



Multi-lingual Detection of Incorrectness Inconsistencies between Code and Documentation

Xiaomeng Xu, Zahin Wahab, Reid Holmes, Caroline Lemieux

ISSTA 2026



THE UNIVERSITY
OF BRITISH COLUMBIA

Documentation vs. Code

```
1 /**
2  * If the passed object is an instance of {@link Service},
3  * add it to the list of services managed by this {@link CompositeService}
4  * @param object object.
5  * @return true if a service is added, false otherwise.
6  */
7 protected boolean addIfService(Object object) {
8     if (object instanceof Service) {
9         addService((Service) object);
10        return true;
11    } else {
12        return false;
13    }
14 }
```

Function-level documentation

What the code ought to do

Code

Documentation vs. Code

```
1  /**
2   * If the passed object is an instance of {@link Service},
3   * add it to the list of services managed by this {@link CompositeService}
4   * @param object object.
5   * @return true if a service is added, false otherwise.
6   */
7  protected boolean addIfService(Object object) {
8      if (object instanceof Service) {
9          addService((Service) object);
10         return true;
11     } else {
12         return false;
13     }
14 }
```

Function-level documentation

Code

What code actually does

Documentation vs. Code

```
1 /**
2  * If the passed object is an instance of {@link Service},
3  * add it to the list of services managed by this {@link CompositeService}
4  * @param object object.
5  * @return true if a service is added, false otherwise.
6  */
7 protected boolean addIfService(Object object) {
8     if (object instanceof Service) {
9         addService((Service) object);
10        return true;
11    } else {
12        return false;
13    }
14 }
```

Function-level documentation

What the code ought to do

Code

What code actually does

Detect and explain inconsistencies between code and documentation

Which Types of Inconsistencies Do We Care About?

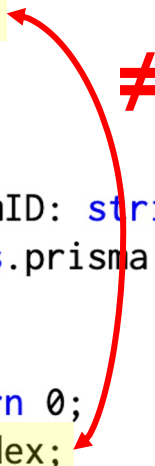
Motivation & Technique

Evaluation

In scope

Direct mismatch: logical conflicts between code and doc

```
1  /**
2   * Returns the count of root collections present for a given teamID. The count
   *   returned is highest OrderIndex + 1
3   * @param teamID The Team ID
4   * @returns Number of Root Collections
5   */
6  private async getRootCollectionsCount(teamID: string) {
7      const rootCollectionCount = await this.prisma.teamCollection.findMany({
8          where: { teamID, parentID: null },
9          orderBy: { orderIndex: 'desc', }, },);
10     if (!rootCollectionCount.length) return 0;
11     return rootCollectionCount[0].orderIndex;
```



In scope

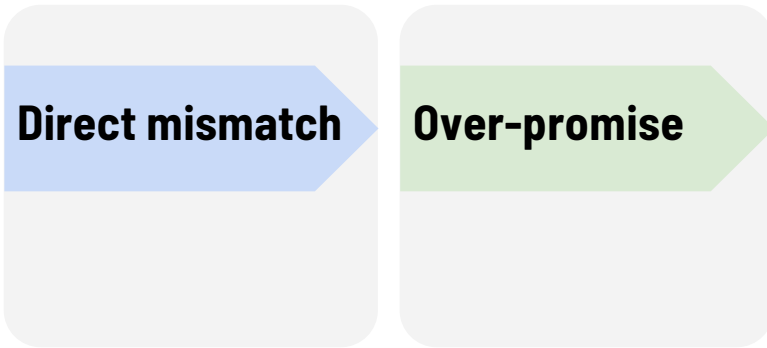
Over-promise: some doc behaviour not implemented

```
1 def set_test_mode(v: bool = True) -> None:
2     """Keeps track of whether numexpr was used. Stores an additional `True` for
   every successful use of evaluate with numexpr since the last `
   get_test_result`. """
3     global _TEST_MODE, _TEST_RESULT
4     _TEST_MODE = v
5     _TEST_RESULT = []
```



missing

Types of Inconsistencies



Incorrectness Inconsistencies:

- outdated documentation
- bugs in the code.

Out of scope

Under-promise: some code behaviour not documented

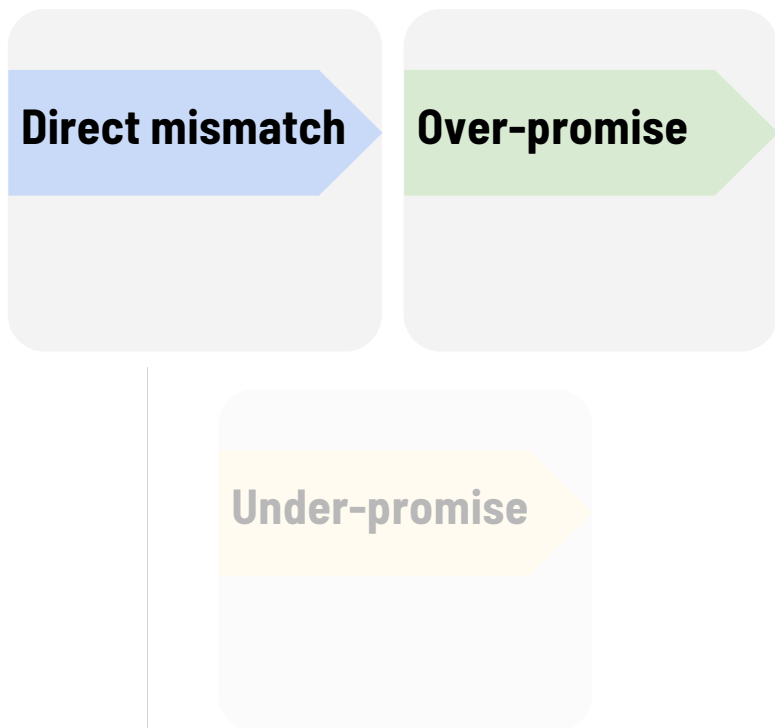
Points out the natural gap between concise doc and detailed code.

```
1 /**
2  * @name SORTED_FLOATS::operator[]
3  * Return the floating point value of the given index into the list.
4  */
5 float SORTED_FLOATS::operator[](int32_t index) {
6     it.move_to_first();
7     return it.data_relative(index)->entry;
8 }
```

missing

Goal of DocPrism (our tool)

Critical



Incorrectness Inconsistencies:

- outdated documentation
- bugs in the code

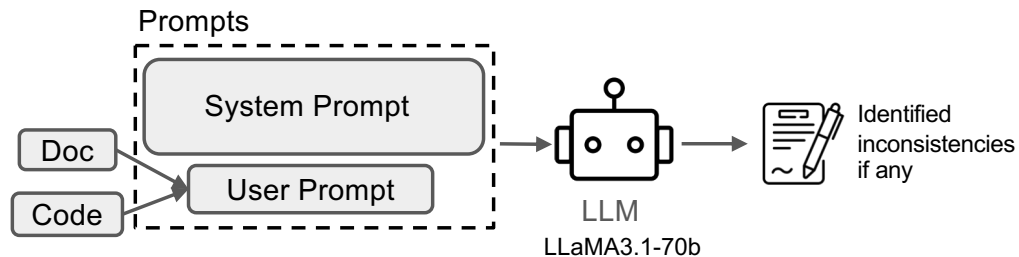
Incompleteness Inconsistencies:

Which Types of Inconsistencies
Do We Care About?

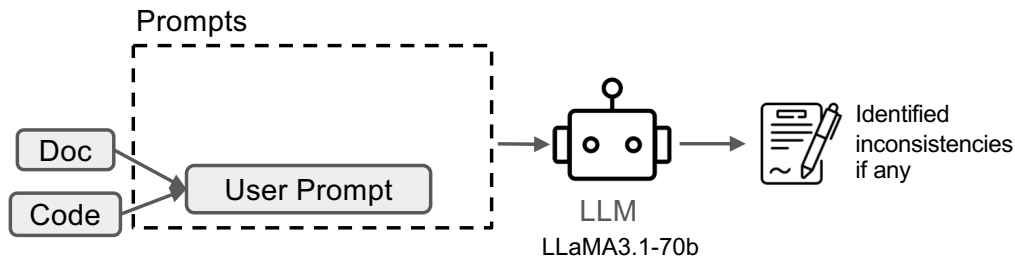
Motivation & Technique

Evaluation

V1: Ask the LLM to detect inconsistencies



V1: Ask the LLM to detect inconsistencies



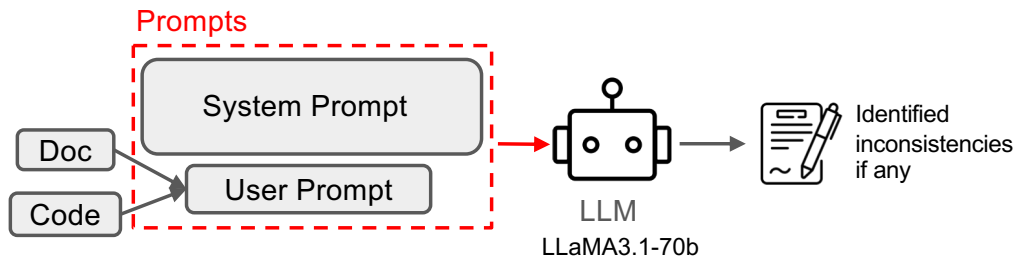
System Prompt

...

Given the code (...) and its accompanying documentation (...), ***please identify any inconsistencies or information mismatches between them.***

...

V1: Ask the LLM to detect inconsistencies



1,991

code-doc pairs
from 20 codebases
across 4 languages

1,852

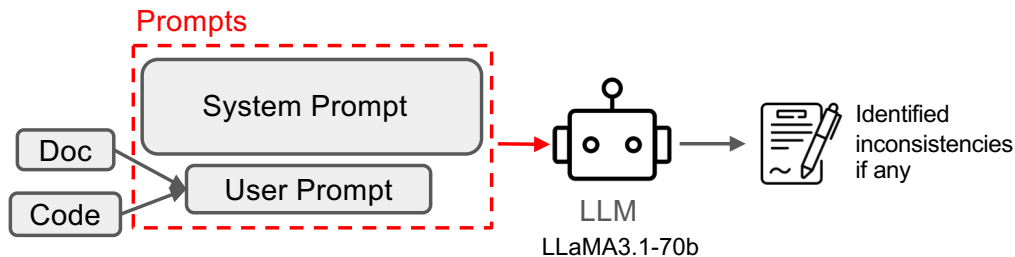
code-doc pairs
flagged as
inconsistent

93%

flag rate

- Practically unusable
- Probably not true

V1: Ask the LLM to detect inconsistencies



1,991

code-doc pairs
from 20 codebases
across 4 languages

1,852

code-doc pairs
flagged as
inconsistent

93%

flag rate

Manual analysis on Python results

96

detected
inconsistencies

69

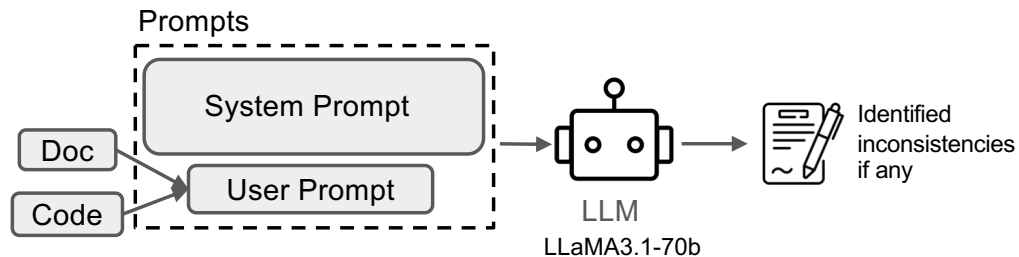
under-promise
inconsistencies

72%

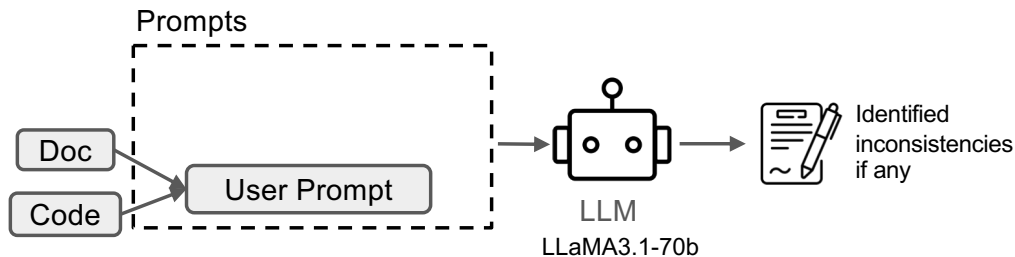
under-promise
rate

Maybe we can try to tell the LLM not to
report under-promise inconsistencies.

V2: Ask the LLM not to report under-promises



V2: Ask the LLM not to report under-promises



System Prompt

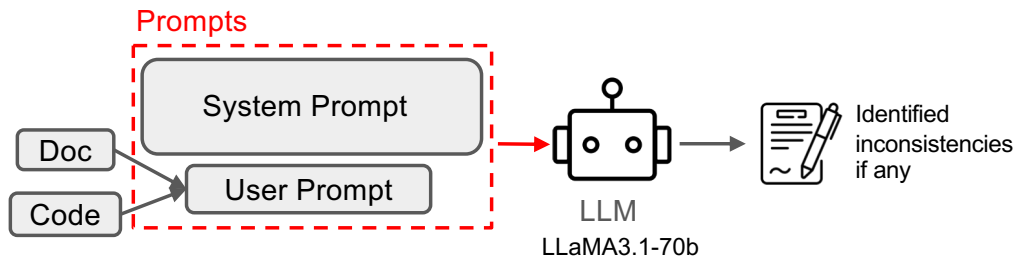
...

++ <introducing three types of inconsistencies>

++ Do not report inconsistencies in the third category, under-promise, in the final output.

...

V2: Ask the LLM not to report under-promises



1,991

code-doc pairs
from 20 codebases
across 4 languages

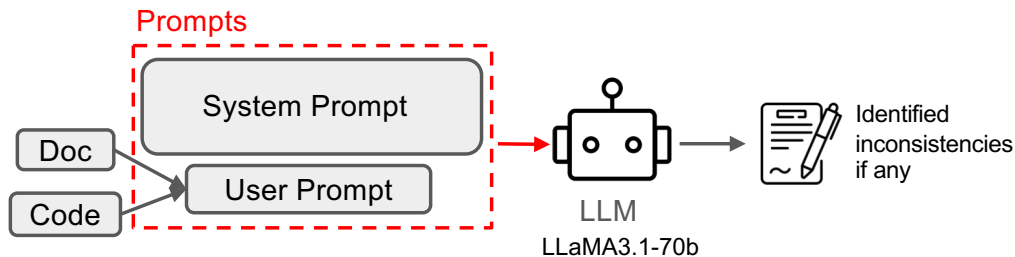
1,871

code-doc pairs
flagged as
inconsistent

94%

flag rate

V2: Ask the LLM not to report under-promises



1,991

code-doc pairs
from 20 codebases
across 4 languages

Manual analysis on Python results

The LLM itself is not good at
filtering out under-promises.

1,871

code-doc pairs
flagged as
inconsistent

95

detected
inconsistencies

46

under-promise
inconsistencies

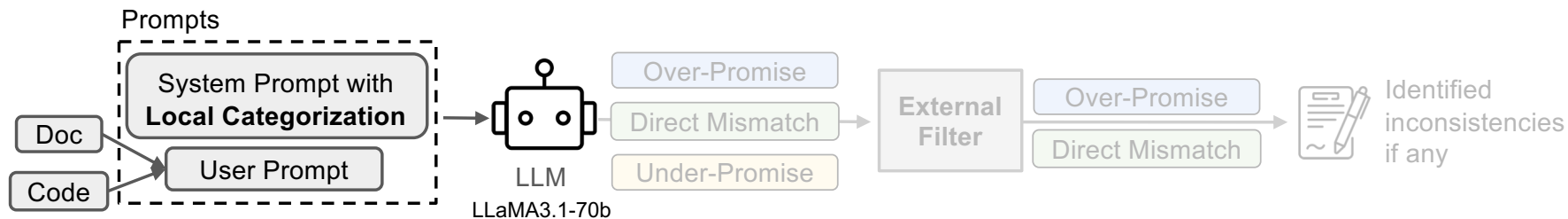
48%

under-promise
rate

94%

flag rate

Our tool DocPrism



Our tool DocPrism

System Prompt with ***Local Categorization***

- Ask the LLM to generate output following this schema

{

“Documentation_Summary”: “...”

prompt the LLM to
do specific task

“Code_Summary”: “...”

Task 1: Detecting over-promise inconsistencies

Task 2: Detecting direct mismatch inconsistencies

Task 3: Detecting under-promise inconsistencies

}

Our tool DocPrism

Task 1: Detecting over-promise inconsistencies

This is the full prompt for task 1.

Check-in key

```
"Is_any_core_part_of_the_documentation_not_implemented_in_the_code?": "Yes" or "No"  
"If_yes_to_Is_any_core_part_of_the_documentation_not_implemented_in_the_code":  
- "original_documentation_snippet_that_is_not_implemented_in_the_code"  
- "explanation"
```

Follow-up key

Our tool DocPrism

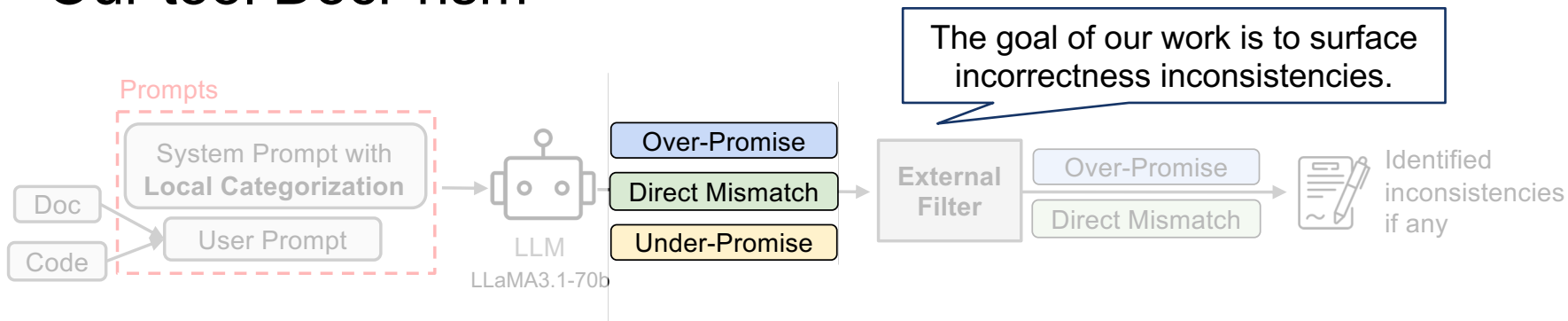
System Prompt with ***Local Categorization***

- Ask the LLM to generate output following this schema

```
{  
  "Documentation_Summary": "..."  
  "Code_Summary": "..."  
  Task 1: Detecting over-promise inconsistencies  
  Task 2: Detecting direct mismatch inconsistencies  
  Task 3: Detecting under-promise inconsistencies  
}
```

As the LLM sequentially generates outputs for each pair of keys, it is *implicitly* guided to detect and list inconsistencies in each category.

Our tool DocPrism



1,991

code-doc pairs
from 20 codebases
across 4 languages

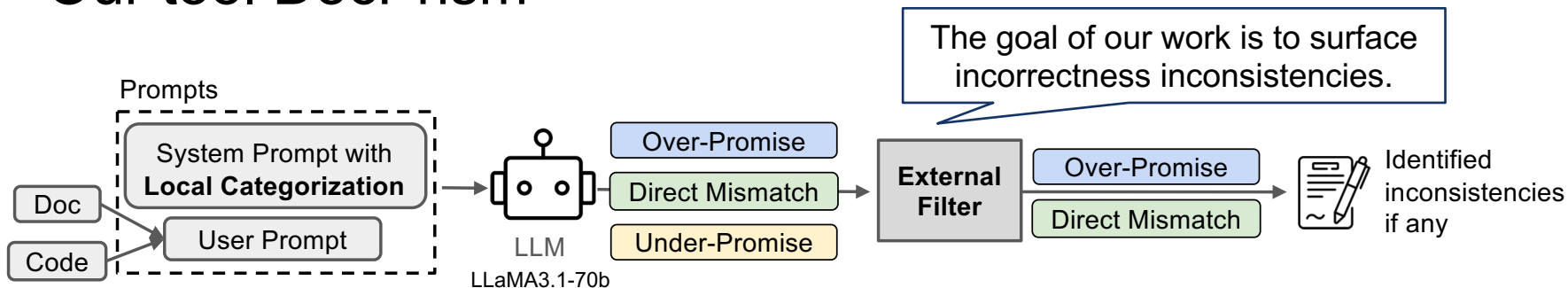
337

code-doc pairs
flagged as
inconsistent

17%

flag rate

Our tool DocPrism



1,991

code-doc pairs
from 20 codebases
across 4 languages

337

code-doc pairs
flagged as
inconsistent

17%

flag rate

Manual analysis on Python results

10

detected
inconsistencies

0

under-promise
inconsistencies

0%

under-promise
rate

Which inconsistencies do we care about?

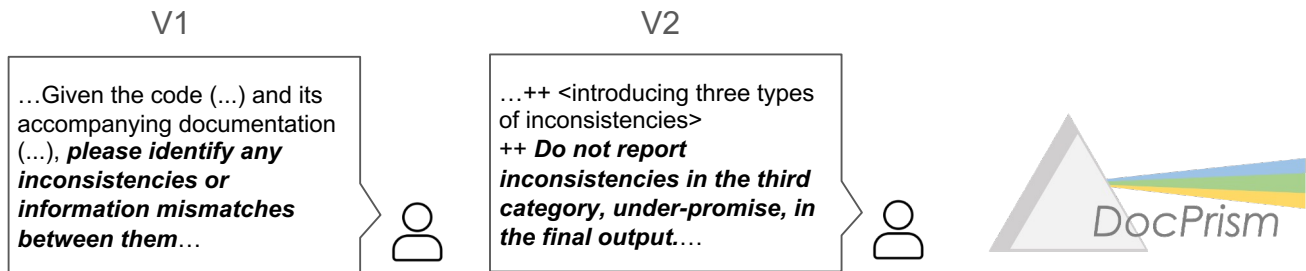
Motivation & Technique

Evaluation

Evaluation

- Ablation study
 - To evaluate effects of different design decision in DocPrism
- Comparison with baseline C4RLLaMA
- Performance on multilingual datasets (TypeScript, Python, Java, C++)
 - taxonomy of correctly and incorrectly identified inconsistencies
- Generalization to LLMs other than LLaMA3.1-70b

Partial Ablation Results



Flag rate	94%	96%	14%
Under-promise rate	72%	48%	0%
Precision	0.12	0.1	0.71

Comparison with C4RLLaMA

Panthaplackel dataset

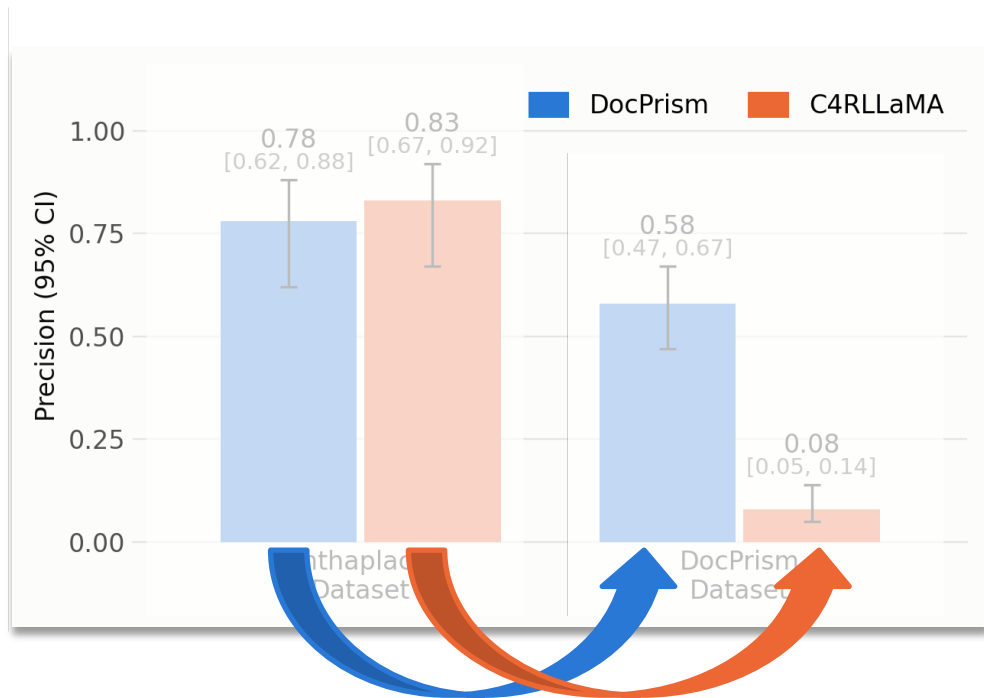
(Used to evaluate C4RLLaMA)

Takeaways:

- DocPrism shows more stable performance than C4RLLaMA across synthetic and wild data.
- C4RLLaMA's performance drop highlights the importance of evaluating on diverse datasets, especially those containing code-doc pairs as they exist in the wild.

500 <code, documentation>

Code-doc pairs in the wild

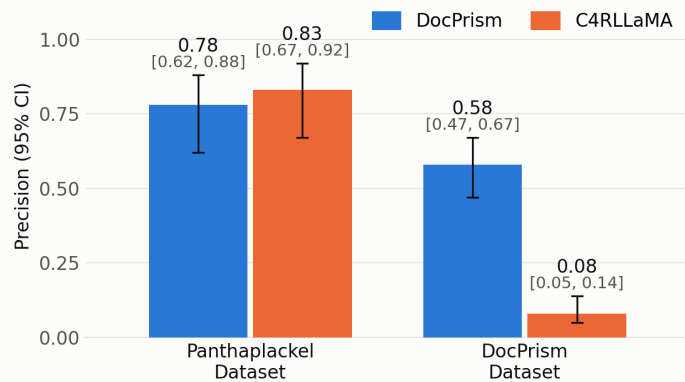


DocPrism: Multi-lingual code-doc inconsistency detection

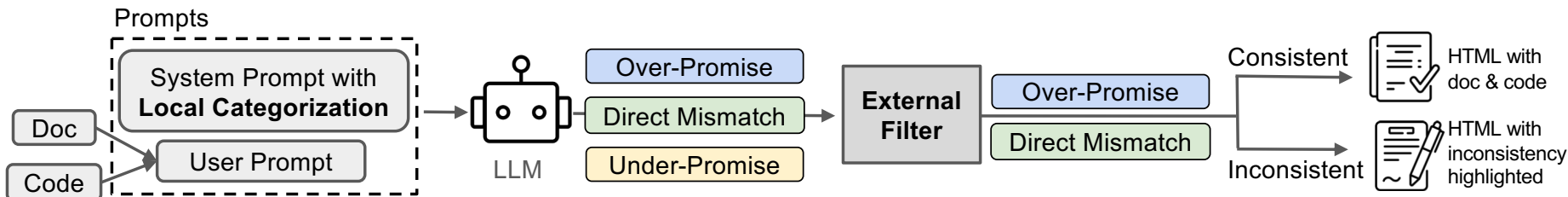
```
1 /**
2  * Returns the count of root collections present for a given teamID. The count
3  *   returned is highest OrderIndex + 1
4  * @param teamID The Team ID
5  * @returns Number of Root Collections
6  */
7 private async getRootCollectionsCount(teamID: string) {
8   const rootCollectionCount = await this.prisma.teamCollection.findMany({
9     where: { teamID, parentID: null },
10    orderBy: {orderIndex: 'desc',},});
11   if (!rootCollectionCount.length) return 0;
12   return rootCollectionCount[0].orderIndex;
```

A red curved arrow points from the comment on line 5 to the return statement on line 12, with a red $\neq$ symbol in the middle, indicating a surface incorrectness inconsistency where the comment does not match the actual code behavior.

Surface incorrectness inconsistencies



SOTA comparison



DocPrism Design