Att2RAG: A Double-Condition Framework for Knowledge Poisoning Attacks on RAG Systems

Zhize Hao, Ming Ye, and Tao Liu

Tianjin University of Technology, Tianjin, China
ltlmc@email.tjut.edu.cn

Abstract. Modern retrieval-augmented generation (RAG) and memory-augmented LLM applications are increasingly deployed in knowledge-intensive settings, where model outputs are grounded in external knowledge stores and, in some cases, persistent vector memory. If these stores are compromised, poisoned content can be retrieved repeatedly and shape downstream responses, producing confident but harmful outputs supported by plausible evidence. Existing studies show that RAG pipelines and LLM agents are vulnerable to knowledge poisoning and prompt injection, yet many attacks treat success as a single end-to-end outcome rather than separating retrieval and generation failure modes. In addition, although long-horizon memory writes may create cumulative risks, such effects are not empirically evaluated in our benchmark setting. We present Att2RAG, a double-condition framework for knowledge poisoning attacks on RAG systems. Att2RAG decomposes attack success into a retrieval condition and a generation condition, and formulates poisoning as maximizing attack success subject to both constraints. We instantiate Att2RAG with a white-box variant using projected gradient descent in embedding space and a black-box Q $\oplus$ I attack that combines question self-similarity with adversarial instruction injection using only query access. We evaluate Att2RAG on representative QA benchmarks under common RAG configurations and practical defenses, including paraphrasing and duplicate filtering. Across settings, Att2RAG achieves mid-90% attack success rates without defenses across white-box and black-box variants, while these defenses reduce success only modestly. The results reveal limitations of semantic-similarity-driven retrieval and motivate defenses beyond surface-form rewriting and naive duplicate removal in practical RAG deployments.

Keywords: RAG · Knowledge poisoning · Adversarial attack · Large language models · RAG security.

1 Introduction

Retrieval-augmented generation (RAG) and LLM applications with vector memory are increasingly used in knowledge-intensive scenarios. In a typical pipeline, an LLM answers a user query by retrieving relevant documents from an external knowledge base and conditioning generation on the retrieved context; in agentic settings, a

long-term memory store may additionally accumulate interaction histories and task-specific state [1-3]. This design allows practitioners to keep LLM parameters fixed while updating the knowledge store and memory content, which simplifies deployment and maintenance.

This flexibility introduces a security risk. In many deployments, the knowledge base and memory store are open to writes from semi-trusted or untrusted sources, including user contributions, third-party connectors, or automated logging of agent outputs. After malicious content is injected, the RAG pipeline may repeatedly retrieve and amplify poisoned passages, causing the LLM to produce harmful or incorrect responses that appear grounded in supporting evidence [4, 5].

A growing body of work studies the security of RAG and LLM-based agents. Prompt injection and jailbreak attacks show that adversarial prompts can override intended behavior even in black-box settings [6, 7]. Other studies show that RAG pipelines can be compromised by inserting adversarial documents into the retrieval corpus, and that backdoor-style triggers can persist in long-term agent memory [4, 5]. While these findings establish that RAG and agent systems are vulnerable, existing attacks often adopt heterogeneous formulations and evaluation protocols, complicating comparisons and obscuring where defenses should intervene [8-10].

Two aspects remain insufficiently developed. First, there is no widely adopted retrieval-generation aligned framework for knowledge poisoning that supports principled attribution of failures. Many attacks optimize an implicit end-to-end objective [4, 11] without explicitly separating retrieval, namely whether malicious documents are retrieved, from generation, namely whether the model output is controlled given those documents. Second, long-term memory evolution can amplify a single poisoning event in real deployments, but most benchmark-based evaluations, including ours, focus on static corpora and single-turn QA; accordingly, we treat memory evolution as an implication rather than an evaluated contribution [12, 13].

To address these gaps, we develop Att2RAG, a double-condition framework for analyzing and instantiating knowledge poisoning attacks on RAG systems. Att2RAG aligns with the two-stage RAG pipeline by defining attack success as jointly satisfying a retrieval condition and a generation condition, and by formulating knowledge poisoning as an optimization problem under these constraints. We design a gradient-based white-box variant and a practical black-box Q $\oplus$ I attack that requires only query access.

2 Research Approach

A typical RAG process contains four steps: the user raises a natural-language question; the retriever retrieves relevant documents or information snippets from an external knowledge base; the retrieved information is input into the LLM together with the user question as context; and the LLM generates the final answer. The essence of knowledge poisoning is to induce the language model to produce predetermined malicious content in specific query domains by injecting malicious text.

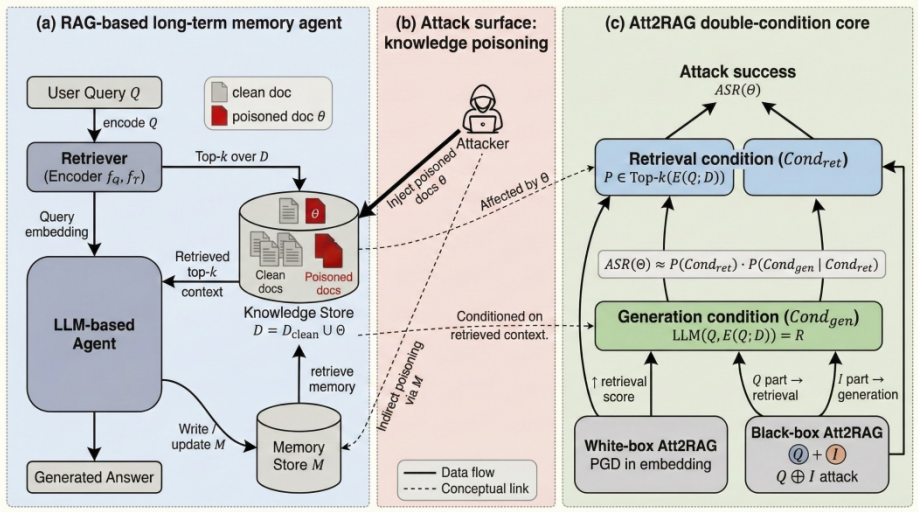

Fig. 1. Overview of Att2RAG. (a) A RAG system answers queries Q using a knowledge store D = Dclean ∪ Θ. (b) Poisoned documents θ ∈ Θ are injected into D and may also be introduced through memory writes in agentic settings. (c) Att2RAG aligns with the RAG pipeline order: retrieval condition Condret followed by generation condition Condgen, instantiated by white-box PGD-based and black-box Q ⊕ I attacks.

This study formulates the attack process as an optimization problem: seek a set of optimal malicious texts such that, after these texts are injected into the knowledge base, the attack success rate for the target question is maximized. The core theoretical innovation is therefore to formulate knowledge poisoning as an optimization problem and design Att2RAG, an attack framework that generates adversarial texts that are both retrievable and effective in inducing the model to produce malicious content by optimizing semantic perturbation strategies.

After determining the attack framework, we conduct experiments on NQ and HotpotQA to quantitatively evaluate effectiveness and stealthiness. The experimental results present the attack success rate under different poisoning and retrieval settings, and analyze the syntactic and semantic behavior of malicious texts. The research process follows the paradigm of adversarial attack and security assessment: theoretical modeling, attack construction, adversarial injection, system evaluation, and defense analysis.

3 Model Analysis

This study identifies a security threat in the knowledge-base subsystem of the RAG architecture and proposes Att2RAG. By injecting adversarial samples into the knowledge base, Att2RAG can induce large-scale pretrained LLMs to generate predetermined malicious content in specific query domains. Unlike attacks that

modify model parameters, this approach exploits the RAG system's reliance on external knowledge bases and therefore achieves a stealthier attack surface.

3.1 Adversarial Attack Model on RAG via Knowledge Poisoning

RAG System Architecture. A RAG system combines an external knowledge base and an LLM [1]. It consists of a knowledge database, a retriever, and an LLM. The knowledge database stores a collection of texts gathered from diverse sources:

$$D = \{T_1, T_2, ..., T_d\}. \tag{1}$$

When a user question Q is given, the retriever finds the top-k most relevant texts from D according to semantic similarity computed by a question encoder fQ and a text encoder fT:

$$S(Q, T_i) = \text{Sim}\big(f_Q(Q), f_T(T_i)\big), \tag{2}$$

$$E(Q; D) = \text{RETRIEVE}\big(Q, f_Q, f_T, D\big). \tag{3}$$

The retrieved results and the original query are then used by the LLM to generate the answer:

$$\text{Answer} = \text{LLM}\big(Q, E(Q; D)\big). \tag{4}$$

The advantage of RAG is dynamic integration of external knowledge, but dependence on the external knowledge base also introduces poisoning risk.

Threat Model. We analyze the threat model from attack objectives, attacker knowledge, and attacker capabilities [4, 14]. The attacker selects target questions Q = {Q1, Q2, ..., QM} and target answers R = {R1, R2, ..., RM}. Each Ri is the incorrect answer the attacker expects for question Qi. For example, for the medical query "What are the side effects of Drug X?", an attacker might set Ri to "No significant side effects" to conceal adverse reactions.

We consider two scenarios. In the black-box scenario, the attacker has no knowledge of the system's internal implementation. In the white-box scenario, the attacker can obtain the retriever encoder parameters fQ and fT, but cannot directly access the LLM parameters or the original knowledge base. For each target question, the attacker injects N malicious texts {P1, P2, ..., PN} into the knowledge base. These texts are designed so that their retrieval similarity satisfies

$$S\big(Q_i, P_j\big) > S(Q_i, T_k), \quad P_j \in \Theta,\ T_k \in D_{\text{clean}}. \tag{5}$$

An attack succeeds when the answer generated by the poisoned system matches the target answer for the corresponding target query.

Att2RAG Architecture. Let the target-question set be Q and target-answer set be R. The malicious texts constructed by the attacker are

$$\Theta = \{P_1, P_2, ..., P_N\}. \tag{6}$$

Each poisoned text must be close to the target question in embedding space and must contain semantic features that guide the model to the target answer. The attack objective is a maximization problem:

$$\max_{\theta} E_{(Q_i,R_i)}\left[I\{\text{LLM}\left(Q_i, E(Q_i; D_{\text{clean}} \cup \theta)\right) = R_i\}\right]. \quad (7)$$

The optimization is constrained by retrieval and generation requirements:

$$P_j \in \text{Top-k}(Q_i, D_{\text{clean}} \cup \theta), \quad \text{LLM}\left(Q_i, E(Q_i; D_{\text{clean}} \cup \theta)\right) = R_i. \quad (8)$$

Att2RAG therefore works through three mechanisms: at the semantic level, it keeps malicious texts close to target questions in embedding space; at the retrieval level, it controls the proportion of poisoned texts in retrieved context; and at the generation level, it optimizes textual content that guides model output.

3.2 Supplementary Explanation

Dual-Condition Theoretical Framework. A successful knowledge poisoning attack must satisfy two interrelated conditions corresponding to the two-stage RAG pipeline. The retrieval condition requires the malicious text P to be included in the top-k retrieval result set for target question Q:

$$\text{Cond}_{\text{ret}}(Q, P) = I\{P \in \text{Top-k}(Q, D)\}, \quad \text{Sim}\left(f_Q(Q), f_T(P)\right) \geq \tau. \quad (9)$$

Here τ denotes the minimum similarity threshold for successful retrieval. The generation condition requires that, when P serves as context, the LLM generates the predetermined target answer R, i.e., LLM(Q, P) = R. The two conditions may trade off against each other: over-optimizing retrieval similarity can harm naturalness and generation control, while over-optimizing instruction strength can reduce retrieval rank.

White-Box Scenario Implementation. In the white-box scenario, the attacker obtains retriever parameters, which is realistic when deployed systems use public retrievers such as Sentence-BERT or ANCE [15, 16]. We design a gradient optimization method using projected gradient descent:

$$P^{t+1} = \Pi_C\left(P^t - \eta \nabla_P J(P^t; Q, R)\right). \quad (10)$$

The objective controls the position of the malicious text in embedding space and provides a rigorous security benchmark for defense design.

Black-Box Scenario Implementation. The black-box setting reflects common real-world threats in which the attacker cannot obtain system parameters and only interacts with APIs. We use a lightweight construction based on question self-similarity:

$$P = Q \oplus I(Q, R). \quad (11)$$

The question-like part improves retrievability, while the instruction component induces the target answer. This construction is practical because it exploits two basic properties: self-similarity of questions and the ability of text concatenation to supply generation instructions.

Table 1. Datasets used in the experiments.

Dataset	Questions	Answer type	Data source	Primary use
Natural Questions	307,372	Long/short answer	Google queries with clicked Wikipedia pages	Open-domain QA training and benchmarking
HotpotQA	112,779	Answer, supporting facts, reasoning path	Multi-hop and comparative QA with 2-10 supporting documents	Multi-hop reasoning and evidence filtering

4 Experimental Results and Analysis

4.1 Dataset Introduction

The experiments use NQ and HotpotQA [17, 18], two benchmark datasets for question answering. NQ focuses on open-domain factoid QA, while HotpotQA focuses on multi-hop and comparative reasoning. Table 1 summarizes their main properties. In processing, NQ requires both paragraph localization and entity recognition because about 18% of questions lack a clear short answer. HotpotQA includes distractor documents, forcing cross-document evidence filtering. The reference paragraphs in NQ average about 1,200 words, while HotpotQA supporting documents may reach 5,000 words, requiring hierarchical encoding to balance efficiency and semantic completeness.

4.2 Experimental Setup and Evaluation Metrics

Basic Experimental Setup. The system uses Python 3.10 and PyTorch 2.0 to build a multi-stage adversarial QA workflow. Core development is performed with PyCharm 2023, and experiments are accelerated by an NVIDIA RTX 4090 GPU. The system contains three modules. The data-processing module uses prepare_dataset.py to download and decompress NQ and HotpotQA, and gen_adv.py to call the GPT-4 API for adversarial text generation; five misleading samples containing incorrect answers are generated per query. The model-construction module loads a Contriever dense encoder [19] through evaluate_beir.py, computes 768-dimensional embeddings, and performs top-k retrieval using dot-product or cosine similarity. The Attacker class in attack.py implements the targeted attack strategy and uses get_emb() to compute

perturbations in text embedding space. Generated answers are parsed by dynamically loading LLM interfaces from the models module. The training-control module uses main.py and run.py for attack configuration, multi-process scheduling, log naming, metric computation, and JSON result persistence. The end-to-end workflow is: data preprocessing, adversarial injection, model execution, and evaluation feedback.

Evaluation Metrics. We use Attack Success Rate (ASR), precision, recall, and F1-score. ASR measures the proportion of successful attacks among all attack attempts:

Table 2. Att2RAG attack effectiveness across LLMs.

Dataset	Attack	Metric	GPT-3.5	GPT-4	Llama2-7B	Llama2-13B	Vicuna-7B	Vicuna-13B
NQ	Black-box	ASR	0.92	0.97	0.97	0.95	0.88	0.95
NQ	Black-box	Target-F1				0.96		
NQ	White-box	ASR	0.99	0.99	0.96	0.95	0.96	0.96
NQ	White-box	Target-F1				1.00		
HotpotQA	Black-box	ASR	0.98	0.93	0.98	0.98	0.94	0.97
HotpotQA	Black-box	Target-F1				1.00		
HotpotQA	White-box	ASR	0.94	0.99	0.99	0.98	0.97	0.91
HotpotQA	White-box	Target-F1				1.00		

$$\text{ASR} = \frac{\#\text{Successful Attacks}}{\#\text{Attack Attempts}}. \tag{12}$$

Precision measures the proportion of truly positive samples among all predicted positives:

$$\text{Precision} = \frac{TP}{TP+FP}. \tag{13}$$

Recall measures the proportion of actual positive samples that are identified:

$$\text{Recall} = \frac{TP}{TP+FN}. \tag{14}$$

The F1-score is the harmonic mean of precision and recall:

$$F_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision}+\text{Recall}}. \tag{15}$$

4.3 Attack Effectiveness Analysis

Table 2 shows that Att2RAG achieves high ASR and Target-F1 under both black-box and white-box settings. The attack is effective across NQ and HotpotQA and across GPT, Llama2, and Vicuna model families, with maximum ASR reaching 0.99 and F1 scores close to 1.0. Although white-box attacks typically have an advantage, black-box attacks can be equally strong in some cases, indicating the influence of adversarial text construction and target-question prefixing.

4.4 Comparative Analysis with Baseline Methods

We compare Att2RAG with five baselines: Plain Attack, Corpus Poisoning Attack, Disinformation Attack, Prompt Injection Attack, and GCG Attack [6, 20]. As shown in Table 3, Att2RAG outperforms the baselines on NQ. On HotpotQA, disinformation and prompt injection are also strong, but Att2RAG remains stable and obtains high Target-F1. The results reveal that RAG systems have serious vulnerabilities when facing carefully designed adversarial attacks.

Table 3. Comparative analysis with five baseline methods.

Attack method	NQ ASR	NQ F1-score	HotpotQA ASR	HotpotQA F1-score
Plain Attack	0.03	1.00	0.06	1.00
Corpus Poisoning Attack	0.01	0.99	0.01	1.00
Disinformation Attack	0.69	0.48	1.00	0.99
Prompt Injection Attack	0.62	0.73	0.93	0.99
GCG Attack	0.02	0.00	0.01	0.00
Att2RAG Black-box	0.97	0.96	0.99	1.00
Att2RAG White-box	0.97	1.00	0.94	1.00

Table 4. Ablation study analysis of retrievers.

Dataset	Attack	Contriever ASR	Contriever F1	Contriever -ms ASR	Contriever -ms F1	ANCE ASR	ANCE F1
NQ	Black-box	0.97	0.96	0.96	0.98	0.95	0.96
NQ	White-box	0.97	1.00	0.97	1.00	0.95	0.97
HotpotQA	Black-box	0.99	1.00	1.00	1.00	1.00	1.00
HotpotQA	White-box	0.94	1.00	0.95	1.00	1.00	1.00

4.5 Ablation Study

Table 4 reports the retriever ablation on Contriever, Contriever-ms, and ANCE. Under black-box settings, ASR exceeds 0.95 for all three retrievers, and F1 scores are no lower than 0.96. Contriever-ms achieves perfect scores on HotpotQA under black-box attack, and ANCE shows full vulnerability on HotpotQA in both black-box and white-box settings. These results confirm that existing retrieval architectures generally suffer from security vulnerabilities and that attackers can bypass simple defenses through carefully crafted malicious texts.

Figure 2 analyzes how the number of retrieved documents k affects attack effectiveness. When k ≤ 5, ASR remains above 90%, precision exceeds 95%, and recall increases with k and approaches 1.0. When k > 5, ASR and precision decline linearly as k increases, but recall remains above 98%. Increasing k can mitigate attack success but may reduce retrieval efficiency, revealing a security-performance trade-off.

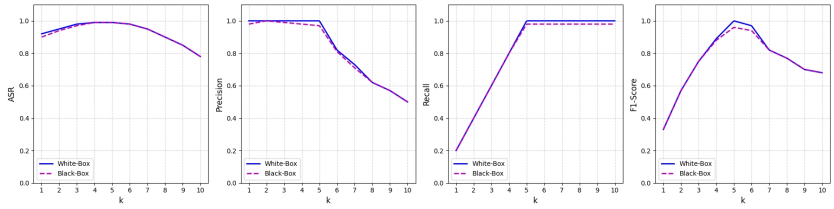

Fig. 2. Analysis of the impact of retrieval number k on attack effectiveness.

4.6 Defense Effectiveness Analysis

Text paraphrasing provides partial protection against Att2RAG attacks. As shown in Table 5, on NQ it reduces black-box ASR from 0.97 to 0.87 and reduces F1 from 0.96 to 0.83. For white-box attacks, ASR decreases from 0.97 to 0.93 and F1 also declines. On HotpotQA, the ASR of white-box attacks decreases by 8.5 percentage points, but F1 remains 1.0, showing that semantic rewriting weakens but does not break the attack chain.

Table 5. Text paraphrasing defense analysis.

Dataset	Attack	No defense ASR	No defense F1-score	With defense ASR	With defense F1-score
NQ	Black-box	0.97	0.96	0.87	0.83
NQ	White-box	0.97	1.00	0.93	0.94
HotpotQA	Black-box	0.99	1.00	0.93	1.00
HotpotQA	White-box	0.94	1.00	0.86	1.00

Duplicate text filtering is largely ineffective. Table 6 shows that applying this defense causes no decrease in ASR or F1 in the tested scenarios. This indicates that Att2RAG evades repetition-based detection by generating semantics-preserving but lexically and syntactically diverse malicious texts. Therefore, output-side and duplicate-based defenses are insufficient; semantic-level attacks require semantic-level defenses, especially at the retrieval stage.

Table 6. Duplicate text filtering defense analysis.

Dataset	Attack	No defense ASR	No defense F1-score	With defense ASR	With defense F1-score

NQ	Black-box	0.97	0.96	0.97	0.96
NQ	White-box	0.97	1.00	0.97	1.00
HotpotQA	Black-box	0.99	1.00	0.99	1.00
HotpotQA	White-box	0.94	1.00	0.94	1.00

5 Conclusion

This study constructs Att2RAG, a knowledge poisoning attack framework for RAG systems. The key innovation is transforming the poisoning attack into an optimization problem: the attacker searches for an optimal set of malicious texts that can both be retrieved by the retriever and induce the model to generate predetermined malicious content. Att2RAG surpasses the limitations of traditional adversarial attacks that target model parameters, providing greater stealthiness and a higher attack success rate.

Experimental results show that Att2RAG achieves strong performance under black-box and white-box settings. On NQ, injecting only five malicious documents per target question leads to an ASR of 97% and an F1 score of 0.96. Compared with baselines such as disinformation attacks and prompt injection attacks, Att2RAG improves ASR by more than 30 percentage points in the NQ setting. The attack remains effective across retrievers including Contriever and ANCE and evades defenses such as text paraphrasing and duplicate text filtering. These results confirm that Att2RAG poses a significant security threat to current RAG systems and that future defenses should build end-to-end protection, especially semantic-level detection and verification in the retrieval stage.

Disclosure of Interests. The authors have no competing interests to declare.

References

1. Lewis, P., Perez, E., Piktus, A., et al.: Retrieval-augmented generation for knowledge-intensive NLP tasks. In: Advances in Neural Information Processing Systems, vol. 33, pp. 9459-9474 (2020)
2. Dong, S., Xu, S., He, P., Li, Y., Tang, J., Liu, T., Liu, H., Xiang, Z.: A practical memory injection attack against LLM agents. arXiv:2503.03704 (2025)
3. Hatalis, K., Christou, D., Myers, J., Jones, S., Vijayaraghavan, V., Makedon, F.: Memory matters: The need to improve long-term memory in LLM-agents. In: Proceedings of the AAAI Symposium Series, vol. 2, pp. 277-280 (2023)
4. Zou, W., Geng, R., Wang, B., Jia, J.: PoisonedRAG: Knowledge corruption attacks to retrieval-augmented generation of large language models. arXiv:2402.07867 (2024)
5. Chen, Z., Xiang, Z., Xiao, C., Song, D., Li, B.: AgentPoison: Red-teaming LLM agents via poisoning memory or knowledge bases. In: Advances in Neural Information Processing Systems, vol. 37, pp. 130185-130213 (2024)
6. Greshake, K., Abdelnabi, S., Mishra, S., Endres, C., Holz, T., Fritz, M.: Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt

injection. In: Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security, pp. 79-90 (2023)
7. Cohen, S., Bitton, R., Nassi, B.: Here comes the AI worm: Unleashing zero-click worms that target GenAI-powered applications. arXiv:2403.02817 (2024)
8. Zhang, H., Huang, J., Mei, K., et al.: Agent Security Bench (ASB): Formalizing and benchmarking attacks and defenses in LLM-based agents. arXiv:2410.02644 (2024)
9. Deng, Z., Guo, Y., Han, C., Ma, W., Xiong, J., Wen, S., Xiang, Y.: AI agents under threat: A survey of key security challenges and future pathways. ACM Computing Surveys 57(7), 1-36 (2025)
10. Wang, K., Zhang, G., Zhou, Z., et al.: A comprehensive survey in LLM(-Agent) full stack safety: Data, training and deployment. arXiv:2504.15585 (2025)
11. Liu, Y., Yuan, Z., Tie, G., Shi, J., Sun, L., Gong, N.Z.: Poisoned-MRAG: Knowledge poisoning attacks to multimodal retrieval augmented generation. arXiv:2503.06254 (2025)
12. Maharana, A., Lee, D.H., Tulyakov, S., Bansal, M., Barbieri, F., Fang, Y.: Evaluating very long-term conversational memory of LLM agents. In: Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics, pp. 13851-13870 (2024)
13. Pan, Z., Wu, Q., Jiang, H., et al.: SeCom: On memory construction and retrieval for personalized conversational agents. In: International Conference on Learning Representations (2025)
14. Tan, X., Luan, H., Luo, M., Sun, X., Chen, P., Dai, J.: Knowledge database or poison base? Detecting RAG poisoning attack through LLM activations. In: Findings of the Association for Computational Linguistics: EMNLP (2025)
15. Reimers, N., Gurevych, I.: Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In: Proceedings of EMNLP-IJCNLP, pp. 3982-3992 (2019)
16. Xiong, L., Xiong, C., Li, Y., et al.: Approximate nearest neighbor negative contrastive learning for dense text retrieval. In: International Conference on Learning Representations (2021)
17. Kwiatkowski, T., Palomaki, J., Redfield, O., et al.: Natural Questions: A benchmark for question answering research. Transactions of the Association for Computational Linguistics 7, 453-466 (2019)
18. Yang, Z., Qi, P., Zhang, S., Bengio, Y., Cohen, W.W., Salakhutdinov, R., Manning, C.D.: HotpotQA: A dataset for diverse, explainable multi-hop question answering. In: Proceedings of EMNLP, pp. 2369-2380 (2018)
19. Izacard, G., Caron, M., Hosseini, L., Riedel, S., Bojanowski, P., Joulin, A., Grave, E.: Unsupervised dense information retrieval with contrastive learning. arXiv:2112.09118 (2021)
20. Zou, A., Wang, Z., Carlini, N., Nasr, M., Kolter, J.Z., Fredrikson, M.: Universal and transferable adversarial attacks on aligned language models. arXiv:2307.15043 (2023)