OrcaLoca: An LLM Agent Framework for Software Issue Localization

Zhongming Yu[*,1] ,Hejia Zhang[*,1], Yujie Zhao[1], Hanxian Huang[1], Matrix Yao[2], Ke Ding[2], Jishen Zhao[1]

* Equal contribution, [1]Uiversity of California, San Diego, [2]Intel Corporation

UC San Diego

Background

- Auto Code Editor has overwhelmed developers' lives
  - Copilot
  - Cursor
  - CodeX
  - …

15M+ users

$9B+ valuation

The default code agent in ChatGPT

Basic Workflow in SWE-agent

- Three major parts[1]



1. *Localization*: Identify file(s)/line(s) causing the issue.
2. *Editing*: Generate fixes addressing the given issue.
3. *Testing*: Write new scripts or modify existing test files to reproduce the issue and/or verify if fixes are correct.

[1] **Swe-agent**: **Agent**-computer interfaces enable automated software engineering

*figure used from flaticon

The importance of Localization Stage

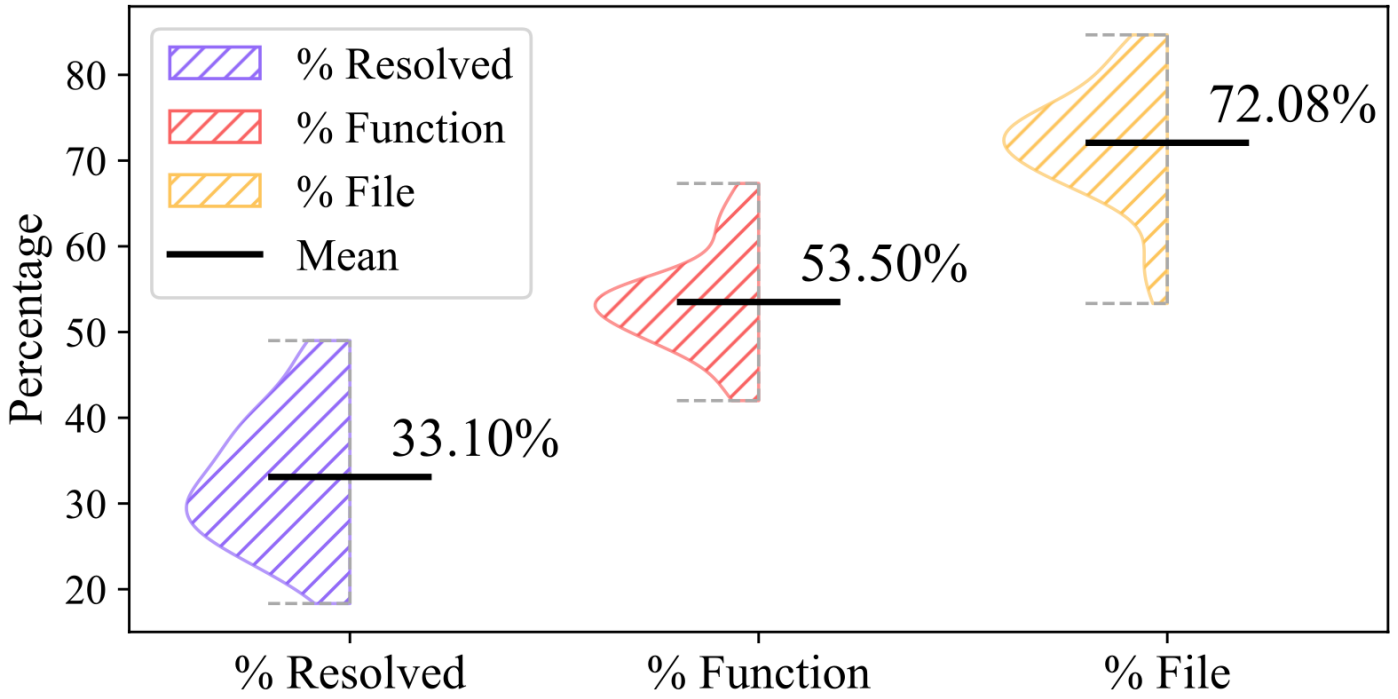
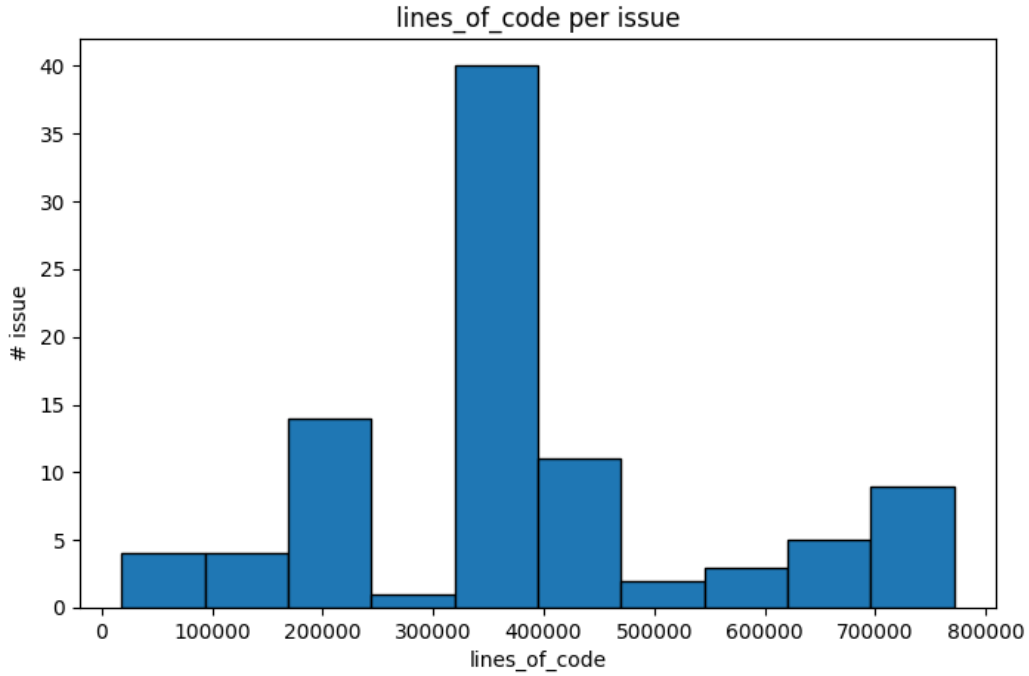


Figure 1. Distribution and average of file / function match rate and resolved rate on SWE-Bench Lite LeaderBoard. [1]

[1] **Swe-bench**: Can language models resolve real-world github issues?

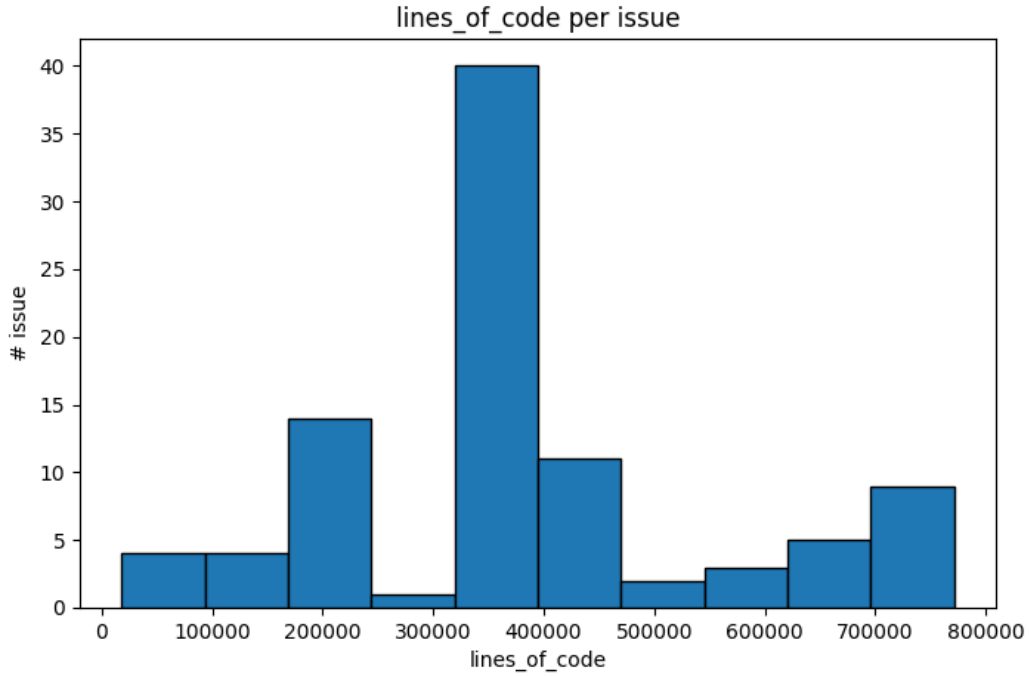
Why Bug Localization is a Hard Problem

- Large Searching Space
  - Hard to locate specific lines of code



Why Bug Localization is a Hard Problem

- Large Searching Space
  - Hard to locate specific lines of code



- Implicit Bug Issue
  - Natural Language Input
  - Implicit Bugs

Issue

*I found a bug in Django... Given the following contents of models.py ... migrations.**CreateModel** ...Missing import statement in generated migration...I think this is a bug of the module **django.db.migrations.writer**, but I'm not sure. ...*

Bug Report

*After examining the serialization process, the bug is in **TypeSerializer.serialize()** method in **django/db/migrations/serializer.py**. The special case [(models.Model, 'models.Model', [])] explicitly sets an empty import list for models.Model, which causes the missing import statement in the generated migration file...*

TypeSerializer.serialize

The real bug is far away from the original locations where the issue is mentioned

Localization Strategy

- Sparse embedding
  - BM25 index
- Dense embedding
  - Code Embedding
- Agent search
  - SWE-Agent
  - Agentless[1]
  - RepoUnderstander[2]

[1] **Agentless**: Demystifying llm-based software engineering agents
[2] Alibaba LingmaAgent: Improving Automated Issue Resolution via Comprehensive Repository Exploration

Challenges for Designing Localization Agent

- How to guide LLM actions to navigate the codebase precisely and efficiently?
- How may search space completeness and context conciseness be combined?
- How to effectively manage the growing context during exploration?

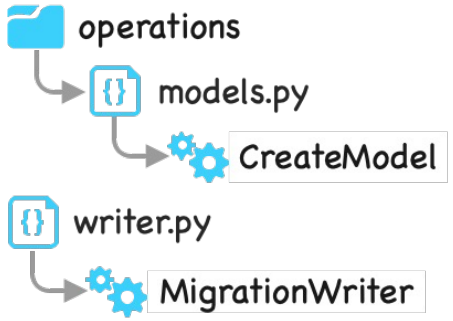
Challenge 1

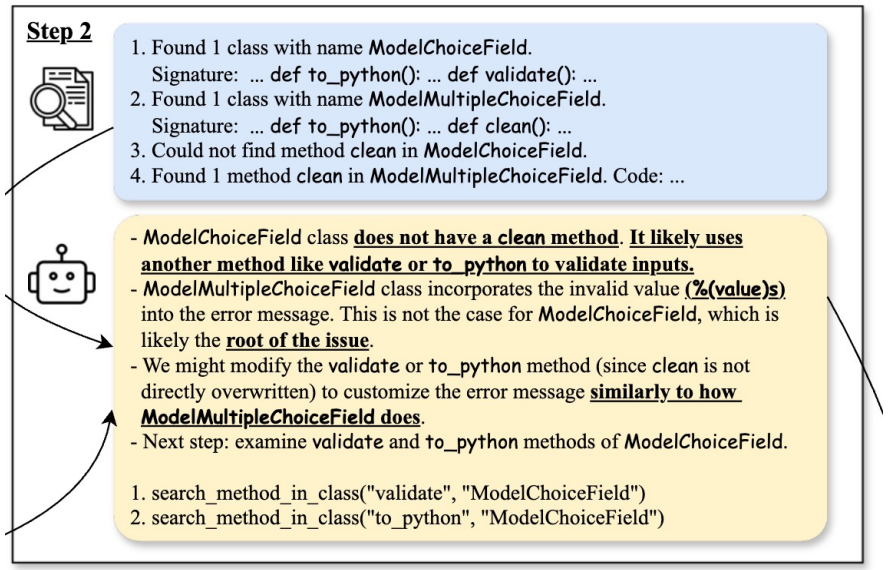
- How to guide LLM actions to navigate the codebase precisely and efficiently?
  - LLM may not know which action should be executed first, given the relevant code

Issue

*I found a bug in Django... Given the following contents of models.py ... migrations.**CreateModel** ...Missing import statement in generated migration...I think this is a bug of the module **django.db.migrations.writer**, but I'm not sure. ...*

Multiple actions will be generated during exploration





Previous solutions like AutoCodeRover[1] execute all actions in a single step, which can lead to imprecise planning due to content overload at each stage.

[1] **Autocoderover**: Autonomous program improvement

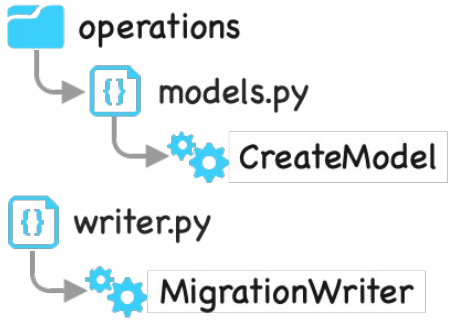
Challenge 1

- How to guide LLM actions to navigate the codebase precisely and efficiently?
  - LLM may not know which action should be executed first, given the relevant code
  - LLM may have hallucinations for search actions during exploration

Issue

*I found a bug in Django... Given the following contents of models.py ... migrations.**CreateModel** ...Missing import statement in generated migration...I think this is a bug of the module **django.db.migrations.writer**, but I'm not sure. ...*

Multiple actions will be generated during exploration



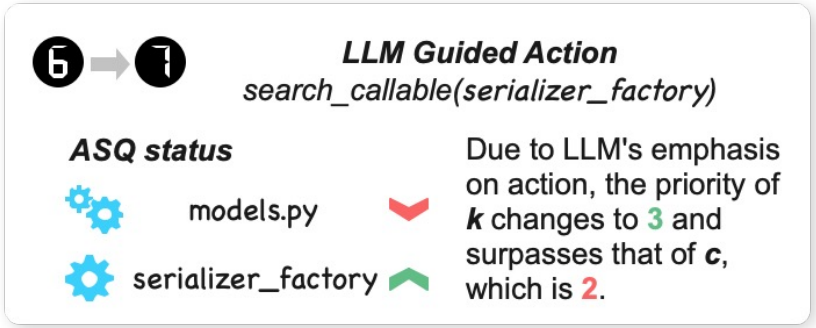
Action 1
search_class(ModelChoiceField)

Action 2
search_class_in_file (ModelChoiceField, django/forms/model.py)

LLM may don't know this action points to the same location as before

Solution 1

- Priority-Based Scheduling for LLM-Guided Actions



while ASQ not empty and not converged **do**
  Generate $O_t, PB_t, SA_t \leftarrow \text{LLM}(s_t)$
  for all $a_k \in SA_t$ **do**
    if a_k is redundant **then**
      Skip a_k
    else if a_k previously seen **then**
      Increment counter C_{a_k} and update priority
    else
      Add a_k to ASQ
    end if
  end for
  Select top-priority a_t from ASQ
  Execute a_t to get SR_t

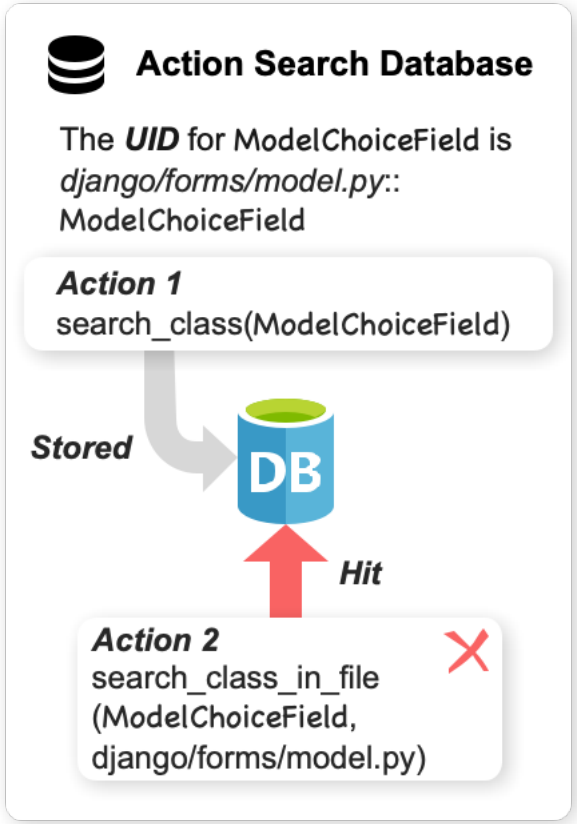
The key point here is constructing a priority queue for managing search actions.

The weight can be defined by configuration and can increase when the LLM proposes that action again during exploration.

(Here we set LLM to only take 1 action per step. So the LLM may propose actions that are in the queue but have not been executed. We leverage this attribute for designing the weight mechanism during dynamic exploration.)

Solution 1

- Search content prefetch-based checking



while ASQ not empty and not converged **do**
  Generate $O_t, PB_t, SA_t \leftarrow \text{LLM}(s_t)$
  for all $a_k \in SA_t$ **do**
    if a_k is redundant **then**
      Skip a_k
    else if a_k previously seen **then**
      Increment counter C_{a_k} and update priority
    else
      Add a_k to ASQ
    end if
  end for
  Select top-priority a_t from ASQ
  Execute a_t to get SR_t

During search content execution, the agent will prefetch the content to check whether it has already been explored.

Challenge 2

- How to achieve both search space completeness and conciseness

(1) File Skeleton Prompt, e.g. Agentless

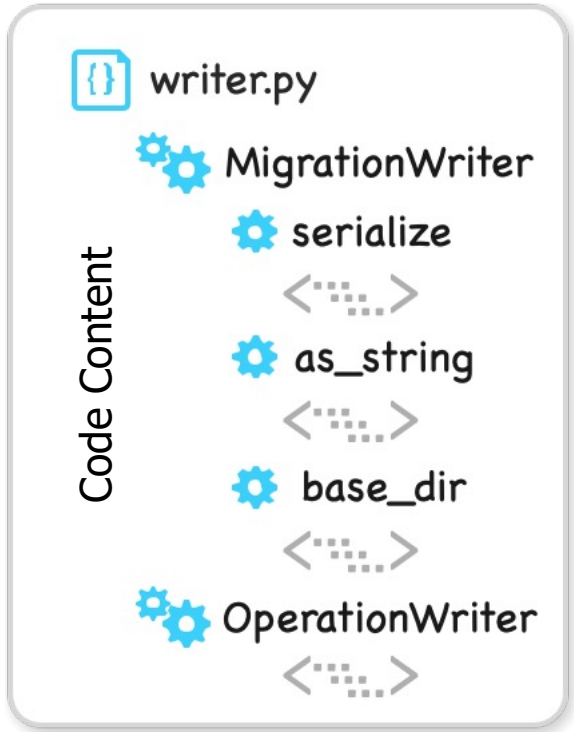
✗ Lack of complete code content info

Challenge 2

- How to achieve both search space completeness and conciseness

(1) File Skeleton Prompt, e.g. Agentless

✗ Lack of complete code content info

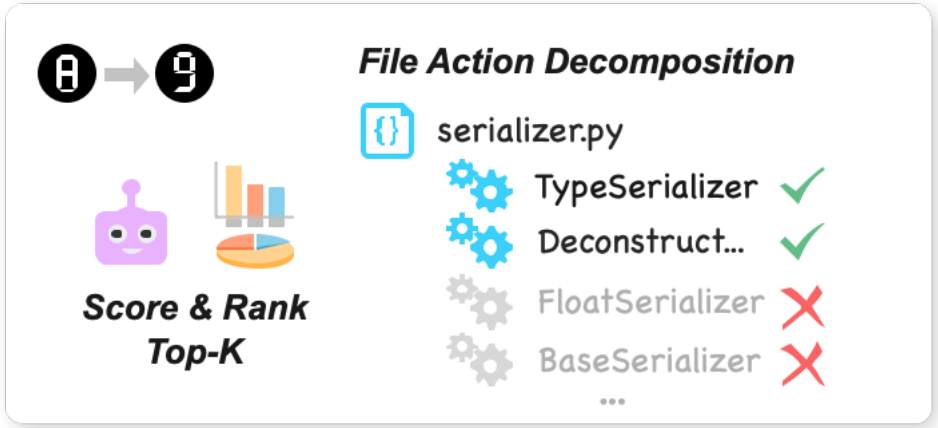


(2) Whole file Content

✗ May introduce irrelevant code info when the file is large

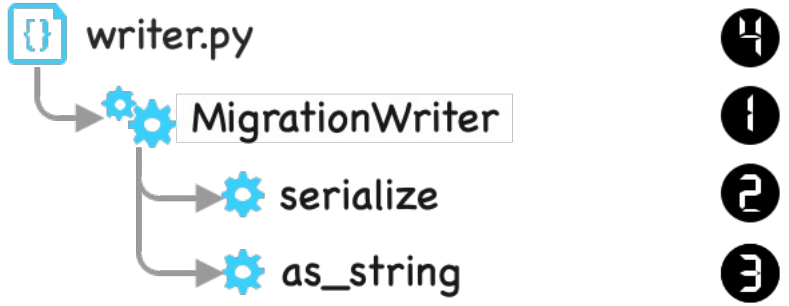
Solution 2

- Action Decomposition with Relevance Scoring



Sub-agent ranking

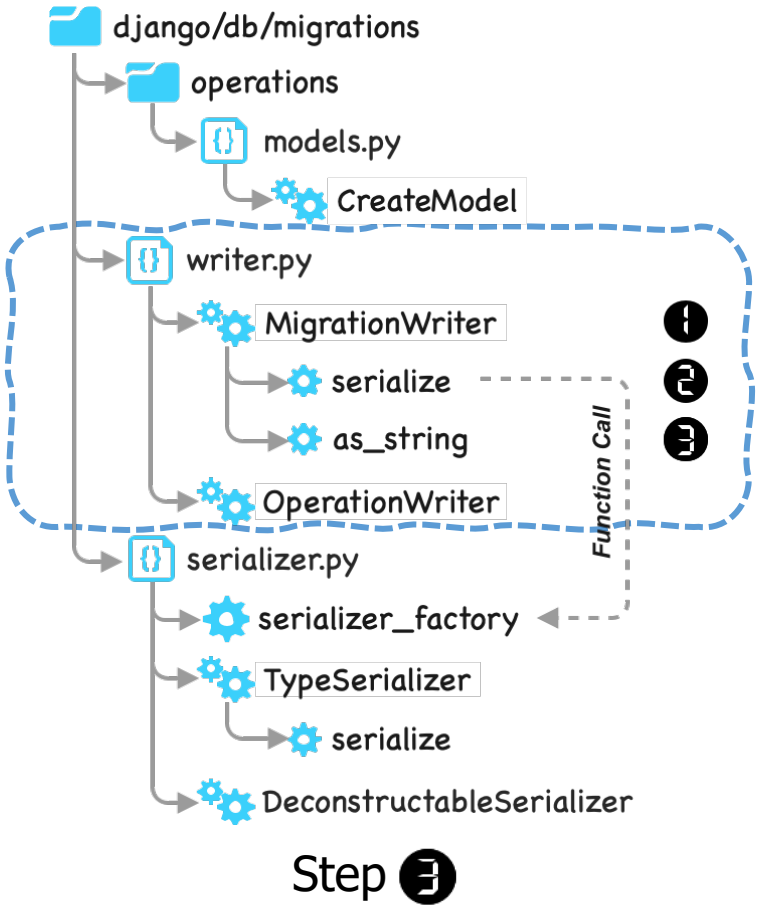
Get relevant top candidates after pruning out low-score code contents.



The main agent will focus on the most related content during exploration

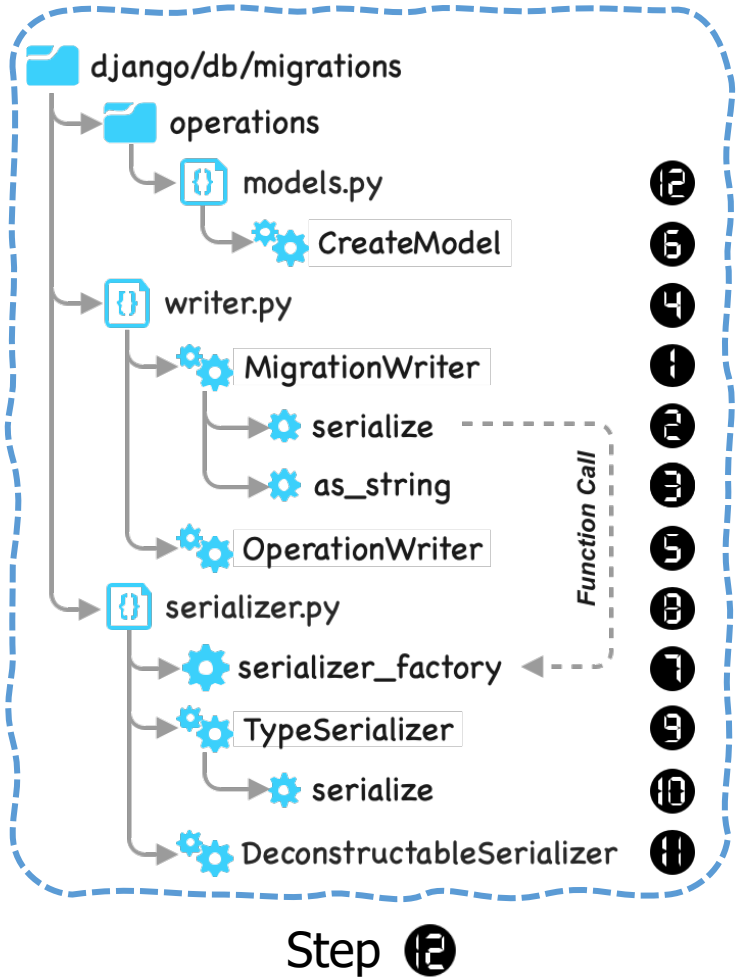
Challenge 3

- How to do content management during exploration



Step 3

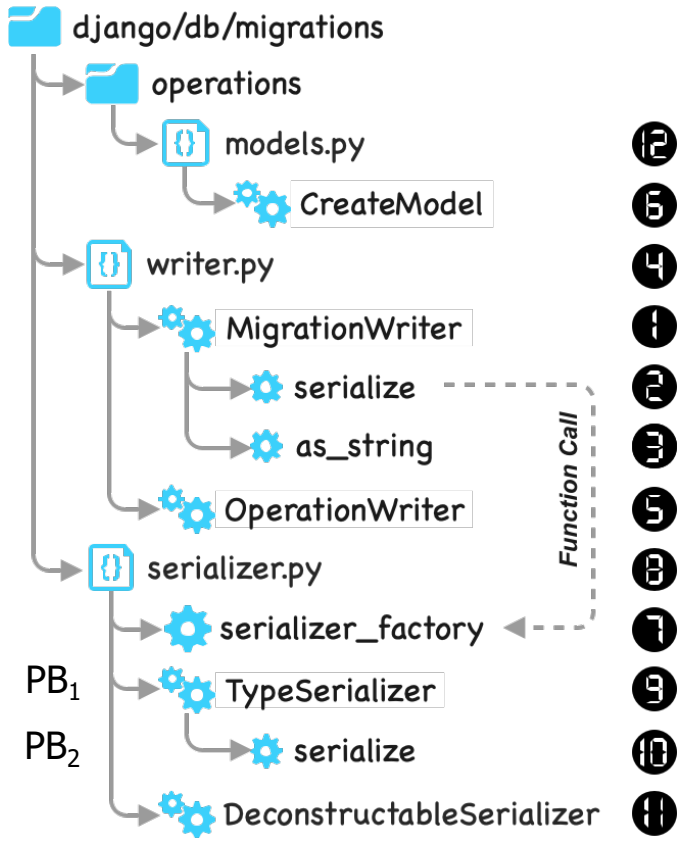
Content continues growing



Step 12

Solution 3

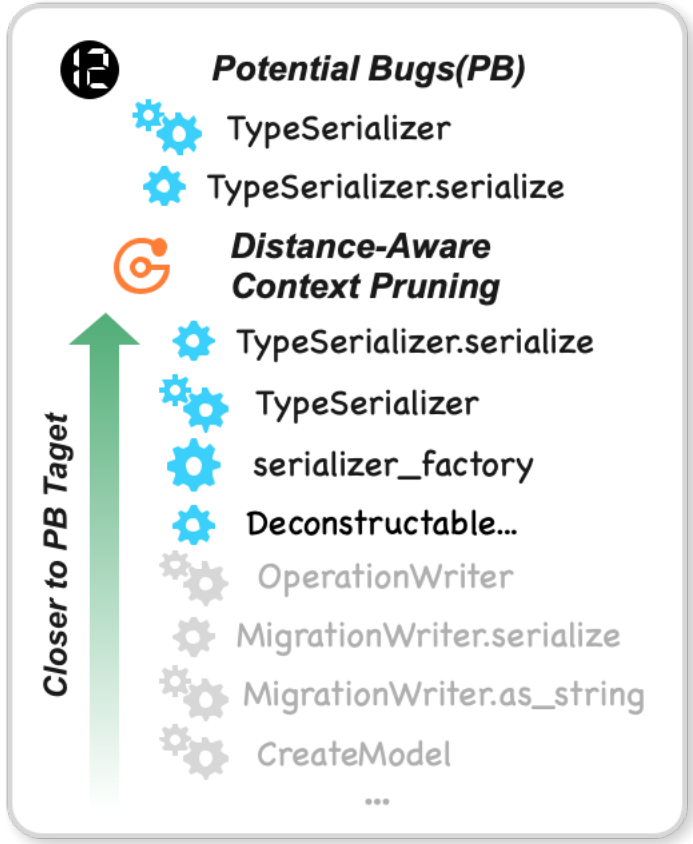
- Distance-Aware Searched Context Pruning



$$\frac{1}{|PB|}\sum_{v\in PB}\min\left(d(v_{SR},v),d(v,v_{SR})\right)$$

Calculate the average distance for given SR: e.g. OperationWriter

Then, rank all the candidates by using the distance heuristic



Experiment Results

LLM Agent	LLM	Resolved Rate (Count)	Resolved Rank	Function Match Rate (Count)	Function Match Rank	File Match Rate (Count)	File Match Rank
Blackbox AI	N/A	**49.00% (147)**	**1**	63.33% (190)	5	81.33% (244)	6
Gru (2024-12-08)	N/A	48.67% (146)	2	61.67% (185)	6	83.33% (250)	3*
Globant Code Fixer	N/A	48.33% (145)	3	**67.33% (202)**	**1**	84.00% (252)	2
devlo	N/A	47.33% (142)	4	66.67% (200)	2	**84.67% (254)**	**1**
OpenCSG Starship	GPT-4o	39.67% (119)	10	49.00% (147)	17	70.67% (212)	16
Bytedance MarsCode	N/A	39.33% (118)	11	56.33% (169)	13	79.67% (239)	7*
Alibaba Lingma	N/A	33.00% (99)	15	57.33% (172)	11	75.00% (225)	13
Kodu-v1	Claude 3.5 Sonnet	44.67% (134)	5 ★	52.00% (156)	15	65.00% (195)	19
OpenHands + CodeAct v2.1	Claude 3.5 Sonnet	41.67% (125)	6	63.67% (191)	4	81.67% (245)	5
PatchKitty-0.9	Claude 3.5 Sonnet	41.33% (124)	7	59.67% (179)	8	75.33% (226)	12
Composio SWE-Kit	Claude 3.5 Sonnet + o1-mini	41.00% (123)	8*	61.00% (183)	7	79.67% (239)	7*
Moatless Tools	Claude 3.5 Sonnet	39.00% (117)	12	59.33% (178)	9	79.33% (238)	9
	DeepSeek V3	30.67% (92)	16	54.33% (163)	14	74.33% (223)	14
AutoCodeRover-v2.0†	GPT-4o	37.33% (112)	13	57.00% (171)	12	77.67% (233)	11
Agentless-1.5‡	Claude 3.5 Sonnet	34.67% (104)	14	58.67% (176)	10	78.67% (236)	10
RepoGraph	GPT-4o	29.67% (89)	17	47.67% (143)	18*	70.33% (211)	17
HyperAgent	Claude 3.5 Sonnet	25.33% (76)	18	47.67% (143)	18*	67.67% (203)	18
SWE-agent	Claude 3.5 Sonnet	23.00% (69)	19	51.67% (155)	16	71.67% (215)	15
	GPT-4o	18.33% (55)	20	42.00% (126)	21	57.67% (173)	21
	GPT-4	18.00% (54)	21	43.67% (131)	20	61.00% (183)	20
	Claude 3 Opus	11.67% (35)	22	33.67% (101)	22	47.67% (143)	22
OrcaLoca	Claude 3.5 Sonnet	41.00% (123)	8*	65.33% (196)	3 ★	83.33% (250)	3* ★

Experiment Results

Table 2. Impact of localization on resolved rate. UL stands for Union of Locations; ML stands for Mean of Locations.

Agent	% Resolved	Function Match Rate	Function Match Precision
OrcaLoca	**41.00%**	**65.33%**	**38.34%**
Agentless (UL)	34.67%	58.67%	29.01%
Agentless (ML)		47.33%	33.72%

Table 3. Ablation study results. Experiment completed on SWE-bench Common dataset.

Methods	Func. Match Rate
ORCALOCA	**76.34%** (71)
- w/o. priority scheduling	73.12% (68)
- w/o. file & class decom.	72.04% (67)
- w/o. disambiguation decom.	70.97% (66)
- w/o. context pruning	72.04% (67)

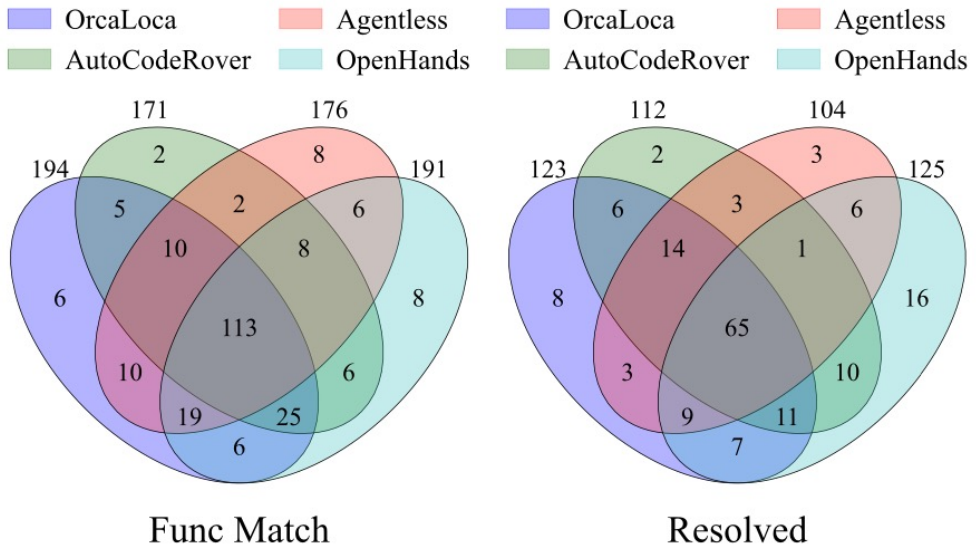


Figure 4. Unique localizations and solutions of open source agents.

Discussion

- Top-K mode support for retrieval
- Parallel batch actions in each step
- Extension for multi-language support in the future
- Extension for different model support

Conclusion

- We design OrcaLoca, an agent framework for software issue localization
  - Priority-Based Scheduling for LLM-Guided Actions
  - Action Decomposition with Relevance Scoring
  - Distance-Aware Searched Context Pruning
  - 6.33 percentage points increase

Repo QR code