Institute for Software and Systems Engineering

TU Clausthal
Clausthal University of Technology

Evaluating Large Language Models for Security Bug Report Prediction

Farnaz Soltaniani, Shoaib Razagh, Mohammad Ghafari

PhD @ Technische Universität Clausthal, Germany

The 33rd IEEE International Conference on Software Analysis, Evolution and Reengineering

March 2026



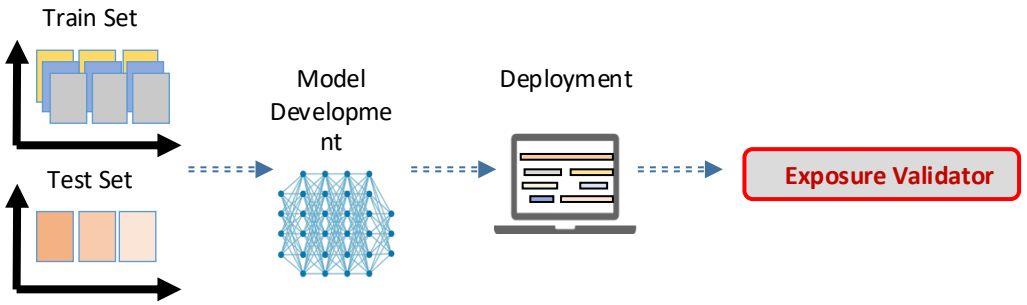
Limassol

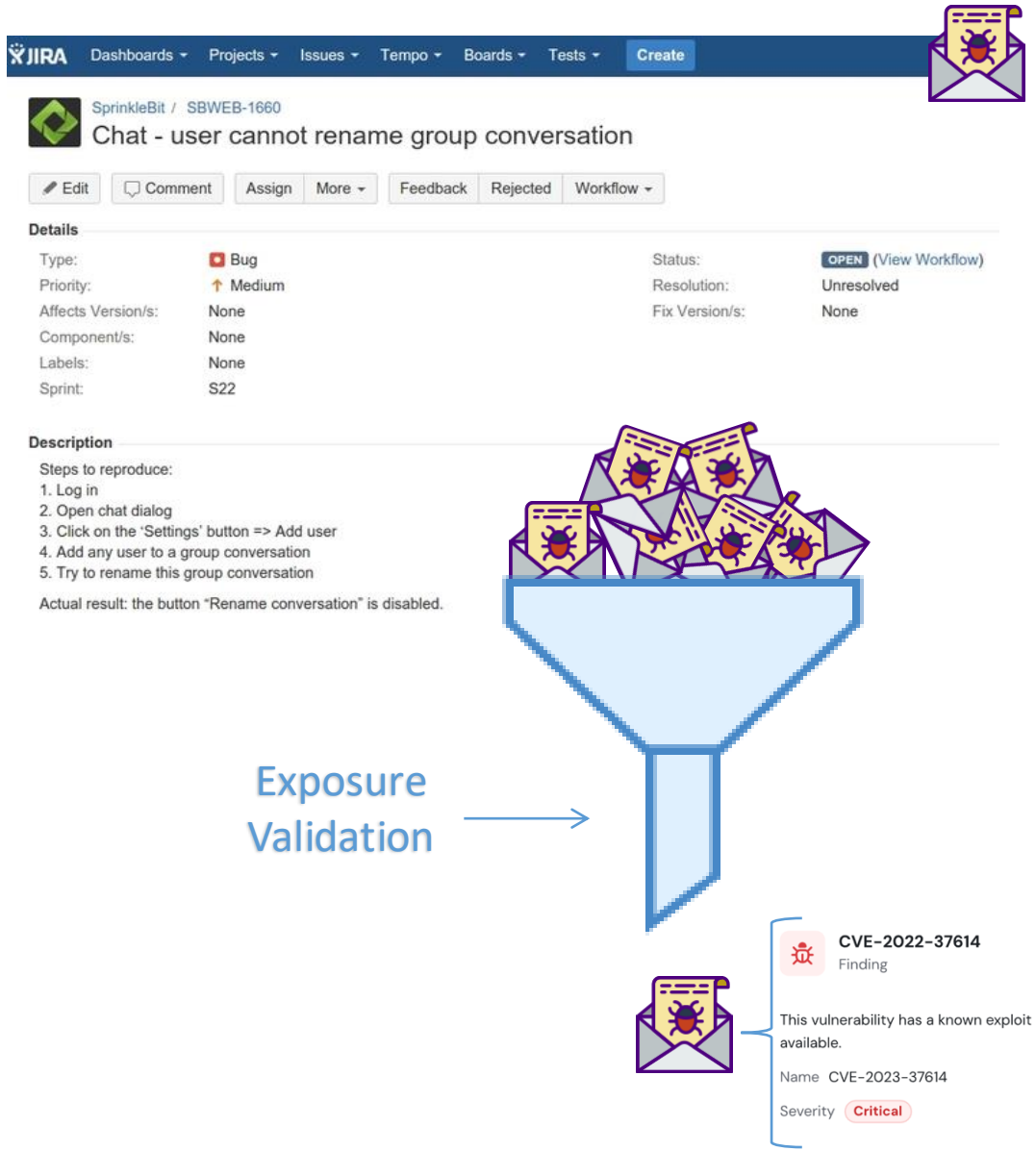
Security-relevant Vulnerabilities

Bug reports (BRs) are primary means to inform development teams of defects in software development lifecycle.

Bug reports with security implications, are known as **Security Bug Reports (SBRs)**.

- The scarcity of SBRs leads to a significant class imbalance.
- Developers do not consider them a priority.
- Lack of security expertise among developers make these vulnerabilities persistent in the open source repositories.
- Static analysis tools produce high numbers of FPs leading to developer fatigue and ignorance.





Security-relevant Vulnerabilities

Text Filtering and Ranking for Security Bug Report Prediction

Fayola Peters, Thein Than Tun, *Member, IEEE*, Yijun Yu, *Member, IEEE*, and Bashar Nuseibeh, *Member, IEEE*

Abstract—Security bug reports can describe security critical vulnerabilities in software products. Bug tracking systems may contain thousands of bug reports, where relatively few of them are security related. Therefore finding unlabelled security bugs among them can be challenging. To help security engineers identify these reports quickly and accurately, text-based prediction models have been proposed. These can often mislabel security bug reports due to a number of reasons such as class imbalance, where the ratio of non-security to security bug reports is very high. More critically, we have observed that the presence of security related keywords in both security and non-security bug reports can lead to the mislabelling of security bug reports. This paper proposes FARSEC, a framework for filtering and ranking bug reports for reducing the presence of security related keywords. Before building prediction models, our framework identifies and removes non-security bug reports with security related keywords. We demonstrate that FARSEC improves the performance of text-based prediction models for security bug reports in 90 percent of cases. Specifically, we evaluate it with 45,940 bug reports from Chromium and four Apache projects. With our framework, we mitigate the class imbalance issue and reduce the number of mislabelled security bug reports by 38 percent.

Security-relevant Vulnerabilities

Data Quality Matters: A Case Study on Data Label Correctness for Security Bug Report Prediction

Xiaoxue Wu, Wei Zheng, Xin Xia, and David Lo

Abstract—In the research of mining software repositories, we need to label a large amount of data to construct a predictive model. The correctness of the labels will affect the performance of a model substantially. However, limited studies have been performed to investigate the impact of mislabeled instances on a predictive model. To bridge the gap, in this article, we perform a case study on the security bug report (SBR) prediction. We found five publicly available datasets for SBR prediction contains many mislabeled instances, which lead to the poor performance of SBR prediction models of recent studies (e.g., the work of Peters *et al.* and Shu *et al.*). Furthermore, it might mislead the research direction of SBR prediction. In this article, we first improve the label correctness of these five datasets by manually analyzing each bug report, and we find 749 SBRs, which are originally mislabeled as Non-SBRs (NSBRs). We then evaluate the impacts of datasets label correctness by comparing the performance of the classification models on both the noisy (i.e., before our correction) and the clean (i.e., after our correction) datasets. The results show that the cleaned datasets result in improvement in the performance of classification models. The performance of the approaches proposed by Peters *et al.* and Shu *et al.* on the clean datasets is much better than on the noisy datasets. Furthermore, with the clean datasets, the simple text classification models could significantly outperform the security keywords-matrix-based approaches applied by Peters *et al.* and Shu *et al.*

Index Terms—Security bug report prediction, data quality, label correctness

Security Bug Report Prediction Within and Across Projects: A Comparative Study of BERT and Random Forest

Farnaz Soltaniani
farnaz.soltaniani@tu-clausthal.de
Technische Universität Clausthal
Germany

Mohammad Ghafari
mohammad.ghafari@tu-clausthal.de
Technische Universität Clausthal
Germany

Mohammed Sayagh
mohammed.sayagh@etsmtl.ca
ETS - Quebec University
Canada

Abstract

Early detection of security bug reports (SBRs) is crucial for preventing vulnerabilities and ensuring system reliability. While machine learning models have been developed for SBR prediction, their predictive performance still has room for improvement. In this study, we conduct a comprehensive comparison between BERT and Random Forest (RF), a competitive baseline for predicting SBRs. The results show that RF outperforms BERT with a 34% higher average G-measure for within-project predictions. Adding only SBRs from various projects improves both models' average performance. However, including both security and nonsecurity bug reports significantly reduces RF's average performance to 46%, while boosts BERT to its best average performance of 66%, surpassing RF. In cross-project SBR prediction, BERT achieves a remarkable 62% G-measure, which is substantially higher than RF.

slowly, with many remaining unresolved for extended periods [4]. Prioritizing SBRs over Non-Security bug reports (NSBRs) is critical to addressing vulnerabilities promptly and preventing potential exploitation. Yet, limited security knowledge among mainstream developers often leads to SBRs being misclassified as NSBRs during reporting [9], delaying vulnerability detection and mitigation.

Researchers have built machine learning techniques to classify SBRs from NSBRs, primarily based on the description field of bug reports [6, 14, 15, 18, 27].

However, they often perform poorly due to class imbalance and limited semantic understanding, especially in large, diverse datasets. Despite efforts to enhance SBR prediction, distinguishing minority classes remains a challenge [13].

BERT [7], a pre-trained transformer-based model, offers exceptional semantic understanding and has achieved state-of-the-art

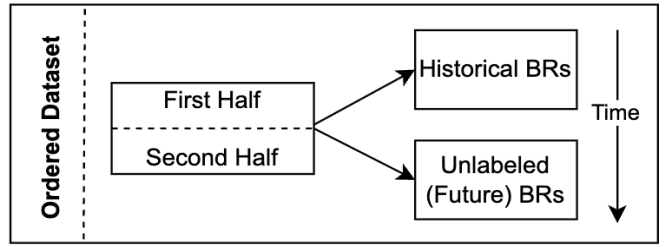
Security-relevant Vulnerabilities

RQ_1: How effective are proprietary LLMs at predicting SBRs?

RQ_2: How effective are fine-tuned LLMs at predicting SBRs?

Security-relevant Vulnerabilities

Project	Description	BRs	NSBRs	SBRs
Chromium	Web browser called Chrome	41,940	41,132	808
Derby	A relational database management system	1,000	821	179
Camel	A rule-based routing and mediation engine	1,000	926	74
Ambari	Hadoop management web UI backed by its RESTful APIs	1,000	944	56
Wicket	Component-based web application framework for Java programming	1,000	953	47



RQ1: How effective are proprietary LLMs at predicting SBRs?

We employed Gemini (gemini-2.5-flash) and GPT (GPT-4.1), which have both consistently demonstrated strong performance on vulnerability detection tasks.

RQ2: How effective are fine-tuned LLMs at predicting SBRs?

We select four models for supervised fine-tuning, including BERT, DistilBERT, DistilGPT-2, and Qwen2.5-Coder.

Prompt Development

Domain-Specific Role Playing:

Initially, the models are prompted to assume the role of a **senior security auditor**, as role-playing **enhances the accuracy and relevance of the responses** by focusing on a specific domain.

Rule-based Verification:

We define a strict **step-by-step task** and ask the models to independently determine, within a **given bug report**, **whether the report is a SBR or NSBR**.

Prompt

– **Role:** You are a Security Vulnerability Classification Expert (CWE/CVE/STRIDE/MITRE ATT&CK).

– **Definitions:**
Security: The bug describes a vulnerability (e.g., null pointer, improper memory handling, privacy leakage, improper permission, etc.).

Non–security: The bug is a functional error, a usability issue, a UI glitch, a performance issue, or a feature request, with no clear exploit path.

– **Task:** determine if the bug report is security–relevant.

– Output only this JSON (no extra text):
{
“explanation”: “concise reason (<= 20 words)”,
“classification”: “security” | “non–security”
}

– BUG REPORT: {Text}

Results

RQ_1: How effective are proprietary LLMs at predicting SBRs?

- **Gemini** detects over **~3× more SBRs** than **GPT**, leading to a higher G-measure across datasets. However, it is accompanied by increased FPR.

- **In each dataset**, Gemini proves to be a more effective tool for SBR prediction; its superior G-measures (reaching 0.86 on Chromium) indicate better performance when **identifying SBR** is **prioritized** over **minimizing false alarms**.

Datasets	Recall		Precision		F1-score		FPR		G-measure	
	GPT	Gem	GPT	Gem	GPT	Gem	GPT	Gem	GPT	Gem
Chromium	0.32	0.91	0.08	0.09	0.13	0.17	0.08	0.19	0.48	0.86
Derby	0.26	0.68	0.56	0.44	0.35	0.53	0.05	0.21	0.41	0.73
Camel	0.13	0.72	0.23	0.22	0.17	0.34	0.04	0.25	0.23	0.73
Ambari	0.25	0.63	0.27	0.16	0.26	0.26	0.02	0.11	0.40	0.74
Wicket	0.17	0.78	0.18	0.19	0.18	0.31	0.04	0.16	0.29	0.81
Average	**0.23**	**0.74**	**0.26**	**0.22**	**0.22**	**0.36**	**0.05**	**0.18**	**0.36**	**0.77**

Results

RQ_2: How effective are fine-tuned LLMs at predicting SBRs?

- Among the fine-tuned LLMs, **DistilBERT** achieves the best average performance across all key metrics.

- Individual datasets exhibit substantial **variability in model effectiveness**; all models achieve their best scores on **Chromium** and **Derby**, yet encounter pronounced difficulties with **Camel**, **Ambari**, and **Wicket.**

Datasets	Recall				Precision				F1-score				FPR				G-measure			
	B	DisB	DisG	Q	B	DisB	DisG	Q	B	DisB	DisG	Q	B	DisB	DisG	Q	B	DisB	DisG	Q
Chromium	0.70	0.74	0.79	0.77	0.90	0.83	0.84	0.87	0.78	0.79	0.82	0.82	0.00	0.00	0.00	0.00	0.83	0.85	0.88	0.87
Derby	0.40	0.39	0.10	0.28	0.68	0.78	0.12	0.45	0.51	0.52	0.11	0.34	0.04	0.03	0.18	0.08	0.57	0.56	0.18	0.43
Camel	0.20	0.20	0.02	0.15	0.60	0.53	0.04	0.35	0.30	0.29	0.03	0.21	0.01	0.02	0.05	0.03	0.33	0.33	0.04	0.26
Ambari	0.25	0.19	0.06	0.13	0.31	0.60	0.04	0.08	0.28	0.29	0.05	0.10	0.02	0.00	0.05	0.05	0.40	0.32	0.12	0.22
Wicket	0.13	0.30	0.02	0.17	0.43	1.00	0.10	0.80	0.20	0.47	0.04	0.29	0.01	0.00	0.02	0.00	0.23	0.47	0.04	0.30
Average	**0.34**	**0.36**	**0.20**	**0.30**	**0.58**	**0.75**	**0.23**	**0.51**	**0.41**	**0.47**	**0.21**	**0.35**	**0.02**	**0.01**	**0.06**	**0.03**	**0.47**	**0.51**	**0.25**	**0.42**

Discussion

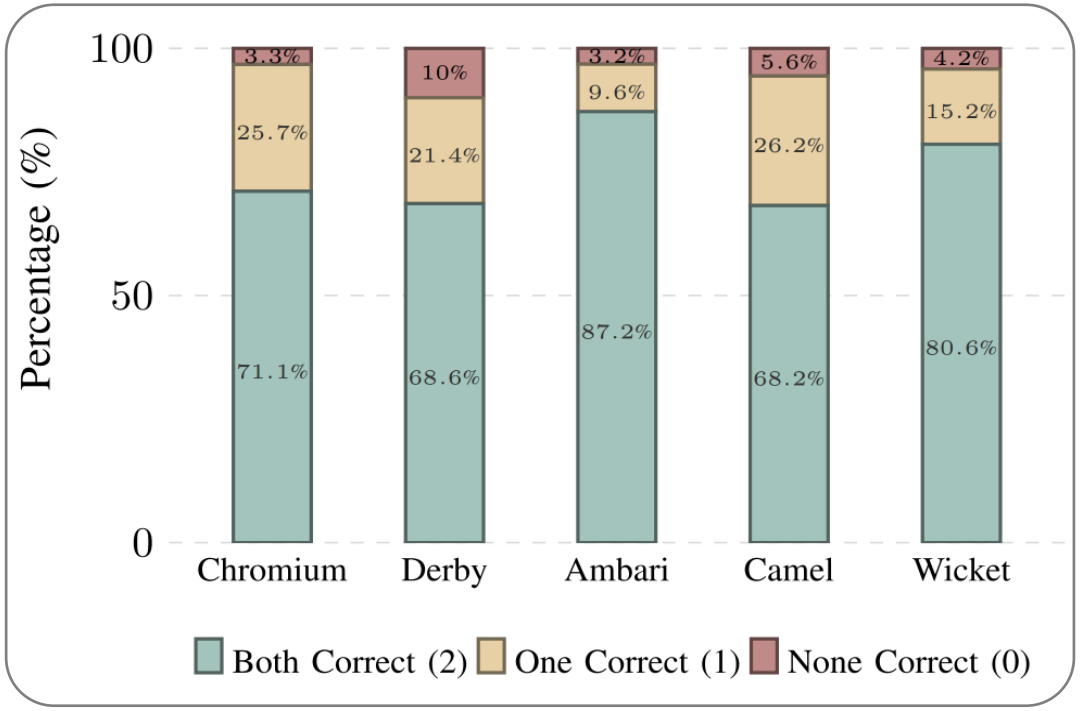
Few-shot and Zero-shot:

Our results showed **no significant performance difference** between few-shot and zero-shot settings.

For example, on the Wicket and Camel datasets, both models showed small, statistically non-significant changes in **F1-scores** as confirmed by paired t-test analysis ($p > 0.05$).

Model Agreement in Proprietary LLMs:

To assess the level of agreement between the two prompted models, we computed a consensus score for each sample across all datasets. This score ranges from 0 (neither model correctly predicts the label) to 2 (both models correctly predict the label).



Discussion

Few-shot and Zero-shot:

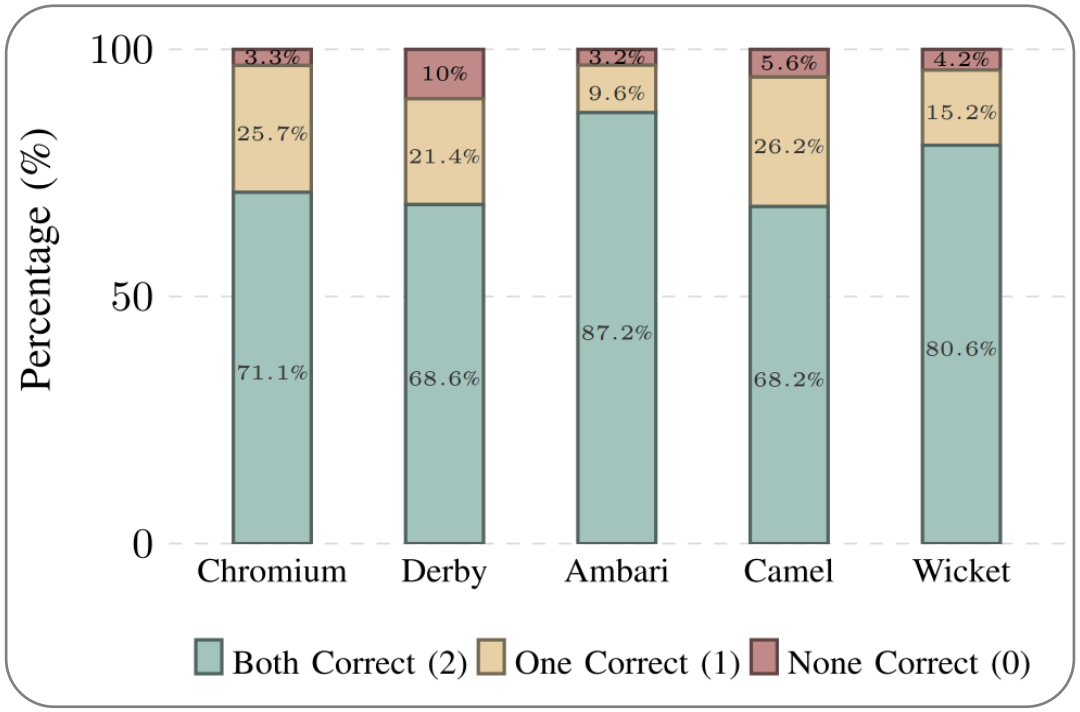
Our results showed **no significant performance difference** between few-shot and zero-shot settings.

For example, on the Wicket and Camel datasets, both models showed small, statistically non-significant changes in **F1-scores** as confirmed by paired t-test analysis ($p > 0.05$).

Score-2 is the dominant category showing that both models are in consensus in identifying the correct labels. (Ambari is the easiest dataset, followed by similarly strong results on Wicket.)

Score-1 cases, where only one model is correct, are notable in Camel (26.2%) and Chromium (25.7%).

Score-0 cases are rare across all datasets indicating that neither model could predict the correct labels.



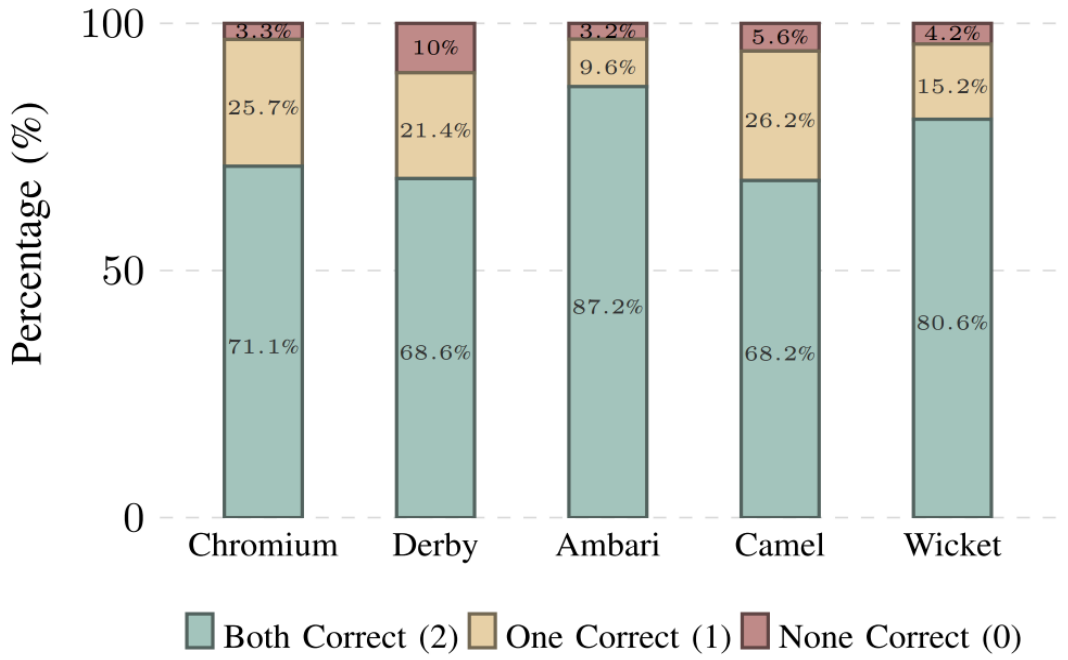
Discussion

Overall, although cases where **both models are incorrect** are **rare**, it is not uncommon for only **one model** to be correct. **Combining model predictions** could further improve results and represents a promising direction for future research.

Score-2 is the dominant category showing that both models are in consensus in identifying the correct labels. (Ambari is the easiest dataset, followed by similarly strong results on Wicket.)

Score-1 Cases, where only one model is correct, are notable in Camel (26.2%) and Chromium (25.7%).

Score-0 cases are rare across all datasets indicating that neither model could predict the correct labels.



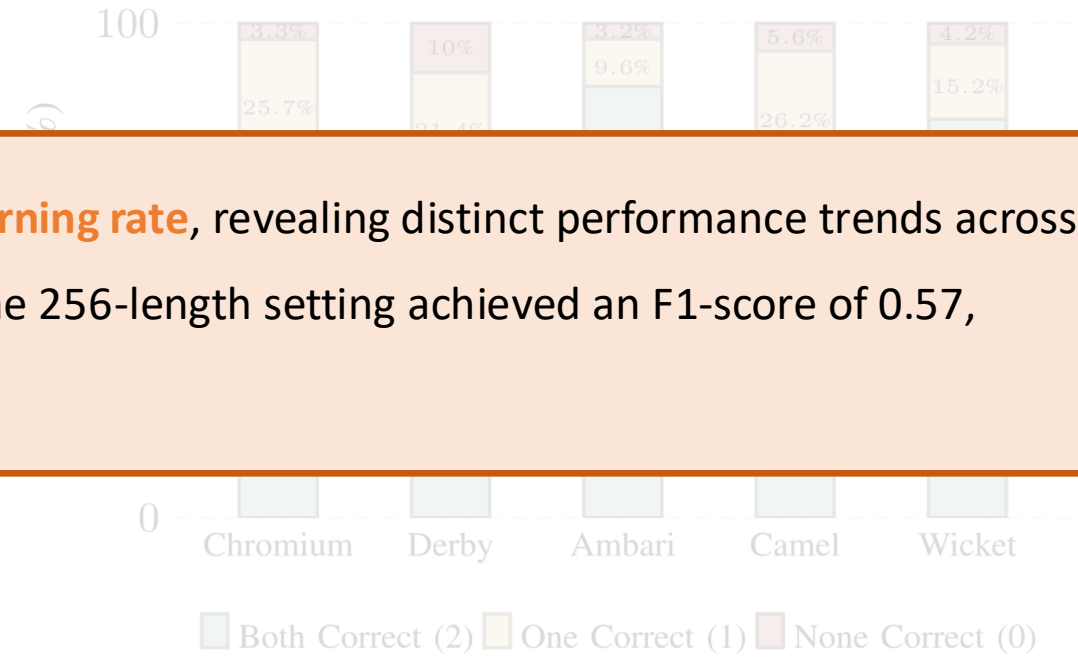
Discussion

Few-shot and Zero-shot:

Overall, although cases where **both models are incorrect** are **rare**, it is not uncommon for only **one model** to be correct. **Combining model predictions** could further improve results and represents a promising direction for future research.

Score-2 is the dominant category showing that both models are in consensus in identifying the correct labels. (Ambari is

Score-0 cases are rare across all datasets indicating that neither model could predict the correct labels.



We evaluated model sensitivity to **maximum sequence length** and **learning rate**, revealing distinct performance trends across the datasets (We have reported the best results). For instance in Camel, the 256-length setting achieved an F1-score of 0.57, surpassing the 128 length (F1-score 0.47) configuration.

Discussion

Prompted vs. Fine-tuned Models:

- **Execution Time.**
  - **Proprietary Models:** GPT consistently achieved faster per-row processing and substantially lower total execution times. **GPT processed each row** -> in **≈ 1.7** seconds, **full dataset** in under **15** minutes, whereas Gemini required ≈ 3–4 seconds per row, with total execution time of 29 minutes. (19.5 vs. 50 hours)
  - **Fine-tuned Models:** require an initial computational investment for training. Chromium, required between 2-7 hours, depending on the maximum sequence length. -> Once trained, entire Chromium dataset in just 10-47 minutes was processed.
- **Cost Analysis.**
  - **Proprietary Models:** The total expenditure for GPT was approximately $25, whereas Gemini incurred a cost of $71
  - **Fine-tuned Models:** Training fine-tuned small LLMs is possible on a local machine, requiring only moderate computational resources.

Discussion

Prompted vs. Fine-tuned Models:

- **Ethical and Operational Implications.**
  - **Proprietary Models:** Transmitting potential security bug report descriptions to third-party API providers (e.g., OpenAI, Google) raises issues of data privacy and sovereignty.
  - **Fine-tuned Models:** Local models, which are trained once and reused without external data transfer, offer a privacy-preserving alternative that keeps all sensitive data within an organization's internal infrastructure.

THANK YOU

Evaluating Large Language Models for Security Bug Report Prediction

Farnaz Soltaniani

PhD @ Technische Universität Clausthal, Germany

The 33rd IEEE International Conference on Software Analysis, Evolution and Reengineering

March 2026



Limassol