IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 11, November 2026

"PatchScope: Confidence-Aware Bug Localization Across Unseen Software Repositories"

Mr. Kedarnath Gurav[1], Prof. Manoj Vasant Nikum[2], Prof. Shivam Limbhare[3]

Research Scholar, Master of Computer Applications, Shri JRIT, Dondaicha, KBC NMU Jalgaon, Maharashtra, India[1]

HOD, Master of Computer Applications, Shri JRIT, Dondaicha, KBC NMU Jalgaon, Maharashtra, India[2]

Professor, Master of Computer Applications, Shri JRIT, Dondaicha, KBC NMU Jalgaon, Maharashtra, India[3]

Abstract: Software debugging often begins with a textual bug report that describes unexpected behaviour without naming the faulty source file, so developers may inspect many files before locating the defect. Information-retrieval-based bug localization ranks files by relevance to a report, but a model that performs well on familiar projects may struggle on repositories absent from its training data, and a ranked list does not tell the developer when it is unreliable. This paper presents PatchScope, a proposed confidence-aware framework for file-level bug localization on unseen software repositories. PatchScope combines reproducible baseline rankers (TF-IDF and BM25) with a confidence estimate and an abstention rule that flags uncertain predictions instead of presenting every ranking as equally reliable. The evaluation protocol uses repository-disjoint training, validation and test partitions so that confidence thresholds are never selected on test repositories, and reports Mean Reciprocal Rank, Recall@k, Mean Average Precision, calibration error and the coverage versus selective-risk trade-off. This manuscript describes the motivation, related work, design and evaluation protocol. Implementation and experiments are not yet complete; therefore no performance results are claimed, and the research gap is stated as preliminary pending a wider literature screening.

Keywords: Bug localization, information retrieval, cross-project generalization, confidence calibration, selective prediction, abstention, software maintenance.

I. INTRODUCTION

Maintaining a large software system involves a constant stream of bug reports. A report typically describes symptoms, such as a crash, a wrong output or an unexpected screen, rather than the source file responsible. A developer must therefore search a repository that may contain thousands of files before the defect is found, and this search consumes a significant part of maintenance effort. Information-retrieval-based (IR-based) bug localization addresses this problem by treating the bug report as a query and the source files as a document collection, and returning a ranked list of candidate files [1], [2].

Research over more than a decade has improved these rankings by using similar previously fixed reports [1], structural information about classes and methods [3], and stack traces, comments and code-change history [4]. More recent work uses deep learning and transfer learning to carry knowledge from one project to another [2], [5]–[7]. In practice, however, a localization tool is most valuable on a repository that it has never seen, and performance measured on familiar projects does not guarantee similar behaviour there.

A second, less studied difficulty is reliability. A ranked list is presented to the developer with the same authority whether the top file is an obvious match or a poor guess. If the system could signal that a particular prediction is uncertain, the developer could fall back on manual search instead of trusting a misleading ranking. Confidence calibration [8] and selective prediction with a reject option [9]–[11] provide established foundations for this idea in general machine learning, but their usefulness for bug-report-to-file ranking must be demonstrated rather than assumed.

This paper proposes **PatchScope**, a framework for studying *confidence-aware* bug localization across unseen software repositories. The intended contributions are: (i) a reproducible baseline ranking pipeline for file-level localization; (ii) a repository-separated evaluation protocol in which confidence thresholds are selected only on validation repositories; (iii) a confidence and abstention mechanism, together with a definition of correctness for accepted predictions; and (iv) an evaluation of the trade-off between coverage and selective risk. The study is guided by three research questions:

RQ1: How well do baseline localization methods perform on repositories unseen during training?
RQ2: Can confidence scores identify unreliable localization predictions?
RQ3: How does the error rate among accepted predictions change as the system abstains more often?
Status of the work. PatchScope is a proposed framework. At the time of writing the pipeline has not been implemented and no experiments have been run, so this paper reports design and protocol only; no performance figures are claimed. The remainder of the paper is organised as follows: Section II reviews related work, Section III describes the proposed methodology, Section IV gives the evaluation plan and current status, and Section V concludes.

II. LITERATURE REVIEW

This section summarises the prior work that frames PatchScope. The set of studies reviewed is a curated starting point rather than an exhaustive survey, and the research gap in Section II-D is accordingly stated as preliminary.

A. *IR-Based Bug Localization*

BugLocator [1] ranks source files with a revised vector-space model and additionally uses similar, previously fixed bug reports. It established the now-standard formulation in which a bug report is a query and source files are searchable documents, and it remains a natural lexical baseline. BLUiR [3] argued that treating source code as flat text discards useful signal, and used structured retrieval that distinguishes constructs such as class and method names. BLIA [4] combined bug-report text, stack traces, comments, structured information and code-change history, and explored method-level localization. A recent survey of 61 deep-learning-based IR bug-localization studies [2] organises representations, matching strategies, datasets, evaluation practice and open challenges, and is the main entry point for locating further primary studies.
Two cautions from this line of work shape PatchScope. First, signals such as project-specific change history may not exist for a genuinely new repository and, if computed carelessly, can leak future information into the evaluation. Second, none of these systems is designed to report whether an individual ranking is trustworthy.

B. *Cross-Project and Cross-Language Localization*

Cross-project localization is already an established direction. Deep Transfer Bug Localization (TRANP-CNN) [5] learns transferable semantic features from source projects and uses labelled data from the target project, which differs from a strict setting with no target labels. CooBa [6] applies adversarial transfer learning to separate shared cross-project information from project-specific information, and highlights negative transfer as a real risk. BLAZE [7] addresses cross-language and cross-project localization using dynamic code chunking and hard-example learning with a code-oriented language model, and introduces the BeetleBox multi-language, multi-project dataset. Because cross-project and cross-language localization have therefore been addressed, PatchScope does not present them as its own contribution; it uses repository-held-out testing as an evaluation condition for the reliability question.

C. *Calibration and Selective Prediction*

Guo et al. [8] showed that modern neural networks are often poorly calibrated, so a raw model score should not automatically be read as a probability, and compared post-processing remedies including temperature scaling. Selective classification [9] lets a model reject uncertain inputs and evaluates the resulting trade-off between coverage and error risk; SelectiveNet [10] learns the prediction and the reject function jointly, and El-Yaniv and Wiener [11] give the theoretical foundations of the risk–coverage trade-off. In natural language processing, Xin et al. [12] study selective prediction and confidence-based abstention. These works are general machine-learning foundations: they do not show that the same methods are effective for source-file ranking, and the correctness of a ranking prediction must be defined for the localization task.

TABLE I SUMMARY OF RELATED WORK AND RELEVANCE TO PATCHSCOPE

Work	Approach	Relevance / caution
BugLocator [1]	Revised vector-space model plus similar fixed reports	Lexical baseline; no calibrated confidence or abstention
BLUiR [3]	Structured IR over classes and methods	Motivates structural features; not selective prediction
BLIA [4]	Text, stack traces, structure, change history	Feature fusion; history may be unavailable or leak in unseen repositories
TRANP-CNN [5]	Deep transfer learning across projects	Cross-project reference; uses labelled target data

 This work is licensed under a Creative Commons Attribution 4.0 International License

CooBa [6]	Adversarial cross-project transfer	Shows negative-transfer risk; abstention not central
BLAZE [7]	Chunking and hard-example learning; BeetleBox	State of the art; cross-language/project not novel for PatchScope
Survey [2]	Review of 61 DL-based IR studies	Secondary source for expanding the review
Calibration [8]	Temperature scaling and related methods	General ML; must be validated for file ranking
Selective prediction [9]–[12]	Reject option, risk–coverage, abstention	General ML/NLP; needs a ranking-specific definition of error

D. *Preliminary Research Gap*

The reviewed work suggests a narrow question that remains to be verified: whether calibrated confidence and selective abstention improve the operational reliability of file-level bug localization when target repositories are excluded from training. PatchScope therefore proposes to evaluate the combination of repository-held-out generalization and confidence-based abstention in a controlled experiment. This is **not yet a validated novelty claim**. Before one is made, further screening must establish whether earlier bug-localization work has calibrated top-ranked file correctness or used a reject option, and whether BLAZE or another recent system already exposes calibrated confidence or selective prediction.

III. METHODOLOGY

PatchScope is organised as a six-stage pipeline (Fig. 1). The design below is proposed and not yet implemented; all choices are intended to keep the experiments reproducible and leakage-free.

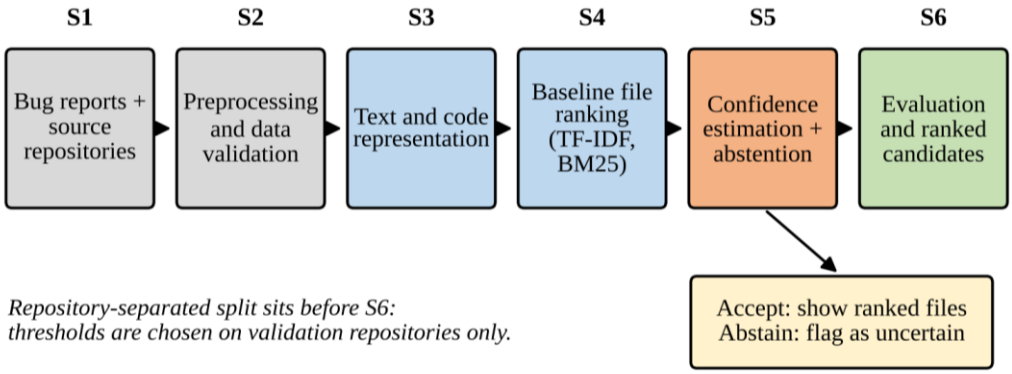


Fig. 1 Proposed PatchScope pipeline: from bug reports and repositories to ranked candidate files with an accept/abstain decision.

A. *Problem Formulation*

Let R be a repository with a set of source files $F(R)$, and let b be a bug report submitted against R. The localization task is to produce a ranking of $F(R)$ by relevance to b. The ground truth $G(b)$ is the set of files modified by the commit that fixed b, as recorded in the dataset; when several files are changed, the definition of relevance is fixed in advance and reported. The system also returns a decision $d(b) \in \{\text{accept, abstain}\}$. If it accepts, the ranked list is shown; if it abstains, the report is flagged as uncertain.

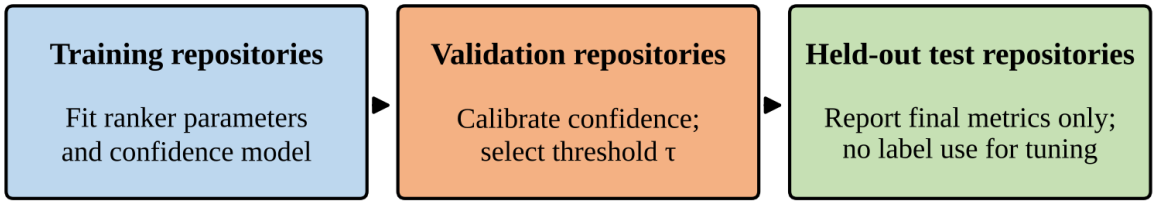
B. *Data and Repository-Separated Splits*

The study will use suitable public bug-report and source-code data, subject to availability and licensing. BeetleBox, introduced with BLAZE [7], is a candidate because it spans multiple repositories and languages; before adoption the exact release, record count, repository identifiers, commit snapshots, labels, licence and overlap with any pretrained-model data will be inspected and the version used will be recorded, since paper and dataset releases may report different counts. Records are split by *repository* into training, validation and held-out test partitions (Fig. 2), so that no repository contributes to more than one partition. A random split over bug reports would let a model see the test repository during training and would overstate generalization. Where a language-held-out evaluation is added, it is reported separately so that unseen-project and unseen-language generalization are not confused.

IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 11, November 2026



No repository appears in more than one partition; an optional language-held-out evaluation is reported separately.

Fig. 2 Repository-disjoint data partitions and the role of each partition.

C. *Preprocessing and Representation*

Bug-report text (title and description) and source files are cleaned and tokenised reproducibly. For source code, identifiers are split on camel case and underscores, language keywords and very short tokens are removed, and fields such as class names, method names and comments can be indexed separately, following the structural motivation of BLUiR [3]. Each bug is paired with the repository snapshot that precedes its fix commit. Signals derived from project history, as in BLIA [4], are excluded from the baseline unless they can be computed strictly from information available before the report date; stack-trace matching is used only when a trace is present in the report.

D. *Baseline File Ranking*

Two lexical baselines are planned. The first represents reports and files as TF-IDF vectors and ranks files by cosine similarity. The second ranks files with BM25. Both are inexpensive, require no target-repository training and give a transparent reference against which later representations can be compared. For a report b the ranker produces scores $s_1 \geq s_2 \geq \ldots$ for the ranked files.

E. *Confidence Estimation and Abstention*

The proposed first confidence signal is the relative margin between the two highest-scoring files, a scale-invariant version of the margin sketched in the project design:

$$m(b) = (s_1 - s_2) / (s_1 + \varepsilon)$$

where ε is a small constant. Further candidate features are the top score, the entropy of the normalised score distribution, report length, the number of candidate files and query-term coverage. A simple model, such as logistic regression, can map these features to the probability $p(b)$ that the top-ranked file is in $G(b)$. This probability is calibrated on the validation repositories only, using Platt scaling or isotonic regression (or temperature scaling [8] where logits exist), and its quality is checked with reliability diagrams. The abstention rule is then

$$d(b) = \text{abstain if } c(b) < \tau, \text{ otherwise accept,}$$

where $c(b)$ is the margin or the calibrated probability. The threshold τ is chosen on the validation repositories, for example as the smallest τ whose validation selective risk does not exceed a target level, and is then applied unchanged to the held-out repositories. The conceptual decision step is:

```
ranked = rank_files(report, repo)      # TF-IDF or BM25
conf   = confidence(report, ranked)    # margin or calibrated p
if conf < tau:                         # tau fixed on validation repos
   return ABSTAIN
return ranked[:k]
```

A fixed post-training threshold is the simplest design; integrated reject-option architectures such as SelectiveNet [10] are noted as an alternative that would require a ranking-specific formulation and is left for future work.

IV. EVALUATION PLAN AND CURRENT STATUS

A. *Metrics*

Ranking quality is measured with Mean Reciprocal Rank (MRR), Recall@k and Mean Average Precision (MAP). Confidence is evaluated separately, because these ranking measures do not by themselves establish that confidence scores are calibrated. Table II lists the planned measures.

TABLE II PLANNED EVALUATION MEASURES

Area	Measure	Meaning and precaution
Ranking	MRR	Mean of 1/rank of the first relevant file; define relevance when several files change
Ranking	Recall@k	Share of reports with a relevant file in the top *k*
Ranking	MAP	Average precision over relevant files, averaged across reports
Calibration	Brier score, ECE, reliability diagram	Fit calibrator on validation repositories only
Abstention	Coverage	Fraction of reports for which a prediction is given
Abstention	Selective risk	Error rate among accepted reports; error = top-ranked file $\notin G(b)$
Abstention	Risk–coverage curve, AURC	Choose τ without test labels; do not equate coverage with accuracy

$$\text{MRR} = (1/|B|)\ \Sigma\ 1/\text{rank}(b), \quad \text{Coverage} = |\text{accepted}| / |B|, \quad \text{Selective risk} = \text{errors among accepted} / |\text{accepted}|$$

B. *Experimental Protocol*

The planned protocol is: (1) validate data and source-file labels; (2) prevent repository leakage across partitions; (3) compare the baseline rankers; (4) calibrate confidence and select τ on validation repositories; (5) report held-out repository performance; and (6) plot selective risk against coverage. Results will be reported per repository and by report category where the data permit, and macro-averaged over repositories so that large repositories do not dominate. Because the number of repositories may be small, uncertainty will be summarised with bootstrap intervals over repositories. Error analysis will examine cases in which similar terminology, very large repositories or sparse report detail make localization difficult. RQ1 is addressed by the held-out ranking metrics, RQ2 by calibration and by whether confidence separates correct from incorrect top-ranked predictions, and RQ3 by the risk–coverage curve.

C. *Threats to Validity*

Ground-truth labels derived from fix commits can be noisy, for example when a commit bundles unrelated changes. A small number of repositories limits statistical power. If code-oriented language models are added later, pretraining data may overlap with benchmark repositories. Finally, a confidence signal that is uninformative would lower the benefit of abstention, and abstention that simply rejects the hardest reports at very low coverage would not be useful. We do not presuppose that abstention helps; a negative answer to RQ2 would itself be an informative finding.

D. *Current Status*

The framework, data protocol and metrics are defined, but the implementation and experiments are not yet available. Accordingly this paper contains no tables or plots of experimental results, and every quantity in Table II is a planned measure rather than an observed value. Results will be added once the experiments are completed.

V. CONCLUSION

This paper proposed PatchScope, a confidence-aware approach to file-level bug localization on unseen repositories. It combines reproducible lexical baselines with a confidence estimate and abstention rule, evaluated under repository-disjoint splits with ranking, calibration and risk–coverage measures. The contribution at this stage is a clearly defined research design: because cross-project and cross-language localization have already been studied [5]–[7], the focus is deliberately narrowed to calibrated reliability and abstention, and the claim of novelty remains conditional on further literature screening.

Future work includes completing the implementation and experiments, adding semantic representations of code and bug reports, evaluating more programming languages and repositories, exploring additional uncertainty measures and integrated reject-option models such as [10], and integrating confidence indicators into developer tools.

ACKNOWLEDGMENT

The author thanks the project guide and the Department of MCA, Shri Jaykumar Rawal Institute of Technology, Dondaicha, for their guidance and support.

REFERENCES

[1]. J. Zhou, H. Zhang and D. Lo, "Where Should the Bugs Be Fixed? More Accurate Information Retrieval-Based Bug Localization Based on Bug Reports", Proc. ICSE, pp. 14-24, 2012. doi: 10.1109/ICSE.2012.6227210.

[2]. F. Niu, C. Li, K. Liu, X. Xia and D. Lo, "When Deep Learning Meets Information Retrieval-Based Bug Localization: A Survey", ACM Computing Surveys, vol. 57, no. 11, Art. 296, pp. 1-41, 2025. doi: 10.1145/3734217.

[3]. R. K. Saha, M. Lease, S. Khurshid and D. E. Perry, "Improving Bug Localization Using Structured Information Retrieval", Proc. ASE, pp. 345-355, 2013. doi: 10.1109/ASE.2013.6693093.

[4]. K. C. Youm, J. Ahn and E. Lee, "Improved Bug Localization Based on Code Change Histories and Bug Reports", Information and Software Technology, vol. 82, pp. 177-192, 2017. doi: 10.1016/j.infsof.2016.11.002.

[5]. X. Huo, F. Thung, M. Li, D. Lo and S.-T. Shi, "Deep Transfer Bug Localization", IEEE Transactions on Software Engineering, vol. 47, no. 7, pp. 1368-1380, 2021. doi: 10.1109/TSE.2019.2920771.

[6]. Z. Zhu, Y. Li, H. Tong and Y. Wang, "CooBa: Cross-project Bug Localization via Adversarial Transfer Learning", Proc. IJCAI, pp. 3565-3571, 2020. doi: 10.24963/ijcai.2020/493.

[7]. P. Chakraborty, M. Alfadel and M. Nagappan, "BLAZE: Cross-Language and Cross-Project Bug Localization via Dynamic Chunking and Hard Example Learning", IEEE Transactions on Software Engineering, vol. 51, no. 8, pp. 2254-2267, 2025. doi: 10.1109/TSE.2025.3579574.

[8]. C. Guo, G. Pleiss, Y. Sun and K. Q. Weinberger, "On Calibration of Modern Neural Networks", Proc. ICML, PMLR vol. 70, pp. 1321-1330, 2017.

[9]. Y. Geifman and R. El-Yaniv, "Selective Classification for Deep Neural Networks", Advances in Neural Information Processing Systems 30, pp. 4878-4887, 2017.

[10]. Y. Geifman and R. El-Yaniv, "SelectiveNet: A Deep Neural Network with an Integrated Reject Option", Proc. ICML, PMLR vol. 97, pp. 2151-2159, 2019.

[11]. R. El-Yaniv and Y. Wiener, "On the Foundations of Noise-free Selective Classification", Journal of Machine Learning Research, vol. 11, pp. 1605-1641, 2010.

[12]. J. Xin, R. Tang, Y. Yu and J. Lin, "The Art of Abstention: Selective Prediction and Error Regularization for Natural Language Processing", Proc. ACL-IJCNLP, 2021.