

面向实用的缺陷导向自动化代码审查

陆俊逸, 姜丽莉, 李晓佳, 方剑冰, 张凤军,
杨立*, 左春

The 42nd International Conference on Machine Learning
(ICML 2025), 2025

Tel: 杨立 yangli2017@iscas.ac.cn 010-62661198

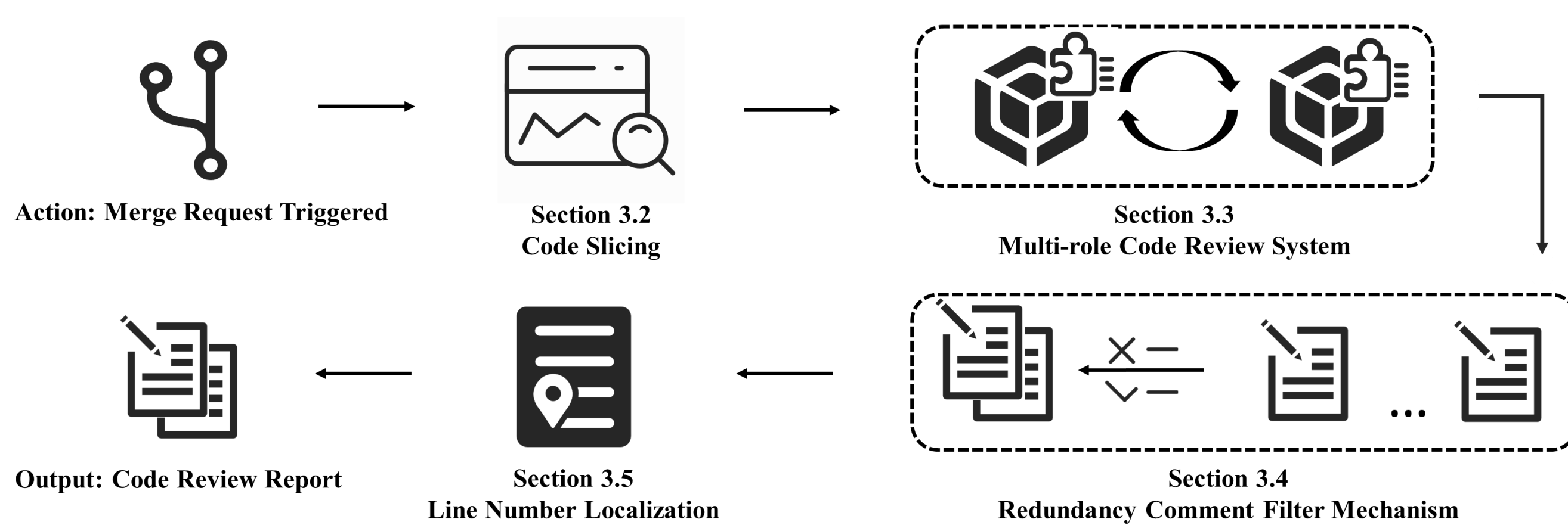
Motivation

- Prior work oversimplifies code review as snippet-level text generation
- 30% of severe incidents stem from preventable code defects
- Text similarity metrics (BLEU) fail to measure defect detection
- **Real-world** integration requires repository-level context

Key Challenges

- **Context Capture:**
 - Balancing comprehensive context with LLM limitations
- **Bug Detection:**
 - Improving Key Bug Inclusion (KBI) rate
- **False Alarms:**
 - Reducing nitpicks and hallucinations
- **Integration:**
 - Line-level comment localization

Framework Architecture



Core Components:

1. Code Slicing Algorithms

- Original Diff
- Parent Function
- Left Flow (L-value tracking)
- Full Flow (L+R values)

2. Multi-Role System

- Reviewer (analyzes code)
- Meta-Reviewer (merges)
- Validator (refines)
- Translator (localizes)

3. Comment Filtering (Q1-Q3 Scoring)

Q1: Is it a nitpick? | Q2: Is it a real problem? | Q3: How critical?

Performance Comparison

Model	KBI ↑	FAR ₁ ↓	CPI ₁ ↑
CodeReviewer	0.00	97.78	0.00
CCT5	2.22	97.58	2.32
LLaMA-Reviewer	2.22	97.62	2.30
Ours (LLaMA3.1-405B)	20.00	75.37	22.07

10× improvement in comprehensive performance

Contributions

- First repository-level, MR-granularity approach
- Real-world DevOps integration
- Validation on industry-scale defects
- Specialized LLM framework for code review

Key Findings

- **Flow-based slicing** (Left/Full) outperforms simpler methods
- **Multi-reviewer setup** improves KBI but increases FAR
- **Validators are essential** for balancing precision and recall
- **CoT prompting** is most effective for complex contexts
- **Inline line numbers** are crucial for practical deployment