PT Jurnal Cendekia Indonesia
Journal of Information Systems and Business Technology
Homepage: https://journal.jci.co.id/jisbt
Vol. 02 No. 04 (2026) Page: 101-110
E-ISSN: 3109-8886

Retrieval-Augmented Triage for Software Engineering Agents: Source-Path and Pre-Fix Hunk Localization with Calibrated Repair-Effort Prediction on SWE-bench Verified

[1]Eric Sullivan, [2]Fang Wei, [3]Chloe Bennett, [4]Jun Ma

[1]Department of Computer Sciences, Northeastern University, Boston, MA, USA
[2]Department of Computer Science, Virginia Tech, Blacksburg, VA, USA
[3]Department of Computer Science, University of Arizona, Tucson, AZ, USA
[4]Department of Computer Science, University of Pittsburgh, Pittsburgh, PA, USA

Email: [1]e.sullivan_dev@gmail.com

Abstract

Repository-scale software agents need constrained context before reasoning about a defect. This study evaluates retrieval and triage on all 500 human-validated SWE-bench Verified tasks from 12 Python repositories. Unified-diff parsing recovered 373 modified source paths and 1,220 source hunks. In the primary issue-only closed-world experiment, a leave-one-repository-out reranker achieved 0.500 Hit@1, 0.747 Recall@5, and 0.625 MRR; its MRR gain over character TF-IDF was 0.065 (95% CI 0.046-0.087; adjusted $p < 0.001$) and remained significant with equal repository weighting. Failing-test identifiers increased post-failure RRF MRR from 0.590 to 0.685, while issue-only word TF-IDF reached 0.684 MRR on gold-selected pre-fix hunks. Nested cross-repository calibration yielded 0.139 PR-AUC, 0.632 ROC-AUC, 0.082 Brier score, and 0.028 ECE for high-effort triage. Lexical localization was effective, but pre-execution effort forecasting remained limited; retrieval evidence, escalation, and executable repair should therefore remain separate stages.

Keywords Software Engineering Agents, SWE-Bench Verified, Bug Localization, Information Retrieval, Reciprocal Rank Fusion, Learning To Rank, Probability Calibration, Selective Triage.

Abstrak

Agen perangkat lunak berskala repositori memerlukan konteks yang dibatasi sebelum melakukan penalaran mengenai suatu cacat (bug). Studi ini mengevaluasi proses temu kembali (retrieval) dan triase pada seluruh 500 tugas SWE-bench Verified yang telah divalidasi manusia dari 12 repositori Python. Penguraian unified-diff berhasil mengidentifikasi 373 jalur sumber yang dimodifikasi dan 1.220 hunk kode sumber. Dalam eksperimen utama closed-world yang hanya menggunakan informasi issue, metode reranker dengan pendekatan leave-one-repository-out mencapai skor Hit@1 sebesar 0,500, Recall@5 sebesar 0,747, dan MRR sebesar 0,625; peningkatan MRR dibandingkan metode TF-IDF karakter adalah 0,065 (CI 95% 0,046–0,087; p yang disesuaikan < 0,001) dan tetap signifikan dengan pembobotan repositori yang setara. Penggunaan pengenal uji yang gagal (failing-test identifiers) meningkatkan MRR RRF pasca-kegagalan dari 0,590 menjadi 0,685, sementara metode TF-IDF kata (khusus issue) mencapai MRR 0,684 pada hunk pra-perbaikan yang dipilih secara ideal (gold-selected). Kalibrasi lintas-repositori bertingkat menghasilkan skor PR-AUC 0,139, ROC-AUC 0,632, Brier 0,082, dan ECE 0,028 untuk triase dengan upaya tinggi. Lokalisasi leksikal terbukti efektif, namun prakiraan upaya pra-eksekusi masih terbatas; oleh karena itu, tahap temu kembali bukti, eskalasi, dan perbaikan yang dapat dieksekusi sebaiknya tetap dipisahkan.

Kata Kunci: Agen Rekayasa Perangkat Lunak, SWE-Bench Verified, Lokalisasi Bug, Temu Kembali Informasi, Reciprocal Rank Fusion, Learning To Rank, Kalibrasi Probabilitas, Triase Selektif.

A. INTRODUCTION

Repository-level issue resolution combines retrieval, program understanding, editing, and testing. SWE-bench contains 2,294 issue-pull-request pairs [1], and SWE-bench Verified retains 500 human-checked tasks [2]. Although flawed tests and contamination ended its use for frontier capability tracking [3], the fixed corpus remains suitable for transparent artifact analysis. This paper evaluates retrieval and pre-repair triage, not frontier-model capability.

Repository navigation is central to end-to-end repair. SWE-agent couples search, editing, and tests [4]; Agentless separates localization, repair, and validation [5]; and AutoCodeRover uses program-aware search [6]. RepoCoder alternates retrieval and generation [7], while data-flow retrieval adds structural evidence [8]. These systems motivate a controlled question: how much repair scope can transparent retrieval recover before generation? A CI-oriented pipeline likewise separates failure

Diterima Redaksi: xx-xx-2022 | Selesai Revisi: xx-xx-2022 | Diterbitkan Online: xx-xx-2022

classification, repair retrieval, and patch recommendation [9], although its evaluation did not use the original CI benchmark. A deterministic DevOps study further separates retrieval quality from command safety [10].

Semantic code search supplies the broader foundation. CodeSearchNet established natural-language-to-code retrieval [11]; CodeBERT [12] and GraphCodeBERT [13] connect language with program tokens; and RAG conditions generation on external evidence [14]. To avoid pretrained-corpus and compute confounds, this study uses deterministic local lexical models. A lightweight CodeSearchNet study similarly framed retrieval-conditioned summaries as a find-then-explain workflow [15]. Outside code, an OMoS-QA evaluation found that word-character lexical fusion can improve page coverage while keeping answerability and abstention as separate decisions [16].

Operational RAG also requires traceability. DevOps studies distinguish recall from citation precision [17] and connect retrieved alerts to incident hypotheses [18]. Evidence-chain methods localize supporting text [19], [20], while calibrated evidence sufficiency supports abstention [21]. Cross-domain work on review-grounded recommendation likewise evaluates ranking quality separately from whether extracted evidence faithfully supports the score [22]. Accordingly, a ranked path or hunk is inspectable evidence, not proof of repair correctness.

Issue-to-file retrieval predates language-model agents. BugLocator uses report similarity and fix history [23], BLUiR indexes structured report and code fields [24], and Bench4BL documents strong project effects [25]. Ranking features can predict localization success [26], and supervised lexical-project evidence improves file ranking [27]. Retrieval policies also trade cost for coverage [28], and ranking reliability can be predicted [29]. Spectral rank aggregation has pairwise-comparison guarantees [30], but RRF better matches the heterogeneous partial rankings used here. A FiQA comparison further found that a rewrite-hybrid-listwise stack underperformed BM25, supporting strong lexical candidate pools and gated reranking rather than assumed complexity gains [31].

Four questions guide the study: RQ1 characterizes task and repair distributions; RQ2 compares source-path rankers; RQ3 isolates comments and failing-test identifiers and evaluates pre-fix hunk retrieval; and RQ4 tests cross-repository patch-scope and effort triage. Figure 1 summarizes the workflow.

The contributions are: auditable file-, hunk-, and line-level parsing of all 500 accepted patches; a common-inventory comparison of transparent rankers under distinct evidence regimes; and repository-held-out scope and effort triage with nested calibration and escalation operating points. Localization scores are never treated as evidence of functional correctness.

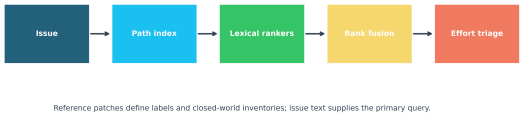

Figure 1 Retrieval and triage pipeline. Reference patches define labels and closed-world inventories; issue text supplies the primary query.

B. METHODS

Dataset And Analytical Scope

The SWE-bench Verified Parquet distribution contains 500 unique instances and 13 string-valued fields. Each record identifies a repository and base commit, provides an issue statement and optional pre-solution issue comments, stores the accepted developer source patch and regression-test patch, lists FAIL_TO_PASS and PASS_TO_PASS test identifiers, records environment and version metadata, and includes an annotator-estimated repair-effort category. Table 1 distinguishes issue-time inputs from evaluation-only labels. For path localization and effort prediction, accepted patches defined evaluation labels and the reference-derived closed-world path inventories; source-code additions and removals did not enter the queries or numeric effort features. For scope prediction, patch counts served only as targets. In the explicitly oracle hunk experiment, old-side context from accepted patches defined the candidate documents, while added solution lines were excluded. Failing-test identifiers were also excluded from the primary agent setting because the official benchmark hides tests from the agent; they were introduced only in the post-failure diagnostic ablation.

Table 1 Dataset Fields, Analytical Roles, And Availability Controls.

Field	Analytical role	Availability
repo	Repository grouping and candidate-pool boundary	Issue time
instance_id	Task key	Issue time
base_commit	Benchmark revision identity	Issue time
patch	Labels, closed-world inventories, and outcome statistics	Evaluation only
test_patch	Reference regression-test change statistics	Evaluation only
problem_statement	Primary retrieval and effort-model input	Issue time
hints_text	Pre-solution issue-comment ablation	Ablation only
created_at	Solution pull request creation date	Evaluation only
version	Evaluation installation version	Evaluation only
FAIL_TO_PASS	Post-failure identifier-evidence ablation	Diagnostic only

Field	Analytical role	Availability
PASS_TO_PASS	Test-suite scale description	Evaluation only
environment_setup_commit	Environment identity	Evaluation only
difficulty	Annotator-estimated effort target	Evaluation only

The corpus covers 12 repositories. Django contributes 231 instances and 123 distinct source-path candidates, whereas Flask contributes one instance and one candidate. In the analyzed Parquet snapshot, difficulty labels comprise 194 tasks below 15 minutes, 261 from 15 to 60 minutes, 42 from one to four hours, and three above four hours. The high-effort target combined the final two categories, yielding 45 positives (9.0%). Table 2 reports repository counts and Figure 2 displays the imbalance directly.

Table 2 Repository Distribution, Closed-World Source-Path Inventory, Hunks, High-Effort Tasks, And Multi-File Repairs.

Repository	N	Paths	Hunks	High effort	Multi-file
django/django	231	123	475	22	32
sympy/sympy	75	87	267	7	8
sphinx-doc/sphinx	44	29	108	5	8
matplotlib/matplotlib	34	27	71	0	4
scikit-learn/scikit-learn	32	30	95	1	2
astropy/astropy	22	24	50	3	3
pydata/xarray	22	13	52	2	5
pytest-dev/pytest	19	13	46	3	2
pylint-dev/pylint	10	19	37	2	6
psf/requests	8	4	12	0	0
mwaskom/seaborn	2	3	6	0	1
pallets/flask	1	1	1	0	0

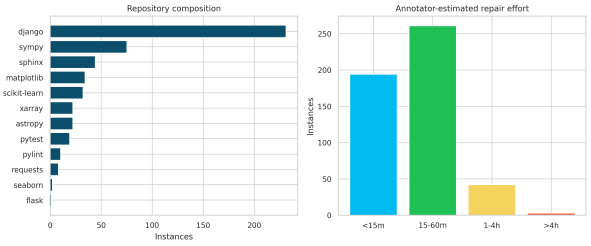

Figure 2 Repository Composition And Annotator-Estimated Repair-Effort Distribution.

The compact distribution does not contain repository source trees, CI logs, rejected candidate patches, agent trajectories, or execution outcomes. Consequently, the experiments evaluate benchmark-local closed-world paths, a gold-selected pre-fix hunk pool, patch-plan attributes, and human effort. They do not report repository-wide fault localization, test passage, issue resolution, or semantic patch correctness. This boundary is essential because a set of accepted reference patches has no negative patch class from which a correctness predictor can learn.

The instance, rather than an individual file or hunk, was the unit of evaluation. Multi-file tasks therefore contributed one task-level recall value instead of one observation per modified path. This prevents broad patches from dominating aggregate localization scores. Repository identifiers defined all outer train-test splits, so examples from the held-out project never entered supervised reranker, patch-plan, or effort-model fitting. Descriptive counts still cover the full snapshot, and the closed-world inventories intentionally use the set of reference-modified paths within each repository. The latter choice guarantees evaluable positives while making the reported localization problem narrower than repository-wide search.

Diff parsing and labels

A deterministic unified-diff parser read each diff --git header, file path, hunk header, context line, removed line, and added line. Source-patch labels were the complete set of developer-modified paths per instance. Hunk labels were the corresponding instance-path-hunk identifiers. Candidate hunk text contained the path, hunk header, unchanged context, and removed pre-fix lines; all added lines were excluded. The pool still consists only of gold-selected hunks, so hunk retrieval is an oracle candidate-set analysis rather than a scan of every repository line.

The parser recovered 623 source-file edits and 1,220 hunks. It counted 4,972 added and 2,195 removed source lines. Test-patch parsing recovered 664 file edits and 1,095 hunks. JSON decoding converted FAIL_TO_PASS and PASS_TO_PASS to test-identifier lists. Descriptive statistics in Table 3 show strong right skew: the median source patch changes seven lines, but the maximum changes 232; the median task has one failing-test identifier, but one task has 438. Figure 3 shows the same distributions without allowing the extreme values to determine the visual scale.

Table 3 Descriptive Statistics For Issues, Reference Source Patches, Regression-Test Patches, And Test Identifiers.

Variable	Mean	Median	Q1	Q3	Max
Issue tokens	258.822	181.000	98.000	311.500	4167
Source files	1.246	1.000	1.000	1.000	21
Source hunks	2.440	1.000	1.000	2.000	45
Source changed lines	14.334	7.000	3.000	13.000	232
Test files	1.328	1.000	1.000	1.000	14
Test changed lines	23.976	16.000	8.000	28.000	471
FAIL_TO_PASS identifiers	3.032	1.000	1.000	2.000	438
PASS_TO_PASS identifiers	120.284	50.500	19.000	111.250	2476

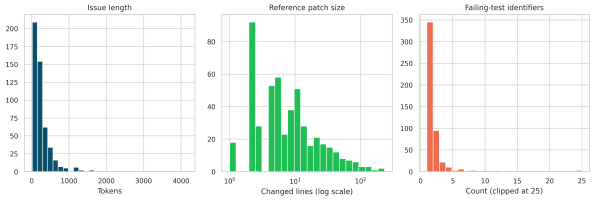

Figure 3 Issue length, reference source-patch size, and failing-test-identifier distributions.

Retrieval Models

For path localization, each query was ranked only against the distinct source paths observed in reference patches from the same repository. This inventory gives every gold path a

candidate but reveals benchmark-wide path membership; results are therefore labeled closed-world. Paths were split on directories, punctuation, underscores, and camel case. The mention baseline assigned weight to exact paths, basenames, stems, and directory components. BM25 used k1 = 1.5 and b = 0.75, following the probabilistic relevance framework [32]. Word TF-IDF used one- and two-token n-grams, sublinear term frequency, L2 normalization, and cosine similarity, grounded in classical term weighting [33]. Character TF-IDF used three- to five-character n-grams, which preserve partial overlap between issue identifiers and path components.

Reciprocal rank fusion combined mention, BM25, word TF-IDF, and character TF-IDF with score 1/(60 + rank), avoiding incompatible raw-score scales [34]. A logistic reranker used ten features: normalized values and within-query rank percentiles for word TF-IDF, character TF-IDF, BM25, and mention matching, plus path depth and log candidate count. It used C = 1, balanced class weights, a fixed random seed, and leave-one-repository-out (LOGO) evaluation. Candidate pairs from the held-out repository never entered training. A random baseline averaged 50 deterministic permutations per task. Table 4 lists the fixed configurations.

All path candidates were sorted before indexing, and stable descending sorts made tied lexical scores deterministic. The supervised reranker used path name as its final tie breaker. Hit@1 asked whether any modified path appeared first; Recall@k measured the fraction of all modified paths recovered within the first k positions; Complete@5 required every modified path to appear in the top five. MRR described the first relevant position, while MAP and nDCG@10 retained information about the ordering of multiple relevant paths. These complementary metrics distinguish finding one plausible file from recovering the complete repair surface.

Table 4 Models, Inputs, And Fixed Hyperparameters.

Model	Input	Configuration
Word-TFIDF	Path or pre-fix hunk text	1-2 word n-grams; sublinear TF; cosine
Char-TFIDF	Path or pre-fix hunk text	3-5 character n-grams; cosine
BM25	Path or pre-fix hunk text	k1=1.5, b=0.75
RRF	Four base rankings	1/(60+rank) fusion
LOGO-Reranker	Ten lexical/rank/path features	balanced logistic; C=1
TFIDF-5NN	Issue text	k=5 cosine neighbors; held-out repository
TFIDF-Logistic	Issue text	balanced logistic; C=1; held-out repository
TFIDF-Ridge	Issue text	log1p target; alpha=10; held-out repository
Effort + Platt	Issue text + closed-world retrieval uncertainty	nested group CV; logistic calibration

Hunk retrieval built one index per repository over the 1,220 pre-fix hunk documents and compared BM25, word TF-IDF, character TF-IDF, and RRF. Four queries were evaluated: issue only; issue plus pre-solution issue comments; issue plus failing-test identifiers; and all three sources. Issue comments capture discussion that occurred before the solution pull request's first commit, while failing-test identifiers are evaluator-side evidence. Neither augmented query is interpreted as the official issue-only condition. Keeping these sources separate is consistent with multi-source operational diagnosis, where metrics, logs, traces, and dependency context contribute non-interchangeable evidence [35]. It also avoids assuming that lexical evidence is always sufficient: in OpenStack failure-injection logs, timing and contextual counters carried the decisive signal even when anomaly-exclusive templates were absent [36].

Patch-Plan And Effort Prediction

Patch-plan models predicted three binary targets from issue text: multiple source files, multiple source hunks, and multiple test files. Training-prevalence, five-nearest-neighbor TF-IDF, and balanced TF-IDF logistic models were evaluated with LOGO folds. Changed source lines were predicted by the training median, five-neighbor median, and ridge regression on log(1 + changed lines) with alpha = 10. These targets describe scope, not patch content.

High-effort prediction used four issue-derived numeric features (token count, character count, code-fence count, and error-term count) and six closed-world retrieval-uncertainty features (top character and RRF scores, top-two margins, maximum mention score, and log candidate count). Numeric-feature, text, and combined logistic models used an outer LOGO test fold. Within each outer training set, five group folds produced out-of-fold decision scores. A one-dimensional logistic calibrator fitted those scores and transformed the held-out repository score. This is Platt calibration in the same post-hoc spirit as modern confidence calibration [37]. Thresholds maximizing macro-F1 were chosen on outer-training out-of-fold probabilities only. Each held-out repository retained its own training-selected threshold, and pooled hard-label metrics applied those stored thresholds task by task. Selective triage then ranked tasks by calibrated high-effort probability, consistent with confidence-based rejection as a measurable risk-coverage policy [38]. This discrimination-calibration-routing separation also appears in operational alert control [39], trajectory-level agent reliability prediction with reconstructed success labels [40], and deterministic cost-aware AIOps routing [41]. The present experiment uses the same decision logic but estimates human repair effort rather than execution success.

The retrieval-uncertainty features summarize score concentration rather than program behavior. In particular, candidate count and rank margins depend on the reference-derived closed-world inventory, so they should not be read as direct estimates of live repository size or execution risk. The text-only model provides a stricter comparison, and the

combined model tests whether uncertainty within the analytical retrieval setting adds useful ranking information. No added or removed source-code line content, test identifier, difficulty label, repository version, or solution date was supplied as an effort predictor.

Evaluation and statistical analysis

Path and hunk retrieval compute Hit@1, mean recall at 3, 5, and 10, complete relevant-set recall at 5, MRR, mean average precision, and nDCG@10. Patch-plan classifiers report PR-AUC, balanced accuracy, macro-F1, and Brier score; changed-line regression reports mean absolute error (MAE), root mean squared logarithmic error (RMSLE), and Spearman correlation. High-effort prediction treats PR-AUC as primary because positives form 9.0% of the corpus, supplemented by ROC-AUC, Brier score, and ten-bin equal-mass expected calibration error (ECE).

Paired task bootstrap used 2,000 resamples for method differences; the primary-table MRR intervals used 5,000 resamples. Wilcoxon signed-rank tests compared task-level reciprocal ranks, and Holm adjustment controlled the three planned comparisons. A cluster sensitivity analysis then averaged each task-level difference within repository, bootstrapped the 12 unweighted repository means, and repeated the Holm-adjusted Wilcoxon tests on those means. This second analysis asks whether a pooled conclusion survives when Django's 231 tasks no longer dominate the inferential unit. All computations used seed 20240813.

C. RESULTS AND DISCUSSION

RQ1: Task and repair structure

The typical Verified task is local, but the tail is operationally important. Of 500 repairs, 429 modify one source file and 71 modify multiple files. The median patch contains one hunk, while 220 tasks contain multiple hunks. Source patches change a mean of 14.33 lines and a median of seven. Regression-test patches change a mean of 23.98 lines and a median of 16. The 1,516 failing-test identifiers contrast with 60,142 passing-test identifiers, showing why a repair executor must protect both targeted behavior and broad regression coverage.

Repository imbalance affects interpretation. Django supplies 46.2% of all tasks, while three repositories together supply 11 tasks. Macro patterns are therefore considered alongside pooled metrics. The benchmark's 373-path inventory also contains 260 paths that appear in only one instance. A path-only model cannot learn a stable history for those files, but character decomposition can still connect issue identifiers to directory and basename fragments.

RQ2: Closed-World Source-Path Localization

Table 5 gives the issue-only results and Figure 4 shows recall as the context budget grows. The LOGO reranker was strongest overall: Hit@1 = 0.500, Recall@5 = 0.747, Complete@5 = 0.714, MRR = 0.625, and nDCG@10 = 0.656. Word TF-IDF obtained the second-highest Hit@1 (0.478), while RRF was second on MRR (0.590). Character TF-IDF achieved Recall@10 = 0.806, indicating that partial identifier overlap is especially useful when a wider context budget is available. All substantive methods greatly exceeded the 50-permutation random baseline, whose Hit@1 was 0.035 and MRR was 0.114.

Table 5 Issue-Only Closed-World Source-Path Localization. MRR Confidence Limits Are 5,000-Resample Task-Bootstrap Intervals.

Method	Hit@1	R@5	Complete@5	MRR	CI low	CI high	nDCG@10
Mention	0.428	0.656	0.622	0.551	0.516	0.587	0.581
BM25	0.420	0.646	0.614	0.540	0.503	0.575	0.573
Word-TFIDF	0.478	0.671	0.636	0.586	0.550	0.623	0.610
Char-TFIDF	0.428	0.699	0.668	0.560	0.525	0.595	0.604
RRF	0.466	0.702	0.672	0.590	0.555	0.626	0.619
LOGO-Reranker	0.500	0.747	0.714	0.625	0.591	0.659	0.656
Random-50	0.035	0.133	0.118	0.114	0.104	0.124	0.122

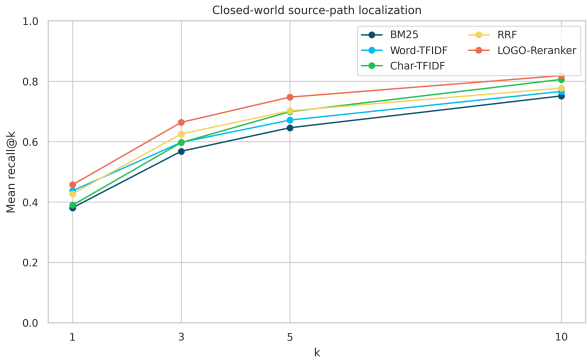

Figure 4 Recall@K For Issue-Only Closed-World Source-Path Localization.

At the task level, the reranker's MRR advantage over character TF-IDF was 0.065 (95% CI 0.046-0.087), with Holm-adjusted $p < 0.001$. RRF improved MRR over character TF-IDF by 0.030 (95% CI 0.002-0.059; adjusted $p = 0.034$). The character TF-IDF versus BM25 difference was not significant after correction. Figure 5 reveals repository heterogeneity: the reranker reaches MRR 0.881 on Astropy and 0.543 on Django, while the one-task Flask score is 1.000 by construction. Single-instance and single-candidate repositories do not estimate generalization and are retained only because the experiment covers all 500 tasks.

With repositories equally weighted, the reranker retains a 0.070 MRR gain (95% CI 0.035-0.113; adjusted $p = 0.006$). RRF's 0.023 gain crosses zero (-0.008 to 0.055; $p = 0.551$).

Thus only the supervised reranker improves localization under both task- and repository-level inference.

Table 6 Planned Paired MRR Comparisons At Task And Repository-Cluster Levels. Confidence Intervals Use Paired Bootstrap; P-Values Are Holm-Adjusted Wilcoxon Tests.

Comparison	Task delta	Task CI low	Task CI high	Task p	Repo delta	Repo CI low	Repo CI high	Repo p
LOGO-Reranker - Char-TFIDF	0.065	0.046	0.087	<0.001	0.070	0.035	0.113	0.006
RRF - Char-TFIDF	0.030	0.002	0.059	0.034	0.023	-0.008	0.055	0.551
Char-TFIDF - BM25	0.020	-0.011	0.053	0.248	0.010	-0.017	0.035	0.551

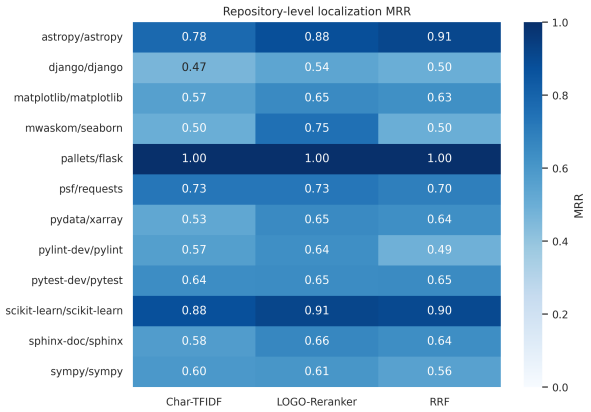

Figure 5 Repository-Level MRR For Character TF-IDF, RRF, And The LOGO Reranker.

Figure 6 plots the reranker effect by repository. The largest positive difference occurs on Seaborn, but it is based on two tasks. More informative gains appear on Xarray, Astropy, Matplotlib, Sphinx, and Django. SymPy changes little, and Requests, Flask, and pytest are essentially neutral. This pattern echoes Bench4BL's warning that localization results depend strongly on project composition [25].

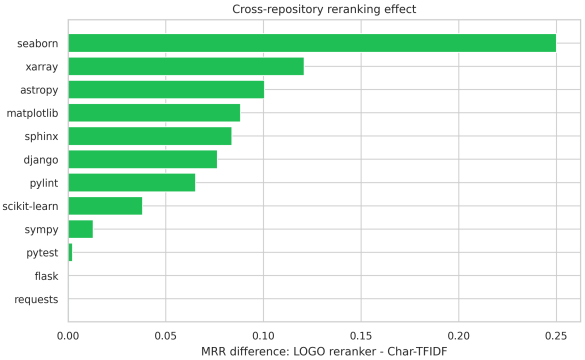

Figure 6 Repository-level MRR difference between the LOGO reranker and character TF-IDF.

RQ3: Diagnostic evidence and pre-fix hunk retrieval

Issue-only RRF obtains 0.466 Hit@1, 0.702 Recall@5, and 0.590 MRR. Comments raise these to 0.542, 0.775, and 0.665; failing-test identifiers to 0.572, 0.795, and 0.685; and all evidence to 0.628, 0.844, and 0.739 (Table 7). Test identifiers are post-failure evidence, not official SWE-bench inputs.

Table 7 RRF Evidence Ablation For Closed-World Source-Path Localization.

Evidence	Hit@1	R@5	Complete@5	MRR	nDCG@10
Issue only	0.466	0.702	0.672	0.590	0.619
Issue + hints	0.542	0.775	0.738	0.665	0.692
Issue + failing tests	0.572	0.795	0.764	0.685	0.717
All evidence	0.628	0.844	0.814	0.739	0.766

For issue-only hunk retrieval, word TF-IDF reaches 0.590 Hit@1, 0.683 Recall@5, and 0.684 MRR; character TF-IDF is close, and RRF does not improve on word TF-IDF. With all evidence, word TF-IDF reaches 0.778 MRR and RRF 0.776 (Table 8). Because candidates are reference-selected, these results measure ordering among known changed regions, not repository-wide line discovery.

Table 8 Gold-Selected Pre-Fix Hunk Retrieval. Added Patch Lines Were Excluded From Candidate Text

Evidence	Method	Hit@1	R@5	Complete@5	MRR	nDCG@10
Issue only	Word-TFIDF	0.590	0.683	0.592	0.684	0.664
Issue only	Char-TFIDF	0.566	0.687	0.592	0.674	0.666
Issue only	BM25	0.510	0.613	0.520	0.615	0.594
Issue only	RRF	0.538	0.685	0.594	0.651	0.650
Issue + hints	RRF	0.636	0.748	0.646	0.737	0.726
Issue + failing tests	RRF	0.606	0.737	0.652	0.711	0.708
All evidence	Word-TFIDF	0.692	0.777	0.674	0.778	0.763
All evidence	Char-TFIDF	0.658	0.774	0.670	0.763	0.751
All evidence	BM25	0.582	0.685	0.584	0.683	0.663
All evidence	RRF	0.672	0.791	0.694	0.776	0.767

The results support staged retrieval: rank paths, order old-side hunks, then generate code. This matches localization-repair-validation workflows [5], [6] while preserving provenance. Retrieval-grounded HDFS analysis likewise combines anomaly ranking, evidence bundles, and refusal

[42]. Host-telemetry work separates sequence detection, suspicious-window localization, and evidence-grounded narrative generation [43], while scientific claim verification separates full-corpus retrieval, candidate reranking, and verdict accuracy [44]. These analogues suggest how failing-test traces could become auditable repair context without conflating evidence ranking with correctness.

RQ4: Patch-plan transfer and calibrated effort triage

Patch-plan transfer was weak. TF-IDF logistic obtains 0.157 PR-AUC for multi-file repair and 0.466 for multi-hunk repair; test-multifile prediction stays near baseline (Table 9). Issue wording therefore weakly determines structural scope across unseen repositories.

Table 9 LOGO Issue-Only Prediction Of Multi-File Source, Multi-Hunk Source, And Multi-File Regression-Test Scope.

Target	Model	Prev.	PR-AUC	Bal. acc.	Macro-F1	Brier
multi_file	Prevalence	0.142	0.123	0.500	0.462	0.122
multi_file	TFIDF-5NN	0.142	0.145	0.497	0.469	0.144
multi_file	TFIDF-Logistic	0.142	0.157	0.506	0.484	0.191
multi_hunk	Prevalence	0.440	0.396	0.500	0.359	0.252
multi_hunk	TFIDF-5NN	0.440	0.440	0.494	0.490	0.305
multi_hunk	TFIDF-Logistic	0.440	0.466	0.523	0.491	0.252
test_multi_file	Prevalence	0.212	0.178	0.500	0.441	0.174
test_multi_file	TFIDF-5NN	0.212	0.200	0.493	0.455	0.203
test_multi_file	TFIDF-Logistic	0.212	0.219	0.488	0.447	0.200

Changed-line regression is also limited: the training median yields 11.074 MAE and 0.979 RMSLE, while ridge yields 11.127 and 0.963 with 0.093 Spearman correlation (Table 10). These are budgeting signals, not precise patch plans.

Table 10 LOGO Issue-Only Prediction Of Reference Source-Patch Changed Lines.

Model	MAE lines	RMSLE	Spearman rho
Training median	11.074	0.979	-0.051
TFIDF-5NN	12.376	1.093	0.103
TFIDF-Ridge	11.127	0.963	0.093

The raw combined effort model obtains 0.142 PR-AUC, 0.642 ROC-AUC, 0.151 Brier, and 0.237 ECE. Platt calibration preserves ranking (0.139 PR-AUC; 0.632 ROC-AUC) while improving Brier to 0.082 and ECE to 0.028; balanced accuracy is 0.561 (Table 11). Calibration corrects probability scale without creating discrimination [37].

Table 11 Nested LOGO high-effort prediction and calibration. High effort denotes an annotator estimate of one hour or longer.

Model	PR-AUC	ROC-AUC	Bal. acc.	Macro-F1	Brier	ECE-10
Numeric features (raw)	0.132	0.629	0.587	0.530	0.211	0.330
Numeric features + Platt	0.124	0.598	0.587	0.530	0.082	0.027
Text (raw)	0.122	0.577	0.551	0.516	0.151	0.260
Text + Platt	0.112	0.558	0.551	0.516	0.082	0.036
Combined (raw)	0.142	0.642	0.561	0.548	0.151	0.237
Combined + Platt	0.139	0.632	0.561	0.548	0.082	0.028
Training prevalence	0.077	0.438	0.500	0.476	0.082	0.026

Figure 7 and Table 12 show selective escalation. Reviewing the top 20% captures 28.9% of high-effort tasks; 40% captures 60.0%; and 50% captures 71.1%, reducing retained high-effort risk to 5.2%. The modest enrichment supports adding execution signals before high-stakes automation.

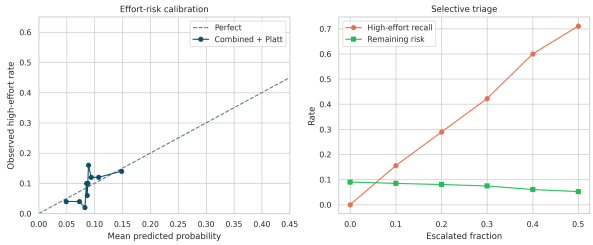

Figure 7 Calibration And Selective Escalation For The Combined High-Effort Model.

Table 12 Selective Escalation Based On Calibrated Combined High-Effort Probability.

Escalated	Coverage	High-effort recall	Escalated precision	Remaining risk
0.000	1.000	0.000	0.000	0.090
0.100	0.900	0.156	0.140	0.084
0.200	0.800	0.289	0.130	0.080
0.300	0.700	0.422	0.127	0.074
0.400	0.600	0.600	0.135	0.060
0.500	0.500	0.711	0.128	0.052

Implications And Validity

This is triage, not end-to-end repair. TravisTorrent links builds, commits, and metadata [45], and Repairnator shows the work needed to reproduce failures [46]. Learned repair systems rank edits from successful patches [47], consistent with code-reuse evidence [48]. Here retrieval can condition a generator, but functional evaluation still requires patch application and FAIL_TO_PASS plus PASS_TO_PASS execution.

Passing tests do not guarantee correctness because repair can overfit a suite [49]. Defects4J illustrates the controlled faulty/fixed infrastructure needed for execution studies [50]. Candidate patches, application status, test outcomes, runtime, and traces are absent here. Interpretable CloudTrail analysis combines schema-aware event features

with HMM smoothing to produce auditable attack-stage labels [51], illustrating why future repair studies should retain ordered tool and test events rather than only final outcomes.

Executable agents need controls beyond retrieval confidence. Refusal policies constrain unsafe actions [52], and incident cards expose evidence during review [53]. Scientific-search evidence cards similarly surface source text, confidence cues, and ranking rationales [54]. Continual guard updates address shifting attacks [55]; risk cards expose permissions and consequences [56]; and evidence-based self-verification adds a pre-action check [57]. These future layers are not evaluated here.

Three constraints bound the findings: the path inventory excludes unchanged files; the hunk pool is gold-selected despite excluding added lines; and 12 imbalanced repositories span 2013-2023 solution dates. LOGO models, repository reporting, evidence ablations, and paired inference reduce but do not remove these threats. Because each repository acts as a domain, multi-source adaptation with approximately shared features offers a principled direction for separating transferable content from project-specific environment features [58].

D. CONCLUSIONS

All 500 Verified tasks were evaluated. The parser recovered 373 source paths and 1,220 hunks. The issue-only LOGO reranker reaches 0.500 Hit@1, 0.747 Recall@5, and 0.625 MRR, with a significant 0.065 gain over character TF-IDF under task- and repository-level inference. Post-failure test identifiers raise RRF MRR to 0.685; all-evidence oracle-pool hunk MRR peaks at 0.778. Scope prediction remains weak. Calibration lowers combined-model ECE from 0.237 to 0.028, but PR-AUC remains 0.139.

The practical design is staged: lexical path retrieval, post-failure test evidence, calibrated escalation, and correctness claims only after repository-specific patch execution. This preserves the distinctions among localization, human effort, test adequacy, and functional repair.

E. REFERENCES

[1] [1] C. E. Jimenez, J. Yang, A. Wettig, S. Yao, K. Pei, O. Press, and K. R. Narasimhan, “SWE-bench: Can language models resolve real-world GitHub issues?” in Proc. Int. Conf. Learn. Representations (ICLR), 2024. [Online]. Available: https://openreview.net/forum?id=VTF8yNQM66

[2] [2] N. Chowdhury et al., “Introducing SWE-bench Verified,” OpenAI, Aug. 13, 2024. [Online]. Available: https://openai.com/index/introducing-swe-bench-verified/

[3] [3] OpenAI, “Why SWE-bench Verified no longer measures frontier coding capabilities,” Feb. 23, 2026. [Online]. Available: https://openai.com/index/why-we-no-longer-evaluate-swe-bench-verified/

[4] [4] J. Yang et al., “SWE-agent: Agent-computer interfaces enable automated software engineering,” in Advances in Neural Information Processing Systems, vol. 37, pp. 50528–50652, 2024.

[5] [5] C. S. Xia, Y. Deng, S. Dunn, and L. Zhang, “Demystifying LLM-based software engineering agents,” Proc. ACM Softw. Eng., vol. 2, no. FSE, pp. 801–824, 2025, doi: 10.1145/3715754.

[6] [6] Y. Zhang, H. Ruan, Z. Fan, and A. Roychoudhury, “AutoCodeRover: Autonomous program improvement,” in Proc. 33rd ACM SIGSOFT Int. Symp. Softw. Test. Anal., 2024, pp. 1592–1604, doi: 10.1145/3650212.3680384.

[7] [7] F. Zhang et al., “RepoCoder: Repository-level code completion through iterative retrieval and generation,” in Proc. EMNLP, 2023, pp. 2471–2484, doi: 10.18653/v1/2023.emnlp-main.151.

[8] [8] W. Cheng, Y. Wu, and W. Hu, “Dataflow-guided retrieval augmentation for repository-level code completion,” in Proc. 62nd Annu. Meeting Assoc. Comput. Linguistics, 2024, pp. 7957–7977, doi: 10.18653/v1/2024.acl-long.431.

[9] [9] H. Zhang, “LLM-Driven CI Failure Diagnosis and Automated Repair: From GitHub Actions Logs to Patch Recommendation,” J. Technol. Informatics Eng., vol. 4, no. 1, pp. 190–214, Apr. 2025, doi: 10.51903/jtie.v4i1.484.

[10] [10] B. Zhang, X. Sun, G. Liu, and B. Zhou, “LLM-Style DevOps Copilot for Cloud-Native Troubleshooting: Retrieval-Augmented Runbook Generation and Command-Safety Evaluation,” J. Technol. Informatics Eng., vol. 5, no. 2, pp. 104–118, Aug. 2026, doi: 10.51903/jtie.v5i2.534.

[11] [11] H. Husain, H.-H. Wu, T. Gazit, M. Allamanis, and M. Brockschmidt, “CodeSearchNet challenge: Evaluating the state of semantic code search,” arXiv:1909.09436, 2019.

[12] [12] Z. Feng et al., “CodeBERT: A pre-trained model for programming and natural languages,” in Findings of EMNLP, 2020, pp. 1536–1547, doi: 10.18653/v1/2020.findings-emnlp.139.

[13] [13] D. Guo et al., “GraphCodeBERT: Pre-training code representations with data flow,” in Proc. Int. Conf. Learn. Representations, 2021. [Online]. Available: https://openreview.net/forum?id=jLoC4ez43PZ

[14] [14] P. Lewis et al., “Retrieval-augmented generation for knowledge-intensive NLP tasks,” in Advances in Neural Information Processing Systems, vol. 33, pp. 9459–9474, 2020.

[15] [15] Y. Li, “Findable then Explainable: Retrieval–Summary Integration for Code Intelligence on a Lightweight CodeSearchNet Subset,” J. Adv. Comput. Syst., vol. 4, no. 7, pp. 65–82, Jul. 2024, doi: 10.69987/JACS.2024.40706.

[16] [16] Y. Chen and H. Xu, “Trust-Calibrated Multilingual RAG for Humanitarian Information Platforms: Empirical Evaluation on OMoS-QA for

Migration Information Access," Int. J. Graph. Des., vol. 4, no. 1, pp. 141–164, Apr. 2026, doi: 10.51903/ijgd.v4i1.3552.

[17] [17] B. Zhang, H. Rao, and D. Zhao, "Evidence-Grounded RAG for Cloud-Native DevOps: Hallucination-Resistant AIOps Question Answering over Private Operations Documents," J. Adv. Comput. Syst., vol. 4, no. 3, pp. 109–125, Mar. 2024, doi: 10.69987/JACS.2024.40308.

[18] [18] G. Liu, C. Li, and E. Zhang, "OpsLLM for Cloud Incident Triage: Bilingual RAG-Based Root Cause Analysis and Alert Summarization for AI Infrastructure Operations," J. Adv. Comput. Syst., vol. 4, no. 4, pp. 97–111, Apr. 2024, doi: 10.69987/JACS.2024.40408.

[19] [19] C. Li, J. Bai, and S. Wang, "Evidence-Chain Reliable RAG: Word-Level Hallucination Detection, Source Attribution, and Provenance Explanation for LLM Applications," J. Adv. Comput. Syst., vol. 4, no. 2, pp. 76–92, Feb. 2024, doi: 10.69987/JACS.2024.40207.

[20] [20] J. Jin, "Evidence-Chain Reliable RAG: Hallucination Detection, Source Attribution, and Deterministic Provenance Explanations," J. Technol. Informatics Eng., vol. 4, no. 2, pp. 520–533, Aug. 2025, doi: 10.51903/jtie.v4i2.535.

[21] [21] Z. S. Zhong, J. Chen, E. Zhong, and X. Sun, "Evidence-Calibrated RAG for Unanswerable Question Answering: Retrieval Coverage, Abstention Calibration, and Hallucination-Proxy Analysis on SQuAD 2.0," J. Technol. Informatics Eng., vol. 4, no. 2, pp. 502–520, Aug. 2025, doi: 10.51903/jtie.v4i2.536.

[22] [22] X. Chang, Y. Lu, and Z. S. Zhong, "Review-Grounded Explainable Recommendation with Faithfulness Evaluation on Amazon Reviews," J. Electr. Eng. Comput. Sci., vol. 11, no. 1, pp. 9–22, May 2026, doi: 10.54732/jeecs.v11i1.2.

[23] [23] J. Zhou, H. Zhang, and D. Lo, "Where should the bugs be fixed? More accurate information retrieval-based bug localization based on bug reports," in Proc. 34th Int. Conf. Softw. Eng., 2012, pp. 14–24, doi: 10.1109/ICSE.2012.6227210.

[24] [24] R. K. Saha, M. Lease, S. Khurshid, and D. E. Perry, "Improving bug localization using structured information retrieval," in Proc. 28th IEEE/ACM Int. Conf. Automated Softw. Eng., 2013, pp. 345–355, doi: 10.1109/ASE.2013.6693093.

[25] [25] J. Lee, D. Kim, T. F. Bissyandé, W. Jung, and Y. Le Traon, "Bench4BL: Reproducibility study on the performance of IR-based bug localization," in Proc. ISSTA, 2018, pp. 61–72, doi: 10.1145/3213846.3213856.

[26] [26] T.-D. B. Le, F. Thung, and D. Lo, "Predicting effectiveness of IR-based bug localization techniques," in Proc. IEEE 25th Int. Symp. Softw. Rel. Eng., 2014, pp. 335–345, doi: 10.1109/ISSRE.2014.39.

[27] [27] X. Ye, R. Bunescu, and C. Liu, "Learning to rank relevant files for bug reports using domain knowledge," in Proc. ACM SIGSOFT Int. Symp. Foundations Softw. Eng., 2014, pp. 689–699, doi: 10.1145/2635868.2635874.

[28] [28] W. Su, S. Chen, and C. Zhao, "Budgeted Multi-Hop Retrieval Agent for Compositional Question Answering: A Retrieval-Policy Evaluation on the Official MultiHop-RAG Benchmark," J. Technol. Informatics Eng., vol. 4, no. 3, pp. 649–662, Dec. 2025, doi: 10.51903/jtie.v4i3.543.

[29] [29] R. Zhang, Z. Wen, C. Wang, C. Tang, P. Xu, and Y. Jiang, "Quality Analysis and Evaluation Prediction of RAG Retrieval Based on Machine Learning Algorithms," arXiv:2511.19481, 2025, doi: 10.48550/arXiv.2511.19481.

[30] [30] Z. S. Zhong and S. Ling, "Improved Theoretical Guarantee for Rank Aggregation via Spectral Method," Inf. Inference: J. IMA, vol. 13, no. 3, Art. no. iaae020, Sep. 2024, doi: 10.1093/imaiai/iaae020.

[31] [31] S. Meng, J. Chen, and I. Zheng, "LLM-Inspired Offline Reranking for Financial Search: Query Rewriting, Hybrid Retrieval, and Listwise Relevance Ranking on FiQA," J. Technol. Informatics Eng., vol. 5, no. 1, pp. 361–378, Apr. 2026, doi: 10.51903/jtie.v5i1.537.

[32] [32] S. Robertson and H. Zaragoza, "The probabilistic relevance framework: BM25 and beyond," Found. Trends Inf. Retr., vol. 3, no. 4, pp. 333–389, 2009, doi: 10.1561/1500000019.

[33] [33] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," Inf. Process. Manage., vol. 24, no. 5, pp. 513–523, 1988, doi: 10.1016/0306-4573(88)90021-0.

[34] [34] G. V. Cormack, C. L. A. Clarke, and S. Büttcher, "Reciprocal rank fusion outperforms Condorcet and individual rank learning methods," in Proc. 32nd Int. ACM SIGIR Conf., 2009, pp. 758–759, doi: 10.1145/1571941.1572114.

[35] [35] G. Liu, S. He, and I. Liu, "LLM-Augmented Multi-Source Root Cause Attribution for CPU and Network Faults in Microservices," J. Adv. Comput. Syst., vol. 3, no. 6, pp. 39–57, Jun. 2023, doi: 10.69987/JACS.2023.30604.

[36] [36] Q. Xin, "Explaining OpenStack Failure-Injection Log Anomalies with Retrieved Normal Prototypes," Emerg. Inf. Sci. Technol., vol. 6, no. 2, pp. 125–146, Nov. 2025, doi: 10.18196/eist.v6i2.31232.

[37] [37] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in Proc. 34th Int. Conf. Mach. Learn., vol. 70, 2017, pp. 1321–1330.

[38] [38] Y. Geifman and R. El-Yaniv, "Selective classification for deep neural networks," in Advances in Neural Information Processing Systems, vol. 30, 2017.

[39] [39] Q. Xin, "Log Anomaly Detection with Conformal Alert Control and Evidence-Grounded Incident Ticket Generation," AVITEC, vol. 8, no. 2,

pp. 247–264, May 2026, doi: 10.28989/avitec.v8i2.3974.

[40] [40] Z. S. Zhong, C. Li, and H. Rao, “Trajectory Reliability Prediction for Generalist AI Agents: Tool-Use Failure Analysis and Success Forecasting on ZClawBench,” J. Technol. Informatics Eng., vol. 5, no. 1, pp. 341–360, Apr. 2026, doi: 10.51903/jtie.v5i1.539.

[41] [41] C. Li, G. Liu, and Z. Zhao, “Cost-Aware LLM-Style Routing for AIOps Log Analysis: Log Parsing, Anomaly Detection, Fault Diagnosis, and Incident Summarization on LogEval Task Files,” J. Technol. Informatics Eng., vol. 5, no. 2, pp. 91–103, Aug. 2026, doi: 10.51903/jtie.v5i2.538.

[42] [42] X. Sun, Z. S. Zhong, and Q. Wu, “Retrieval-Grounded HDFS Log Anomaly Detection and Deterministic Failure Narrative Generation,” J. Comput. Syst. Appl., vol. 3, no. 1, pp. 15–30, Jun. 2026, doi: 10.64229/j6d7fr94.

[43] [43] Q. Xin, “Host-Based Intrusion Detection with System Call Sequences: Window Localization and Forensic Narratives,” AVITEC, vol. 8, no. 2, pp. 325–334, Jun. 2026, doi: 10.28989/avitec.v8i2.3973.

[44] [44] W. Su, S. Chen, and E. Qian, “Narrative-Aware Scientific Claim Verification Agent with Evidence Ranking for ClimateCheck,” J. Technol. Informatics Eng., vol. 5, no. 1, pp. 327–340, Apr. 2026, doi: 10.51903/jtie.v5i1.549.

[45] [45] M. Beller, G. Gousios, and A. Zaidman, “TravisTorrent: Synthesizing Travis CI and GitHub for full-stack research on continuous integration,” in Proc. IEEE/ACM 14th Int. Conf. Mining Softw. Repositories, 2017, pp. 447–450, doi: 10.1109/MSR.2017.24.

[46] [46] S. Urli, Z. Yu, L. Seinturier, and M. Monperrus, “How to design a program repair bot? Insights from the Repairnator project,” in Proc. ICSE Softw. Eng. Practice, 2018, pp. 95–104, doi: 10.1145/3183519.3183540.

[47] [47] F. Long and M. Rinard, “Automatic patch generation by learning correct code,” in Proc. 43rd ACM SIGPLAN-SIGACT Symp. Principles Program. Lang., 2016, pp. 298–312, doi: 10.1145/2837614.2837617.

[48] [48] E. T. Barr, Y. Brun, P. Devanbu, M. Harman, and F. Sarro, “The plastic surgery hypothesis,” in Proc. ACM SIGSOFT Int. Symp. Foundations Softw. Eng., 2014, pp. 306–317, doi: 10.1145/2635868.2635898.

[49] [49] E. K. Smith, E. T. Barr, C. Le Goues, and Y. Brun, “Is the cure worse than the disease? Overfitting in automated program repair,” in Proc. ESEC/FSE, 2015, pp. 532–543, doi: 10.1145/2786805.2786825.

[50] [50] R. Just, D. Jalali, and M. D. Ernst, “Defects4J: A database of existing faults to enable controlled testing studies for Java programs,” in Proc. ISSTA, 2014, pp. 437–440, doi: 10.1145/2610384.2628055.

[51] [51] J. Bai, S. Chen, D. Zheng, and M.-J. Kuo, “Interpretable Attack-Chain Stage Detection from AWS CloudTrail Event Sequences via Linear Models and HMM Smoothing,” Infotron, vol. 6, no. 1, pp. 28–43, May 2026, doi: 10.33474/infotron.v6i1.24923.

[52] [52] D. Zheng and C. Li, “Behavior-Level Jailbreak Resistance via Multi-Stage Refusal + Utility Preservation,” J. Adv. Comput. Syst., vol. 4, no. 1, pp. 83–99, Jan. 2024, doi: 10.69987/JACS.2024.40107.

[53] [53] J. Nie, G. Liu, C. Li, and T. Zou, “Evidence-Constrained Incident Visualization Cards for Distributed Cloud Logs: A UI/UX Framework for Turning Hadoop, OpenStack, and ZooKeeper Logs into Actionable SRE Design Interfaces,” Int. J. Graph. Des., vol. 4, no. 1, pp. 179–185, Apr. 2026, doi: 10.51903/ijgd.v4i1.3703.

[54] [54] J. Jin, “LLM-Style Evidence Cards for Scientific Search Interfaces: A UI/UX Design Framework for Retrieval Transparency, Ranking Trust, and Visual Evidence Hierarchy,” Int. J. Graph. Des., vol. 3, no. 2, pp. 397–414, Oct. 2025, doi: 10.51903/ijgd.v3i2.3698.

[55] [55] D. Zheng, C. Li, and H. Davidson, “Continual Red-Teaming for In-the-Wild Jailbreaks via Online Guardrail Updates and Guardrail Distillation,” J. Adv. Comput. Syst., vol. 3, no. 2, pp. 35–49, Feb. 2023, doi: 10.69987/JACS.2023.30203.

[56] [56] W. Su, H. Rao, and E. Ma, “Privacy and Data-Integrity Risk Cards for LLM Agents: A UI/UX Design Framework for Secure Human Oversight under Prompt-Injection Attacks,” Int. J. Graph. Des., vol. 4, no. 1, pp. 186–191, Apr. 2026, doi: 10.51903/ijgd.v4i1.3699.

[57] [57] D. Zheng, B. Zhang, and J. Geibel, “VerifySafe: Toxicity-Safe Agent Responses under Adversarial Prompts with Evidence-Based Self-Verification,” J. Adv. Comput. Syst., vol. 4, no. 1, pp. 67–82, Jan. 2024, doi: 10.69987/JACS.2024.40106.

[58] [58] Z. S. Zhong, X. Pan, and Q. Lei, “Bridging Domains with Approximately Shared Features,” in Proc. 28th Int. Conf. Artif. Intell. Stat. (AISTATS), PMLR, vol. 258, pp. 559–567, 2025. [Online]. Available: https://proceedings.mlr.press/v258/zhong25a.html.