A Retrieval-Augmented Generation System for Exploring Intelligent Transportation System Reference Architectures

Afef Awadid *, Mateo Becquart , Maxence Gagnant

Posted Date: 2 October 2025

doi: 10.20944/preprints202510.0186.v1

Keywords: intelligent transportation systems; reference architectures; retrieval-augmented generation system; large language models

 doi:10.20944/preprints202510.0186.v1

Article

A Retrieval-Augmented Generation System for Exploring Intelligent Transportation System Reference Architectures

Afef Awadid *, Mateo Becquart and Maxence Gagnant

IRT SystemX, Palaiseau, France
* Correspondence: afef.awadid@irt-systemx.fr

Abstract

At the core of Intelligent Transportation Systems (ITS) lies a wide range of applications leveraging communication and information technologies to support safer, more efficient, and more sustainable mobility. However, due to their inherent complexity, ITS architectural design remains challenging, particularly for designers unfamiliar with the domain. A promising approach is to build upon existing ITS reference architectures, which are validated frameworks encapsulating essential architectural knowledge. Yet these reference architectures remain underutilized, in part because the architectural elements and reusable components they provide are insufficiently known and disseminated. To address this gap, we propose a Retrieval-Augmented Generation (RAG) system that enables users to explore such architectures through a question–answer interface. The system combines the generative capabilities of large language models (LLMs) with structured knowledge from established ITS reference architectures, including ARC-IT and FRAME, and has been evaluated from both technical and user-oriented perspectives.

Keywords: intelligent transportation systems; reference architectures; retrieval-augmented generation system; large language models

1. Introduction

Intelligent Transportation Systems (ITS), such as autonomous vehicles, apply advanced information and communication technologies to enhance the sustainability, efficiency, and safety of transportation networks [1]. Their adoption is expanding rapidly in response to increasing demands for safer, more efficient, and environmentally sustainable mobility solutions [2]. From a safety perspective, ITS improve road security by reducing the frequency, severity, and cost of accidents, thereby lowering fatalities and enhancing personal safety. From an environmental perspective, they aim to reduce energy consumption and mitigate adverse impacts by minimizing fuel use and emissions due to congestion, while also decreasing noise pollution and limiting traffic disruption in residential areas [3]. These benefits have been demonstrated across diverse transport environments [4].

Nevertheless, the benefits of ITS are accompanied by several key challenges, foremost among them their inherent complexity. This complexity arises, first, from the fact that ITS consist of multiple components and subsystems characterized by dynamic, time-dependent interactions [5]. Second, it stems from the multidisciplinary nature of ITS, which integrate telecommunications, electronics, information technologies, and traffic engineering. As a result, the architectural design of ITS becomes increasingly intricate and time-consuming, particularly when system designers must rely primarily on their own domain knowledge.

 doi:10.20944/preprints202510.0186.v1

To address this challenge, a reliable approach is to build on existing ITS reference architectures. A reference architecture can be defined as a shared and validated system description, developed by domain experts and adopted by a community of interest as a guide for the development and evolution of systems [6]. It constitutes a key outcome of the domain design process, encapsulating established knowledge about how to design system architectures within a specific application domain [7].

However, despite their recognized significance, feedback from joint research and industry projects conducted by our institute—such as TAM (Trusted Autonomous Mobility), SCA (Secure Cooperative Autonomous Systems), and RTI (Resilience of Intelligent Transport)—indicates that ITS reference architectures remain underutilized. A key reason is that the architectural elements and reusable components they encompass are insufficiently known and disseminated.

To address this gap, this paper proposes a Retrieval-Augmented Generation (RAG) system that assists designers in exploring ITS reference architectures in terms of the architectural elements they encompass, their definitions, purposes, and interactions. The key idea is to combine the generative capabilities of large language models (LLMs) with structured knowledge derived from established ITS reference architectures, including ARC-IT [8] and FRAME [9], in order to provide designers with relevant answers through a question–answer interface.

The remainder of this paper is organized as follows. Section 2 introduces the background of this research. Section 3 provides a methodological and technical description of the proposed RAG system pipeline. Section 4 presents the results of the pipeline implementation evaluation, covering both technical and user-oriented perspectives. Finally, Section 5 discusses the findings and concludes the paper.

2. Background

This section first presents the key motivations underlying this research, followed by the fundamental notions on which it is based.

2.1. Motivation

The notion of a reference architecture is well-established and highly valued within the standardization community, where it is defined as "a shared and agreed reference system description used by a community of interest to achieve its business purposes. It is typically generic and instantiated as system architectures specific to individual business purposes" [10]. Despite its recognized importance, feedback from joint research and industry projects conducted by our institute—such as TAM (Trusted Autonomous Mobility), SCA (Secure Cooperative Autonomous Systems), and RTI (Resilience of Intelligent Transport)—indicates that ITS reference architectures are often underutilized, in part because the architectural elements and reusable components they encompass are insufficiently known and disseminated. This observation is further supported by feedback from the systems engineering community within AFIS (French Association for Systems Engineering), which highlights that ITS reference architectures remain largely unknown within the field.

This highlights the need to make ITS reference architectures more accessible to practitioners. Addressing this need provides the primary motivation for this research, which seeks to do so through the development of a Retrieval-Augmented Generation (RAG) system that enables users to explore such architectures via a question–answer interface. The choice of a Retrieval-Augmented Generation approach is justified by its ability to combine the capability of large language models (LLMs) to generate natural-language answers with the validated and domain-specific knowledge provided by ITS reference architectures, thereby ensuring that the answers are both accessible and reliable.

 doi:10.20944/preprints202510.0186.v1

2.2. Key Notions

2.2.1. Intelligent Transportation Systems

As previously stated, intelligent transportation systems (ITS) refer to systems that capitalize on information and communication technologies, such as sensor networks and real-time data, to enhance safety, reduce travel time, alleviate traffic congestion, and thereby mitigate environmental impacts. ITS considers all facets of transportation, covering a variety of system types designed to address specific challenges within each transportation domain.

Intelligent transportation systems are distinguished from traditional transportation systems by the following characteristics [11]:

- Proactive capabilities: ITS employ techniques from artificial intelligence and predictive analytics to implement preemptive measures and anticipate potential issues.
- Data-driven nature: ITS continuously collect and analyze large volumes of data to optimize system performance.
- User-centric orientation: ITS prioritize delivering personalized guidance and real-time information to travelers, thereby enhancing the overall travel experience.
- Interconnected architecture: ITS leverage advanced information and communication technologies to enable seamless data exchange between infrastructure, vehicles, and travelers.

2.2.2. ITS Reference Architectures

Given the complexity of intelligent transportation systems (ITS), the design of their architectures is particularly challenging when system designers must rely solely on their own knowledge of the domain. In this context, domain-specific reference architectures offer a robust solution: they save time in the design process since they provide a reusable, shared, validated, and agreed-upon system description developed by domain experts [6].

In this sense, a reference architecture provides a consistent set of architectural best practices for use by other teams. High-level, generic architectural designs that can be reused, tailored, and instantiated for specific situations or projects play a key role in avoiding repeated solution design efforts [12].

When dealing with ITS reference architectures, several distinct viewpoints are typically considered [13]:

- User Needs: the expectations that an ITS deployment and its associated services are required to meet.
- Functional Viewpoint: the set of functionalities that the ITS must provide to satisfy user needs, usually organized into functional areas and further detailed into specific functions.
- Physical Viewpoint: the organization of functions into physical components and their allocation to modules or subsystems.
- Communications Viewpoint: the communication links necessary to support the exchange of physical data flows.

Two of the most prominent ITS reference architectures are the Architecture Reference for Cooperative and Intelligent Transportation (ARC-IT) [8] and the European ITS Framework Architecture (FRAME) [9].

The European ITS Framework Architecture (FRAME) is technology-independent and supports the development of the ITS services and equipment market. It provides a flexible, high-level framework that individual countries can tailor to their specific requirements and user needs [14]. FRAME is organized around four key viewpoints: User Needs, Functional Viewpoint, Physical Viewpoint, and Communications Viewpoint [13].

The Architecture Reference for Cooperative and Intelligent Transportation (ARC-IT) provides a unified framework for planning, defining, and integrating Intelligent Transportation Systems. It establishes a common language for describing system architectures across diverse transportation domains. Like FRAME, ARC-IT is organized around four key viewpoints: Enterprise, Functional, Physical, and Communications [8]. Although ITS reference architectures may differ in the specifics

 doi:10.20944/preprints202510.0186.v1

of architectural representation [15], they converge on a shared high-level view of ITS architecture, as reflected in these core viewpoints.

2.2.3. Large Language Models and Retrieval-Augmented Generation

Large language models (LLMs) are designed to process and interpret unstructured data—typically natural language—in ways that approximate human cognitive abilities [16,17]. They have been widely adopted across multiple subfields of natural language processing (NLP), particularly for tasks requiring the analysis of large volumes of text. This success is largely attributed to the Transformer architecture, which enables LLMs to capture complex patterns and linguistic structures [18].

This capability enables LLMs to process the large-scale and heterogeneous data streams intrinsic to modern intelligent transportation systems (ITS), thereby enhancing adaptability and responsiveness to the dynamic challenges of transportation environments. Furthermore, the integration of LLMs into ITS represents a significant technological advancement, supporting the development of smarter and more adaptive transportation systems that are better equipped to meet evolving demands [19].

Nonetheless, LLMs are prone to hallucinations—generating outputs that are syntactically correct but factually inaccurate—particularly when sufficient or contextually relevant information is lacking [20]. Such inaccuracies pose serious risks in safety-critical domains, especially transportation and, more specifically, intelligent transportation systems, where system failures may lead to significant harm or even loss of life [21].

One way to mitigate the risks inherent to LLMs is to ground them in established, domain-specific reference architectures, such as those employed in the intelligent transportation domain. This can be achieved through Retrieval-Augmented Generation (RAG), a methodology that integrates the generative capabilities of LLMs with information retrieval mechanisms [22]. RAG enables models to dynamically access and incorporate relevant external knowledge during the generation process [23].

This approach has been recognized as a promising strategy, not only for mitigating hallucinations but also for enhancing domain-specific accuracy and temporal relevance [22,24]. As a result, RAG improves the controllability and interpretability of model outputs, making them more reliable and better aligned with the requirements of the target domain.

3. Retrieval-Augmented Generation System Pipeline

This section provides both a methodological overview (i.e., a general description of the pipeline steps) and a technical perspective (i.e., a detailed account of how each step is implemented in our context) of the proposed Retrieval-Augmented Generation (RAG) system for exploring Intelligent Transportation System (ITS) reference architectures.

As illustrated in Figure 1, the system pipeline comprises three main steps—indexing and embedding, retrieval, and generation—which are highlighted in blue and will be described in detail in the following subsections.

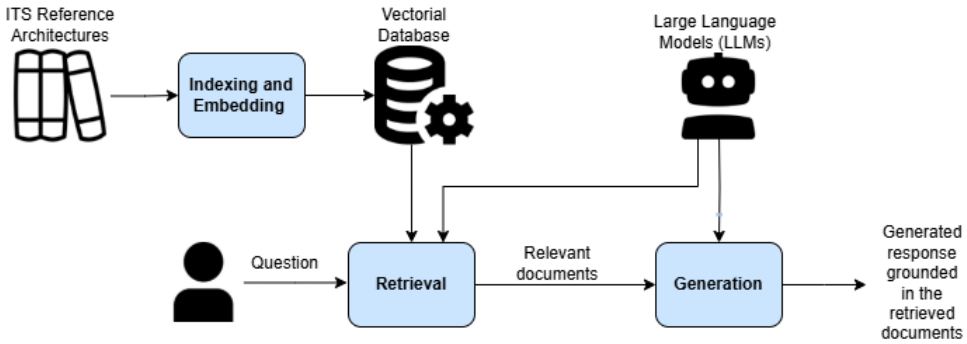

Figure 1. RAG system pipeline.

 doi:10.20944/preprints202510.0186.v1

3.1. Indexing and Embedding

The information required to construct the database of a RAG system may originate from diverse sources and exist in multiple formats (e.g., HTML, PDF, CSV). Consequently, an initial extraction and normalization step is required to convert all content into a uniform plain-text representation. In the case of web-based data, this extraction phase is commonly referred to as web scraping.

To align with the context window of the embedding model, the extracted documents are segmented into smaller chunks. Each chunk is then transformed into a dense numerical representation by the embedding model. Notably, chunking not only mitigates context window limitations but also improves embedding quality: by restricting the number of tokens per input, the model can focus more effectively on the salient content within each chunk [25].

The resulting document embeddings are stored in an index, commonly referred to as a vector store. The vector store must support a similarity search metric, which plays a critical role in the retrieval step, as will be detailed in the following subsection.

In the context of our system, we consider two prominent and widely recognized ITS reference architectures as sources of data: The Architecture Reference for Cooperative and Intelligent Transportation (ARC-IT) [8] and the European ITS Framework Architecture (FRAME) [9], both of which were presented earlier (see Section 2.2.2).

As ARC-IT and FRAME differ in their specific architectural representations and terminology, we have relied on a high-level system architecture framework to facilitate the mapping and convergence of these ITS reference architectures. This high-level framework has been widely used in our joint research and industry projects across various application domains, including automotive, defense, and aeronautics.

As illustrated in Figure 2, the framework considers three main architectural levels: (1) the operational level, which defines the services provided by the system to its environment; (2) the functional level, which specifies the functions required to realize these services; and (3) the organic/physical level, which identifies the resources used to perform these functions.

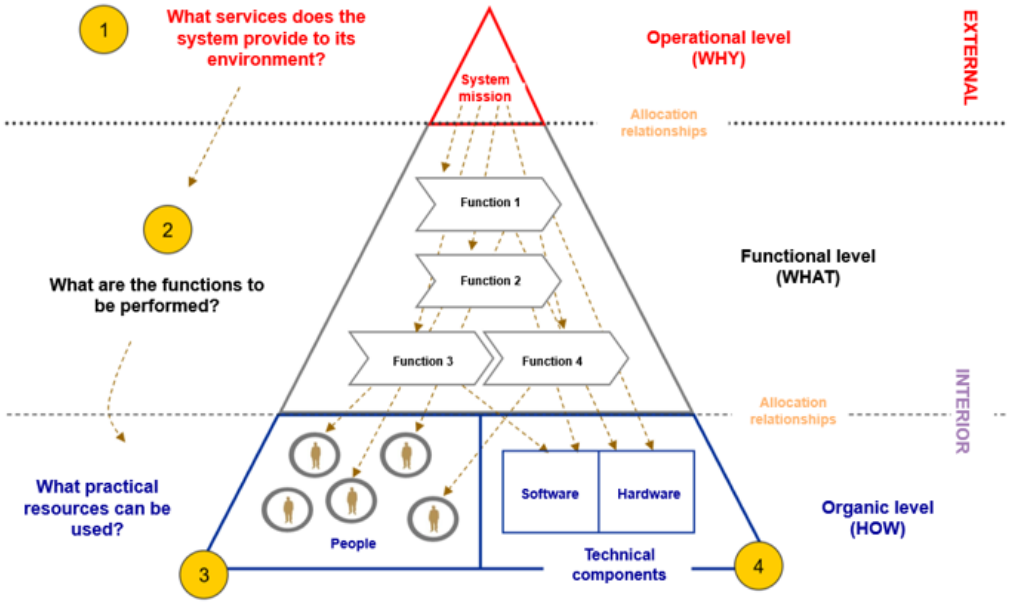

Figure 2. General framework for high-level system architecture description.

Based on this framework, we performed a mapping between the two reference architectures. At the functional level, we compared the functional views of both architectures and incorporated FRAME's low-level functions that addressed aspects not covered by ARC-IT's processes. The index was ultimately constructed using data from both the functional and physical views of these reference architectures.

 doi:10.20944/preprints202510.0186.v1

Having identified the underlying structure of our database, we proceeded to implement an initial indexing approach for constructing the vector database. We employed the Python library BeautifulSoup to scrape the webpages of both reference architectures. A corpus of LangChain Document objects was constructed, each containing information about a specific object, process, or community. We used the gte-multilingual-base model [26] to generate embedding vectors, which offers a favorable trade-off between size (305M parameters) and performance. The resulting vectors have a dimensionality of 768 and are stored in a Chroma index, which supports a wide range of similarity metrics.

Raw data from the reference architectures were first extracted, divided into manageable chunks, and subsequently embedded and stored in an index. Given the size of the database, it would be inefficient for the LLM to search through the entire dataset. Therefore, a retrieval step is implemented to generate a shortlist of query-relevant documents—also referred to as the context—which is then provided as input to the LLM to generate its response. This retrieval step will be described in detail in the following subsection.

3.2. Retrieval

3.2.1. Retrieval – Similarity-Based Methods

In this subsection, we present the first family of retrieval modules we implemented, which are all based on global semantic search. The first module is a simple retriever that operates according to the steps outlined above. However, this straightforward approach proved to be highly sensitive to query formulation: for instance, the set of retrieved documents varied noticeably when two queries expressed the same semantic meaning but were phrased differently.

To address this sensitivity issue, we introduced a pre-retrieval module implemented using the MultiQueryRetriever method from LangChain. This module employs an LLM to generate a set of alternative queries derived from the initial user query, with each query expressing the same topic from a different perspective. By aggregating the responses across all variants, the resulting context becomes more robust to variations in query formulation. For this step, we used the Llama3.1 model (8 billion parameters) to generate the alternative queries. Although this approach alleviated the problem of wording sensitivity, we observed that the retriever still failed to capture relationships between entities. For example, given the query "Provide the list of ITS services within the Data Management domain", the retriever would return either services or documents related to data management, but would not capture the hierarchical relationship between domains and services.

Given that our index is optimized for precise searches (e.g., identifying a single entity), it was necessary to develop a method to capture the inherent dependencies between objects. One potential solution, inspired by graph-based RAG methods [27], involves distinguishing local entities from global ones in the user's query, and then constraining the similarity search to a specific community through a hybrid search method. LangChain provides a related functionality via the SelfQueryRetriever method, which leverages an LLM to generate a structured query and retrieves documents by combining similarity search with metadata-based filters. Although this approach appeared promising, we faced challenges in stabilizing the generation of structured queries, likely due to insufficient granularity in our metadata descriptions. Nevertheless, we retained the principle of filtering our vector database rather than relying solely on similarity search. The following subsection details the filtering strategies we evaluated.

3.2.2. Retrieval – Pre-Filtering Methods

These methods operate by narrowing the scope of retrievable documents through the application of filters. In practice, similarity search is performed only on documents that possess the required attributes. By adopting such pre-filtering, we aim to improve both the recall and the accuracy of the retriever. The main challenge, however, lies in ensuring that the system can reliably extract the appropriate filters from the user's query.

 doi:10.20944/preprints202510.0186.v1

The first step in this workflow is the design of the metadata structure, defined as a set of attributes that can be used to identify either individual entities in the database or collections thereof. As noted earlier, a thorough understanding of the database structure is essential for building such a framework. The attributes used depend on the chosen retrieval method. Importantly, this metadata structure captures the relationships between items in the database—an aspect that earlier approaches were unable to represent.

For each diagram object, we store the base64-encoded diagram within its metadata field. Upon retrieval, the system outputs the corresponding object, and a decoder is employed to reconstruct the original image. This strategy allows us to handle multimodal data representations without requiring a model that natively supports multimodality.

The first filter-based retrieval method we implemented is based on an Extract-and-Associate (EA) module. In this approach, an LLM extracts keywords from the user query and links them to specific metadata fields: type (domain, service, object, or diagram), domain/service/object name, and nature. To guide this process, the LLM is provided with a ResponseSchema and a tailored prompt. The schema specifies the objects to be identified in the query and reflects the metadata structure of the documents. Each document's metadata thus serves as a unique identifier, while the keywords extracted from the query act as the query's identifier. This pre-filtering step ensures that the retrieved context remains concise. However, despite generally accurate keyword extraction, none of the tested models achieved consistent results, as correctly associating keywords with their corresponding metadata fields proved challenging.

To improve the system's consistency in constructing filters, we divided the EA module into two submodules: extraction and alignment. The extraction module employs an LLM to identify keywords in the user query, without relying on a predefined schema. The alignment module then links each extracted keyword to an item in the database. For this purpose, we constructed a dedicated vector store, derived from the main index, that contains only item names and types. A distance-based metric is applied to align keywords with entries in this index.

This Extract-and-Align (EAL) module proved more reliable than the Extract-and-Associate (EA) approach, as it requires the LLM only to extract keywords, leaving alignment to the secondary module. This division enhances robustness by mapping user terms to index terminology, thereby increasing tolerance to variations in query formulation.

The main limitation of the EAL module lies in database scale: with a large number of items in the alignment index, aligned keywords do not always correspond to the intended entities. To address this issue, we implemented pre- and post-processing functions. Pre-processing improves similarity scores and ensures that irrelevant words are removed from the keyword list, while post-processing constructs coherent filters from the aligned keywords.

With this method, we improved the reliability of keyword extraction. However, the alignment module continued to struggle with linking keywords to the intended items. To address this, we designed a final approach, termed the Adaptive Retriever, which combines the Extract-and-Associate (EA) module with the alignment submodule.

Building on the limitations of the EA method, we simplified the metadata structure so that the model identifies only two fields: item and community. The first field answers the question "What is the user asking for?", while the second specifies "Where should it be retrieved from?". For example, if the user asks for the services linked to a given domain, the model infers that item corresponds to "service" (what) and community to the specified domain (where).

The second step of this approach consists of aligning the extracted keywords. Instead of maintaining a single index containing all items, we constructed two databases: one for communities (services and domains, viewed as collections of items) and another for individual items. This separation enables a more guided alignment process, thereby mitigating the risk of incorrect associations.

 doi:10.20944/preprints202510.0186.v1

3.3. Generation

By providing the generation LLM with a contextual background, we ensure that its responses remain grounded in reliable sources. The prompt for the generation LLM is constructed from the retrieved context combined with the user's query. To reduce the length of the context, we introduce a summarization step, performed by a dedicated LLM. For both summarization and generation, we employ mistral-small3.1:24b, which offers a good balance between model size and consistency. By contrast, lighter models such as gemma3:4b tended to omit key points during summarization.

The source code for the RAG system pipeline, including indexing and embedding, retrieval, and generation, is available in the following Git repository: [Link]. Access to this repository is currently restricted to collaborators within our institute, in accordance with the institute's internal open-source policy. However, the repository will be made available to reviewers upon request to ensure reproducibility and facilitate the evaluation of our work.

4. Results

This section presents the results of the pipeline implementation evaluation, encompassing both technical aspects (i.e., the evaluation of the implemented retrieval methods) and user-oriented perspectives (i.e., the assessment of the system by end-users).

4.1. Evaluation of Retrieval Methods

Evaluating a retrieval-augmented generation (RAG) system requires assessing the quality of its retrieval component. As highlighted in [28], large language models (LLMs) often struggle to correct false or missing information, which makes accurate retrieval essential. This evaluation is typically performed by comparing the retrieved corpus chunks with the expected ones for a given query. Commonly used metrics include Top-K Recall, Mean Reciprocal Rank (MRR), and Normalized Discounted Cumulative Gain (NDCG). Retrieval quality can also be assessed through the informativeness of the retrieved chunks [29], and recent studies have explored leveraging LLMs themselves to judge the quality of retrieved contexts [30,31].

In our case, as described in Section 3.2, we implemented two families of retrieval methods: pre-filtering-based methods and semantic-search-based methods. Since the first family (i.e., pre-filtering-based methods) requires the use of an LLM to extract the query keywords, we first analyze the results obtained with different LLMs for these methods.

4.1.1. Evaluation of Models for the Filter-Extraction Task

To ensure reproducibility and limit computational costs, we considered only open-source LLMs. Due to GPU memory constraints, the experiments focused on relatively small models (i.e., fewer than 30 billion parameters). Accordingly, we selected state-of-the-art generalist models, including Gemma 3 [32], LLaMA 3 [33], Mistral-small-2501 [34], Phi-4 [35], and Qwen 2.5 [36]. For Gemma 3, multiple model sizes were tested to evaluate their impact on performance. Based on the findings in [37], we adopted 4-bit quantized models, which provide substantial memory savings while preserving most of the performance. All models were served using the Ollama framework and were used in the GGUF format with Q4_M quantization.

We evaluated performance using 35 real-world queries suggested by domain experts. Each query required the extraction of specific metadata filters, and the examples were selected to cover a comprehensive range of scenarios involving diverse metadata types. Filter extraction was framed as a classification task, where each label was expected to take a specific value. Accuracy was measured as the proportion of correct answers among all predictions. For cases where no value was expected for a given label, the LLM was required to return a blank response. By incorporating these negative cases, we also computed precision, recall, and F1 score.

To account for the inherent variability of LLM outputs, each test was repeated 10 times, and only responses occurring above a predefined frequency threshold were retained. For instance, if an LLM

 doi:10.20944/preprints202510.0186.v1

was queried 10 times, only responses appearing more than 8 times would be accepted. After testing different configurations, we set the threshold to 0.4.

The results of each model with the Extract-and-Align Module (cf. Section 3.2.2) are presented in Table 1.

Table 1. Results of the metadata extraction task with different LLMs using the Extract-and-Align module.

Model	Accuracy	Recall	Precision	F1
gemma3:1b	0,33	0,18	0,29	0,22
qwen3:1,7b	0,75	0,62	0,76	0,68
gemma3:4b	0,57	0,47	0,59	0,52
mistral:7b	0,56	0,45	0,52	0,48
llama3.1:8b	0,57	0,43	0,55	0,48
gemma3:12b	0,71	0,71	0,64	0,67
mistral-nemo:12b	0,7	0,59	0,68	0,63
phi4:14b	0,73	0,74	0,67	0,7
qwen2.5:14b	0,68	0,64	0,66	0,65
qwen3:14b	0,81	0,8	0,78	0,79
mistral-small3.1:24b	**0,84**	**0,86**	**0,8**	**0,83**
gemma2:27b	0,74	0,76	0,68	0,72

As shown in Table 1, language models exhibit varying levels of performance. First, model size appears to have a significant impact: smaller models tend to hallucinate more frequently, producing outputs that are poorly formatted and consequently yield lower scores. In contrast, larger models generally produce more consistent results, with Mistral-small 3.1:24B achieving an F1 score of 83%.

Based on these results, we concluded that Mistral-small 3.1:24B is the most suitable model for the keyword-extraction task, independent of the retrieval method. We next evaluate the methods in terms of the quality of the retrieved contexts.

4.1.2. Retrieved Context Quality Evaluation

In this sub-section, we evaluate the quality of the contexts retrieved by the system after filters have been extracted (see Section 4.1.1). While the previous sub-section focused on selecting the best LLM for extracting metadata filters from queries, here we assess how effectively the system retrieves relevant chunks of the corpus using those filters. To measure the quality of the retrieved contexts, we compare them with manually constructed reference contexts. We used the same question set introduced in the previous sub-section and built each reference context by applying the expected filters and selecting, among the filtered documents, only the most relevant paragraphs.

After building the test set, we selected a set of automatic evaluation metrics for text generation. Our main selection criterion was the metric's correlation with human judgment. The selected metrics are:

- METEOR [38]: originally developed for machine translation evaluation, this metric relies on unigram matching and incorporates stemming, synonymy, and paraphrasing.
- BERTScore [39]: designed for text generation evaluation, this metric computes token similarity using contextual embeddings rather than exact matches.
- BLEURT [40]: extends BERT by improving its correlation with human ratings through pre-training on synthetic data and fine-tuning on human evaluation datasets.
- LLM-as-a-Judge: an LLM is prompted to assess the quality of the retrieved context, given both the reference context and the user query. The evaluation considers two aspects: accuracy (whether all assertions are factually correct) and relevance (the closeness to the reference context). The model assigns a score from 1 (off-topic context) to 4 (high accuracy and relevance), accompanied by an explanation. These explanations are essential to verify the alignment

 doi:10.20944/preprints202510.0186.v1

between the model's judgment and human ratings. We used Mistral-small 3.1:24B as the judging model.

We evaluated the following five retrieval methods:

- Vanilla: a similarity-based retrieval method relying on basic semantic search without query processing, applied to an index built from raw-extracted data of the reference architectures using a recursive chunking method (see Section 3.2.1).
- Structured Vanilla: a similarity-based retrieval method that applies the vanilla approach to the structured index (see Section 3.2.1).
- Extract-and-Associate (EA): a pre-filtering-based retrieval method (see Section 3.2.2).
- Extract-and-Align (EAL): a pre-filtering-based retrieval method (see Section 3.2.2).
- Adaptive: a pre-filtering-based retrieval method (see Section 3.2.2).

Table 2 presents the results of the retrieved context quality evaluation for each method.

Table 2. Retrieved context quality evaluation of retrieval methods.

Method	Meteor	Bert f1	Bleurt	Judge
Vanilla	0,23	0,27	0,4	0,45
Structured Vanilla	**0,32**	**0,41**	**0,45**	0,55
EA	0,12	0,15	0,15	0,33
EAL	0,29	0,31	0,38	0,48
Adaptative	0,31	**0,41**	**0,45**	**0,59**

The results indicate that all four metrics are strongly correlated, with correlation coefficients ranging from 0.92 (Bert score/Bleurt) to 0.99 (Bert score/ LLM as a Judge). Analysis of the explanations provided by the judge LLM further confirmed the consistency of its evaluation process. Therefore, the subsequent observations are primarily based on this metric.

By comparing the first two methods, Vanilla and Structured Vanilla, we observe that structuring the database improves the quality of the retrieved context. Among all methods, the Adaptive retrieval approach achieves the highest judge score and outperforms the first pre-filtering method, EA(Extract-and-Associate), owing to its robustness to query formulation. Results also show that guiding the alignment module enhances retrieval consistency compared to the EAL (Extract-and-Align) method. Overall, the Adaptive method obtains the best scores in three out of the four evaluation metrics. The Structured Vanilla method ranks second, ahead of the other pre-filtering-based methods.

In our use case, where many queries require the retrieval of multiple entities, it is crucial for the retriever to capture relationships between indexed items. The Structured Vanilla method performs better than naïve retrieval in this respect, but the retrieved contexts are often incomplete. Consequently, this method cannot be either fully discarded or solely relied upon. Instead, we employ it as a fallback strategy when the Adaptive method fails to generate a coherent filter, thereby ensuring that the system always returns a context relevant to the query.

To reduce the complexity of the final prompt, we applied an LLM-based summarization step to the retrieved context. To ensure that no relevant information was lost in this process, the summarized contexts were evaluated using the same metrics as before. As shown in Table 3, summarizing the retrieved context improves the judge score for the two best retrieval methods, namely Structured Vanilla and Adaptive.

Table 3. Evaluation of summarized versus raw retrieved contexts.

Method	Judge
Structured vanilla	0,55
Structured vanilla (summarized)	0,69
Adaptative	0,59
Adaptative (summarized)	**0,74**

 doi:10.20944/preprints202510.0186.v1

As shown in Table 3, the judge LLM consistently values conciseness. This indicates that not all information contained in the retrieved documents is relevant to the query, and that summarization effectively removes extraneous content while preserving the essential elements.

4.2. User-Oriented Evaluation of the System

The main user interface of our RAG system, through which end-users interact with the retrieval and generation components, is shown in Figure 3.

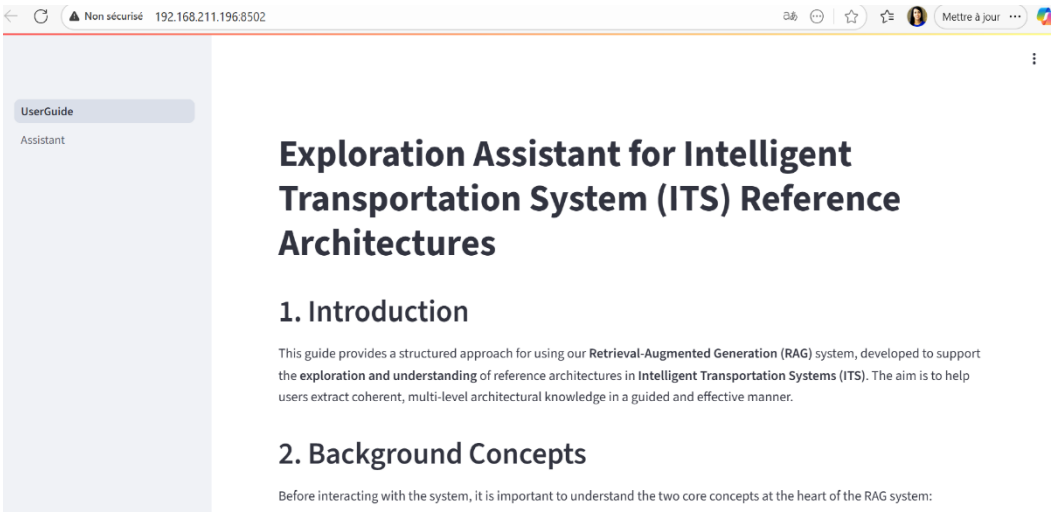

Figure 3. Main user interface of the RAG system.

As shown in Figure 3, the system is accompanied by a user guide designed to help users obtain the most relevant answers and to support effective exploration of the reference architectures. The guide introduces both the system and the reference architectures considered, specifies the types of questions that can be asked, and recommends a top-down approach, progressing from the most general to the most detailed level of information. This approach follows the general framework for high-level system architecture description presented in Figure 2 (see Section 3.1), which distinguishes three levels: operational (why?), describing the services provided by the system; functional (what?), specifying the functions to be performed; and organic (how?), identifying the concrete resources employed. We adopt this hierarchical logic to guide the exploration of reference architectures, using a top-down approach from the most general level of questions to the most detailed. This process consists of the following seven steps:

1. Identify ITS domains;
- Action: Ask the system for the list of ITS domains defined in the reference architectures;
- Expected output: A list of key domains (e.g., Traffic Management, Public Transport, etc.);
2. Explore a selected domain;
- Action: Choose one domain and ask for detailed information;
- Expected output: A description of the domain and a list of ITS services it contains;
3. Investigate an ITS service;
- Action: Select a service relevant to your needs and ask for its functional overview;
- Expected Output: A brief explanation of the service's objectives and its main architectural components;
4. Retrieve architectural objects;
- Action: Ask for the list of functional or physical objects associated with the selected service;
- Expected output: A structured list of relevant architectural elements (e.g., sensors, systems, users);
5. Deep dive into specific objects;
- Action: Request a description of a specific object;
- Expected Output: A detailed explanation of the object's role, purpose, and characteristics;

 doi:10.20944/preprints202510.0186.v1

6. Discover related objects;
- Action: Ask the system to identify objects related to a selected object;
- Expected output: A list of linked or interacting objects in the architecture;

7. Visualize the architecture;
- Action: Ask for the physical architecture diagram of the selected ITS service;
- Expected output: A visual representation showing the key objects and their interactions;

Figures 4–6 illustrate the use of the RAG system following the top-down approach described above. They present the results corresponding to the first two steps as well as the final step of this approach.

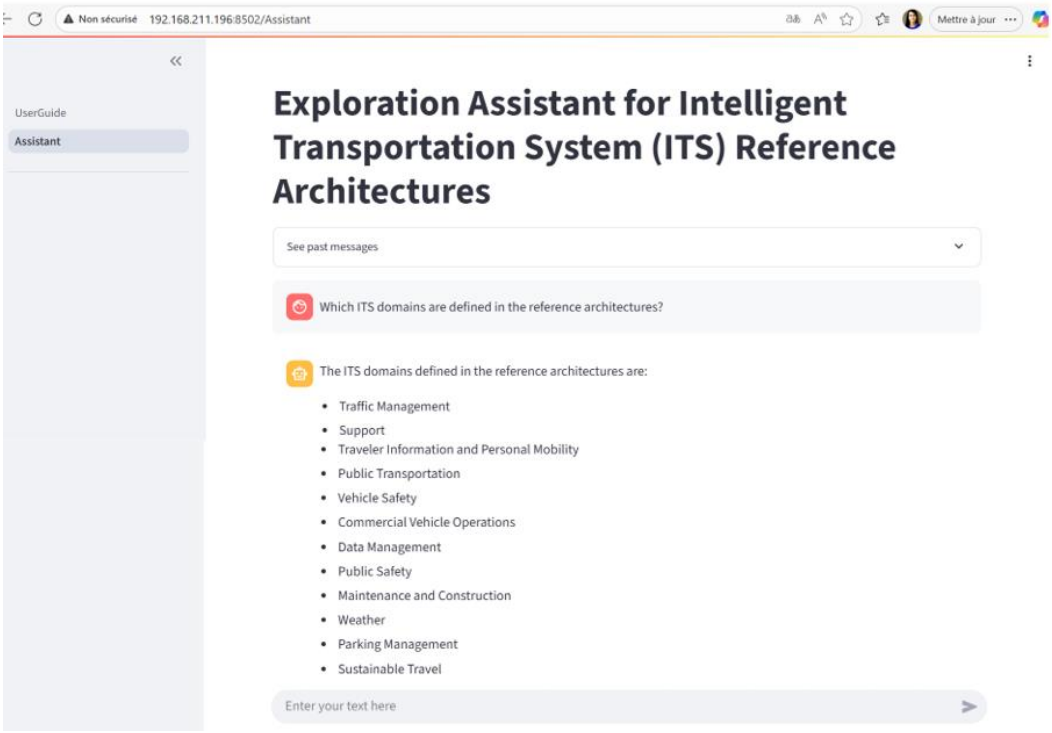

Figure 4. Step 1: identification of ITS domains.

 doi:10.20944/preprints202510.0186.v1

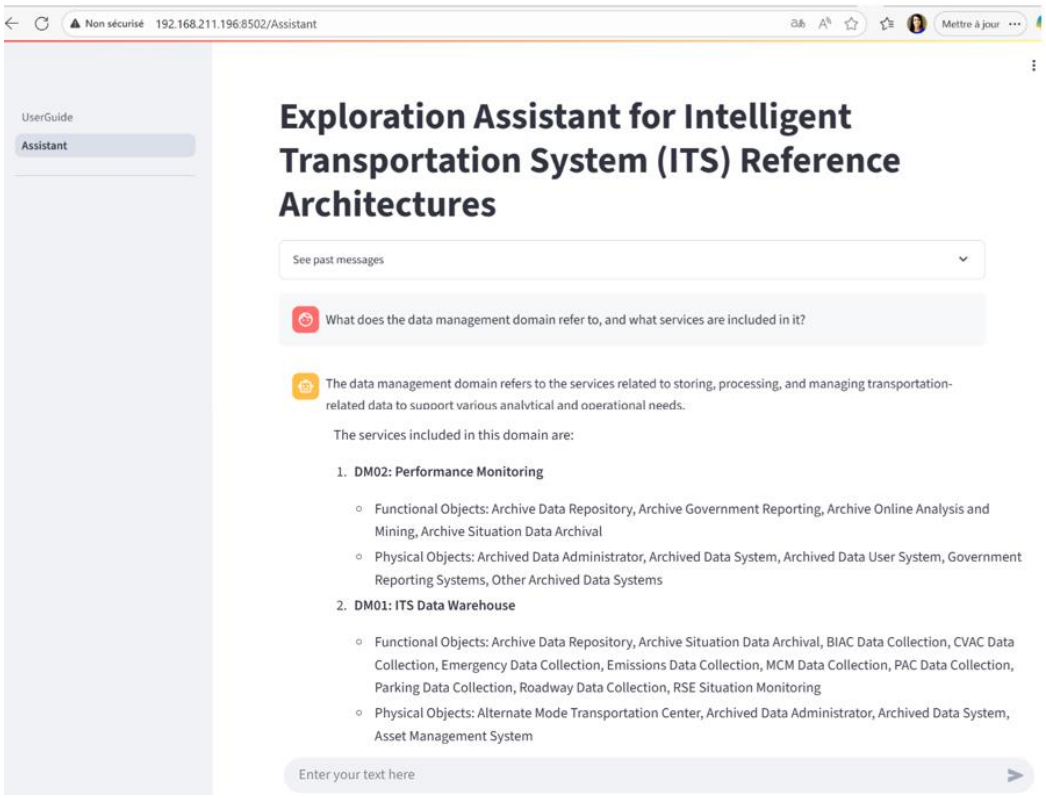

Figure 5. Step 2: exploration of a selected domain.

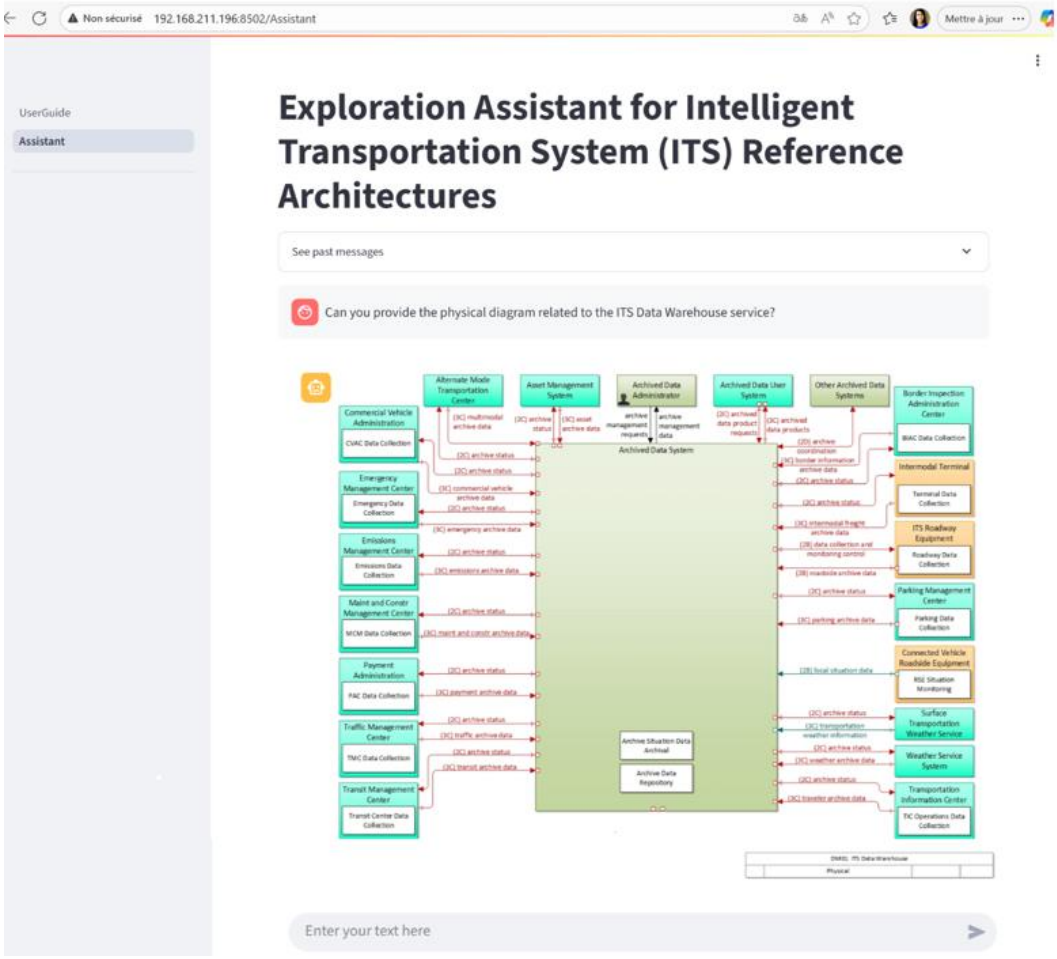

Figure 6. Step 7: visualization of the physical architecture diagram for the selected ITS service.

 doi:10.20944/preprints202510.0186.v1

The system was deployed for evaluation by internal users, namely colleagues from different departments of our institute. These evaluators had diverse backgrounds (systems engineers, systems architects, software developers, etc.) and no prior knowledge of ITS reference architectures. To assess the clarity and accessibility of the user guide, users were allowed to formulate queries freely, without intervention, in order to measure the system's robustness when faced with a variety of query formulations.

For the user-oriented evaluation, three quantitative criteria were defined to measure user satisfaction. These criteria, their definitions, and the associated scoring system are presented in Table 4.

Table 4. User-oriented evaluation criteria, definitions, and corresponding scores.

Criterion	Definition	Score	Label	Description
Conciseness and synthesis	Assesses whether the system presents the main points in a clear, synthesized form, rather than copying raw content.	0	No synthesis	The answer is a raw copy-paste or poorly structured content.
		1	Some synthesis	Partial reformulation, but the answer is still redundant or lacks clarity.
		2	Well-synthesized	The answer is clearly reformulated and structured in a concise, coherent way.
Comprehensibility	Evaluates whether the answer is understandable and adapted to non-expert users.	0	Hard to understand	The answer is too technical or unclear for non-specialists.
		1	Moderately clear	Some parts are understandable; others require expert background.
		2	Clear and accessible	The answer is easy to understand and well-adapted for non-expert users.
Informative value	Measures the extent to which the answer introduces new, relevant information that the user may not have known or expected.	0	No new content	The answer contains only familiar or obvious information.
		1	Partially new	The answer includes some new or enriching information, but nothing substantial.
		2	Highly informative	The answer brings clearly new, useful knowledge or insights for the user.

 doi:10.20944/preprints202510.0186.v1

The evaluation results of our RAG system, obtained from the first ten users considered in our study, are summarized in Table 5.

Table 5. Evaluation results of the system by end users according to the defined criteria.

User	Criteria			Total score
	Conciseness & synthesis	Comprehensibility	Informative value	
User 1	2	1	2	5
User 2	2	2	2	6
User 3	1	2	2	5
User 4	1	1	2	4
User 5	2	1	2	5
User 6	2	2	2	6
User 7	1	2	2	5
User 8	2	2	2	6
User 9	2	1	2	5
User 10	1	2	2	5

As shown in Table 5, the evaluation scores assigned by the ten users indicate a high level of satisfaction with the system. Each user rated the system on three criteria — conciseness and synthesis, comprehensibility, and informative value — with individual scores ranging from 0 to 2 per criterion (maximum total score per user = 6). The total scores for all users sum to 52 out of a maximum possible 60, corresponding to an overall satisfaction rate of 86.7%. This high satisfaction can be attributed, in part, to the utility of the user guide, and specifically the seven-step top-down approach it describes, which is designed to help users obtain the most relevant answers and support effective exploration of the reference architectures.

5. Discussion and Conclusions

Intelligent Transportation Systems (ITS) represent a major research focus for addressing the challenges of safer, more efficient, and environmentally sustainable mobility. Nonetheless, the architectural design of such systems is inherently complex, particularly for designers who are not deeply familiar with the domain.

An effective way to address this complexity is to build on ITS reference architectures, which offer validated frameworks that consolidate essential architectural expertise. Nevertheless, their potential is often insufficiently exploited, partly because the architectural elements and reusable components they encompass are not widely disseminated or easily accessible. Feedback from academic and industrial partners confirmed this gap in awareness and usage. To respond to this issue, this paper proposed a Retrieval-Augmented Generation (RAG) system that allows users to interact with these architectures through a question–answer interface.

The system integrates the generative capabilities of large language models (LLMs) with structured knowledge from established ITS reference architectures, including ARC-IT and FRAME. It was assessed from both technical (i.e., evaluation of retrieval methods) and user-oriented perspectives (i.e., evaluation by end users). The addition of a user guide, designed around a seven-step top-down approach, played a significant role in user satisfaction, as shown by the evaluation results.

These positive outcomes point toward several promising directions for further work. A first objective is to make the system publicly accessible, enabling testing and validation by a broader range of users, and potentially opening new opportunities for collaboration with research and industry partners worldwide. Furthermore, we plan to increase the system's adaptability by introducing autonomous agents capable of selecting the most relevant tools to deliver high-quality answers.

 doi:10.20944/preprints202510.0186.v1

Finally, since the interface supports conversational interaction, another line of work will focus on leveraging dialogue history to improve not only system accuracy and robustness but also the overall user experience.

References

1. Tochygin, M. (2024, November). Integration of Intelligent Transportation Systems in the Context of Sustainable Development: Challenges and Opportunities for Smart Cities. In International Scientific Conference Intelligent Transport Systems: Ecology, Safety, Quality, Comfort (pp. 120-131). Cham: Springer Nature Switzerland.
2. Elassy, M., Al-Hattab, M., Takruri, M., & Badawi, S. (2024). Intelligent transportation systems for sustainable smart cities. Transportation Engineering, 100252.
3. Qureshi, K. N., & Abdullah, A. H. (2013). A survey on intelligent transportation systems. Middle-East Journal of Scientific Research, 15(5), 629-642.
4. Greer, L., Fraser, J. L., Hicks, D., Mercer, M., & Thompson, K. (2018). Intelligent transportation systems benefits, costs, and lessons learned: 2018 update report (No. FHWA-JPO-18-641). United States. Dept. of Transportation. ITS Joint Program Office.
5. Othman, K. M., Alzaben, N., Alruwais, N., Maray, M., Darem, A. A., & Mohamed, A. (2024). Smart Surveillance: Advanced Deep Learning based Vehicle Detection and Tracking Model on UAV Imagery. Fractals.
6. ISO/IEC/IEEE 42020:2019:2019-07.: Software, systems and enterprise — Architecture processes. Int. Organ. Stand. Geneva, Switzerland (2019).
7. Nakagawa, E. Y., Antonino, P. O., Becker, M.: Reference architecture and product line architecture: A subtle but critical difference. In: European Conference on Software Architecture. Springer, Berlin, Heidelberg, pp. 207-211 (2011).
8. ARC-IT Version 9.3 (2025). Available at https://www.arc-it.net/
9. The FRAME Architecture. Available at https://frame-online.eu/frame-architecture/
10. ISO/IEC/IEEE 42010:2022: Software, Systems and Enterprise Architecture Description. ISO. https://www.iso.org/standard/74393.html
11. Zemmouchi-Ghomari, L. (2025). Artificial intelligence in intelligent transportation systems. Journal of Intelligent Manufacturing and Special Equipment.
12. Husak, A., Politis, M., Shah, V., Eshuis, H., & Grefen, P. W. P. J. (2015). Reference architecture for mobility-related services: a reference architecture based on GET service and SIMPLI-CITY project architectures.
13. Molinete, B., Campos, S., Olabarrieta, I., Torre, A. I., Perallos, A., Hernandez-Jayo, U., & Onieva, E. (2015). Reference ITS architectures in Europe. Intelligent Transportation Systems: Technologies and Applications, 3-17.
14. Ezgeta, D., Čaušević, S., & Mehanović, M. (2023, May). Challenges of physical and digital integration of transport infrastructure in Bosnia and Herzegovina. In International Conference "New Technologies, Development and Applications" (pp. 696-701). Cham: Springer Nature Switzerland.
15. Awadid, A. (2022, March). Reference Architectures for Cyber-Physical Systems: Towards a Standard-Based Comparative Framework. In Future of Information and Communication Conference (pp. 611-628). Cham: Springer International Publishing.
16. Katz, D. M., Bommarito, M. J., Gao, S., & Arredondo, P. (2024). Gpt-4 passes the bar exam. Philosophical Transactions of the Royal Society A, 382(2270), 20230254.
17. Yadav, B. (2024). Generative AI in the Era of Transformers: Revolutionizing Natural Language Processing with LLMs. J. Image Process. Intell. Remote Sens, 4(2), 54-61.
18. Rezk, N. M., Purnaprajna, M., Nordström, T., & Ul-Abdin, Z. (2020). Recurrent neural networks: An embedded computing perspective. IEEE Access, 8, 57967-57996.
19. Mondal, H., Mondal, S., & Behera, J. K. (2025). Artificial intelligence in academic writing: Insights from journal publishers' guidelines. Perspectives in Clinical Research, 16(1), 56-57.

 doi:10.20944/preprints202510.0186.v1

20. Xia, Y., Jazdi, N., & Weyrich, M. (2024, September). Enhance FMEA with Large Language Models for Assisted Risk Management in Technical Processes and Products. In 2024 IEEE 29th International Conference on Emerging Technologies and Factory Automation (ETFA) (pp. 1-4). IEEE.
21. Awadid, A., Robert, B., & Langlois, B. (2024, February). Mbse to support engineering of trustworthy ai-based critical systems. In 12th International Conference on Model-Based Software and Systems Engineering.
22. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. Advances in neural information processing systems, 33, 9459-9474.
23. Lei, L., Zhang, H., & Yang, S. X. (2023). ChatGPT in connected and autonomous vehicles: benefits and challenges. Intelligence & Robotics, 3(2), 144-147.
24. Singh, G. (2023). Leveraging chatgpt for real-time decision-making in autonomous systems. Eduzone: International Peer Reviewed/Refereed Multidisciplinary Journal, 12(2), 101-106.
25. Narimissa, E., & Raithel, D. (2024). Exploring Information Retrieval Landscapes: An Investigation of a Novel Evaluation Techniques and Comparative Document Splitting Methods. arXiv preprint arXiv:2409.08479.
26. Zhang, X., Zhang, Y., Long, D., Xie, W., Dai, Z., Tang, J., ... & Zhang, M. (2024). mgte: Generalized long-context text representation and reranking models for multilingual text retrieval. arXiv preprint arXiv:2407.19669.
27. Guo, Z., Xia, L., Yu, Y., Ao, T., & Huang, C. (2024). Lightrag: Simple and fast retrieval-augmented generation. arXiv preprint arXiv:2410.05779.
28. Chen, J., Lin, H., Han, X., & Sun, L. (2024, March). Benchmarking large language models in retrieval-augmented generation. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 38, No. 16, pp. 17754-17762).
29. Yu, H., Gan, A., Zhang, K., Tong, S., Liu, Q., & Liu, Z. (2024, August). Evaluation of retrieval-augmented generation: A survey. In CCF Conference on Big Data (pp. 102-120). Singapore: Springer Nature Singapore.
30. Es, S., James, J., Anke, L. E., & Schockaert, S. (2024, March). Ragas: Automated evaluation of retrieval augmented generation. In Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations (pp. 150-158).
31. Ru, D., Qiu, L., Hu, X., Zhang, T., Shi, P., Chang, S., ... & Zhang, Z. (2024). Ragchecker: A fine-grained framework for diagnosing retrieval-augmented generation. Advances in Neural Information Processing Systems, 37, 21999-22027.
32. Team, G., Kamath, A., Ferret, J., Pathak, S., Vieillard, N., Merhej, R., ... & Iqbal, S. (2025). Gemma 3 technical report. arXiv preprint arXiv:2503.19786.
33. Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., ... & Ganapathy, R. (2024). The llama 3 herd of models. arXiv e-prints, arXiv-2407.
34. https://mistral.ai/news/mistral-small-3
35. Abdin, M., Aneja, J., Behl, H., Bubeck, S., Eldan, R., Gunasekar, S., ... & Zhang, Y. (2024). Phi-4 technical report. arXiv preprint arXiv:2412.08905.
36. Yang, A., Yu, B., Li, C., Liu, D., Huang, F., Huang, H., ... & Zhang, Z. (2025). Qwen2. 5-1m technical report. arXiv preprint arXiv:2501.15383.
37. Jin, R., Du, J., Huang, W., Liu, W., Luan, J., Wang, B., & Xiong, D. (2024, August). A comprehensive evaluation of quantization strategies for large language models. In Findings of the Association for Computational Linguistics ACL 2024 (pp. 12186-12215).
38. Banerjee, S., & Lavie, A. (2005, June). METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization (pp. 65-72).
39. Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., & Artzi, Y. (2019). Bertscore: Evaluating text generation with bert. arXiv preprint arXiv:1904.09675.
40. Sellam, T., Das, D., & Parikh, A. P. (2020). BLEURT: Learning robust metrics for text generation. arXiv preprint arXiv:2004.04696.

 doi:10.20944/preprints202510.0186.v1