CrossCodeEval: A Diverse and Multilingual Benchmark for Cross-File Code Completion

Yangruibo Ding[*1], Zijian Wang*[2], Wasi Uddin Ahmad*[2], Hantian Ding[2], Ming Tan, Nihal Jain, Murali Krishna Ramanathan, Ramesh Nallapati, Parminder Bhatia, Dan Roth, Bing Xiang (*Equal contribution)

[1] Columbia University
[2] AWS AI Labs

https://crosscodeeval.github.io/

Problem and Motivation

Cross-file Context is Important Yet Overlooked

Existing Code Completion Benchmarks[1, 2]

```
from typing import List
def has_close_elements(numbers: List[float], threshold: float):
    """
    Check if in given list of numbers,
    are any two numbers closer to each other than given threshold.
    >>> has_close_elements([1.0, 2.0, 3.0], 0.5)
    False
    >>> has_close_elements([1.0, 2.8, 3.0, 4.0, 5.0, 2.0], 0.3)
    True
    """
    [CURSOR_POSITION]
```

Real-world Code Completion

```
# maigret/test_utils.py
"""
Maigret utils test functions
"""
import itertools
import re
from maigret.utils import CaseConverter

def test_case_convert_camel_to_snake():
    a = 'SnakeCasedString'
    [CURSOR_POSITION]
```

```
# maigret/utils.py
class CaseConverter:
    @staticmethod
    def camel_to_snake(camelcased_string: str) -> str:
        ...
    @staticmethod
    def snake_to_camel(snakecased_string: str) -> str:
        ...
    @staticmethod
    def snake_to_title(snakecased_string: str) -> str:
        ...
```

Code Language Models

```
    for idx, elem in enumerate(numbers):
```

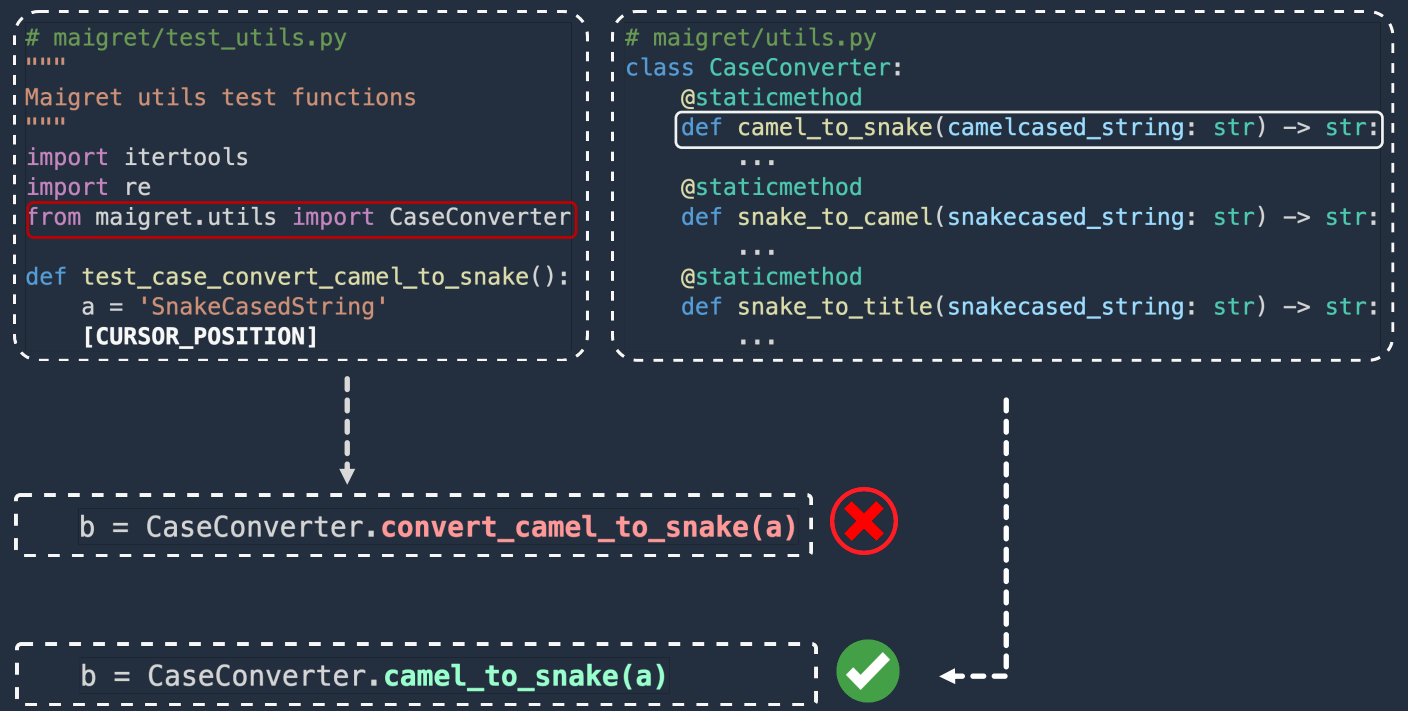
```
    b = CaseConverter.convert_camel_to_snake(a)
```

```
    b = CaseConverter.camel_to_snake(a)
```

[1] Chen et al., Evaluating Large Language Models Trained on Code. 2021
[2] Austin et al., Program Synthesis with Large Language Models, 2021

Cross-file Context is Overlooked Yet Important

Real-world Code Completion



Benchmark Construction

Locate Cross-file Dependencies

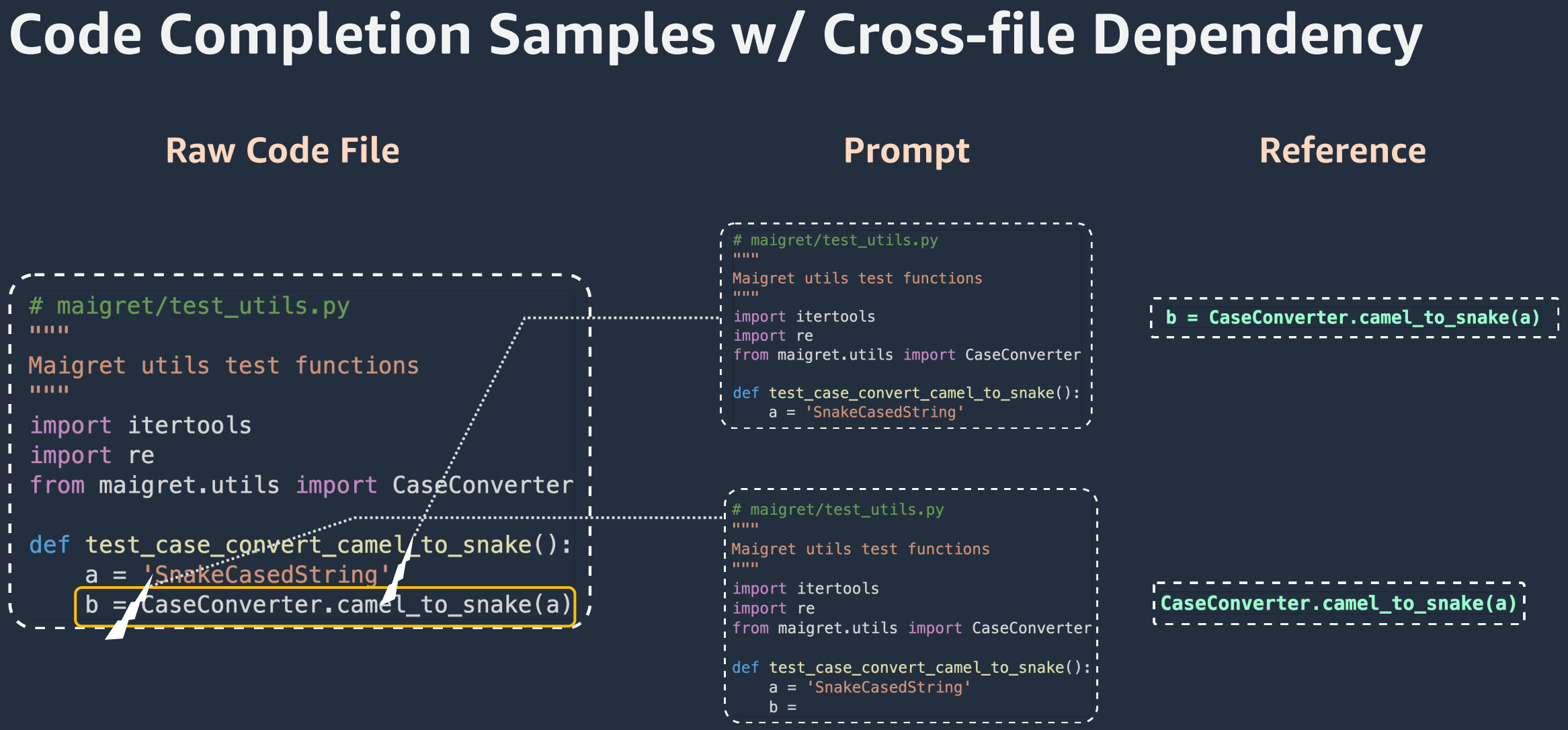
Static Analysis

```
# maigret/test_utils.py
"""
Maigret utils test functions
"""
import itertools
import re
from maigret.utils import CaseConverter

def test_case_convert_camel_to_snake():
    a = 'SnakeCasedString'
    b = CaseConverter.camel_to_snake(a)
```

Identify Intra-project Imports

Cross-file Dependency Detected!

Code Completion Samples w/ Cross-file Dependency



Data Collection: Quality Control

Permissive License

Created March to June, 2023

Code Repositories

Repos w/ few stars

Files w/ few lines

Samples w/ short references

Low-quality Data Filter

Memorized Sample

Strong In-file Hints

Easy Samples Removal

Correctness

Cross-file Dependency

Manual Verification

Diverse and Multilingual

- **Diverse: In total, more than 1,000 real-world code repositories.**
- **Multilingual: Four popular programming languages.**

Python

- ✓ Repositories: 471
- ✓ Files: 1368
- ✓ Samples: 2665

Java

- ✓ Repositories: 239
- ✓ Files: 745
- ✓ Samples: 2139

TypeScript

- ✓ Repositories: 193
- ✓ Files: 779
- ✓ Samples: 3356

C#

- ✓ Repositories: 99
- ✓ Files: 642
- ✓ Samples: 1768

Experiments

Experimental Setup

- **Models:** CodeGen (350M to 16B), StarCoder (1B to 15.5), and GPT-3.5-turbo.
- **Experiments:** Zero-shot prompting.
- **Metrics:** Code Match and Identifier Match

Incorporating Cross-file Context

In-file Context

```python
# maigret/test_utils.py
"""
Maigret utils test functions
"""
import itertools
import re
from maigret.utils import CaseConverter

def test_case_convert_camel_to_snake():
    a = 'SnakeCasedString'
    b = CaseConverter.
```

Retrieve

Cross-file Context

```python
# Here are relevant code fragments from other files:
# The below code fragment can be found in:
# maigret/utils.py
# class CaseConverter:
#     @staticmethod
#     def camel_to_snake(camelcased_string: str) -> str:
#         ...
#     @staticmethod
#     def snake_to_camel(snakecased_string: str) -> str:
#         ...
#     @staticmethod
#     def snake_to_title(snakecased_string: str) -> str:
#         ...
```

Prompt Format: In-file + Cross-file

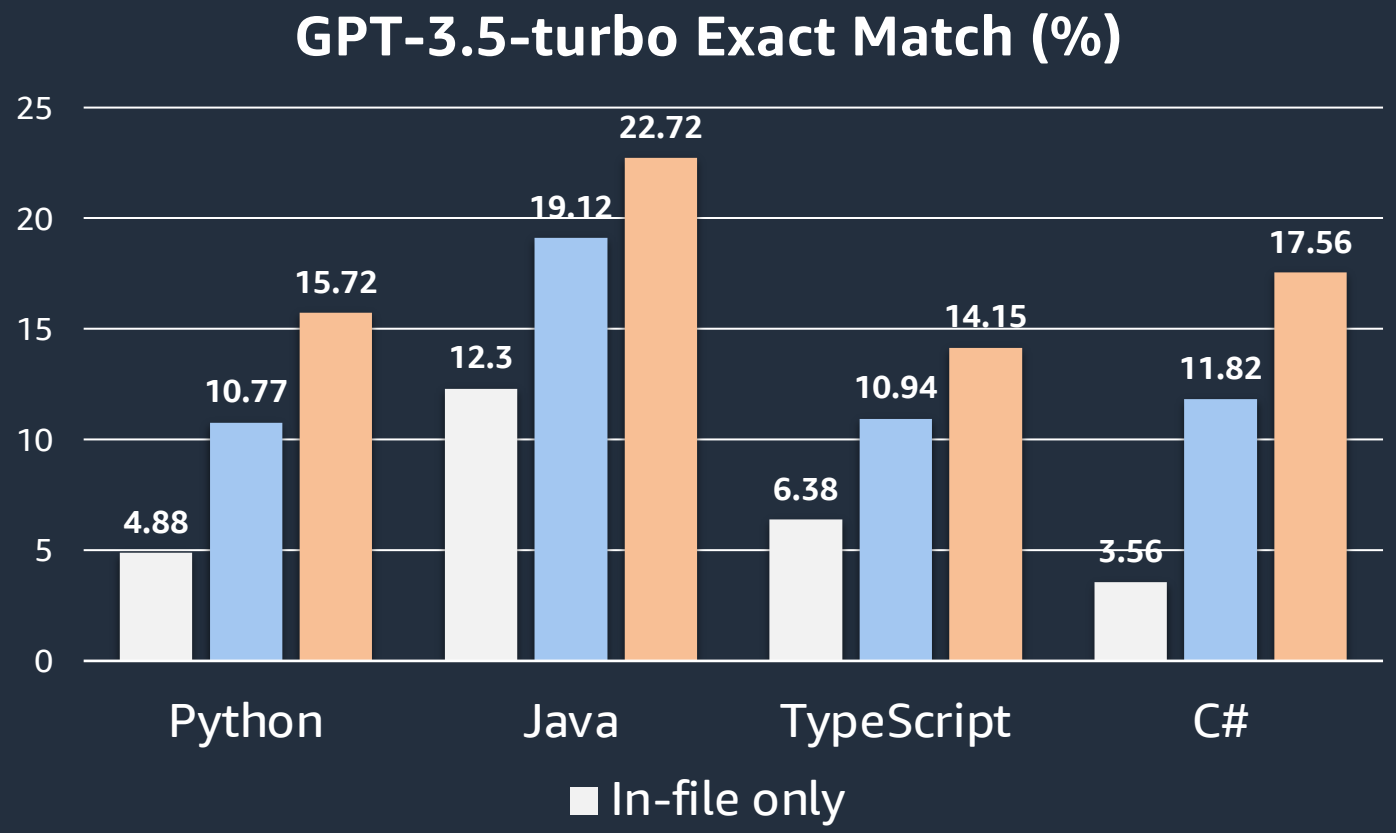
```
# Here are relevant code fragments from other files:
# The below code fragment can be found in:
# maigret/utils.py
# class CaseConverter:
#     @staticmethod
#     def camel_to_snake(camelcased_string: str) -> str:
#         ...
#     @staticmethod
#     def snake_to_camel(snakecased_string: str) -> str:
#         ...
#     @staticmethod
#     def snake_to_title(snakecased_string: str) -> str:
#         ...
"""
Maigret utils test functions
"""
import itertools
import re
from maigret.utils import CaseConverter

def test_case_convert_camel_to_snake():
    a = 'SnakeCasedString'
    b = CaseConverter.
```

Cross-file Context

In-file Context

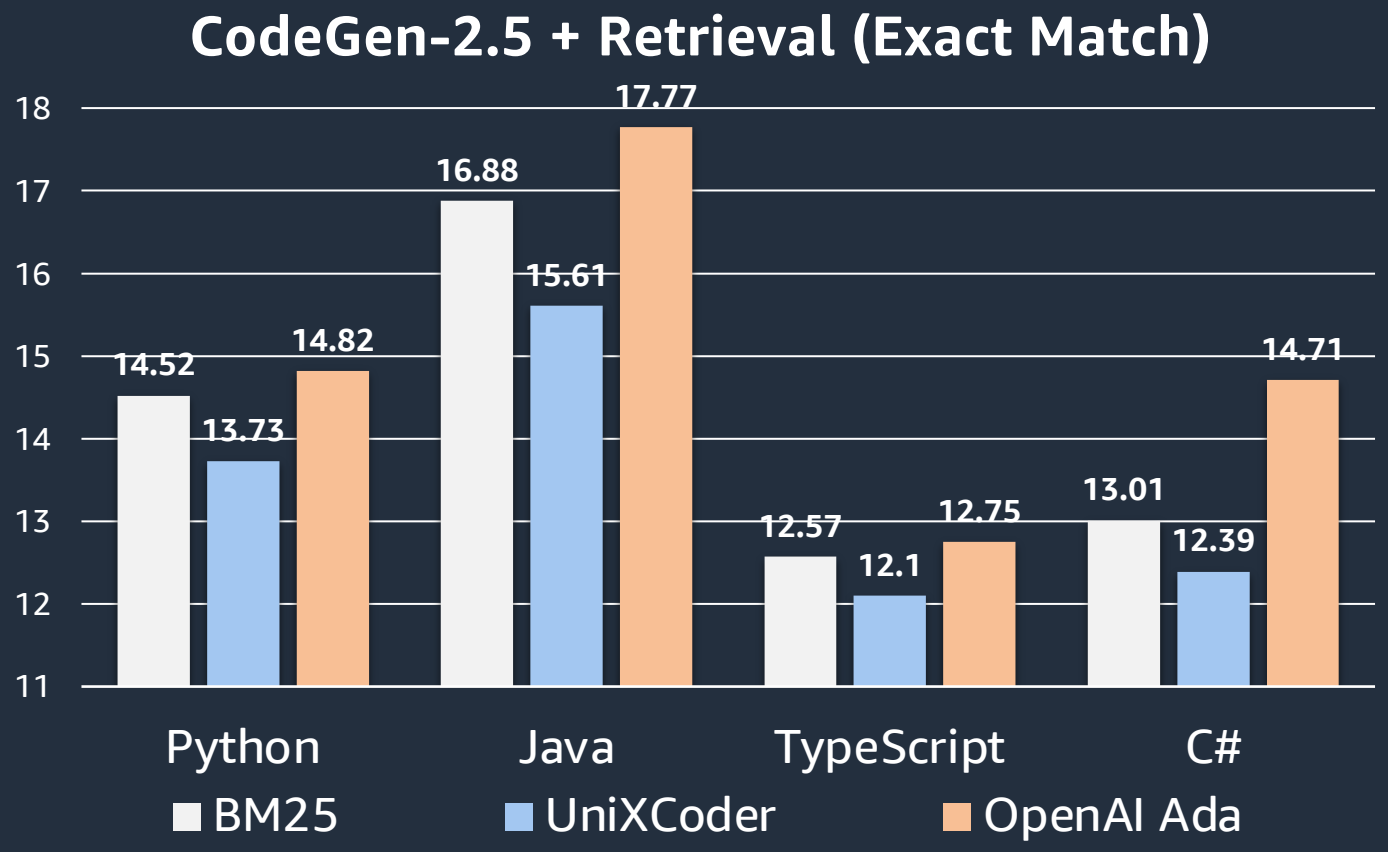
Cross-file Context Is Important For Code Completion

- CrossCodeEval is challenging, requiring cross-file context to complete.
- Cross-file context significantly improves code LMs' performance.
- With retrieved context, the overall accuracy is still not promising.



Context Retrieval Is Difficult

- Context retrievers' quality has notable impacts on performance.
- Better approaches to retrieve accurate cross-file context are needed.
- CrossCodeEval can also be used as a context retrieval benchmark.



Conclusion

Conclusion

https://crosscodeeval.github.io/

- CrossCodeEval is a diverse and multilingual benchmark that effectively assesses cross-file code completion capabilities of code LMs.
- SOTA code LMs perform poorly on CrossCodeEval with only in-file context. Stronger models to incorporating extensive context are necessary.
- Existing retrieval approaches could not always retrieve relevant context for CrossCodeEval samples. More effective retrievers for cross-file context are needed.

Thank you!