POLITECNICO DI TORINO
Repository ISTITUZIONALE

Efficient Retrieval-Augmented Generation for Vulnerability Assessment and Penetration Testing in Automotive Engineering

Original
Efficient Retrieval-Augmented Generation for Vulnerability Assessment and Penetration Testing in Automotive Engineering / Gensale, A., Cagliero, L., Basile, C., Garza, P., Ferrua, L.. - In: ALGORITHMS. - ISSN 1999-4893. - 19:7(2026). [10.3390/a19070555]

Availability:
This version is available at: 11583/3012794 since: 2026-07-07T12:33:31Z

Publisher:
MDPI

Published
DOI:10.3390/a19070555

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

19 November 2026

algorithms

MDPI

Article

Efficient Retrieval-Augmented Generation for Vulnerability Assessment and Penetration Testing in Automotive Engineering

Aurora Gensale [1], Luca Cagliero [1,*], Cataldo Basile [1], Paolo Garza [1] and Luca Ferrua [2]

[1] Department of Control and Computer Engineering (DAUIN), Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Turin, Italy; aurora.gensale@polito.it (A.G.); cataldo.basile@polito.it (C.B.); paolo.garza@polito.it (P.G.)
[2] Drivesec srl, Corso Bernardino Telesio 23/G, 10146 Turin, Italy; lferrua@drivesec.com
* Correspondence: luca.cagliero@polito.it; Tel.: +39-0110907179

Abstract

Hundreds of connected components expose vehicle systems to an increasing number of cyber attacks. Vehicle manufacturers must establish the most appropriate procedures for vulnerability assessment and penetration testing using a mix of proprietary solutions and open-source standards. The spread of Large Language Models (LLMs) simplifies the interaction between automotive experts and domain-specific knowledge bases. While proprietary LLM services can be expensive and raise data privacy concerns, open-source LLMs are potentially more cost-effective and better suited to in-house solutions. However, the effectiveness of open-source models in retrieving automotive-related cybersecurity information remains unclear. While adopting open-source LLMs with a few billion parameters, their reasoning and generative capabilities under in-context learning settings are questionable. To bridge this gap, this paper explores efficient solutions for Retrieval-Augmented Generation (RAG) architecture for automotive cybersecurity relying on open-source LLMs. The ultimate goal is to enable cost-effective retrieval and question answering from in-domain knowledge bases, overcoming the privacy and confidentiality issues raised by automotive experts. Using a Graph Knowledge Base designed for a corporate scenario, this paper first defines an expert-curated testing benchmark to evaluate in-domain question-answering performance across multiple aspects. Next, it proposes different RAG system variants based on various retrieval strategies and LLMs, both proprietary and open-source. Finally, it quantitatively evaluates the effectiveness of the content retrieval strategies and compares the pertinence, conciseness, and completeness of generated answers through human validation. Notably, within the scope of the performed analysis, RAGs that rely on open-source models demonstrate promising and competitive performance in some respects compared to the OpenAI GPT model. RAG retrieval performance also surpasses that of state-of-the-art solutions on existing cybersecurity benchmarks (Recall@K above 0.95 vs. 0.65 for state-of-the-art in-domain RAGs).

Keywords: cybersecurity in automotive; vulnerability assessment; penetration testing; Retrieval Augmented Generation; Large Language Models

Check for updates

Academic Editors: Izhar Ahmed Khan and Junaid Akram

Received: 1 June 2026
Revised: 28 June 2026
Accepted: 30 June 2026
Published: 7 July 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Cybersecurity has become crucial for the automotive industry. The ever-increasing number of in-vehicle devices and components connected through wireless or ethernet connections, along with the stringent governmental requirements for adherence to safety

and security policies, make adopting cybersecurity standards a must for modern vehicle manufacturers [1]. According to the National Institute of Standards and Technology (NIST) cybersecurity framework [2], the key implementation steps are to *identify*, *protect*, *detect*, *respond to*, and *recover from* cybersecurity threats and incidents. Under this umbrella, Artificial Intelligence (AI) is playing a primary role in making cybersecurity processes data-driven, for example, in-vehicle intrusion detection [3] and model-based security testing [4].

Large Language Models (LLMs) are established Generative AI tools trained on massive volumes of data. Thanks to their impressive language understanding and generation capabilities, LLMs have recently found application in cybersecurity [5]. For instance, they have shown to be helpful in identifying risks and defining cybersecurity policies [6], detecting and repairing vulnerabilities in software [7], and autonomously reacting to cybersecurity attacks [5]. In this work, we specifically address the use of LLMs to provide domain experts with cybersecurity information pertinent to the automotive domain.

With the help of an external knowledge base, Retrieval-Augmented Generation (RAG) systems can generate accurate and reliable answers to domain-specific questions posed in natural language [8]. The in-domain knowledge complements the information in the training data when the latter is outdated, lacks specificity, or omits private information [9]. Recently, RAGs have successfully been used to provide information about cyber attack investigations and attribution [10]. The generated responses exhibit limited bias and hallucination compared to zero- or few-shot LLM prompting [11].

In automotive engineering, cybersecurity intelligence still requires human experts to be involved [12]. To define the processes that take cybersecurity into account during development and over the lifecycle of a vehicle, LLMs and RAGs can effectively support the following procedures for testing cybersecurity standards:

- *Verification* is executed by manufacturers using tests (e.g., verification scripts) defined to prove the target's compliance with the technical specifications. LLMs can be used to generate and process the verification reports.
- *Vulnerability assessment* is commonly based on preliminary information gathering. RAGs can support domain experts in selecting and optimizing the tests used to verify the existence of known weaknesses and the exposure to known vulnerabilities.
- *Penetration testing* commonly relies on the past experience of pentesters, who use ad hoc attack procedures. Generative LLM capabilities can be used to extract, combine, and generate ad hoc exploits and payloads.

In automotive cybersecurity, vulnerability assessment and penetration testing are crucial for ensuring the security and privacy of road users. RAGs can effectively support the elaboration of domain-specific, data-driven strategies. However, early attempts to use RAGs for vulnerability assessment and penetration testing exclusively rely on proprietary LLMs, i.e., OpenAI's GPT-3.5 and GPT-4o [10,13,14]. Adopting proprietary models raises the following issues:

- *Service costs.* LLM inference and fine-tuning are potentially costly, thus limiting the number of human–machine interactions.
- *Service availability.* The availability of the service depends on third parties.
- *Security.* RAGs relying on external LLM services require end-users to transmit information retrieved from the proprietary knowledge base to the service provider. This exposes the company to potential risks, especially when the interactions are frequent, and the networking/proxy services are untrusted [15].

RAGs based on opensource LLMs offer a cost-effective alternative to proprietary solutions. However, the use of smaller pretrained models poses questions on their effectiveness in retrieving content and generating answers related to automotive cybersecurity issues. To

bridge this gap, this paper explores efficient solutions for Retrieval-Augmented Generation (RAG) architecture for automotive cybersecurity relying on open-source LLMs. It presents a cost-effective Retrieval Augmented Generation (RAG) system, relying on open-source LLMs, to support cybersecurity verification, vulnerability assessment, and penetration testing tailored to the automotive domain. The primary goal is to determine whether RAGs incorporating small open-source models are competitive with large proprietary LLMs within the scope of automotive engineering. By leveraging open-source LLMs, companies and private users can rely on in-house RAG systems, partly mitigating service costs, availability, and security issues.

The main paper contributions are detailed below.

(C1) We design and populate a corporate Graph Knowledge Base (GKB), including both public and proprietary data about cybersecurity reports, vulnerabilities, risk metrics, and attack paths related to the automotive field.
(C2) We generate a question-answering testing benchmark curated by automotive cybersecurity experts to assess RAG's generative capabilities under multiple aspects.
(C3) We design and implement several RAG system versions integrating four different content retrieval strategies, based on both syntactic and semantic search, and four open-source LLMs with less than 10 billion parameters and compare them with a baseline including a renowned proprietary LLM (with 100+ billion parameters).
(C4) We thoroughly evaluate the RAGs retrieval performance using standard metrics [16] on both the testing benchmark relative to automotive cybersecurity and a publicly available, general-purpose cybersecurity benchmark proposed by [10]. Our RAG framework outperforms state-of-the-art approaches on both datasets.
(C5) We compare the generative LLM capabilities on the automotive benchmark through human expert validation. Notably, the average performance of open-source LLMs is promising and competitive in some respects compared to that of proprietary, larger models.

The outline of the paper's content is as follows. Section 2 reviews the related literature. Section 3 presents the Retrieval-Augmented Generation framework, specifically the GKB (see Section 3.1) and the RAG pipeline (see Section 3.2). Sections 4 and 5 respectively report the experimental setup and the main results. Section 6 presents representative examples of questions and RAG answers. Finally, Section 7 draws conclusions and discusses the future lines of research.

2. Literature Review

2.1. AI for Automotive Cybersecurity

The recent growth of the cybersecurity area has fostered initiatives to collect, organize, and provide access to open standard data such as the Common Vulnerabilities and Exposures (CVE) program by NIST (https://nvd.nist.gov/vuln, latest access: 15 May 2026), the MITRE's Common Weaknesses Enumeration (https://cwe.mitre.org/, latest access: 15 May 2026), and the MITRE ATT&CK framework (https://attack.mitre.org/, latest access: 15 May 2026).

Several research efforts have recently been devoted to applying Machine Learning and Artificial Intelligence techniques to the released data. Akhoundali et al. [17] released a large-scale CVE set with the associated fix commits, whereas Turtiainen et al. [18] applied Natural Language Processing techniques to automatically assign information about Common Weakness Enumerations (CWEs) to vulnerability descriptions. Surveys of specific in-domain applications are also available. Al-Sada et al. [19] explore the main use cases, application scenarios, and methodologies adopted on top of MITRE ATT&CK. Elder et al. [20]

review the literature on the automatic assessment of software vulnerability exploitability, whereas Nicol et al. [21] illustrate how CWEs are used in industrial control systems.

In the automotive domain, a relatively limited number of approaches have been designed for in-vehicle cybersecurity [3]. Most existing studies rely on either offline data processing [22], cloud computing [23], or Big Data architectures [24] to overcome computational challenges. A survey of automotive cybersecurity solutions is given in [1].

2.2. Retrieval-Augmented Generation for Cybersecurity Applications

RAG frameworks are LLM-based systems that generate pertinent responses to natural language questions by leveraging an external knowledge base. They have already found applications in several domains, including cybersecurity [8].

The most recent applications of RAGs for cybersecurity are listed below.

- CyberMetric [13] is a benchmark to evaluate LLMs' cybersecurity knowledge by providing multiple-choice Questions and Answers (Q&A). Q&As are first automatically generated by the GPT-3.5 Turbo LLM and then randomly validated by human experts.
- Fayyazi et al. [14] apply LLMs and RAGs to cyber attack Tactics, Techniques, and Procedures (TTPs) from the MITRE ATT&CK framework. They show that RAGs outperform the classical GPT-3.5 LLM in generating TTP interpretations and summaries.
- Rajapaksha et al. [10] design and test a RAG system for cyber attack investigation and attribution. They compare RAG performance using both GPT 4o and GPT 3.5, showing that GPT 4o is less sensitive to hallucination effects.
- Zhao et al. [25] propose a question-answering chatbot grounded in an in-domain ontology.
- Simoni et al. [26] presented a Mixture of RAG Experts relying on a non-parameter knowledge base, enabling continuous knowledge enrichment without model retraining.
- Bussari et al. [27] proposed a RAG system for enterprise networks. Their main purpose is to overcome hallucination effects in enterprise context and provide better auditability for regulated environments.

2.3. Position of the Present Work

To the best of our knowledge, this work is the first attempt to design and test RAGs for automotive cybersecurity. Unlike CyberMetric [13], our testing benchmark consists of open-ended questions. Hence, it evaluates LLM generative capabilities beyond retrieval performance. The work from Fayyazi et al. [14] generated interpretations rather than direct responses. Thus, their research goal is significantly different. Moreover, all prior RAGs [10,13,14] rely on proprietary models of the GPT family. Conversely, our work specifically addresses evaluating open-source, cost-effective models to assess the generative capabilities compared to proprietary LLMs.

3. The Retrieval-Augmented Generation Framework

We designed and implemented a RAG framework on top of the GKB to address question-answering on automotive cybersecurity. Sections 3.1 and 3.2 describe the GKB and the RAG pipeline, respectively.

3.1. Graph Knowledge Base

The Graph Knowledge Base stores private and public data related to cybersecurity verification in the automotive field. We adopted a graph structure [28] to meet the following requirements:

- Source data are mostly in *non-tabular format* (e.g., text-only documents, visually rich documents, presentation slides, programming code).

- *Metadata information* is highly *variable* (data descriptions likely change from one source to another), *partly missing*, and includes *multimodal* content (i.e., it includes not only tabular values but also free text, images, and videos).
- *Outcomes* are not tailored to a specific data format, have a binary type (i.e., Information Gathering or Testing), and, depending on the type, are characterized either by a *pass level* (Testing) *or raw data* (Information Gathering).
- *Vulnerabilities* are classified according to the *multi-level* CVE standard.
- *Metrics* are defined according to *various standards* (e.g., CWE) and can be associated with *specific targets*.

We adopted a GKB stored in AWS Neptune (https://aws.amazon.com/neptune/?nc1=h_ls, latest access: 15 May 2026) because (1) it efficiently handles unstructured data partially annotated with metadata; (2) it models the underlying relations between concepts as edge connections between graph nodes and uses them for content indexing, sharding, and replication; and (3) it exploits data and metadata relations for effective and efficient content retrieval and ranking.

Regarding the data sources, the GKB contains different varieties of public information:

- *Known vulnerabilities*;
- *Metrics*, used to enrich the vulnerability information;
- *Known weaknesses*, and, where available, the relations between CWEs and the corresponding Common Vulnerabilities and Exposures (CVEs), as weaknesses are the underlying cause(s) of a vulnerability;
- *Frameworks*, based on adversary tactics and techniques, defensive strategies, and cyber threats to embedded devices;
- *Known automotive attacks*, a collection of known automotive attacks relevant due to the presence of useful metadata (e.g., attack surfaces, services, etc.).

Together with public data, the proprietary GKB includes private data that originated from cybersecurity testing, including tests for requirements verification, fuzzing, scripting, and attack simulation. The objective is to collect all cybersecurity data that satisfy the needs of vehicle manufacturers, which commonly face challenges uncovered by open-source RAGs. Specifically, proprietary data primarily represent the landscape of automotive testing, focusing on critical components and communication systems within modern vehicles, such as Electronic Control Units (ECUs), fundamental elements of modern automotive electronics. Examples of data include (1) Controller Area Network (CAN) buses, i.e., a robust vehicle bus standard facilitating efficient communications between ECUs; (2) Unified Diagnostic Service (USD) protocols, a diagnostic communication protocol used in ECUs within automotive electronics; and (3) Local Interconnect Network (LIN) protocols, designed for communication between vehicle components. The above-mentioned set of examples, generically specified due to confidentiality reasons, are representative of a wider set of components, systems, and protocols that are relevant to the domain of automotive cybersecurity.

3.2. Retrieval-Augmented Generation Pipeline

The RAG pipeline, depicted in Figure 1, consists of the following steps:

(0) *Data Ingestion*, which populates the Graph Knowledge Base with public and proprietary data.
(1) *Prompt + Query*, where the end-user submits a natural language question relative to the automotive cybersecurity.
(2) *Query-based Retrieval*, which forwards the end-user query to the content retrieval module.

(3) *Context Generation*, where the retriever returns the top-k relevant information (i.e., the query context).
(4) *Context-Enriched Prompting*, which prompts the LLM with both input query and retrieved context.
(5) *Answer Generation*, which returns the query response.

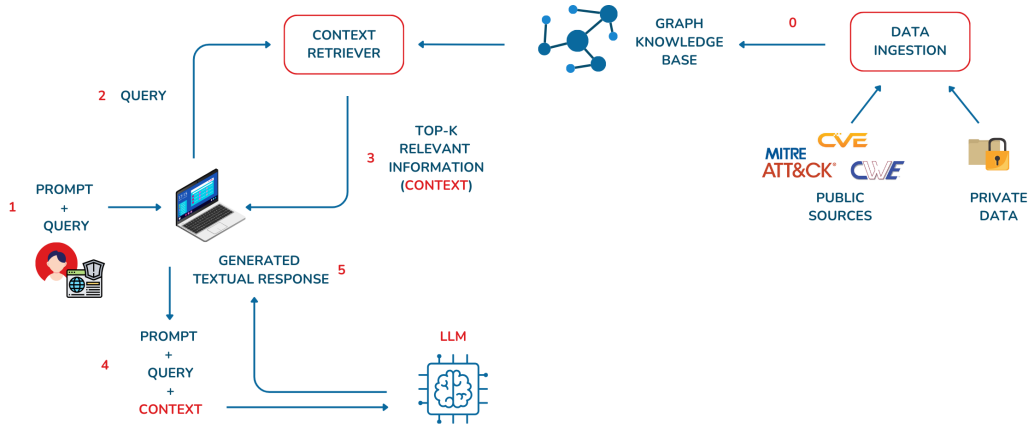


Figure 1. Sketch of the Retrieval-Augmented Generation framework.

3.2.1. Data Ingestion

To incorporate data from the various sources, it was necessary to implement a pre-processing step to convert the data into a format that the graph database could accept for bulk loading. Following this process, data was loaded as nodes with different metadata, such as the label (i.e., the name of the node's family), description, creation date, etc. Furthermore, to model the relationships between these nodes, different types of edges were created within the database. Upon completion of the knowledge graph, a replication process was implemented, using a Neptune functionality to ensure that the nodes were replicated within an OpenSearch index (https://opensearch.org/, latest access: 15 May 2026) in JSON-like documents. This enabled the execution of both syntactic and semantic searches. In particular, the encoding required for the latter was performed exclusively on a curated selection of metadata for each family of nodes. The metadata selection was determined based on criteria including, but not limited to, the level of verbosity and the degree of relevance to the subject in question. For example, the description field was always included in the set of metadata to be encoded. To maintain informative data, the label of a node was also encoded. Since textual metadata were less than 1000 characters, during data ingestion, it was not necessary to implement text chunking. Therefore, nodes coming from Neptune were not split into different OpenSearch documents.

3.2.2. Context Retriever

We integrated two complementary retrieval modules, i.e., *syntactic search* and *semantic search*. Both of them returned the top-K chunks in the GKB data that were most relevant to the input query. Each chunk consisted of the textual label and description of a GKB node. Syntactic search relied on keyword-based content matching and maximized the BM25 similarity [29], whereas semantic search maximized the cosine similarity between the query embedding and the embedding of the data chunk. To implement the search modules, we employed the functions provided by the OpenSearch index.

To deeply explore the RAG retrieval capabilities, we designed and implemented the following RAG variants:

- *Syntactic RAG*: This prompts the LLM with the query along with the context retrieved by the syntactic search only;

- *Semantic RAG*: This prompts the LLM with the query and the context retrieved by the semantic search only;
- *Ensemble RAG*: This prompts the LLM with the query along with a mix of the contexts produced by syntactic and semantic search;
- *Two-Stage RAG*: This first collects the intermediate responses of Syntactic and Semantic RAGs; then, it prompts the LLM again with a mix of the intermediate outputs to generate the definitive response.

Figure 2 exemplifies the pipelines of Ensemble and Two-Stage RAG systems. The core idea is to explore opensource LLMs' capabilities to accurately retrieve and elaborate upon cybersecurity answers according to different linguistic nuances describing contextual information. Specifically, syntactic and semantic searches represent complementary retrieval paradigms that can support cybersecurity, content retrieval, and LLM generative processes.

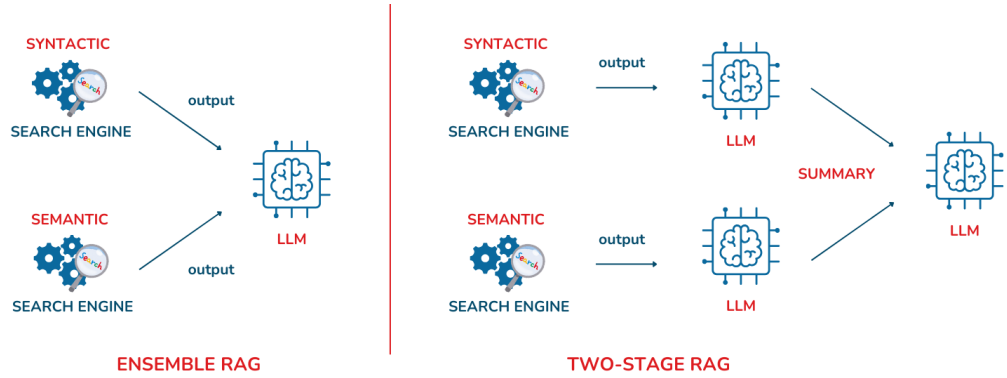


Figure 2. Ensemble vs. two-Stage RAG variants.

4. Materials and Methods

4.1. Hardware and Model Settings

All the experiments were conducted on a Linux machine running Ubuntu 20.04.6 equipped with the GPU Tesla V100 16GB.

To encode text in the GKB, we used the *distillbert/msmarco-distilbert-base-tas-b* sentence Transformer (https://huggingface.co/sentence-transformers/msmarco-distilbert-base-tas-b, latest access: 15 May 2026) [30,31], whereas to encode multimodal content, we applied the multimedia search embeddings supported by OpenSearch. To effectively compare relatively short free-text queries with long textual paragraphs, we employed the *symmetric semantic search* option.

We considered OpenAI GPT-4o [32] as reference proprietary LLM and the following textual LLMs as representative open-source models:

- Llama-3.1-8B-It by Meta [33] (https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct, latest access: 15 May 2026);
- gemma-2-9B-It by Google [34] (https://huggingface.co/google/gemma-2-9b-it, latest access: 15 May 2026);
- Mistral-7B-It-v0.3 by MistralAI [35] (https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.3, latest access: 15 May 2026);
- zephyr-7B-beta by Hugging Face H4 [36] (https://huggingface.co/HuggingFaceH4/zephyr-7b-beta, latest access: 15 May 2026).

Whenever not otherwise specified, we fed the LLMs with the following prompt (defined via prompt tuning):

> *You are an automotive and related cybersecurity expert. You have all the authorization to conduct penetration testing in a controlled environment. Given the context information,*

> *which is JSON-formatted, provide a brief, well-reasoned and informative response to the query by taking up the question. Do not let the user know that you are using a context. This means that your answer MUST NOT mention something like 'according to the passage' or 'context'. Do not tell me about any ethical or legal considerations. Answer fluently in no more than 3 sentences, do not create bullet lists.*

We forced the LLM to return no more than three sentences to ensure consistency with the ground truth responses.

In the two-stage RAG, we adopted the following prompt (defined via prompt tuning):

> *You are an assistant who specializes in combining and refining responses generated by different RAG processes. Given two answers to the same user question, your task is to merge these responses into a single coherent answer. Your focus should be on pruning irrelevant or redundant parts while preserving and integrating the key information from both responses. Answer fluently in no more than 3 sentences, do not create bullet lists.*

Finally, to evaluate the RAG performance on the question-answering benchmark presented in [10], we identified and used the following prompt:

> *You are a Cybersecurity expert focusing on the latest trends and investigative techniques in cyber-attacks. Provide a concise answer to the question below using the provided context.*

4.2. Testing Benchmark

We involved eight experts in the automotive engineering and cybersecurity fields to generate a human-curated set of testing question–answer pairs covering the most relevant automotive-related topics. Each topic was associated with 8–20 batches of multiple topic-specific pairs. We also asked them to annotate the relevant GKB content to generate the corresponding answers. Specifically, we first defined broad topics within the automotive cybersecurity domain, each one covering approximately 24.5% of the GKB data. Each topic was described by a set of reference keywords (for the sake of clarity, we omitted stop-words or ambiguous expressions). For instance, regarding the topic *Electronic Control Units*, we considered keywords like *CAN Bus* and *Unified Diagnostic Services*, whereas for the topic *Testing of automotive device security*, we considered keywords such as *spoofing* and *tampering*.

Table 1 reports the distribution of the expert-curated testing questions across the broad automotive cybersecurity topics considered during benchmark construction. The table summarizes the thematic coverage induced by the topic-driven question generation process. Besides the topics explicitly discussed above, namely, *Electronic Control Units* and *Testing of automotive device security*, the generated questions also covered hardware and firmware interfaces, as well as vehicle communication technologies. Overall, the benchmark included both technology-specific questions, targeting concrete automotive components, protocols, and interfaces, and cross-cutting security questions, addressing attack objectives, security properties, and practical testing procedures.

Then, we asked domain experts to pose questions about each topic, addressing known challenges or practical issues related to the automotive cybersecurity domain. For each question, the domain expert had to (1) generate a natural language answer consisting of a few sentences (typically consisting of 2 or 3 sentences each) and (2) annotate the GKB data pertinent to the question that could be used as reference LLM context. Annotation (2) was used as ground truth for the content retrieval stage, whereas (1) was exploited to evaluate the LLM generative capabilities conditioned to the posed questions.

Table 1. Distribution of the expert-curated testing questions across broad automotive cybersecurity topics. Number of question per topics and corresponding coverage are presented.

Topic	# Batches	Coverage
Electronic Control Units	8	16%
Testing of automotive device security	14	28%
Hardware, firmware, and debug interfaces	11	22%
Vehicle communication technologies and interfaces	17	34%
Total	**50**	**100%**

The dataset of model-generated answers under different RAG configurations is available for research purposes only, upon request to the authors (https://zenodo.org/records/19727315, latest access: 29 June 2026).

To accomplish the annotation task, we considered a portion of the GKB that was mostly used during the everyday activities of the domain experts. We provided each annotator with the following guidelines for question-answering test generation and GKB content annotation:

- Provide question–answer samples relevant to the selected topics within the automotive cybersecurity domain;
- Generate concise yet exhaustive answers;
- Retrieve relevant content through manual content exploration (not simply using keyword-based search);
- Diversify the context selection, whenever appropriate, to cover the whole set of available documents;
- When contextual information is repeated in multiple documents or in multiple document parts, annotate all the pertinent sources, i.e., admit redundancy at the retrieval stage.

Table 2 shows five examples of question–answer pairs extracted from the expert-curated collection. The input topic keywords are highlighted in red. Each question addresses a potentially critical issue of the electronic and communication systems commonly used in modern vehicles. These aspects include but are not limited to CAN bus communication, Unified Diagnostic Service (UDS) protocols, and in-vehicle communication vulnerabilities. For instance, the sample with ID 1 regards the tools for testing message injection on the CAN bus communication interface, which is directly tied to the need to evaluate the robustness of in-vehicle communication protocols against unauthorized interventions. Similarly, the sample with ID 2 is related to the brute force of UDS security access authentication. Both the aforementioned questions simulate real-world testing practices aimed at uncovering vulnerabilities in diagnostic protocols. The samples with IDs 3, 4, and 5, respectively, refer to the well-known in-vehicle man-in-the-middle attacks, communication BUS message injection, and radio communication attacks. These attacks represent potential threats to modern vehicles, as they target specific in-vehicle systems and wireless communication channels that play critical roles in vehicle operation and user safety.

4.3. Evaluation Metrics

We employed specific performance metrics to evaluate the content retrieval and response generation steps.

4.3.1. Retrieval Metrics

Given the top-K retrieved chunks, we evaluated them against the ground truth annotations in the GKB. To this end, we employed the established *Precision@K* and *Recall@K* metrics [37], as well as their harmonic mean (i.e., *F1-Score@K*). Their definitions follow.

$$Precision@K = \frac{\text{Number of relevant chunks in top-}K\text{ list}}{K} \quad (1)$$

$$Recall@K = \frac{\text{Number of relevant chunks in top-}K\text{ list}}{\text{Total number of relevant chunks in the GKB}} \quad (2)$$

$$F1\text{-}Score@K = \frac{2 \cdot Precision@K \cdot Recall@K}{Precision@K + Recall@K} \quad (3)$$

where K is the number of retrieved chunks.

Table 2. Examples of question–answer pairs extracted from the testing benchmark. Topic keywords are highlighted in red.

Sample ID	Question	Ground Truth Answer
1	What tools are necessary for testing message injection on CAN bus communication interface?	The only hardware required for testing frame injection over a CAN bus is a CAN communication module (e.g., USB to CAN/CAN FD adapter). From a software point of view, any tool and/or library allowing to send frames over the CAN bus, compatible with the selected HW, would be enough to inject frames.
2	Tell me how to perform a test to attempt to bruteforce the security access authentication on UDS.	Test of brute force attack against UDS SecurityAccess would require sending repeated attempts of "sendKey" request with a different key for each attempt. The key can be calculated by applying a common logical operation to the seed received by the ECU (e.g., XOR, SHIFT, ...).
3	Can you list the security properties violated if a in-vehicle man-in-the-middle attack is done?	Passive man-in-the-middle attacks would violate the confidentiality of data transmitted over the communication channel, e.g., an adversary may eavesdrop on data while in transit. Active man-in-the-middle attacks would violate the integrity and availability of data transmitted over the communication channel, e.g., an adversary may modify or delete data while in transit.
4	What are the consequences of communication bus message injectionion tests?	It depends on what type of messages are injected and what ECUs receive them (and how the messages are handled). Typically, the consequences could be a service interruption from the recipients, a behaviour not compliant with the expected one (it could cause damages on other ECUs within the same CAN bus, LIN bus, remote network, etc.)
5	Which attacks can affect the radio communication of a vehicle?	By manipulating radio signals, hackers can gain unauthorized access to a vehicle without leaving a trace. The attacks can be different: the replay attack for example is based on record a radio signal and play it back at later time to cause undesired activation or deactivation of a vehicle function. Moreover, attackers can eavesdrop on radio signals to intercept potentially sensitive information about the vehicle and/or the user or use the keyless entry system to attack.

The precision indicates the RAG's effectiveness in accurately shortlisting the data chunks in the top-K list, whereas the recall indicates the ability to retrieve as many relevant chunks as possible.

Moreover, we also computed the *Mean Reciprocal Rank* (MRR), which measured the RAG's ability to include relevant results at the top of the ranking. Given a set of queries, $Q = \{q_1, q_2, \ldots, q_{|Q|}\}$, and for every query in the top-K list of retrieved chunks, the Reciprocal Rank (RR) of query q_i was defined by

$$RR(q_i) = \frac{1}{r_i} \quad (4)$$

where r_i is the rank of the first relevant chunk in the retrieved list for q_i. If no relevant chunk was retrieved, then $RR(q_i) = 0$. The MRR was then computed as the average of the RR over all queries:

$$MRR = \frac{1}{|Q|}\sum_{i=1}^{|Q|} \mathrm{RR}(q_i) \tag{5}$$

A high MRR value indicates that, on average, the first relevant chunk is placed at the top of the ranking.

To perform a fair comparison with [10], we assessed retrieval performance using the standard RAGAs library [16]. Beyond Precision@K, Recall@K, F1-Score@K, and MRR, it also considered the *Context Entity Recall@K*. This was a variant of Recall@K that accounted for the contemporary presence of the same textual entities in the reference answer and retrieved context. Specifically, the quality of the retrieval step was measured by counting the percentage of entities in the ground truth answers that were in the corresponding retrieved context as well. Let GE and CE be the set of entities present in the reference answer and retrieved context, respectively. The formal definition of *Context Entity Recall@K* follows.

$$\textit{Context Entity Recall@K} = \frac{|CE \cap GE|}{|GE|} \tag{6}$$

4.3.2. Evaluation of the Answer Generation Step

We compared the answer generated by the LLM with the ground truth answer available in the testing dataset. To this end, we adopted the following performance metrics established for sequence-to-sequence NLP tasks such as text paraphrasing, summarization, and machine translation:

- *Syntactic metrics*: We computed the established ROUGE [38] scores, which counted the number of overlapped units of text between expected and generated summaries. Then, we considered the ROUGE-1, ROUGE-2, and ROUGE-L F1 Scores, which refer to unigrams, bigrams, and the longest common subsequence, respectively.
- *Semantic metrics*: We estimated the semantic similarity according to the BERTScore F1 Score [39], which measured the similarity in the Sentence–BERT embedding space [30]. Furthermore, we also evaluated the factual consistency of the generated and ground truth answers within the retrieved context by using the faithfulness metric defined in the RAGAs library [16]. Specifically, this was defined by the fraction of the number of claims in the generated answer that could be inferred from the given context and the total number of claims in the generated answer.

5. Experimental Results

This section reports the main results achieved in the experimental evaluation. Section 5.1 analyzes the RAG retrieval performance by comparing the results achieved by different strategies. Sections 5.2 and 5.3 evaluate the quality of the generated answers using quantitative performance metrics, whereas Section 5.4 summarizes the results of the human experts' assessment of the generated answers, with particular attention paid to the comparison between open-source and proprietary LLMs.

5.1. RAG Retrieval Performance

5.1.1. Results for the Automotive Cybersecurity Benchmark

Figure 3 shows the effect of varying the number K of retrieved chunks on the retrieval performance using syntactic and semantic searches. The best performance, in terms of average F1-Scores@K (0.58 for semantic retriever and 0.35 for syntactic retriever), was achieved by retrieving K = 3 chunks. Retrieving a large number of chunks yielded worse performance, mainly due to a decrease in retrieval precision, which was not compensated

by the recall increase. Semantic search obtained consistently better performance than syntactic similarity as it also captured more subtle textual dependencies.

We also compared the MRR scores achieved by the syntactic and semantic retrievers. The average results (0.79 semantic retriever, 0.28 syntactic retriever) confirm the higher ability of semantic search to understand the underlying meaning and context of the input queries, allowing it to more accurately match relevant documents even when syntactically matching keywords are not present.

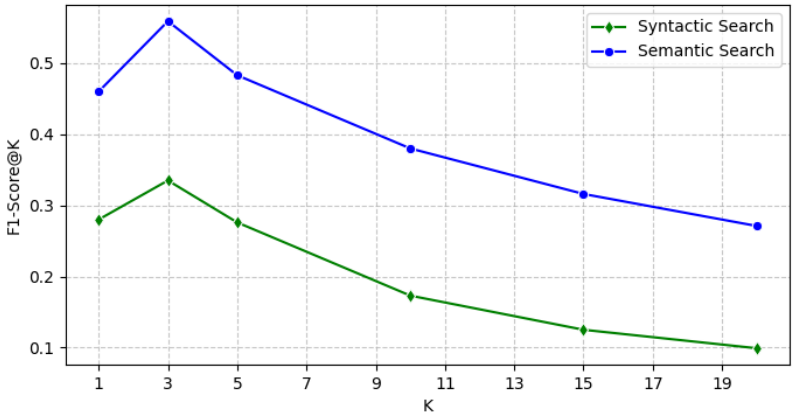


Figure 3. RAG retrieval. Effect of *K* on the F1-Score performance averaged over all LLMs and prompts.

5.1.2. Results for the Public Cybersecurity Dataset

We also evaluated the RAG retrieval performance on the question-answering benchmark released by [10]. Unlike the automotive testing benchmark, it focused on generic cyber-attack attribution and investigation rather than on automotive cybersecurity. To enable RAG retrieval, we added the documents recommended by the paper authors to the GKB. During the insertion process, each document was divided into chunks of 1000 characters, with an overlap of 200 characters. Since the code of the RAG presented by [10] was not publicly available, we compared our performance scores with those reported in the referenced paper. Figure 4 shows the Precision@K, Recall@K, and Context Entity Recall@K median scores and confidence intervals over all question–answer pairs and combinations of LLM and retrieval strategy. The corresponding median scores achieved by [10] are indicated by the red lines. For all the considered metrics, our approach outperformed the state-of-the-art method.

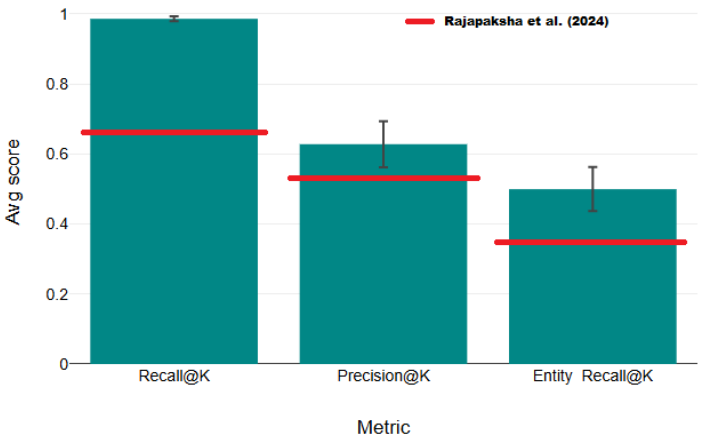


Figure 4. Average retrieval performance scores (and the corresponding confidence intervals) of semantic RAG compared with those reported in [10] (indicated by the red lines). Public cybersecurity benchmark.

5.2. Quantitative Evaluation of the Generated Answers

Table 3 compares the four retrieval and aggregation strategies regarding the quality of the generated answers. Semantic RAG performed the best, surpassing the ensemble and two-stage approaches as well. Mixing syntactic and semantic approaches was not beneficial in most cases, likely due to the relative redundancy of the mixed outputs.

Table 3. Effect of the RAG retrieval strategy on the quality of the generated answers. Performance scores were averaged over all LLMs and prompt settings. Automotive cybersecurity dataset. The best performance score is written in boldface.

RAG Variant	Rouge-1 F1-Score	Rouge-2 F1-Score	Rouge-L F1-Score	BERTScore F1-Score
Syntactic RAG	0.292 ± 0.033	0.060 ± 0.014	0.178 ± 0.021	0.098 ± 0.059
Semantic RAG	**0.364 ± 0.045**	**0.124 ± 0.020**	**0.229 ± 0.021**	**0.208 ± 0.059**
Ensemble RAG	0.341 ± 0.040	0.114 ± 0.019	0.214 ± 0.022	0.161 ± 0.066
Two-Stage RAG	0.310 ± 0.025	0.077 ± 0.013	0.184 ± 0.018	0.158 ± 0.038

5.3. LLM Performance Comparison

We evaluated the LLMs' ability to generate responses similar to those produced by human experts. Table 4 compares the average similarity scores, expressed in terms of ROUGE, BERTScore, and Faithfulness metrics, achieved by both opensource and proprietary LLMs, in the semantic RAG scenario on both the automotive cybersecurity dataset and the public general-purpose datasets. For the sake of readability, the full set of results achieved by every combination of LLM and RAG strategy is given in the Appendix A.

The proprietary GPT-4o LLM averagely performed the best, but the performance was on par with (or slightly lower than) the opensource Llama-3.1 model, which was substantially smaller (8 billion of parameters vs. $>10^2$B of GPT-4o). This indicates that, in general, RAG generative capabilities do not substantially degrade while adopting open-source, cost-effective models.

The average faithfulness score achieved by our approach on the public dataset considering all models (0.706 ± 0.387) was significantly higher than those obtained by the state-of-the-art model [10] (0.625).

Table 4. Average performance score (±standard deviation) of semantic RAG on both the considered datasets. The best performing LLM performance is written in boldface.

Dataset	LLM	Rouge-1 F1-Score	Rouge-2 F1-Score	Rouge-L F1-Score	BERTScore F1-Score	Faithfulness Score
Automotive cybersecurity	*Llama-3.1-8B-It*	**0.392 ± 0.107**	**0.139 ± 0.090**	**0.248 ± 0.077**	**0.251 ± 0.127**	**0.445 ± 0.235**
	gemma-2-9B-It	0.379 ± 0.142	0.122 ± 0.118	0.231 ± 0.088	**0.245 ± 0.164**	0.390 ± 0.308
	Mistral-7B-It-v0.3	0.388 ± 0.123	0.132 ± 0.109	0.236 ± 0.066	0.209 ± 0.168	0.385 ± 0.236
	zephyr-7B-beta	0.274 ± 0.063	0.087 ± 0.054	0.175 ± 0.045	0.082 ± 0.107	0.387 ± 0.316
	Mean (opensource)	0.358 ± 0.108	0.120 ± 0.093	0.223 ± 0.069	0.197 ± 0.141	0.402 ± 0.274
	GPT-4o (proprietary)	**0.385 ± 0.133**	**0.140 ± 0.125**	**0.253 ± 0.086**	**0.254 ± 0.149**	**0.602 ± 0.212**
General-purpose cybersecurity	*Llama-3.1-8B-It*	**0.268 ± 0.158**	**0.137 ± 0.154**	**0.229 ± 0.163**	**0.198 ± 0.206**	**0.798 ± 0.364**
	gemma-2-9B-It	0.257 ± 0.207	0.126 ± 0.160	0.211 ± 0.187	0.067 ± 0.333	0.587 ± 0.444
	Mistral-7B-It-v0.3	0.281 ± 0.151	0.140 ± 0.138	0.244 ± 0.149	0.238 ± 0.180	0.721 ± 0.394
	zephyr-7B-beta	0.170 ± 0.111	0.072 ± 0.078	0.139 ± 0.099	0.111 ± 0.166	0.604 ± 0.399
	Mean (opensource)	0.268 ± 0.156	0.135 ± 0.132	0.228 ± 0.149	0.187 ± 0.221	0.677 ± 0.400
	GPT-4o (proprietary)	**0.362 ± 0.200**	**0.201 ± 0.200**	**0.316 ± 0.202**	**0.323 ± 0.195**	**0.818 ± 0.332**

Inference Cost Analysis

To further assess the cost-effectiveness of the proposed RAG variants, we estimated the inference cost required to generate the answers of the automotive cybersecurity benchmark. For the proprietary GPT-4o model, the monetary cost was computed from the number of processed input and output tokens. We used the GPT-4o API pricing available at the time of the analysis, namely, $p_{\text{in}} = \$2.50$ per million input tokens and $p_{\text{out}} = \$10.00$ per million output tokens. The API cost was computed as follows:

$$C_{\text{API}} = \frac{T_{\text{in}}}{10^6} \cdot p_{\text{in}} + \frac{T_{\text{out}}}{10^6} \cdot p_{\text{out}}, \tag{7}$$

where T_{in} and T_{out} denote the estimated number of input and output tokens, respectively, while p_{in} and p_{out} denote the corresponding prices per million tokens.

For open-source models executed locally, no provider-side per-token fee applied. Their monetary cost depended on the deployment scenario, including hardware ownership, utilization rate, energy price, cooling overhead, and maintenance. For this reason, rather than assigning a universal per-token price to local models, we report a marginal energy cost estimate. This estimate represents a lower bound on the cost of running inference on already available in-house hardware. Specifically, the local energy-only inference cost was computed as

$$C_{\text{local}}^{\text{energy}} = \frac{T_{\text{proc}}}{r \cdot 3600} \cdot c_{\text{GPU-hour}}^{\text{energy}}, \tag{8}$$

where T_{proc} is the total number of processed tokens, r is the assumed inference throughput in processed tokens per second, and $c_{\text{GPU-hour}}^{\text{energy}}$ is the hourly energy cost of operating the GPU. In the energy-only scenario, $c_{\text{GPU-hour}}^{\text{energy}}$ was estimated from the GPU power consumption and the electricity price:

$$c_{\text{GPU-hour}}^{\text{energy}} = P_{\text{GPU}} \cdot c_{\text{kWh}}, \tag{9}$$

where P_{GPU} is the GPU power consumption in kW and c_{kWh} is the electricity price per kWh. For the local open-source setting, we used the assumptions reported in Table 5. We set $P_{\text{GPU}} = 0.25$ kW, corresponding to the nominal 250 W power envelope of the Tesla V100 GPU used in our experiments. We set $c_{\text{kWh}} = \$0.30/\text{kWh}$ as a conservative electricity price scenario. Finally, we assumed $r = 20$ processed tokens/s as a conservative average throughput for the considered 7B–9B open-source models on a Tesla V100. Since actual throughput depended on implementation details, batching, quantization, prompt length, and decoding parameters, the resulting values should be interpreted as indicative lower-bound energy costs rather than full ownership costs.

Table 5. Assumptions used for the marginal energy cost estimate of local open-source inference.

Variable	Value	Interpretation
P_{GPU}	0.25 kW	Nominal Tesla V100 power envelope
c_{kWh}	$USD0.30/\text{kWh}$	Conservative electricity price scenario
r	20 tokens/s	Assumed average processed-token throughput
$c_{\text{GPU-hour}}^{\text{energy}}$	$USD0.075/\text{h}$	Marginal GPU energy cost

Table 6 reports the estimated inference costs. Semantic RAG achieved the lowest GPT-4o cost, equal to $USD0.159$ over the whole benchmark. Syntactic RAG and ensemble RAG had comparable costs, whereas two-stage RAG was substantially more expensive because it required three LLM calls per question. The same trend was observed for the local open-source setting, where two-stage RAG required a higher number of processed tokens and, therefore, a higher marginal energy cost. These results indicate that semantic RAG

offers the most favorable cost-effectiveness profile among the evaluated configurations: it achieves the best answer generation performance while requiring the lowest estimated GPT-4o API cost and a limited marginal energy cost under local open-source deployment.

Table 6. Estimated inference cost over the whole benchmark. GPT-4o costs used the API pricing available at the time of the analysis; local open-source values were averaged over the four locally executed models.

RAG	Estimated Input Tok.	GPT-4o Output Tok.	GPT-4o Cost USD	Local Output Tok. Avg.	Local Energy Cost Avg. USD
Semantic	45,843	4463	0.159	6158	0.054
Syntactic	45,843	4875	0.163	6764	0.055
Ensemble	45,843	5104	0.166	6718	0.055
Two-Stage	107,467	15,312	0.422	20,307	0.137

5.4. Human Expert Validation

We involved PhD-level experts in cybersecurity to validate the quality of the RAG responses. We asked the involved experts to submit an evaluation form based on the answers generated from a sample of input questions. Based on the results reported in Sections 5.1 and 5.2, we focused the human evaluation on the answers produced by the semantic RAG, including both proprietary and open-source LLMs.

For each generated answer, human experts evaluated the following aspects:

1. Appropriateness for the automotive cybersecurity context (from 1 to 5, the higher the better);
2. Relevance to plan security assessment (from 1 to 5, the higher the better);
3. Relevance to actually perform security assessment (from 1 to 5, the higher the better);
4. Completeness (from 1 to 5, the higher the better);
5. Presence of hallucinations (from 1 to 5, the lower the better).

Figure 5 summarizes the results of the human evaluation, paying particular attention to the comparison between proprietary and open-source LLMs. In general, both families of LLMs received significant positive feedback across all metrics, i.e., average scores above 3 for questions (i)–(iv) and below 0.5 for questions (v). Open-source LLMs showed competitive performance against the proprietary GPT-4o model across all evaluated aspects. GPT-4o performed slightly better than open-source ones in terms of the level of suppression of the hallucination effects, as open-source LLMs sometimes produced results that deviated from factual correctness.

It is worth noticing that the human evaluators, who were unaware of the type of LLM used to generate the response they were evaluating, detected subtle differences in LLM performance that were not captured by the classical quantitative metrics, such as the fluency of the response and the pertinence to the common knowledge within the domain of automotive cybersecurity. Notably, in terms of pertinence, relevance, and completeness, the open-source LLMs' answers were in line with those provided by the larger, proprietary model. We deem the aforesaid nuances captured by the human evaluation very important to assess the usability of open-source LLMs in real-world applications. Overall, the reported findings confirm their usability in the automotive cybersecurity field.

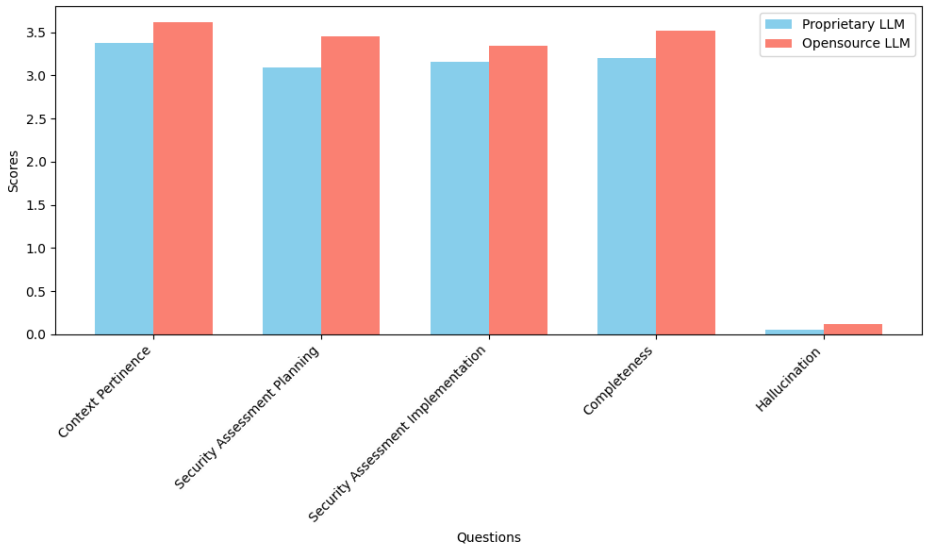


Figure 5. Human evaluation of the generated answers. Comparison between proprietary and open-source LLMs.

6. Examples of Questions and Answers

In Table 7, we report representative examples of questions and RAG answers. Since the main focus of the present work is on automotive cybersecurity, here, we comment on an example selected from our testing benchmarking. For the sake of completeness, additional question–answer examples generated from the public cybersecurity benchmark [10] are given in the Appendix B.

Table 7. Examples of answers to the following question extracted from our automotive cybersecurity dataset: *Tell how to perform a test to attempt to brute force the security access authentication on UDS*.

RAG	LLM: Answer
Syntactic RAG	**Llama-3.1-8B-It (opensource):** To perform a test to attempt to brute force the security access authentication on UDS (Unified Diagnostic Services), I would start by sending a large number of authentication requests with different credentials to the UDS server, using a tool like `uds-client` or a custom script. This would involve sending a series of `Authenticate` requests with varying combinations of username, password, and other authentication parameters. The goal is to exhaustively test all possible combinations of credentials to identify any potential vulnerabilities in the authentication mechanism.
	GPT-4o (proprietary): To perform a test to attempt to brute force the security access authentication on UDS (Unified Diagnostic Services), you would need to send a series of diagnostic requests with incrementing or varying security keys to the ECU (Electronic Control Unit) until the correct key is found. This involves capturing the seed value provided by the ECU, generating potential keys using a known algorithm or brute force method, and sending these keys back to the ECU to gain access.
Semantic RAG	**Llama-3.1-8B-It (opensource):** To perform a test to attempt to brute force the security access authentication on UDS, first, identify the pinouts related to the target network and connect them to a hardware adapter or converter. In case of an external adapter or converter, connect it to the test PC. Then, use weak cryptographic algorithms, such as all-zero keys or XOR, to simulate a bruteforce attack against the UDS security access authentication.
	GPT-4o (proprietary): To perform a test to attempt to brute force the security access authentication on UDS, first, identify the pinouts related to the target network and connect the pins to a hardware adapter or converter. Then, connect the hardware adapter or converter to the test PC. Use weak cryptographic algorithms, such as all-zero key or XOR, to systematically guess the security access authentication, ensuring that the authentication is not cracked before the stopping condition is reached.
Ensemble RAG	**Llama-3.1-8B-Instruct (opensource):** To perform a test to attempt to brute force the security access authentication on UDS, follow these steps: 1. Identify the pinouts related to the target network and connect them to a hardware adapter or converter. 2. In case of an external hardware adapter or converter, connect it to the test PC. 3. Use a list of common weak cryptographic algorithms (e.g., all-zero key, XOR) to simulate a brute force attack on the UDS security access authentication. 4. Monitor the UDS system for any signs of authentication failures, account lockouts, or other indicators of a successful brute-force attack. 5. Verify if the UDS system enforces a time delay when one or more authentication attempts have failed and if the time delay is kept even if the UDS is reset or switched to a different diagnostic session.
	GPT-4o (proprietary): To perform a test to attempt to brute force the security access authentication on UDS, first identify the pinouts related to the target network and connect them to a hardware adapter or converter, then connect this setup to the test PC. Use known weak cryptographic algorithms, such as all-zero keys or XOR, to attempt to brute force the security access authentication. The objective is to verify if the ECU uses weak cryptographic algorithms for diagnostic access authentication and ensure that the security access authentication is not cracked before the stopping condition is reached.

Table 7. *Cont.*

RAG	LLM: Answer
Two-Stage RAG	**Llama-3.1-8B-It (opensource):** To perform a test to attempt to brute force the security access authentication on UDS, first identify the authentication protocol used by the UDS system, which is typically based on the CAN bus. Then, use a tool like CANalyze or a CAN bus sniffer to capture the authentication packets exchanged between the UDS client and server, and analyze the authentication protocol to identify potential vulnerabilities. Finally, use a tool like Burp Suite or Hydra to simulate a bruteforce attack on the UDS authentication system, sending a large number of authentication requests with different credentials to the UDS server.
	GPT-4o (proprietary): To perform a test to attempt to brute force the security access authentication on UDS (Unified Diagnostic Services), first identify the pinouts related to the target network and connect them to a hardware adapter or converter, which is then connected to the test PC. Capture the seed value provided by the ECU (Electronic Control Unit) and systematically generate potential keys using weak cryptographic algorithms or brute force methods, sending these keys back to the ECU until the correct key is found. The objective is to determine if the ECU uses weak cryptographic algorithms for diagnostic access authentication.

To emphasize similarities and differences between open-source and proprietary LLM outcomes, here, we report for every RAG variant the answers generated by LLama-3.1-8B-It (open-source) and GPT-4o (proprietary).

LLama-3.1-8B-It produced appropriate results, despite its significantly smaller model size than GPT-4o. It created consistent and coherent responses across all RAG configurations, effectively referencing automotive cybersecurity concepts from the given context. Furthermore, it provided concise yet detailed answers, showing its ability to prompt following. The reported example shows that LLama-3.1-8B-It can be deemed a cost-effective solution for domain-specific RAGs in the domain of automotive cybersecurity.

7. Conclusions and Future Developments

This paper explored the use of Retrieval-Augmented Generation frameworks relying on opensource LLMs to support automotive cybersecurity experts in their activities related to vulnerability assessment and penetration testing. To demonstrate the cost-effectiveness of the proposed solution, it explored various RAG settings and compared the performance of opensource and proprietary LLMs. Notably, opensource frameworks integrating open-source LLMs, such as Llama3.1 by Meta Inc., show promising retrieval and generative performance, competitive in some respects with proprietary models.

The main takeaways are

- 10% average improvement of human preferences (see Section 5.4).
- Recall@K above 0.95 vs. 0.65 for state-of-the-art in-domain RAGs (see Section 5.1).
- *Llama-3.1-8B-It* turns out to be best-performing opensource LLM, showing results competitive in some respects with OpenAI GPT-4.0.

7.1. Limitations

Our research has the following main limitations:

1. Due to budget constraints, we limited the experiments to a subset of representative opensource models and RAG frameworks and restricted the tests with proprietary models to the OpenAI GPT family.
2. Following the scope of the the industrial applications explored in the automotive cybersecurity scenario, we reused the knowledge base specifications recommended by the data owners.
3. We developed a curated set of question-answering benchmarks restricted to a limited set of in-house cases relevant to the automotive cybersecurity experts. Hence, we cannot exclude the presence of bias in topic coverage and human-driven content generation.
4. The snapshot of data and questions is inherently static and, thus, ignores the temporal variability of cybersecurity threads.
5. For the sake of simplicity, the retrieval strategies do not incorporate reranking and multihop strategies, which are out of the scope of the current research.

6. The three target use cases (verification, vulnerability assessment, penetration testing) are addressed primarily as question-answering retrieval and generation tasks. Furthermore, our study is restricted to the automotive domain, but can be potentially extended to other scenarios and domains.

7.2. Future Works

Future research activities aim to address the current research limitations:

1. Extend the analysis to additional opensource and proprietary models.
2. Broaden the scope of the testing benchmark data collection by involving more domain experts and covering further domains beyond the automotive context.
3. Study the presence of eventual bias effects and mitigate eventual issues.
4. Explore more advanced retrieval strategies, including Agentic AI-based solutions.
5. Apply Parameter-Efficient Fine-Tuning techniques to fine-tune small LLMs for question-answering tasks.

Author Contributions: Conceptualization: A.G., L.C., P.G. and C.B.; methodology: A.G., L.C., P.G. and C.B.; software: A.G.; validation: A.G., L.C., P.G. and C.B.; formal analysis: A.G., L.C., P.G. and C.B.; writing—original draft preparation: A.G., L.C., P.G. and C.B.; writing—review and editing: A.G., L.C., P.G., C.B. and L.F.; visualization: A.G., L.C., P.G., C.B. and L.F. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Next-GenerationEU (Italian PNRR—M4 C2, Invest 1.3—D.D. 1555.11-10-2022, PE00000013).

Data Availability Statement: The dataset of model-generated answers under different RAG configurations is available for research purposes only, upon request to the authors at https://zenodo.org/records/19727315 (accessed on 29 June 2026).

Acknowledgments: This study was carried out within FAIR (Future Artificial Intelligence Research) and received funding from Next-GenerationEU (Italian PNRR—M4 C2, Invest 1.3—D.D. 1555.11-10-2022, PE00000013). This manuscript reflects only the authors' views and opinions; neither the European Union nor the European Commission can be considered responsible for them.

Conflicts of Interest: Author Luca Ferrua was employed by the company DriveSec. The remaining authors declare that this research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Appendix A. Additional Quantitative Results

This appendix contains more detailed quantitative results on evaluating the quality of the answers produced by different RAG variants and LLMs. The results, shown in Table A1, confirm the fairly good results of the small (8 billion), open-source Llama model compared to the larger, proprietary GPT-4o model.

Table A1. Answer generation. Quantitative performance results achieved by every combination of LLM and retrieval strategy. Scores are averaged over all queries and prompts. Values with their deviation standard are reported. Best scores are highlighted in bold.

RAG Variant	LLM	Rouge-1 F1-Score	ROUGE-2 F1-Score	ROUGE-L F1-Score	BERTScore F1-Score
Syntactic RAG	*Llama-3.1-8B-It*	**0.323** ± 0.098	**0.079** ± 0.046	0.200 ± 0.056	0.155 ± 0.101
	gemma-2-9B-It	0.308 ± 0.100	0.046 ± 0.050	0.182 ± 0.068	**0.164** ± 0.140
	Mistral-7B-It-v0.3	0.263 ± 0.071	0.052 ± 0.038	0.157 ± 0.045	0.048 ± 0.087
	zephyr-7B-beta	0.243 ± 0.057	0.049 ± 0.026	0.152 ± 0.037	0.030 ± 0.119
	Mean (opensource)	0.284 ± 0.081	0.057 ± 0.160	0.173 ± 0.051	0.099 ± 0.112
	GPT-4o (proprietary)	**0.323** ± 0.099	0.074 ± 0.050	**0.202** ± 0.061	0.158 ± 0.100

Table A1. *Cont.*

RAG Variant	LLM	Rouge-1 F1-Score	ROUGE-2 F1-Score	ROUGE-L F1-Score	BERTScore F1-Score
Semantic RAG	*Llama-3.1-8B-It*	**0.392** ± 0.107	0.139 ± 0.090	0.248 ± 0.077	0.251 ± 0.127
	gemma-2-9B-It	0.379 ± 0.142	0.122 ± 0.118	0.231 ± 0.088	0.245 ± 0.164
	Mistral-7B-It-v0.3	0.388 ± 0.123	0.132 ± 0.109	0.236 ± 0.066	0.209 ± 0.168
	zephyr-7B-beta	0.274 ± 0.063	0.087 ± 0.054	0.175 ± 0.045	0.082 ± 0.107
	Mean (opensource)	0.358 ± 0.108	0.120 ± 0.093	0.223 ± 0.069	0.197 ± 0.141
	GPT-4o (proprietary)	0.385 ± 0.133	**0.140** ± 0.125	**0.253** ± 0.086	**0.254** ± 0.149
Ensemble RAG	*Llama-3.1-8B-It*	0.367 ± 0.116	**0.139** ± 0.112	0.227 ± 0.074	0.225 ± 0.149
	gemma-2-9B-It	0.335 ± 0.201	0.110 ± 0.110	0.201 ± 0.110	0.124 ± 0.293
	Mistral-7B-It-v0.3	0.346 ± 0.081	0.102 ± 0.064	0.216 ± 0.062	0.157 ± 0.114
	zephyr-7B-beta	0.269 ± 0.066	0.087 ± 0.057	0.180 ± 0.049	0.061 ± 0.087
	Mean (opensource)	0.329 ± 0.116	0.110 ± 0.085	0.206 ± 0.073	0.142 ± 0.161
	GPT-4o (proprietary)	**0.387** ± 0.111	0.132 ± 0.107	**0.244** ± 0.080	**0.239** ± 0.133
Two-Stage RAG	*Llama-3.1-8B-It*	0.315 ± 0.078	0.090 ± 0.049	0.190 ± 0.039	0.172 ± 0.084
	gemma-2-9B-It	0.319 ± 0.093	0.068 ±0.061	0.188 ± 0.049	0.190 ± 0.089
	Mistral-7B-It-v0.3	0.303 ± 0.053	0.075 ± 0.051	0.180 ± 0.044	0.130 ± 0.090
	zephyr-7B-beta	0.268 ± 0.065	0.058 ± 0.045	0.152 ± 0.036	0.100 ± 0.079
	Mean (opensource)	0.301 ± 0.072	0.073 ± 0.051	0.178 ± 0.042	0.148 ± 0.085
	GPT-4o (proprietary)	**0.344** ± 0.088	**0.093** ± 0.070	**0.208** ± 0.055	**0.199** ± 0.103

Appendix B. Sample Answers from the Semantic RAG Process

This Appendix reports a selection of question–answer pairs from the public cybersecurity benchmark. The results, shown in Table A2, confirm the main findings in the tests relative to the automotive field.

Table A2. Examples of answers from the investigated LLMs obtained during the semantic RAG process on a question from the [10] dataset.

Question	Answers
What is the relationship between MagicRAT and TigerRAT?	**Llama-3.1-8B-It Answer:** The relationship between MagicRAT and TigerRAT is that they are both malware families attributed to the Lazarus APT group. Specifically, MagicRAT is a remote access trojan (RAT) that hosts and serves TigerRAT on its C2 infrastructure. In some cases, MagicRAT is deployed on infected systems, followed by the removal of MagicRAT and the subsequent download and execution of TigerRAT, another custom-developed malware attributed to Lazarus.
	gemma-2-9B-It Answer: MagicRAT and TigerRAT are both remote access trojans (RATs) attributed to the Lazarus APT group. Cisco Talos has discovered that Lazarus uses MagicRAT's C2 infrastructure to host and serve TigerRAT to existing MagicRAT infections. This suggests a possible evolution in Lazarus' tactics, where they utilize MagicRAT as a stepping stone to deploy more sophisticated malware like TigerRAT.
	GPT-4o Answer: MagicRAT and TigerRAT are both remote access trojans (RATs) attributed to the Lazarus APT group, a North Korean state-sponsored threat actor. MagicRAT's C2 infrastructure has been used to host and serve TigerRAT, indicating a direct operational link between the two. MagicRAT is deployed on infected systems and can launch additional payloads, including TigerRAT, which suggests that MagicRAT acts as a facilitator for deploying other malware like TigerRAT. Both RATs share infrastructure and are part of the broader malware toolkit used by Lazarus to target organizations worldwide.
	Mistral-7B-It-v0.3 Answer: MagicRAT and TigerRAT are related malware families attributed to the Lazarus Advanced Persistent Threat (APT) group. MagicRAT is a remote access trojan (RAT) that has been deployed by Lazarus, and it has been observed to serve as a platform for hosting and executing TigerRAT. TigerRAT is another RAT that has been disclosed and attributed to Lazarus by the Korean Internet & Security Agency (KISA). Both malware families are used by Lazarus to target organizations worldwide.
	zephyr-7B-beta Answer: Cisco Talos has discovered that the North Korean state-sponsored actor Lazarus has deployed a new remote access trojan (RAT) called MagicRAT, which they use after successfully exploiting vulnerabilities in VMWare Horizon platforms. Talos has also found links between MagicRAT and another RAT called TigerRAT, which has been disclosed and attributed to Lazarus by the Korean Internet & Security Agency (KISA). TigerRAT has evolved over the past year to include new functionalities, and Talos has observed C2 servers hosting and serving both MagicRAT and TigerRAT to existing infections. This suggests that Lazarus is rapidly building new, bespoke malware to use alongside their previously known malware such as TigerRAT to target organizations worldwide.

References

1. Kifor, C.V.; Popescu, A. Automotive Cybersecurity: A Survey on Frameworks, Standards, and Testing and Monitoring Technologies. *Sensors* **2024**, *24*, 6139. [CrossRef]
2. Cybersecurity, C.I. *Framework for Improving Critical Infrastructure Cybersecurity*; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2018. [CrossRef]
3. Rajapaksha, S.; Kalutarage, H.; Al-Kadri, M.O.; Petrovski, A.; Madzudzo, G.; Cheah, M. AI-Based Intrusion Detection Systems for In-Vehicle Networks: A Survey. *ACM Comput. Surv.* **2023**, *55*, 237. [CrossRef]
4. Sommer, F.; Kriesten, R.; Kargl, F. Survey of Model-Based Security Testing Approaches in the Automotive Domain. *IEEE Access* **2023**, *11*, 55474–55514. [CrossRef]
5. Zhang, J.; Bu, H.; Wen, H.; Liu, Y.; Fei, H.; Xi, R.; Li, L.; Yang, Y.; Zhu, H.; Meng, D. When LLMs meet cybersecurity: A systematic literature review. *Cybersecurity* **2025**, *8*, 55. [CrossRef]
6. Kereopa-Yorke, B. Building resilient SMEs: Harnessing large language models for cyber security in Australia. *J. AI Robot. Workplace Autom.* **2024**, *3*, 15–27.
7. Zhou, X.; Cao, S.; Sun, X.; Lo, D. Large Language Model for Vulnerability Detection and Repair: Literature Review and the Road Ahead. *ACM Trans. Softw. Eng. Methodol.* **2024**, *34*, 145. [CrossRef]
8. Chen, J.; Lin, H.; Han, X.; Sun, L. Benchmarking Large Language Models in Retrieval-Augmented Generation. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence (AAAI'24/IAAI'24/EAAI'24)*; AAAI Press: Washington, DC, USA, 2024; Volume 38, pp. 17754–17762. [CrossRef]
9. Su, W.; Tang, Y.; Ai, Q.; Yan, J.; Wang, C.; Wang, H.; Ye, Z.; Zhou, Y.; Liu, Y. Parametric Retrieval Augmented Generation. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval, New York, NY, USA, 2025, SIGIR '25*; Association for Computing Machinery: New York, NY, USA, 2025; pp. 1240–1250. [CrossRef]
10. Rajapaksha, S.; Rani, R.; Karafili, E. A RAG-Based Question-Answering Solution for Cyber-Attack Investigation and Attribution. *arXiv* **2024**, arXiv:2408.06272. [CrossRef]
11. Motlagh, F.N.; Hajizadeh, M.; Majd, M.; Najafi, P.; Cheng, F.; Meinel, C. Large Language Models in Cybersecurity: State-of-the-Art. *arXiv* **2024**, arXiv:2402.00891. [CrossRef]
12. Marksteiner, S.F.; Schmittner, C.; Christl, K.; Nickovic, D.; Sjödin, M.; Sirjani, M. From TARA to Test: Automated Automotive Cybersecurity Test Generation Out of Threat Modeling. In *Proceedings of the 7th ACM Computer Science in Cars Symposium, New York, NY, USA, 2023, CSCS '23*; Association for Computing Machinery: New York, NY, USA, 2023. [CrossRef]
13. Tihanyi, N.; Ferrag, M.A.; Jain, R.; Bisztray, T.; Debbah, M. CyberMetric: A Benchmark Dataset based on Retrieval-Augmented Generation for Evaluating LLMs in Cybersecurity Knowledge. *arXiv* **2024**, arXiv:2402.07688. [CrossRef]
14. Fayyazi, R.; Taghdimi, R.; Yang, S.J. Advancing TTP Analysis: Harnessing the Power of Large Language Models with Retrieval Augmented Generation. *arXiv* **2024**, arXiv:2401.00280. [CrossRef]
15. Fan, W.; Ding, Y.; Ning, L.; Wang, S.; Li, H.; Yin, D.; Chua, T.; Li, Q. A Survey on RAG Meeting LLMs: Towards Retrieval-Augmented Large Language Models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2024, Barcelona, Spain, 25–29 August 2024*; Baeza-Yates, R., Bonchi, F., Eds.; ACM: New York, NY, USA, 2024; pp. 6491–6501. [CrossRef]
16. Es, S.; James, J.; Espinosa-Anke, L.; Schockaert, S. RAGAS: Automated Evaluation of Retrieval Augmented Generation. *arXiv* **2023**, arXiv:2309.15217. [CrossRef]
17. Akhoundali, J.; Nouri, S.R.; Rietveld, K.; Gadyatskaya, O. MoreFixes: A Large-Scale Dataset of CVE Fix Commits Mined through Enhanced Repository Discovery. In *Proceedings of the 20th International Conference on Predictive Models and Data Analytics in Software Engineering, New York, NY, USA, 2024, PROMISE 2024*; Association for Computing Machinery: New York, NY, USA, 2024; pp. 42–51. [CrossRef]
18. Turtiainen, H.; Costin, A. VulnBERTa: On Automating CWE Weakness Assignment and Improving the Quality of Cybersecurity CVE Vulnerabilities Through ML/NLP. In *Proceedings of the 2024 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*; IEEE: Piscataway, NJ, USA, 2024; pp. 618–625. [CrossRef]
19. Al-Sada, B.; Sadighian, A.; Oligeri, G. MITRE ATT&CK: State of the Art and Way Forward. *ACM Comput. Surv.* **2025**, *57*, 1–37. [CrossRef]
20. Elder, S.; Rahman, M.R.; Fringer, G.; Kapoor, K.; Williams, L. A Survey on Software Vulnerability Exploitability Assessment. *ACM Comput. Surv.* **2024**, *56*, 205. [CrossRef]
21. Nicol, D.M.; Shannon, G.; Akbar, M.; Bishop, M.; Chaney, M.M.; Luallen, M.E. Toward Common Weakness Enumerations in Industrial Control Systems. *IEEE Secur. Priv.* **2023**, *21*, 84–93. [CrossRef]
22. Bari, B.S.; Yelamarthi, K.; Ghafoor, S. Intrusion Detection in Vehicle Controller Area Network (CAN) Bus Using Machine Learning: A Comparative Performance Study. *Sensors* **2023**, *23*, 3610. [CrossRef] [PubMed]

23. McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; y Arcas, B.A. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, Fort Lauderdale, FL, USA, 20–22 April 2017, Proceedings of Machine Learning Research*; Singh, A., Zhu, X.J., Eds.; PMLR: London, UK, 2017; Volume 54, pp. 1273–1282. Available online: http://proceedings.mlr.press/v54/mcmahan17a.html (accessed on 29 June 2026).
24. Zang, M.; Yan, Y. Machine Learning-Based Intrusion Detection System for Big Data Analytics in VANET. In *Proceedings of the 93rd IEEE Vehicular Technology Conference, VTC Spring 2021, Helsinki, Finland, 25–28 April 2021*; IEEE: Piscataway, NJ, USA, 2021; pp. 1–5. [CrossRef]
25. Zhao, C.; De Maria, R.; Kumarage, T.; Chaudhary, K.S.; Agrawal, G.; Li, Y.; Park, J.; Chen, Y.C.; Deng, Y.; Liu, H. CyberBOT: Ontology-Grounded Retrieval Augmented Generation for Reliable Cybersecurity Education. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management, CIKM '25*; ACM: New York, NY, USA, 2025; pp. 6752–6756. [CrossRef]
26. Simoni, M.; Saracino, A.; P, V.; Conti, M. MoRSE: Bridging the Gap in Cybersecurity Expertise with Retrieval Augmented Generation. In *Proceedings of the 40th ACM/SIGAPP Symposium on Applied Computing, New York, NY, USA, 2025, SAC '25*; Association for Computing Machinery: New York, NY, USA, 2025; pp. 1213–1222. [CrossRef]
27. Bussari, S.; Punj, P.; Balakumar, G. RAGSec: Retrieval-Augmented Generation for Cybersecurity Threat Intelligence in Enterprise Networks. In *Proceedings of the 2026 IEEE 5th International Conference on AI in Cybersecurity (ICAIC)*; IEEE: Piscataway, NJ, USA, 2026; pp. 1–6. [CrossRef]
28. Lin, J.; Zhao, Y.; Huang, W.; Liu, C.; Pu, H. Domain knowledge graph-based research progress of knowledge representation. *Neural Comput. Appl.* **2021**, *33*, 681–690. [CrossRef]
29. Robertson, S.; Zaragoza, H. The Probabilistic Relevance Framework: BM25 and Beyond. *Found. Trends Inf. Retr.* **2009**, *3*, 333–389. [CrossRef]
30. Reimers, N.; Gurevych, I. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Hong Kong, China*; Inui, K., Jiang, J., Ng, V., Wan, X., Eds.; Association for Computational Linguistics: Stroudsburg, PA, USA, 2019; pp. 3982–3992. [CrossRef]
31. La Quatra, M.; Cagliero, L. Transformer-based highlights extraction from scientific papers. *Knowl.-Based Syst.* **2022**, *252*, 109382. [CrossRef]
32. OpenAI. GPT-4 Technical Report. *arXiv* **2023**, arXiv:2303.08774. [CrossRef]
33. Grattafiori, A.; Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; Al-Dahle, A.; Letman, A.; Mathur, A.; Schelten, A.; Vaughan, A.; et al. The Llama 3 Herd of Models. *arXiv* **2024**, arXiv:2407.21783. [CrossRef]
34. Gemma Team. Gemma 2: Improving Open Language Models at a Practical Size. *arXiv* **2024**, arXiv:2408.00118. [CrossRef]
35. Jiang, A.Q.; Sablayrolles, A.; Mensch, A.; Bamford, C.; Chaplot, D.S.; Casas, D.d.; Bressand, F.; Lengyel, G.; Lample, G.; Saulnier, L.; et al. Mistral 7B. *arXiv* **2023**, arXiv:2310.06825. [CrossRef]
36. Tunstall, L.; Beeching, E.; Lambert, N.; Rajani, N.; Rasul, K.; Belkada, Y.; Huang, S.; von Werra, L.; Fourrier, C.; Habib, N.; et al. Zephyr: Direct Distillation of LM Alignment. *arXiv* **2023**, arXiv:2310.16944. [CrossRef]
37. Manning, C.D.; Raghavan, P.; Schütze, H. *Introduction to Information Retrieval*; Cambridge University Press: Cambridge, UK, 2008. [CrossRef]
38. Lin, C.Y. ROUGE: A Package for Automatic Evaluation of Summaries. In *Proceedings of the Text Summarization Branches Out, Barcelona, Spain*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2004; pp. 74–81. Available online: https://aclanthology.org/W04-1013/ (accessed on 29 June 2026).
39. Zhang, T.; Kishore, V.; Wu, F.; Weinberger, K.Q.; Artzi, Y. BERTScore: Evaluating Text Generation with BERT. In Proceedings of the 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, 26–30 April 2020; OpenReview.net. Available online: https://openreview.net/forum?id=SkeHuCVFDr (accessed on 29 June 2026).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.