SecFlow: An Agentic LLM-Based Framework for Modular Cyberattack Analysis and Explainability

Francesco Blefari[1,2], Cristian Cosentino[1,*], Angelo Furfaro[1], Fabrizio Marozzo[1] and Francesco Aurelio Pironti[1]

[1]University of Calabria, Via Pietro Bucci, 87036 Rende, Italy
[2]IMT School for Advanced Studies, Piazza San Francesco, 55100 Lucca, Italy

Abstract

Large Language Models (LLMs) are rapidly reshaping the software engineering landscape, with applications ranging from code synthesis to intelligent assistance in debugging and testing. In this work, we propose SecFlow, a modular agentic framework designed for intelligent classification and contextual explanation of web-based cyberattacks. Unlike traditional detection pipelines, SecFlow integrates fine-tuned LLM-based classifiers and a Retrieval-Augmented Generation (RAG) component into a cohesive architecture, enabling contextualized reasoning and transparent report generation from raw logs. We focus on a single attack vector, Server-Side Template Injection (SSTI), to illustrate the benefits of a dynamic and explainable security pipeline. The system autonomously identifies malicious payloads, retrieves relevant domain knowledge, and produces human-readable reports detailing the nature of the attack, its potential impact, and recommended mitigation strategies. Extensive evaluations show that our approach outperforms monolithic models in accuracy, robustness, and interpretability. Furthermore, the architecture is fully extensible, allowing integration with existing security infrastructures and future inclusion of additional classifiers or automated defense routines. Our findings highlight that LLMs, even without directly generating code, can serve as reliable reasoning agents in software workflows where transparency, explainability, and modular decision-making are critical.

Keywords

Large Language Models, Agentic Retrieval-Augmented Generation, Cyber Threat Detection, Fine-tuned Security Classifiers, Intrusion Detection Systems

1. Introduction

In recent years, software engineering has been increasingly shaped by the integration of Large Language Models into the development, analysis, and maintenance of complex systems. Originally created for general-purpose language understanding, LLMs are now proving effective as intelligent engines capable of orchestrating and automating software processes, particularly in critical domains such as cybersecurity.

One of the most urgent challenges in this domain is the overload of alerts generated by monitoring systems like Intrusion Detection and Prevention Systems (IDS/IPS), which often produce hundreds of thousands of notifications per hour. This volume makes manual analysis unsustainable and calls for intelligent pipelines that can not only filter and classify events but also interpret and explain them in a trustworthy and actionable manner.

To address this, we introduce SecFlow, a modular framework that leverages LLMs as autonomous agents for the classification and explanation of web-based attacks. SecFlow is designed to operate within software-oriented security workflows, where it coordinates specialized modules such as attack classifiers and retrieval-augmented knowledge systems to generate structured, human-readable reports.

Unlike static and monolithic solutions, SecFlow follows a dynamic and extensible design, enabling integration with existing infrastructures (e.g., SIEM platforms, dashboards) and future extension toward remediation and automated decision-making.

GeCoIn 2025: Generative Code Intelligence Workshop, co-located with the 28th European Conference on Artificial Intelligence (ECAI-2025), October 26, 2025 — Bologna, Italy
*Corresponding author
francesco.blefari@unical.it (F. Blefari); cristian.cosentino@unical.it (C. Cosentino); angelo.furfaro@unical.it (A. Furfaro); fabrizio.marozzo@unical.it (F. Marozzo); francesco.pironti@unical.it (F. A. Pironti)

This work presents a case study centered on *Server-Side Template Injection (SSTI)*, demonstrating how SecFlow can transform raw logs into contextualized threat interpretations. More broadly, we position SecFlow as a reusable and interpretable LLM-powered software component, applicable to any setting where textual traces must be converted into intelligent, actionable insights.

2. Background

The incorporation of LLMs into software pipelines has opened new directions for the development of intelligent, semi-autonomous systems in cybersecurity. Beyond their traditional use as conversational engines, LLMs are now being deployed as decision-making components capable of interpreting data, coordinating specialized tools, and generating structured outputs, particularly when orchestrated through agent-based architectures.

SecFlow builds on this vision by treating the LLM as an autonomous control unit within a modular pipeline. Each functional block, such as the threat classifier, the contextual retrieval engine, and the reporting module, is isolated and reusable, adhering to software engineering principles like separation of concerns and stateful orchestration. This design enables flexible integration with upstream alert sources (e.g., IDS/IPS) and downstream interfaces (e.g., SIEM systems or dashboards).

The system activates when a flagged network event is received, delegating initial classification to a specialized transformer-based model. The result triggers a RAG loop, where the LLM queries a curated knowledge base to collect supporting context. Rather than performing a single retrieval step, the agent iteratively refines the query and retrieved content, allowing it to produce a well-grounded, human-readable explanation of the event.

This RAG loop functions as an internal reasoning engine, treating retrieval as a software subroutine that is re-invoked until consistency and clarity are achieved. The resulting report includes not only a classification outcome, but also contextual evidence, mapped vulnerabilities, and possible mitigation strategies.

To demonstrate this architecture, we focus on a single use case, Server-Side Template Injection, a complex and realistic attack pattern. This allows us to illustrate the system's modularity and extensibility: new types of attacks can be supported by adding additional classifiers, without changing the core coordination logic.

In essence, SecFlow exemplifies how LLMs can be embedded into software architectures not as monolithic predictors but as interpretable agents within intelligent pipelines, enabling structured interaction, evidence-backed reasoning, and scalable adaptation to evolving threat landscapes.

3. Related Work

The integration of LLMs into cybersecurity workflows has recently attracted significant attention from both the academic and industrial communities. Early research emphasized their dual-use potential: while LLMs can assist in automating security tasks, they can also lower the entry barrier for adversaries by enabling automatic generation of phishing content, malware, and exploit scripts [1, 2].

Prior to the emergence of LLMs, automation efforts in log analysis primarily relied on traditional machine learning techniques applied to both system and network logs, as demonstrated in [3, 4]. These approaches aimed to support security analysts by modeling the progression of attacks through visual and interpretable representations, such as Petri Nets or Directly Follows Graphs, thereby facilitating a better understanding of the attack flow and enabling more informed decision-making.

On the defensive side, efforts have focused on using LLMs to enhance classification, explanation, and response mechanisms in threat detection systems. Frameworks such as *CyberSecEval 2* [5] and *SecBench* [6] provide comprehensive benchmarking environments for evaluating LLMs in security tasks, revealing important trade-offs between performance and adversarial robustness. Additional studies have investigated RAG-based systems that enrich the interpretability of threat assessment by grounding LLM outputs in structured knowledge [7].

Recent research has also explored the use of LLMs for enhancing transparency and trust in AI-assisted cybersecurity. For example, Cantini et al. [8] present a methodology to integrate LLMs into post-hoc analysis pipelines, while in [9], a multi-task setting is adopted for jointly detecting false information and categorizing cyber incidents. In [10], topic-aware BERT models are used to unmask misinformation campaigns, showing how NLP tools can be applied to related problems in cyber threat intelligence. Similarly, Cosentino et al. [11] explore how LLMs can be orchestrated to improve the interpretability of social-media-based event detection, a technique that may be transferred to cyber attack explanation.

Despite the growing popularity of these approaches, many existing systems rely on single-pass retrieval or black-box model outputs, limiting their adaptability and transparency. SecFlow differs from such approaches by adopting an agentic architecture, in which a central LLM agent dynamically orchestrates the interaction between a specialized classifier, a retrieval-augmented module, and a report generator. This architecture provides a controlled environment for iterative reasoning, enabling the system to refine its interpretation in the face of ambiguity.

Compared to previous works, SecFlow also emphasizes modularity and extensibility. Its architecture allows new attack classes to be incorporated by simply adding a new specialized classifier, without retraining the agent or altering the reasoning flow. This makes the system particularly suitable for large-scale, dynamic environments, where threat landscapes evolve rapidly and require continuous adaptation.

4. Methodology

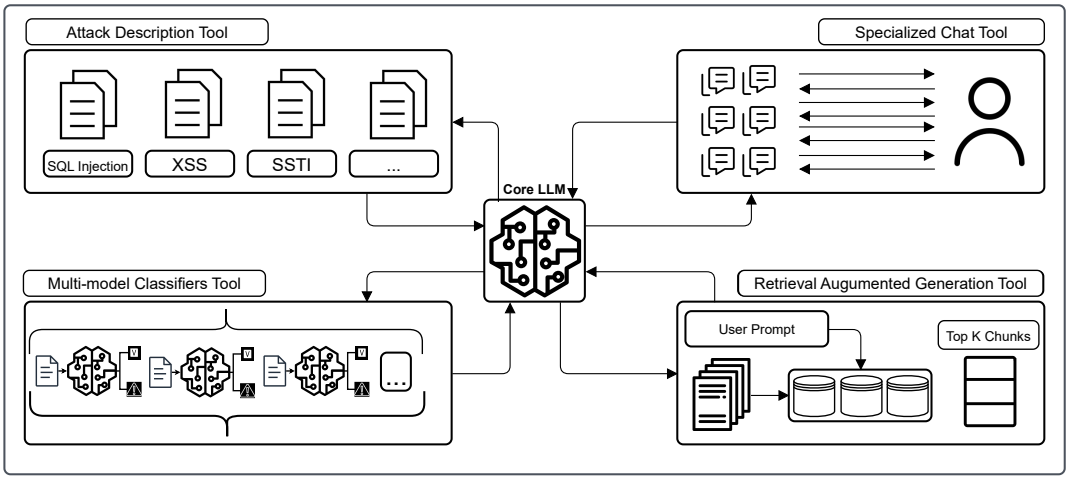

Figure 1: Overview of the SecFlow architecture: components are modular and coordinated by a central LLM-based agent.

SecFlow is designed following modular software engineering principles, combining reusable components into an orchestrated pipeline managed by a central Core LLM Agent. The architecture, shown in Figure 1, is composed of four main modules: the Multi-model Classifiers Tool, the Retrieval Augmented Generation Tool, the Attack Description Tool, and the Specialized Chat Tool , all coordinated by the Core LLM Agent, which governs logic flow, reasoning, and adaptivity across modules.

At runtime, the system is triggered by an alert generated by a traditional IDS/IPS or a log analysis service. The alert is forwarded to the Multi-model Classifiers Tool, which manages transformer-based models, each theoretically specialized in detecting a specific type of web attack. Although currently only the classifier for Server-Side Template Injection (SSTI) has been actually implemented, the architecture is designed to be easily extensible. Attack types such as SQL Injection and Cross-Site Scripting (XSS) are shown as illustrative examples to highlight the generalizability of the methodology, which relies on a homogeneous structure of the extracted information that facilitates the integration of additional

classifiers.

The output of each classifier, i.e., the predicted class together with its confidence score, is combined with the original input to produce an enriched textual representation. This refined output serves as the basis for the Retrieval Augmented Generation Tool, which transforms it into a natural language query. The query is then executed against a curated cybersecurity knowledge base composed of indexed documents from sources such as OWASP guidelines, CVE repositories, and MITRE ATT&CK patterns. Retrieval is performed using dense vector search enhanced with Maximal Marginal Relevance (MMR) to optimize both relevance and diversity. Unlike static RAG workflows, our pipeline allows the Core LLM Agent to iteratively refine the query when uncertainty is detected, continuing until sufficient context is gathered.

Once the most relevant contextual documents have been retrieved, the Core LLM Agent synthesizes the evidence into a structured and explainable output via the Attack Description Tool. This module presents the final report, which includes the threat category, semantic interpretation of the attack, matched vulnerabilities (e.g., CVEs), and recommended remediation actions. For example, in the case of SSTI, a malicious payload such as `{{__class__.__mro__[1].__subclasses__()}}` would trigger the classifier to label it as a "template injection vector referencing internal object hierarchy." The retrieval module would gather OWASP references and known vulnerabilities, and the agent would generate a report explaining the nature of the vulnerability and suggesting mitigation strategies such as input sanitization or disabling unsafe template features.

Finally, the system provides an interactive interface via the Specialized Chat Tool, allowing users to query the generated report using natural language. This supports interpretability, iterative refinement, and just-in-time clarification through LLM-powered dialogue.

The modular nature of SecFlow enables seamless extensibility: new attack types can be integrated by plugging additional specialized classifiers into the Multi-model Classifiers Tool, while the logic of reasoning, retrieval, and reporting remains unchanged. This design supports scalability across evolving cybersecurity scenarios, enhances explainability by grounding outputs in semantically clear steps, and promotes adaptability by placing LLMs at the core of orchestration rather than using them as isolated predictors.

5. Experimental Results

The experimental validation of SecFlow focuses exclusively on the SSTI scenario, chosen for its real-world relevance and structural complexity. The goal is to assess the ability of our modular pipeline to classify SSTI payloads, generate coherent and grounded explanations, and handle adversarial or ambiguous inputs via a RAG loop.

5.1. Attack Classification Performance

The classifier for the SSTI attack was implemented using four transformer-based models: bert-base-uncased, albert-base-v2, distilbert-base-uncased, and roberta-base. Each model was fine-tuned on a custom-built dataset, given the absence of publicly available benchmarks specifically tailored for SSTI detection, despite the availability of more general web attack datasets.

The SSTI dataset was constructed by aggregating validated real-world payloads from the PayloadsAllTheThings repository and outputs generated using the SSTImap tool. The positive class consists of a diverse set of payloads known to trigger SSTI vulnerabilities across different template engines (e.g., Jinja2, Twig), while the negative class includes benign and non-malicious strings such as emails, names, numbers, and generic text fragments that do not result in anomalous behavior. The resulting dataset captures the syntactic and semantic variety found in real-world inputs and is made available to the research community upon request through password-protected access [12], in order to prevent misuse while still ensuring reproducibility.

As shown in Figure 2, among the tested models, albert-base-v2 achieved the best overall performance, particularly excelling in precision and AUC-based metrics, highlighting its superior sensitivity to

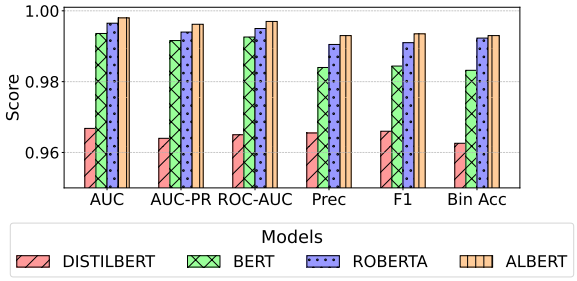

Figure 2: Classification performance of BERT-based models on SSTI detection.

the fine-grained syntax that characterizes SSTI payloads. This confirms the model's effectiveness in identifying malicious patterns, even when obfuscated or slightly altered.

5.2. RAG-Based Explanation and Refinement

Once an SSTI pattern is detected, the classifier activates the RAG module to generate a context-aware explanation. The retrieval engine queries a domain-specific knowledge base composed of documents from OWASP, CVE reports, and curated SSTI examples. The retrieved context is passed to a selected language model, which synthesizes a human-readable justification combining domain knowledge with features of the input payload.

To evaluate the quality of the generated explanations, we conducted a comparative benchmark across five open-weight language models, all with approximately 7–8 billion parameters. The selected models are: DeepSeek-R1 7B[13], Gemma 3B:4B[14], LLaMA3.1 8B[15], Mistral 7B[16], and Qwen2.5 7B[17]. All models were executed within the `Ollama` environment[18], ensuring a unified and reproducible evaluation.

For each model, we generated explanations starting from SSTI payloads and their corresponding labels. Manually curated reference reports were used as ground truth: they describe the attack context, the underlying vulnerability mechanisms, and relevant syntactic cues embedded in the payloads.

We assessed two primary dimensions of explanation quality: (i) the semantic fidelity of the explanation with respect to the real attack characteristics, and (ii) the readability and completeness of the generated text.

To this end, we employed standard Natural Language Generation (NLG) metrics: BLEU[19], ROUGE[20], and METEOR[21] for surface-level lexical overlap. For semantic similarity, we used BERTScore[22]. Finally, we introduced a custom Factual Consistency score to measure alignment between the generated explanation and the retrieved context.

Table 1
Evaluation scores of RAG-generated explanations for SSTI (per model).

Model	BLEU	ROUGE	METEOR	BERTScore	Factual Consistency
LLaMA3.1:8B	0.88	0.90	0.86	0.94	0.96
Qwen2.5:7B	0.87	0.89	0.85	0.94	0.96
Gemma:4B	0.84	0.87	0.82	0.92	0.94
Mistral:7B	0.85	0.88	0.84	0.93	0.95
DeepSeek-R1:7B	0.86	0.89	0.84	0.93	0.95

All models performed well across the board, but LLaMA3.1:8B and Qwen2.5:7B consistently outperformed the others in both semantic similarity and factual grounding. These results highlight the importance of evaluating not only classification accuracy but also the explanatory capabilities of language models when integrated into a RAG-based pipeline. Based on this evaluation, LLaMA3.1:8B was selected as the default RAG controller within our SecFlow architecture.

5.3. Robustness Evaluation

After evaluating the classification performance and explanatory capabilities of our modular RAG-based system, we focus on a critical aspect for real-world deployment: the system's *robustness under input perturbation and its ability to handle unseen attack categories.*

To assess this, we designed a targeted evaluation covering two known failure points in machine learning models:

- **Adversarial Inputs**: legitimate SSTI payloads slightly modified using techniques such as character obfuscation, spacing noise, or encoding distortion, intended to simulate evasion while preserving the malicious semantics;
- **Out-of-Distribution (OOD) Inputs**: examples from attack categories not included in training (e.g., Path Traversal, Command Injection), used to test the model's behavior when exposed to unfamiliar threats.

For each of the five evaluated language models (LLaMA3.1:8B, Qwen2.5:7B, Mistral:7B, DeepSeek-R1:7B, and Gemma:4B), we constructed two benchmark sets:

- **100 adversarial SSTI queries**, generated through controlled perturbations of known payloads;
- **100 OOD examples**, collected from disjoint web attack datasets and explicitly excluded from training.

We use the following metric to quantify the system's resistance to evasion and overgeneralization:

$$\text{Correct Classification (\%)} = \frac{\text{\# Correct Predictions}}{\text{\# Total Queries}} \times 100\% \tag{1}$$

An adversarial input is counted as correct if the model still assigns the correct SSTI label despite the obfuscation. An OOD input is considered correctly handled if the model abstains from misclassifying it as one of the known classes.

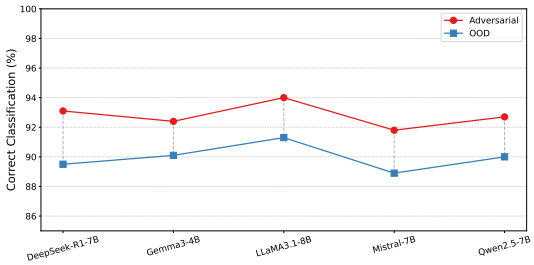

Figure 3: Robustness evaluation of each model on 100 adversarial SSTI queries and 100 out-of-distribution (OOD) inputs. The metric reports the percentage of correct classifications for each condition.

As shown in Figure 3, the `LLaMA3.1:8B` model achieved the best overall performance, correctly classifying 94% of adversarial samples and 91% of OOD inputs. `Mistral:7B` followed closely with 93% accuracy on adversarial inputs, though slightly lower performance (88.5%) on unseen categories. `DeepSeek-R1:7B` and `Gemma:4B` performed reasonably well under adversarial stress but showed a larger drop on OOD queries, indicating limitations in generalization.

These findings confirm that RAG-based reasoning not only improves interpretability but also contributes to robustness, particularly when orchestrated by semantically aware models capable of contextual understanding.

6. Conclusion and Future Directions

The results presented in Section 5 demonstrate that a modular, agent-based architecture can be effective for the detection and interpretation of web-based attacks. SecFlow integrates specialized LLM classifiers with a RAG component, enabling the system to both identify malicious payloads and generate contextual, explainable reports grounded in domain knowledge.

The clear separation between classification and explanation improves flexibility and robustness. In the SSTI use case, the specialized classifier achieved high accuracy even under obfuscation, while the RAG loop produced structured explanations that were quantitatively evaluated through standard NLG metrics.

Looking ahead, we plan to extend SecFlow to support additional attack categories such as SQL Injection, XSS, Path Traversal, and Command Injection. Thanks to its modular design, new classifiers can be integrated without altering the reasoning logic. Moreover, the incorporation of structured knowledge sources, such as security-specific knowledge graphs, will strengthen the system's ability to capture relationships and higher-level abstractions.

Another direction is the integration of SecFlow with automation frameworks like SOAR and SIEM. This will enable the system not only to enhance interpretability, but also to provide actionable recommendations or trigger automated responses under explainability constraints.

In summary, SecFlow represents a modular and explainable framework for detecting and contextualizing web-based cyberattacks. By focusing on SSTI as a representative case study, we showed that the architecture combines interpretability, robustness, and extensibility. Future work will further explore the integration of new attack types, structured knowledge, and automated workflows, positioning SecFlow as a practical foundation for secure, transparent, and auditable AI in cybersecurity.

Declaration on Generative AI

The authors did not use any generative AI tools to create the textual content.

Acknowledgments

This work was supported by the research project "INSIDER: INtelligent ServIce Deployment for advanced cloud-Edge integRation" granted by the Italian Ministry of University and Research (MUR) within the PRIN 2022 program and European Union - Next Generation EU (grant n. 2022WWSCRR, CUP H53D23003670006).

This work was partially supported by project SERICS (PE00000014) and project FAIR (PE0000013) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU.

The work of Francesco A. Pironti was supported by *Agenzia per la Cybersicurezza Nazionale* under the 2024-2025 funding program for promotion of XL cycle PhD research in cybersecurity (CUP H23C24000640005).

References

[1] J. Xu, J. W. Stokes, G. McDonald, X. Bai, D. Marshall, S. Wang, A. Swaminathan, Z. Li, Autoattacker: A large language model guided system to implement automatic cyber-attacks, 2024. arXiv:2403.01038.

[2] J. Gennari, S.-h. Lau, S. Perl, J. Parish, G. Sastry, Considerations for evaluating large language models for cybersecurity tasks, SEI Insights 20 (2024).

[3] F. Blefari, F. A. Pironti, A. Furfaro, Toward a log-based anomaly detection system for cyber range platforms, in: The 19th International Conference on Availability, Reliability and Security (ARES 2024), ACM, 2024. doi:10.1145/3664476.3669976.

[4] F. Romeo, F. Blefari, F. A. Pironti, A. Furfaro, et al., Unveiling attack patterns from ctf network logs with process mining techniques, in: Proceedings of the Joint National Conference on Cybersecurity (ITASEC & SERICS 2025), 2025.
[5] M. Bhatt, S. Chennabasappa, Y. Li, C. Nikolaidis, D. Song, S. Wan, F. Ahmad, C. Aschermann, Y. Chen, D. Kapil, D. Molnar, S. Whitman, J. Saxe, Cyberseceval 2: A wide-ranging cybersecurity evaluation suite for large language models, 2024. arXiv:2404.13161.
[6] P. Jing, M. Tang, X. Shi, X. Zheng, S. Nie, S. Wu, Y. Yang, X. Luo, Secbench: A comprehensive multi-dimensional benchmarking dataset for llms in cybersecurity, 2025. arXiv:2412.20787.
[7] S. Rajapaksha, R. Rani, E. Karafili, A rag-based question-answering solution for cyber-attack investigation and attribution, in: Computer Security. ESORICS 2024 International Workshops, Springer Nature Switzerland, Cham, 2025, pp. 238–256.
[8] R. Cantini, C. Cosentino, F. Marozzo, D. Talia, P. Trunfio, Harnessing prompt-based large language models for disaster monitoring and automated reporting from social media feedback, Online Social Networks and Media 45 (2025) 100295.
[9] R. Cantini, C. Cosentino, F. Marozzo, Multi-dimensional classification on social media data for detailed reporting with large language models, in: IFIP International Conference on Artificial Intelligence Applications and Innovations, Springer, 2024, pp. 100–114.
[10] R. Cantini, C. Cosentino, I. Kilanioti, F. Marozzo, D. Talia, Unmasking deception: a topic-oriented multimodal approach to uncover false information on social media, Machine Learning 114 (2025) 13.
[11] C. Cosentino, M. Gündüz-Cüre, F. Marozzo, Ş. Öztürk-Birim, Exploiting large language models for enhanced review classification explanations through interpretable and multidimensional analysis, in: International Conference on Discovery Science, Springer, 2024, pp. 3–18.
[12] F. Blefari, C. Cosentino, A. Furfaro, F. Marozzo, F. A. Pironti, SSTI dataset, https://github.com/francescopirox/ssti_dataset, 2025.
[13] DeepSeek-AI, Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL: https://arxiv.org/abs/2501.12948. arXiv:2501.12948.
[14] G. DeepMind, Gemma: Open models based on gemini research and technology, 2024. URL: https://ai.google.dev/gemma, accessed: 2024-03.
[15] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, G. Lample, Llama 3: Open foundation and instruction models, 2024. arXiv:2404.14219.
[16] Y. Jiang, T. Nguyen, P. Garcia, et al., Introducing mistral 7b, 2023. URL: https://mistral.ai/news/announcing-mistral-7b.
[17] A. Cloud, Qwen2: A family of open-source language models by alibaba cloud, 2024. URL: https://huggingface.co/Qwen/Qwen2-7B.
[18] Ollama Project, Ollama: Run llms locally, 2024. URL: https://ollama.com.
[19] K. Papineni, S. Roukos, T. Ward, W.-J. Zhu, Bleu: a method for automatic evaluation of machine translation, in: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, ACL '02, Association for Computational Linguistics, USA, 2002, p. 311–318. doi:10.3115/1073083.1073135.
[20] C.-Y. Lin, ROUGE: A package for automatic evaluation of summaries, in: Text Summarization Branches Out, Association for Computational Linguistics, Barcelona, Spain, 2004, pp. 74–81. URL: https://aclanthology.org/W04-1013/.
[21] A. Lavie, A. Agarwal, Meteor: an automatic metric for mt evaluation with high levels of correlation with human judgments, StatMT '07, Association for Computational Linguistics, USA, 2007, p. 228–231.
[22] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, Y. Artzi, Bertscore: Evaluating text generation with bert, in: International Conference on Learning Representations (ICLR), 2020.