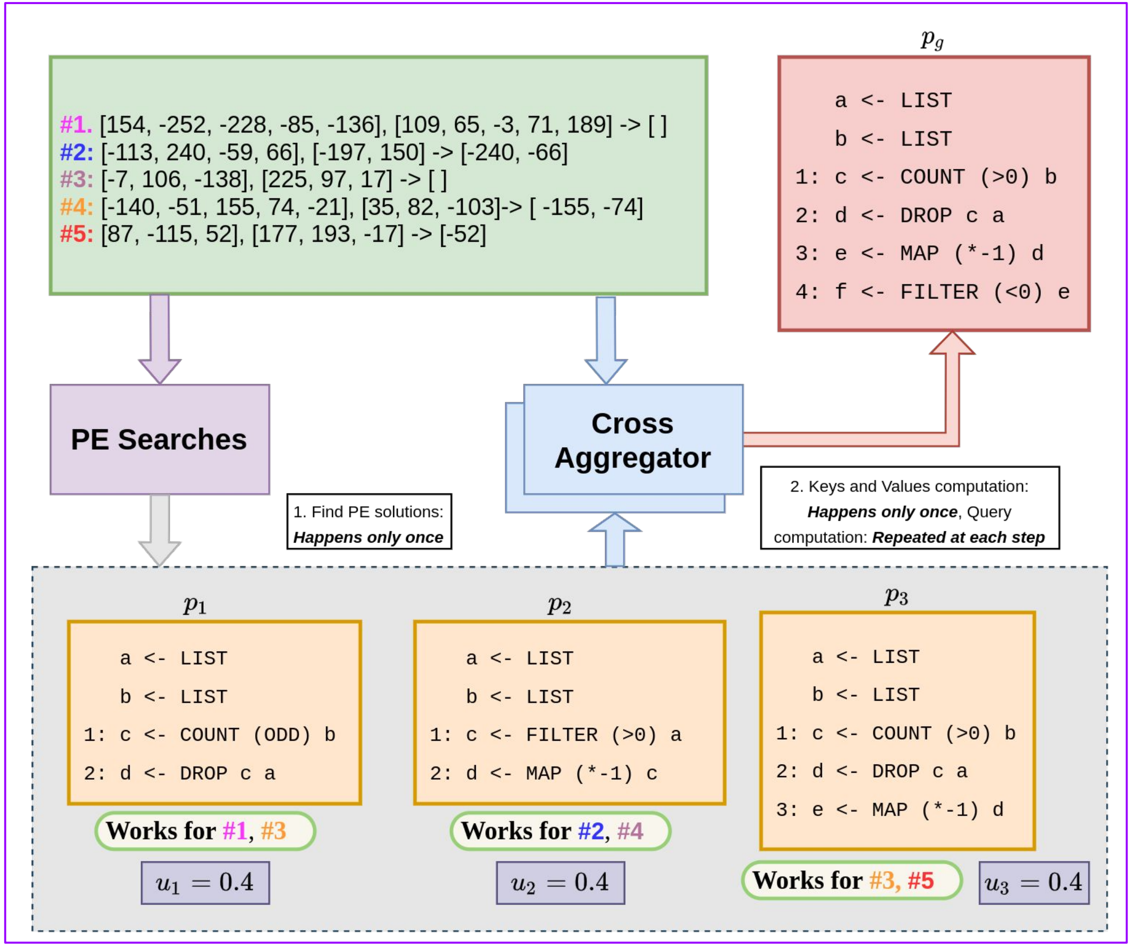
```
   a <- LIST
   b <- LIST
1: c <- COUNT (>0) b
```

Global Program Synthesis (**GPS**)

- Find global solution p_g that satisfies all IO examples ***simultaneously***

- Break a hard problem into smaller, easy to solve subproblems
- Learn to combine the solutions of the sub-problems such that the harder problem is solved

Neural Per-Example Program Synthesis (N-PEPS)



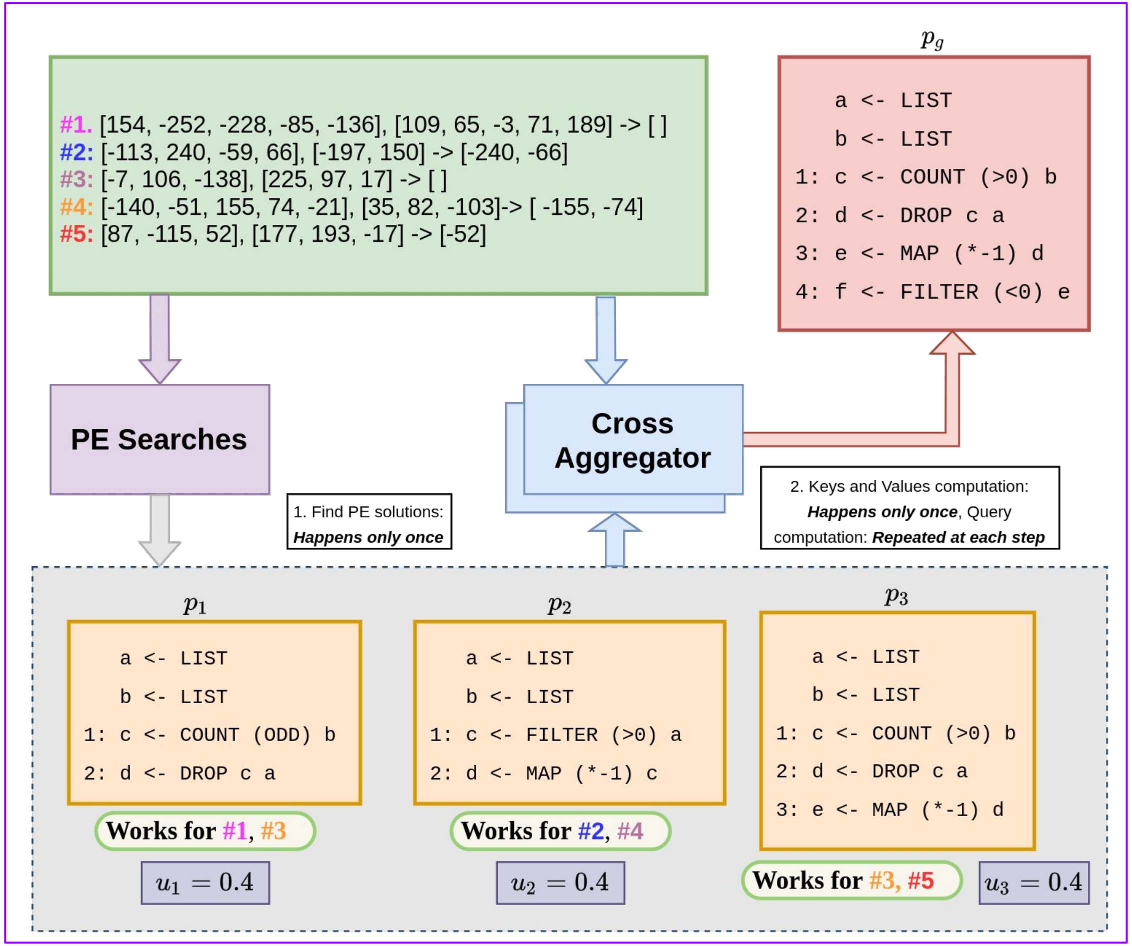
Global Program Synthesis (**GPS**)

- Find global solution p_g that satisfies all IO examples ***simultaneously***
- Can be hard

Per Example Program Synthesis (**PEPS**). Break into two stages:

- **Enhance:** Find programs that satisfy a single example (PE solutions) - fast
- **Predict:** Combine the PE solutions such that it leads to the global solution

Neural Per-Example Program Synthesis (N-PEPS)



Global Program Synthesis (**GPS**)

- Find global solution p_g that satisfies all IO examples ***simultaneously***
- Can be hard

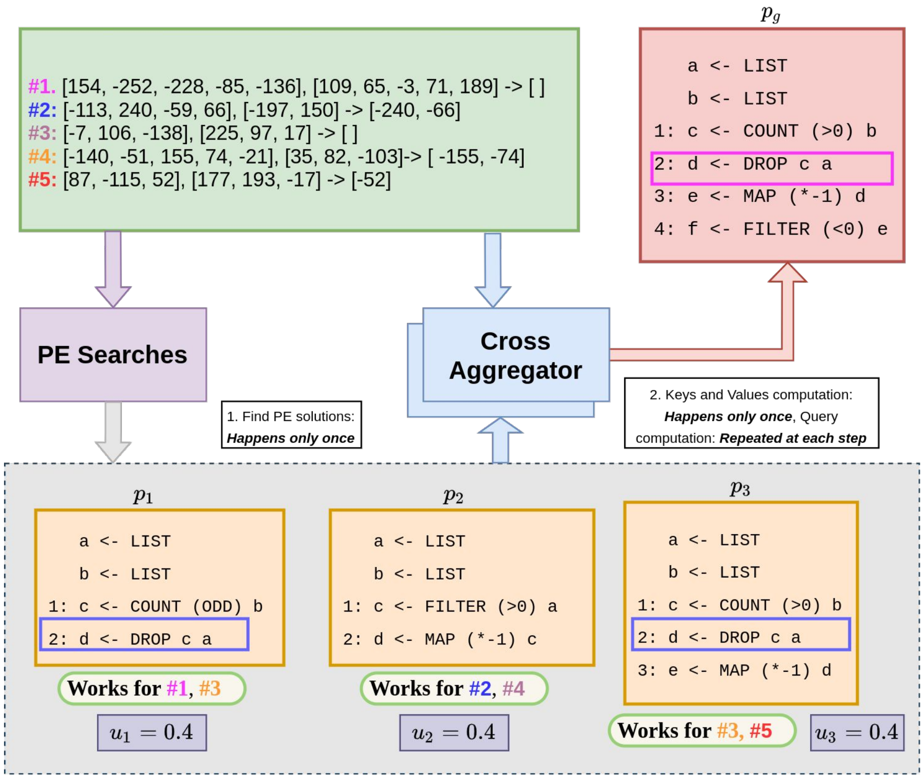
Per Example Program Synthesis (**PEPS**): Break into two stages:

- **Enhance:** Find programs that satisfy a single example (***PE solutions***) - fast
- **Predict:** Combine the PE solutions such that it leads to the global solution
- We propose an architecture called **Cross Aggregator (CA)** that ***learns*** to combine the PE solutions.

We use neural networks for both these stages (PE Searches and CA): **N-PEPS**

Cross Aggregator (CA)

Idea: If a PE program state* has high relevance with the global program state at a given step, then the following PE program line is likely to be useful for synthesizing the next line of p_g.

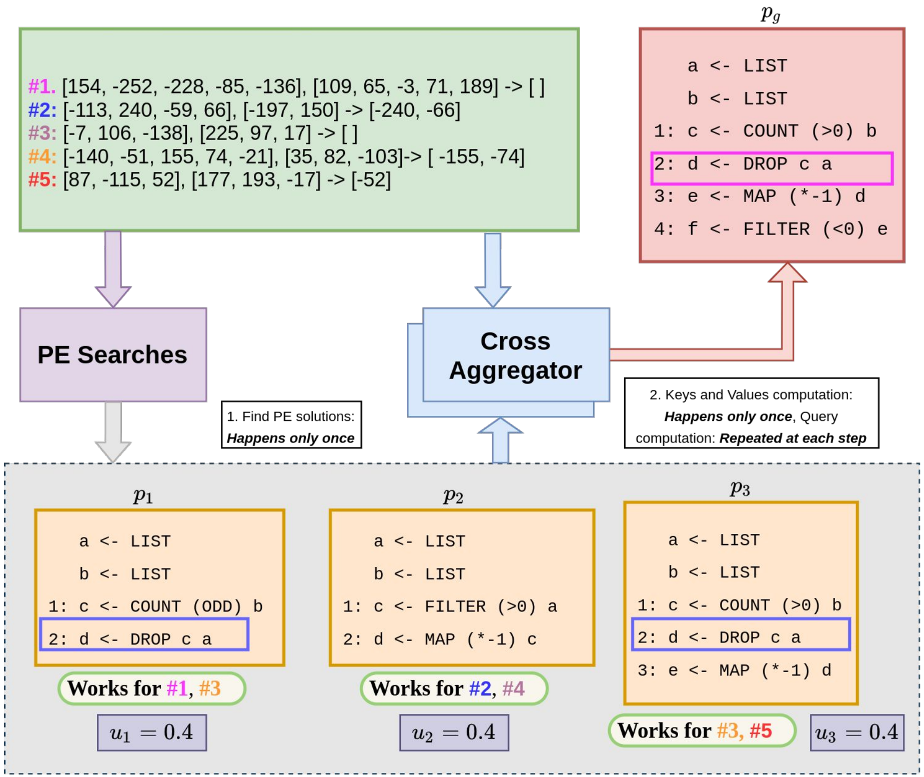


* Program state at step t = Vector representing the values of variables obtained by executing t lines of the program.

*[Automatic program synthesis of long programs with a learned garbage collector". Zohar & Wolf, *NeurIPS 2018*](#)

Cross Aggregator (CA)

Idea: If a PE program state* has high relevance with the global program state at a given step, then the following PE program line is likely to be useful for synthesizing the next line of p_g.



Model: Multi-head cross-attention mechanism

Query = Global program state at step t

Key = PE program state at step t

Value = PE program line t+1

*Automatic program synthesis of long programs with a learned garbage collector". Zohar & Wolf, *NeurIPS 2018*

Results

Timeout for all methods = 5s

Model	Success Ratio
PCCoder [29]	77.75 ± 0.38
Sum-PEPS	82.71 ± 0.32
Mean-PEPS	82.68 ± 0.33
Mean-PEPS+$\mathcal{U}$	82.70 ± 0.32
N-PEPS	86.22 ± 0.25
N-PEPS+$\mathcal{U}$	$\mathbf{87.07 \pm 0.28}$

GPS*

Use aggregation mechanisms other than CA

Leading neural program synthesis technique for the space of programs we work on

Train: programs uptil length 4
Test: programs of length 4

*Automatic program synthesis of long programs with a learned garbage collector". Zohar & Wolf, *NeurIPS 2018*

Results

Timeout for all methods = 5s

Model	Success Ratio
PCCoder [29]	77.75 ± 0.38
Sum-PEPS	82.71 ± 0.32
Mean-PEPS	82.68 ± 0.33
Mean-PEPS+$\mathcal{U}$	82.70 ± 0.32
N-PEPS	86.22 ± 0.25
N-PEPS+$\mathcal{U}$	$\mathbf{87.07 \pm 0.28}$

GPS

Use aggregation mechanisms other than CA

N-PEPS significantly improves the success rate over GPS and other ablation baselines.

Train: programs uptil length 4
Test: programs of length 4

Model	Length = 5	Length = 8	Length = 10	Length = 12	Length=14
PCCoder [29]	70.91 ± 0.35	44.17 ± 0.45	28.18 ± 0.33	19.69 ± 0.34	14.71 ± 0.23
Sum-PEPS	76.45 ± 0.33	43.4 ± 0.56	28.96 ± 0.27	20.94 ± 0.32	15.67 ± 0.32
Mean-PEPS	75.79 ± 0.31	44.42 ± 0.51	29.55 ± 0.29	21.45 ± 0.27	16.35 ± 0.27
Mean-PEPS+$\mathcal{U}$	75.99 ± 0.32	44.49 ± 0.52	29.75 ± 0.25	21.74 ± 0.30	16.45 ± 0.33
N-PEPS	79.18 ± 0.31	$\mathbf{47.23 \pm 0.49}$	$\mathbf{32.3 \pm 0.34}$	$\mathbf{23.34 \pm 0.28}$	$\mathbf{17.35 \pm 0.31}$
N-PEPS+$\mathcal{U}$	$\mathbf{79.19 \pm 0.30}$	46.31 ± 0.61	31.84 ± 0.36	22.71 ± 0.28	16.68 ± 0.21

Train: programs uptil length 12

Test: programs of lengths 5, 8, 10, 12 and 14

Takeaways

Connection to our Framework

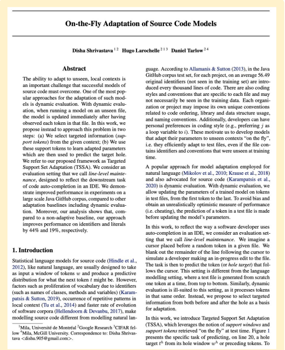
- **Input X** = set of given IO examples, **Target Y** = step t of the global solution
- **Context Meta-Info W** = Same as X
- **Support Context Z** = PE solutions (values) + step-wise PE execution states (keys) + execution state of step t -1 of the global solution (query)
- **Enhance** = PE model (for PE solutions) + code interpreter (for execution states)
- **Predict** = Cross Aggregator (CA)

Future Work

- Generalize to programs with loops and conditionals.
- Extend the idea to LLMs.

Outline

On-the-Fly Adaptation of Source Code Models
(CAP Workshop, NeurIPS 2020)



RepoFusion: Training Code Models to Understand Your Repository
(under review at NeurIPS 2023)

Introduction → Article 1 → **Article 2** → Article 3 → Article 4 → Final Thoughts

Learning to Combine Per-Example Solutions for Neural Program Synthesis
(NeurIPS 2021)

Repository Level Prompt Generation for Large Language Models of Code
(ICML 2023)

On-the-Fly Adaptation of Source Code Models

Workshop on Computer Assisted Programming
NeurIPS 2020

Université de Montréal

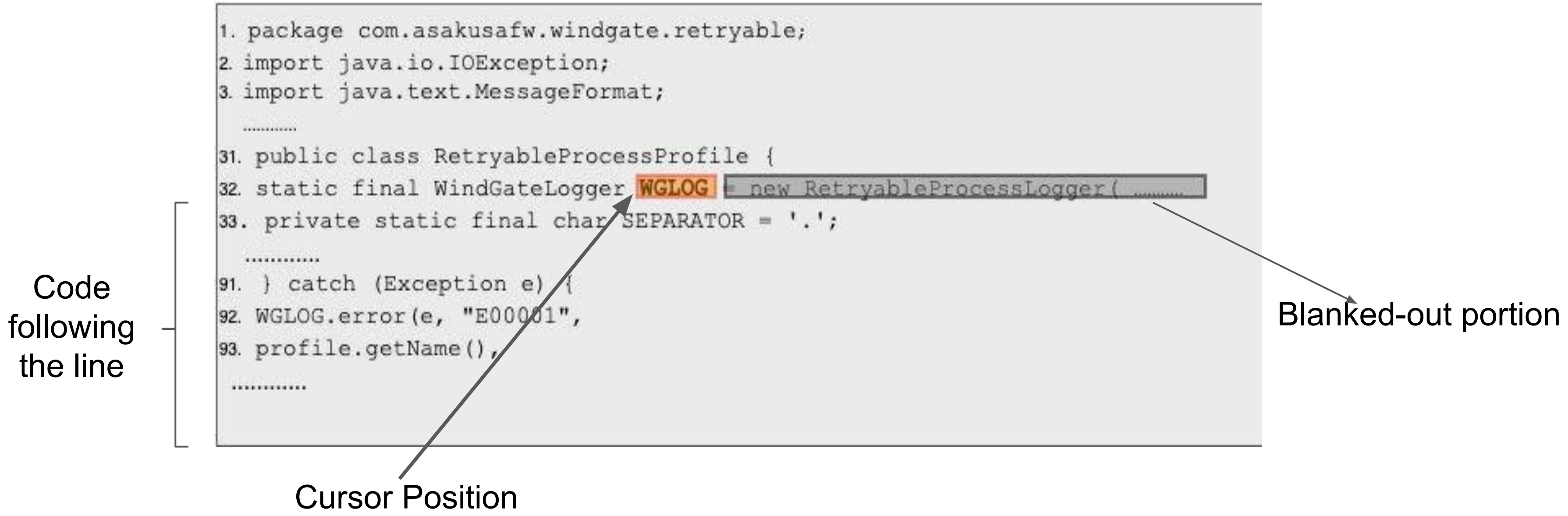
Mila

Google Research

Task: Code Completion in an IDE

Our setting simulates editing a file in an IDE

- Objective: **Complete the first token following the cursor** (***target hole***)
- There can be code following the completion line.
- **Rest of the line** is blanked.

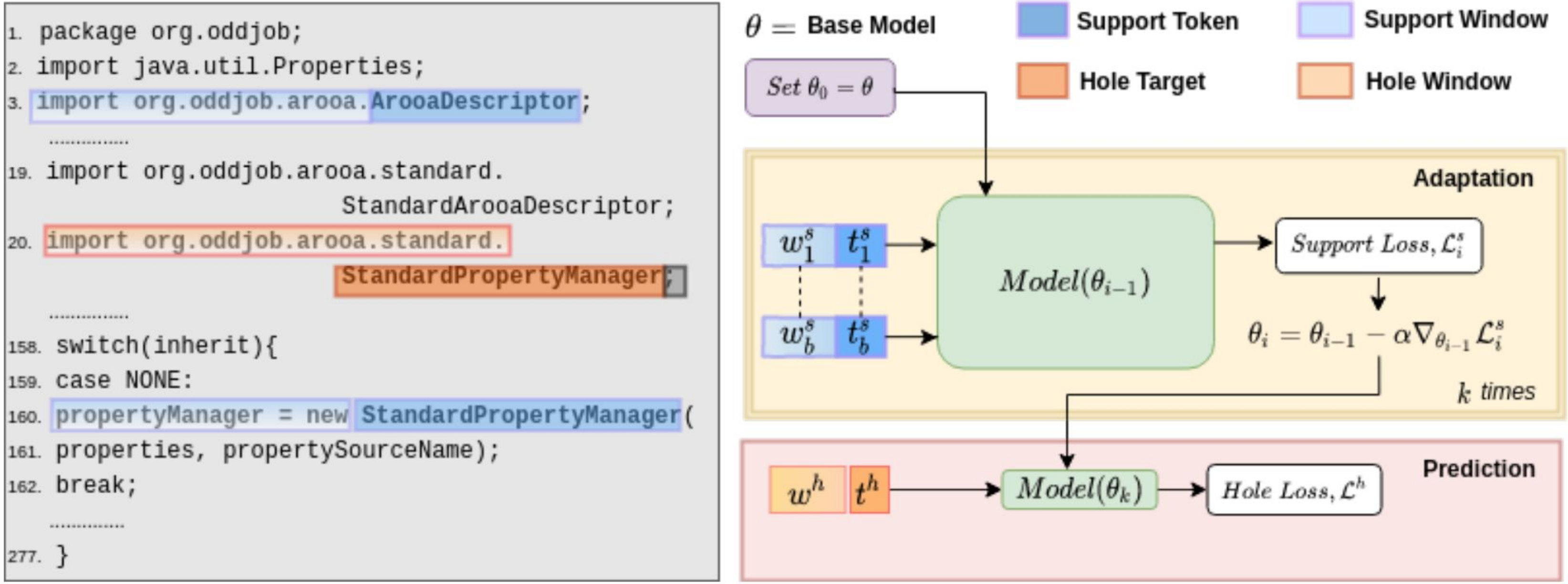


Motivation: Why Adaptation of Source Code Models?

- Models struggle when encountered with code not seen during training.
- Models need to adapt to local, unseen context
  - New Identifiers
  - Organization or project specific coding constructs
    - Variable naming conventions (get_access vs getAccess)
    - Data structures/ libraries used (from google3 import b)
  - Developer-specific preferences
    - for (int i = 0, …) vs for (int j = 0, …)
    - Comments before each line or each method

Targeted Support Set Adaptation (TSSA)

- **Enhance:** Obtain support tokens, e.g. frequent in current file but rare overall.
- **Predict:** Adapt the model based on the support context.
    - *Inner update*: support window -> support token (k steps of gradient update)
    - *Outer update*: hole window -> hole target (using updated parameters)



Results

Test Performance on Target Hole Prediction

Model	Cross Entropy	MRR@10 (All)(%)	MRR@10 (Identifiers)(%)	Recall@10 (All)(%)	Recall@10 (Identifiers) (%)
Base Model	5.222 ± 0.10	65.20 ± 0.42	24.90 ± 0.64	75.74 ± 0.42	36.20 ± 0.78
Dynamic Evaluation	3.540 ± 0.08	$\mathbf{68.95 \pm 0.41}$	34.44 ± 0.70	80.39 ± 0.39	48.86 ± 0.82
TSSA-1	3.461 ± 0.07	66.94 ± 0.40	35.76 ± 0.70	81.00 ± 0.38	52.04 ± 0.82
TSSA-8	3.383 ± 0.06	67.52 ± 0.40	35.14 ± 0.70	80.65 ± 0.38	50.27 ± 0.82
TSSA-16	$\mathbf{3.240 \pm 0.06}$	68.63 ± 0.40	$\mathbf{36.74 \pm 0.70}$	$\mathbf{81.51 \pm 0.38}$	$\mathbf{52.34 \pm 0.82}$

- **Model architecture:** Seq2seq encoder decoder network with single-layer GRU.
- **Base Model:** no adaptation
- **Dynamic Evaluation*:** Support tokens consist of tokens from context before the target hole.
- **TSSA-k:** TSSA with k updates with support tokens from both before and after the target hole.
- We set k = avg. # of updates performed by dynamic evaluation = 16 for our test data.

TSSA improves upon adaptation (dynamic evaluation) and non-adaptation baselines, even with half the #steps on some metrics.

*Open-Vocabulary Models for Source Code Karampatsis et al. (2020)

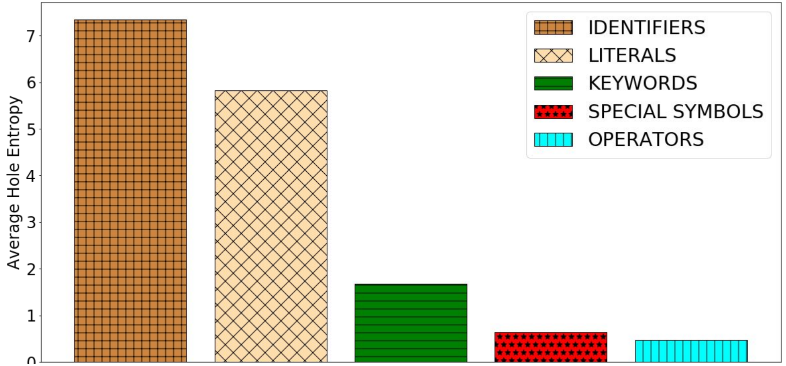
Results

Test Performance on Target Hole Prediction

Model	Cross Entropy	MRR@10 (All)(%)	MRR@10 (Identifiers)(%)	Recall@10 (All)(%)	Recall@10 (Identifiers) (%)
Base Model	5.222 ± 0.10	65.20 ± 0.42	24.90 ± 0.64	75.74 ± 0.42	36.20 ± 0.78
Dynamic Evaluation	3.540 ± 0.08	$\mathbf{68.95 \pm 0.41}$	34.44 ± 0.70	80.39 ± 0.39	48.86 ± 0.82
TSSA-1	3.461 ± 0.07	66.94 ± 0.40	35.76 ± 0.70	81.00 ± 0.38	52.04 ± 0.82
TSSA-8	3.383 ± 0.06	67.52 ± 0.40	35.14 ± 0.70	80.65 ± 0.38	50.27 ± 0.82
TSSA-16	$\mathbf{3.240 \pm 0.06}$	68.63 ± 0.40	$\mathbf{36.74 \pm 0.70}$	$\mathbf{81.51 \pm 0.38}$	$\mathbf{52.34 \pm 0.82}$

Most of the improvement comes from identifiers and literals.

Test Performance across different token-types



Token Type	Base model	TSSA-16	% Improvement
Identifiers	13.16	7.35	44.15
Literals	7.18	5.82	18.94

Takeaways

Connection to our Framework

- **Input X** = hole window, **Target Y** = target hole (next token after the cursor)
- **Context Meta-Info W** = position of the cursor + current file
- **Support Context Z** = support tokens + support windows from the current file
- **Enhance** = targeted selection of support context, e.g. strategies based on frequency of occurrence of tokens
- **Predict** = TSSA

Future Work

- Better ways of obtaining the support context
  - Extend the scope from current file to the entire repository.
  - Automated, Example-specific selection
- Leverage the power of pretrained LLMs
  - Expensive to perform gradient updates

Outline

On-the-Fly Adaptation of Source Code Models
(CAP Workshop, NeurIPS 2020)

RepoFusion: Training Code Models to Understand Your Repository
(under review at NeurIPS 2023)

Introduction → **Article 1** → **Article 2** → **Article 3** → **Article 4** → **Final Thoughts**

Learning to Combine Per-Example Solutions for Neural Program Synthesis
(NeurIPS 2021)

Repository Level Prompt Generation for Large Language Models of Code
(ICML 2023)

Repository-Level Prompt Generation for Large Language Models of Code

ICML 2023

Université de Montréal

Mila

Google Research

Code, data and trained checkpoints: https://github.com/shrivastavadisha/repo_level_prompt_generation

Motivation: Large Language Models (LLMs) of Code

- Used in code-assistants (e.g. GitHub Copilot, Bard).
- Struggle when encountered with code not seen during training.
    - Proprietary Software
    - WIP Code Project
- Finetuning on code from the local repository is often impractical
    - Black-box access to strong code LLMs.
    - Computationally expensive as well as challenging to update frequently.
- Building upon previous work, leverage relevant context from other files in the repository (e.g. imports, parent classes), but only during inference.

Motivation: Large Language Models (LLMs) of Code

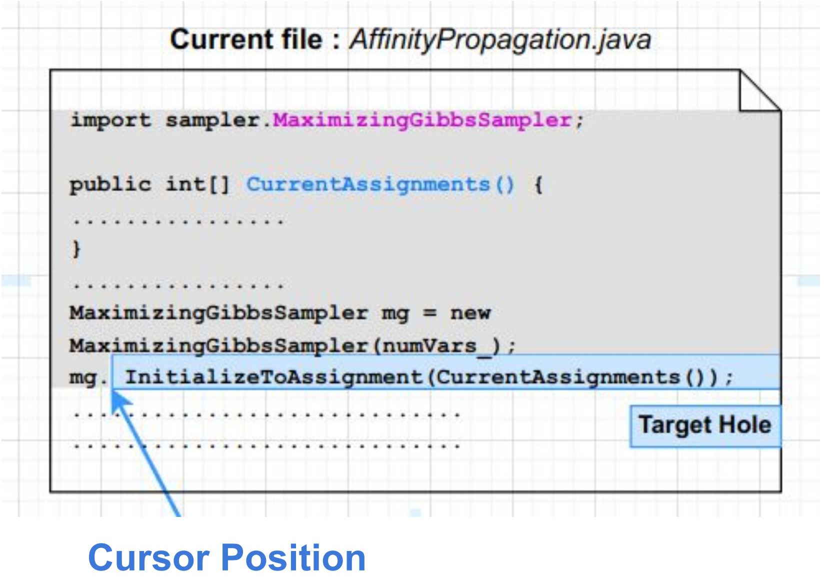
- Used in code-assistants (e.g. GitHub Copilot, Bard).
- Struggle when encountered with code not seen during training.
    - Proprietary Software
    - WIP Code Project
- Finetuning on code from the local repository is often impractical
    - Black-box access to strong code LLMs.
    - Computationally expensive as well as challenging to update frequently.
- Building upon previous work, leverage relevant context from other files in the repository (e.g. imports, parent classes), but only during inference.

Select relevant repository context in a way that doesn't require access to the weights of the LLM.

Task: Single-line Code Completion in an IDE

Our setting simulates editing a file in an IDE

- Objective: **Complete the line following the cursor** (***target hole***)
- There can be code after the cursor line.



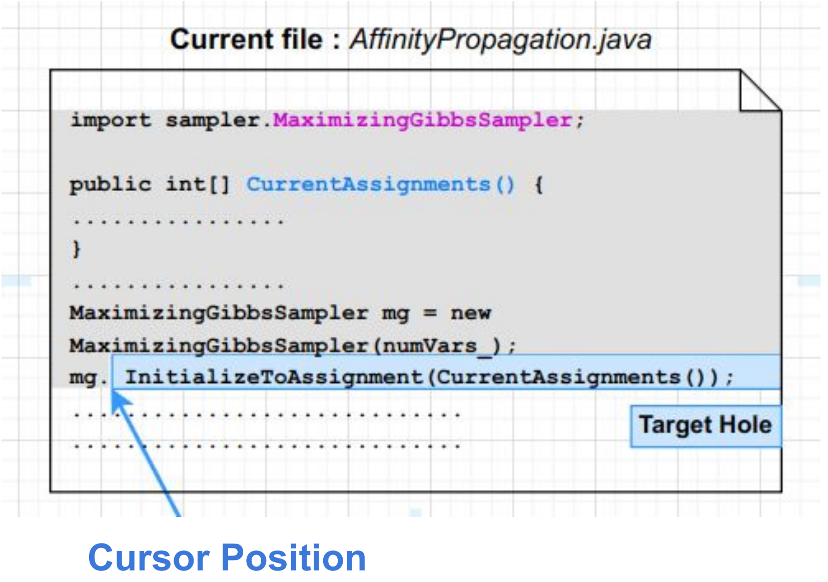
Task: Single-line Code Completion in an IDE

Our setting simulates editing a file in an IDE

- Objective: **Complete the line following the cursor** (***target hole***)
- There can be code after the cursor line.

Vanilla Training: given a prefix of code, predict the next tokens.

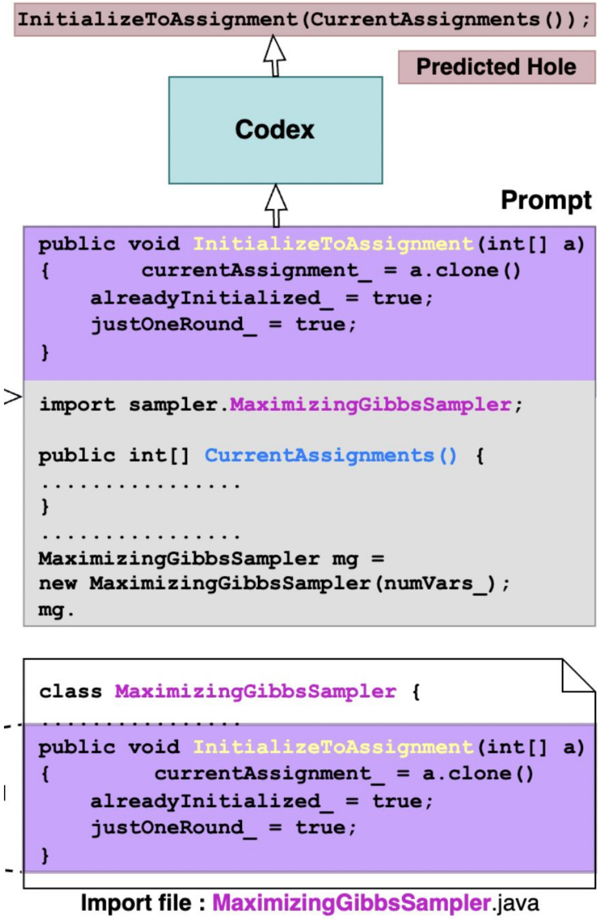
Vanilla Inference (to match the training): take **context prior to the cursor** in the current file and predict the **target hole**.



Repository Context in the Prompt

Take an LLM trained in the usual way, but use it differently during inference.

During inference, in addition to the prior context in the current file, we add relevant context from the repository in the prompt.



Repository Context in the Prompt

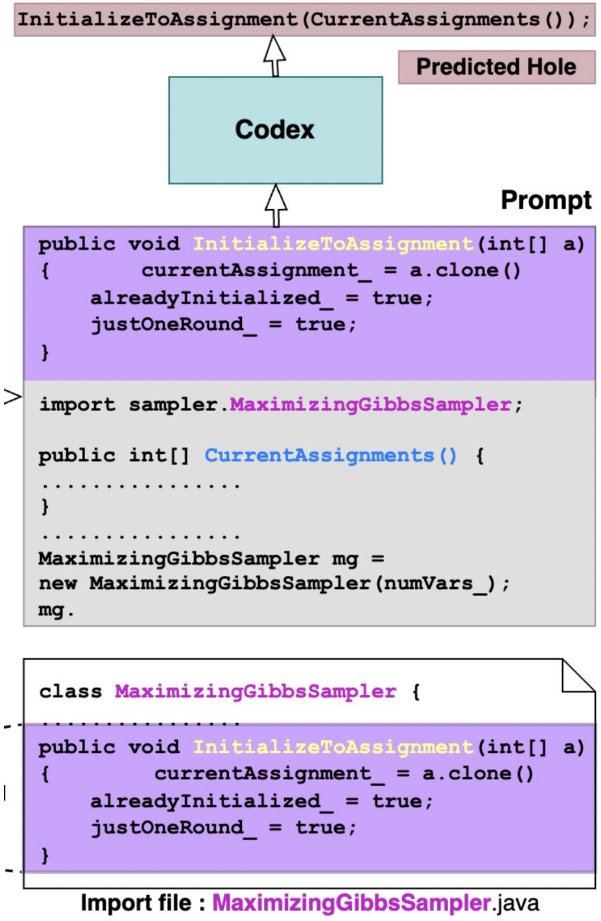
Take an LLM trained in the usual way, but use it differently during inference.

During inference, in addition to the prior context in the current file, we add relevant context from the repository in the prompt.

To select relevant context, we want a method that

- Utilizes Structure of the repository
- Utilizes Context in relevant files

Solution: Use domain knowledge to guide the selection of relevant context via a set of **prompt proposals.**



Prompt Proposals

- **Prompt Source**: where to take the context from?
- **Prompt Context Type**: what to take from the prompt source?

Prompt Proposals

- **Prompt Source**: where to take the context from?
- **Prompt Context Type**: what to take from the prompt source?

10 Prompt Sources

- Current file
- Parent Class file
- Sibling file
- Similar name file
- Child Class file
- Import of the above

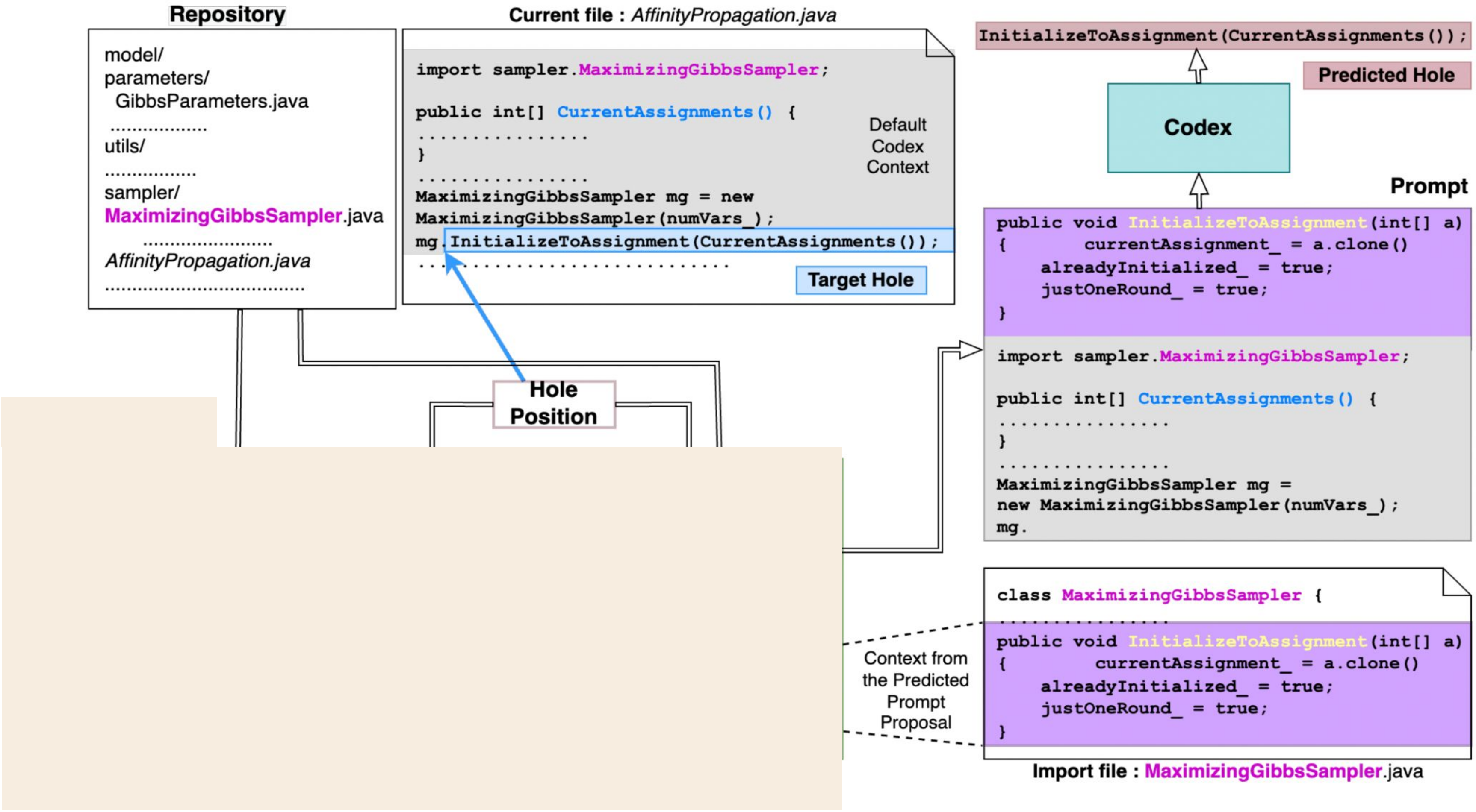
In total, we propose a list of **63** prompt proposals

7 Prompt Context Types*

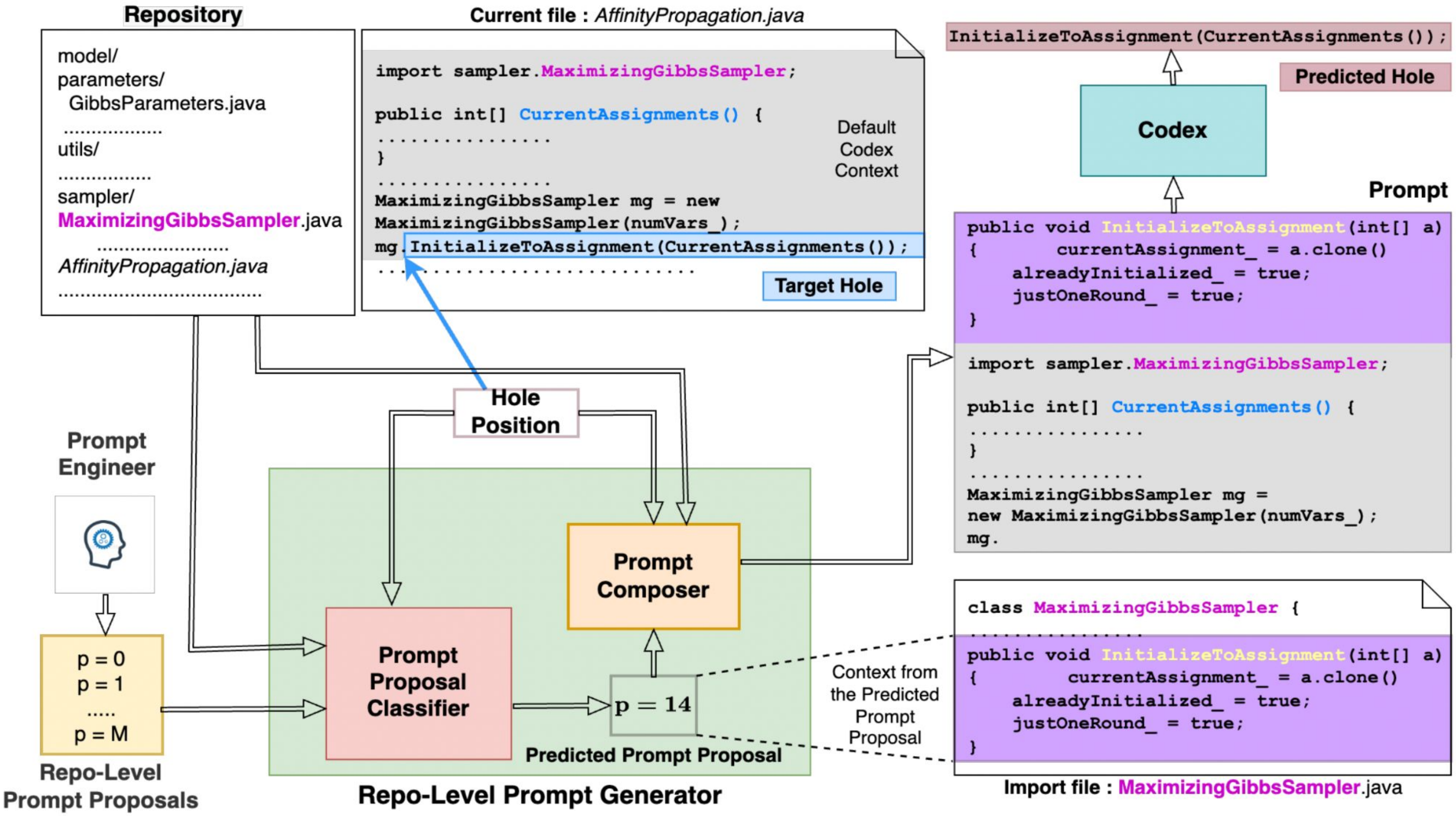
- Lines after the cursor
- Identifiers
- Field declarations
- Type identifiers
- String literals
- Method names
- Method names and bodies

*Inspired by findings from On-the-Fly Adaptation of Source Code Models, *Disha Shrivastava, Hugo Larochelle, Daniel Tarlow*

Repo-Level Prompt Generator (RLPG)



Repo-Level Prompt Generator (RLPG)



Prompt Proposal Classifier

- Multi-label binary classifier that *learns* to select a prompt proposal that is likely to lead to a successful prediction for the target hole.

- *Success* = When inclusion of the context from the prompt proposal in the prompt leads to an accurate prediction of the hole.

- *Example-Specific:* different prediction conditioned on the hole.

Results

Table 2. Performance of the oracle relative to Codex.

Data Split	Success Rate Codex(%)	Success Rate Oracle(%)	Rel. ↑ over Codex(%)
Train	59.78	80.29	34.31
Val	62.10	79.05	27.28
Test	58.73	79.63	35.58

Including contexts from our prompt proposals during inference is quite useful even though Codex has not seen them during training.

Results

Table 2. Performance of the oracle relative to Codex.

Data Split	Success Rate Codex(%)	Success Rate Oracle(%)	Rel. ↑ over Codex(%)
Train	59.78	80.29	34.31
Val	62.10	79.05	27.28
Test	58.73	79.63	35.58

Table 3. Success Rate (SR) of different methods on the test data when averaged across all holes.

	Method	**Success Rate(%)**	**Rel. ↑(%)**
	Codex (Chen et al., 2021)	58.73	-
	Oracle	79.63	35.58
Retrieval Baselines	Random	58.13	-1.02
	Random NN	58.98	0.43
	File-level BM25	63.14	7.51
	Identifier Usage (Random)	64.93	10.55
	Identifier Usage (NN)	64.91	10.52
Non-learned RLPG	Fixed Prompt Proposal	65.78	12.00
	RLPG-BM25	66.41	13.07
Learned RLPG	RLPG-H	68.51	16.65
	RLPG-R	67.80	15.44

Using RLPG with prompt proposal classifier shows significant improvements.

Takeaways

Connection to our Framework

- **Input X** = all tokens prior to the cursor in the current file, **Target Y** = tokens after the cursor till end of line.
- **Context Meta-Info W** = position of the cursor + current file's repository
- **Support Context Z** = context from a single prompt proposal predicted by RLPG
- **Enhance** = Prompt Proposals + RLPG
- **Predict** = LLM of Code

Future Work

- Automatically combine contexts from multiple prompt proposals.
- Scale the evaluation to larger data and include comparisons with more code LLMs.

Outline

On-the-Fly Adaptation of Source Code Models
(CAP Workshop, NeurIPS 2020)

RepoFusion: Training Code Models to Understand Your Repository
(under review at NeurIPS 2023)

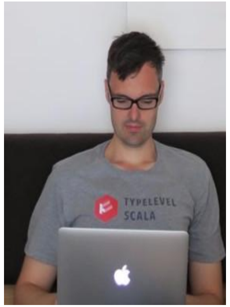
Introduction | **Article 1** | **Article 2** | **Article 3** | **Article 4** | **Final Thoughts**

Learning to Combine Per-Example Solutions for Neural Program Synthesis
(NeurIPS 2021)

Repository Level Prompt Generation for Large Language Models of Code
(ICML 2023)

RepoFusion: Training Code Models to Understand Your Repository

arXiv 2023 (under review)



Université de Montréal

Mila

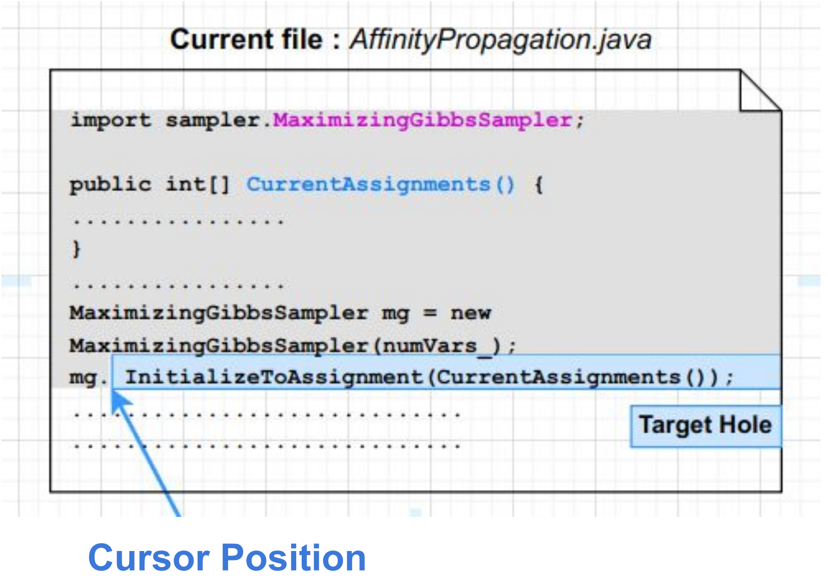
servicenow

Code, data and trained checkpoints: https://huggingface.co/RepoFusion

Task: Single-line Code Completion in an IDE

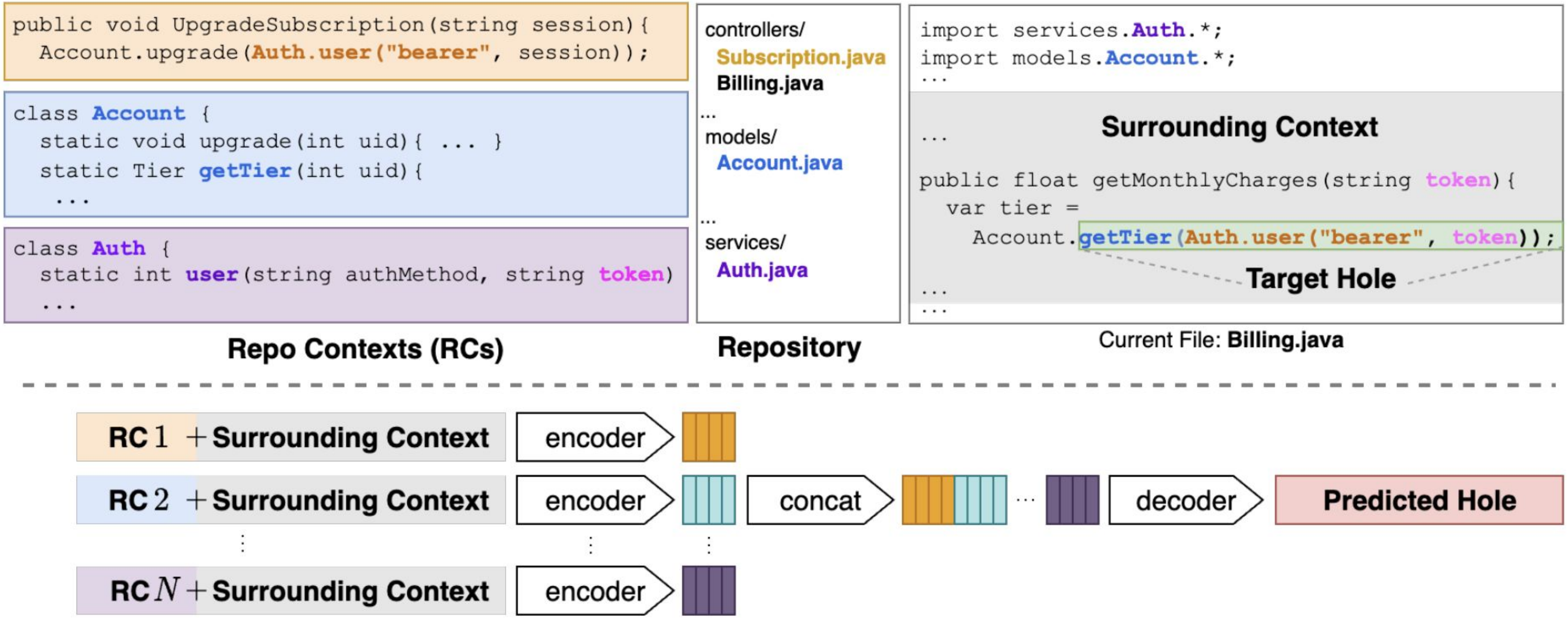
Our setting simulates editing a file in an IDE

- Objective: **Complete the line following the cursor** (***target hole***)
- There can be code after the cursor line.



RepoFusion

Train a model to **combine multiple relevant contexts** coming from the repository (repo contexts) such that it leads to an **accurate** prediction of the target hole.



Results

RepoFusion (220M) outperforms ~73X larger (CodeGen-16B) models trained with next-token prediction.

N = #RCs
l = size (# tokens) of each RC

RepoFusion (220M) is at par with ~70X larger StarCoder-15.5B model trained with Fill-in-the-Middle.

Model	Size (#params)	Effective context length	Context type	Success Rate (%)
CodeT5-base (FT)	0.22B	2048	prior	41.82 ± 0.12
CodeT5-base (FT)	0.22B	4096	prior	46.45 ± 0.12
CodeT5-large (FT)	0.77B	2048	prior	44.73 ± 0.12
CodeT5-large (FT)	0.77B	4096	prior	48.92 ± 0.12
SantaCoder	1.1B	2048	prior	39.51 ± 0.12
CodeGen	2B	2048	prior	49.45 ± 0.12
CodeGen	6B	2048	prior	49.19 ± 0.12
CodeGen	16B	2048	prior	50.20 ± 0.12
CodeT5-base (FT)	0.22B	2048	post+prior	48.89 ± 0.12
CodeT5-base (FT)	0.22B	4096	post+prior	49.97 ± 0.12
CodeT5-large (FT)	0.77B	2048	post+prior	51.72 ± 0.12
CodeT5-large (FT)	0.77B	4096	post+prior	52.43 ± 0.12
SantaCoder	1.1B	2048	post+prior	56.78 ± 0.12
CodeGen	2B	2048	post+prior	53.18 ± 0.12
CodeGen	6B	2048	post+prior	54.03 ± 0.12
CodeGen	16B	2048	post+prior	54.09 ± 0.12
RepoFusion ($N = 4$, $l = 512$)	0.22B	2048	NT-Prior-Last	65.96 ± 0.12
RepoFusion ($N = 8$, $l = 512$)	0.22B	4096	NT-Prior-Last	70.38 ± 0.11
RepoFusion ($N = 32$, $l = 768$)	0.22B	24576	NT-Prior-Last	77.32 ± 0.10

StarCoderBase	15.5B	8192	prior	52.97 ± 0.45
StarCoderBase	15.5B	8192	post+prior	79.79 ± 0.36
RepoFusion ($N = 16$, $l = 512$)	0.22B	8192	NT-Prior-Last	73.67 ± 0.43
RepoFusion ($N = 32$, $l = 2500$)	0.22B	80000	NT-Prior-Last	78.33 ± 0.37

Results

Rep
outper
(CodeGen
with nex

Training smaller models with repository context using RepoFusion is better or at par with training significantly larger models without such context.

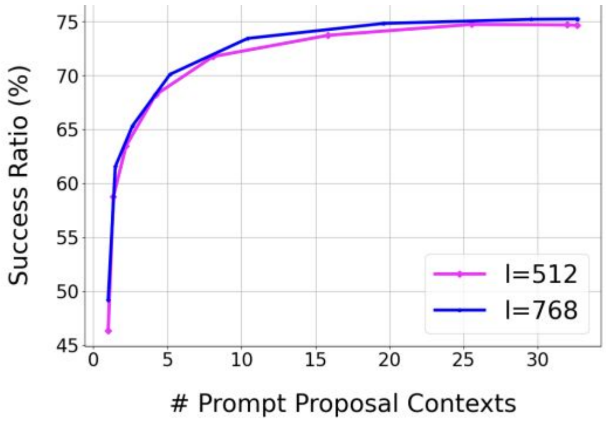
Model	Size (#params)	Effective context length	Context type	Success Rate (%)
CodeT5-base (FT)	0.22B	2048	prior	41.82 ± 0.12
CodeT5-base (FT)	0.22B	4096	prior	46.45 ± 0.12
CodeT5-large (FT)	0.77B	2048	prior	44.73 ± 0.12
CodeT5-large (FT)	0.77B	4096	prior	48.92 ± 0.12
SantaCoder	1.1B	2048	prior	39.51 ± 0.12
CodeGen	2B	2048	prior	49.45 ± 0.12
				49.19 ± 0.12
				50.20 ± 0.12
			or	48.89 ± 0.12
			or	49.97 ± 0.12
			or	51.72 ± 0.12
			or	52.43 ± 0.12
			or	56.78 ± 0.12
			or	53.18 ± 0.12
			or	54.03 ± 0.12
			or	54.09 ± 0.12
RepoFusion ($N = 4$, $l = 512$)	0.22B	2048	NT-Prior-Last	65.96 ± 0.12
RepoFusion ($N = 8$, $l = 512$)	0.22B	4096	NT-Prior-Last	70.38 ± 0.11
RepoFusion ($N = 32$, $l = 768$)	0.22B	24576	NT-Prior-Last	77.32 ± 0.10

N = #RCs
l = size (# tokens) of each RC

StarCoderBase	15.5B	8192	prior	52.97 ± 0.45
StarCoderBase	15.5B	8192	post+prior	79.79 ± 0.36
RepoFusion ($N = 16$, $l = 512$)	0.22B	8192	NT-Prior-Last	73.67 ± 0.43
RepoFusion ($N = 32$, $l = 2500$)	0.22B	80000	NT-Prior-Last	78.33 ± 0.37

RepoFusion (220M) is at par with ~70X larger StarCoder-15.5B model trained with Fill-in-the-Middle.

Results



Performance scales with incorporation diverse repo contexts from multiple sources.

Model	Size (#params)	Effective context length	Context type	Success Rate (%)
CodeT5-base (FT)	0.22B	2048	prior	41.82 ± 0.12
CodeT5-base (FT)	0.22B	4096	prior	46.45 ± 0.12
CodeT5-large (FT)	0.77B	2048	prior	44.73 ± 0.12
CodeT5-large (FT)	0.77B	4096	prior	48.92 ± 0.12
SantaCoder	1.1B	2048	prior	39.51 ± 0.12
CodeGen	2B	2048	prior	49.45 ± 0.12
CodeGen	6B	2048	prior	49.19 ± 0.12
CodeGen	16B	2048	prior	50.20 ± 0.12
CodeT5-base (FT)	0.22B	2048	post+prior	48.89 ± 0.12
CodeT5-base (FT)	0.22B	4096	post+prior	49.97 ± 0.12
CodeT5-large (FT)	0.77B	2048	post+prior	51.72 ± 0.12
CodeT5-large (FT)	0.77B	4096	post+prior	52.43 ± 0.12
SantaCoder	1.1B	2048	post+prior	56.78 ± 0.12
CodeGen	2B	2048	post+prior	53.18 ± 0.12
CodeGen	6B	2048	post+prior	54.03 ± 0.12
CodeGen	16B	2048	post+prior	54.09 ± 0.12
RepoFusion ($N = 4$, $l = 512$)	0.22B	2048	NT-Prior-Last	65.96 ± 0.12
RepoFusion ($N = 8$, $l = 512$)	0.22B	4096	NT-Prior-Last	70.38 ± 0.11
RepoFusion ($N = 32$, $l = 768$)	0.22B	24576	NT-Prior-Last	77.32 ± 0.10

StarCoderBase	15.5B	8192	prior	52.97 ± 0.45
StarCoderBase	15.5B	8192	post+prior	79.79 ± 0.36
RepoFusion ($N = 16$, $l = 512$)	0.22B	8192	NT-Prior-Last	73.67 ± 0.43
RepoFusion ($N = 32$, $l = 2500$)	0.22B	80000	NT-Prior-Last	78.33 ± 0.37

Takeaways

Connection to our Framework

- **Input X** = all tokens prior to the cursor in the current file, **Target Y** = tokens after the cursor till the end of line.
- **Context Meta-Info W** = position of the cursor + current file's repository
- **Support Context Z** = multiple repo contexts
- **Enhance** = module for obtaining repo contexts
- **Predict** = RepoFusion

Future Work

Leverage contextual cues from other relevant sources such as API documentations, StackOverflow, bug reports, GitHub issues.

We create and release Stack-Repo, a dataset of 200 Java repositories with permissive licenses and near-deduplicated files that are augmented with three types of repository contexts.

Outline

On-the-Fly Adaptation of Source Code Models
(CAP Workshop, NeurIPS 2020)


On-the-Fly Adaptation of Source Code Models

Disha Shrivastava, Hugo Larochelle, Daniel Tarlow

RepoFusion: Training Code Models to Understand Your Repository
(under review at NeurIPS 2023)


RepoFusion: Training Code Models to Understand Your Repository

Disha Shrivastava, Denis Kocetkov, Harm de Vries, Dzmitry Bahdanau, Torsten Scholak

Introduction → Article 1 → Article 2 → Article 3 → Article 4 → **Final Thoughts**


Learning to Combine Per-Example Solutions for Neural Program Synthesis

Disha Shrivastava, Hugo Larochelle, Daniel Tarlow


Repository-Level Prompt Generation for Large Language Models of Code

Disha Shrivastava, Hugo Larochelle, Daniel Tarlow

Learning to Combine Per-Example Solutions for Neural Program Synthesis
(NeurIPS 2021)

Repository Level Prompt Generation for Large Language Models of Code
(ICML 2023)

Broad Applicability of Our Framework

Size of the Support Context

- Limited context can be given as input to Predict
- Combining multiple relevant contexts such as in RepoFusion
  - Determining the optimal number and size of each relevant context
- LLM with large context window
- Retrieval-augmented models that work with external memory
- Comes with increased inference costs

Broad Applicability of Our Framework

Size of the Support Context

Capturing the Dependence between Enhance and Predict

- Predict should learn to effectively leverage Z provided by Enhance
- Enhance should use the feedback signal from Predict to guide the selection of Z
- Joint training of Enhance and Predict difficult in practise.
- Separate training offers more flexibility
  - Predict: Larger LLM, trained on large data
  - Enhance: Smaller model, task-specific training on curated data.

Broad Applicability of Our Framework

Size of the Support Context

Capturing the Dependence between Enhance and Predict

Generality of the Support Context

- Automatic selection of Z conditioned on the task
- Instruction-tuned LLM as both Enhance and Predict
  - Generate relevant contextual cues when prompted with instructions capturing the task (challenging to make this work across diverse tasks)
  - Use the generated contextual cues as input to generate predictions
  - Can do these iteratively to refine the predictions.

Broad Applicability of Our Framework

Size of the Support Context

Capturing the Dependence between Enhance and Predict

Generality of the Support Context

Human-in-the-loop

- Human-interpretable contextual cues from Enhance
  - More control over what goes in the Predict stage such as prompt proposals
- Utilize human feedback to come up with better metrics and refine predictions to better align with user's preferences.

Broad Applicability of Our Framework

Size of the Support Context

Capturing the Dependence between Enhance and Predict

Generality of the Support Context

Human-in-the-loop

Performance-Latency Tradeoff

Optimizing resource allocation between Enhance and Predict (especially during inference) to match specific time and computational requirements.

Going Forward

Modeling the Code Ecosystem

Derive contextual cues from the complex programming workflow

- Iterative and dynamic aspect
  - Different program stages: *writing -> testing-> committing -> maintaining*
  - Codebases keep evolving
- Interaction with tools
  - Compiler
  - Static Analyzer
  - GitHub
  - Web, e.g. StackOverflow
- Interaction with other developers
  - Code reviewers
  - Collaborators

Going Forward

Modeling the Code Ecosystem

Derive contextual cues from the complex programming workflow

- Iterative and dynamic aspect
  - Different program stages: *writing -> testing-> committing -> maintaining*
  - Codebases keep evolving
- Interaction with tools
  - Compiler
  - Static Analyzer
  - GitHub
  - Web, e.g. StackOverflow
- Interaction with other developers
  - Code reviewers
  - Collaborators

Modeling the User

Inform the selection of contextual cues and predictions based on user interactions

- Metrics based on user preferences
  - Acceptance rate
  - User edits
- Mode of user interaction [1]
  - *Accelerated:* fixed contextual cues, single, short predictions
  - *Exploratory:* diverse contextual cues, several, long predictions
- Changing user beliefs [2]
  - Dynamically adapt the model
  - Align more with user values: *agency, creativity, trust, verifiability*

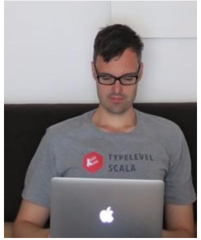
[1] "Grounded Copilot: How Programmers Interact with Code-Generating Models". Barke et al. (2022)
[2] "Approach Intelligent Writing Assistants Usability with Seven Stages of Action". Bhat et al. (2023)

Thank You

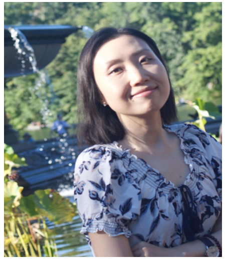


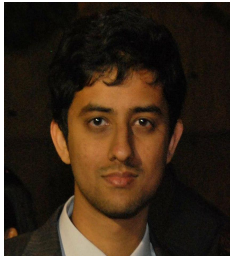








Questions/ Comments