IEEE YESIST12 IEngage Track Problem Statement

AI-Based Duplicate Bug Detection, Defect Clustering, Missing Field Analysis, and Report Improvement

2. Abstract

Software teams often receive multiple defect reports describing the same underlying issue. Duplicate defects increase triage effort, fragment engineering discussions, and delay resolution. Incomplete bug reports further slow debugging because key fields such as reproduction steps, expected result, actual result, environment, or logs may be missing. This problem focuses on building a system that detects duplicate or possible duplicate defect reports using similarity search and clustering, assigns cluster IDs, suggests missing fields, and generates an improved defect summary while marking duplicates only when similarity exceeds a defined threshold.

3. Keywords

Duplicate Defect Detection, Bug Report Enhancement, Similarity Search, Embeddings, Vector Database, Clustering, DBSCAN, Defect Triage, QA Automation

4. Introduction

Defect management is central to software quality assurance. As products scale and user bases grow, teams receive bug reports from testers, customers, monitoring systems, and support channels. Many reports are duplicates or near-duplicates, while others lack sufficient diagnostic information. An AI-driven duplicate finder and bug enhancer can reduce triage effort, improve reporting quality, and help teams resolve issues faster.

5. Background and Motivation

The motivation is to support QA analysts, developers, and product teams by consolidating similar defects and improving the quality of new reports. The solution should compare a new defect report against an existing CSV/JSON defect dataset, identify top matches, decide whether it is a duplicate, possible duplicate, or new defect, and generate a clearer report with missing fields highlighted.

6. Problem Statement

Build a system that detects duplicate defect reports, assigns cluster IDs, suggests missing fields, and generates an improved defect summary. The system must only mark duplicates if similarity is above a defined threshold and must show top matches. Inputs include an existing defect dataset with fields such as defect_id, title, description, steps, expected, actual, environment, and logs, along with a new defect report in JSON format.

Scope & Constraints

In Scope:

- Ingestion of defect datasets in CSV or JSON format.
- Text normalization across title, description, steps, expected, actual, environment, and logs.
- Embedding generation and vector database based similarity search.
- Top match retrieval for a new defect report, with up to five closest matches.
- Decisioning into duplicate, possible_duplicate, or new_defect based on thresholds.
- Cluster assignment using DBSCAN or nearest-cluster logic.
- Missing field detection and improved report generation.

Out of Scope:

- Automatically closing defects without human review.
- Determining root cause or fixing source code.
- Replacing defect management tools such as Jira, Bugzilla, or Azure DevOps end-to-end.
- Using similarity alone as the only signal when business rules require manual validation.

7. Scope of the Problem

The problem domain includes the technical and operational activities required to design, implement, evaluate, and demonstrate the proposed solution.

- Ingestion of defect datasets in CSV or JSON format.
- Text normalization across title, description, steps, expected, actual, environment, and logs.
- Embedding generation and vector database based similarity search.
- Top match retrieval for a new defect report, with up to five closest matches.
- Decisioning into duplicate, possible_duplicate, or new_defect based on thresholds.

8. Objectives

- Reduce duplicate defect creation and triage overhead.
- Improve quality and completeness of new bug reports.
- Provide transparent top matches with similarity scores.
- Group related defects into meaningful clusters.
- Suggest missing fields required for efficient debugging.
- Generate clear, safe, and concise defect summaries.

9. Constraints and Assumptions

- Defect descriptions may be inconsistent, incomplete, or written in different styles.
- Similarity thresholds must be tuned to avoid false duplicate marking.
- Logs and environment details may contain noisy or sensitive data.
- Clustering quality depends on dataset size, embedding model, and normalization strategy.
- The system must distinguish exact duplicates from related but distinct defects.
- Improved summaries must not invent technical details not present in the input.

Assumptions:

- Participants may use open-source frameworks, public datasets, or synthetically prepared sample data where appropriate.
- The final solution should include a working prototype, documented architecture, sample inputs/outputs, and measurable evaluation results.
- All generated outputs should remain explainable, traceable, and safe for human review.

10. Significance of the Problem

Academic Relevance

- Supports research in semantic similarity, clustering, information extraction, and software repository mining.
- Enables comparison of vector search, classical NLP, and hybrid duplicate detection methods.
- Provides practical evaluation scenarios for bug report quality enhancement.

Industry and Societal Impact

- Reduces QA and development triage effort by identifying duplicates early.
- Improves defect report completeness and developer productivity.
- Helps teams avoid fragmented tracking of the same issue across multiple tickets.
- Supports faster defect resolution and better release quality.

11. Expected Outcomes

Technical Outputs

- Decision: duplicate, possible_duplicate, or new_defect.
- Top matches up to five records with similarity scores and relevant fields.
- Cluster ID for duplicate or related defect grouping.
- Improved report containing title, summary, and missing fields.
- Confidence score or confidence label for the decision.

Suggested KPIs to Track

Metric / KPI	Expected Measurement / Target Direction
Duplicate Detection Precision	Minimize false duplicate decisions.
Duplicate Detection Recall	Identify most true duplicate or near-duplicate reports.
Top-5 Match Accuracy	Relevant duplicate appears within top five matches.
Missing Field Detection Accuracy	Correctly identifies absent or weak defect fields.
Triage Time Reduction	Lower time spent by QA/dev teams on duplicate analysis.
Summary Usefulness	Improved reports are clear, accurate, and useful for debugging.