CoditT5: Pretraining for Source Code and Natural Language Editing

Jiyang Zhang, Sheena Panthaplackel, Pengyu Nie, Junyi Jessy Li, Milos Gligoric

TEXAS
The University of Texas at Austin

ASE 2022

Supported by NSF
CCF-1652517
CCF-2107291
IIS-2107524
IIS-2145479

Pretraining + Code

- Language models pretrained on code and natural language
  - PLBART[1], CodeT5[2], Copilot[3]
- Impressive performance on generation tasks
- Does not capture the editing nature of software development
  - not designed for making edits
  - frequently copy the inputs without making edits (34.25% of the times)
  - make irrelevant edits

[1] Wasi A., Saikat C., Baishakhi R., and Kai-Wei C.. 2021. Unified Pre-training for Program Understanding and Generation. In Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2655–2668.
[2] Yue W., Weishi W., Shafiq J., and Steven C.H. H.. 2021. CodeT5: Identifier-aware Unified Pre-trained Encoder-Decoder Models for Code Understanding and Generation. In Empirical Methods in Natural Language Processing. 8696–8708.
[3] https://github.com/features/copilot

CoditT5: Pretrained Model for Edits

- Pretrained objective that explicitly models edits
- CoditT5: designed for software-editing tasks

```java
public Integer getMinElement(List myList) {
    if (myList.size() >= 0) {
        return ListManager.getFirst(myList);
    }
    return 0;
}
```

edit →

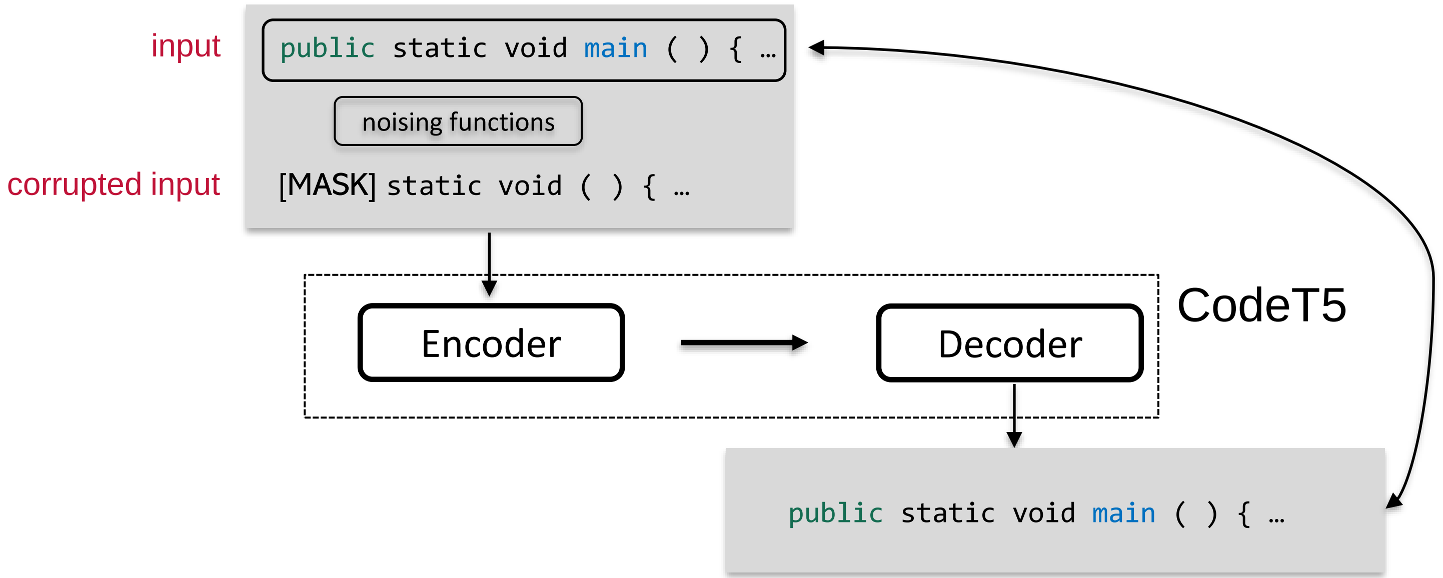
```diff
public Integer getMinElement(List myList) {
    if (myList.size() >= 0) {
-           return ListManager.getFirst(myList);
+           return ListManager.min(myList);
    }
-   return 0;
+   return null;
}
```

Our Contributions

- Propose a novel pretraining objective for editing tasks
- Build a large pretrained language model: CoditT5
- Evaluate on three downstream tasks
  - Comment updating
  - Bug fixing
  - Automated code review
- Combine CoditT5 with a standard generation model

Existing pretraining objective

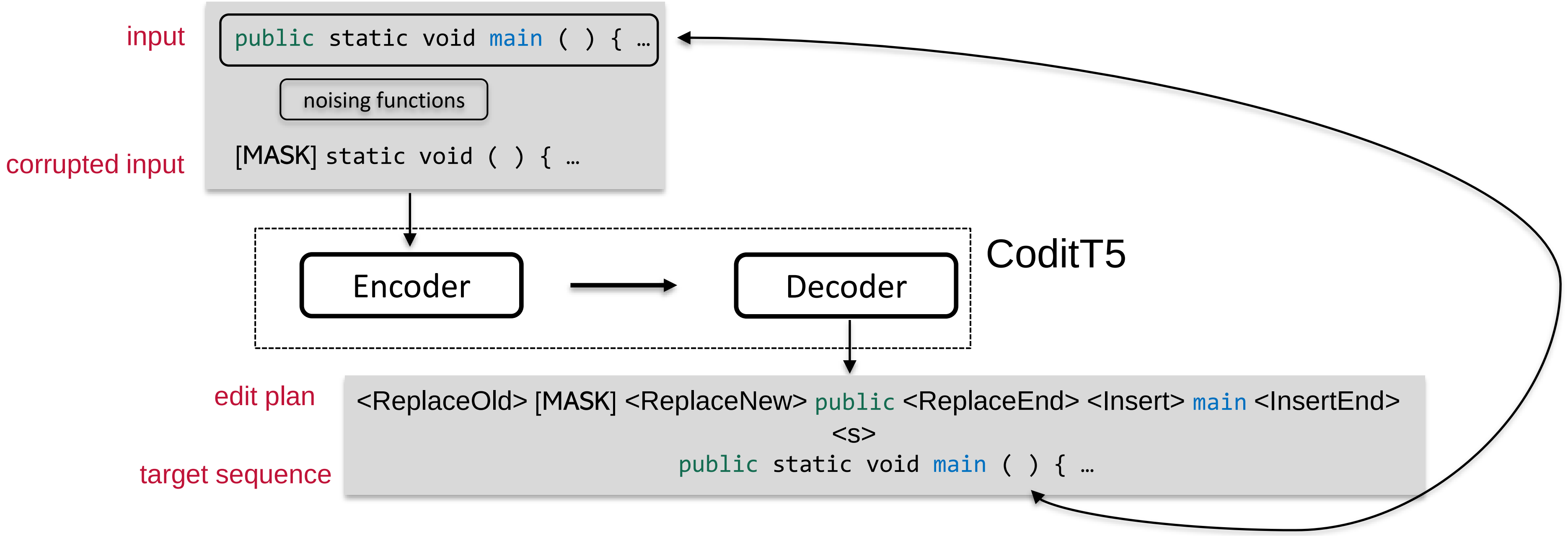
Denoising autoencoding



input
public static void main ( ) { …
noising functions
corrupted input
[MASK] static void ( ) { …
Encoder
Decoder
CodeT5
public static void main ( ) { …

CoditT5

- noising functions
- edit plan
- target sequence


input
public static void main ( ) { …
noising functions
corrupted input
[MASK] static void ( ) { …
Encoder
Decoder
CoditT5
edit plan
<ReplaceOld> [MASK] <ReplaceNew> public <ReplaceEnd> <Insert> main <InsertEnd>
<s>
target sequence
public static void main ( ) { …

Noising Functions

- Deleting random spans of tokens in the input

```
public static void main ( ) { …
```

```
public static void ( ) { …
```

- Inserting [MASK] token at random positions

```
public static void main ( ) { …
```

```
Public [MASK] static void main ( ) { …
```

- Randomly masking spans with the special [MASK] token

```
public static void main ( ) { …
```

```
[MASK] static void main ( ) { …
```

Noising Functions Cont’d

- Collect statistics from real-world edits:

	Code	Natural Language
Prob. Delete	0.50	0.07
Prob. Insert	0.21	0.11
Prob. Replace	0.30	0.82
Avg. # Tokens	6.50	3.00
Avg. # Spans	1.90	1.40

Edit Plan

Concrete edit actions to be applied to the input

- Example:

corrupted input	[MASK] `static void ( ) { …`
edit plan	<ReplaceOld> [MASK] <ReplaceNew> `public` <ReplaceEnd> <Insert> `main` <InsertEnd>
Input	`public static void main ( ) { …`

Edit Plan Formats

- Delete

  <Delete> [span of tokens] <DeleteEnd>

- Insert

  <Insert> [span of tokens] <InsertEnd>

- Replace

  <ReplaceOld> [span of old tokens]

  <ReplaceNew> [span of new tokens] <ReplaceEnd>

Target Sequence

The final edited sequence after applying the edit plan

- Why?
  - can not always derive the target sequence deterministically
  - maintaining fluency and coherency of the edited sequence

corrupted input	[MASK] `static void ( ) {` …
edit plan	<ReplaceOld> [MASK] <ReplaceNew> `public` <ReplaceEnd> <Insert> `main` <InsertEnd>

Pretraining Data

- CodeSearchNet[1]:
  - 6 programming languages: Python, Java, Ruby, Php, Go, JavaScript
  - natural language comments
- 5.9M methods and 1.6M comments after preprocessing

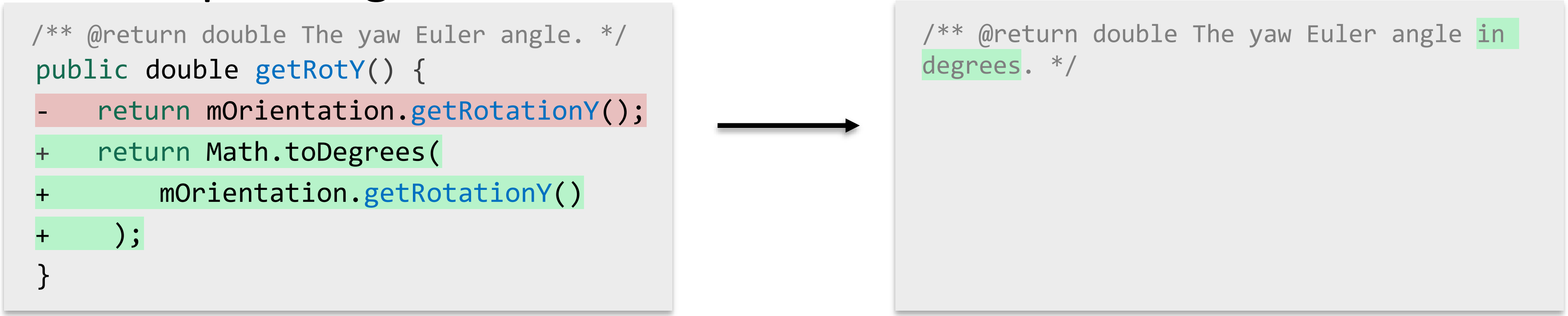
[1] Hamel H., Ho-Hsiang W., Tiferet G., Miltiadis A., and Marc B.. 2019. Codesearchnet challenge: Evaluating the state of semantic code search. arXiv preprint arXiv:1909.09436 (2019)

Downstream Tasks

- Comment Updating
- Bug Fixing
- Automated Code Review

Comment Updating

- Updating a natural language comment to reflect changes in the corresponding body of code
- Dataset[1]: Java method changes paired with changes in the corresponding comments



```
/** @return double The yaw Euler angle. */
public double getRotY() {
-    return mOrientation.getRotationY();
+    return Math.toDegrees(
+        mOrientation.getRotationY()
+    );
}
```

$\longrightarrow$

```
/** @return double The yaw Euler angle in
degrees. */
```

[1] Panthaplackel, S., Nie, P., Gligoric, M., Li, J. J., & Mooney, R. (2020, July). Learning to Update Natural Language Comments Based on Code Changes. In Annual Meeting of the Association for Computational Linguistics.1853–1868
[2] Lin, B., Wang, S., Liu, Z., Xia, X., & Mao, X. (2022). Predictive Comment Updating with Heuristics and AST-Path-Based Neural Learning: A Two-Phase Approach. IEEE Transactions on Software Engineering.

Baselines & Metrics: Comment Updating

- Baselines:
  - PLBART
  - CodeT5
  - RNN edit model[1]
- Metrics (from 0 to 100):
  - xMatch: pct. of the predictions exactly matches the ground truths
  - GLUE, SARI: edit actions overlap
  - BLEU, METEOR: token-level overlap

[1] Panthaplackel, S., Nie, P., Gligoric, M., Li, J. J., & Mooney, R. (2020, July). Learning to Update Natural Language Comments Based on Code Changes. In Annual Meeting of the Association for Computational Linguistics.1853–1868

Eval: Comment Updating

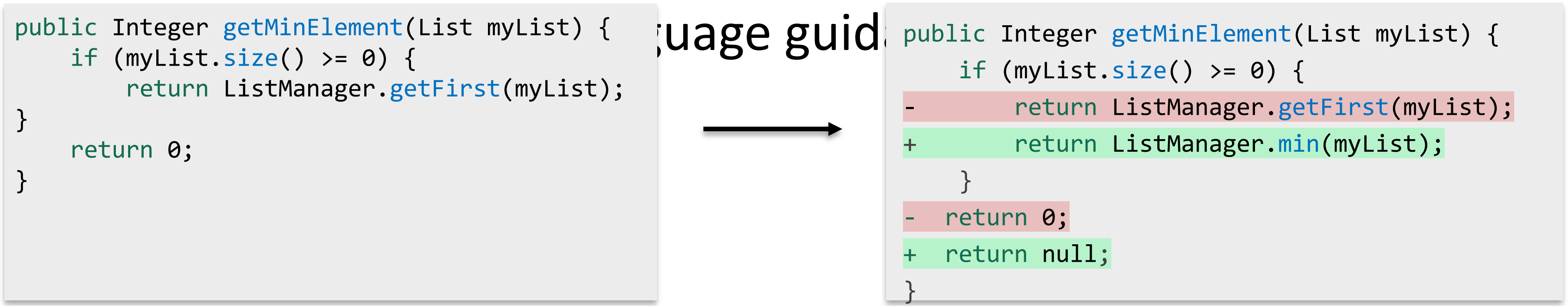
	xMatch	SARI	GLEU	BLEU	METEOR
RNN Edit Model	33.33	56.23	51.88	56.55	52.26
PLBART	35.33	52.83	54.75	62.04	56.79
CodeT5	38.00	58.80	58.84	**65.20**	59.63
CoditT5	**43.33**	**61.41**	**59.53**	64.56	**60.75**

CoditT5 achieves higher performance for most of the metrics, highlighting the benefit of explicitly modeling edits for the editing tasks

Bug Fixing

- Generating a fixed code snippet given buggy code snippet
- Dataset[1]: Java BugFixPairs-Small ($B2F_S$) and BugFixPairs-Medium ($B2F_m$) datasets with buggy code snippet, fixed code

uage guid



```java
public Integer getMinElement(List myList) {
    if (myList.size() >= 0) {
        return ListManager.getFirst(myList);
}
    return 0;
}
```

```diff
public Integer getMinElement(List myList) {
    if (myList.size() >= 0) {
-       return ListManager.getFirst(myList);
+       return ListManager.min(myList);
    }
-  return 0;
+  return null;
}
```

[1] Chakraborty, S., & Ray, B. 2021. On Multi-Modal Learning of Editing Source Code. In Automated Software Engineering. 443–455
[2] Dawn D., Chen W., Alexey S., and Neel S. 2021. Generating bug-fixes using pretrained transformers. In nternational Symposium on Machine Programming. 1–8
[3] Antonio M., Simone S., Nathan C., David N. P., Denys P., Rocco O., and Gabriele B.. 2021. Studying the usage of text-to-text transfer transformer to support code-related tasks. In International Conference on Software Engineering. 336–347.

Baselines & Metrics: Bug Fixing

- Baselines:
  - MODIT[1] (PLBART)
  - CodeT5
- Metrics:
  - xMatch: pct. of the predictions exactly matches the ground truths

[1] Chakraborty, S., & Ray, B. 2021. On Multi-Modal Learning of Editing Source Code. In Automated Software Engineering. 443–455

Eval: Bug Fixing

	xMatch	
	B2F$_s$	**B2F$_m$**
PLBART	31.09	24.18
CodeT5	34.81	26.66
CoditT5	**37.52**	**29.96**

CoditT5 is better than baselines on xMatch

Automated Code Review

- Generating the revised code snippet, given a code snippet under review and a brief natural language sentence prescribing code edits
- Dataset[1]: Java methods (before and after the review) paired with pu

```
"Generally better to qualify than making static import"
public List<Pattern> getExcludedResponseHeaderPatterns() {
-    return emptyList();
+    return List.emptyList();
}
```

[1] Rosalia T., Luca P., Michele T., Denys P., and Gabriele B. 2021. Towards Automating Code Review Activities. In International Conference on Software Engineering. 163–174
[2] Zhiyu L., Shuai L, Daya G., Nan D., Shailesh J., Grant J., Deep M., Jared G., Alexey S., Shengyu F., et al. 2022. CodeReviewer: Pre-Training for Automating Code Review Activities. arXiv preprint arXiv:2203.09095 (2022).
[3] Rosalia T., Simone M., Antonio M., Luca P., Denys P., and Gabriele B.. 2022. Using Pre-Trained Models to Boost Code Review Automation. In International Conference on Software Engineering. 2291–2302

Baselines & Metrics: Automated Code Review

- Baselines
  - PLBART
  - CodeT5
- Metrics:
  - xMatch: pct. of the predictions exactly matches the ground truths
  - BLEU: token-level overlap

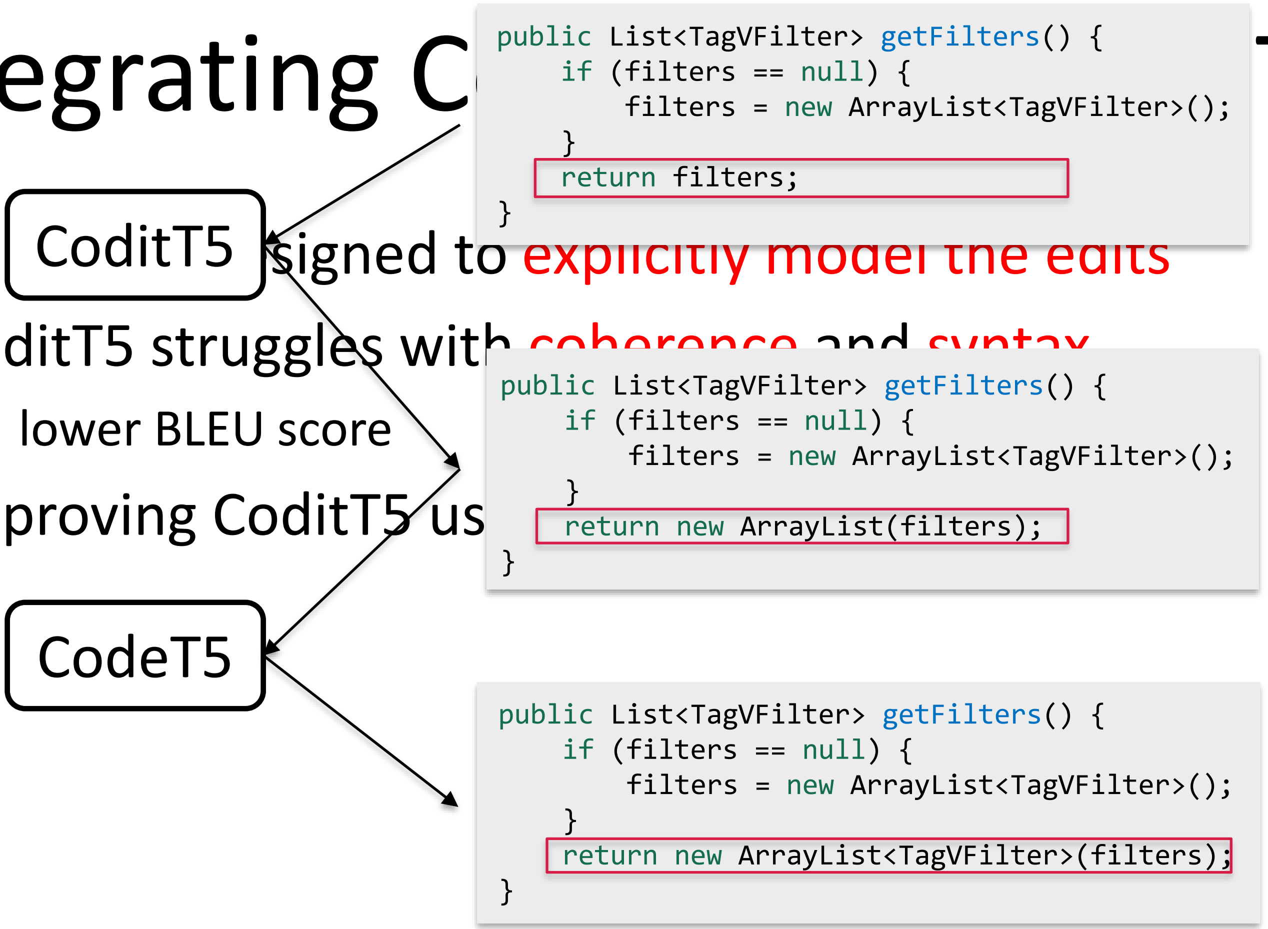
Eval: Automated Code Review

	xMatch	**BLEU**
PLBART	26.78	79.38
CodeT5	34.98	**83.20**
CoditT5	**37.19**	80.50

CoditT5 has better performance on xMatch

Integrating C T5

- Co signed to explicitly model the edits
- CoditT5 struggles with coherence and syntax
  - lower BLEU score
- Improving CoditT5 us



```
public List<TagVFilter> getFilters() {
    if (filters == null) {
        filters = new ArrayList<TagVFilter>();
    }
    return filters;
}
```

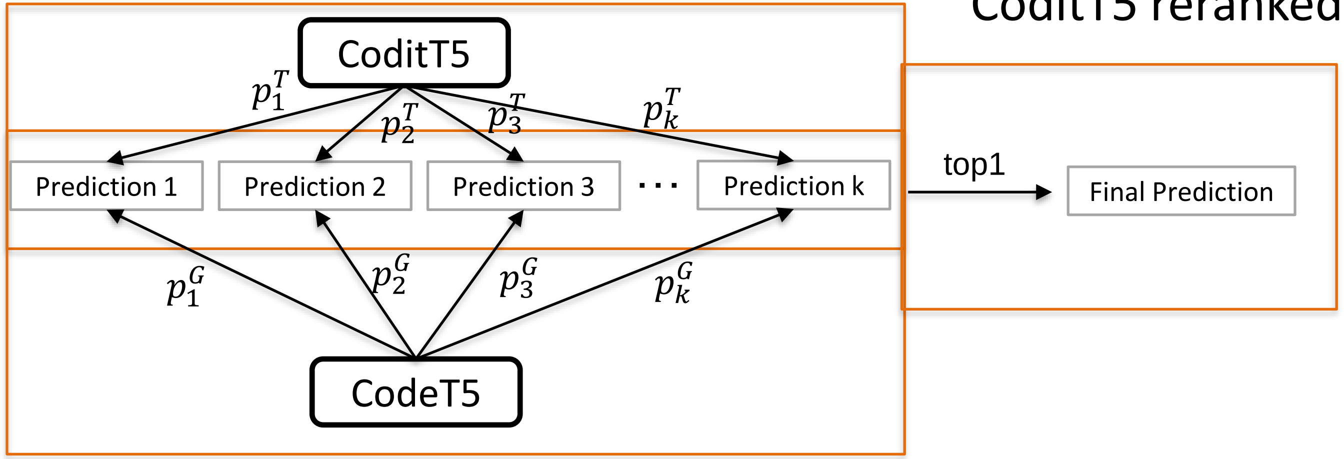
```
public List<TagVFilter> getFilters() {
    if (filters == null) {
        filters = new ArrayList<TagVFilter>();
    }
    return new ArrayList(filters);
}
```

```
public List<TagVFilter> getFilters() {
    if (filters == null) {
        filters = new ArrayList<TagVFilter>();
    }
    return new ArrayList<TagVFilter>(filters);
}
```

Integrating CoditT5 and CodeT5

- Combine two models using simple likelihood-based reranking strategies at test time

CoditT5 reranked w/ CodeT5



Eval: Combination

	Comment Update		$B2F_s$	$B2F_m$	Code Review	
	xMatch	BLEU	xMatch		xMatch	BLEU
CodeT5	38.00	65.20	34.81	26.66	34.98	83.20
CoditT5	43.33	64.56	37.52	29.96	37.19	80.50
CoditT5 reranked w/ CodeT5	**45.33**	**66.80**	**40.22**	32.06	40.98	**84.12**
CodeT5 reranked w/ CoditT5	44.00	65.58	39.56	**32.24**	**43.42**	83.92

Summary

- Novel pretraining objective that explicitly models edits
- CoditT5: a large pretrained model for software editing tasks
- Combining our edit-based model with a standard generation model through simple reranking strategies
- Evaluate on three downstream tasks

https://github.com/EngineeringSoftware/CoditT5

Jiyang Zhang <jiyang.zhang@utexas.edu>