An Agentic AI Approach to End-to-End Bug Resolution

Integrating Issue-Commit Traceability, Explainability, and Automated Fixing

B.Tech Term Project – I (CS47005)

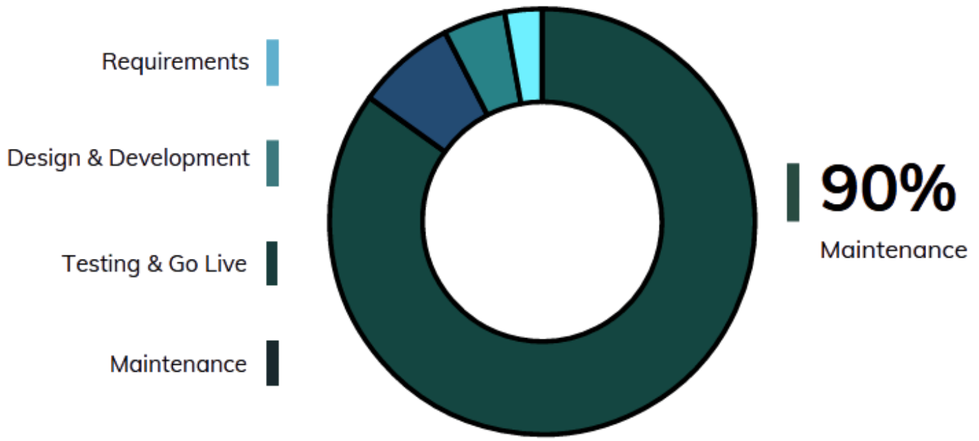
Under the Supervision of
Prof. Partha Pratim Chakrabarti

Presented by
TUHIN MONDAL
22CS10087

Motivation | The Rise of Agentic AI in Software Engineering

Software Engineering (SWE 3.0): Human + AI Agents

- Rise of AI Agents collaborating with human developers
- Automation of complex workflows that required human judgment
- Perform tasks like code review, refactoring, version control etc.
- Automated bug management lifecycle, accelerating software evolution



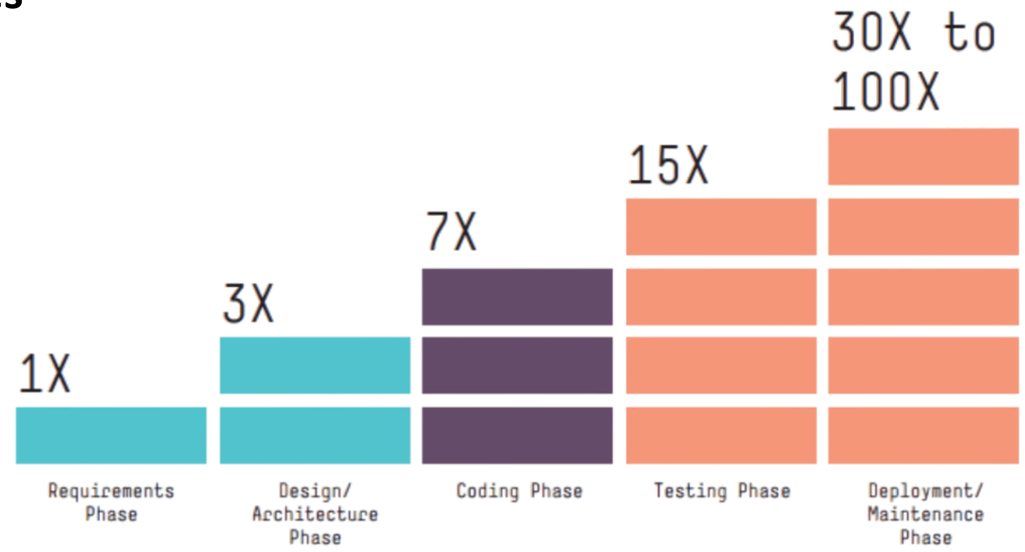
AI Agents for Bug Fixing

Manual bug fixing significant portion of software development expenses

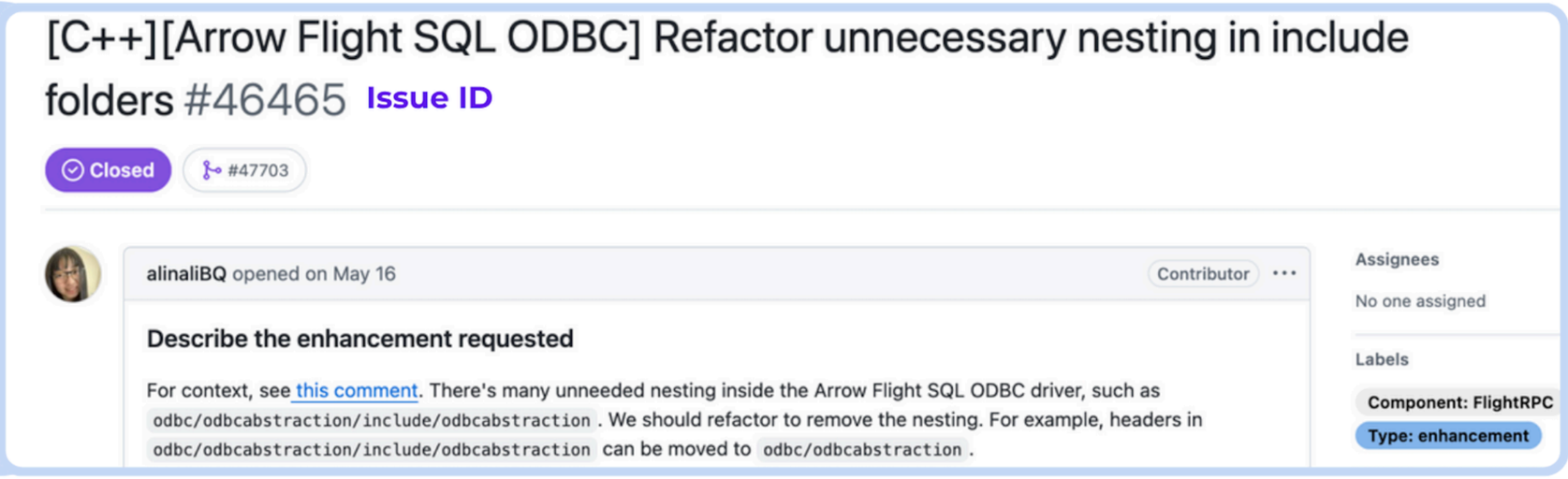
> 25% of development time

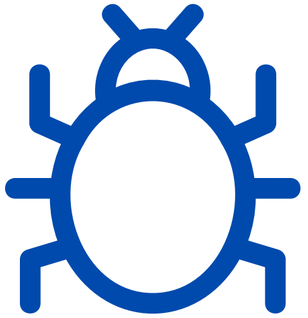
> 90% of maintenance effort

AI Agents like EXPEREPAIR achieves 49.3% Pass@1 on SWE-Bench Lite showing the potential of Agentic AI in reducing Bug Resolution Costs, manual efforts and time efficiency



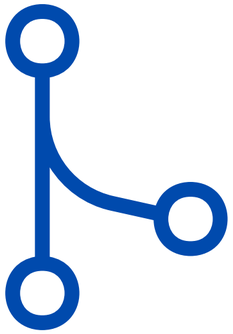
One-to-many Mapping : A gap in existing research





Issue == Bug

GitHub issue #46465 was resolved by a series of three commits bundled in pull request #47703.



Commit == Fix

Literature Review

Software Traceability | Issue–Commit Linking

2009
LINKSTER
Simplified manual inspection via query interfaces.

2014
ReLink
First automated approach using cosine similarity.

2020
RCLinker
Random forest + auto-generated commit messages.

2021
HybridLinker
Ensemble of textual and structural models.

Rule-Based and Heuristic Approaches

2012
MLINK
Used textual + structural features.

2017
FRLink
Included non-source docs to improve recall.

2020
PULink
Used PU learning to handle unlinked pairs.

SCOPE OF WORK

Most studies model the problem as a one-to-one binary classification task. In practice, software issues are resolved across multiple commits, each addressing partial aspects such as refactoring, incremental fixes.

2021
DeepLink
Used code knowledge-graph for semantic linking.

2023
BTLink
Dual BERT encoders with fusion layer.

Deep Learning and Transformer-Based Approaches

2022
T-BERT
Fine-tuned BERT on issue–commit data.

2023
EALink
Used knowledge distillation + contrastive learning.

Literature Review

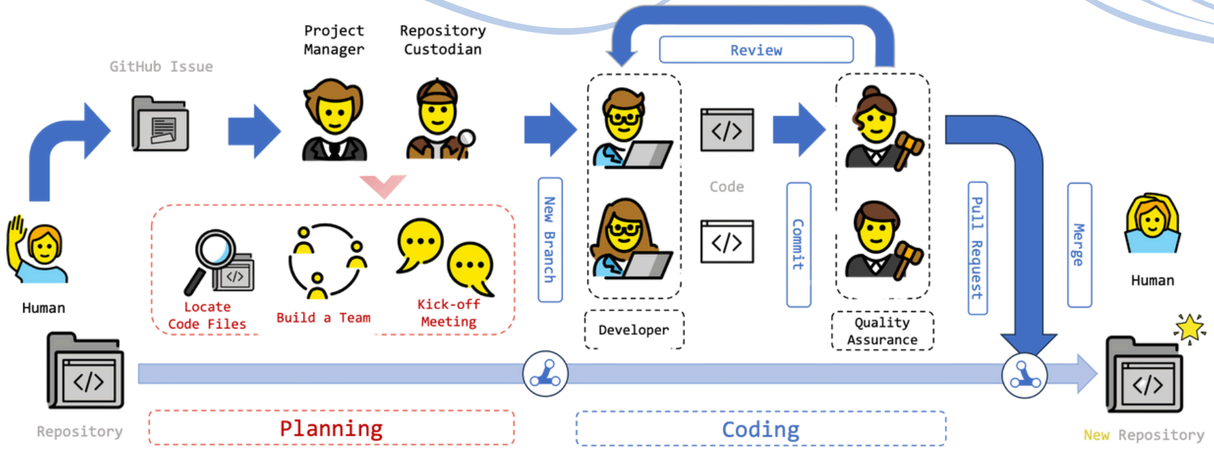
Works on AI Agentic Systems for Bug Resoultion

Model	% Resolved	Org	Date
NEW ExpeRepair-v1.0 + Claude 4 Sonnet	60.33		2025-06-25
Refact.ai Agent	60.00		2025-04-25
NEW KGCompass + Claude 4 Sonnet (20250514)	58.33		2025-09-06
✅ SWE-agent + Claude 4 Sonnet	56.67		2025-05-26
NEW ✅ EntroPO + R2E + Qwen3-Coder-30B-A3B-Instruct	49.67		2025-09-01
ExpeRepair-v1.0	48.33		2025-06-13
✅ SWE-agent + Claude 3.7 Sonnet	48.00		2025-02-26
DARS Agent	47.00		2025-02-05
KGCompass + Claude 3.5 Sonnet (20241022)	46.00		2025-06-19

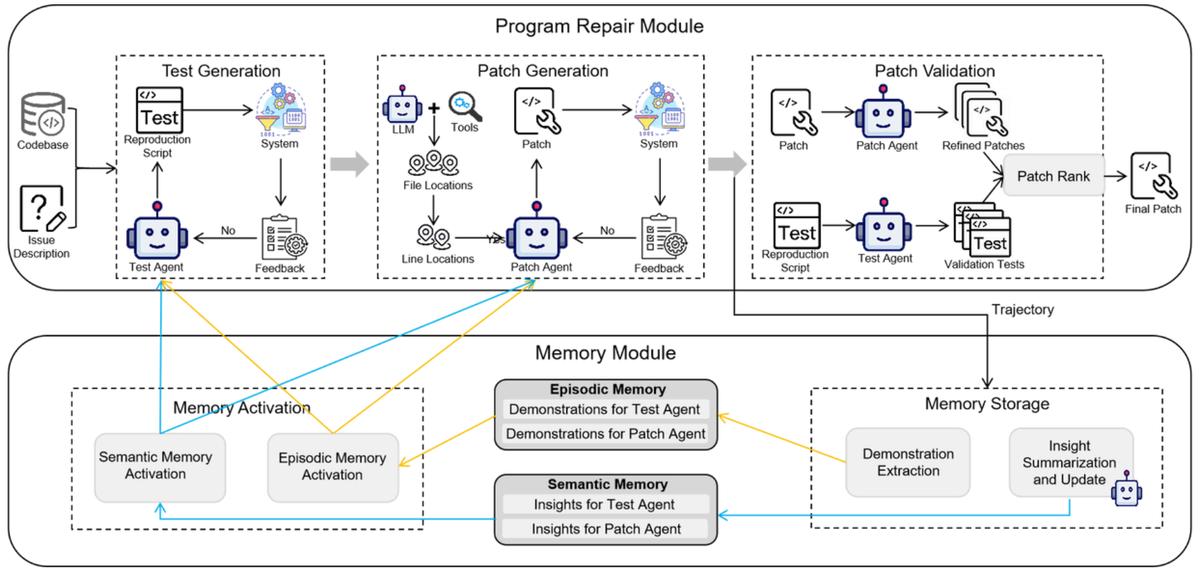
SCOPE OF WORK

- Agentic systems (e.g., MAGIS) enable automated fixing
- Operate without explicit traceability grounding.
- No past knowledge on how resolved issues are fixed

By adding a strong foundational traceability layer we can reuse past knowledge to fix new issues similar to past issues



MAGIS: Multi-Agent GitHub Issue Resolution



ExpeRepair: Dual-Memory Enhanced LLM-based Repo-Level Program Repair

Objectives | Traceability → Explainability → Resolution

3 Stages of building an Agentic Bug Fixing System

Traceability: Accurately detecting and mapping issues to their corresponding code commits

Explainability: Providing interpretable reasoning for the links to build trust and support human understanding

Resolution: Building an agentic framework to propose and submit code changes to resolve a bug

Traceability : The Critical Foundation

Goal: Recovering traceability links (one-to-many)between issues and commits

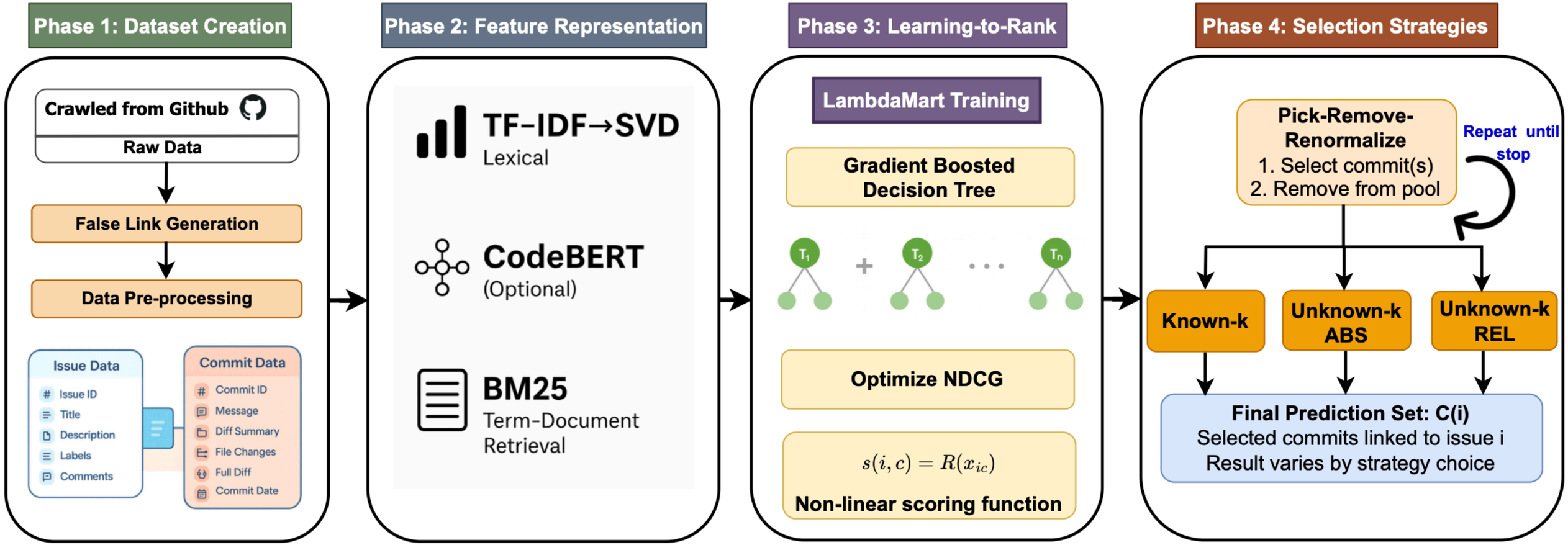
ISSUE		
string	issue_id	Unique identifier for the issue
date	issue_date	Date issue was created
string	title	Title of the issue
string	description	Detailed description of the issue
string	labels	Labels/tags assigned to the issue
string	comments	Discussion or notes on the issue

linked to

COMMIT		
string	commit_id	Unique identifier for the commit
date	commit_date	Date the commit was made
string	message	Commit message
string	diff_summary	Summary of changes
string	file_changes	List of changed files
string	full_diff	Full diff of the commit

LinkRank Workflow

Our approach is organized into four phases. The complete workflow is



I - Dataset Creation

Table 1: Statistics of selected GitHub repositories used for dataset construction.

Project Name	Commits	Issues	Number of Issues with N Commits				
			Two Commits	Three Commits	Four Commits	Five Commits	Six Commits
Apache/Beam	3104	486	206	110	81	54	31
Apache/Datafusion	3534	496	135	132	95	79	52
Apache/Superset	3136	498	230	105	65	56	41
Apache/Mxnet	1216	187	69	49	36	19	14
Apache/Dubbo	1205	201	97	46	22	20	16
Apache/Iceberg	1750	257	97	57	36	39	26
Kubernetes	2948	483	238	102	66	43	31
grpc	2162	486	131	84	51	34	28
TensorFlow	2719	408	167	86	69	45	35
PyTorch	748	125	51	45	14	9	6

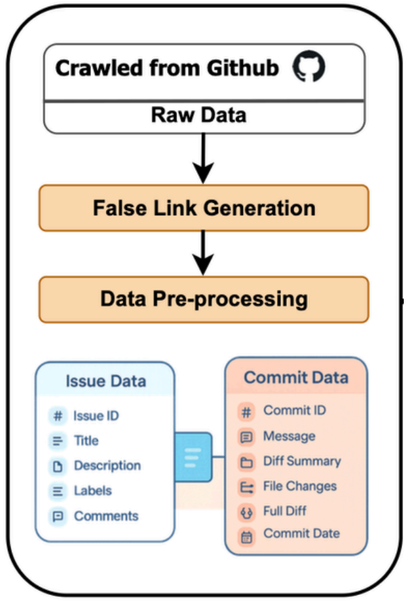
$$C_p = \{c_1, c_2, \ldots, c_m\}, \quad 2 \leq m \leq 6$$

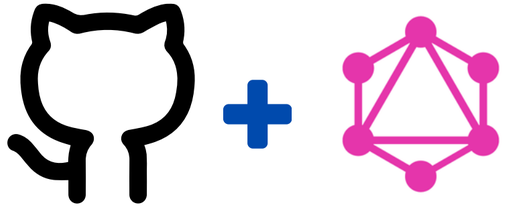
$$T = \{(i_p, c_j) \mid p \in P_{\text{valid}},\ c_j \in C_p\}$$

Table 2: Programming languages in the selected GitHub repositories and their total stars.

Project	Owner	Stars	Java	Python	Rust	C++	Go	JS/TS
Beam	Apache	8.3k	✓	✓	✗	✗	✗	✗
DataFusion	Apache	7.7k	✗	✗	✓	✗	✗	✗
Superset	Apache	67.8k	✗	✓	✗	✗	✗	✓
MXNet	Apache	20.8k	✗	✓	✗	✓	✗	✗
Dubbo	Apache	41.3k	✓	✗	✗	✗	✗	✗
Iceberg	Apache	7.9k	✓	✗	✗	✗	✗	✗
Kubernetes	Kubernetes	117k	✗	✗	✗	✗	✓	✗
gRPC (grpc)	grpc	43.6k	✗	✗	✗	✗	✓	✗
TensorFlow	TensorFlow	191k	✗	✓	✗	✓	✗	✗
PyTorch	PyTorch	93.2k	✗	✓	✗	✓	✗	✗

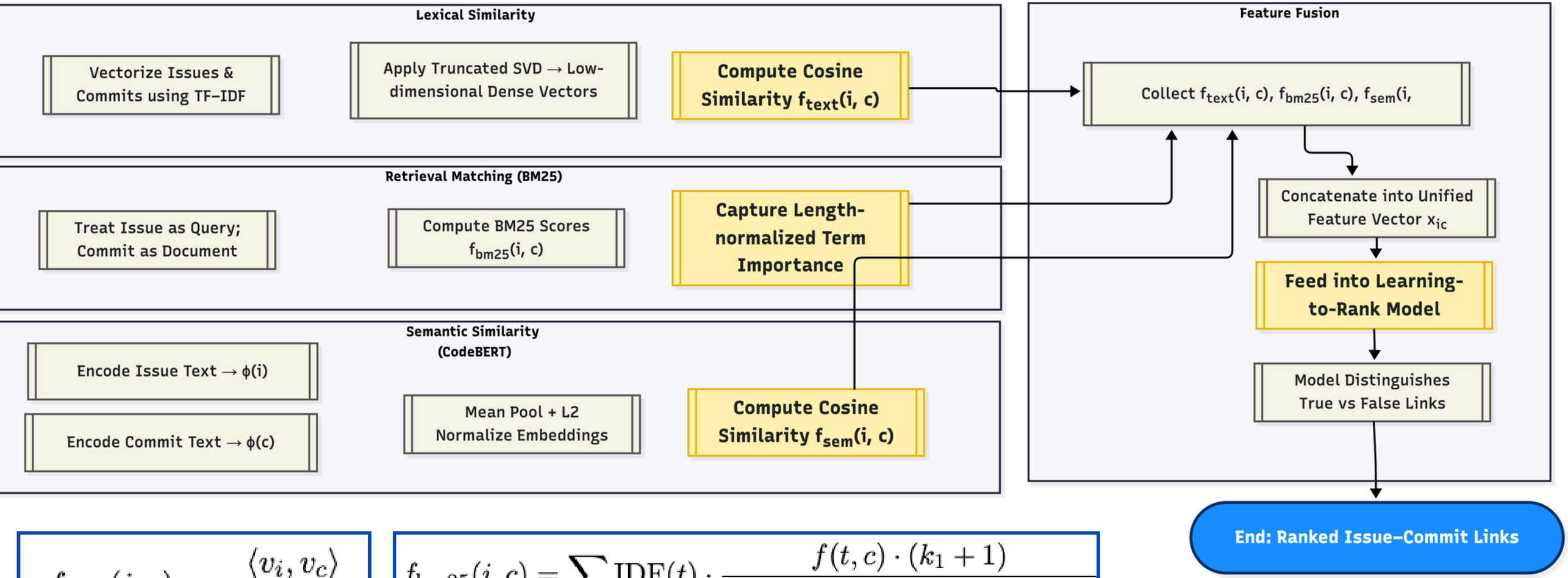
✓ = present / primary language ✗ = not present / minor traces





$$D = T \cup \bigcup_{p \in P_{\text{valid}}} F_p$$

II - Feature Representation

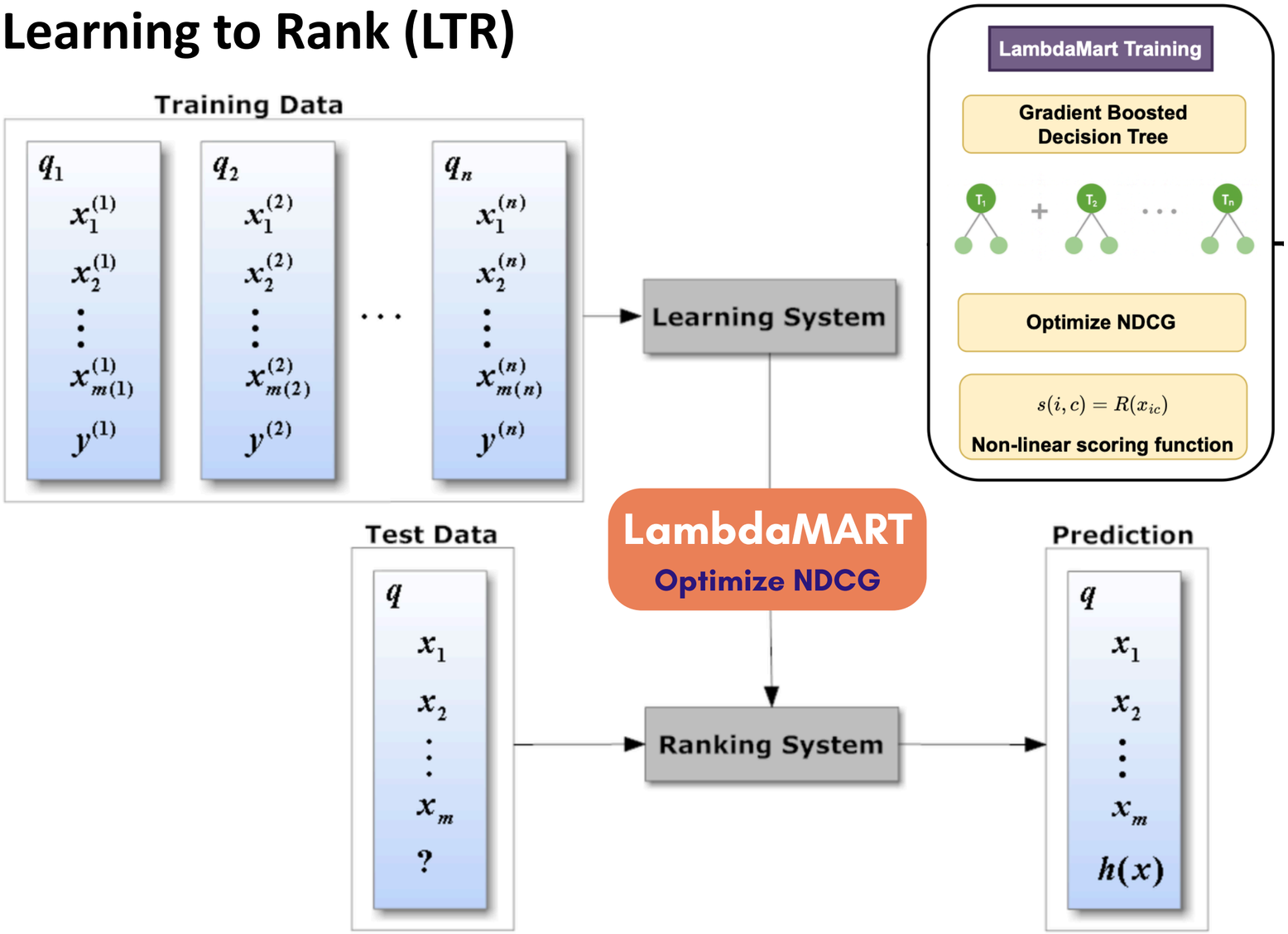


$$f_{\text{text}}(i, c) = \frac{\langle v_i, v_c \rangle}{\|v_i\| \cdot \|v_c\|}$$

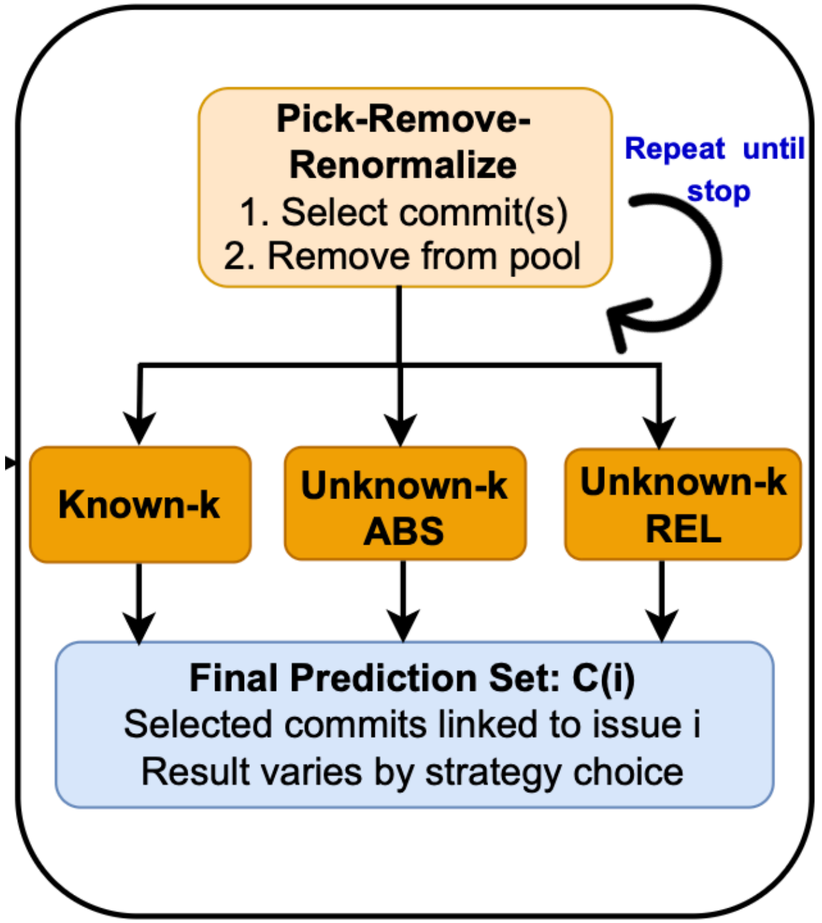
$$f_{\text{bm25}}(i, c) = \sum_{t \in i} \text{IDF}(t) \cdot \frac{f(t, c) \cdot (k_1 + 1)}{f(t, c) + k_1 \cdot \left(1 - b + b \cdot \frac{|c|}{\text{avgdl}}\right)}$$

III - Learning-to-Rank with LambdaMART + Selection Stratergies

Learning to Rank (LTR)



Selection Stratergies



IV- Algorithm: LinkRank

Algorithm 1 LinkRank

Input: Issues $\mathcal{I}$, commits $\mathcal{C}$, labeled pairs $\mathcal{D}$
Output: Predicted links $\{\widehat{\mathcal{C}}(i)\}_{i\in\mathcal{I}}$
Build corpora $S_{\mathcal{I}}, S_{\mathcal{C}}$; compute TF–IDF+SVD and BM25 features; *(optional)* CodeBERT features
for all $(i, c, y) \in \mathcal{D}$ **do**
    Build feature $x_{i,c}$
end for
Train LambdaMART ranker $\mathcal{R}$ grouping by **Issue ID**
for all $i \in \mathcal{I}$ **do**
    for all $c \in \mathcal{C}$ **do**
        $s(i, c) \leftarrow \mathcal{R}(x_{i,c})$
    end for
    Compute per-issue normalized scores $\tilde{s}(i, c)$ via min–max; let $s_{\max}(i) \leftarrow \max_c s(i, c)$
    Iterative selection: repeat
        Pick $c^\star \in \arg\max_c s(i, c)$ if admissible by the chosen policy
        Add $c^\star$ to $\widehat{\mathcal{C}}(i)$; remove $c^\star$ from the candidate pool
        Recompute $\tilde{s}(i, c)$ (min–max) and update $s_{\max}(i)$
    until stopping criterion is met
    Selection policies:
        Known-K: select exactly the top K commits
        Unknown-K (ABS): select all c with $\tilde{s}(i, c) \geq \tau$
        Unknown-K (REL): select all c with $s(i, c) \geq \gamma \cdot s_{\max}(i)$
end for

1,000,000 Commits?

(Very Slow)

V - Algorithm: LinkRank-C2I (Bi-directional Variant)

Algorithm 2 LinkRank-C2I

Input: Issues $\mathcal{I}$, commits $\mathcal{C}$, labeled pairs $\mathcal{D}$
Output: Predicted links $\{\widehat{\mathcal{C}}(i)\}_{i\in\mathcal{I}}$
Build corpora $S_{\mathcal{I}}, S_{\mathcal{C}}$; compute TF–IDF+SVD and BM25 features; *(optional)* CodeBERT features
for all $(i, c, y) \in \mathcal{D}$ **do**
    Build feature $x_{i,c}$
end for
Train commit→issue ranker $\mathcal{R}_{\text{C2I}}$ (group by **Commit ID**) and issue→commit ranker $\mathcal{R}_{\text{I2C}}$ (group by **Issue ID**)
Stage 1 , Commit→Issues (shortlist)
for all $c \in \mathcal{C}$ **do**
    $s_{\text{C2I}}(c, i) \leftarrow \mathcal{R}_{\text{C2I}}(x_{c,i})$ for all $i \in \mathcal{I}$
    shortlist $\mathcal{S}(c) \leftarrow$ top-K issues ranked by s_{C2I}
end for
Stage 2 , Issue→Commits (final selection)
for all $i \in \mathcal{I}$ **do**
    pool $\mathcal{P}(i) \leftarrow \{\, c \in \mathcal{C} : i \in \mathcal{S}(c) \,\}$
    $s(i, c) \leftarrow \mathcal{R}_{\text{I2C}}(x_{i,c})$ for $c \in \mathcal{P}(i)$
    sort by s; let $s_{\max}$ be top; compute per-issue $\tilde{s}$ (min–max)
    Iterative rule: after each pick, remove it from $\mathcal{P}(i)$ and recompute per-issue normalization
    Selection strategies: apply Known-K, Unknown-K ABS, or Unknown-K REL as in Algorithm 1
end for

$\mathcal{R}_{\text{C2I}}$ **(Commit → Issue Ranker):**

$\mathcal{R}_{\text{I2C}}$ **(Issue → Commit Ranker):**

shortlist $\mathcal{S}(c) \leftarrow$ top-K issues

pool $\mathcal{P}(i) \leftarrow \{\, c \in \mathcal{C} : i \in \mathcal{S}(c) \,\}$

Experimental Setup | Configuration + Evaluation Metrics

System & Software Specifications

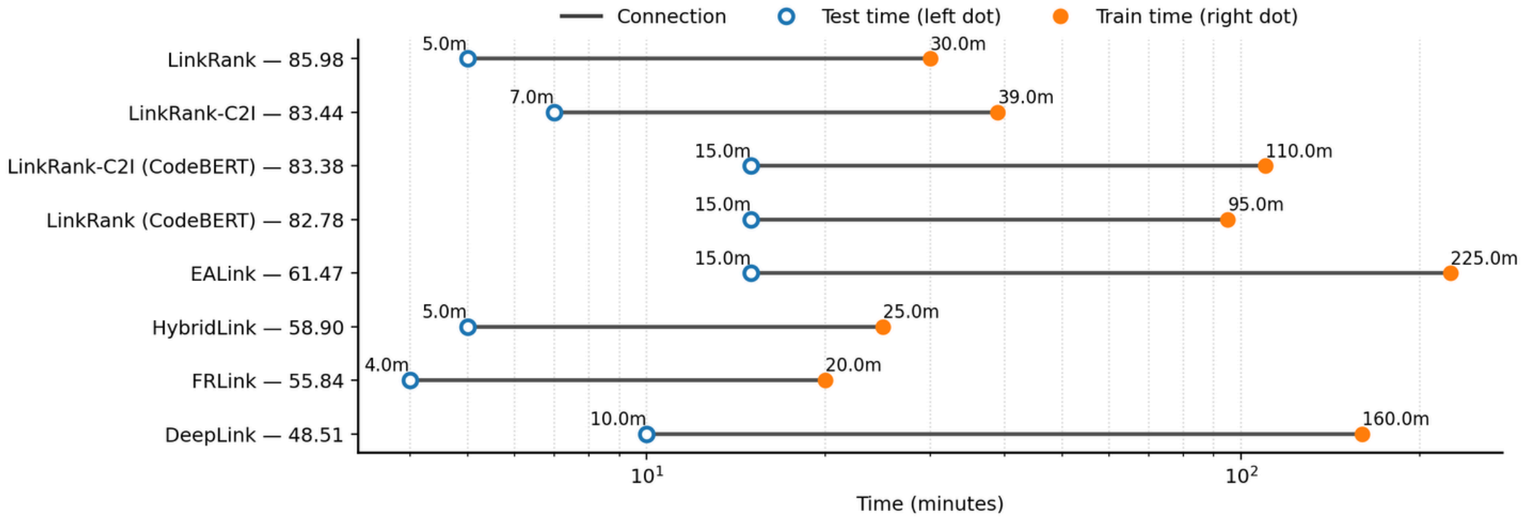
GPU: NVIDIA RTX 4500 Ada Generation (24 GB VRAM)

CPU: Multi-core Intel Processor (RAM: 64 GB)

Operating System: Ubuntu Linux (64-bit)

CUDA: Version 12.5 | **NVIDIA Driver:** Version 555.42.06 | **Core Software:** Python 3.10

Key Libraries: scikit-learn , xgboost HuggingFace Transformers (CodeBERT embeddings)



Key observation: CodeBERT embeddings increases both training and testing costs by 3–4× (e.g., 95–110 minutes of training), but delivers only marginal improvements in F-score.

Evaluation Metrics

$$\text{Precision}^{(i)} = \frac{TP_i}{TP_i + FP_i} \qquad \text{Recall}^{(i)} = \frac{TP_i}{TP_i + FN_i}$$

Precision: "Of all the commits the model claimed are linked to this issue, what fraction were actually correct?"

Recall: "Of all the commits that should have been linked to this issue, what fraction did the model actually find?"

$$\text{Precision} = \frac{1}{|I|}\sum_{i \in I} \text{Precision}^{(i)} \qquad \text{Recall} = \frac{1}{|I|}\sum_{i \in I} \text{Recall}^{(i)}$$

$$\text{F}_\beta^{(i)} = (1+\beta^2)\frac{\text{Precision}^{(i)} \cdot \text{Recall}^{(i)}}{(\beta^2 \cdot \text{Precision}^{(i)}) + \text{Recall}^{(i)}}$$

$$\text{F1}^{(i)} = \frac{2 \cdot \text{Precision}^{(i)} \cdot \text{Recall}^{(i)}}{\text{Precision}^{(i)} + \text{Recall}^{(i)}}$$

$$\text{F1} = \frac{1}{|I|}\sum_{i \in I} \text{F1}^{(i)}$$

Results | Performance of LinkRank and LinkRank-C2I

LinkRank and LinkRank-C2I v/s Baselines

These values are averaged across all datasets.

Models		*Precision*	*Recall*	*F-score*
LinkRank	Known-K	93.05	93.05	93.05
	No-K - ABS	87.04	92.72	88.80
	No-K - REL	83.99	89.92	85.98
LinkRank - C2I	Known-K	92.11	92.11	92.11
	No-K - ABS	84.61	93.16	87.45
	No-K - REL	85.83	83.92	83.44
Baseline	EALINK [34]	57.67	71.73	61.47
	HybridLink [30]	61.70	61.70	58.90
	FRLink [28]	46.43	73.85	55.84
	DeepLink [31]	49.58	51.422	48.51

Across all evaluations, both LinkRank and LinkRank-C2I consistently outperform prior baselines by large margins, often exceeding them by 25–35 points in F-score.

(see Appendix A for more data)

Effectiveness of LR & LR-C2I for O2M Recovery

Evaluation Strategies:

Known-K (Oracle): Assumes 'K' is known. Unrealistic, but gives an upper-bound

Unknown-K (ABS): Uses a global, absolute score threshold. Practical + simple

Unknown-K (REL): Uses a per-issue, relative score threshold. Practical + adaptive

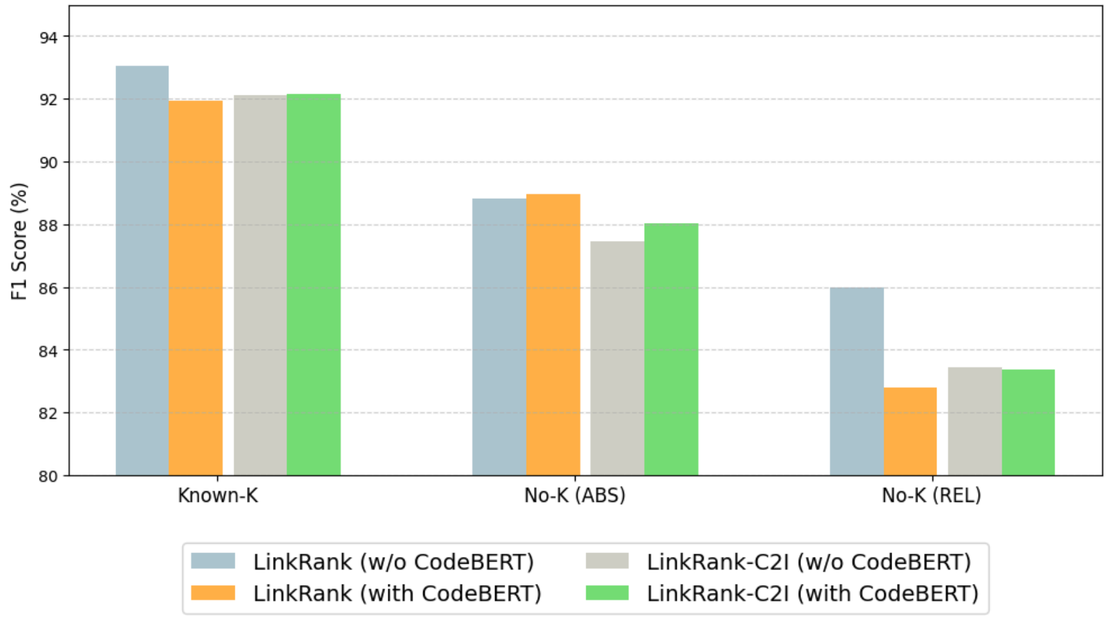
Strategy	LinkRank Performance	LinkRank-C2I Performance
Unknown-K (ABS) (Global Threshold)	Excellent & Stable (e.g., F1: 91.35% on Superset)	Highly Competitive (e.g., F1: 90.64% on Kubernetes)
Unknown-K (REL) (Per-Issue Threshold)	Good (e.g., F1: 87.92% on Superset)	Generally Outperforms LR (e.g., F1: 88.08% on Mxnet)
Key Takeaway	Most robust & simplest approach using a global (ABS) threshold.	Shines in complex scenarios using a relative (REL) threshold to adapt to per-issue diff.

Results | Performance of LinkRank and LinkRank-C2I

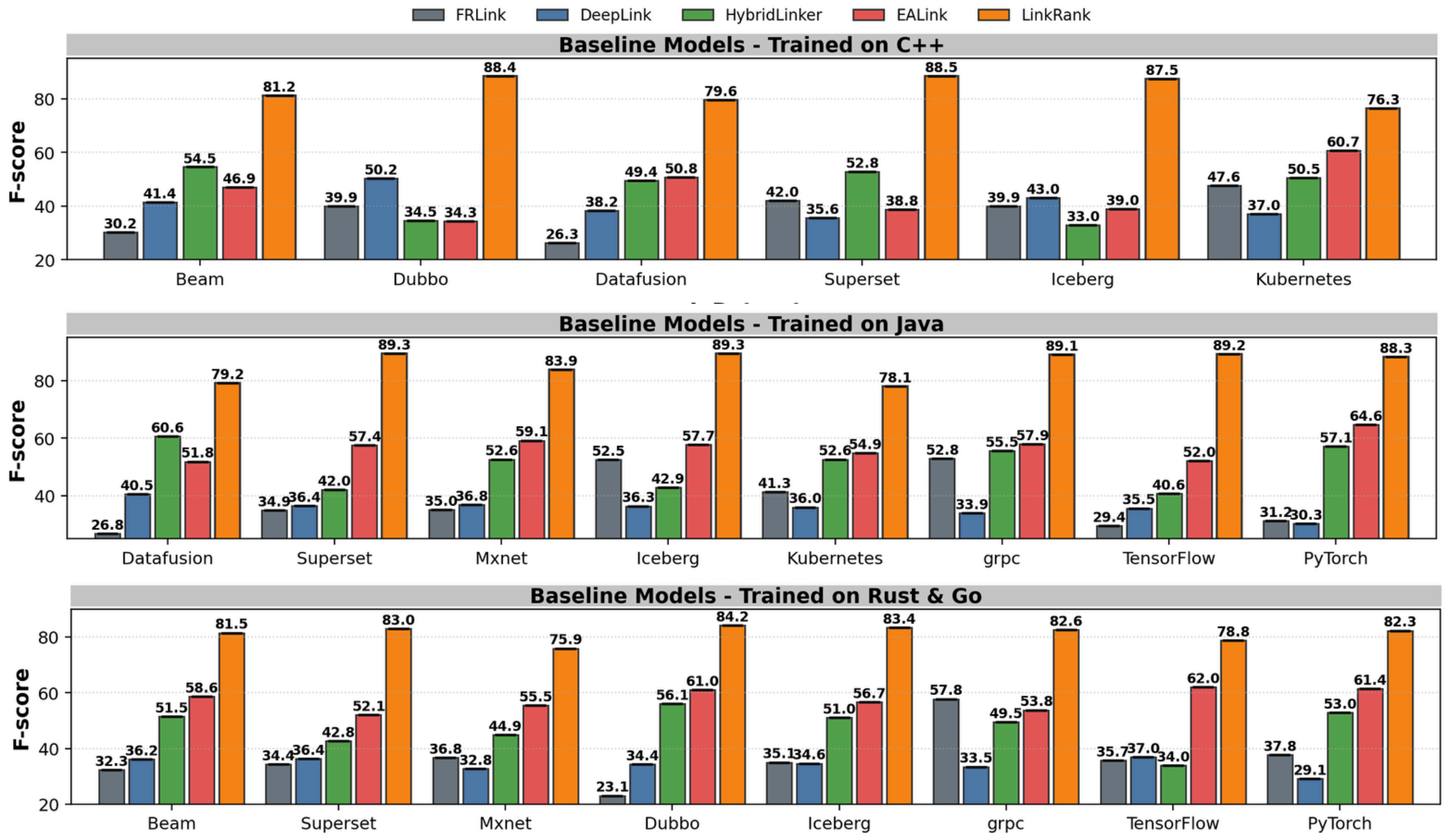
Impact of CodeBERT Embeddings

These values are averaged across all datasets.

Models		*Precision*	*Recall*	*F-score*
LinkRank (CodeBERT)	Known-K	91.93	91.93	91.93
	No-K - ABS	86.75	93.48	88.95
	No-K - REL	86.52	82.26	82.78
LinkRank - C2I (CodeBERT)	Known-K	92.14	92.14	92.14
	No-K - ABS	85.86	92.65	88.04
	No-K - REL	85.38	84.39	83.38



Cross-Language / Cross-Repo Performance

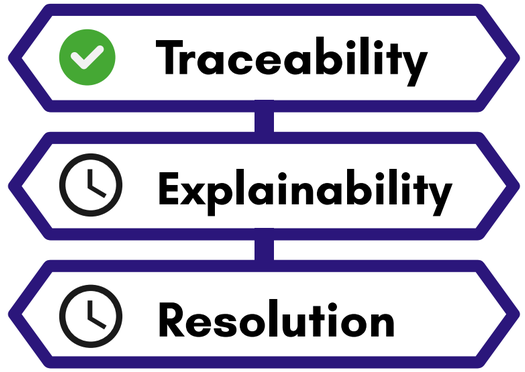


LinkRank remains comparatively more stable and effective across language boundaries than the baseline methods (more data in Appendix B)

Key Contributions of Our Work

- **New One-to-Many Dataset Construction**
- **LinkRank:** Iterative Issue-Centric Ranking Framework
- Efficiency via **IR-Style Features (Low Dependency on CodeBERT)**
- **LinkRank-C2I:** Bidirectional Refinement Pipeline
- **Issue-Wise (Macro) Evaluation Protocol**

Next Plans???



The ultimate goal is to integrate all three components—into a single, closed-loop Agentic AI framework.

Dissemination

LinkRank: A Learning-to-Rank Framework for One-to-Many Issue–Commit Traceability

Abhishek Kumar, Tuhin Mondal, Partha Pratim Das, and Partha Pratim Chakrabarti

[Under Review]

THANK YOU

Appendix - A : Repo wise Results

Table 4: Performance (in %) of LinkRank and LinkRank-C2I

Across 10 datasets under three regimes: Known-K, and Unknown-K with ABS and REL.

Dataset	Variations	LinkRank			LinkRank-C2I		
		Precision	Recall	F-score	Precision	Recall	F-score
Apache/Beam	Known K	91.29	91.29	91.29	90.35	90.32	90.33
	Unknown-K (ABS)	84.67	89.92	86.04	83.24	90.63	85.59
	Unknown-K (REL)	78.15	86.69	81.10	81.59	79.11	78.58
Apache/Datafusion	Known K	93.01	93.01	93.01	91.06	91.06	91.06
	Unknown-K (ABS)	84.38	93.22	87.46	83.99	92.52	86.86
	Unknown-K (REL)	82.16	88.72	84.09	83.50	81.52	80.89
Apache/Superset	Known K	95.04	95.04	95.04	91.89	91.89	91.89
	Unknown-K (ABS)	90.57	93.67	91.35	82.52	91.65	85.28
	Unknown-K (REL)	87.04	90.26	87.92	86.71	83.34	83.82
Apache/Mxnet	Known K	92.01	92.01	92.01	94.31	94.31	94.31
	Unknown-K (ABS)	86.79	92.79	89.14	86.43	94.10	88.98
	Unknown-K (REL)	85.93	88.73	86.63	89.45	88.38	88.08
Apache/Dubbo	Known K	92.72	92.72	92.72	91.42	91.42	91.42
	Unknown-K (ABS)	87.31	93.15	89.23	76.00	95.42	83.54
	Unknown-K (REL)	83.17	91.09	86.11	85.90	88.18	85.79
Apache/Iceberg	Known K	94.22	94.22	94.22	94.56	94.56	94.56
	Unknown-K (ABS)	86.81	94.02	89.52	91.96	93.55	91.97
	Unknown-K (REL)	87.93	91.41	89.22	88.07	85.04	85.40

Table 4: Performance (in %) of LinkRank and LinkRank-C2I

Across 10 datasets under three regimes: Known-K, and Unknown-K with ABS and REL.

Dataset	Variations	LinkRank			LinkRank-C2I		
		Precision	Recall	F-score	Precision	Recall	F-score
Kubernetes	Known K	89.03	89.03	89.03	93.66	93.66	93.66
	Unknown-K (ABS)	81.92	87.13	82.83	87.65	95.92	90.64
	Unknown-K (REL)	73.57	83.12	76.73	89.54	88.32	87.68
grpc	Known K	95.75	95.75	95.75	86.44	86.44	86.44
	Unknown-K (ABS)	89.24	93.47	90.23	83.68	84.99	82.57
	Unknown-K (REL)	87.06	94.46	89.74	78.19	72.00	72.68
Tensorflow	Known K	95.75	95.75	95.75	92.70	92.70	92.70
	Unknown-K (ABS)	89.84	93.75	90.85	83.36	96.86	88.59
	Unknown-K (REL)	87.95	92.59	89.40	86.07	88.31	85.91
Pytorch	Known K	91.70	91.70	91.70	94.73	94.73	94.73
	Unknown-K (ABS)	88.89	96.04	91.39	87.25	96.01	90.48
	Unknown-K (REL)	86.90	92.13	88.89	89.25	85.00	85.59

Appendix - A : Repo wise Results

Table 5: Performance (in %) of LinkRank and LinkRank-C2I with CodeBERT embeddings

Across 10 datasets under three regimes: Known-K, and Unknown-K with ABS and REL.

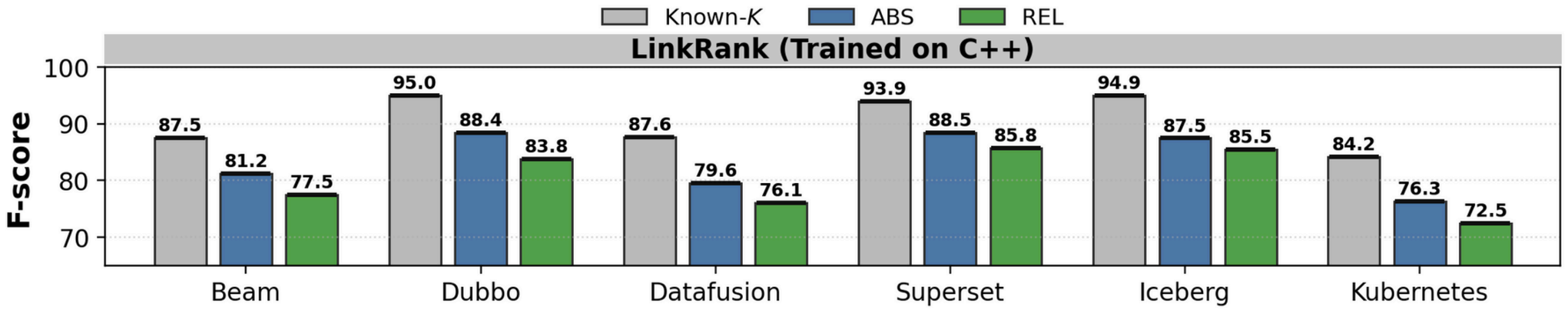
Dataset	Variations	LinkRank			LinkRank-C2I		
		Precision	Recall	F-score	Precision	Recall	F-score
Apache/Beam	Known K	89.03	89.03	89.03	89.48	89.46	89.47
	Unknown-K (ABS)	82.73	91.58	85.53	84.22	88.91	85.14
	Unknown-K (REL)	80.84	76.39	76.57	80.29	76.94	76.66
Apache/Datafusion	Known K	91.45	91.45	91.45	90.65	90.65	90.65
	Unknown-K (ABS)	84.35	93.37	87.69	84.78	92.32	87.43
	Unknown-K (REL)	85.72	79.46	80.62	84.86	80.14	80.49
Apache/Superset	Known K	94.48	94.48	94.48	93.29	93.29	93.29
	Unknown-K (ABS)	90.64	94.51	91.79	88.65	92.28	89.49
	Unknown-K (REL)	89.31	85.88	86.60	87.25	86.50	86.07
Apache/Mxnet	Known K	90.94	90.94	90.94	94.58	94.58	94.58
	Unknown-K (ABS)	87.26	92.57	89.19	86.33	93.59	88.70
	Unknown-K (REL)	85.76	83.55	83.46	87.98	87.34	86.49
Apache/Dubbo	Known K	93.10	93.10	93.10	90.59	90.59	90.59
	Unknown-K (ABS)	88.46	91.81	89.11	82.48	91.28	85.57
	Unknown-K (REL)	87.31	82.64	83.59	84.12	87.22	84.53
Apache/iceberg	Known K	93.16	93.16	93.16	94.61	94.61	94.61
	Unknown-K (ABS)	88.77	93.51	90.08	90.06	94.66	91.61
	Unknown-K (REL)	87.99	85.53	85.59	87.35	85.57	85.41

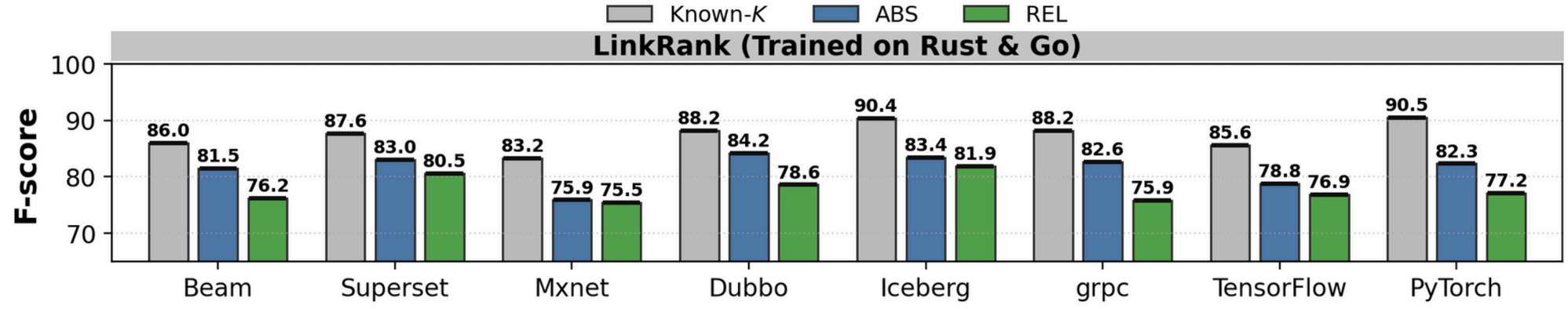
Table 5: Performance (in %) of LinkRank and LinkRank-C2I with CodeBERT embeddings

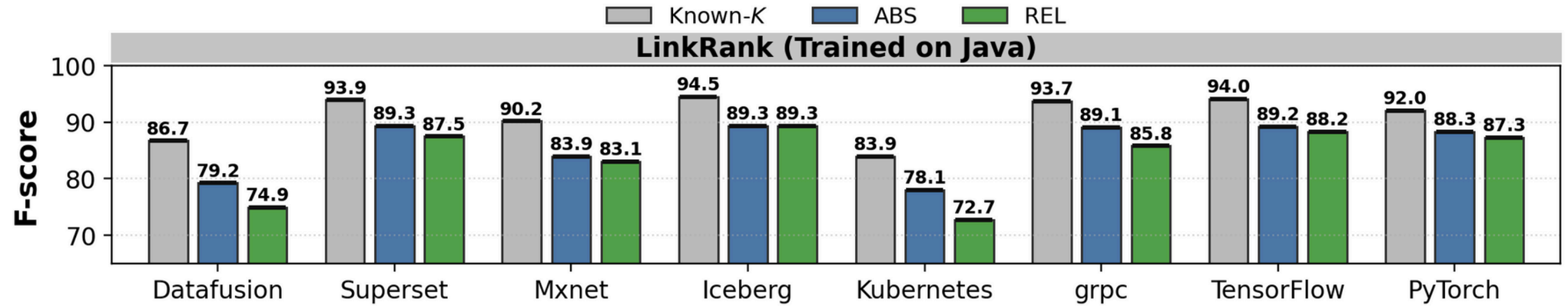
Across 10 datasets under three regimes: Known-K, and Unknown-K with ABS and REL.

Dataset	Variations	LinkRank			LinkRank-C2I		
		Precision	Recall	F-score	Precision	Recall	F-score
Kubernetes	Known K	87.80	87.80	87.80	93.36	93.36	93.36
	Unknown-K (ABS)	81.46	91.46	84.93	88.38	94.63	90.32
	Unknown-K (REL)	81.78	70.22	72.97	88.35	88.15	86.67
grpc	Known K	94.49	94.49	94.49	88.02	88.02	88.02
	Unknown-K (ABS)	88.38	95.40	90.71	81.30	88.62	83.29
	Unknown-K (REL)	90.12	87.45	87.30	78.51	77.16	75.50
Tensorflow	Known K	94.62	94.62	94.62	92.06	92.06	92.06
	Unknown-K (ABS)	87.94	95.70	90.61	82.51	95.07	87.27
	Unknown-K (REL)	89.78	82.09	84.00	84.41	88.15	85.03
Pytorch	Known K	90.19	90.19	90.19	94.78	94.78	94.78
	Unknown-K (ABS)	87.52	94.85	89.89	89.89	95.17	91.62
	Unknown-K (REL)	86.55	89.36	87.09	90.63	86.75	87.00

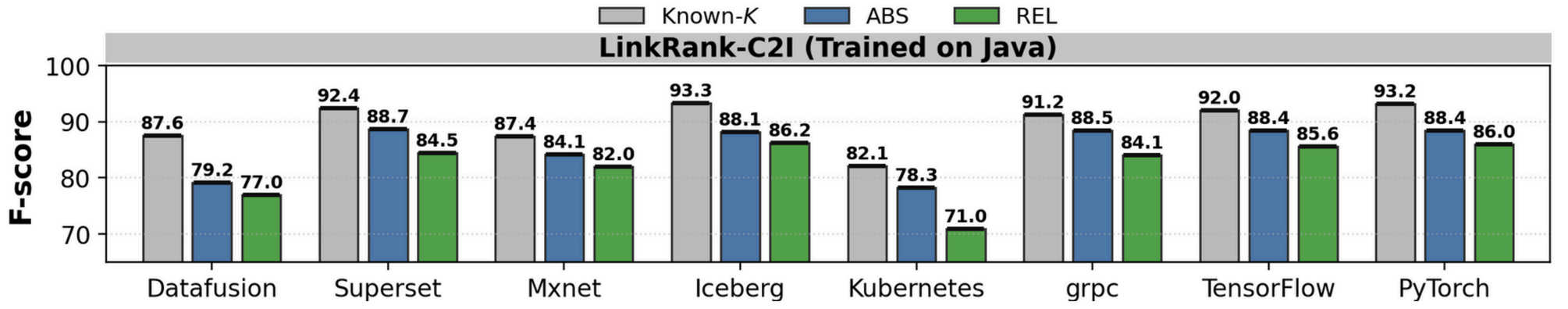
Appendix - B : Cross Repository Performance - LinkRank

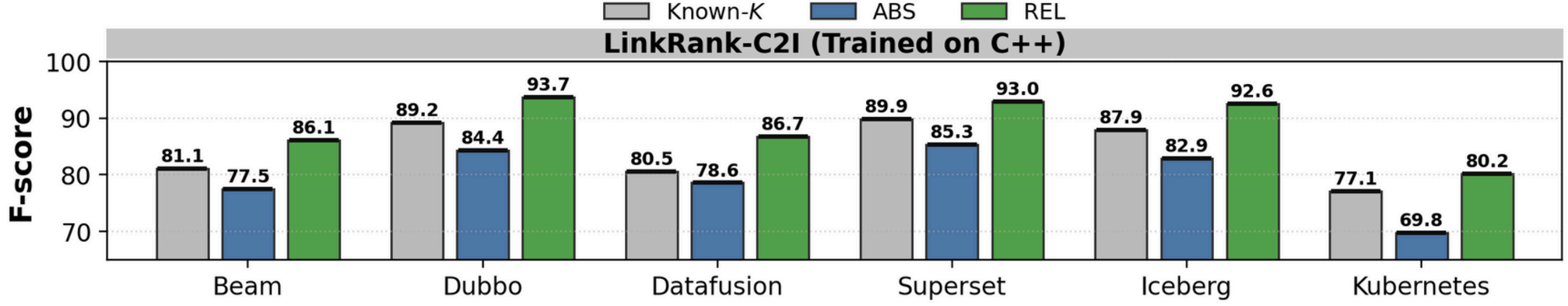


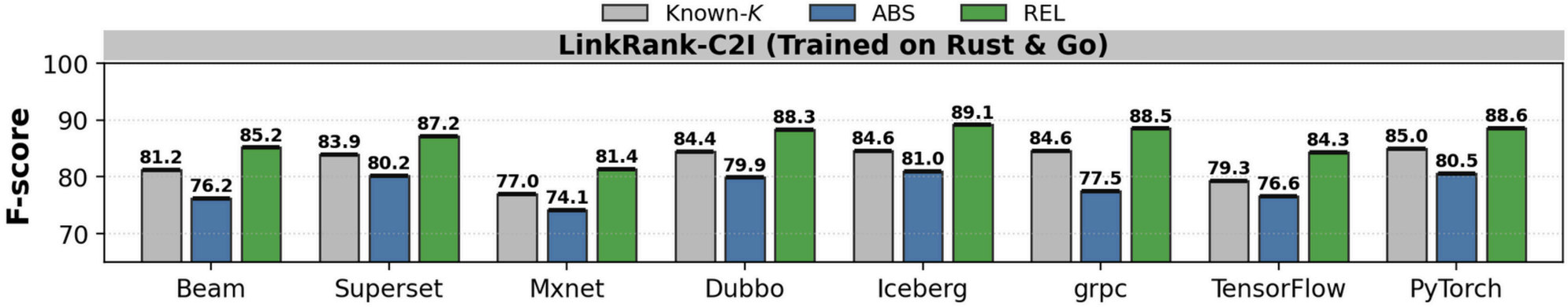




Appendix - B : Cross Repository Performance - LinkRank-C2I







Appendix - C : References

[1] S. Murugesan, "The rise of agentic ai: Implications, concerns, and the path forward," *IEEE Intelligent Systems*, vol. 40, no. 2, pp. 8–14, March-April 2025.

[2] A. E. H. Hao Li, Haoxiang Zhang, "The rise of ai teammates in software engineering (se) 3.0: How autonomous coding agents are reshaping software engineering," 2025.

[3] J. Lin, Y. Liu, Q. Zeng, M. Jiang, and J. Cleland-Huang, "Traceability transformed: Generating more accurate links with pre-trained bert models," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 2021, pp. 324–335.

[4] J. Lan, L. Gong, J. Zhang, and H. Zhang, "Btlink: automatic link recovery between issues and commits based on pre-trained bert model," *Empirical Software Engineering*, vol. 28, no. 4, pp. 1–55, 2023.

[5] A. Bachmann, C. Bird, F. Rahman, P. Devanbu, and A. Bernstein, "The missing links: bugs and bug-fix commits," in *Proceedings of the eighteenth ACM SIGSOFT international symposium on Foundations of software engineering*, 2010, pp. 97–106.

[6] M. Fischer, M. Pinzger, and H. Gall, "Populating a release history database from version control and bug tracking systems," in *International Conference on Software Maintenance, 2003. ICSM 2003. Proceedings.* IEEE, 2003, pp. 23–32.

[7] B. Ghotra, S. McIntosh, and A. E. Hassan, "Revisiting the impact of classification techniques on the performance of defect prediction models," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, vol. 1. IEEE, 2015, pp. 789–800.

Appendix - C : References

[1] S. Murugesan, "The rise of agentic ai: Implications, concerns, and the path forward," *IEEE Intelligent Systems*, vol. 40, no. 2, pp. 8–14, March-April 2025.

[2] A. E. H. Hao Li, Haoxiang Zhang, "The rise of ai teammates in software engineering (se) 3.0: How autonomous coding agents are reshaping software engineering," 2025.

[3] J. Lin, Y. Liu, Q. Zeng, M. Jiang, and J. Cleland-Huang, "Traceability transformed: Generating more accurate links with pre-trained bert models," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 2021, pp. 324–335.

[4] J. Lan, L. Gong, J. Zhang, and H. Zhang, "Btlink: automatic link recovery between issues and commits based on pre-trained bert model," *Empirical Software Engineering*, vol. 28, no. 4, pp. 1–55, 2023.

[5] A. Bachmann, C. Bird, F. Rahman, P. Devanbu, and A. Bernstein, "The missing links: bugs and bug-fix commits," in *Proceedings of the eighteenth ACM SIGSOFT international symposium on Foundations of software engineering*, 2010, pp. 97–106.

[6] M. Fischer, M. Pinzger, and H. Gall, "Populating a release history database from version control and bug tracking systems," in *International Conference on Software Maintenance, 2003. ICSM 2003. Proceedings.* IEEE, 2003, pp. 23–32.

[7] B. Ghotra, S. McIntosh, and A. E. Hassan, "Revisiting the impact of classification techniques on the performance of defect prediction models," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, vol. 1. IEEE, 2015, pp. 789–800.

Appendix - C : References

[8] M. J. Baker and S. G. Eick, "Visualizing software systems," in *Proceedings of 16th International Conference on Software Engineering*. IEEE, 1994, pp. 59–67.

[9] X. Huo, M. Li, Z.-H. Zhou *et al.*, "Learning unified features from natural and programming languages for locating buggy source code." in *IJCAI*, vol. 16, 2016, pp. 1606–1612.

[10] H. Wang, J. Gong, H. Zhang, J. Xu, and Z. Wang, "Ai agentic programming: A survey of techniques, challenges, and opportunities," 2025.

[11] M. Puvvadi, S. K. Arava, A. Santoria, S. S. P. Chennupati, and H. V. Puvvadi, "Coding agents: A comprehensive survey of automated bug fixing systems and benchmarks," in *2025 IEEE 14th International Conference on Communication Systems and Network Technologies (CSNT)*, Bhopal, India, 2025, pp. 680–686.

[12] W. Tao, Y. Zhou, Y. Wang, W. Zhang, H. Zhang, and Y. Cheng, "Magis: Llm-based multi-agent framework for github issue resolution," in *Advances in Neural Information Processing Systems 37 (NeurIPS 2024) – Main Conference Track*, 2024, neurIPS 2024.

[13] F. Mu, J. Wang, L. Shi, S. Wang, S. Li, and Q. Wang, "Experepair: Dual-memory enhanced llm-based repository-level program repair," 2025, preprint / project description provided by the authors; Accessed: 2025-10-28.

[14] H. Huang, R. Widyasari, T. Zhang, I. C. Irsan, J. Shi, H. W. Ang, F. Liauw, E. L. Ouh, L. K. Shar, H. J. Kang, and D. Lo, "Back to the basics: Rethinking issue-commit linking with llm-assisted retrieval," 2025.

[15] N. Serrano and I. Ciordia, "Bugzilla, itracker, and other bug trackers," *IEEE software*, vol. 22, no. 2, pp. 11–13, 2005.

Appendix - C : References

[16] T. Sedano, P. Ralph, and C. Péraire, "The product backlog," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 2019, pp. 200–211.

[17] T. Menzies, B. Turhan, A. Bener, G. Gay, B. Cukic, and Y. Jiang, "Implications of ceiling effects in defect predictors," in *Proceedings of the 4th international workshop on Predictor models in software engineering*, 2008, pp. 47–54.

[18] S. Kim, E. J. Whitehead, and Y. Zhang, "Classifying software changes: Clean or buggy?" *IEEE Transactions on Software Engineering*, vol. 34, no. 2, pp. 181–196, 2008.

[19] C. Ebert, "Classification techniques for metric-based software development," *Software Quality Journal*, vol. 5, no. 4, pp. 255–272, 1996.

[20] E. Yourdon, *Modern structured analysis*. Yourdon press, 1989.

[21] C. Catal and B. Diri, "Investigating the effect of dataset size, metrics sets, and feature selection techniques on software fault prediction problem," *Information Sciences*, vol. 179, no. 8, pp. 1040–1058, 2009.

[22] C. Bird, A. Bachmann, F. Rahman, and A. Bernstein, "Linkster: enabling efficient manual inspection and annotation of mined data," in *Proceedings of the eighteenth ACM SIGSOFT international symposium on Foundations of software engineering*, 2010, pp. 369–370.

[23] R. Wu, H. Zhang, S. Kim, and S.-C. Cheung, "Relink: recovering links between bugs and changes," in *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*, 2011, pp. 15–25.

[24] A. T. Nguyen, T. T. Nguyen, H. A. Nguyen, and T. N. Nguyen, "Multi-layered approach for recovering links between bug reports and fixes," in *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering*, 2012, pp. 1–11.

Appendix - C : References

[25] T.-D. B. Le, M. Linares-Vásquez, D. Lo, and D. Poshyvanyk, "Rclinker: Automated linking of issue reports and commits leveraging rich contextual information," in *2015 IEEE 23rd International Conference on Program Comprehension*. IEEE, 2015, pp. 36–47.

[26] L. F. Cortés-Coy, M. Linares-Vásquez, J. Aponte, and D. Poshyvanyk, "On automatically generating commit messages via summarization of source code changes," in *2014 IEEE 14th International Working Conference on Source Code Analysis and Manipulation*. IEEE, 2014, pp. 275–284.

[27] M. Linares-Vásquez, L. F. Cortés-Coy, J. Aponte, and D. Poshyvanyk, "Changescribe: A tool for automatically generating commit messages," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, vol. 2. IEEE, 2015, pp. 709–712.

[28] Y. Sun, Q. Wang, and Y. Yang, "Frlink: Improving the recovery of missing issue-commit links by revisiting file relevance," *Information and Software Technology*, vol. 84, pp. 33–47, 2017.

[29] Y. Sun, C. Chen, Q. Wang, and B. Boehm, "Improving missing issue-commit link recovery using positive and unlabeled data," in *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2017, pp. 147–152.

[30] P. R. Mazrae, M. Izadi, and A. Heydarnoori, "Automated recovery of issue-commit links leveraging both textual and non-textual data," in *2021 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2021, pp. 263–273.

[31] R. Xie, L. Chen, W. Ye, Z. Li, T. Hu, D. Du, and S. Zhang, "Deeplink: A code knowledge graph based deep learning approach for issue-commit link recovery," in *2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2019, pp. 434–444.

Appendix - C : References

[32] H. Ruan, B. Chen, X. Peng, and W. Zhao, "Deeplink: Recovering issue-commit links based on deep learning," *Journal of Systems and Software*, vol. 158, p. 110406, 2019.

[33] J. Lin, Y. Liu, Q. Zeng, M. Jiang, and J. Cleland-Huang, "Traceability transformed: Generating more accurate links with pre-trained bert models," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 2021, pp. 324–335.

[34] C. Zhang, Y. Wang, Z. Wei, Y. Xu, J. Wang, H. Li, and R. Ji, "Ealink: An efficient and accurate pre-trained framework for issue-commit link recovery," in *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2023, pp. 217–229.

[35] GitHub, "Github copilot — ai pair programmer," 2021, product page; Accessed: 2025-10-28. [Online]. Available: https://github.com/features/copilot

[36] Anthropic, "Claude (including claude code) — helpful, honest, and harmless ai assistants," 2023, product page; Accessed: 2025-10-28. [Online]. Available: https://www.anthropic.com/claude

[37] Cursor, "Cursor — ai for developers (ide-powered coding assistant)," 2023, product page; Accessed: 2025-10-28. [Online]. Available: https://www.cursor.so

[38] Windsurf, "Windsurf — workspace-aware coding assistant," 2024, product page; Accessed: 2025-10-28. [Online]. Available: https://windsurf.ai

[39] GitHub, "Rate limits for the rest api," 2022, accessed: February 7, 2025. [Online]. Available: https://docs.github.com/en/rest/using-the-rest-api/rate-limits-for-the-rest-api?apiVersion=2022-11-28

Appendix - C : References

[40] G. Nguyen-Truong, H. J. Kang, D. Lo, A. Sharma, A. E. Santosa, A. Sharma, and M. Y. Ang, "Hermes: Using commit-issue linking to detect vulnerability-fixing commits," in *2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2022, pp. 51–62.

[41] J. Lin, Y. Liu, Q. Zeng, M. Jiang, and J. Cleland-Huang, "Traceability transformed: Generating more accurate links with pre-trained bert models," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 2021, pp. 324–335.

[42] S. Robertson, H. Zaragoza *et al.*, "The probabilistic relevance framework: Bm25 and beyond," *Foundations and Trends® in Information Retrieval*, vol. 3, no. 4, pp. 333–389, 2009.

[43] Z. Feng, D. Guo, D. Tang, N. Duan, X. Feng, M. Gong, L. Shou, B. Qin, T. Liu, D. Jiang *et al.*, "Codebert: A pre-trained model for programming and natural languages," *arXiv preprint arXiv:2002.08155*, 2020.

[44] C. J. Burges, "From ranknet to lambdarank to lambdamart: An overview," *Learning*, vol. 11, pp. 23–581, 2010.

[45] M. Stone, "Cross-validatory choice and assessment of statistical predictions," *Journal of the royal statistical society: Series B (Methodological)*, vol. 36, no. 2, pp. 111–133, 1974.