Repoformer: Selective Retrieval for Repository-Level Code Completion

Di Wu[1], Wasi Uddin Ahmad[2], Dejiao Zhang[2],
Murali Krishna Ramanathan[2], Xiaofei Ma[2]

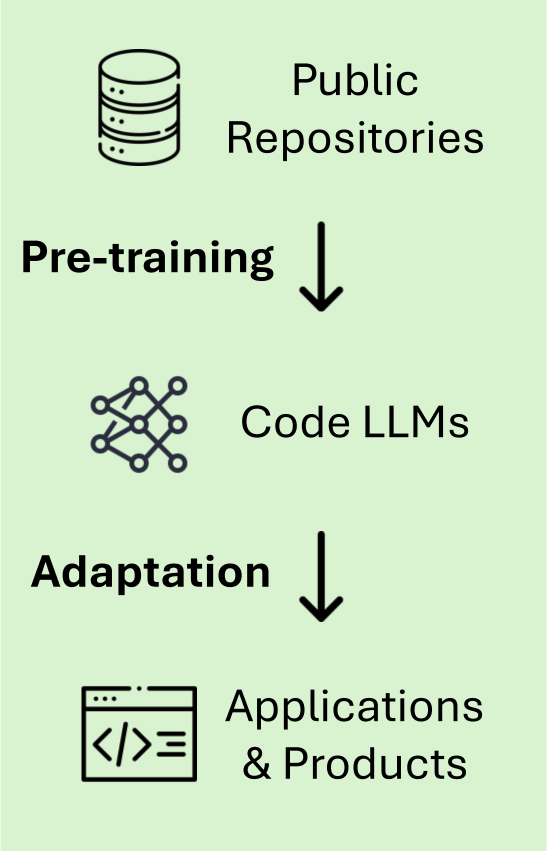
[1]University of California Los Angeles, [2]AWS AI Labs

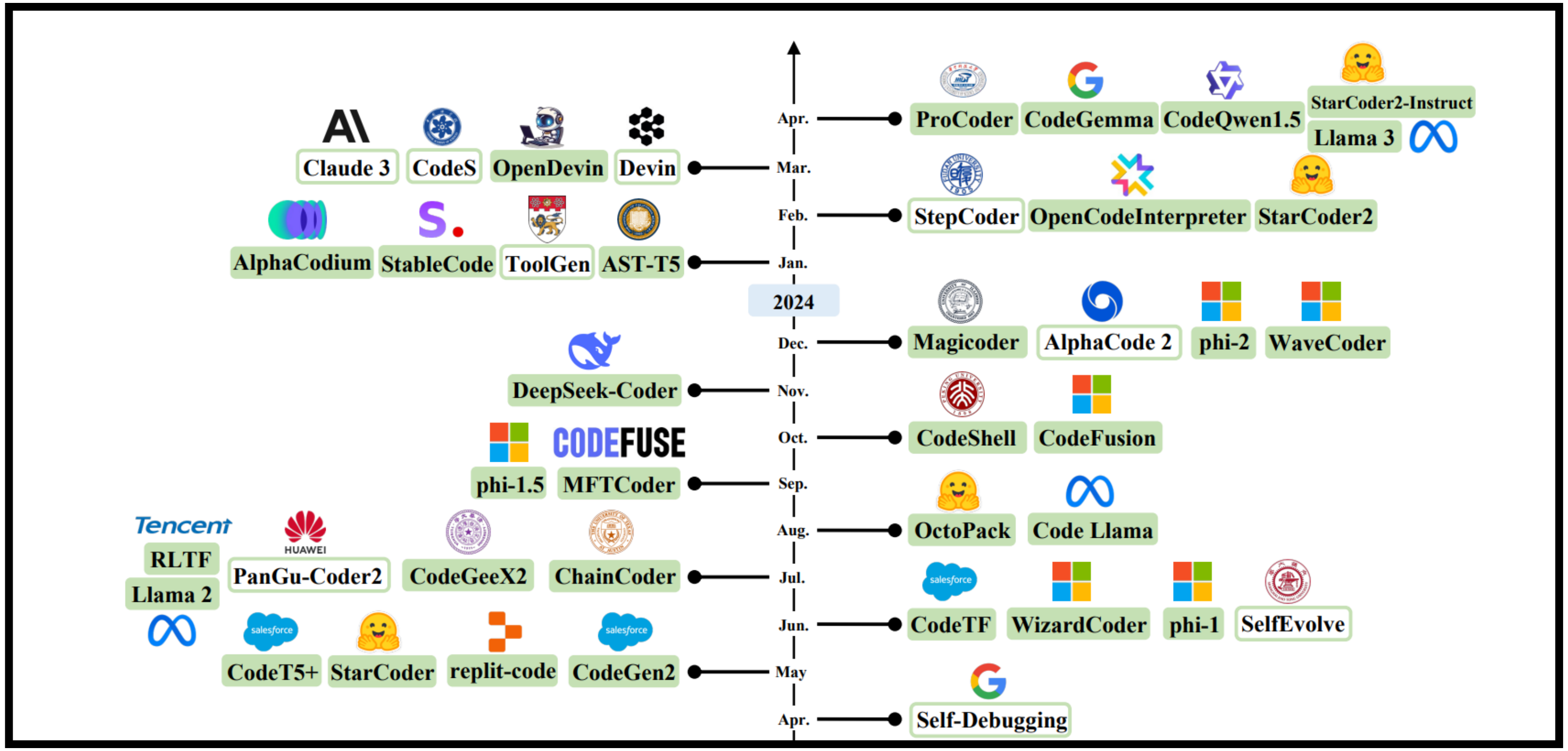




LLMs for Code

- Large language models (LLMs) have been seen as promising solutions to code automation.





[1] A Survey on Large Language Models for Code Generation. Jiang et al. 2024

The Knowledge Gap

- However, applying LLMs in ***private repositories*** is challenging.

Public Repositories | Pre-training → | Code LLMs | Application → | Private User Repositories?

Open-Domain Knowledge

- **Programming Languages**
- **Logic; Problem Solving**
- **Design Patterns**
- **Reading/Writing Documentation**

Knowledge Gap!

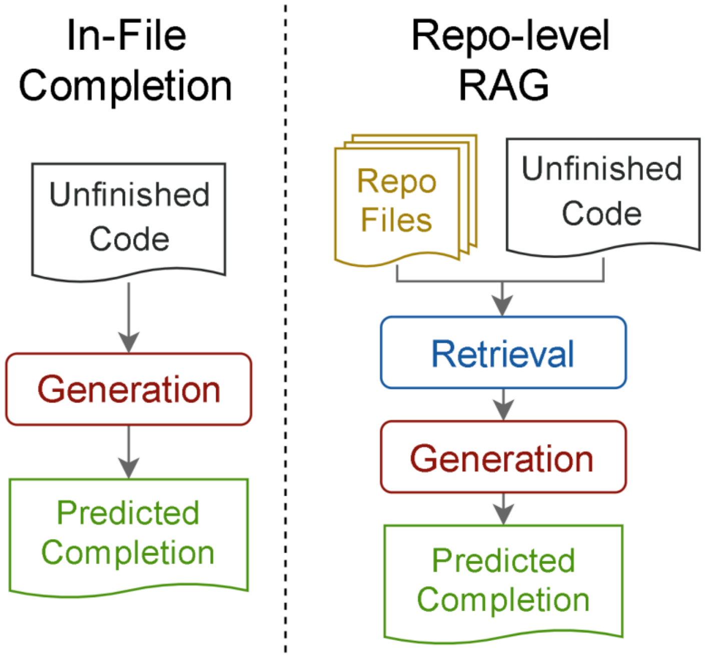
↔

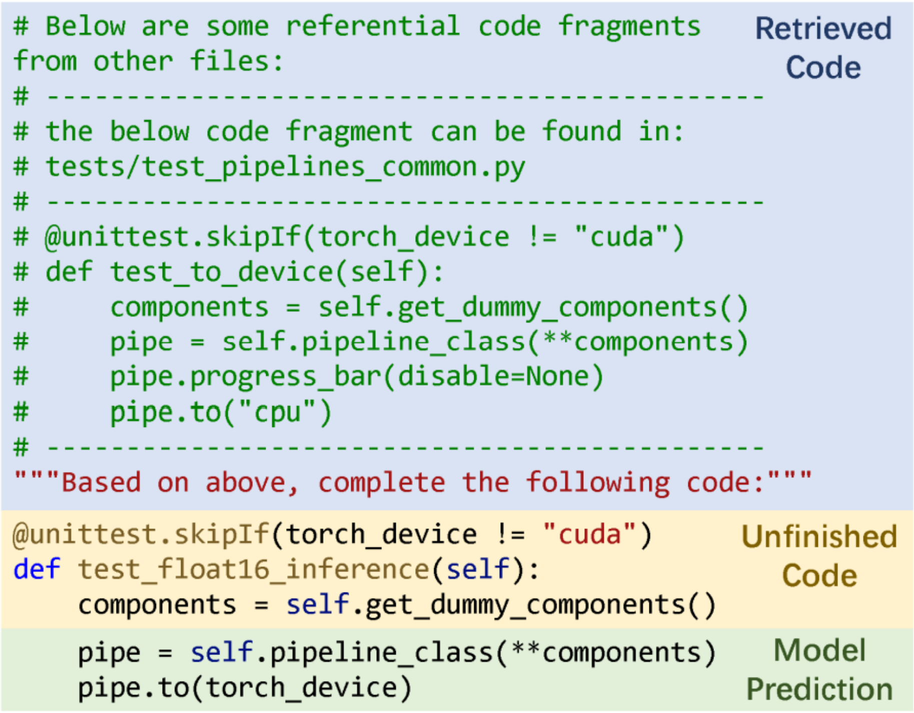
Private Knowledge

- **Private APIs**
- **Cross-Module Dependencies**
- **Development Conventions**

Retrieval-Augmented Generation

- By augmenting LLMs with retrieved repository contexts, RAG improves code completion performance.





[2] RepoCoder: Repository-Level Code Completion Through Iterative Retrieval and Generation

Retrieval-Augmented Generation

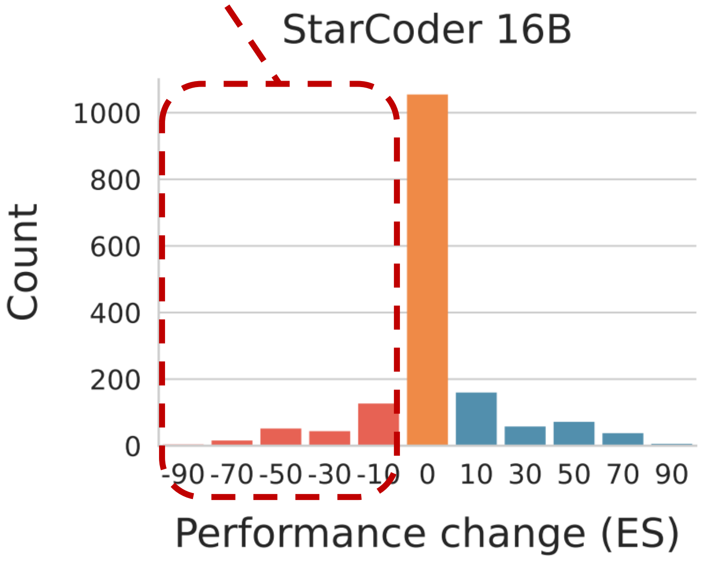
- By augmenting LLMs with retrieved repository contexts, RAG improves code completion performance.

However, should we always perform retrieval?

[2] RepoCoder: Repository-Level Code Completion Through Iterative Retrieval and Generation

Should we always perform retrieval?

- Surprisingly, always augmenting repository-level contexts is both **harmful** to accuracy and **inefficient**, especially for black-box LLMs.

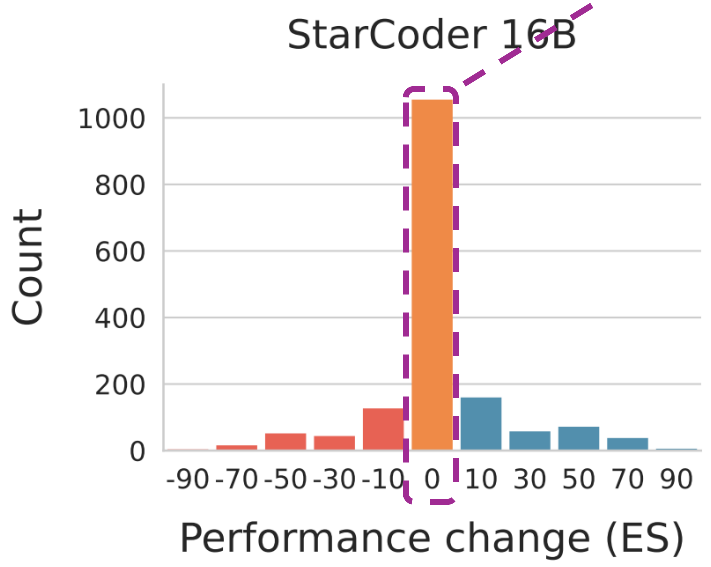


Model	Size	Performance (UT)		UT Change		
		$X_l + X_r$	$X_l + X_r + CC$	$\downarrow$	$=$	$\uparrow$
CodeGen-Mono	16B	23.74	24.18	23	407	25
CodeGen-Mono	2B	30.55	32.51	18	400	37
StarCoder	16B	34.73	42.86	16	386	53
StarCoderBase	1B	22.20	25.71	16	407	32

Table 1. The performance change on RepoEval function completion exhibited by four models from retrieved cross-file contexts.

Should we always perform retrieval?

- Surprisingly, always augmenting repository-level contexts is both **harmful** to accuracy and **inefficient**, especially for black-box LLMs.

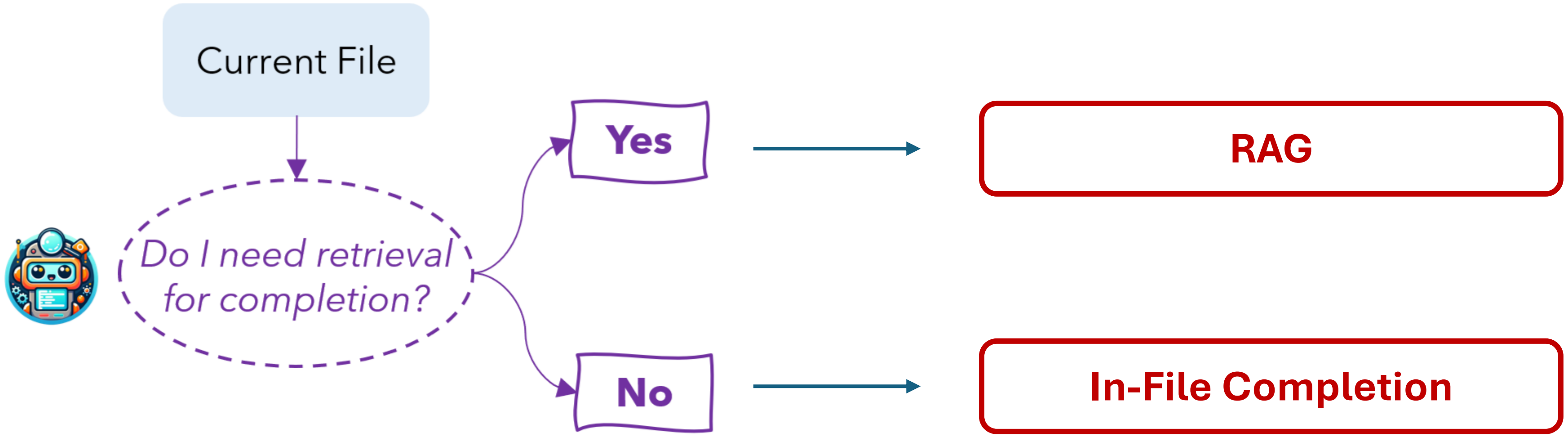


Model	Size	Performance (UT)		UT Change		
		$X_l + X_r$	$X_l + X_r + CC$	$\downarrow$	$=$	$\uparrow$
CodeGen-Mono	16B	23.74	24.18	23	407	25
CodeGen-Mono	2B	30.55	32.51	18	400	37
StarCoder	16B	34.73	42.86	16	386	53
StarCoderBase	1B	22.20	25.71	16	407	32

Table 1. The performance change on RepoEval function completion exhibited by four models from retrieved cross-file contexts.

Solution: Selective RAG

- We propose to **selectively** trigger repository-level retrieval.
- Specifically, our proposal takes the form of **self-assessment**.



Selective RAG Inference

File to Complete

```
import pandas as pd
class TableManager:
    def __init__(self, data)
        self.data = pd.DataFrame(data)
    ...
    def normalize_col(self, col):
        """Normalize the values in col
        to the range [0, 1]."""
```

Selective RAG Inference

No cross-file dependency

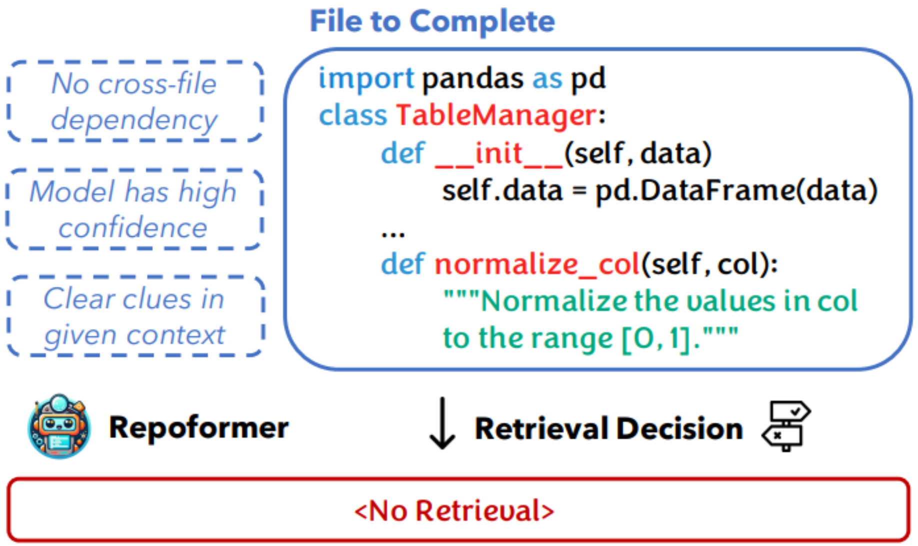
Model has high confidence

Clear clues in given context

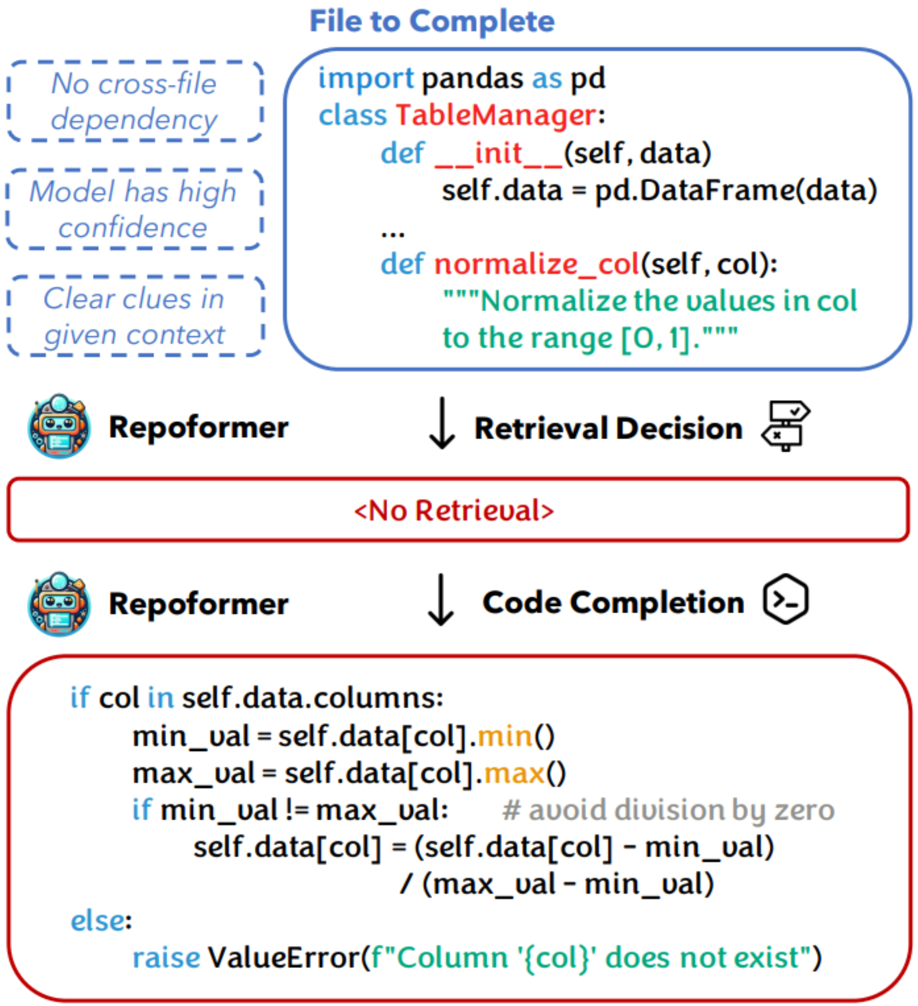
File to Complete

```
import pandas as pd
class TableManager:
    def __init__(self, data)
        self.data = pd.DataFrame(data)
    ...
    def normalize_col(self, col):
        """Normalize the values in col
        to the range [0, 1]."""
```

Selective RAG Inference



Selective RAG Inference



Selective RAG Inference

File to Complete

No cross-file dependency

Model has high confidence

Clear clues in given context

```python
import pandas as pd
class TableManager:
    def __init__(self, data)
        self.data = pd.DataFrame(data)
    ...
    def normalize_col(self, col):
        """Normalize the values in col
        to the range [0, 1]."""
```

Repoformer ↓ **Retrieval Decision**

<No Retrieval>

Repoformer ↓ **Code Completion**

```python
if col in self.data.columns:
    min_val = self.data[col].min()
    max_val = self.data[col].max()
    if min_val != max_val:      # avoid division by zero
        self.data[col] = (self.data[col] - min_val)
                         / (max_val - min_val)
else:
    raise ValueError(f"Column '{col}' does not exist")
```

File to Complete

Local imports

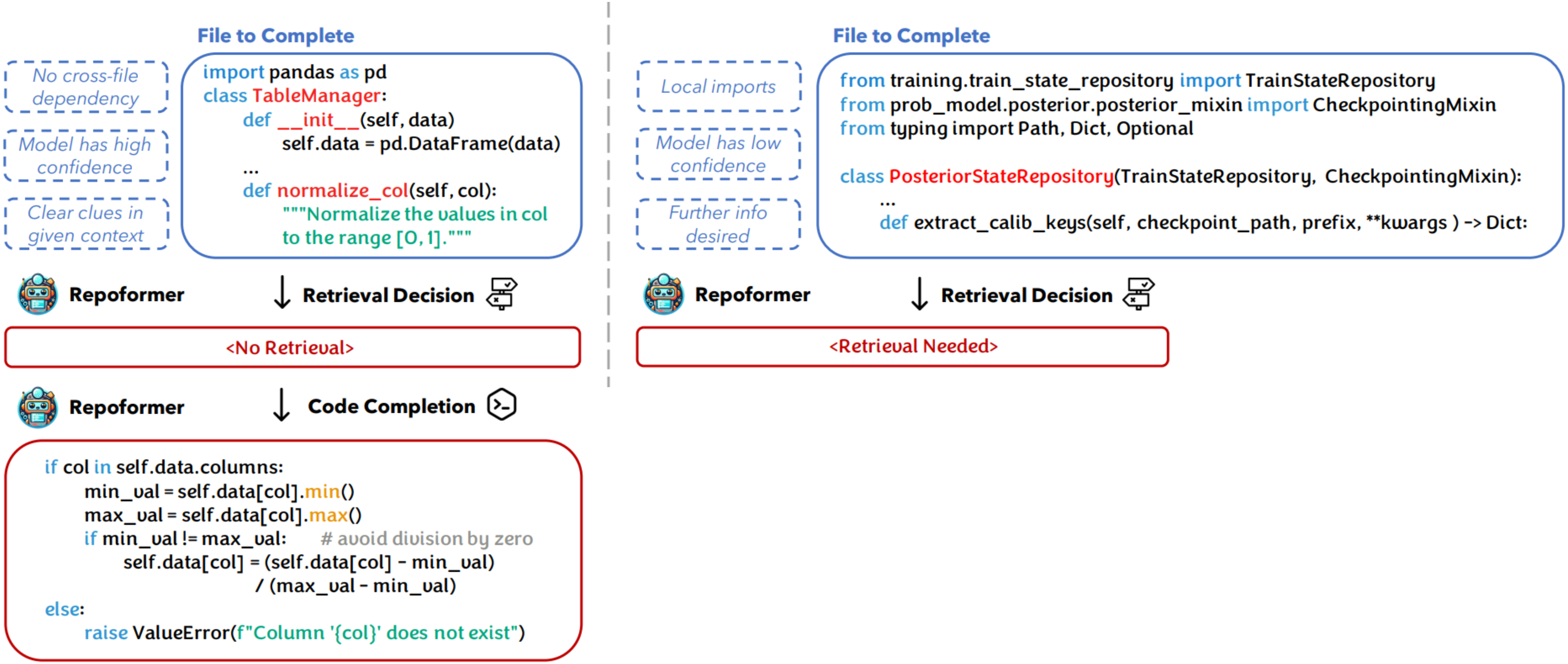
Model has low confidence

Further info desired

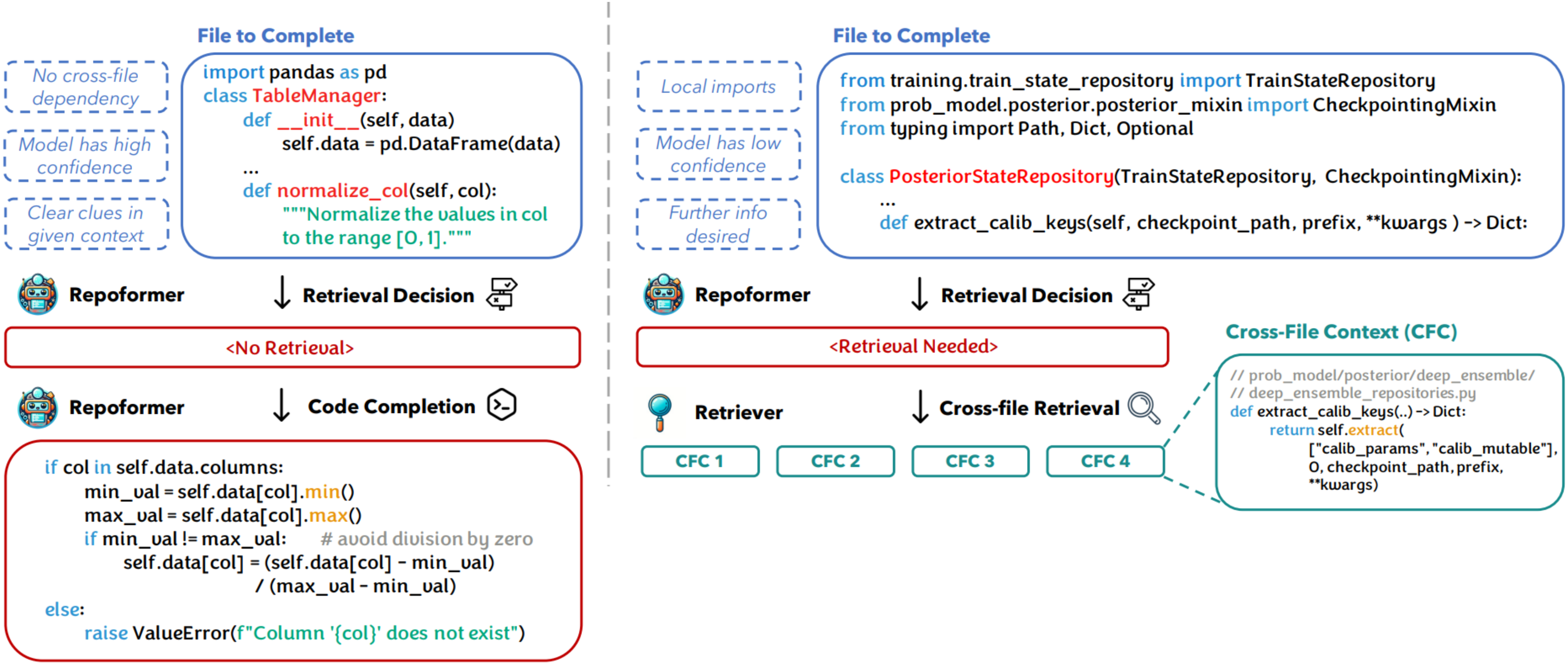
```python
from training.train_state_repository import TrainStateRepository
from prob_model.posterior.posterior_mixin import CheckpointingMixin
from typing import Path, Dict, Optional

class PosteriorStateRepository(TrainStateRepository, CheckpointingMixin):
    ...
    def extract_calib_keys(self, checkpoint_path, prefix, **kwargs ) -> Dict:
```

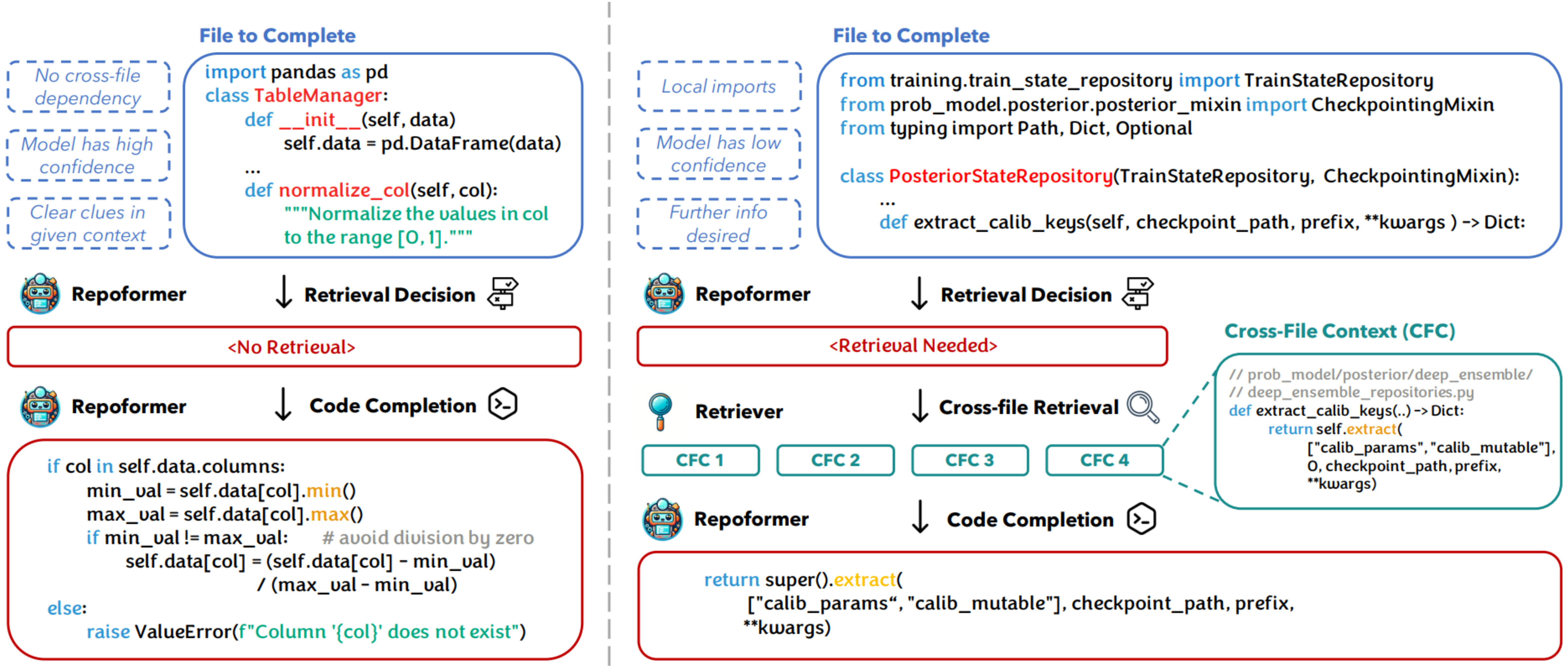
Selective RAG Inference



Selective RAG Inference



Selective RAG Inference



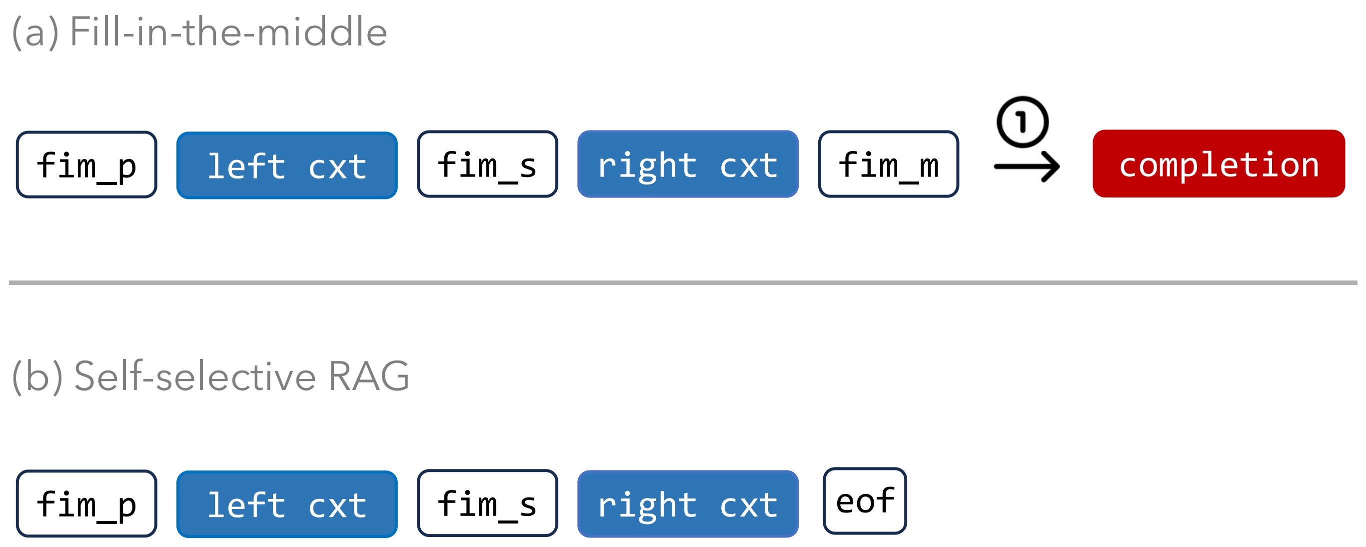
Selective RAG Inference

- Conveniently modeled as an extension to fill-in-the-middle.

(a) Fill-in-the-middle

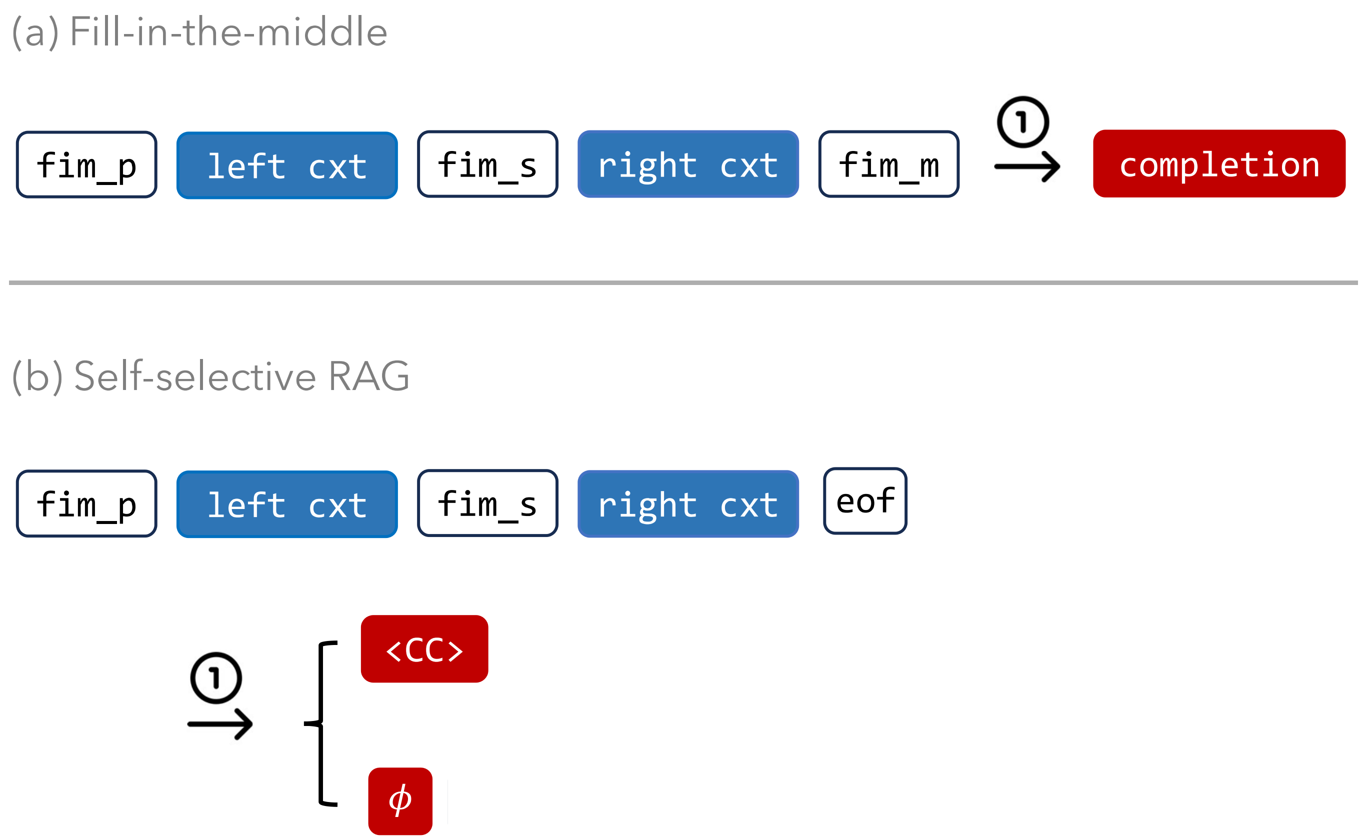
Selective RAG Inference

- Conveniently modeled as an extension to fill-in-the-middle.



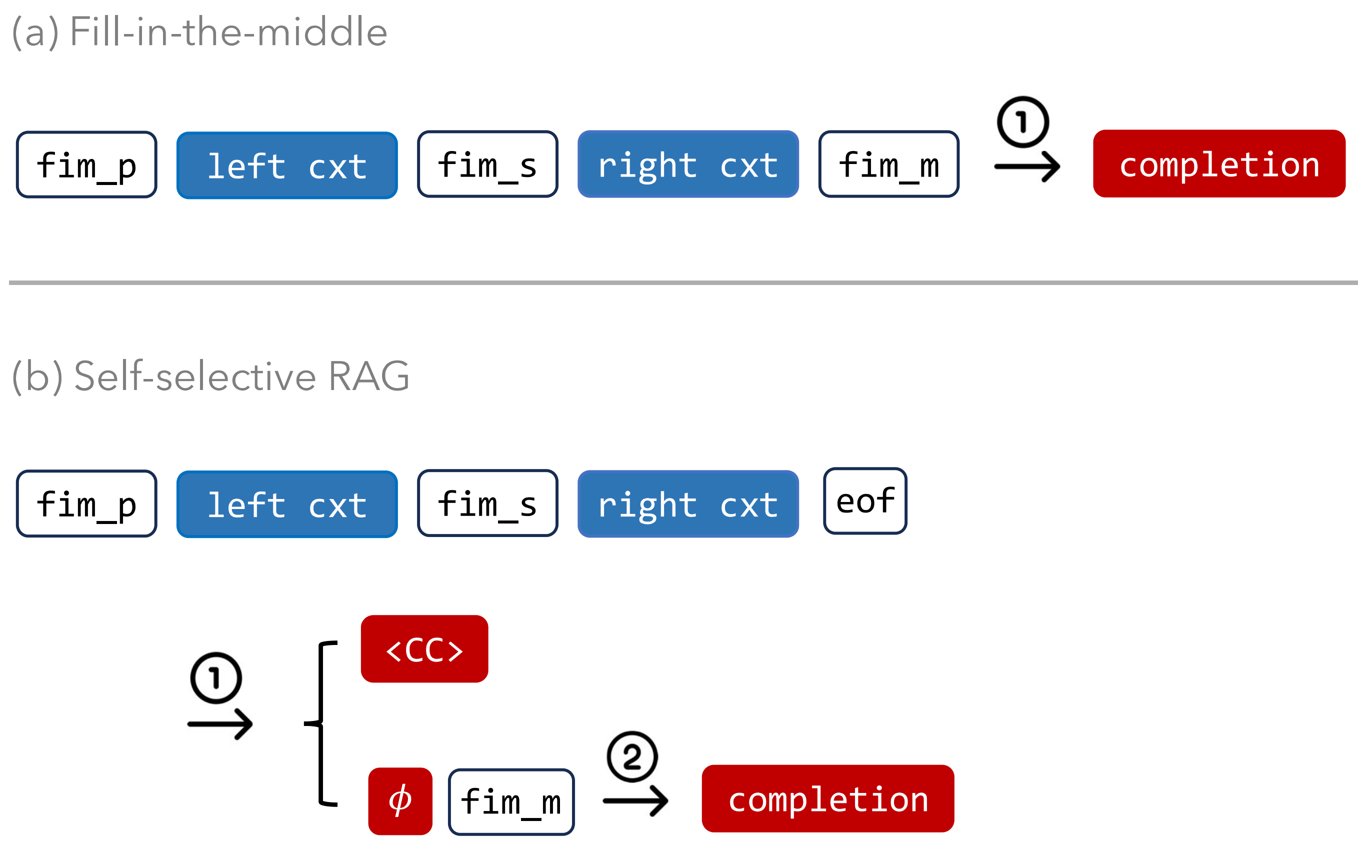
Selective RAG Inference

- Conveniently modeled as an extension to fill-in-the-middle.



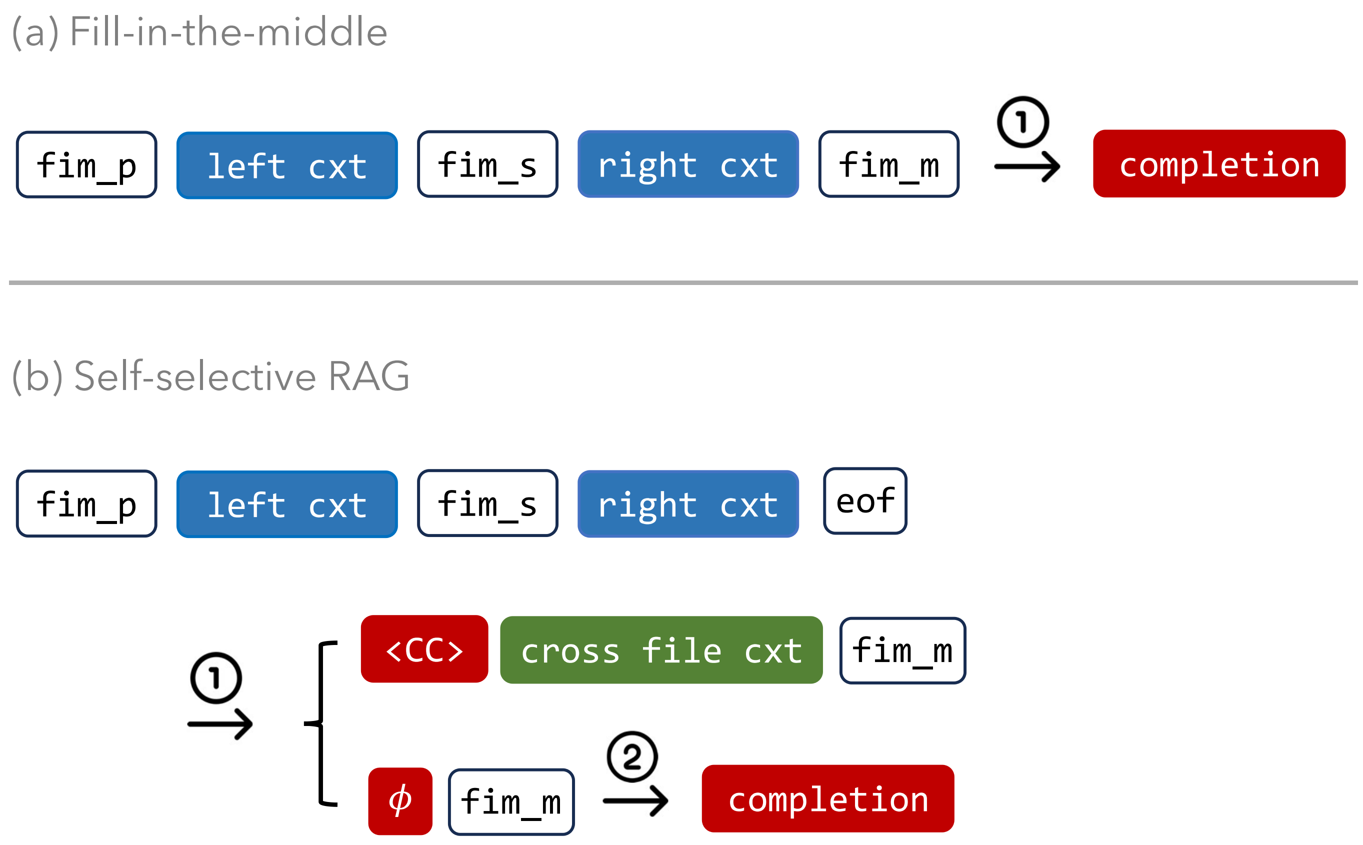
Selective RAG Inference

- Conveniently modeled as an extension to fill-in-the-middle.



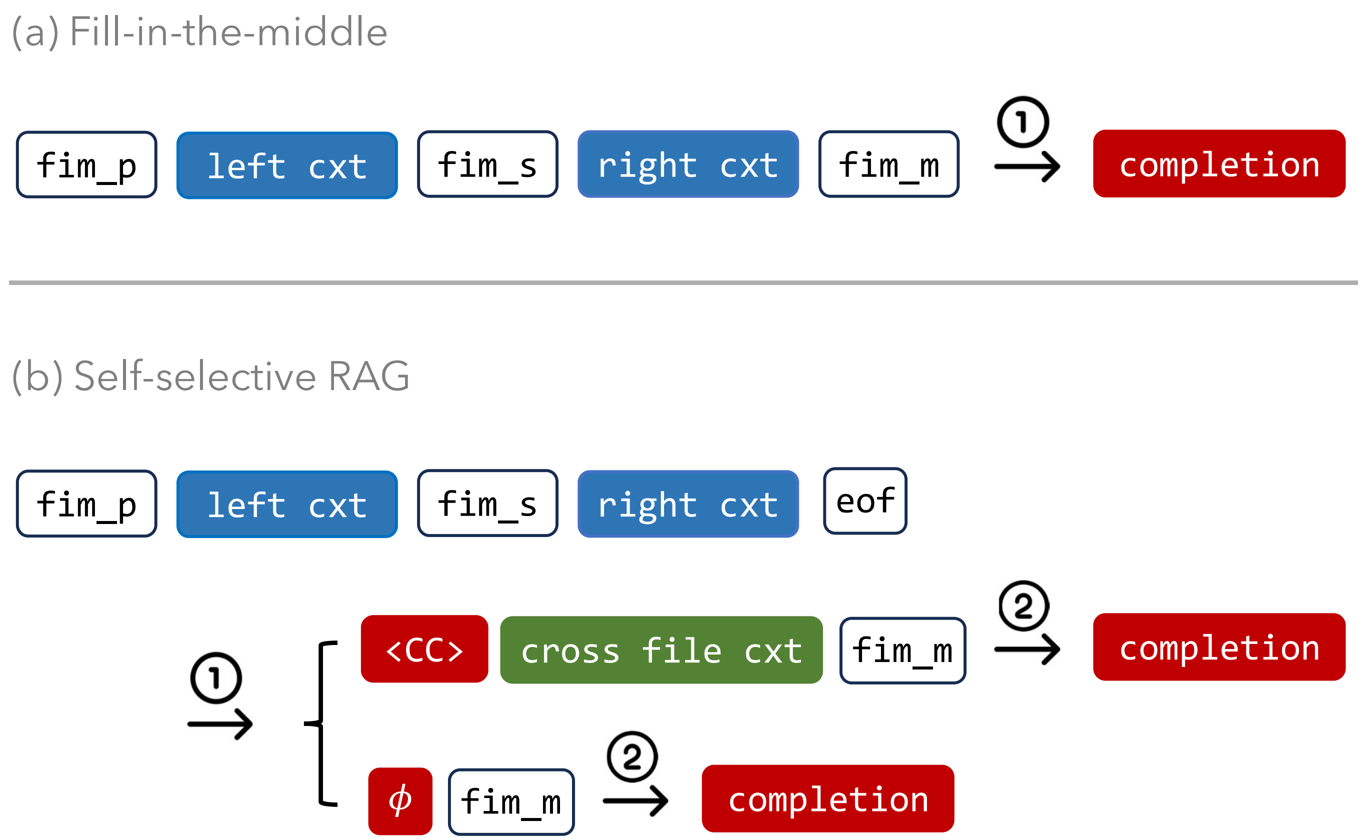
Selective RAG Inference

- Conveniently modeled as an extension to fill-in-the-middle.



Selective RAG Inference

- Conveniently modeled as an extension to fill-in-the-middle.



Learning Selective RAG

- Desiderata: ***performance-oriented self-reflection***

Can I solve the problem better with repository knowledge?

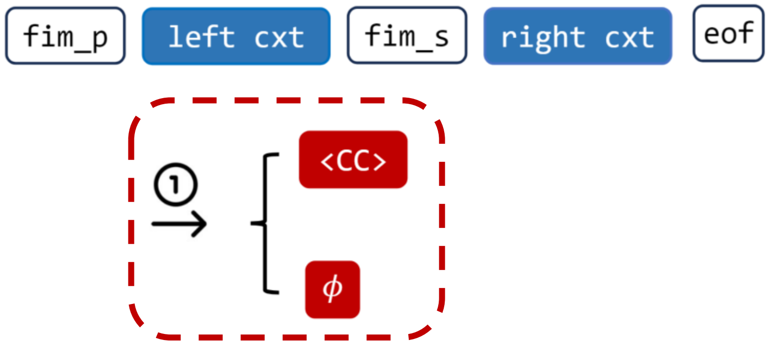
- Does the problem use **cross-file info**?
- Am I **confident** with answering?
- ...

- Insight: we could directly learn from **RAG simulation**
  - **Sample** diverse blanks for code completion.
  - Let the an LLM **attempt** with and without repository-level retrieval.
  - If the completion quality improves, **label** retrieval_required = True.

Self-Supervised Multi-Task Training

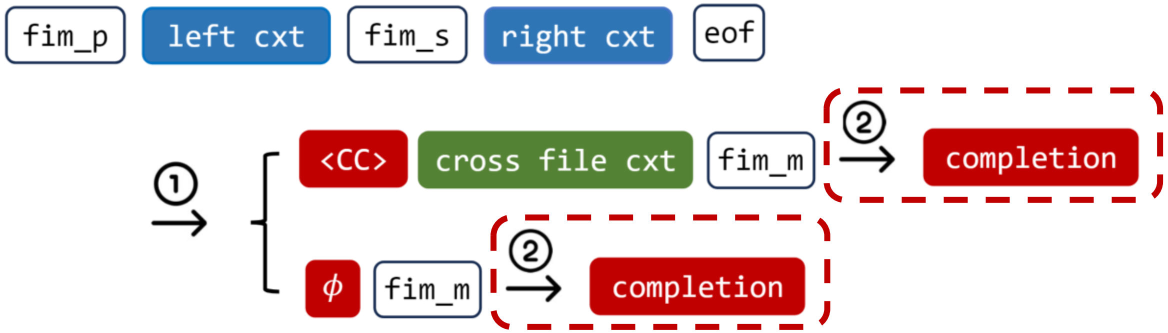
- **Self-Evaluation** for Selective Retrieval

$$\mathcal{L}_{eval} = -\log p_{\mathcal{M}}(\texttt{<cc>}|X_l, X_r)$$



- **Code Generation** with Optional Cross-File Context

$$\mathcal{L}_{gen} = \begin{cases} -\log p_{\mathcal{M}}(Y|X_l, X_r, CC), & \text{if } \textit{label} \\ -\log p_{\mathcal{M}}(Y|X_l, X_r), & \text{otherwise} \end{cases}$$



Accuracy Evaluation

- SOTA completion accuracy on **RepoEval** and **CrossCodeLongEval**, a new benchmark tailored to long-form code completion.

Size	Model	RAG Policy	RepoEval [2]			CrossCodeLongEval (Ours)	
			Line ES	API ES	Function UT	Chunk ES	Function ES
1B	StarCoderBase	No	67.77	66.54	22.20	60.09	47.49
		Always	72.30	69.17	25.71	63.73	50.50
	Repoformer	Selective_G	74.50	71.00	24.00	68.08	52.09
		Selective_T	**76.00**	**72.70**	**28.79**	**69.97**	**53.71**
3B	StarCoderBase	No	72.12	69.02	24.84	64.65	49.88
		Always	76.68	72.62	29.67	67.74	53.39
	Repoformer	Selective_G	77.60	73.60	28.57	70.70	54.47
		Selective_T	**79.02**	**74.96**	**32.96**	**72.23**	**56.24**
16B	StarCoder	No	76.07	71.00	34.73	69.40	54.20
		Always	79.24	74.50	42.86	71.90	58.06
	Repoformer	Selective_G	78.81	76.23	42.42	73.36	57.71
		Selective_T	**80.34**	**77.93**	**44.18**	**75.50**	**58.93**

Table 1: Results on RepoEval and CrossCodeLongEval.

Latency Evaluation

- Repoformer improves **both accuracy and latency** in online serving.

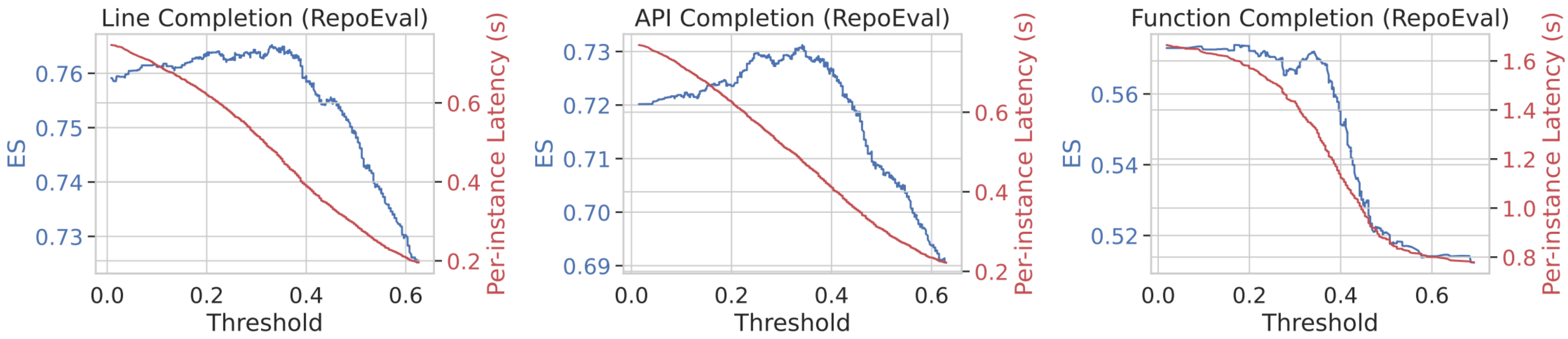


Figure 12. Latency-accuracy trade-off of self-selective RAG for REPOFORMER-1B.

Latency Evaluation

- Repoformer improves **both accuracy and latency** in online serving.

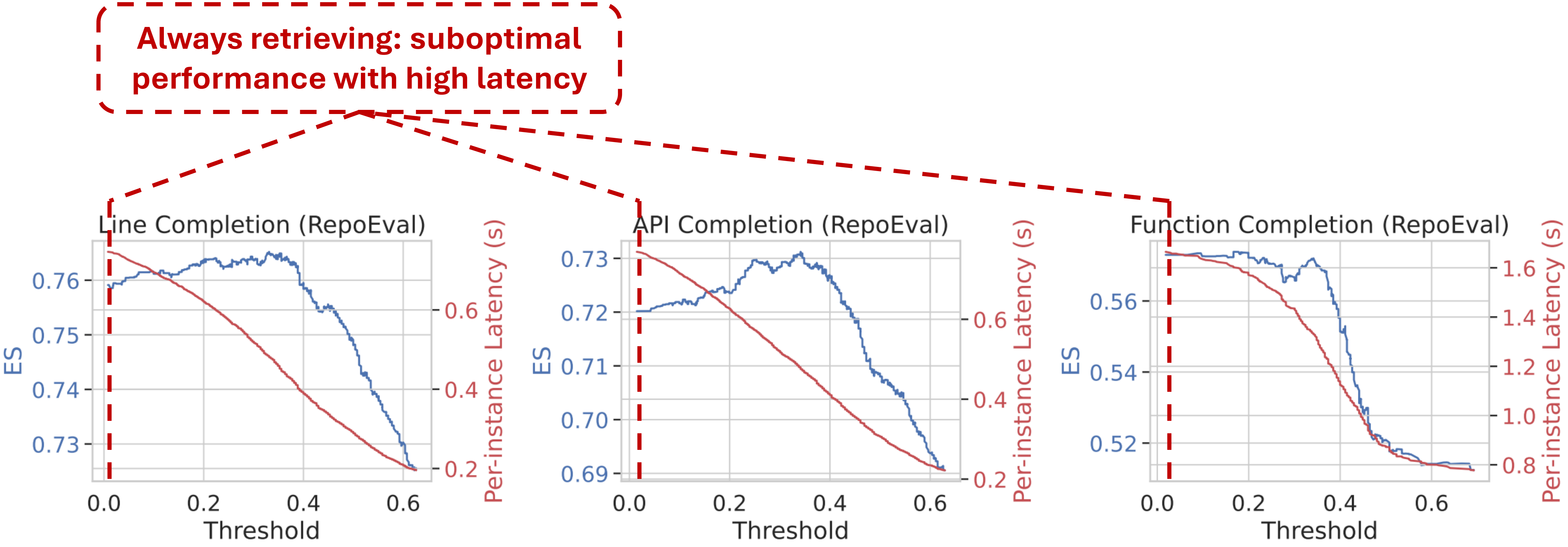


Figure 12. Latency-accuracy trade-off of self-selective RAG for REPOFORMER-1B.

Latency Evaluation

- Repoformer improves **both accuracy and latency** in online serving.

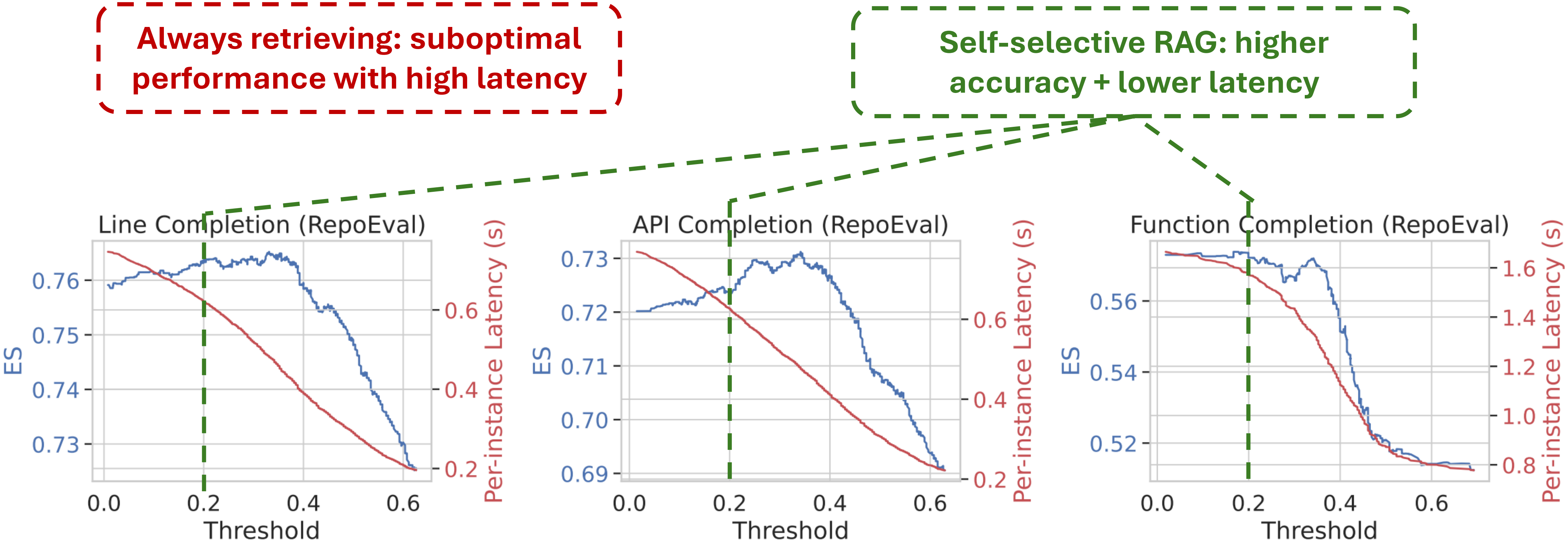


Figure 12. Latency-accuracy trade-off of self-selective RAG for REPOFORMER-1B.

Repoformer as a Plug-and-Play Policy

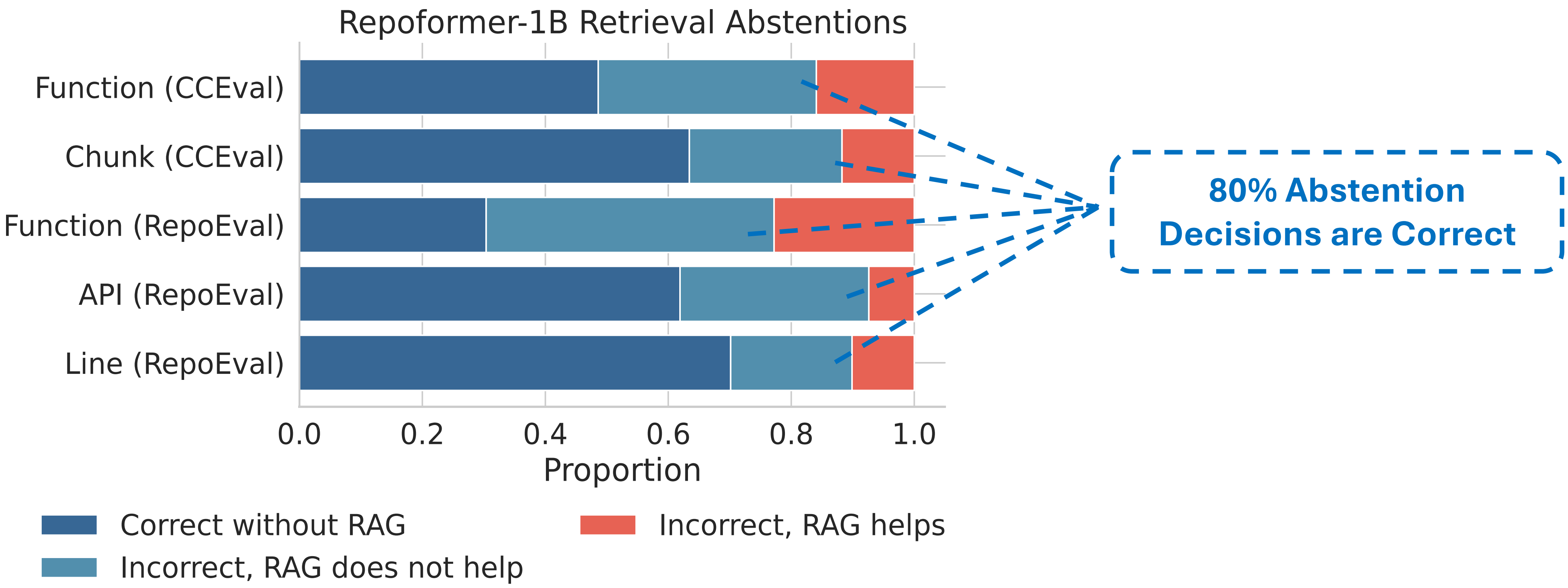
- **Repoformer helps larger models** to prevent uninformative and potentially harmful retrievals.

Model	RAG Policy	API		Line	
		ES	Speedup	ES	Speedup
StarCoderBase-16B	Always Retrieving	74.50	0%	79.24	0%
	Repoformer-1B	74.84	24%	79.48	24%
CodeLlama-16B	Always Retrieving	61.08	0%	61.58	0%
	Repoformer-1B	62.10	32%	62.45	30%
ChatGPT	Always Retrieving	63.38	0%	61.76	0%
	Repoformer-1B	64.01	28%	61.92	18%

Table 2: Accuracy and latency of larger LLMs as the code completion model and with Repoformer-1B as the policy model for selective RAG.

Accuracy of Retrieval Decisions

- Repoformer learns to make accurate abstention judgments.



Robustness to Retrieval

- Repoformer training improves the robustness to noisy retrieval.

Summary

- We propose **selective retrieval** for repository-level code completion.
- A **self-supervised learning** recipe for retrieval decision + code generation.
- Selective retrieval improves **accuracy** + **latency**
  - **Transferable** across code LLMs.

Paper

Code

Discussion

- Different approaches to “when to retrieve”

Self-RAG

FLARE

When and how to rely on retrieval in the kNN-LM

Question Difficulty

Model Uncertainty

Retrieval Quality

Performance-Oriented Learning

Efficient kNN-LM

RECOMP

SKR

Repoformer