The 31st ACM/IEEE International Conference on Program Comprehension (ICPC), 2023

Automating Method Naming with Context-Aware Prompt-Tuning

Jie Zhu, Lingwei Li, Li Yang*, Xiaoxiao Ma, Chun Zuo

(Tel: Li Yang*, yangli2017@iscas.ac.cn, 010-62661198)

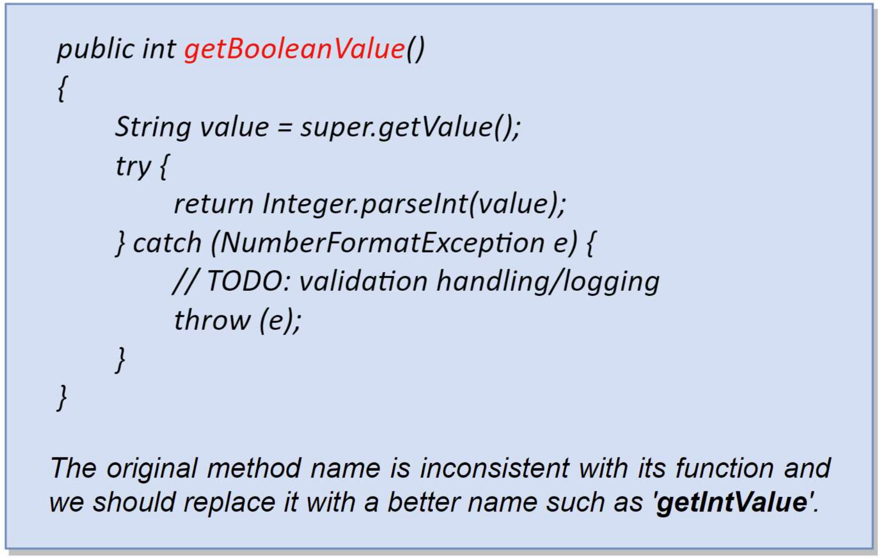
ISCAS 中国科学院大学 University of Chinese Academy of Sciences 中科软科技 Sinosoft Co.,Ltd.

Background

In software development, method names provide significant information of program functionality with developers.

However, method names could also be confusing, making programs even harder to understand and more error-prone.

To ease developers' burden on method naming, many approaches have been proposed to improve the naming quality and code readability.



```
public int getBooleanValue()
{
        String value = super.getValue();
        try {
                return Integer.parseInt(value);
        } catch (NumberFormatException e) {
                // TODO: validation handling/logging
                throw (e);
        }
}
```

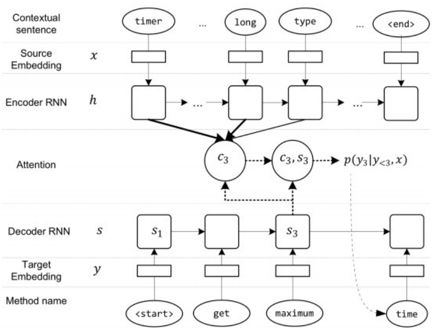
*The original method name is inconsistent with its function and we should replace it with a better name such as '**getIntValue**'.*

Task1: detect inconsistent method name

Task2: recommend a high-quality method name as candidate

Phil Karlton: "There are only two hard things in Computer Science: cache invalidation and naming things. "

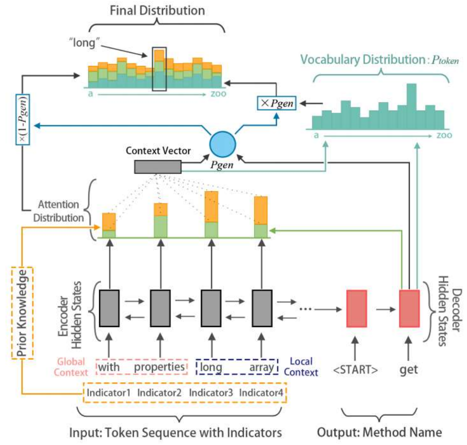
Related Work



MNire[1]

MNire first generates a candidate name with a RNN-based model and compares the current name against it.

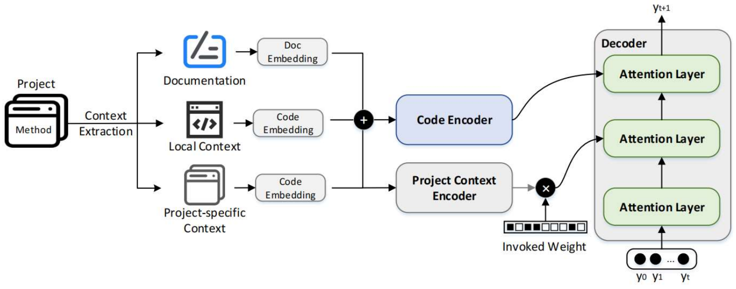
Generate-then-Compare



Cognac[2]

Cognac guides method name recommendation with local and global contexts.

Global Context



GTNM[3]

GTNM considers different contexts and employs a transformer-based neural network to suggest method names.

Transformer

[1] S. Nguyen, H. Phan, T. Le, and T. N. Nguyen, "Suggesting natural method names to check name consistencies," in ICSE '20: 42nd International Conference on Software Engineering

[2] S. Wang, M. Wen, B. Lin, and X. Mao, "Lightweight global and local contexts guided method name recommendation with prior knowledge," in ESEC/FSE '21: 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering

[3] F. Liu, G. Li, Z. Fu, S. Lu, Y. Hao, and Z. Jin, "Learning to recommend method names with global context," in ICSE '22: 44th International Conference on Software Engineering

Motivation

Limitation 1 All models of previous deep learning based methods are trained from scratch, learning two distinct objectives simultaneously:

- ❑ Learn the semantic representation of PL & NL
- ❑ Learn the relationship between method name and its implementation

↓

The misalignment between the two optimizing objectives decreases the efficiency of training and thus leads to sub-optimal results.

Motivation

Limitation 2

Most recent method name consistency checking approaches, including MNire and Cognac, follow a generate-then-compare strategy to detect inconsistent method names, facing difficulty measuring the semantic consistency.

```
protected void doSetup(Context context) {
     super.bindCurrentConfiguration(context.getConfiguration());
}
```

Generated name: *doBindCurrentConfiguration*

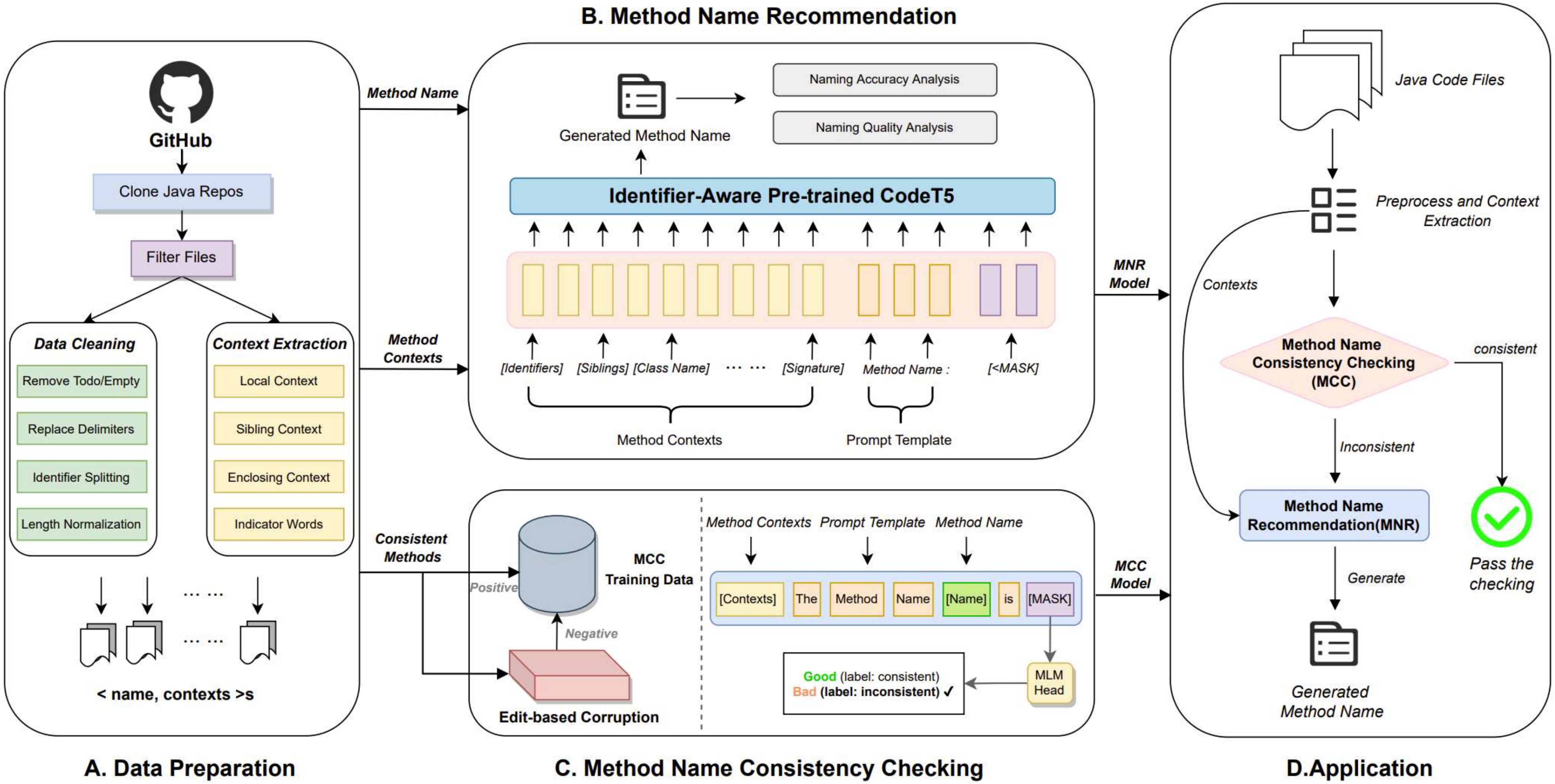
Original name: *doSetup*

$$Sim(p,c) = \frac{numOfSharedTokens(p,c)}{(numOfTokens(p) + numOfTokens(c))/2}$$

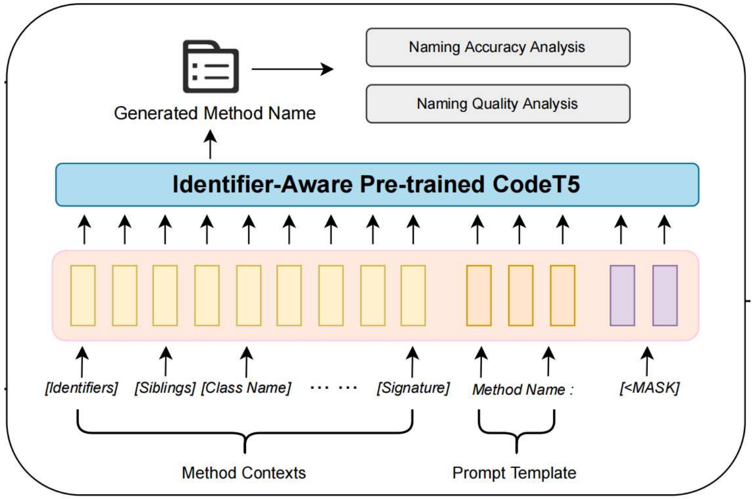
$$\text{similarity} = \frac{1}{(4+3)*0.5} \approx 0.286 < threshold$$

However, method names with completely different sub-tokens could be semantically similar, while names with high lexical similarity could have totally different meanings. Thus, we need to develop a new approach to detect inconsistent names without requiring calculating lexical similarity.

Approach

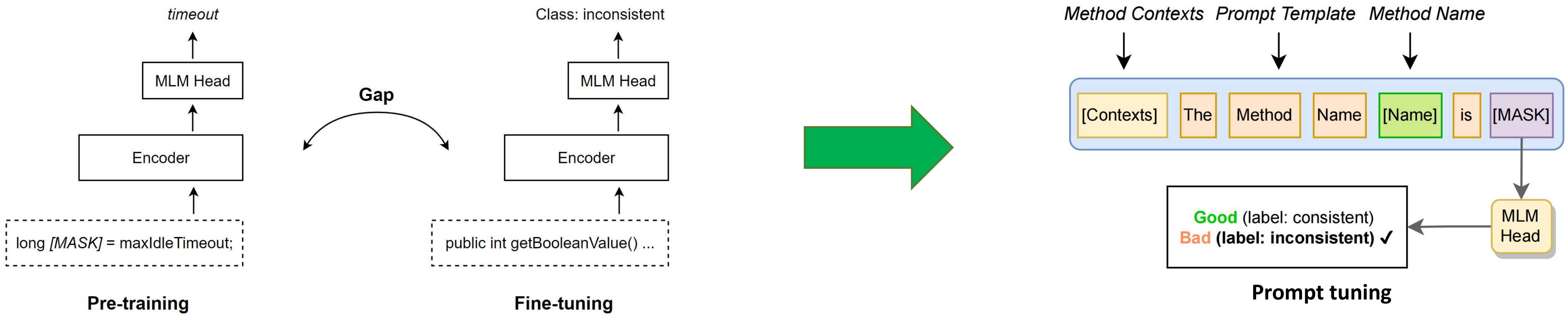

B. Method Name Recommendation
GitHub
Clone Java Repos
Filter Files
Data Cleaning
Remove Todo/Empty
Replace Delimiters
Identifier Splitting
Length Normalization
Context Extraction
Local Context
Sibling Context
Enclosing Context
Indicator Words
… …
… …
< name, contexts >s
Method Name
Method Contexts
Consistent Methods
Generated Method Name
Naming Accuracy Analysis
Naming Quality Analysis
Identifier-Aware Pre-trained CodeT5
[Identifiers]
[Siblings]
[Class Name]
… …
[Signature]
Method Name :
[<MASK]
Method Contexts
Prompt Template
MNR Model
MCC Model
Positive
Negative
MCC Training Data
Edit-based Corruption
Method Contexts
Prompt Template
Method Name
[Contexts]
The
Method
Name
[Name]
is
[MASK]
MLM Head
Good (label: consistent)
Bad (label: inconsistent) ✓
Java Code Files
Preprocess and Context Extraction
Contexts
Method Name Consistency Checking (MCC)
consistent
Inconsistent
Method Name Recommendation(MNR)
Generate
Pass the checking
Generated Method Name
A. Data Preparation
C. Method Name Consistency Checking
D.Application

Approach



Adopt the "pre-train, prompt, and predict" paradigm to detect inconsistent method names and generate high-quality names.

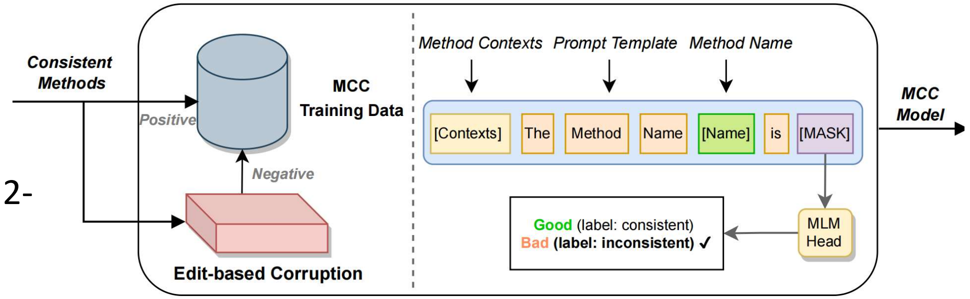
Prompt tuning helps bridge the gap between pre-training and tuning on downstream tasks, which contributes to fully exploiting the knowledge and capacity of the pre-trained CodeT5 model.



Approach

Training Method Name Consistency Checking Model

- How to detect inconsistency without calculating lexical similarity
  - Model this problem as a binary classification task directly and train a 2-class classification model to detect.
  - But how to collect enough classification training data?
- Collect enough training data for method name consistency model training
  - Consistent Examples: Easy to acquire. (from MNR dataset)
  - Inconsistent Examples: Difficult to collect (no open large dataset, and also almost impossible to crawl from OSS community)
  - Another important point is that even inconsistent method names are usually closely related to its context.
    - Thus, we need Related-but-Inconsistent names as training examples (hard negative mining).



C. Method Name Consistency Checking

Option1: Random sample names from other methods as inconsistent (inconsistent, but not related)

Option2: Random sample names from other methods within the same file as inconsistent (close, but not necessarily inconsistent)

Option3: Corrupt the original name to make it slightly different (related and inconsistent) √

Approach

How to corrupt the original name to make it close-but-inconsistent?

- I try to follow the idea of BART to corrupt the original method name

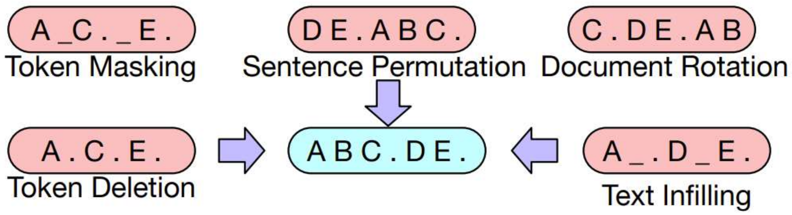
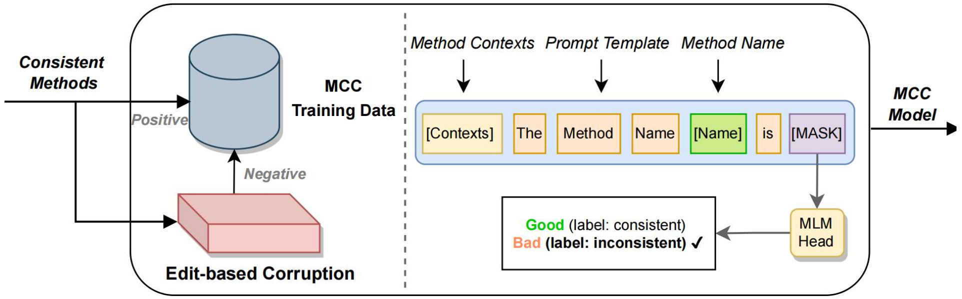


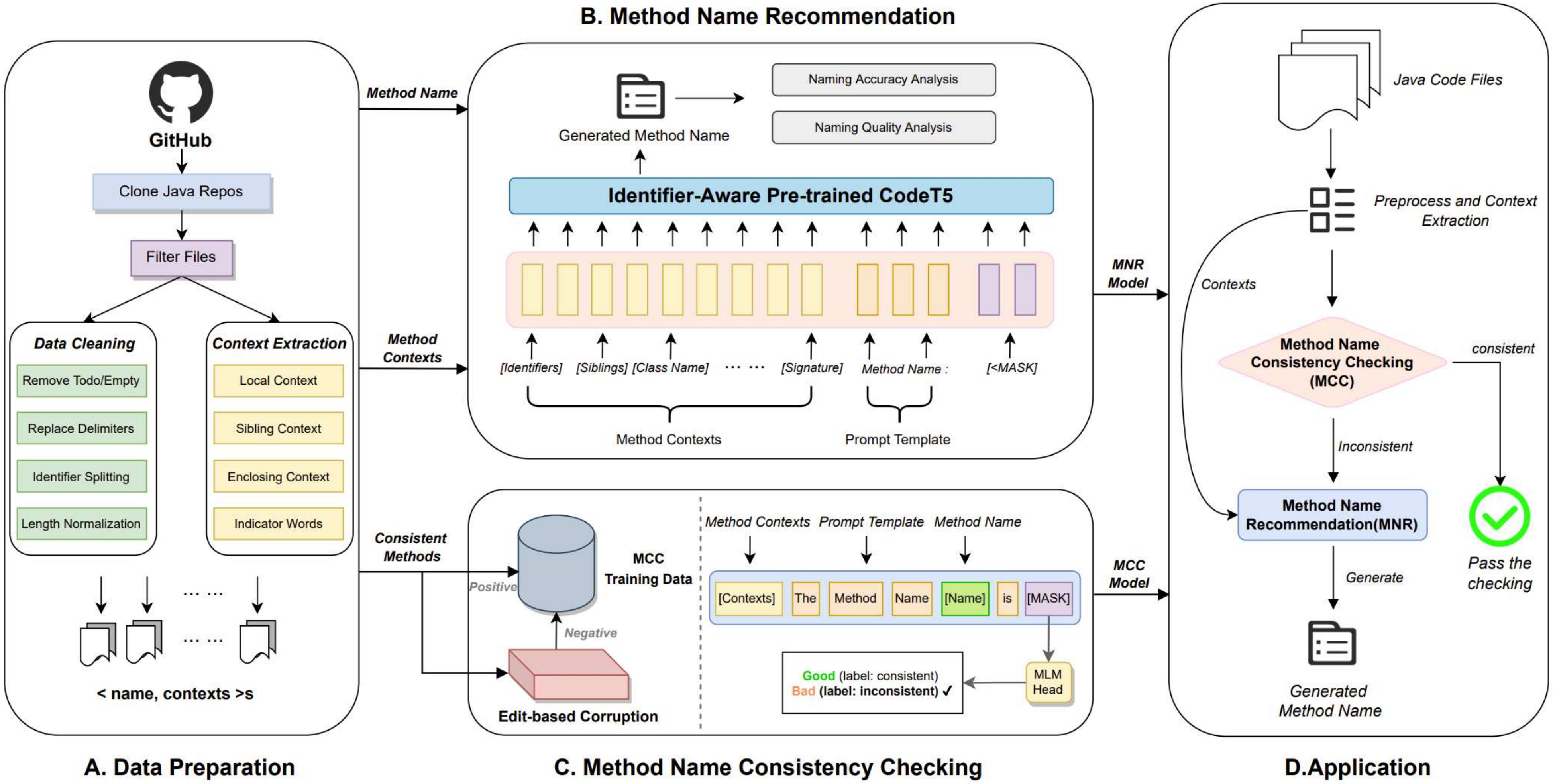
Figure 2: Transformations for noising the input that we experiment with. These transformations can be composed.



C. Method Name Consistency Checking

Original Name: *check Initial Padding* => Generated Inconsistent Name: *get Initial Padding*

Approach


B. Method Name Recommendation
GitHub
Clone Java Repos
Filter Files
Data Cleaning
Remove Todo/Empty
Replace Delimiters
Identifier Splitting
Length Normalization
Context Extraction
Local Context
Sibling Context
Enclosing Context
Indicator Words
… …
< name, contexts >s
Method Name
Method Contexts
Consistent Methods
Generated Method Name
Naming Accuracy Analysis
Naming Quality Analysis
Identifier-Aware Pre-trained CodeT5
[Identifiers]
[Siblings]
[Class Name]
… …
[Signature]
Method Name :
[<MASK]
Method Contexts
Prompt Template
MNR Model
MCC Model
Positive
Negative
MCC Training Data
Edit-based Corruption
Method Contexts
Prompt Template
Method Name
[Contexts]
The
Method
Name
[Name]
is
[MASK]
MLM Head
Good (label: consistent)
Bad (label: inconsistent) ✓
Java Code Files
Preprocess and Context Extraction
Contexts
Method Name Consistency Checking (MCC)
consistent
Inconsistent
Method Name Recommendation(MNR)
Pass the checking
Generate
Generated Method Name
A. Data Preparation
C. Method Name Consistency Checking
D.Application

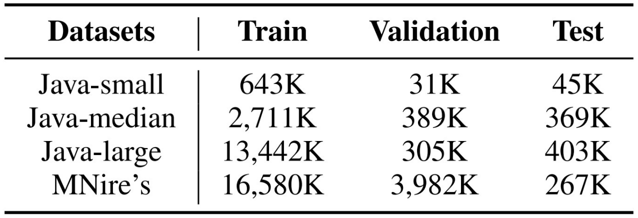
Evaluation

Method Name Recommendation Task

1. Focus on learning method name recommendation with better understanding of NL & PL tokens from pretrained model

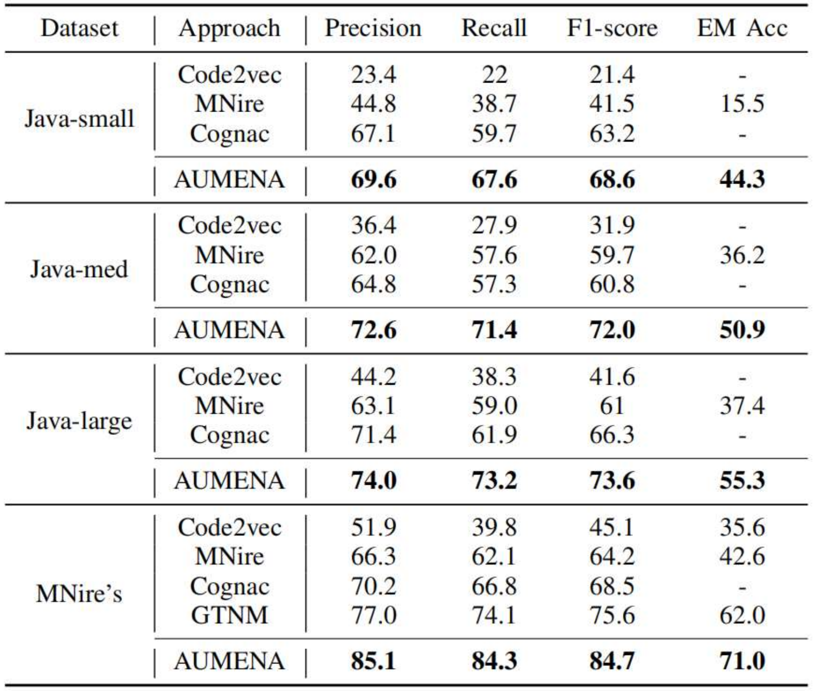
2. Fully exploit the knowledge and capacity of pre-trained model with prompt-tuning

TABLE I
STATISTICS OF THE MNR DATASETS



Datasets	**Train**	**Validation**	**Test**
Java-small	643K	31K	45K
Java-median	2,711K	389K	369K
Java-large	13,442K	305K	403K
MNire's	16,580K	3,982K	267K

TABLE III
RESULTS OF METHOD NAME RECOMMENDATION



Dataset	Approach	Precision	Recall	F1-score	EM Acc
Java-small	Code2vec	23.4	22	21.4	-
	MNire	44.8	38.7	41.5	15.5
	Cognac	67.1	59.7	63.2	-
	AUMENA	**69.6**	**67.6**	**68.6**	**44.3**
Java-med	Code2vec	36.4	27.9	31.9	-
	MNire	62.0	57.6	59.7	36.2
	Cognac	64.8	57.3	60.8	-
	AUMENA	**72.6**	**71.4**	**72.0**	**50.9**
Java-large	Code2vec	44.2	38.3	41.6	-
	MNire	63.1	59.0	61	37.4
	Cognac	71.4	61.9	66.3	-
	AUMENA	**74.0**	**73.2**	**73.6**	**55.3**
MNire's	Code2vec	51.9	39.8	45.1	35.6
	MNire	66.3	62.1	64.2	42.6
	Cognac	70.2	66.8	68.5	-
	GTNM	77.0	74.1	75.6	62.0
	AUMENA	**85.1**	**84.3**	**84.7**	**71.0**

Evaluation

Method Name Consistency Checking Task

1. Better initialization from pretrained model

2. Improving the overall accuracy via measuring semantic consistency instead of calculating lexical similarity

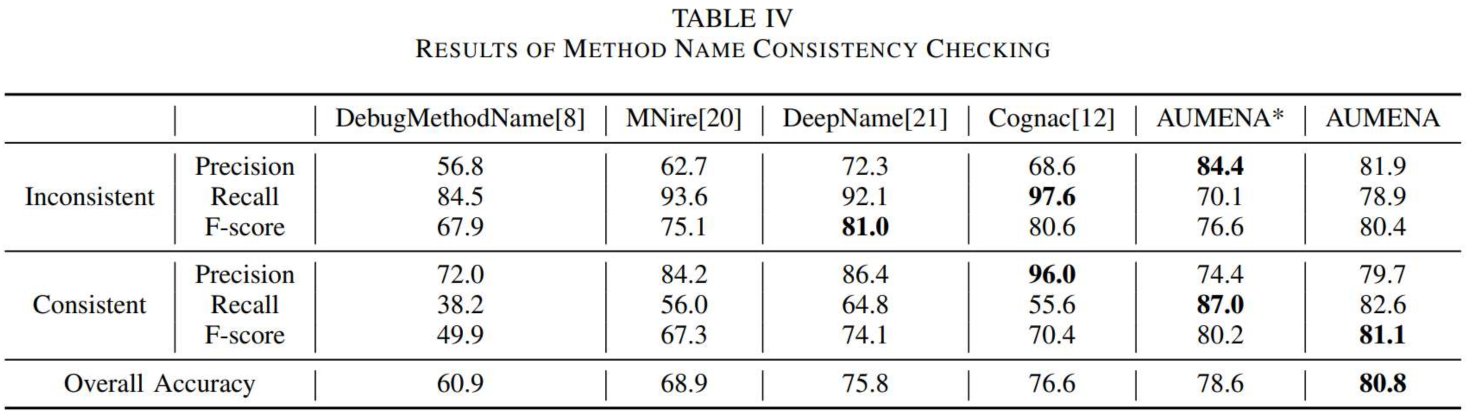
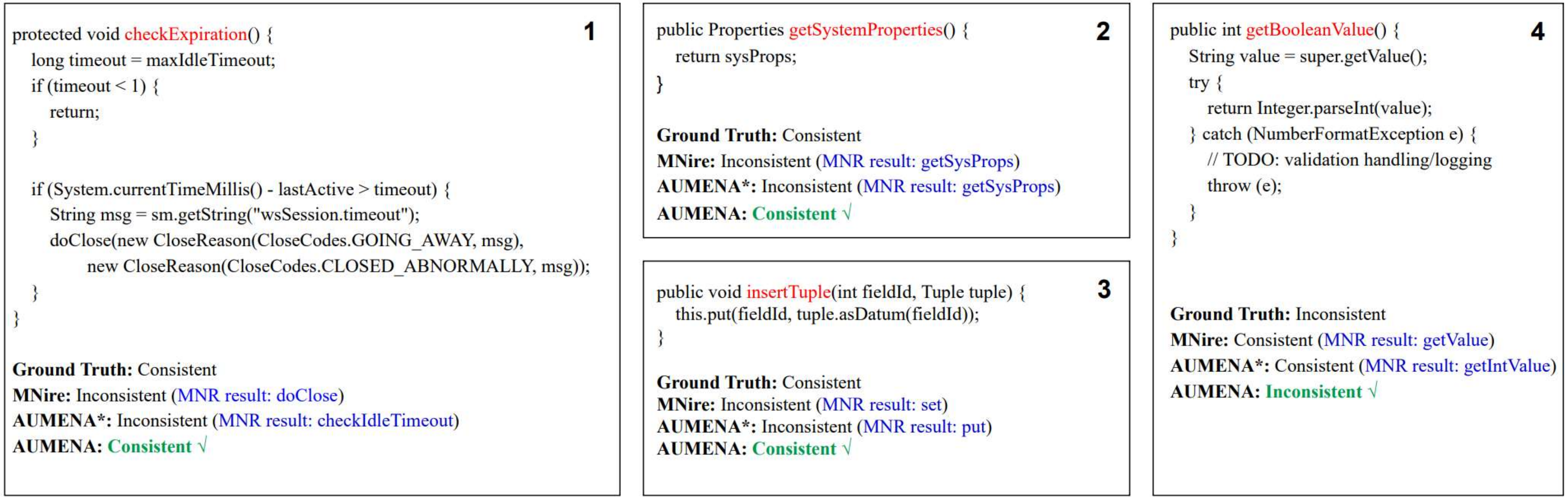


TABLE IV
RESULTS OF METHOD NAME CONSISTENCY CHECKING

		DebugMethodName[8]	MNire[20]	DeepName[21]	Cognac[12]	AUMENA*	AUMENA
Inconsistent	Precision	56.8	62.7	72.3	68.6	**84.4**	81.9
	Recall	84.5	93.6	92.1	**97.6**	70.1	78.9
	F-score	67.9	75.1	**81.0**	80.6	76.6	80.4
Consistent	Precision	72.0	84.2	86.4	**96.0**	74.4	79.7
	Recall	38.2	56.0	64.8	55.6	**87.0**	82.6
	F-score	49.9	67.3	74.1	70.4	80.2	**81.1**
Overall Accuracy		60.9	68.9	75.8	76.6	78.6	**80.8**

Case Study

Capable of measuring semantic consistency instead of totally depending on lexical similarity



1

```
protected void checkExpiration() {
    long timeout = maxIdleTimeout;
    if (timeout < 1) {
        return;
    }

    if (System.currentTimeMillis() - lastActive > timeout) {
        String msg = sm.getString("wsSession.timeout");
        doClose(new CloseReason(CloseCodes.GOING_AWAY, msg),
                new CloseReason(CloseCodes.CLOSED_ABNORMALLY, msg));
    }
}
```

Ground Truth: Consistent
MNire: Inconsistent (MNR result: doClose)
AUMENA*: Inconsistent (MNR result: checkIdleTimeout)
AUMENA: **Consistent √**

2

```
public Properties getSystemProperties() {
    return sysProps;
}
```

Ground Truth: Consistent
MNire: Inconsistent (MNR result: getSysProps)
AUMENA*: Inconsistent (MNR result: getSysProps)
AUMENA: **Consistent √**

3

```
public void insertTuple(int fieldId, Tuple tuple) {
    this.put(fieldId, tuple.asDatum(fieldId));
}
```

Ground Truth: Consistent
MNire: Inconsistent (MNR result: set)
AUMENA*: Inconsistent (MNR result: put)
AUMENA: **Consistent √**

4

```
public int getBooleanValue() {
    String value = super.getValue();
    try {
        return Integer.parseInt(value);
    } catch (NumberFormatException e) {
        // TODO: validation handling/logging
        throw (e);
    }
}
```

Ground Truth: Inconsistent
MNire: Consistent (MNR result: getValue)
AUMENA*: Consistent (MNR result: getIntValue)
AUMENA: **Inconsistent √**

Fig. 4. Some examples from MCC testset

Quality and Length Analysis

AUMENA could generate method names with similar or even higher quality compared to human-written ones from the perspective of method naming standards.

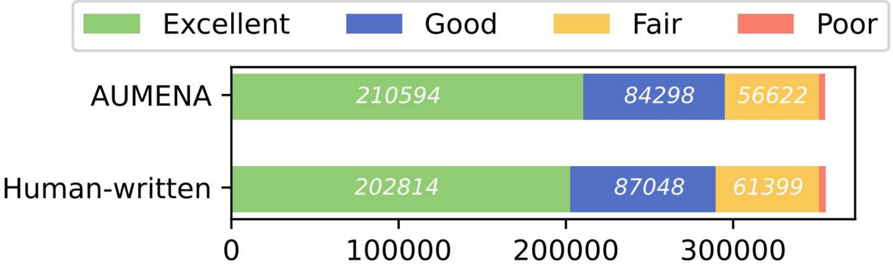


Fig. 5. Results of Naming Quality Analysis

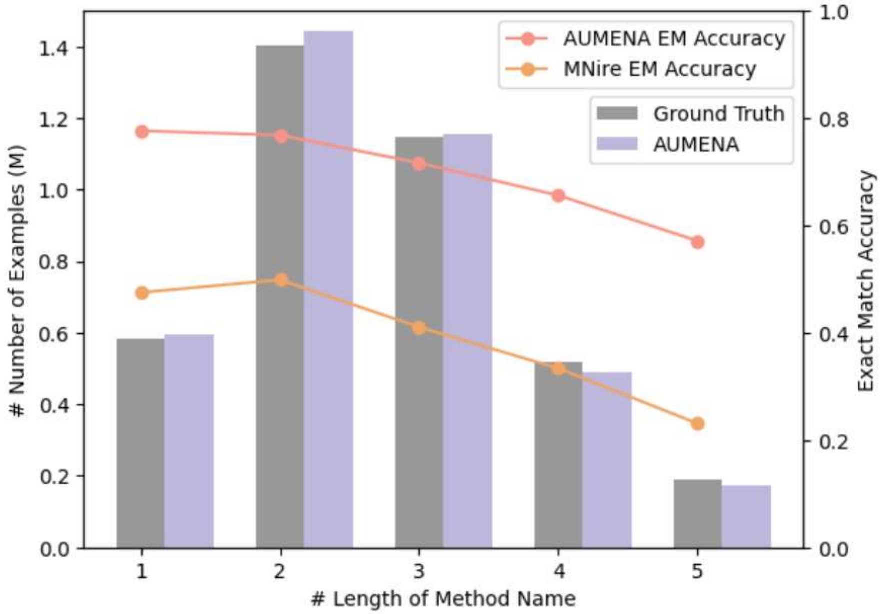


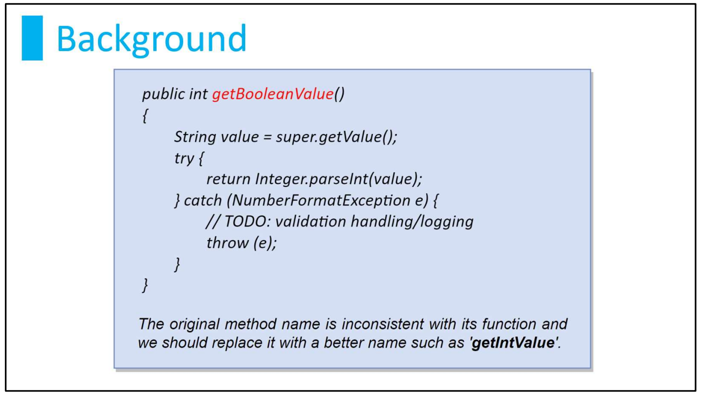
Fig. 6. The method name length distribution and the exact match accuracy with different lengths

[1] S. Nguyen, H. Phan, T. Le, and T. N. Nguyen, "Suggesting natural method names to check name consistencies," in ICSE '20: 42nd International Conference on Software Engineering

[4] R. S. Alsuhaibani, C. D. Newman, M. J. Decker, M. L. Collard, and J. I. Maletic, "An approach to automatically assess method names," in Proceedings of the 30th IEEE/ACM International Conference on Program Comprehension, ICPC 2022, Virtual Event, May 16-17, 2022

Conclusion

Method names provides significant information for program comprehension.

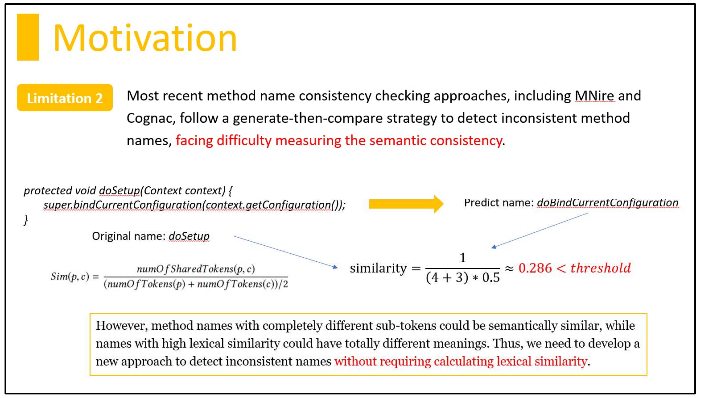


Background

```
public int getBooleanValue()
{
      String value = super.getValue();
      try {
            return Integer.parseInt(value);
      } catch (NumberFormatException e) {
            // TODO: validation handling/logging
            throw (e);
      }
}
```

*The original method name is inconsistent with its function and we should replace it with a better name such as '**getIntValue**'.*

Current approaches face difficulty in training sufficiently and measuring semantic consistency.



Motivation

Limitation 2 Most recent method name consistency checking approaches, including MNire and Cognac, follow a generate-then-compare strategy to detect inconsistent method names, facing difficulty measuring the semantic consistency.

```
protected void doSetup(Context context) {
    super.bindCurrentConfiguration(context.getConfiguration());
}
```

Predict name: *doBindCurrentConfiguration*

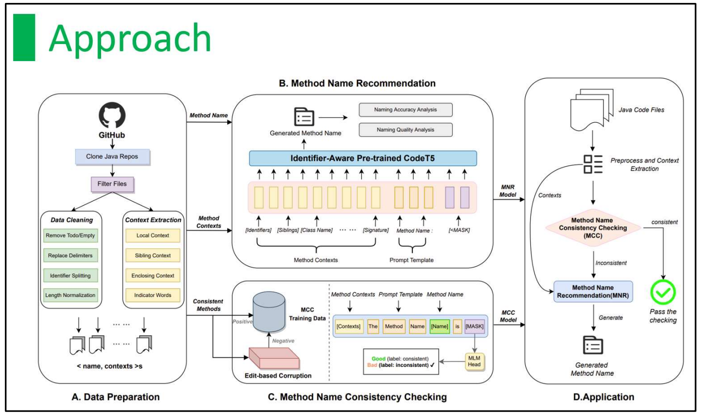
Original name: *doSetup*

$$Sim(p,c) = \frac{numOfSharedTokens(p,c)}{(numOfTokens(p) + numOfTokens(c))/2}$$

$$\text{similarity} = \frac{1}{(4+3)*0.5} \approx 0.286 < threshold$$

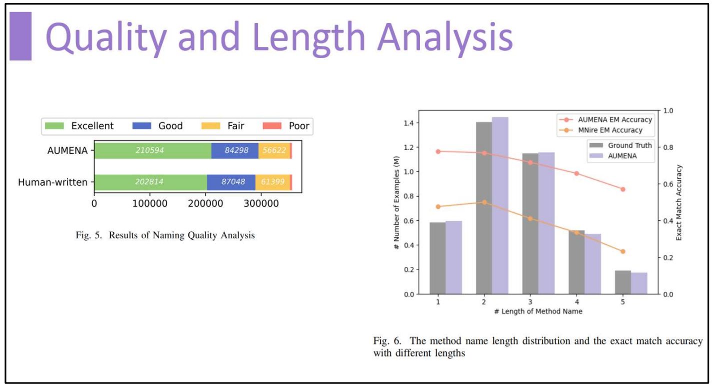
However, method names with completely different sub-tokens could be semantically similar, while names with high lexical similarity could have totally different meanings. Thus, we need to develop a new approach to detect inconsistent names without requiring calculating lexical similarity.

Leveraging hard negative mining and prompt tuning with T5.



Approach

Outperform all baselines in metrics and also perform well on naming standards.



Quality and Length Analysis

Fig. 5. Results of Naming Quality Analysis

Fig. 6. The method name length distribution and the exact match accuracy with different lengths

Thanks!

Q&A

ISCAS

中国科学院大学
University of Chinese Academy of Sciences

中科软科技
Sinosoft Co.,Ltd.