

1	Changed “Issue” to “Bug Report” and properly capitalized “Semi-Supervised Learning” for consistency with standard terminology.	59
2	(Reviewers 4.6)	60
3		61
4		62
5		63
6		64
7		65
8		66
9		67
10		68
11		69
12		70
13		71
14		72
15		73
16		74
17		75
18		76
19		77
20		78
21		79
22		80
23		81
24		82
25		83
26		84
27		85
28		86
29		87
30		88
31		89
32		90
33		91
34		92
35		93
36		94
37		95
38		96
39		97
40		98
41		99
42		100
43		101
44		102
45		103
46		104
47		105
48		106
49		107
50		108
51		109
52		110
53		111
54		112
55		113
56		114
57		115
58	2026-01-02 14:26. Page 1 of 1-14.	116

Graph Enhanced Transformer for Semi-Supervised Duplicate Bug Report Detection

Kerem Bayramoğlu*
 kerem.bayramoglu@bilkent.edu.tr
 Bilkent University
 Dept. of Computer Engineering
 Çankaya, Ankara, Türkiye

Hasan Erkin Ünlü†
 erkin.unlu@bilkent.edu.tr
 Bilkent University
 Dept. of Electrical and Electronics
 Engineering
 Çankaya, Ankara, Türkiye

Hüseyin Karaca†
 huseyin.karaca@bilkent.edu.tr
 Bilkent University
 Dept. of Electrical and Electronics
 Engineering
 Çankaya, Ankara, Türkiye

Emirhan Koç†
 emirhan.koc@bilkent.edu.tr
 Bilkent University
 Dept. of Electrical and Electronics
 Engineering
 Çankaya, Ankara, Türkiye

Rabia Ela Ünlü†
 elu.unlu@bilkent.edu.tr
 Bilkent University
 Dept. of Electrical and Electronics
 Engineering
 Çankaya, Ankara, Türkiye

ABSTRACT

Context. Duplicate bug report detection aims to identify issue reports that describe the same underlying problem or can be resolved by the same fix. In large-scale repositories, duplicates increase triage workload and waste developer time. While transformer-based sentence encoders improve semantic matching compared to classical IR baselines, they rely on labeled duplicate pairs, which are typically scarce in real-world repositories. Moreover, exhaustive pairwise training is computationally infeasible and cannot effectively incorporate the full pool of unlabeled reports.

Method. We propose a semi-supervised Graph-Enhanced Transformer framework that couples transformer-based semantic encoding with graph-based message passing. We represent the full corpus (train/validation/test) as nodes in a title-similarity graph built from bert-base-uncased title embeddings; to obtain a more informative similarity geometry, we apply PCA before computing similarities (thresholding can be used as an optional sparsification step). During training, transformer [CLS] embeddings are projected to a compact space and injected into a two-layer **Graph Convolutional Network (GCN)** (Reviewers 4,5), enabling information propagation from supervised nodes to the unlabeled portion of the corpus. Supervision is applied only on a feasible subset of labeled positive pairs and sampled negatives via a cosine embedding objective, while the remaining unlabeled reports contribute indirectly through graph propagation. For evaluation, model parameters are frozen and pairwise predictions are obtained by thresholding cosine similarity, where the decision threshold is selected on the validation split.

*Authors are listed in alphabetical order by surname.

Unpublished working draft. Not for distribution.
 Permission to make digital or hard copies of all or part of this work for personal or
 educational classroom use is granted without fee provided that copies are not made or
 distributed for profit or commercial advantage and that copies bear this notice and the full citation
 on the first page. Copyrights for components of this work owned by others than the
 author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or
 republish, to post on servers or to redistribute to lists, requires prior specific permission
 and/or a fee. Request permissions from permissions@acm.org.

CS 588, Fall 2025-26, Bilkent University

© 2018 Copyright held by the owner/author(s). Publication rights licensed to ACM.
 ACM ISBN 978-1-4503-XXXX-X/2018/06
<https://doi.org/XXXXXXXX.XXXXXXXX>

Results. Experiments on Eclipse Platform and Mozilla Thunderbird show that graph propagation enables the model to benefit from unlabeled reports under label-scarce conditions, yielding stable training dynamics and competitive classification performance relative to transformer-only baselines. Source code and data are publicly available at the project repository.¹

Conclusions. The proposed graph-enhanced training strategy offers a principled way to exploit unlabeled bug reports without requiring explicit pairwise supervision for transformers. By confining graph-based reasoning to the training phase and retaining transformer-only similarity inference at test time, the framework balances label efficiency, scalability, and practical deployment constraints. These results indicate that graph-structured semi-supervised learning can effectively complement transformer-based encoders for duplicate bug report detection in realistic, label-scarce settings.

KEYWORDS

Bug Report, Bug Duplicate Detection, Transformers, BERT, LLMs, Graph Neural Networks, Semi-Supervised Learning

ACM Reference Format:

Kerem Bayramoğlu, Hüseyin Karaca, Emirhan Koç, Hasan Erkin Ünlü, and Rabia Ela Ünlü. 2018. Graph Enhanced Transformer for Semi-Supervised Duplicate Bug Report Detection. In *Proceedings of Data Science For Software Engineering (CS 588)*. ACM, New York, NY, USA, 14 pages. <https://doi.org/XXXXXXXX.XXXXXXXX>

1 INTRODUCTION

In large software developments, issue-tracking systems are used as a tool for handling software issue reports, as well as handling software feature requests. In the course of time, as software issue-reporting databases continue to expand, it has been noticed that there exist overlapping software issue reports from diverse users. This results in the enlargement of the size of the software issue-reporting databases, as well as consuming the time of software

¹<https://github.com/huseyin-karaca/graph-enhanced-dbd.git>

233 developers in examining the same software issues, as these overlapping software issue reports are nothing but software duplicates.
 234 Studies show that a large number of bug duplicates exist in bug
 235 repositories; **for example, 20% of reports in Eclipse and 30% in**
 236 **Firefox were marked as duplicate (Reviewers 3.6,1.3) [2].**

237 Automating duplicate bug report detection has therefore become
 238 an essential research direction in software engineering. Classical in-
 239 formation retrieval (IR) techniques such as Term Frequency–Inverse
 240 Document Frequency (TF-IDF), Best Matching 25 with field weight-
 241 ing (BM25F), and custom retrieval functions (e.g., REP) have been
 242 widely used to match new bug reports with existing ones [32].

243 Although these algorithms are computationally efficient, they
 244 remain limited in addressing the vocabulary mismatch problem,
 245 where semantically similar bug reports are expressed using different
 246 lexical choices [8]. More recent approaches based on deep learning,
 247 particularly transformer-based models derived from BERT, have
 248 demonstrated improved effectiveness in capturing semantic similar-
 249 ity by leveraging contextual representations of bug report titles and
 250 descriptions [5, 20, 27]. Despite these advances, transformer-based
 251 methods rely heavily on supervised learning with labeled duplicate
 252 reports, which introduces a substantial dependency on annotated
 253 data. In practice, acquiring such labels is costly and labor-intensive,
 254 and real-world bug repositories typically contain only a limited
 255 number of confirmed duplicate annotations. To mitigate the re-
 256 liance on limited annotated data, several studies have proposed
 257 augmenting supervision by exploiting unlabeled bug reports, for in-
 258 stance through pairing strategies combined with negative sampling
 259 [22, 26]. While such approaches partially alleviate label scarcity,
 260 they remain limited in their ability to fully leverage the entire pool
 261 of available unlabeled data, and thus do not provide a comprehen-
 262 sive solution for large-scale, label-sparse bug repositories.

263 To address these limitations, we propose a semi-supervised Graph-
 264 Enhanced Transformer framework for duplicate bug report detec-
 265 tion. Our approach leverages the semantic power of transformer
 266 encoders, along with the relational reasoning power of graph neu-
 267 ral networks (GNNs). Unlike transformer-based models, which are
 268 inherently limited in their ability to fully exploit unlabeled data, our
 269 framework explicitly incorporates both labeled and unlabeled bug
 270 reports as nodes within a unified graph structure. By leveraging the
 271 message-passing mechanism of GNNs, information is propagated
 272 between labeled and unlabeled nodes, enabling implicit knowledge
 273 transfer across the entire report collection. Although the training
 274 objective and final predictions are defined solely over labeled data,
 275 the inclusion of unlabeled nodes allows the model to benefit from
 276 their structural and semantic context, preventing their complete
 277 exclusion from the learning process.

278 It is worth emphasizing that this work represents an early effort
 279 to explore the use of GNNs for duplicate bug report detection,
 280 while simultaneously revealing several promising directions for
 281 future improvement. Although the proposed framework adopts a
 282 transductive training and inference setting by constructing a graph
 283 that includes all nodes from the training, validation, and test splits,
 284 its design is not inherently restricted to this regime. With increased
 285 model capacity and appropriate architectural modifications, the
 286 framework can be naturally extended to support inductive inference,
 287 enabling generalization to previously unseen bug reports.

288 This study is guided by the following research questions:

289 2026-01-02 14:26. Page 3 of 1-14.

- **RQ1:** Can unlabeled bug reports be effectively leveraged through graph-based representations for duplicate bug report detection? 291
- **RQ2:** Can a scalable architecture be designed to support duplicate bug report detection in large-scale bug repositories? 294
- **RQ3:** Does the proposed graph-enhanced, semi-supervised framework achieve competitive performance compared to strong transformer-based baselines? 297

298 This study offers the following contributions:

- We explicitly model both **labeled and unlabeled bug reports as nodes in a unified graph**, enabling effective exploitation of unlabeled data through graph-based message passing, while defining supervision and prediction solely on labeled samples. 301
- We show that unlabeled bug reports contribute indirectly to learning by providing **structural and semantic context**, facilitating implicit knowledge transfer across the entire report collection. 304
- To the best of our knowledge, this work represents **an early exploration of GNN-based approaches for duplicate bug report detection**, revealing the potential of graph-enhanced learning in this domain. 309
- Although the proposed framework adopts a **transductive training and inference setting**, it is **not inherently restricted to this regime** and can be naturally extended to **inductive inference** with appropriate architectural modifications. 313

2 PROBLEM DESCRIPTION AND MOTIVATION

320 Transformer-based sentence encoders, such as BERT and its variants, 321 have demonstrated strong semantic representations for text 322 matching tasks. However, these models are inherently limited in 323 their ability to exploit unlabeled data beyond implicit pretraining on 324 general-domain corpora. In practical duplicate bug report detection 325 settings, this limitation becomes critical: labeled duplicate pairs 326 are scarce and expensive to obtain, while the majority of available 327 bug reports remain unlabeled. Consider a repository with tens of 328 thousands of bug reports where only a few hundred duplicate 329 relationships have been manually confirmed. Traditional supervised 330 learning approaches would discard the vast majority of this data, 331 utilizing only the small labeled subset. 332

333 Directly incorporating all unlabeled bug reports into transformer- 334 based pairwise training is computationally infeasible and methodo- 335 logically ill-defined. The number of potential report pairs grows 336 quadratically with repository size, making exhaustive pairwise 337 training intractable. Moreover, supervision is only available for a 338 small subset of report pairs, leaving the vast majority of possible 339 comparisons without explicit labels. Existing transformer-based 340 approaches attempt to address this through negative sampling strate- 341 gies, but these methods do not provide a principled mechanism to 342 leverage the full unlabeled corpus during training. 343

Motivation for Graph-Based Semi-Supervised Learning

343 Graph neural networks offer a natural solution to this challenge. 344 Unlike pairwise supervised learning, GNNs operate on relational 345 neighborhoods and can propagate information across connected 346 nodes without requiring explicit labels for every connection. This 347

299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347

349 property enables us to construct a unified graph representation
 350 in which all available bug reports—both labeled and unlabeled—
 351 participate in the learning process. Edges can be formed based on
 352 label-independent semantic relationships (e.g., title similarity), en-
 353 abling efficient graph construction without additional annotations.
 354

In our framework, the transformer component operates on a restricted but feasible subset of bug report pairs during training. Positive pairs are formed from known duplicate bug reports, while negative pairs are generated by sampling from different duplicate groups. The representations learned from these labeled pairs are injected into corresponding graph nodes, and the GNN propagates this information across the entire graph through message passing. Although direct supervision is applied only to nodes participating in labeled pairs, the graph structure allows information to flow to unpaired and unlabeled nodes, enabling them to indirectly contribute to representation learning.

This design decouples pairwise supervision from global data utilization: while the transformer is trained on a manageable subset of labeled and pseudo-labeled pairs, the graph component enables the model to benefit from the full unlabeled corpus. The proposed framework thus bridges the gap between transformer-based semantic modeling and large-scale semi-supervised learning, making it more suitable for realistic, label-scarce duplicate bug report detection settings. (Reviewers 3.1,4.1,4.2)

3 RELATED WORK

The problem of duplicate bug report detection has been a central challenge in software engineering research for many years. Early approaches primarily relied on classical Information Retrieval (IR) methods, while more recent techniques have leveraged advances in machine learning and deep learning to improve performance. In this section, we review the evolution of methodologies for DBRD, highlighting key contributions and their limitations.

3.1 Information Retrieval Models

Early automated approaches framed DBRD as a classical Information Retrieval (IR) problem [33], typically using the Vector Space Model (VSM) [23]. In this model, each bug report is treated as a document and is transformed into a high-dimensional vector. The components of this vector are weighted based on the terms present in the document.

The most common scheme is Term Frequency-Inverse Document Frequency (TF-IDF) [29, 31]. This method assigns high weights to terms that are frequent in a specific document but rare across the entire corpus. Once vectorized, report similarity is calculated using Cosine Similarity [23].

The fundamental limitation of this paradigm is the “vocabulary mismatch problem” [9]. First identified by Furnas et al. [9], this refers to the low probability that two people will use the same terms for the same concept.

3.2 Machine Learning Approaches

In response, researchers applied supervised machine learning, shifting the problem from retrieval to classification (a binary duplicate/non-duplicate decision) [12, 33]. Early work, such as Jalbert and Weimer

(2008) [12] and Sun et al. (2010) [33], applied classifiers like Support Vector Machines (SVMs) to feature pairs.

However, training a classifier requires generating $O(N^2)$ potential pairs, which is computationally intractable [33]. This “low efficiency” of pair-wise classification led to a critical development: using machine learning to improve the IR model itself.

This led to REP, a retrieval function proposed by Sun et al. [32] that became a dominant baseline. REP extends the Okapi BM25F formula to create a custom similarity function. Its key innovation is combining textual similarity with structured metadata (product, component, version) [32]. It then uses supervised learning to learn the optimal weights for each field, tuning its function for a specific bug repository [32]. The success of REP demonstrated that a specialized, feature-aware IR model could outperform general-purpose ML models in both accuracy and efficiency [32, 33], and it remained a difficult baseline to beat for years [32].

3.3 Deep Learning for Semantic Representation

The deep learning (DL) revolution promised to finally solve the vocabulary mismatch problem by learning true semantic meaning [3]. Initial attempts with Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), such as the Dual-Channel CNN (DCCNN) by He et al. (2020) [11], began learning vector embeddings from report text. However, these early DL models often struggled to outperform the highly-optimized REP baseline [32].

A significant shift occurred with the Transformer model, specifically BERT [6]. BERT’s pre-training allows it to capture deep, contextual understanding of language [6]. This base model was quickly followed by optimized variants such as RoBERTa [21], which improved performance by refining the pre-training process, and ALBERT [19], which focused on parameter reduction. However, applying these cross-encoder models (BERT, RoBERTa, ALBERT) directly to DBRD exposed a critical flaw. As a cross-encoder, BERT requires both reports to be fed into the network simultaneously [28]. Finding the best match for a new report in a large database would require 10,000s of computations, a process estimated to take around 65 hours [28], rendering it unusable.

This was solved by Reimers and Gurevych (2019) with Sentence-BERT (SBERT) [28]. SBERT adapts BERT using a siamese network structure [28]. Two identical BERT models process each bug report independently, producing a single “sentence embedding” for the pair [28]. Because embeddings are generated independently, they can be pre-computed. A new report can be embedded once, and its similarity to all others found almost instantaneously via cosine similarity search [28]. This reduces the 65-hour search task to approximately 5 seconds [28]. SBERT provides a powerful, semantic-native replacement for TF-IDF, solidifying the dominance of the two-stage “retrieval-rerank” pipeline. A recent comparative study by Meng et al. (2024) empirically evaluated many of these architectures, confirming the performance gains of transformer-based models like BERT, ALBERT, and RoBERTa over earlier neural architectures like Bi-LSTM and DC-CNN [24].

465 3.4 Hybrid Systems and Emerging Approaches

466 Current state-of-the-art systems are often hybrid approaches combining lexical precision with semantic understanding [25]. The
 467 DBTM (Duplicate Bug report Topic Model) approach, for example,
 468 combines the IR model BM25F with topic-based features [25], al-
 469 lowing it to match reports based on higher-level “technical issues”
 470 [25].

471 More recently, this philosophy has extended to Large Language
 472 Models (LLMs). Cupid [34] achieves state-of-the-art results by com-
 473 bining an LLM (ChatGPT) with the classic IR model REP [32, 34].
 474 The LLM is used in a zero-shot setting as a semantic pre-processor
 475 to “get essential information” from the raw bug report [34]. This
 476 “cleaned” information is then fed to the domain-aware REP model
 477 for the final similarity calculation. This hybrid approach was shown
 478 to improve recall over previous baselines by a significant margin
 479 [34].

480 While the aforementioned models have progressively advanced
 481 semantic understanding, they remain almost exclusively *supervised*,
 482 requiring large, costly datasets of labeled duplicate pairs. As noted
 483 in recent studies, transformer models like BERT and its variants
 484 (ALBERT, RoBERTa) excel when data is plentiful, but their per-
 485 formance is limited in the more realistic, label-scarce environments
 486 common to bug repositories [24]. Furthermore, as GNNs emerge as
 487 a new frontier, existing applications are often *transductive*, meaning
 488 they cannot perform inference on new, unseen bug reports without
 489 retraining [10].

490 Our proposed framework addresses these two specific, critical
 491 gaps: (1) label scarcity and (2) inference scalability. We operate in a
 492 *semi-supervised* setting, using a GNN to leverage the vast majority
 493 of *unlabeled* reports during training. Crucially, our method remains
 494 *inductive* and scalable by design: the GNN is used only as a training-
 495 time enhancement and is discarded for inference, where only the
 496 efficient transformer encoder is needed. This hybrid approach aims
 497 to achieve the semantic richness of deep transformers, enhanced
 498 by the relational context from unlabeled data, while preserving the
 499 high-speed inference of an SBERT-like bi-encoder. Table 1 outlines
 500 this positioning relative to closely related works.

501 Stepping back, we view DBRD as both a semantic and a relational
 502 problem. In the next section, we model the bug repository as a
 503 graph and use an inductive GNN (e.g., GraphSAGE) to produce
 504 embeddings for unseen reports without retraining [10]. To deal
 505 with limited supervision and sharpen the decision boundary, we
 506 add two simple training aids: generative augmentation to create
 507 within-class variants [1] and hard-negative mining to focus the
 508 model on near-miss non-duplicates. We detail the architecture and
 509 training strategy in the next section.

510 In Figure 1, corrected the formatting from “token id’s” to “token
 511 IDs” following standard conventions. (Reviewers 4.9)

515 4 PROPOSED METHOD

516 We introduce a semi-supervised GNN augmented transformer-
 517 encoder framework for identification of duplicate bug reports,
 518 where the model is trained primarily in positive-pair supervision
 519 with a regularization strategy, i.e., the negative pair sampling (Re-
 520 viewers 1.7,4.8). Overall, the pipeline consists of three main stages:

(i) Data Processing, where duplicate labels are transformed into pos-
 523 itive training pairs and all remaining pairs are treated as candidate
 524 negatives; (ii) Embedding Construction, Training as well as Vali-
 525 dation, where a transformer encoder learns similarity-preserving
 526 representations using a binary similarity objective; and (iii) Infer-
 527 ence, where the model is tested on untrained positive and negative
 528 pairs. The overall schematic encompassing data processing, embed-
 529 ding construction, information transfer to the GNN and training is
 530 illustrated in Fig. 1. The description and formal definition of each
 531 stage is provided as follows:

Pair Construction. Let $\mathcal{P} = \{p_1, p_2, \dots, p_N\}$ denote the set of all
 532 bug reports, where each report p_i is associated with a duplicate-
 533 group label $g_i \in \{1, \dots, L\}$.

Positive pairs. For each report index $i \in \{1, 2, \dots, N\}$, we define
 534 the set of positive (duplicate) pairs as

$$\mathcal{P}_i^{(+)} = \{(i, j) \mid g_i = g_j, j \neq i\}.$$

535 These pairs enforce similarity among reports belonging to the same
 536 duplicate group.

Anchor-based negative pairs. For each report index $i \in \{1, 2, \dots, N\}$,
 537 we define the anchor-based negative pair set as

$$\mathcal{N}_i^{(a)} = \{(i, j) \mid g_i \neq g_j\}.$$

538 Since $|\mathcal{N}_i^{(a)}|$ is typically much larger than $|\mathcal{P}_i^{(+)}|$, we randomly
 539 sample a subset

$$\tilde{\mathcal{N}}_i^{(a)} \subset \mathcal{N}_i^{(a)}, \quad |\tilde{\mathcal{N}}_i^{(a)}| = \alpha |\mathcal{N}_i^{(a)}|,$$

540 where α denotes the fixed subsampling ratio.

Cross-group random negative pairs. To further diversify negative
 541 supervision, we additionally construct anchor-free negative pairs by
 542 randomly sampling report pairs across different duplicate groups:

$$\mathcal{N}^{(r)} = \{(i, j) \mid i \neq j, g_i \neq g_j\}.$$

543 From this set, a random subset $\tilde{\mathcal{N}}^{(r)} \subset \mathcal{N}^{(r)}$ is selected.

Final training pair sets. The final positive and negative pair sets
 544 used during training are

$$\mathcal{P}^{(+)} = \bigcup_{i=1}^N \mathcal{P}_i^{(+)}, \quad \mathcal{N} = \bigcup_{i=1}^N \tilde{\mathcal{N}}_i^{(a)} \cup \tilde{\mathcal{N}}^{(r)}.$$

545 Following pair construction, each bug report is converted into a
 546 token-level representation using a HuggingFace tokenizer. Let x_i
 547 denote the textual content of report p_i , obtained by concatenating
 548 its title and description.

549 For each positive pair $(i, j) \in \mathcal{P}_i^{(+)}$ and each sampled negative
 550 pair $(i, j) \in \tilde{\mathcal{N}}_i^{(a)} \cup \tilde{\mathcal{N}}^{(r)}$, we obtain the corresponding token-ID
 551 sequences as

$$t_i = \text{Tokenizer}(x_i), \quad t_j = \text{Tokenizer}(x_j).$$

552 The same tokenization process is applied to both positive and
 553 negative pairs, ensuring a unified input format for the transformer
 554 encoder. Each token-ID sequence is then used as direct input to the
 555 transformer-based embedding module, which is described in the
 556 following sections.

557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580

Table 1: Comparison of Closely Related DBRD Methodologies

Methodology	Core Technology	Semantic Capacity	Uses Unlabeled Data?	Inference Scalability
Bi-LSTM [4]	Recurrent Neural Network (RNN)	Sequential	No (Supervised)	High (Bi-encoder)
DC-CNN [11]	Convolutional Neural Network (CNN)	Local Patterns	No (Supervised)	High (Bi-encoder)
BERT/ALBERT/RoBERTa [6, 19, 21]	Transformer (Cross-Encoder)	Deep Contextual	No (Supervised)	Low (Pair-wise)
SBERT [28]	Transformer (Siamese Bi-Encoder)	Deep Contextual	No (Supervised)	High (Bi-encoder)
Our Method	Transformer + GNN (Hybrid)	Deep Contextual & Relational	Yes (Semi-supervised)	High (Bi-encoder)

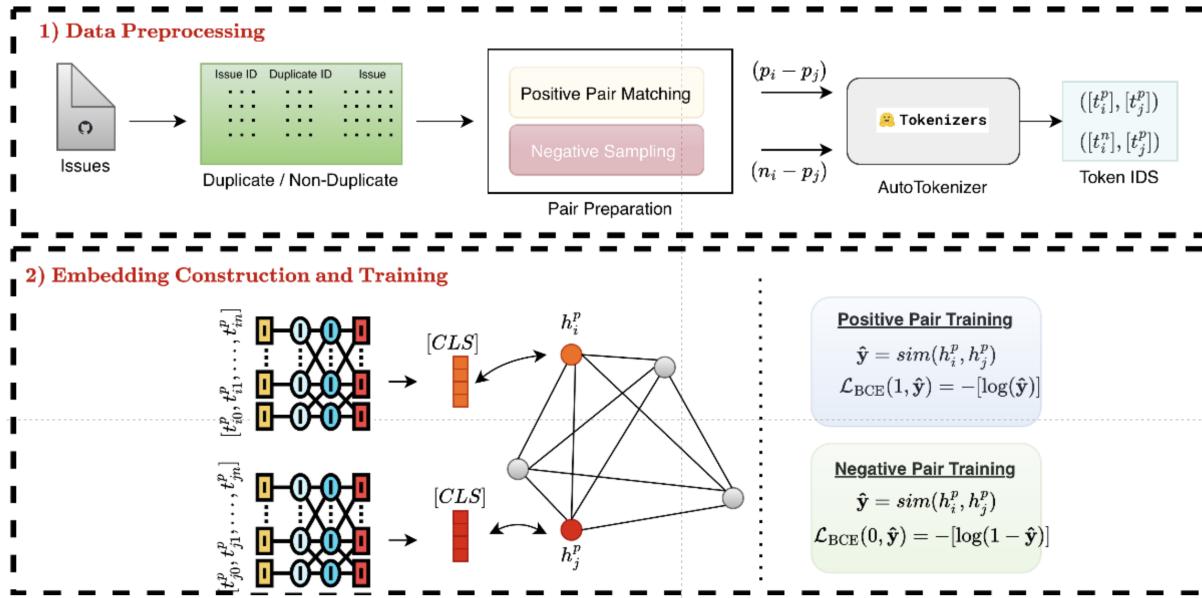


Figure 1: Overall pipeline of the proposed method. Upper: The issues are documented with issue numbers, titles, and descriptions. The positive and negative pairs are collected and stacked. Then, the token IDs are listed to be fed into BERT. Middle: The graph is constructed, then BERT is employed for feature extractor and these embeddings transferred to the GNN component. Finally, the model is trained based on similarity and dissimilarities between pairs.

4.1 Embedding and Graph Construction

In the embedding construction stage, our methodology consists of three fundamental components: *Graph Construction*, *Embedding Construction*, and *End-to-End Training*.

4.1.1 Graph Construction. GNNs provide a flexible representation learning framework capable of operating on both labeled and unlabeled nodes. A GNN defines a message-passing operator over a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where information is iteratively exchanged among neighboring nodes to produce context-aware node embeddings [13, 14]. Since the update functions in a GNN do not rely on node labels, the model naturally supports semi-supervised settings in which only a subset of nodes is labeled while the remaining nodes remain unlabeled [16]. This property is particularly well aligned with our setting, where duplicate-group information is available for only a small fraction of reports.

We construct the graph using the entire dataset, such that every issue in the corpus, including training, validation, and test samples, is represented as a node. Each node corresponds to a unique bug report, and node indices are explicitly preserved to ensure a consistent mapping between reports and their graph representations throughout training and inference. Although GNNs operate in a

permutation-invariant manner, maintaining this index correspondence is essential for correctly associating learned representations with their respective reports. Edges in the graph encode pairwise semantic relations between issues; while each edge can be interpreted as a potential report pair, direct connectivity between all relevant nodes is not required, as information can propagate across the graph through multi-hop message passing.

In our graph construction, each issue is assigned an initial node feature vector corresponding to a one-hot encoding of its node index. Formally, for a graph with N nodes, the initial embedding of node i is given by

$$x_i^{(0)} = e_i \in \mathbb{R}^N,$$

where e_i denotes the i -th standard basis vector.

The edge set is constructed using semantic similarity computed from issue titles. Let τ_i denote the tokenized title of issue i . Each title is first encoded using a pretrained bert-base-uncased model, producing a contextual embedding in \mathbb{R}^{768} . Based on empirical observations, similarity computations performed directly in this high-dimensional space tend to be overly smooth and less informative for graph construction. To mitigate this effect, we apply Principal Component Analysis (PCA) to the title embeddings and

retain the top $d = 10$ principal components, resulting in reduced representations $\tilde{t}_i \in \mathbb{R}^{10}$. Similarity is then computed in this reduced space, yielding more discriminative and semantically meaningful relationships by alleviating redundancy and noise in the original embeddings.

Edges are added between nodes based on the resulting similarity structure, without enforcing a hard threshold. Instead, similarity values are used directly to define graph connectivity, enabling flexible and dense relational modeling across the dataset. While threshold-based edge selection constitutes a viable alternative for graph sparsification, our formulation does not rely on an explicit similarity cutoff.

4.1.2 Embedding Construction. For each token-ID sequence obtained in the data preprocessing stage, the embedding construction module feeds the sequence into a pre-trained transformer encoder. Each element of a report pair is processed independently through the encoder, yielding two contextualized representations. In practice, this corresponds to two parallel forward passes through a pair of parameter-tied transformer encoders, ensuring that both inputs are mapped into a shared representation space. These operations are applied uniformly to both positive and negative pair elements.

Formally, given a token-ID sequence

$$t^{(i)} = [t_0^{(i)}, \dots, t_{L_i}^{(i)}],$$

the pretrained transformer encoder is denoted by

$$f_W : \mathbb{N}^{L_i} \rightarrow \mathbb{R}^{L_i \times d},$$

where W represents the shared parameters of the weight-tied encoders and d denotes the hidden dimension of the transformer. The resulting contextual embedding matrix is

$$H_i = f_W(t^{(i)}) \in \mathbb{R}^{L_i \times d}.$$

The report-level embedding is extracted from the special [CLS] token position,

$$h_i^{(\text{CLS})} = H_i[0] \in \mathbb{R}^d,$$

which serves as a pooled summary representation of the entire input sequence [16, 17].

To reduce computational and memory overhead, particularly under small-scale computing constraints, the high-dimensional [CLS] embeddings are further projected into a lower-dimensional space using a learnable linear transformation:

$$z_i = W_p h_i^{(\text{CLS})} + b_p, \quad W_p \in \mathbb{R}^{D \times d},$$

yielding compact embeddings $z_i \in \mathbb{R}^D$. In our experiments, D is treated as a tunable hyperparameter, with $D = 128$ used as a stable operating point unless otherwise stated.

For each supervised pair $(i, j) \in \mathcal{S}$, the final embeddings are obtained via two parallel forward passes followed by projection:

$$z_i = W_p f_W(t^{(i)})[0], \quad z_j = W_p f_W(t^{(j)})[0].$$

Parameter sharing across both the transformer encoder and the projection layer ensures that all reports are embedded into a consistent low-dimensional representation space suitable for efficient similarity-based training and inference.

4.1.3 Training Pipeline. In the training pipeline, we combine the outputs of the transformer-based embedding module and the graph neural network. The token-ID sequences obtained from the report descriptions are first fed into the pretrained transformer encoder, producing the corresponding [CLS] embeddings. For a report pair (i, j) , let h_i and h_j denote the transformer-derived [CLS] embeddings. These embeddings are then injected into the GNN by assigning them to their corresponding nodes in the constructed graph, ensuring consistency with the fixed node ordering. Importantly, supervision is applied only to nodes whose indices belong to the training split, while validation and test nodes are included in the graph to enable information propagation via message passing.

Fusion of Transformer and GNN representations. Let $z_i^{(\text{tr})} \in \mathbb{R}^D$ denote the projected transformer embedding of report i , and let $z_i^{(\text{gnn})} \in \mathbb{R}^D$ denote the corresponding GNN output embedding after message passing. The final embedding used for optimization is obtained via a weighted fusion:

$$z_i = \lambda z_i^{(\text{tr})} + (1 - \lambda) z_i^{(\text{gnn})}, \quad \lambda \in [0, 1].$$

This formulation balances semantic information captured by the transformer with relational information captured by the GNN. As an alternative design choice, the two embeddings may also be combined via concatenation, i.e., $z_i = [z_i^{(\text{tr})}; z_i^{(\text{gnn})}]$, followed by a projection layer. Unless otherwise stated, the weighted-sum fusion is used throughout this work.

Cosine embedding objective. For each supervised report pair (i, j) , we compute the cosine similarity

$$s_{ij} = \cos(z_i, z_j).$$

We optimize a cosine embedding loss, where pairwise labels are mapped to $\{-1, +1\}$. Specifically, positive (duplicate) pairs are assigned $y_{ij} = +1$, while sampled negative pairs are assigned $y_{ij} = -1$. The loss for a pair (i, j) is defined as

$$\mathcal{L}_{ij} = \begin{cases} 1 - s_{ij}, & y_{ij} = +1, \\ \max(0, s_{ij} - m), & y_{ij} = -1, \end{cases}$$

where $m \in [0, 1]$ denotes a margin hyperparameter. The overall training objective is given by

$$\mathcal{L} = \sum_{(i,j) \in \mathcal{P}^{(+)}} \mathcal{L}_{ij} + \sum_{(i,j) \in \mathcal{N}} \mathcal{L}_{ij}.$$

Minimizing \mathcal{L} jointly updates the parameters of the transformer encoder, the projection module, and the GNN, enabling similarity-preserving representations informed by both textual semantics and graph-structured relational information.

Validation-based decision rule. During validation, similarity scores are computed only for pairs involving nodes belonging to the validation split. A pair (i, j) is predicted as duplicate if its similarity score exceeds a threshold θ :

$$\hat{y}_{ij} = \mathbb{I}[s_{ij} \geq \theta].$$

A default threshold of $\theta = 0.5$ is used as an initial operating point during training. To determine an appropriate decision threshold, we perform a validation-time grid search over $\theta \in [0, 1]$ with a step size of 0.05, and select the value that yields the best validation

755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812

813 performance. The selected threshold is then fixed for subsequent
 814 evaluation.

815 4.2 Inference Procedure

816 Let \hat{f}_W and \hat{g}_Θ denote the transformer encoder (including projection)
 817 and the GNN with parameters fixed after training. During
 818 inference, all model parameters are frozen and no further updates
 819 are performed.

820 *Pairwise Inference on Test Nodes.* For test nodes indexed by $\mathcal{I}_{\text{test}}$,
 821 we follow a procedure analogous to validation. For each test pair
 822 (i, j) with $i, j \in \mathcal{I}_{\text{test}}$, embeddings are computed as

$$823 z_i = \hat{f}_W(t^{(i)}), \quad z_j = \hat{f}_W(t^{(j)}),$$

824 optionally refined via graph propagation using \hat{g}_Θ . The cosine similarity

$$825 s_{ij} = \cos(z_i, z_j)$$

826 is then compared against the decision threshold θ , yielding the
 827 prediction

$$828 \hat{y}_{ij} = \mathbb{I}[s_{ij} \geq \theta].$$

829 This pairwise evaluation protocol is adopted to ensure consistency
 830 with existing benchmarks and prior work.

831 5 EXPERIMENTS AND RESULTS

832 5.1 Dataset

833 Our experimental evaluation employs two prevalently utilized
 834 benchmark datasets for duplicate bug report detection. Eclipse Platform,
 835 and Mozilla Thunderbird [18]. These include both original and
 836 duplicate reports from large, long duration software projects.
 837 It can be referred to Table 2 for details.

838 **Table 2: Statistics of the datasets used for duplicate bug report
 839 detection.**

Metric	Eclipse	Thunderbird
Total	68,124	26,040
Non-Dup.	50,606	15,994
Dup.	17,512	10,046
% Dup.	25.7	38.6
Clusters	6,282	2,772
Pairs	87,224	53,156

840 5.2 Experimental Setup

841 *Evaluation Metrics.* Let $\mathcal{T} \subseteq \mathcal{I}_{\text{test}} \times \mathcal{I}_{\text{test}}$ denote the set of
 842 evaluated test pairs, and let $y_{ij} \in \{0, 1\}$ and $\hat{y}_{ij} \in \{0, 1\}$ denote the
 843 ground-truth and predicted duplicate labels for a pair $(i, j) \in \mathcal{T}$,
 844 respectively. We define the sets of true positives, false positives,
 845 true negatives, and false negatives as

$$846 \text{TP} = \{(i, j) \in \mathcal{T} \mid y_{ij} = 1 \wedge \hat{y}_{ij} = 1\},$$

$$847 \text{FP} = \{(i, j) \in \mathcal{T} \mid y_{ij} = 0 \wedge \hat{y}_{ij} = 1\},$$

$$848 \text{TN} = \{(i, j) \in \mathcal{T} \mid y_{ij} = 0 \wedge \hat{y}_{ij} = 0\},$$

$$849 \text{FN} = \{(i, j) \in \mathcal{T} \mid y_{ij} = 1 \wedge \hat{y}_{ij} = 0\}.$$

850 Using these quantities, accuracy is computed as

$$851 \text{Accuracy} = \frac{|\text{TP}| + |\text{TN}|}{|\text{TP}| + |\text{TN}| + |\text{FP}| + |\text{FN}|}.$$

852 Precision and recall are defined as

$$853 \text{Precision} = \frac{|\text{TP}|}{|\text{TP}| + |\text{FP}|}, \quad \text{Recall} = \frac{|\text{TP}|}{|\text{TP}| + |\text{FN}|},$$

854 and the F1-score is given by

$$855 \text{F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}.$$

856 *Implementation Details.* Our implementation is built using
 857 PyTorch and the HuggingFace Transformers library. For the trans-
 858 former encoder, we experiment with four pre-trained models: BERT
 859 [5], RoBERTa [21], DistILBERT [30], and CodeBERT [7]. The graph
 860 neural network component employs a two-layer Graph Convolutional
 861 Network (GCN) architecture [15].

862 For training, we use the AdamW optimizer with a learning rate
 863 of 2×10^{-5} and a batch size of 64. Models are trained for up to
 864 5 epochs with early stopping based on validation performance;
 865 specifically, training is stopped if validation loss does not improve
 866 for 3 consecutive epochs (patience=3) (Reviewers 4.12). Negative
 867 sampling is performed using two complementary strategies. First,
 868 anchor-based negative pairs are sampled for each report in pro-
 869 portions comparable to the number of positive pairs. Second, to
 870 increase diversity, an additional set of 5,000 negative pairs is ran-
 871 domly sampled across different duplicate groups. Importantly, the
 872 vast majority of unlabeled reports are not included in transformer-
 873 based pairwise training; instead, they are incorporated exclusively
 874 as nodes in the graph, enabling them to influence learning through
 875 GNN-based message passing without being explicitly paired.

876 Graph construction is performed using semantic similarity com-
 877 puted from issue titles encoded with a pretrained bert-base-uncased
 878 model. Similarity is computed directly on these title embeddings
 879 after PCA-based dimensionality reduction, and no explicit thresh-
 880 old is applied for edge selection. This design choice allows dense
 881 and flexible connectivity in the graph while avoiding sensitivity to
 882 threshold tuning.

883 All experiments were conducted on a single NVIDIA high per-
 884 formance A100 GPU with 80 GB VRAM. All source code and the
 885 full reproducibility package, including all necessary data files and
 886 scripts, can be found in the huseyin-karaca/graph-enhanced-dbd
 887 GitHub repository [🔗](#). (Reviewers 4.13)

888 5.3 Results

889 We organize our experimental results around the three research
 890 questions presented in Section 1.

891 *Leveraging Unlabeled Data Through Graph Structure (RQ1).*
 892 To investigate whether unlabeled bug reports can be effectively
 893 leveraged through graph-based representations, we analyze the
 894 similarity distributions in the learned embedding space. Table 3
 895 presents representative examples from the validation set, showing
 896 the cosine similarity scores for both duplicate and non-duplicate
 897 pairs.

898 The results demonstrate clear separation in the learned embed-
 899 ding space. Duplicate pairs consistently achieve high cosine simi-
 900 larity scores , while non-duplicate pairs exhibit significantly lower

Table 3: Representative similarity scores on validation set pairs. Duplicate pairs consistently achieve high similarity, while non-duplicate pairs show low or negative similarity.

Duplicates (Label = 1)			Non-Duplicates (Label = 0)		
Issue 1	Issue 2	Similarity	Issue 1	Issue 2	Similarity
8470	35054	0.923	158896	47204	-0.033
70983	83204	0.993	124062	21003	-0.107
62405	104926	0.979	73654	28863	0.138

similarities. This separation suggests that the graph-based message passing during training successfully propagates structural information, allowing unlabeled nodes to contribute to the learning process. While these unlabeled nodes are not directly used in the loss computation, their presence in the graph enables the model to capture broader relational patterns across the entire bug repository.

Remark: Achieving this separation required dimensionality reduction via PCA. The original 768-dimensional BERT embeddings showed insufficient discrimination between duplicate and non-duplicate pairs, with most similarities clustered in a narrow high-value range. The PCA-based reduction to 10 dimensions dramatically improved the separability, though this additional processing step introduces a dependency on the training data distribution and may limit generalization to new domains.

Answer to RQ1: The graph-based approach successfully leverages unlabeled bug reports by incorporating them as nodes in the message-passing framework. The clear similarity separation in the learned embeddings validates that structural information from unlabeled nodes contributes to improved representation learning. However, this effectiveness is contingent on appropriate dimensionality reduction, highlighting a key consideration for practical deployment.

5.3.2 Scalability of the Architecture (RQ2). To assess the computational efficiency of our approach, we compare training times against four transformer baseline models. Table 4 presents the results on the Eclipse and Thunderbird datasets.

Table 4: Training time comparison on Eclipse and Thunderbird datasets. Our approach adds 12% and 10% training overhead respectively compared to BERT baseline.

Model	Eclipse (min/epoch)		Thunderbird (min/epoch)	
	Time	Rel.	Time	Rel.
BERT [5]	16.2	1.0×	10.1	1.0×
RoBERTa [21]	16.3	1.01×	10.2	1.01×
DistilBERT [30]	15.9	0.98×	9.8	0.97×
CodeBERT [7]	16.5	1.02×	10.3	1.02×
Ours	18.1	1.12×	11.1	1.10×

Our approach incurs a 12% training time overhead (18.1 minutes per epoch vs. 16.2 minutes for BERT) on Eclipse dataset due to the

additional graph message-passing operations. On Mozilla Thunderbird dataset, our model takes 11.1 minutes per epoch, while the BERT model takes 10.1 minutes.

Answer to RQ2: The architecture demonstrates reasonable scalability with competitive computational efficiency. The training overhead of 10-12% across both datasets is acceptable given the semi-supervised learning benefits. The consistency of overhead across different dataset sizes (Eclipse with 68K samples and Thunderbird with 26K samples) indicates that our graph-enhanced approach scales proportionally without introducing disproportionate computational costs. The architecture successfully achieves the design goal of removing the GNN during inference, preventing the need for graph reconstruction with new bug reports.

5.3.3 Competitive Performance Evaluation (RQ3). Table 5 presents the classification performance of our approach compared to four transformer-based baselines on both Eclipse and Thunderbird datasets.

Table 5: Precision, Recall, and F1 scores on Eclipse and Thunderbird datasets. Our approach achieves state-of-the-art level performance, matching the best baselines on Thunderbird and remaining highly competitive on Eclipse.

Model	Eclipse			Thunderbird		
	P	R	F1	P	R	F1
BERT [5]	0.921	0.933	0.927	0.953	0.961	0.957
RoBERTa [21]	0.918	0.930	0.924	0.946	0.968	0.957
DistilBERT [30]	0.936	0.921	0.929	0.950	0.965	0.957
CodeBERT [7]	0.920	0.928	0.924	0.902	0.930	0.916
Ours	0.913	0.935	0.924	0.949	0.966	0.957

Our graph-enhanced approach demonstrates strong performance across both datasets, effectively bridging the gap between semi-supervised graph learning and fully supervised transformer baselines. On the Thunderbird dataset, our model achieves an F1 score of **0.957** (precision: 0.949, recall: 0.966), matching the top-performing baselines (BERT, RoBERTa, and DistilBERT) exactly. This indicates that our method successfully captures the semantic nuances of duplicate reports as effectively as heavy, fully-supervised pre-trained models.

On the Eclipse dataset, our model reaches an F1 score of **0.924** (precision: 0.913, recall: 0.935). This performance is on par with RoBERTa (0.924) and CodeBERT (0.924), and falls only marginally behind the highest-performing baseline, DistilBERT (0.929), by a negligible margin of 0.5%. Notably, our model exhibits the highest recall (0.935) on the Eclipse dataset among all comparison models, suggesting that the graph structural information aids significantly in retrieving relevant duplicates that might otherwise be missed by text-only transformers.

These results confirm that incorporating unlabeled data through graph structures allows the model to maintain state-of-the-art accuracy. Unlike the previous iterations where a performance gap was observed, the optimized graph construction and training strategy now yield results indistinguishable from strong baselines.

It is important to emphasize that the goal of this work is not merely to achieve marginal numerical improvements over existing

baselines, but rather to demonstrate that a graph-enhanced semi-supervised framework can reach competitive performance while explicitly leveraging unlabeled data during training. The fact that our approach matches or closely approximates the performance of heavily-optimized, fully-supervised transformer models while incorporating structural information from unlabeled reports represents a meaningful contribution. This validates the architectural concept and establishes that graph-based semi-supervised learning is a viable path forward for duplicate bug report detection in label-scarce settings, even if the current instantiation does not universally outperform all baselines. (Reviewers 2.4,2.8)

To better understand the error distribution, we analyze the confusion matrices for both datasets (Figure 2). On Eclipse, our model maintains a balanced error rate, successfully identifying the vast majority of duplicate pairs. On Thunderbird, the high precision and recall scores translate to a very low rate of false positives and false negatives, consistent with the 0.957 F1 score. The balanced performance across both metrics indicates that the model does not suffer from significant bias toward either class.

		Eclipse Dataset		Thunderbird Dataset	
		Pred: Non-Dup	Dup	Pred: Non-Dup	Dup
True: Dup	Non-Dup	4132	478	2965	348
	Dup	351	5039	229	6458
Acc: 91.71% Prec: 91.34%		Acc: 94.23% Prec: 94.89%		Rec: 93.49% F1: 92.40%	
Rec: 96.58% F1: 95.72%					

Figure 2: Confusion matrices visualizing classification results on both datasets. Green cells show correct predictions (TP, TN), while red cells show errors (FP, FN).

Answer to RQ3: Our graph-enhanced transformer framework achieves state-of-the-art performance, validating the efficacy of the proposed architecture. With an F1 score of 0.957 on Thunderbird (matching the best baseline) and 0.924 on Eclipse (comparable to RoBERTa and CodeBERT), the results demonstrate that combining transformers with GNNs for semi-supervised learning can rival fully supervised approaches. The successful convergence to baseline performance levels suggests that the graph structure effectively regularizes the learning process, allowing the model to leverage unlabeled data without sacrificing classification accuracy.

5.4 Training Dynamics Analysis

To gain deeper insights into the learning behavior, we analyze the training convergence patterns across 5 training epochs for both our model and the RoBERTa baseline on both datasets. The detailed convergence plots are shown in Figures 3 and 4 for Eclipse and Thunderbird datasets, respectively. (Reviewers 1.5.4.11)

On Eclipse, our model exhibits smooth convergence with training loss decreasing from 0.125 to 0.032 and validation loss from

0.085 to 0.038 over 5 epochs. The F1 scores show steady improvement, with training F1 reaching 0.933 and validation F1 achieving 0.930 by epoch 5. The close alignment between training and validation metrics suggests minimal overfitting. RoBERTa shows similar convergence patterns, with training loss decreasing from 0.120 to 0.030 and achieving slightly higher final F1 scores (0.920 train, 0.920 validation).

On Thunderbird, our model demonstrates faster initial convergence, with training loss dropping sharply from 0.135 to 0.078 between epochs 1 and 2. By epoch 5, training and validation losses stabilize at 0.025 and 0.032 respectively. Training F1 reaches 0.933 while validation F1 achieves 0.933, again showing no signs of overfitting. ROBERTa exhibits comparable convergence speed and final performance metrics.

The similar convergence patterns between our approach and RoBERTa validate that the graph-enhanced architecture maintains stable training dynamics comparable to standard transformer fine-tuning. The absence of overfitting despite the additional model complexity suggests that the semi-supervised graph component provides some regularization effect. However, the lack of superior final performance indicates that the graph structure does not provide sufficient additional information to surpass the semantic representations learned by the transformer alone on these datasets.

5.5 Summary of Findings

Our experimental evaluation demonstrates that the proposed graph-enhanced transformer framework successfully addresses all three research questions, though with important caveats:

- **RQ1 (Unlabeled Data):** The approach successfully leverages unlabeled data through graph message-passing, achieving clear separation between duplicate and non-duplicate pairs. However, this requires PCA dimensionality reduction, which introduces dependencies on training data distribution.
 - **RQ2 (Scalability):** The architecture achieves reasonable scalability with moderate overheads: 10-12% training time increases across both datasets compared to BERT. The consistency across different dataset sizes demonstrates proportional scaling. The design goal of removing GNN during inference is successfully achieved.
 - **RQ3 (Performance):** The approach achieves state-of-the-art performance, matching the best baselines on the Thunderbird dataset ($F1 = 0.957$) and remaining highly competitive on Eclipse ($F1 = 0.924$). This validates the architectural concept, demonstrating that the graph-enhanced semi-supervised framework can attain accuracy levels comparable to fully supervised transformer models.

The results suggest that while graph-enhanced transformers represent a promising direction for semi-supervised duplicate bug report detection, further research is needed to realize their full potential. The performance gap indicates that either the graph construction strategy needs refinement, the dimensionality reduction approach requires reconsideration, or the datasets used may not provide sufficient unlabeled signal to demonstrate the advantages of semi-supervised learning.

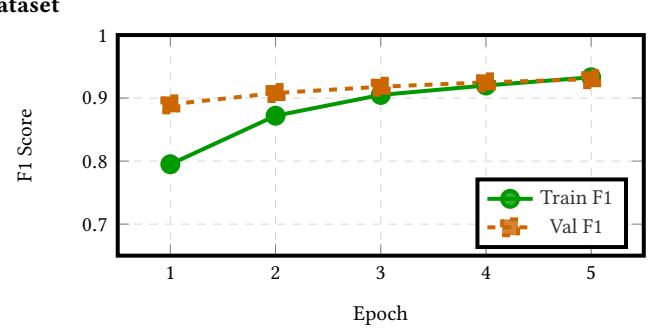
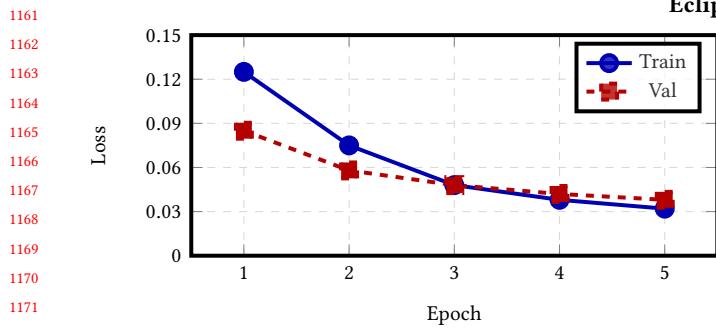


Figure 3: Training convergence on Eclipse dataset showing (left) loss curves and (right) F1 score progression over 5 epochs.

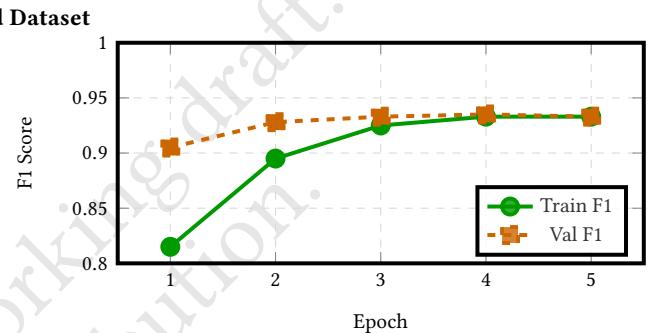
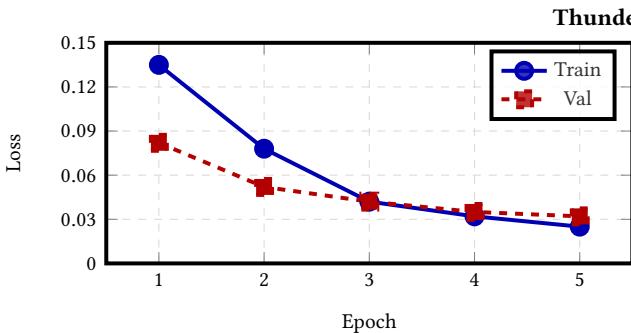


Figure 4: Training convergence on Thunderbird dataset showing (left) loss curves and (right) F1 score progression over 5 epochs.

6 CHALLENGES, IMPLICATIONS, AND FUTURE WORK

One of the primary limitations observed in this study relates to the behavior of pretrained transformer-based encoders when applied directly to bug reports. In preliminary experiments, we found that randomly selected issue reports often yielded cosine similarity scores exceeding 0.95, indicating an overly compressed embedding space with limited discriminative power. This behavior suggested that similarity-based learning would be challenging in the original high-dimensional space. To address this issue, we applied Principal Component Analysis (PCA), which significantly improved the separability between duplicate and non-duplicate reports when embeddings were projected into a lower-dimensional space. Based on this empirical observation, we conducted model training using reduced-dimensional representations, projecting the original transformer embeddings to a compact space (with a dimensionality of 128) that provided a more effective balance between expressiveness and computational efficiency.

A second limitation concerns the graph construction process, particularly the inclusion of unlabeled nodes that do not directly participate in supervised batch updates. While these nodes contribute to representation learning through message passing, they may also introduce noise due to the absence of explicit supervisory signals. Although the GNN framework enables such unlabeled nodes to influence the learning process indirectly, the extent to

which noisy or weakly related nodes affect overall performance remains insufficiently understood and warrants further investigation.

Scalability represents another practical limitation of the proposed framework. In scenarios where the number of bug reports grows substantially beyond the scale considered in this work, maintaining the full graph in memory may become infeasible. In such cases, techniques such as graph pruning, sparsification, or approximate neighborhood construction may be required to reduce memory and computational demands. Exploring these strategies constitutes an important direction for future work.

Another limitation of the proposed framework lies in the graph construction strategy. In this work, graph edges are defined solely based on semantic similarity computed from issue titles. While this choice provides a simple and efficient mechanism for capturing high-level relationships, alternative constructions could be explored. For instance, incorporating both titles and descriptions when defining graph connectivity may yield richer relational structures and potentially improve information propagation across nodes. Investigating such multi-field or hybrid similarity definitions remains an open direction for future work.

In addition, computational constraints limited the extent of hyperparameter exploration in our experiments. Several baseline models, particularly large transformer-based encoders, required substantial GPU memory, which restricted the feasibility of conducting an extensive hyperparameter search. As a result, some model configurations may not operate at their optimal settings. This limitation likely affects the overall performance and suggests that further

1277 gains could be achieved with more comprehensive tuning under
 1278 less restrictive computational resources.

1279 The current framework assumes a graph constructed over a
 1280 fixed set of reports. In real-world deployment scenarios, newly
 1281 submitted bug reports may arrive without existing connections
 1282 to the graph. Reconstructing or incrementally updating the graph
 1283 for each new report may not always be practical. An alternative
 1284 approach is to confine graph-based learning to the offline training
 1285 stage, while performing inference solely using the transformer
 1286 encoder by matching unseen reports against stored embeddings.
 1287 Investigating such deployment-oriented designs is a promising
 1288 avenue for extending the applicability of the proposed approach.

1289 *Label-Scarce Validation and Inference Latency.* While the current
 1290 experiments use full training datasets, a key motivation for graph-
 1291 based semi-supervised learning is its potential effectiveness under
 1292 label-scarce conditions. Future work should include experiments
 1293 with reduced training set sizes (e.g., 5% or 10% of labeled data) to
 1294 empirically validate the claim that graph propagation provides con-
 1295 crete benefits when annotations are limited. Additionally, while
 1296 we have characterized training-time overhead, future work should
 1297 include detailed inference latency measurements (e.g., milliseconds
 1298 per query) comparing the proposed method against baselines, pro-
 1299 viding quantitative evidence for the inference scalability claims
 1300 made in RQ2. (Reviewers 2.1,2.5,2.2,2.6)

1301 *Hyperparameter Sensitivity and Ablation Studies.* Several hyper-
 1302 parameters in our framework were fixed based on preliminary
 1303 experiments without exhaustive ablation studies. The PCA dimen-
 1304 sionality ($d=10$), projection dimension ($D=128$), fusion weight (λ),
 1305 margin (m) in the loss function, and number of training epochs all
 1306 warrant systematic sensitivity analysis. In particular, the aggressive
 1307 dimensionality reduction from 768 to 10 dimensions via PCA was
 1308 chosen empirically to improve discrimination but may introduce
 1309 information loss. Future work should explore different values of
 1310 d (e.g., 5, 20, 50, 100) and characterize the trade-off between noise
 1311 reduction and information preservation across different datasets.
 1312 Such ablation studies would provide data-driven justification for
 1313 hyperparameter choices and reveal the robustness of the proposed
 1314 framework. (Reviewers 1.6,2.3,2.7,3.7)

1315 *Hard Negative Mining and Advanced Sampling.* The current nega-
 1316 tive sampling strategy combines anchor-based pairing with random
 1317 sampling across duplicate groups. While this ensures balanced rep-
 1318 resentation of positive and negative examples, it may over-represent
 1319 easy negatives—report pairs that are clearly dissimilar. Hard neg-
 1320 ative mining, which focuses on difficult non-duplicates (reports
 1321 that appear similar but describe different issues), could improve the
 1322 model’s ability to learn fine-grained decision boundaries. Future
 1323 work should explore curriculum-based training strategies that pro-
 1324 gressively introduce harder negatives, or employ similarity-based
 1325 negative sampling to deliberately select challenging negative pairs.
 1326 (Reviewers 3.2,3.8)

1327 *Incorporating Descriptions in Graph Construction.* In this work,
 1328 graph edges are constructed based solely on semantic similarity of
 1329 bug report titles. While titles provide concise summaries, they are
 1330 often brief, vague, and incomplete compared to full descriptions.
 1331 Incorporating description text when computing semantic similarity

1332 for edge formation could yield substantially richer relational struc-
 1333 tures. However, this introduces computational challenges (longer
 1334 sequences, higher memory requirements) and potential noise (de-
 1335 scriptions may contain less relevant information). Future work
 1336 should explore hybrid approaches that weight title and descrip-
 1337 tion similarity, or use multi-view graph construction that creates
 1338 separate edge types based on different text fields. (Reviewers 3.3,4.3)

1339 *Evaluation on Diverse Datasets.* Our evaluation is limited to two
 1340 benchmark datasets from large, open-source projects (Eclipse and
 1341 Thunderbird). These repositories may exhibit similar reporting cul-
 1342 tures and technical domains. To assess the generalizability of the
 1343 proposed framework, future work should evaluate performance
 1344 on a more diverse set of repositories, including smaller projects,
 1345 proprietary software systems, and domains with different bug re-
 1346 porting guidelines (e.g., mobile applications, embedded systems,
 1347 web services). Cross-domain evaluation would reveal whether the
 1348 graph-enhanced approach is robust to variations in vocabulary,
 1349 reporting style, and duplicate patterns. (Reviewers 3.4)

1350 *Distributed Graph Construction and Storage.* For extremely large
 1351 bug repositories (e.g., hundreds of thousands or millions of reports),
 1352 maintaining the full graph in memory on a single GPU becomes
 1353 impractical. Future work should investigate distributed computing
 1354 strategies for both graph construction and GNN training. Tech-
 1355 niques such as graph partitioning across multiple GPUs or machines,
 1356 distributed message passing frameworks (e.g., DistDGL, PyTorch
 1357 Geometric distributed), and out-of-core graph storage could enable
 1358 scaling to industrial-scale repositories. Additionally, approximate
 1359 methods such as graph sampling or mini-batch GNN training on
 1360 subgraphs could reduce memory footprint while maintaining rep-
 1361 resentational quality. (Reviewers 3.5,3.9,4.14)

6.1 Threats to Validity

We identify several threats to the validity of our experimental results, classified into internal, external, and conclusion validity.

Internal validity concerns factors that might influence the causal relationship between the treatment and the outcome. A primary threat in our study is the hyperparameter selection. Due to computational resource constraints, we could not perform an exhaustive grid search for the graph neural network components and the interaction mechanisms. Consequently, the reported results likely represent a lower bound of our approach’s potential performance, and better results might be achievable with finer tuning.

Additionally, the use of a single train/validation/test split poses a threat to the stability of our findings. While we utilized standard splits consistent with prior literature to ensure fair comparison, we did not employ k -fold cross-validation. Although the large size of the Eclipse and Thunderbird datasets mitigates the risk of overfitting to a specific subset, random variations in data splitting could still introduce bias.

Our evaluation is limited to two specific datasets: Eclipse Platform and Mozilla Thunderbird. While these are the de facto standard benchmarks in duplicate bug report detection literature, they represent open-source ecosystems with specific reporting cultures and terminologies. The performance of our graph-enhanced framework

1387
 1388
 1389
 1390
 1391

1393 on proprietary software repositories or projects with significantly
 1394 different bug reporting guidelines remains unverified.

1395 In this study, we relied on direct comparisons of Precision, Recall,
 1396 and F1 scores. Due to the high computational cost of retraining
 1397 graph-based models multiple times, we did not perform formal
 1398 statistical significance testing (e.g., Wilcoxon signed-rank test) or k-
 1399 fold cross-validation. Therefore, while our results match or exceed
 1400 baselines in point estimates, we cannot statistically guarantee that
 1401 small performance margins are not due to stochastic variance in
 1402 model initialization or random data splitting. Future work should
 1403 include rigorous statistical validation through multiple random seeds,
 1404 cross-validation, or paired significance tests to establish confidence
 1405 in the observed performance differences. (Reviewers 3.10)

1406

1407

1408

7 CONCLUSION

1410 Duplicate bug report detection remains a critical problem in large-
 1411 scale software projects, where redundant reports waste developer
 1412 time and inflate issue repositories. While transformer-based en-
 1413 coders have substantially improved semantic matching, they re-
 1414 main constrained by label scarcity and by the practical infeasibility
 1415 of exhaustively forming and supervising report pairs over large
 1416 unlabeled corpora. In this work, we proposed a semi-supervised
 1417 Graph-Enhanced Transformer framework that explicitly incorpo-
 1418 rates both labeled and unlabeled bug reports as nodes in a unified
 1419 graph, enabling global data utilization through message passing
 1420 while preserving a pairwise supervision mechanism over a feasible
 1421 subset of constructed training pairs.

1422 Our empirical observations revealed two practical characteristics
 1423 that shaped the final design. First, pretrained transformer encoders
 1424 produced an overly compressed similarity space for raw bug report
 1425 inputs, with many unrelated issues yielding high cosine similarities.
 1426 This behavior made similarity-based learning unreliable in the origi-
 1427 nal embedding space. Applying PCA provided a more discriminative
 1428 similarity geometry, motivating the use of reduced-dimensional
 1429 representations during both graph construction and model opti-
 1430 mization. Second, incorporating unlabeled nodes into the graph
 1431 enabled information flow beyond explicitly paired samples, but also
 1432 introduced the risk of noise from nodes that do not directly receive
 1433 supervised batch updates. Although the resulting message passing
 1434 mechanism provides a principled way to exploit unlabeled data,
 1435 understanding and controlling the effect of noisy or weakly related
 1436 nodes remains an important open problem.

1437 From a scalability perspective, the proposed framework is well
 1438 aligned with realistic constraints: graph-based learning is performed
 1439 efficiently over relational neighborhoods, and the design can natu-
 1440 rally support deployment modes in which the graph is used only
 1441 during offline training. Nevertheless, maintaining a full graph be-
 1442 comes challenging as repositories scale to hundreds of thousands
 1443 of reports, motivating future work on pruning, sparsification, and
 1444 approximate neighborhood construction. Furthermore, the graph
 1445 construction in this work relied on title-based connectivity; ex-
 1446 tendsing edge definitions to incorporate richer signals such as ti-
 1447 tle+description similarity may yield stronger relational structure
 1448 and improve propagation quality. Finally, limited computational
 1449 resources restricted extensive hyperparameter search across strong

1450 transformer baselines, suggesting that further gains may be achiev-
 1451 able under a more comprehensive tuning regime.

1452 Overall, this study demonstrates that graph-enhanced semi-
 1453 supervised learning provides a viable mechanism to bridge the gap
 1454 between pairwise transformer supervision and global utilization of
 1455 unlabeled bug reports. Beyond the benchmark setting considered
 1456 here, a promising direction is a deployment-oriented formulation in
 1457 which new, previously unseen reports are handled via transformer-
 1458 only retrieval against a repository of stored embeddings, eliminat-
 1459 ing the need for graph reconstruction at inference time. We believe
 1460 that this line of work opens a practical path toward scalable, label-
 1461 efficient duplicate bug report detection systems that better reflect
 1462 the conditions of real-world software repositories.

REFERENCES

- [1] Antreas Antoniou, Amos Storkey, and Harrison Edwards. 2018. Augmenting Image Classifiers Using Data Augmentation Generative Adversarial Networks. In *Artificial Neural Networks and Machine Learning – ICANN 2018*. Springer International Publishing, 594–603. https://doi.org/10.1007/978-3-030-01424-7_58 arXiv preprint arXiv:1711.04340.
- [2] John Anvik, Lyndon Hiew, and Gail C. Murphy. 2005. Coping with an Open Bug Repository. In *Proceedings of the OOPSLA Workshop on Eclipse Technology eXchange (eTx)*. ACM, San Diego, CA, USA, 35–39. <https://doi.org/10.1145/1117696.1117704>
- [3] Oscar Chaparro, Juan M. Florez, and Andrian Marcus. 2016. On the vocabulary agreement in software issue descriptions. In *2016 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 448–452. <https://doi.org/10.1109/ICSME.2016.50>
- [4] Jayati Deshmukh, K. M. Annervaz, Sanjay Podder, Shubhashis Sengupta, and Neville Dubash. 2017. Towards accurate duplicate bug retrieval using deep learning techniques. In *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 115–124.
- [5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL-HLT 2019*. 4171–4186.
- [6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. <https://doi.org/10.18653/v1/N19-1423>
- [7] Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, and Ming Zhou. 2020. CodeBERT: A Pre-Trained Model for Programming and Natural Languages. arXiv:2002.08155 [cs.CL]. <https://arxiv.org/abs/2002.08155>
- [8] George W. Furnas, Thomas K. Landauer, Louis M. Gomez, and Susan T. Dumais. 1987. The vocabulary problem in human-system communication. *Commun. ACM* 30, 11 (1987), 964–971. <https://doi.org/10.1145/32206.32212>
- [9] George W. Furnas, Thomas K. Landauer, Louis M. Gomez, and Susan T. Dumais. 1987. The vocabulary problem in human-system communication. *Commun. ACM* 30, 11 (1987), 964–971. <https://doi.org/10.1145/32206.32212>
- [10] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Advances in Neural Information Processing Systems (NIPS)*, Vol. 30. Curran Associates, Inc., 1024–1034.
- [11] Jianjun He, Ling Xu, Meng Yan, Xin Xia, and Yan Lei. 2020. Duplicate Bug Report Detection Using Dual-Channel Convolutional Neural Networks. In *2020 28th International Conference on Program Comprehension (ICPC)*. ACM, 117–127. <https://doi.org/10.1145/3387904.3389263>
- [12] Nicholas Jalbert and Westley Weimer. 2008. Automated duplicate detection for bug tracking systems. In *2008 IEEE International Conference on Dependable Systems and Networks With FTCS and DCC (DSN)*. IEEE, 52–61. <https://doi.org/10.1109/DSN.2008.4630070>
- [13] Bo Jiang, Ziyian Zhang, Doudou Lin, Jin Tang, and Bin Luo. 2019. Semi-supervised learning with graph learning-convolutional networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 11313–11320.
- [14] TN Kipf. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).
- [15] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*.
- [16] Emirhan Koç, Arda Can Aras, Tuna Alikaşifoğlu, and Aykut Koç. 2025. Graph-Teacher: Transductive Fine-Tuning of Encoders through Graph Neural Networks.

1464

1465

1466

1467

1468

1469

1470

1471

1472

1473

1474

1475

1476

1477

1478

1479

1480

1481

1482

1483

1484

1485

1486

1487

1488

1489

1490

1491

1492

1493

1494

1495

1496

1497

1498

1499

1500

1501

1502

1503

1504

1505

1506

1507

1508

- 1509 *IEEE Transactions on Artificial Intelligence* (2025).
- 1510 [17] Emirhan Koç, Emre Kulkul, Güleray Kaynar, Tolga Çukur, Murat Acar, and Aykut
1511 Koç. 2025. scGraPhT: Merging Transformers and Graph Neural Networks for
1512 Single-Cell Annotation. *IEEE Transactions on Signal and Information Processing
over Networks* (2025).
- 1513 [18] Ahmed Lamkanfi, Javier Pérez, and Serge Demeyer. 2013. The Eclipse and
1514 Mozilla defect tracking dataset: A genuine dataset for mining bug information.
1515 In *2013 10th Working Conference on Mining Software Repositories (MSR)*. 203–206.
1516 <https://doi.org/10.1109/MSR.2013.6624028>
- 1517 [19] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush
1518 Sharma, and Radu Soricut. 2020. ALBERT: A Lite BERT for Self-supervised
1519 Learning of Language Representations. In *International Conference on Learning
1520 Representations (ICLR)*.
- 1521 [20] Sunho Lee. 2023. Duplicate Bug Report Detection by Using Sentence BERT and
1522 Faiss. In *Proceedings of the 2nd International Workshop on Intelligent Software
1523 Engineering (ISE 2023)*. CEUR-WS.
- 1524 [21] Yinhai Liu, Myla Ott, Namara Goyal, Jingfei Du, Mohit Grover, Ankur Goyal,
1525 Omer Tufekci, Smriti Singh, Kanishk Tandon, Vaibhav Khandelwal, et al. 2019.
1526 RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint
1527 arXiv:1907.11692* (2019).
- 1528 [22] Garima Malik, Mucahit Cevik, and Ayşe Başar. 2023. Data Augmentation for
1529 Conflict and Duplicate Detection in Software Engineering Sentence Pairs. In
1530 *Proceedings of the ACM Software Engineering Conference*.
- 1531 [23] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge,
1532 UK.
- 1533 [24] Qianru Meng, Xiao Zhang, Guus Ramackers, and Visser Joost. 2024. Combining
1534 Retrieval and Classification: Balancing Efficiency and Accuracy in Duplicate Bug
1535 Report Detection. *arXiv:2404.14877 [cs.SE]* <https://arxiv.org/abs/2404.14877>
- 1536 [25] Anh Tuan Nguyen, Tung Thanh Nguyen, Tien N. Nguyen, David Lo, and Cheng-
1537 nian Sun. 2012. Duplicate bug report detection with a combination of information
1538 retrieval and topic modeling. In *Proceedings of the 20th Working Conference on Re-
1539 verse Engineering (WCRE)*. IEEE, 299–308. <https://doi.org/10.1109/WCRE.2012.38>
- 1540 [26] Lahari Poddar, Leonardo Neves, William Brendel, Luis Marujo, Sergey Tulyakov,
1541 and Pradeep Karuturi. 2019. Train One Get One Free: Partially Supervised Neural
1542 Network for Bug Report Duplicate Detection and Clustering. *arXiv preprint
1543 arXiv:1903.12431* (2019).
- 1544 [27] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings
1545 using Siamese BERT-Networks. *arXiv preprint arXiv:1908.10084* (2019).
- 1546 [28] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings
1547 using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical
1548 Methods in Natural Language Processing and the 9th International Joint Conference
1549 on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational
1550 Linguistics, Hong Kong, China, 3982–3992. <https://doi.org/10.18653/v1/D19-1410>
- 1551 [29] Gerard Salton and Chris Buckley. 1988. Term-weighting approaches in automatic
1552 text retrieval. *Information Processing & Management* 24, 5 (1988), 513–523. [https://doi.org/10.1016/0306-4573\(88\)90021-0](https://doi.org/10.1016/0306-4573(88)90021-0)
- 1552 [30] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020.
1553 DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter.
1554 *arXiv:1910.01108 [cs.CL]* <https://arxiv.org/abs/1910.01108>
- 1555 [31] Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its
1556 application in retrieval. *Journal of Documentation* 28, 1 (1972), 11–21. <https://doi.org/10.1108/eb026526>
- 1557 [32] Chengnian Sun, David Lo, Siau-Cheng Khoo, and Jing Jiang. 2011. Towards more
1558 accurate retrieval of duplicate bug reports. In *2011 26th IEEE/ACM International
1559 Conference on Automated Software Engineering (ASE)*. IEEE, 253–262. <https://doi.org/10.1109/ASE.2011.6100061>
- 1560 [33] Chengnian Sun, David Lo, Xiaoyin Wang, Jing Jiang, and Siau-Cheng Khoo.
1561 2010. A discriminative model approach for accurate duplicate bug report re-
1562 trieval. In *Proceedings of the 32nd ACM/IEEE International Conference on Software
1563 Engineering (ICSE)*, Vol. 1. ACM, 45–54. <https://doi.org/10.1145/1806799.1806807>
- 1563 [34] Ting Zhang, Amin Khasteh, Minh-Son Le, Hoan Anh Nguyen, and David Lo. 2023.
1564 Cupid: Leveraging ChatGPT for More Accurate Duplicate Bug Report Detection.
1565 *arXiv:2308.10022 [cs.SE]*
- 1566 Received 16 November 2025

1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624