PDF Download
3796550.pdf
20 February 2026
Total Citations: 0
Total Downloads: 32

Accepted: 10 January 2026
Revised: 11 November 2025
Received: 28 April 2025

Citation in BibTeX format

RESEARCH-ARTICLE

Dialogue Meets Data: A Large Language Model Approach for Efficient Linked Data Retrieval

OMAR MUSSA, Saudi Electronic University, Riyadh, Ar Riyad, Saudi Arabia

OMER F RANA, Cardiff University, Cardiff, South Glamorgan, Wales, U.K.

BENOÎT GOOSSENS, College of Biomedical and Life Sciences, Cardiff, Wales, U.K.

PABLO OROZCO TER WENGEL, College of Biomedical and Life Sciences, Cardiff, Wales, U.K.

CHARITH PERERA, Cardiff University, Cardiff, South Glamorgan, Wales, U.K.

Open Access Support provided by:

Saudi Electronic University

Cardiff University

College of Biomedical and Life Sciences

ACM Transactions on the Web
https://doi.org/10.1145/3796550
EISSN: 1559-114X

Dialogue Meets Data: A Large Language Model Approach for Efficient Linked Data Retrieval

OMAR MUSSA, College of Computing and Informatics, Saudi Electronic University, Riyadh, Saudi Arabia and School of Computer Science and Informatics, Cardiff University, Cardiff, United Kingdom
OMER RANA, School of Computer Science and Informatics, Cardiff University, Cardiff, United Kingdom
BENOIT GOOSSENS, School of Biosciences, Cardiff University, Cardiff, United Kingdom
PABLO OROZCO TER WENGEL, School of Biosciences, Cardiff University, Cardiff, United Kingdom
CHARITH PERERA, School of Computer Science and Informatics, Cardiff University, Cardiff, United Kingdom

While large language models (LLMs) have captured global attention for their linguistic abilities, our work harnesses their power to overcome traditional barriers in querying Linked Data (LD) and Resource Description Framework (RDF) triplestores. This paper presents an innovative framework that integrates LLMs into conversational user interfaces (UIs), enabling the dynamic generation of precise SPARQL queries without the necessity for constant retraining. Most conversational UI models struggle with adaptability as they require frequent retraining whenever datasets are updated or expanded. This limitation impedes their effectiveness as general-purpose extraction tools. To address this challenge, our approach[1] seamlessly incorporates LLMs into the conversational UI process, fostering a more sophisticated understanding and interpretation of user queries and enhancing overall responsiveness. By leveraging the advanced natural language processing capabilities of LLMs, our method improves RDF entity extraction in web systems that utilise conventional chatbots. Furthermore, it extends the functionality of these chatbots, allowing them to respond directly to queries based on the RDF schema while providing an assistive interface that deepens understanding of the dataset and its underlying domain. This facilitates the extraction of more meaningful information. By adopting this approach, interactions become more refined and context-sensitive, a crucial advancement for managing the intricate query structures commonly found in RDF datasets and Linked Open Data (LOD) endpoints. We have evaluated our approach in practical settings by assessing the tool's ability to address complex queries and to answer general ecological queries, with the outputs evaluated by human experts. The results demonstrate a notable improvement in system expressiveness and response accuracy, showcasing the potential of LLMs to transform information retrieval. The findings not only confirm their adaptability in enhancing existing systems but also open up exciting possibilities for their deployment in specialised web information domains, paving the way for future research in this evolving field.

CCS Concepts: • **Information systems → Information extraction**; **Resource Description Framework (RDF)**; **Search interfaces**; *Language models*; • **Human-centered computing → User interface toolkits**.

Additional Key Words and Phrases: Conversational UIs, Large Language Models, Linked Data, RDF Triplestore, Information Retrieval.

[1] This article is an extended version of the paper published in the Proceedings of the WISE 2024 Conference [26].

Authors' Contact Information: Omar Mussa, College of Computing and Informatics, Saudi Electronic University, Riyadh, Saudi Arabia and School of Computer Science and Informatics, Cardiff University, Cardiff, United Kingdom; e-mail: O.Mousa@seu.edu.sa; Omer Rana, School of Computer Science and Informatics, Cardiff University, Cardiff, United Kingdom; e-mail: RanaOF@cardiff.ac.uk; Benoit Goossens, School of Biosciences, Cardiff University, Cardiff, United Kingdom; e-mail: GoossensBR@cardiff.ac.uk; Pablo Orozco Ter Wengel, School of Biosciences, Cardiff University, Cardiff, United Kingdom; e-mail: Orozco-terWengelPA@cardiff.ac.uk; Charith Perera, School of Computer Science and Informatics, Cardiff University, Cardiff, United Kingdom; e-mail: PereraC@cardiff.ac.uk.

ACM 1559-114X/2026/2-ART
https://doi.org/10.1145/3796550

1 Introduction

In recent years, significant advancements have been made in the development of large language models (LLMs), which have shown robust capabilities across a broad spectrum of natural language processing (NLP) tasks. These models can execute tasks without the need for specific fine-tuning, as they can be directed through instructional prompts embedded within the requests, thereby facilitating their application across various domains [8, 39]. This scientific progress has catalysed a surge in research aimed at examining their applicability and efficacy across a diverse array of topics and methodologies such as education [4], healthcare [33], and finance [11], as well as for specific NLP tasks like replacing traditional question-answering models [13, 35]. In particular, commercial versions of these models, such as GPT, have received considerable attention and have been promoted as optimal solutions for integration within chatbot applications and backend text processing solutions. This has opened up new avenues for leveraging artificial intelligence to enhance user interactions and streamline data processing.

Despite these advancements, one area that still needs to be explored is the potential of LLMs to serve as entity extractors within existing linked data (LD) systems and Resource Description Framework (RDF) triplestores. The integration of LLMs in such capacities could potentially revolutionise the way entities are extracted and managed, offering more dynamic and context-aware data handling capabilities. However, the efficacy and reliability of LLMs in this specific role have not been thoroughly tested, raising questions about their practicality and performance in real-world settings.

In this study, we employ a novel toolkit known as ForestQB [25], specifically developed to explore observational LD to support bioscientists and wildlife conservation efforts. This system integrates an interface that combines a chatbot with a traditional form-based graphical user interface (GUI). It facilitates the extraction of data from diverse observational LD endpoints. The chatbot within this toolkit is configured to interpret user queries and handle entity extraction. This process automates populating the GUI with appropriate selections and executes the search process. Consequently, the chatbot, as a general-purpose tool, is designed to adapt to changes in the dataset, thereby not being restricted to specific names or sentences within its training data in an ontology-independent manner. Therefore, the model used by the chatbot lacks the ability to comprehend the relationships within the knowledge graph concerning the sensors and their observations (see Figure 1). As a result, constructing queries that incorporate an arbitrary array of filters and entities through the Conversational UI is unfeasible. In addition, general queries about the descriptions of the sensors and entities within the dataset cannot be accommodated.

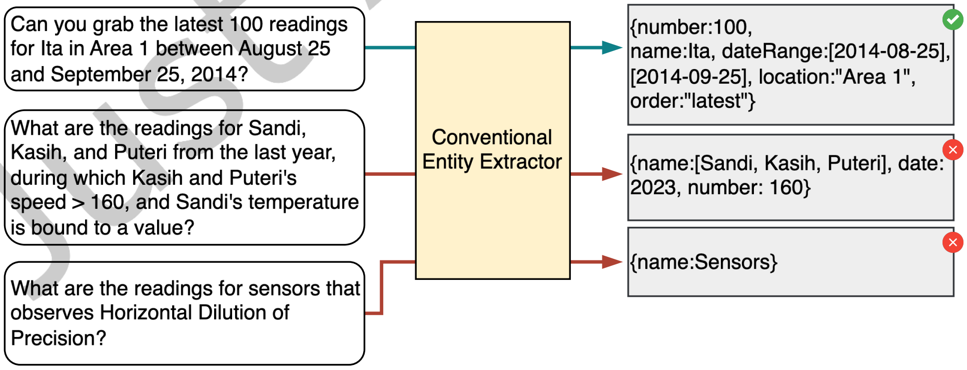

Can you grab the latest 100 readings for Ita in Area 1 between August 25 and September 25, 2014?
What are the readings for Sandi, Kasih, and Puteri from the last year, during which Kasih and Puteri's speed > 160, and Sandi's temperature is bound to a value?
What are the readings for sensors that observes Horizontal Dilution of Precision?
Conventional Entity Extractor
{number:100, name:Ita, dateRange:[2014-08-25],[2014-09-25], location:"Area 1", order:"latest"}
{name:[Sandi, Kasih, Puteri], date: 2023, number: 160}
{name:Sensors}

Fig. 1. Illustration of the limitations of conventional entity extraction within our current system, arising from adherence to a general-purpose approach. The first query demonstrates a scenario where conventional entity extraction is effective. In contrast, the subsequent two queries highlight challenges in accurately extracting relevant entities and understanding their relationships.

In this study, we address the following research questions:

RQ1: Can LLMs be integrated with *traditional* Chatbots to access Linked Data in the context of Wildlife conservation?

RQ2: What are the challenges associated with incorporating LLMs into our current research?

RQ3: How can LLMs offer distinctive strategies to support wildlife conservation efforts?

The objective of our research is to critically evaluate the integration of LLMs to enhance expressivity and functionality of systems dealing with LD and RDF triplestores. By conducting empirical tests we aim to provide insights into the benefits and limitations of using LLMs as entity extractors, thus contributing to the broader discourse on the applicability of LLMs in complex data environments. In summary, our contributions are as follows:

- We introduce a novel strategy for enhancing entity extraction, question answering over observational linked data by utilising LLMs as a component in conversational UIs systems.
- A comprehensive set of design principles to assist practitioners in enhancing their research through the integration of LLMs.
- We conducted empirical testing with state-of-the-art LLMs to assess the benefit of our methodology. The findings indicate a significant potential to improve information retrieval and reasoning processes over RDF data.
- We conducted a user study to evaluate the effectiveness of LLMs in supporting wildlife researchers. The study was focused on better understanding how data discovery and knowledge acquisition, through enabling exploratory questions, could deepen understanding of data structures and applications.

The paper is structured as follows. Section 2 provides an overview of existing techniques and challenges, offering insight into the proposed solutions. Section 3 outlines the approach, and introduces the proposed use cases. Section 4 presents an in-depth discussion of the challenges and techniques employed to implement the solution. Section 5 describes the experimental setup and evaluation for Use-Case 1 and Use-Case 2, accompanied by a discussion of the results. Section 6 details the setup for Use-Case 3 and the associated human evaluation, with further discussion of the findings. Section 7 discusses the research limitations of the proposed approach. Finally, Section 8 concludes the paper and outlines potential directions for future work.

2 Background and Related Work

ForestQB [25] represents an innovative SPARQL query builder that merges a chatbot interface with a form-based graphical user interface (GUI) for the construction and execution of SPARQL queries. This integration facilitates the extraction of pertinent information from observational linked data. The implementation of a chatbot in ForestQB has notably accelerated the query construction process, enabling users to directly translate their inquiries into the GUI through interactions with the chatbot. The comprehensive experimentation conducted with ForestQB indicates a heightened level of user satisfaction and a significantly reduced learning curve. Nevertheless, the chatbot's functionality within ForestQB is not without limitations, particularly in its inability to perform certain tasks achievable by the form-based GUI. Specifically, it falls short in extracting an arbitrary set of entities from user queries to apply a corresponding set of filters to them. This limitation stems from the adoption of an ontology-independent approach in the chatbot's design, aimed at ensuring compatibility with various observational linked datasets that the GUI can query.

2.1 Limitations of LLMs

While LLMs exhibit significant capabilities, they have notable limitations that render them unsuitable for directly replacing chatbots employed within conversational UIs. These UIs, such as the one utilised in our experiment, are designed to automate the process of populating user selections on the UI, thereby constructing SPARQL queries to extract information from LD based on user input. One major issue is that LLM responses are not guaranteed to be consistently structured as expected, which can disrupt or completely break the UI interactivity, which poses a significant risk [14, 17, 39]. Additionally, LLMs are prone to hallucinations, generating responses that may be plausible but are factually incorrect or irrelevant, further compromising the reliability of the interaction [17, 30]. To mitigate these issues, fine-tuning LLMs or employing few-shot prompting is often necessary to improve their ability to provide accurate and contextually appropriate responses to better align their output with the expected format and content, thereby restricting their responses to be more consistent and relevant to the task at hand [5, 39]. Fine-tuning can enhance the performance of LLMs in specialised applications, but these measures do not entirely eliminate the inherent risks associated with LLMs in contexts of system interaction [5, 17].

2.2 Generating SPARQL queries using LLMs

In our initial exploration, we considered LLMs to generate SPARQL queries to enhance the expressivity of ForestQB. This approach was aimed at addressing the challenges we encountered in increasing the system's capabilities. However, upon reviewing existing research, it became apparent that LLMs are currently unreliable for generating accurate SPARQL queries [12, 13, 24, 35]. To verify this, we conducted an experiment using GPT-4, which involved generating queries from a set of simple inputs specified by the toolkit in conjunction with the necessary RDF definition file to guide the model (the dataset is discussed in Section 5.1). However, the outcomes were predominantly unsuccessful; all but one of the generated queries failed to produce any results, and the one that did was incorrect. Although the queries were structurally sound and conformed to SPARQL standards, the model defaulted to its intrinsic understanding of the SOSA [16] ontology embedded within the dataset. This resulted in a disregard for the specific context and data provided. Despite demonstrating adeptness in precise encoding, effective entity extraction, and appropriate filter application, our findings indicate that LLMs are not yet dependable for the direct translation of user queries into SPARQL. Nevertheless, they may still contribute to enhancing data extraction methods. Further details can be found at (https://github.com/i3omar/LLM-Integration-Data).

2.3 Entity Extraction

Traditional entity extraction, which involves identifying and classifying named entities within unstructured text, has historically relied on rule-based systems and machine learning algorithms. Rule-based approaches typically use handcrafted rules, regular expressions, and domain-specific dictionaries to identify entities. While these systems can be effective in well-defined contexts, they lack flexibility and adaptability, limiting their capacity to handle diverse linguistic structures, new entity types, and evolving terminology. Machine learning-based methods, such as Conditional Random Fields (CRF), which is commonly used for NER, and Hidden Markov Models (HMM), improved these systems by learning patterns from labelled data. However, these methods still depend heavily on the availability and quality of annotated datasets and are often domain-specific, requiring frequent tuning and retraining when applied to new domains, which can be resource-intensive [15].

When discussing more advanced entity extraction, we refer to Named Entity Recognition (NER), a process that identifies entities within user input and traditionally classifies them into broad categories such as organisations, people's names, locations, dates, times, and quantities [3, 15]. Following this, Named Entity Disambiguation (NED) is undertaken to determine the specific meaning of each entity within its context [3]. For instance, "Apple" could denote the fruit in general contexts or the technology company in business-related ones. Finally, Entity Linking aims to connect the extracted entities to their appropriate resource URIs, often using embeddings and external knowledge bases, such as Wikidata or DBpedia, to facilitate the construction of accurate queries [3, 22, 28]. A significant challenge in this process, especially within chatbot applications, arises when entities are mentioned indirectly rather than by their canonical names, such as referring to "the iPhone company" instead of "Apple." This challenge exceeds the capabilities of traditional entity extraction and Entity Linking approaches [3, 28].

To illustrate the depth of this issue, consider a scenario in which a user refers to "the sensor that observes water level in project 1" as a way of identifying an entity named "SensorA," without knowledge of its precise designation. Similarly, identifying the word "heat" as referring to a temperature entity in RDF data highlights the nuanced contextual understanding required in entity linking. Traditional linking methods typically rely on pretraining with embeddings, which map words and phrases to dimensional vectors and leverage knowledge bases for accuracy [28]. However, this approach is limited in highly dynamic, real-time data contexts, such as observational linked data, where data frequently updates. This issue also arises when aiming to create a general-purpose approach that can work with any dataset without requiring extensive training. The continual change in data necessitates concurrent retraining, rendering traditional methods less viable for such flexible, general-purpose applications.

2.4 Question-Answering over LD and RDF Triplestores

Traditional approaches to developing models for question-answering systems require specific training for each dataset or adjustments due to structural changes in the data. This process necessitates developing and maintaining models trained at comprehending and interpreting data, which is both resource-intensive and time-consuming due to the complexity and variability inherent in the data [10]. As a result, conventional methods are impractical for creating general-purpose solutions that require adaptability, such as toolkits that can interface with multiple datasets or manage changes within the same dataset. In response to these challenges, the adoption of LLMs provides a promising alternative [5, 34]. These models enhance the adaptability and efficiency of question-answering systems by reducing the need for extensive retraining and facilitating more flexible integration with diverse and dynamically evolving datasets, such as those in observational LD [5, 7, 14, 23, 29, 35].

2.5 Insights from Current Literature

In light of recent advancements in SPARQL query construction tools, incorporating chatbot interfaces with traditional form-based GUIs in ForestQB has significantly expedited query formulation and enhanced user satisfaction by enabling users to directly translate their natural language questions into automated search actions via the GUI [25]. This design, while user-friendly and effective in reducing the learning curve, is limited by its ontology-independent approach, as it relies on general entity extraction tools such as spaCy (https://spacy.io) and Duckling (https://github.com/facebook/duckling). Such an approach enhances flexibility across various datasets but restricts the complexity of queries that the chatbot can accurately handle. Notably, the system struggles with extracting arbitrary entities from user input and reasoning over the knowledge graph that are essential for performing more nuanced analyses (e.g., determining sensor functionalities or translating qualitative descriptors such as "fast" into measures of animal speed).

A possible solution to these challenges involves incorporating LLMs to serve as a more sophisticated entity extractor, as well as an RDF reasoning and exploratory interface that enables users to formulate general queries, understand the dataset and domain, and ultimately extract more useful information. LLMs can leverage RDF data definitions to better interpret and process the underlying semantics of user queries. This integration has the potential to enhance the chatbot's capabilities, allowing it to perform direct information extraction and complex reasoning tasks that are particularly relevant in specialised fields such as wildlife ecology. Such an approach promises to bridge the gap between the agility of chatbot interfaces and the depth of understanding required for intricate query interpretation. To the best of our knowledge, our conference paper [26] represents the first study addressing this specific challenge by proposing a general-purpose solution that does not require training on a specific dataset, thereby paving the way for further research into adaptable, ontology-aware conversational agents in the context of observational LD.

3 Proposed Approach

ForestQB [25] is a novel SPARQL query builder that combines a chatbot interface with a form-based GUI to construct and execute SPARQL queries, enhancing the extraction of relevant information from observational LD; the source code is available at (https://github.com/i3omar/ForestQB). The integration of a chatbot has sped up the query construction, allowing users to seamlessly translate inquiries into the GUI (see Figure 2). The chatbot is implemented with the RASA framework, an open-source platform for conversational agents that provides natural language understanding through intent classification and entity recognition, together with rule-based and learning-based dialogue management. This enables ForestQB to map free-text requests to structured parameters in the GUI while remaining ontology-agnostic. Testing of ForestQB shows increased user satisfaction and a significantly reduced learning curve. However, the chatbot has limitations, notably its inability to perform every task achievable by the form-based GUI, such as extracting and filtering an arbitrary set of entities from user queries. These limitations follow from a design choice to keep the chatbot ontology independent, which maintains compatibility with diverse datasets but constrains the complexity of questions and the resulting accuracy.

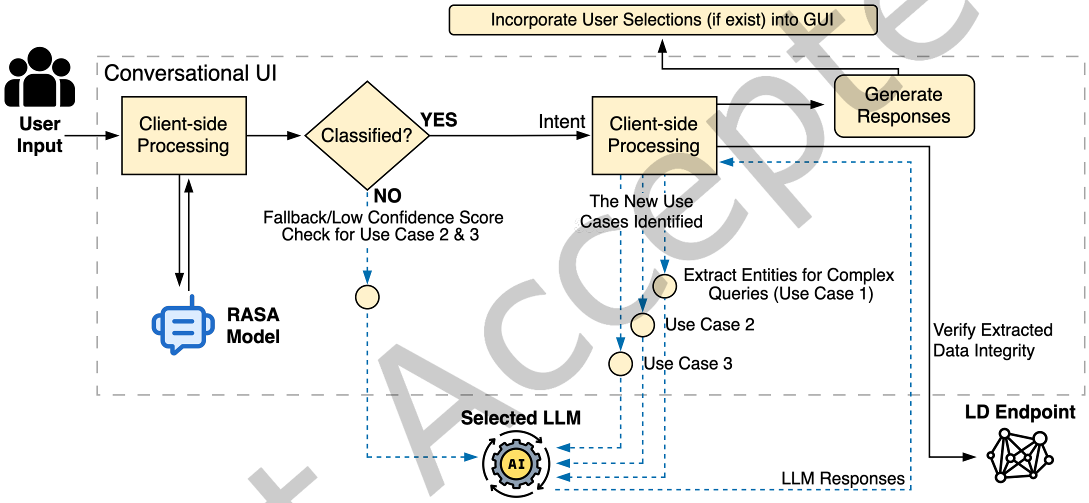

Incorporate User Selections (if exist) into GUI
Conversational UI
User Input
Client-side Processing
Classified?
YES
Intent
Client-side Processing
Generate Responses
NO
Fallback/Low Confidence Score Check for Use Case 2 & 3
The New Use Cases Identified
Extract Entities for Complex Queries (Use Case 1)
Use Case 2
Use Case 3
RASA Model
Verify Extracted Data Integrity
Selected LLM
AI
LD Endpoint
LLM Responses

Fig. 2. A conceptual representation of the conversational UI workflow within the current system. The blue dashed arrows illustrate the proposed enhancements to improve query formulation and information retrieval processes.

To overcome these limitations, it is proposed that an LLM be integrated as an advanced entity extractor and URI linker, explicitly grounded in the dataset's RDF schema, as shown in Figure 2 and 5. Rather than invoking a traditional RDFS/OWL reasoner, the LLM is intended to interpret retrieved schema definitions and relationships, such as subclassing, property inheritance, and term labelling, provided alongside the user's query so as to improve entity linking and query planning. This is operationalised via retrieval-augmented generation (RAG): for each utterance, relevant schema fragments are retrieved from a vector index and a targeted prompt is constructed instead of a static prompt. In this way, prompts are augmented with contextually pertinent labels, axioms, and property notes, yielding more reliable URI linking and more accurate SPARQL generation. For example, when a user requests items of a certain type and then specifies a threshold on a characteristic, the LLM aligns the type mention with the correct class URI using labels and hierarchy, identifies the appropriate property for the characteristic from the retrieved schema notes, interprets any unit hints, and applies the corresponding filter in the planned query.

This improvement would enable the chatbot to directly utilise RDF data definitions and better respond to complex queries, particularly enhancing its application in wildlife ecology. Therefore, we propose three use cases to address these shortcomings and enhance the expressivity of the tool using LLMs as follows:

Use Case 1: Applying Filters Freely on the Sensor Properties using the Conversational UI. This refers to enabling users to apply all available filters within the graphical user interface freely using the chatbot, without requiring retraining to perform these tasks specifically (see Figure 3). The approach is centred on two primary tasks:

(1) **Identifying Explicit Entities and Contextual URI Mapping**: This task involves accurately recognising explicit entities, including sensors, their properties, and filters, in the user's input and dynamically linking each to the appropriate Uniform Resource Identifier (URI) based on the surrounding context. This mapping ensures each filter operation is accurately directed to the relevant sensor properties, thereby maintaining data consistency and reliability.

(2) **Recognising Implicit Entities for Enhanced Usability**: Beyond explicit entities, the system also identifies implicit entities, which are terms or references indirectly linked to sensors, their properties, or filters. By associating these implicit entities with the correct URIs, the interface allows users to communicate naturally with the chatbot, reducing the need for exact phrasing and enhancing conversational flow.

This use case not only streamlines the process of applying filters to sensor properties but also enhances the quality of interactions within the conversational user interface, establishing it as a highly effective information retrieval tool for crafting more advanced queries.

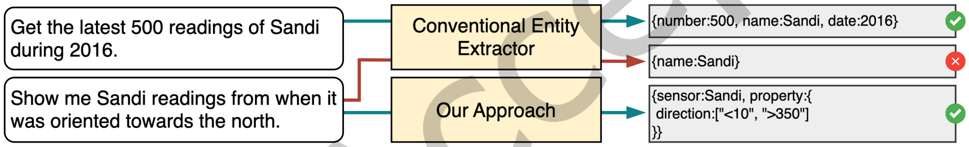

Get the latest 500 readings of Sandi during 2016.
Show me Sandi readings from when it was oriented towards the north.
Conventional Entity Extractor
Our Approach
{number:500, name:Sandi, date:2016}
{name:Sandi}
{sensor:Sandi, property:{ direction:["<10", ">350"] }}

Fig. 3. Illustration of Use Case 1: A scenario demonstrating the limitations of conventional entity extractors.

Use Case 2: Querying RDF Data Schemas by using Conversational UI. This use case explores the capability of enabling users to query sensor data and associated metadata within RDF data schemas through the conversational UI. The objective is to create a generalisable and adaptable system that can interface seamlessly with any LD endpoint, particularly those containing observational data. By leveraging RDF data structures, users can explore diverse sensor information, data properties, and relationships.

Based on the embedded RDF definitions, the system should dynamically interpret and respond to user queries. This adaptability ensures that the conversational UI can efficiently retrieve and convey relevant information across multiple domains and RDF triplestores. It should also reduce the need to retrain the chatbot model for each new dataset, making the approach both scalable and practical for a wide range of applications.

Use Case 3: Supporting Wildlife Researchers with Data Discovery and Knowledge Acquisition. Aligned with the primary aim of this research to improve information retrieval for bioscience researchers, this use case leverages LLMs to support wildlife researchers by addressing inquiries that foster both data discovery and knowledge acquisition. By enabling researchers to pose exploratory questions for which direct answers may not exist within the available data, the LLMs facilitate a deeper understanding of potential data structures and applications, thus enhancing their information retrieval experience.

For example, a researcher may ask general questions about collar data, not only to understand its basic characteristics but also to learn what types of questions might be relevant or what insights could be expected from such data. This approach provides foundational knowledge that helps researchers formulate more precise, contextually relevant queries, ultimately enhancing both their data discovery capabilities and their decision-making processes. Consequently, this application of LLMs enables a more informed, insightful, and productive engagement with data, aligning with the overarching goal of advancing information retrieval within bioscience research.

3.1 Filtering Noisy and Out-of-Domain Queries

Queries that are noisy, ambiguous, or out-of-domain should not be forwarded to the LLM. Forwarding all inputs to the LLM may appear to simplify the architecture, but it introduces practical drawbacks. Costs grow with input length and frequency. Latency rises and makes interactive querying less fluid. Most importantly, it can introduce unstable behaviour in the dialogue. We therefore employ a lightweight gating layer in front of the LLM that filters requests before they reach the language model. This layer is implemented through RASA intent classification (see Figure 2).

RASA provides a modular natural language understanding pipeline and rule-based dialogue management. In ForestQB, the NLU component assigns each utterance to a curated set of supported intents with an associated confidence score. We include a dedicated `out_of_scope` intent to capture inputs that do not match the task, as well as guards for incomplete or malformed requests. If `out_of_scope` is predicted, the system does not call the LLM. Instead, it triggers an explicit recovery policy that offers clarification or returns a concise explanation of what is supported.

This design yields several benefits. First, it reduces cost and improves responsiveness by avoiding unnecessary LLM calls. Second, it improves reliability by preventing the LLM from attempting to answer requests that lie outside the domain. Third, it reduces the risk of model hallucination, prompt injection, or leakage of irrelevant context. Finally, it provides a clear and auditable boundary between supported tasks and unsupported inputs, which is important for evaluation and maintenance.

This pipeline keeps ForestQB ontology independent while maintaining efficiency and accuracy. RASA handles intent recognition and conversation control. The LLM is reserved for cases where generative capabilities are genuinely beneficial.

4 LLM Integration

4.1 Prompts Templates and RDF Annotation

An essential step in enhancing the accuracy of outputs generated by LLMs is the preparation of an appropriate prompt. Recent studies have demonstrated the high sensitivity of LLMs to the specificity of prompts [1, 19, 40]. Consequently, to facilitate the generation of valid JSON from user queries, a JSON template has been embedded within the prompt to provide the LLM with a clear indication of the expected output. In addition, two distinct types of prompts were developed to evaluate the most effective method for extracting entities using zero-shot examples. The first prompt was succinct, offering minimal instruction, while the second was more elaborate, clarifying relevant filters to assess whether this would lead to enhanced output accuracy. We conducted experiments using `gpt-3.5-turbo-0125` on a set of 20 questions, equally divided between 10 straightforward and 10 complex inquiries, to ascertain whether the outputs differed significantly between the two prompts. Following the initial study, we extended the evaluation to include few-shot prompting alongside the original zero-shot setting. In few-shot prompting, we added a small number of representative examples to the prompt to illustrate the desired output format and reasoning, enabling the model to generalise without further training.

4.1.1 Non-Annotated Prompt. Initially, we conducted a test on the prompt without including RDF data to assess the entity extraction capabilities of the LLM in the absence of any annotation. As noted earlier, the first prompt was brief, merely introducing the model along with its intended function and subsequently presenting an abstract

JSON output template. Conversely, the second prompt provided an expanded explanation of the expected filters. Under zero-shot conditions, Prompt-1 outperformed Prompt-2 for entity extraction and overall task success. As outlined in Table 1, the more detailed Prompt-2 offered a modest gain only on filter application. These findings suggest that, in the absence of context, additional instruction can distract the model from adhering to the JSON template despite achieving marginally improved filter extraction.

Introducing few-shot prompting substantially altered the results. Without RDF, overall task success rose from 15% to 60% for Prompt 1 and from 10% to 65% for Prompt 2. Under these conditions, Prompt 2 offered a modest advantage, largely attributable to stronger filter application. In short, few-shot examples mitigated the earlier issues with adherence to the JSON template and improved filter reasoning even in the absence of annotations.

Table 1. Comparison of accuracy for the two LLM prompts without RDF annotations, evaluated in zero-shot and few-shot settings across three tasks: extracting entities, extracting properties, and applying filters, with overall task success reported.

	Zero-shot (NoRDF)		Few-shot (NoRDF)	
Task	Prompt-1	Prompt-2	Prompt-1	Prompt-2
Extracting Entities	**65%**	35%	90%	90%
Extracting Properties	**50%**	45%	70%	70%
Applying Filters	15%	**20%**	65%	**85%**
Full Task Success	**15%**	10%	60%	**65%**

4.1.2 Annotated Prompt: In our subsequent experiment, we enriched the prompt with the necessary RDF data to provide contextual relevance to the query while maintaining the exact prompt details and JSON template. As indicated in Table 2, the inclusion of annotations led to a significant enhancement in performance, with a relative accuracy improvement of at least 133% when comparing the non-annotated Prompt-1 and its annotated counterpart. This result underscores the significance of incorporating annotations to augment the entity extraction process.

Adding few-shot learning to the use of annotations produced the strongest results. Full task success increased from 35% to 70% for Prompt-1 and from 50% to 85% for Prompt-2. Entity extraction reached a ceiling of 90 to 95% under both zero-shot and few-shot settings with RDF, while the most pronounced gains were observed in filter application and, to a lesser extent, property extraction. This pattern indicates that the principal remaining limitation is the model's ability to apply filters accurately rather than to recognise entities.

Table 2. Comparison of accuracy and improvement percentages for two LLM prompts, evaluated in zero-shot and few-shot settings across three tasks: entity extraction, property extraction, and filter application, with full task success reported.

	Zero-shot			Few-shot		
Task	Prompt-1	Prompt-2	Improvement	Prompt-1	Prompt-2	Improvement
Extracting Entities	90%	95%	5.56%	90%	95%	5.56%
Extracting Properties	65%	85%	30.77%	85%	90%	5.88%
Applying Filters	35%	50%	42.86%	70%	85%	21.43%
Full Task Success	35%	50%	**42.86**%	70%	85%	**21.43**%

Contrary to the initial experiment, we observed a 42.86% relative improvement in zero-shot accuracy for the more detailed second prompt over the first, and a 21.43% improvement with few-shot. Few-shot alone, without

RDF, closed a substantial part of the gap to annotated performance, particularly for filter application. The highest overall accuracy was achieved when few-shot and annotations were combined, with full task success of 85% for Prompt 2. Without RDF, many property names were incorrectly specified, which hindered accurate data extraction. These results suggest that data annotation, combined with more comprehensive instructions, enables the model to generate more accurate outputs. Accordingly, the subsequent research experiments will implement Prompt-2, which provides enhanced detail.

4.2 RDF Embeddings and Prompt Augmentation

Incorporating RDF data into the prompts of LLMs like GPT presents significant challenges primarily due to the voluminous nature of RDF datasets. The principal issue arises from the economic implications of embedding large RDF datasets into LLM prompts, especially with fee-based services such as OpenAI's GPT, where costs are proportional to the number of tokens processed. Additionally, although the maximum allowable prompt size has recently expanded, it remains insufficient for embedding substantial RDF data. For instance, models like `LLAMA-3` accommodate a context length of up to 8192 tokens, which is considerably smaller than even a modest-sized RDF graph, particularly those containing observational data. To address these limitations, we have adopted the following approach:

1. Selective Data Inclusion: The definitions of sensors, their properties, and the corresponding properties each sensor monitors are the essential data for the LLM to identify entities and understand their relationships within the graph. Consequently, we omit less critical observational data from the LLM context. This selective inclusion significantly reduces the volume of RDF data while still transmitting valuable information. However, this does not imply that the entire data definition can be included, as it may still be excessively large.

2. Subgraph Generation through RDF Walks: Unlike document embeddings, which are tailored for unstructured text, RDF embeddings are specifically designed to manage structured graph data. A viable approach involves decomposing extensive RDF graphs into smaller, more manageable subgraphs. This decomposition is achieved by using RDF walks, inspired by the RDF2Vec technique [32], to create subgraphs that encapsulate each triple's directly connected nodes. Figure 4 illustrates the first subgraph resulting from the initial walk. However, RDF2Vec was found to be impractical for our specific applications due to the high computational demands associated with dynamically generating subgraphs from user queries. For example, take the query, 'What is the latest reading of Sensor A?' Within this query, the identifiable entities are 'Sensor A' and 'latest reading.' Subsequently, it becomes imperative to construct a simplified RDF graph or subgraph that accurately encapsulates the semantic structure of the user query. Instead, we decompose the RDF graph into individual triples stored in memory as an array. These triples are then iteratively processed to identify and amalgamate directly connected triples into smaller subgraph arrays. The process of constructing these subgraphs is outlined in Algorithm 1, where entities and their relationships are iteratively expanded to form meaningful substructures.

RDF Triples Example. The following RDF triples illustrate sensor-based observations related to tracking an elephant's location and a zebra's speed. These triples are structured using the SOSA ontology, which provides a framework for representing sensor data and observations.

```
1.  :obs1 sosa:hasFeatureOfInterest :elephant1
2.  :obs1 sosa:observedProperty sosa:Location
3.  :obs1 sosa:resultTime "2024-10-21T10:00:00Z"
4.  :obs1 sosa:hasResult :loc1
5.  :loc1 geo:lat "1.3521"
6.  :loc1 geo:long "103.8198"
7.  :obs2 sosa:hasFeatureOfInterest :zebra1
```

Algorithm 1 Construct RDF Subgraphs from RDF Walks

```
Input: A list of RDF triples in string format, triples[]
Output: A list of subgraphs as strings, subgraphs[]

subgraphs[] ← empty list                                   ▷ Stores all subgraphs

for each triple t in triples[] do
    entities ← empty dictionary           ▷ Track subject and object occurrences
    used ← empty dictionary                                  ▷ Track used triples
    subgraph[] ← empty list                              ▷ Stores current subgraph
    Add t to subgraph[]
    Mark index of t in used
    Split t into subject, predicate, and objects
    Increment count of subject in entities
    for each object o in objects do
        if o is not a string literal then
            Increment count of o in entities
        end if
    end for
    for each entity e in entities do
        idx[] ← findEntityIndexes(e, triples[])
        for each index i in idx[] do
            if i not in used then
                Add triple at i to subgraph[]
                Mark index i in used
            end if
        end for
    end for
    if subgraph[] is not empty then
        Add subgraph[] to subgraphs[]
    end if
end for

return subgraphs[]
```

```
8.  :obs2 sosa:observedProperty sosa:Speed
9.  :obs2 sosa:resultTime "2024-10-21T09:00:00Z"
10. :obs2 sosa:hasResult :speedResult1
11. :speedResult1 sosa:hasValue "50 km/h"
```

These triples represent two distinct observations: one monitoring an elephant's location at a specific time and another measuring the speed of a zebra.

Construction of the First Subgraph. The process of constructing RDF subgraphs begins by identifying a central entity and expanding its relationships. The steps involved in building the first subgraph are detailed below.

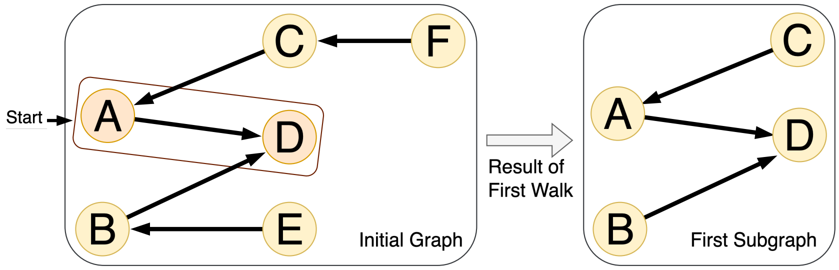


Fig. 4. An example of the first iteration of RDF Walk showing the first generated subgraph.

Step 1: Initialisation of the Subgraph. The process begins by selecting an RDF triple to serve as the starting point. In this case, the following triple is chosen:

```
:obs1 sosa:hasFeatureOfInterest :elephant1
```

Here, the subject (`:obs1`) and object (`:elephant1`) are identified as key entities. These entities are tracked in a dictionary to facilitate the identification of additional triples that reference them.

Step 2: Inclusion of Related Triples. Next, all RDF triples containing the subject `:obs1` are retrieved, as they provide further contextual information regarding the observation. These triples include:

```
:obs1 sosa:observedProperty sosa:Location
:obs1 sosa:resultTime "2024-10-21T10:00:00Z"
:obs1 sosa:hasResult :loc1
```

These triples are added to the subgraph, as they are connected by the common subject (`:obs1`). The subgraph now contains the following triples:

```
:obs1 sosa:hasFeatureOfInterest :elephant1
:obs1 sosa:observedProperty sosa:Location
:obs1 sosa:resultTime "2024-10-21T10:00:00Z"
:obs1 sosa:hasResult :loc1
```

Step 3: Expansion with Geolocation Data. To further enrich the subgraph, the object `:loc1` is examined. Since this entity appears in additional triples that specify geographical coordinates, these triples are added to the subgraph:

```
:loc1 geo:lat "1.3521"
:loc1 geo:long "103.8198"
```

The inclusion of these triples results in the finalised subgraph:

```
:obs1 sosa:hasFeatureOfInterest :elephant1
:obs1 sosa:observedProperty sosa:Location
:obs1 sosa:resultTime "2024-10-21T10:00:00Z"
:obs1 sosa:hasResult :loc1
:loc1 geo:lat "1.3521"
:loc1 geo:long "103.8198"
```

Subsequent Processing. Once the first subgraph is constructed, the algorithm proceeds to process the remaining RDF triples. Since all triples related to `:obs1` and `:loc1` have already been included, they are excluded from further iterations to prevent redundancy.

The process proceeds by identifying and processing any remaining RDF triples that reference distinct subjects and objects, thereby generating additional subgraphs as required. In this example, the algorithm subsequently processes the triples associated with `:obs2` and `:zebra1`, resulting in the construction of a second subgraph representing the zebra's observation data. This systematic approach ensures that all semantically related data are grouped into distinct subgraphs, preventing redundancy while preserving completeness.

Through this approach, decomposing RDF graphs into subgraphs ensures that related triples remain semantically linked while avoiding duplication. These subgraphs can subsequently be utilised for downstream applications, such as knowledge graph embeddings, machine learning tasks, or similarity-based analyses.

3. Vector Embedding of RDF Triples: As each RDF triple selected in this step is enriched with textual annotations, including labels and descriptive comments specific to the entity, we utilised the 'paraphrase-TinyBERT-L6-v2', a pre-trained model capable of capturing semantic similarities within textual data [31], to represent each triple as a vector. These vectors are then stored in Qdrant, an open-source vector database. As summarised in Figure 5, when a user query is received, it is similarly encoded, and a cosine similarity search is performed in Qdrant to identify the most closely related triple. Some entities in the questions are associated with disjoint subgraphs. By merging the highest-scoring entities, we significantly increase the likelihood of including all essential data in the injected RDF. Therefore, the two highest-scoring triples are identified, and a union of their corresponding subgraphs is retrieved. These subgraphs are then incorporated into the LLM prompt to ensure that all relevant data is included.

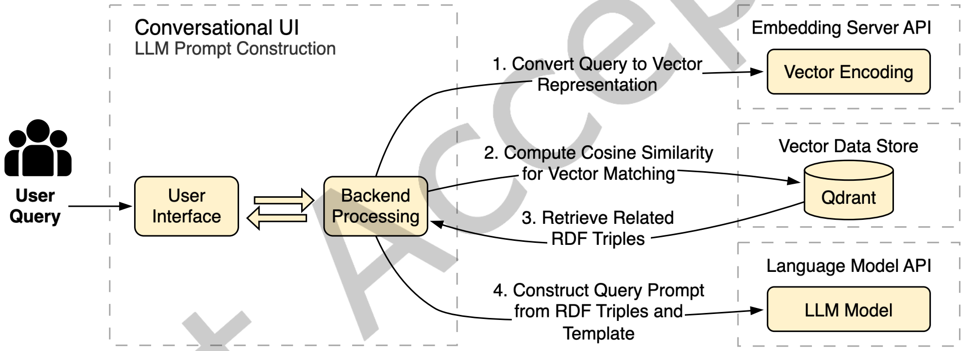

Conversational UI
LLM Prompt Construction
User Query
User Interface
Backend Processing
1. Convert Query to Vector Representation
2. Compute Cosine Similarity for Vector Matching
3. Retrieve Related RDF Triples
4. Construct Query Prompt from RDF Triples and Template
Embedding Server API
Vector Encoding
Vector Data Store
Qdrant
Language Model API
LLM Model

Fig. 5. A summary of the workflow for generating LLM prompts: It begins with the conversion of user queries into vector representations, followed by similarity calculations to identify relevant RDF triples. These triples form a subgraph that, combined with a prompt template, aids in producing context-aware prompts.

Our approach undertakes LLM-assisted reasoning over a bounded, clearly specified subset of ontological inferences by supplying, within the prompt, the RDF triples relevant to the user's query. This enables the LLM to recognise the pertinent relations and return accurate results. In the absence of such data, the prompt is under-specified (i.e., lacks context), which is liable to produce incorrect outputs. The selection of this particular pretrained embedding model was motivated by its ability to identify all related triples for the queries associated with the experiment for use cases 1 and 2. We conducted evaluations on multiple state-of-the-art pre-trained sentence embedding models available on Hugging Face, utilising the SentenceTransformers framework. These evaluations involved conducting semantic searches using Qdrant with the vector representations derived from these models. Typically, we observed an enhancement in the average accuracy across all models by 13.15%, achieved through the integration of subgraphs from the two highest-ranked indices in the similarity searches. This approach culminated in attaining 100% accuracy with our preferred model, as detailed in Table 3.

Table 3. Comparison of similarity search results using different pretrained embedding models, detailing the accuracy achieved when using only the highest scoring index and when incorporating the top two highest scoring indices. Achieving 100% accuracy for our experimental questions was possible when incorporating the top two indices.

Model Name	Top Index	Top-2 Indices
paraphrase-TinyBERT-L6-v2 [31]	**86%**	**100%**
multi-qa-mpnet-base-cos-v1	78%	90%
all-MiniLM-L6-v2	69%	86%
multi-qa-mpnet-base-dot-v1	79%	85%
hkunlp/instructor-large	75%	85%
sentence-t5-base [27]	73%	83%
all-mpnet-base-v2	65%	81%
multi-qa-MiniLM-L6-cos-v1	73%	80%
msmarco-distilbert-cos-v5 [31]	64%	79%
LaBSE	61%	79%
paraphrase-MiniLM-L6-v2 [31]	58%	76%
facebook-DPR [18]	46%	63%
paraphrase-multilingual-mpnet-base-v2 [31]	44%	55%

4.2.1 Balancing Embedding Accuracy and Latency. Subgraph generation through RDF walks raises two main issues, namely accuracy and latency. Cost is also a concern, since both latency and cost increase with prompt size. Accuracy requires that the LLM has access to sufficient context. We address this by including only data that are relevant to the prompt. Latency and cost rise when the prompt includes data that are irrelevant to the user's question. Our objective is to minimise prompt size while preserving correctness, using a lightweight approach that runs in the browser.

To this end, we developed an RDF-walk algorithm (Algorithm 1) that partitions the graph into minimal, meaningful subgraphs by isolating each entity or sensor together with the data required for it. We then evaluated performance on our question set. A first pass selected the nearest subgraph to each question using similarity search and achieved 86% coverage (Table 3). In other words, 14% of questions lacked essential context and could not be fairly evaluated. Further analysis showed that several questions referenced multiple, sometimes disjoint, entities. When we merged the two highest ranked subgraphs, coverage improved and the tests achieved 100%. This indicates that the prompt contained all necessary information and that the evaluation was not confounded by missing context. Merging additional subgraphs further increases the likelihood that all required data are present, although this comes at the cost of a larger prompt, higher latency, and higher token usage.

5 Experiment Evaluation for Use-Case 1 and 2

In this section, the effectiveness of incorporating LLMs into the existing system to improve information retrieval is evaluated, alongside an analysis of the experimental results. As explained in Section 3, the proposed approach enables the Conversational UI to answer questions that the conventional baseline cannot address [25]. Since all experimental questions are unanswerable under the baseline and would score zero on every metric, we therefore omit baseline results. The results and evaluation metrics are available at (https://github.com/i3omar/LLM-Integration-Data).

5.1 Dataset

The experiment was conducted using a private SPARQL endpoint hosting observational LD, structured in accordance with the Sensor, Observation, Sample, and Actuator (SOSA) ontology [16]. The dataset is publicly available via the GitHub repository (https://github.com/i3omar/ForestRDF) and encompasses multiple dimensions of forest observations. It comprises detailed descriptions of sensors, including their types, locations, and measurement capabilities, and records individual observations that are time-stamped and associated with the corresponding sensor. These observations represent environmental parameters such as temperature, humidity, soil moisture, and other key indicators of forest ecosystems. Furthermore, metadata concerning the sampling process is embedded within the dataset, providing essential context regarding the conditions under which the data were collected.

5.2 Selected Large Language Models

In order to rigorously evaluate our approach, we have selected a diverse range of state-of-the-art language models. Our selection primarily includes six versions of OpenAI's latest `GPT-3.5` and `GPT-4` models (https://platform.openai.com/docs/models): `GPT-4-turbo-2024-04-09`, `GPT-4o-2024-05-13`, `GPT-4-0125-preview`, `GPT-4-0613`, `GPT-3.5-turbo-0125`, and `GPT-3.5-turbo-0613`. Additionally, we integrate two open-source models into our study: `openchat-3.5-0106` [37] and `Meta-Llama-3-8B-Instruct` [2]. The former has been finetuned from the `Mistral-7B-v0.1` base model, while the latter is the most recent iteration within the Meta `LLAMA-3` series. Both open-source models were chosen due to their lightweight architecture, making them practical for real-world applications. Furthermore, all selected models have demonstrated superior performance in various NLP tasks relevant to our study. The primary objective of this selection is not to compare these models directly; rather, it is to validate our approach's effectiveness and achieve high accuracy in its application.

5.3 Experiment Setup

We have formulated a set of 40 questions designed to assess the application of LLMs across use cases 1 and 2, where the LLM was prompted with relevant RDF data (see Appendix A). These questions are structured to target different types of queries, systematically divided into four distinct groups to test the LLM's ability to interpret and respond to varied levels of query complexity:

(1) **Simple Direct Queries**: These questions involve a single, clearly identified entity within the RDF data. The purpose of this group is to assess the model's accuracy in retrieving direct information where minimal inference is required.
(2) **Complex Direct Queries**: This category encompasses queries that include multiple entities explicitly mentioned in the RDF data. Here, the focus is on evaluating the model's capacity to accurately process and respond to questions requiring direct reference to multiple data points.
(3) **Simple Indirect Queries**: These queries pertain to a single entity not directly referenced in the RDF data. Instead, they employ synonyms or alternate phrasing, challenging the model to infer and identify relevant information despite variations in terminology.
(4) **Complex Indirect Queries**: This group involves more intricate queries concerning multiple entities not explicitly stated in the data, using synonyms and diverse phrasing. This category is designed to gauge the model's reasoning and comprehension skills when interpreting questions requiring a deeper understanding and synthesis of information.

For each of these query types, we conducted six trials, divided equally between zero-shot and few-shot settings to examine the consistency of the outputs and to determine whether the model's performance improves with exposure to example queries. In the zero-shot setting, the model received no prior examples, while in the few-shot setting, it was provided with example queries to establish a baseline understanding.

Consequently, each model was tested with a total of 240 queries, comprising 120 queries in a zero-shot setting and another 120 in a few-shot setting. Notably, for Use Case 2, only zero-shot testing was conducted, amounting to 120 queries. This setup offers a robust framework to evaluate the LLM's effectiveness across both explicit and implicit query challenges, highlighting its performance in handling complex and varied data interactions.

5.4 Evaluation Metrics

To evaluate our approach, we conducted manual evaluations for both use cases and automatic validation of the JSON output for Use Case 1, applying targeted metrics to ensure practical and technical validity, as follows:

5.4.1 A. Manual Evaluation (Accuracy). The effectiveness of the LLM in enhancing entity extraction and RDF reasoning capabilities was manually evaluated through meticulous testing in practical scenarios. The output is manually classified in a binary manner as correct or incorrect. For Use Case 1, the output is deemed correct if the generated JSON output includes the correct answer and does not exhibit issues such as incorrect key naming or invalid JSON format (see Figure 10 in Appendix A). Meanwhile, for Use Case 2, the output comprises plain text that has been manually evaluated against reference data to verify the accuracy of the generated output.

5.4.2 B. Automatic Structural Evaluation. In practical settings for Use Case 1, two models can generate outputs that are both correct and compatible with the system, although one model may introduce additional keys (which remain operational). This evaluation aims to determine the extent to which each model complies with specified instructions to yield a more reliable JSON output. These adapted metrics, based on foundational work in the field as detailed by [21, 31], evaluate performance by comparing JSON outputs from LLMs to a reference answer, enabling a deeper examination of quality beyond mere accuracy. The metrics used are as follows:

1. Structural Similarity Score (StrSS). This metric uses a specific transformation and comparison methodology to measure the similarity between a reference JSON structure and the generated JSON output. Initially, both JSON objects are flattened into a format where each key represents a path that describes navigation through the original JSON structure. Subsequently, to ensure fairness in scoring, any full URI in the generated JSON is simplified to match the abbreviated form used in the reference JSON. For instance, the URI 'http://www.w3.org/ns/sosa/resultTime' is converted to 'sosa:resultTime'. This step is crucial as the scoring algorithm would otherwise incorrectly penalise the full URI format despite it being an acceptable variation. There is a lesser penalty for extra keys in the output, acknowledging that models sometimes generate correct, albeit additional, keys. This is intended to slightly diminish the match score, enabling a more refined comparison among models that more precisely limit their outputs to the expected keys. The following is the complete metric equation to evaluate the outputs:

$$\text{StrSS} = \frac{|K_{\text{ref}} \cap K_{\text{gen}}|}{|K_{\text{ref}}| + \beta \cdot (|K_{\text{gen}} - K_{\text{ref}}|)} \tag{1}$$

where K_{ref} and K_{gen} denote the sets of keys in the reference and generated JSON objects, respectively. The term $|K_{\text{ref}} \cap K_{\text{gen}}|$ quantifies the intersection, representing keys common to both sets. Conversely, $|K_{\text{gen}} - K_{\text{ref}}|$ measures the keys unique to K_{gen}. The weighting factor $\beta = 0.1$ moderates the effect of these unique keys in the scoring mechanism, reducing their impact.

***2. Semantic Similarity Score (SemSS):*.** This metric assesses the semantic similarity between the values present in the reference JSON and those in the generated JSON. To facilitate this, the values from both the reference and the generated JSON are converted into separate vector embeddings using the 'paraphrase-TinyBERT-L6-v2' model. The cosine similarity between these embeddings is then computed to evaluate their semantic closeness. The overall score is derived by averaging the cosine similarities across all pairs of embeddings corresponding to matching key values in the JSON structures. This approach quantifies the semantic alignment between the generated output and the expected reference data. The SemSS score is defined as:

$$\text{SemSS} = \begin{cases} \frac{1}{N} \sum_{i=1}^{N} \text{cos\_sim}\left(\text{emb}(v_{\text{ref},i}), \text{emb}(v_{\text{gen},i})\right) & \text{if } N > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where N is the count of keys shared between K_{ref} and K_{gen}. For the i-th matching key, $v_{\text{ref},i}$ and $v_{\text{gen},i}$ denote the associated values in the reference and generated JSONs, respectively. The function $\text{emb}(v)$ computes the embedding of value v, and $\text{cos\_sim}(a, b)$ calculates the cosine similarity between vectors a and b.

***B.3. Overall JSON Similarity Score (OJSS):*.** This score is a weighted average that combines the two distinct similarity scores: the SemSS Score and the StrSS Score. Both scores are assigned equal weight, each contributing 50% to the final calculation, denoted by a coefficient of 0.5 for each. This equal weighting underscores the balanced importance of both the structural integrity of the keys and the semantic accuracy of the values in determining the overall score. The following is the OJSS complete equation:

$$\text{OJSS} = \alpha \times \text{StrSS} + (1 - \alpha) \times \text{SemSS} \tag{3}$$

where $\alpha = 0.5$, α is a weighting factor between 0 and 1 that balances the importance of structural and semantic similarity.

5.5 Use Case 1: Results

Table 4 presents the performance of the selected LLMs in generating accurate JSON outputs, reflecting the queries employed in our research. This practical experiment relied on the system's capability to retrieve and process the generated responses accurately and to construct the queries correctly to be deemed successful. Figure 6 illustrates responses across question groups.

Zero-shot. During the zero-shot testing phase, the performance levels were notably low as most LLMs did not adhere precisely to the instructions. For instance, `LLAMA-3`, despite explicit instructions to generate solely JSON outputs, produced additional explanatory text alongside the JSON, thereby rendering the outputs impractical for further processing, culminating in a complete failure rate. In contrast, `GPT-4-turbo` achieved an accuracy of 68.3%, with a StrSS score of 71.9%, indicating a well-structured and highly practical output.

Table 4. Performance metrics comparison of LLMs in Use Case 1 for few-shot and zero-shot scenarios, showing percentages for accuracy, structural similarity (StrSS%), semantic similarity (SemSS%), and overall JSON similarity (OJSS%). Sorted by overall accuracy (OA%) in descending order.

	Few-shot					Zero-shot					
Model Name	Accuracy		StrSS	SemSS	OJSS	Accuracy		StrSS	SemSS	OJSS	OA
GPT-4-turbo-2024-04-09	87.5	±0.03	**87.4**	**84.2**	**85.8**	**68.3**	±0.04	**71.9**	**63.2**	**67.6**	**77.92**
GPT-4o-2024-05-13	**89.2**	±0.06	83.0	79.6	81.3	60.8	±0.08	69.0	60.0	64.5	75.00
GPT-4-0613	88.3	±0.03	60.8	63.2	62.0	45.8	±0.06	35.6	34.5	35.0	67.08
GPT-3.5-turbo-0613	81.7	±0.11	53.1	56.1	54.6	37.5	±0.09	4.5	4.4	4.5	59.58
GPT-4-0125-preview	74.2	±0.07	58.4	59.9	59.2	44.2	±0.06	30.3	25.8	28.1	59.17
GPT-3.5-turbo-0125	84.2	±0.07	56.6	56.5	56.5	25.0	±0.09	28.2	25.6	26.9	54.58
Meta-Llama-3-8B-Instruct	71.7	±0.05	58.8	57.7	58.3	0.0	±0.00	0.0	0.0	0.0	35.83
Openchat-3.5-0106	57.5	±0.05	55.8	53.9	54.9	5.0	±0.00	24.3	17.2	20.8	31.25

Few-shot. In the few-shot tests, there was a substantial improvement in results. LLMs utilised the provided examples as historical references, enhancing their ability to generate structured outputs, as evidenced by the StrSS scores. GPT-4o achieved the highest accuracy rate at 89.2%, followed by GPT-4-0613 and GPT-4-turbo, with scores of 88.3% and 87.5% respectively. GPT-4-turbo also scored 87.4% in StrSS, 84.2% in SemSS, and 85.8% in OJSS, demonstrating fewer errors in practical applications within web information systems, despite not being the most accurate model. Furthermore, LLAMA-3 showed a notable increase in accuracy to 71.7%, a commendable achievement for a lightweight model, illustrating greater adherence to instructions when provided with few-shot examples.

5.5.1 Use Case 1: Result Analysis. In zero-shot learning contexts where language models operate without specific prior examples, most models exhibited significant difficulties in adhering strictly to a predefined JSON template. Notably, Meta-Llama-3-8B-Instruct failed to produce the correct JSON format, generating unnecessary explanatory text despite explicit instructions to generate JSON only. Similarly, Openchat-3.5-0106 also struggled significantly with the template adherence, performing poorly in instruction following and entity extraction. Specifically, Openchat ranked as the second worst, only slightly outperforming LLAMA-3, the least effective model under these zero-shot conditions. Contrarily, all models, except for GPT-4-turbo-2024-04-09 and GPT-4o-2024-05-13, demonstrated unreliability in such scenarios. On the other hand, GPT-4-turbo-2024-04-09 and GPT-4o-2024-05-13 showcased superior adherence to instructions, with StrSS scores of 71.9% and 69%, respectively. These scores reflect a greater capacity in preserving the structural integrity of the output formats, underscoring their potential dependability for applications requiring strict adherence to input prompts.

Transition to few-shot scenarios resulted in significant improvements, as evidenced by the enhanced overall accuracy and StrSS scores of the models. Remarkably, StrSS scores for LLAMA-3 and Openchat increased from 0% and 24.3% to 58.8% and 55.8%, respectively, bringing them back into the competition. This improvement indicates that LLMs become more effective as entity extractors in practical settings with the introduction of few-shot learning. For example, four models reached 100% accuracy for questions in Group 1, with Openchat achieving the lowest at 76.7%, which remains a strong performance. The most challenging questions in Groups 1 and 3 involved assigning the direction property to "north," which no model could resolve in zero-shot settings, yet even the lowest-performing model managed some success during few-shot learning. Responses that were structurally correct with nearly accurate angle values were considered acceptable, showing that the system was functioning correctly, and minor inaccuracies could be manually adjusted.

In more complex tasks involving RDF data analysis, Tasks 2.8 and 4.8 were particularly challenging, requiring models to analyse RDF data and identify sensors monitoring "Horizontal Dilution of Precision." While GPT-4-turbo-2024-04-09, GPT-4o-2024-05-13, and GPT-4-0125-preview performed commendably on Task 2.8, they encountered difficulties with Task 4.8, with GPT-4o-2024-05-13 succeeding in only one attempt. Intriguingly, GPT-3.5-turbo-0613 exhibited better performance on Task 4.8 than 2.8, despite the latter being ostensibly simpler. Surprisingly, GPT-3.5-turbo-0125 excelled in both tasks across all attempts, demonstrating superior reasoning with RDF data. Other models displayed inconsistent performance, indicating limited reliability in practical scenarios for such tasks.

This task-based performance evaluation highlights the variability in how different models handle specific task requirements in zero-shot versus few-shot scenarios. While some models show potential for practical applications by adhering closely to structured outputs and adeptly handling linguistic variations, others struggle with consistency and reliability, particularly when confronted with complex questions or data structures. These findings underscore the necessity for continued advancements in model training and refinement to enhance their practical applicability and reliability in real-world settings.

5.5.2 Entity Extraction Challenges. The implementation of LLMs as advanced entity extractors in practical applications introduces several risks and challenges. Given that the output is entirely hidden from the user, it

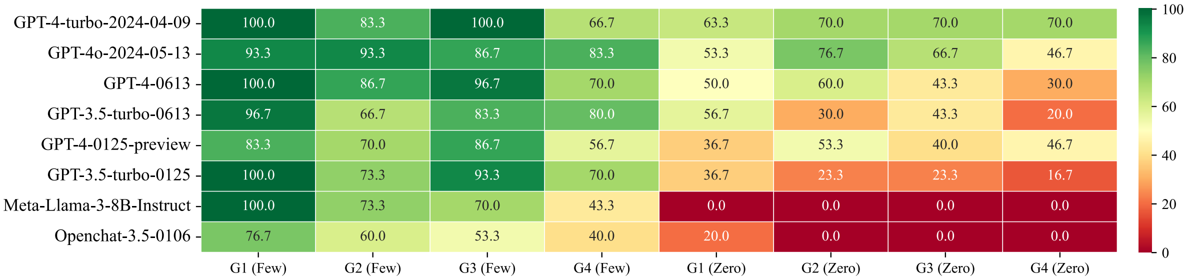


Fig. 6. Accuracy of selected LLMs for Use Case 1, categorised by question group (G) and test type (few-shot or zero-shot). Green indicates higher accuracy, while red shows lower accuracy.

must be managed through the tool as part of the process to retrieve the correct entities and apply appropriate filters. Consequently, the output must be well-structured to function correctly with the tool; otherwise, it may cause parsing errors and be disregarded. To address this, the correct template was embedded within the prompt, and the model was instructed to adhere strictly to the template and return output solely in JSON format.

However, despite explicit instructions to return only JSON output, some GPT models enclose their outputs with Markdown conventions, specifically GitHub Flavored Markdown (GFM). For instance, in zero-shot settings, `gpt-4o-2024-05-13` consistently wraps text with triple backticks and a JSON specifier (` ```json ...``` `). This behaviour also occurs with `gpt-4-0125-preview` and `gpt-3.5-turbo-0125` at lower percentages of 74.2% and 5.8%, respectively.

This formatting is widely used to denote code blocks and enable syntax highlighting in text-based environments. GPT models, trained on vast amounts of internet text where this convention is prevalent, have learned to mimic this behaviour when generating code or structured data like JSON. Understanding this origin helps explain why the models include such formatting. In automated systems where the output is parsed programmatically, these additional characters can cause parsing errors. Our system mitigates this issue by preprocessing the generated text to strip out any unnecessary formatting, ensuring that the JSON is correctly parsed for entity extraction. This problem has been reduced to 0% using few-shot learning (see Table 5).

In addition, in zero-shot settings, some outputs contained the correct answers, but the JSON format itself was incorrect and could not be processed correctly within the system, rendering it impractical and marked as failed, especially for the open-source models. For example, in zero-shot settings, `Meta-Llama-3-8B-Instruct`, `openchat-3.5-0106`, and `gpt-4-0613` returned valid outputs at rates of 0.0%, 72.5%, and 96.7%, respectively. However, in the few-shot setting, the lowest valid output rate was 96.7% for `Meta-Llama-3-8B-Instruct`, which is a significant improvement in terms of practicality. Ideally, we aim for 100% valid outputs, as practical applications cannot extract information from invalid outputs.

Another challenge was the occurrence of empty JSON outputs, as some models continued to return empty JSON responses. Specifically, in zero-shot settings, `gpt-4-0125-preview` produced empty JSON outputs in 18.3% of all responses, accounting for 32.77% of the total incorrect responses. In the few-shot setting, not much improvement was observed, with 16.7% being empty JSON responses. For `gpt-3.5-turbo-0613`, the percentage was 4.2% and decreased to 0% in few-shot settings. However, in experiments with `gpt-4-turbo-2024-04-09`, the percentage of empty JSON outputs increased from 5% in zero-shot settings to 6.67% in few-shot settings, contrary to expectations of improvement. This suggests that few-shot learning has limited impact on addressing this particular challenge. Table 5 presents the percentage of valid JSON outputs after removing the GFM, alongside the percentage of Markdown occurrences, the percentage of empty JSON outputs, and the overall average percentage of valid JSON.

Table 5. Comparison of Few-shot and Zero-shot Performance on JSON Output Validity across the selected language models, including: the percentage of valid JSON output after removing any Markdown (Valid-JSON), the percentage of outputs containing Markdown tags that impact JSON validity (Markdown), the percentage of responses returning empty JSON (Empty), and the overall average percentage of valid JSON.

	Few-shot			Zero-shot			
Model Name	Valid-JSON	Markdown	Empty	Valid-JSON	Markdown	Empty	Overall
GPT-3.5-turbo-0613	**100.0%**	**0.0%**	**0.0%**	**100.0%**	**0.0%**	4.2%	**100.0%**
GPT-4-0125-preview	**100.0%**	**0.0%**	16.7%	**100.0%**	74.2%	18.3%	**100.0%**
GPT-3.5-turbo-0125	**100.0%**	**0.0%**	**0.0%**	99.2%	5.8%	**0.0%**	99.6%
GPT-4o-2024-05-13	99.2%	**0.0%**	**0.0%**	**100.0%**	100.0%	**0.0%**	99.6%
GPT-4-turbo-2024-04-09	99.2%	**0.0%**	6.7%	**100.0%**	**0.0%**	5.0%	99.6%
GPT-4-0613	99.2%	**0.0%**	**0.0%**	96.7%	**0.0%**	**0.0%**	97.9%
Openchat-3.5-0106	97.5%	**0.0%**	**0.0%**	72.5%	**0.0%**	**0.0%**	85.0%
Meta-Llama-3-8B-Instruct	96.7%	**0.0%**	**0.0%**	0.0%	**0.0%**	**0.0%**	48.3%

5.6 Use Case 2: Results

The use of LLMs for reasoning over RDF data to answer queries related to the data schema has proven to be highly effective. Employing the `GPT-4-turbo` model, we achieved an outstanding accuracy rate of 100% (see Table 6). As seen in Figure 7, most models, including open-source ones, demonstrated strong performance. However, the `GPT-3.5-turbo-0613` and `Openchat-3.5-0106` models achieved lower accuracy rates of 74.17% and 73.33%, respectively. In particular, `GPT-3.5-turbo-0613` exhibited greater difficulty with tasks that required an understanding of RDF relationships.

Table 6. Comparative performance metrics of LLMs in Use Case 2, showing percentages for accuracy and responses containing RDF snippets (Snippet).

Model Name	Accuracy(%)		Snippet(%)
GPT-4-turbo-2024-04-09	**100.0**	±0.00	0.00
GPT-4-0613	98.33	±0.03	1.67
GPT-4o-2024-05-13	98.33	±0.01	20.83
GPT-4-0125-preview	96.67	±0.04	4.17
GPT-3.5-turbo-0125	90.83	±0.03	0.00
Meta-Llama-3-8B-Instruct	88.33	±0.03	24.17
GPT-3.5-turbo-0613	74.17	±0.07	0.00
Openchat-3.5-0106	73.33	±0.10	3.33

The capacity of these models to interpret RDF relationships varies; for instance, the Openchat model exhibited some reasoning abilities but struggled with comparisons between RDF entities and occasionally generated responses not present in the RDF data. In contrast, the `LLAMA-3` model demonstrated more substantial reasoning capabilities, which is notable for its smaller size. Overall, `GPT-4` models have shown significant improvements in reasoning over RDF data, suggesting a higher level of training for such tasks.

Moreover, we observed that some models embed RDF snippets in their responses, reducing readability and diminishing accessibility, particularly when the data is meant to remain hidden from the user, thus adding to the confusion. As shown in Table 6, the highest rates of RDF snippet inclusion were observed in the responses from `LLAMA-3` and `GPT-4o`, amounting to 24.17% and 20.83% respectively.

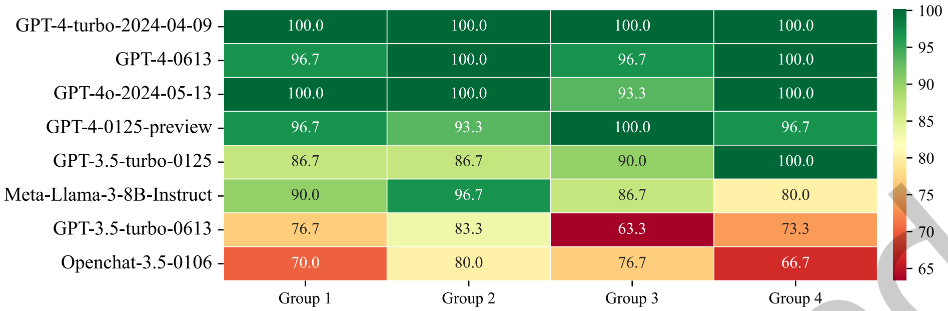


Fig. 7. Accuracy of selected LLMs for Use Case 2 questions, categorised by group number. Higher accuracy is indicated in green, while lower accuracy is shown in red.

5.6.1 Analysis of Error Patterns. While most models performed well in the given tasks, the observed error patterns highlight differences in their capabilities. The tasks referenced in this analysis correspond to the questions presented in the Appendix A, specifically in Tables 14 to 17. Some models omitted key information, while others generated entirely unrelated responses. Additionally, certain models produced errors in some instances while generating correct answers in others, whereas some consistently failed across all three attempts. The following sections discuss these error patterns, which are also summarised in Table 7.

One of the most prevalent errors identified is **Sensor Name Omission**, which refers to instances where models failed to include the correct sensor names as specified in the RDF data. This omission does not necessarily indicate that the generated response was incorrect but rather that a crucial piece of information about the sensor was missing. For example, in Question 5.9, concerning VHF chip tracking, the correct answer explicitly identifies the "Juling" sensor as being associated with the Python VHF chip. However, `GPT-4-0125-preview`, `GPT-4-0613`, `GPT-3.5-turbo-0125`, `GPT-3.5-turbo-0613`, and `Openchat` repeatedly failed to include this identifier. In particular, `GPT-3.5-turbo-0125`, `GPT-3.5-turbo-0613`, and `Openchat` omitted "Juling" in all three attempts. Similarly, `GPT-4-0125-preview` failed to identify the camera trap sensors in Question 8.3 in two out of three attempts. This pattern suggests that while some models, particularly the `GPT-4` variants, generally include most of the required details, they remain prone to occasional lapses in extracting specific information from RDF triplestores.

The most frequently occurring error was **Observable Property Name** Omission, affecting six out of eight models. This error type refers to instances where models failed to include critical observable property names from the RDF data. In Questions 5.5 and 7.5, which concern tree age estimation, `GPT-4-0613` omitted the correct property name once, while `GPT-3.5-turbo-0125` and `GPT-3.5-turbo-0613` omitted it multiple times, with `GPT-3.5-turbo-0613` failing to include it five times across the two questions. Additionally, in Question 7.8, the omission of the external temperature property name was observed across multiple models. Notably, `GPT-3.5-turbo-0613` and `Openchat` omitted this property in a manner that compromised the integrity of the responses by stripping away essential RDF references that underpin the factual accuracy of the answers.

Another observed error type is **Sensor Misattribution**, which is characterised by the incorrect assignment of sensor functionalities. In particular, `LLAMA-3` misattributed the VHF chip in Questions 5.9 and 7.9 in all six attempts, incorrectly stating that it was attached to elephants rather than pythons. This misattribution not only misinterprets the RDF data but also results in a fundamental misunderstanding of the sensor's intended function.

Closely related to Sensor Misattribution is **Observable Property Misattribution**, which occurs when a model assigns an incorrect property to a sensor or confuses one observable property with another. For example, in comparative questions regarding the observable properties of elephant collars, `LLAMA-3` incorrectly swapped the PDOP and HDOP values between different collars in Question 6.1. Similarly, in Questions 7.4 and 8.8, `Openchat` occasionally misassigned the external temperature property to inappropriate sensors. While this type of error was less frequent than omission errors, it nonetheless indicates significant challenges in accurately mapping RDF-derived observable properties to the appropriate sensor contexts.

Despite strict instruction constraints, some models generated responses that were not based on the RDF data provided in the prompt. In this context, we refer to this error pattern as **Hallucination**, which does not imply that the generated responses were nonsensical. Instead, it describes instances where the model produced responses based solely on its pre-trained knowledge while disregarding the RDF data and instructions provided in the prompt. This phenomenon was particularly evident in complex or indirect questions, such as Questions 7.9 and 7.10, where `GPT-3.5-turbo-0613` and `Openchat` produced answers containing fabricated details. In the case of `GPT-3.5-turbo-0613`, all three attempts at answering questions related to telemetry device tracking and soil sensor measurements resulted in responses that were inconsistent with the RDF data. Similarly, `Openchat` produced several hallucinated responses, undermining the reliability of RDF-based data analysis.

Another notable issue is the **Failure to Compare**, which reflects the inability of some models to accurately highlight differences between RDF elements when explicitly required by the question. Comparative questions, such as Question 6.1, which asks for differences in observable properties between elephant collars, proved particularly challenging. Models such as `GPT-3.5-turbo-0125` and `Openchat` struggled in this regard, often generating responses that either omitted essential comparative analysis or contained erroneous comparisons.

Finally, the error category **Providing Incorrect Information** encompasses all instances where models produced factually inaccurate responses. For example, `GPT-3.5-turbo-0125` misrepresented warning levels in river sensor data in Question 2.10. Additionally, `GPT-3.5-turbo-0613` repeatedly provided incorrect details across multiple questions, including inaccurate descriptions of sensor properties, particularly those concerning elephant sensor characteristics. Notably, 43.33% of the total errors in `GPT-3.5-turbo-0613` fell into this category, highlighting significant challenges in ensuring factual accuracy and fidelity to the RDF data.

These errors have revealed varying capabilities among the models examined. The `GPT-4` variants appear to possess a more robust understanding of the relationships between RDF elements, which suggests that RDF data may have been incorporated into their training. While `GPT-3.5-turbo-0125` and `LLAMA-3` generally performed well, `GPT-3.5-turbo-0125` exhibited minor difficulties with the naming of RDF elements, whereas `LLAMA-3` predominantly encountered issues related to misattribution errors. In contrast, models such as `GPT-3.5-turbo-0613` and `Openchat` were more susceptible to hallucination and factual inaccuracies, implying that these models treat RDF data more as unstructured text than as interrelated entities. Although they may function adequately in practice, their overall accuracy is comparatively less reliable.

5.7 Approach Scalability

We extend a published tool for extracting information from observational linked data. The tool has been applied to two datasets that use the SOSA and FOO ontologies, and it allows users to customise sensor discovery when source data are not standardised to its defaults [25]. It is not, however, expressly designed to operate across arbitrary, out-of-domain triplestores.

Our contribution is a scalable augmentation that couples this tool with a general-purpose LLM that had no prior exposure to our dataset, which was unpublished at the time. We index each dataset's RDF schema as embeddings in a vector store and, at query time, retrieve the most relevant fragments to inject into the prompt. This enables the LLM to reason over the appropriate vocabulary without any per-dataset training. To promote generality, we

Table 7. Error Distribution by Model. Each cell shows the number of errors in a given category. Cell background color indicates performance: lower error counts are in green (better) and higher counts in red (worse). Models are sorted in ascending order by total errors. Error categories are: (1) Sensor Name Omission, (2) Observable Property Name Omission, (3) Sensor Misattribution, (4) Observable Property Misattribution, (5) Hallucination, (6) Failure to Compare, and (7) Providing Incorrect Information.

Model Name	Sensor Name	Property Name	Sensor Misattr.	Prop. Misattr.	Hallucination	Compare	Incorrect Info.
GPT-4-turbo-2024-04-09	0	0	0	0	0	0	0
GPT-4-0613	1	1	0	0	0	0	0
GPT-4o-2024-05-13	0	2	0	0	0	0	0
GPT-4-0125-preview	3	0	0	0	0	0	0
GPT-3.5-turbo-0125	3	4	0	0	0	2	3
Meta-Llama-3-8B-Instruct	0	4	6	1	0	3	0
GPT-3.5-turbo-0613	3	8	0	0	6	0	13
Openchat-3.5-0106	5	3	1	2	9	6	6

provide only the minimal schema context required. We construct local subgraphs using an RDF-walk algorithm, embed the resulting triples for similarity search, and inject the top-ranked fragments into the prompt. This keeps the method ontology-aware and retraining-free. In practice, the LLM identifies both explicit and implicit entities and links them to their corresponding URIs, allowing the same pipeline to operate across heterogeneous schemas.

Accordingly, the results indicate that that LLMs, when grounded in retrieved RDF, function as effective advanced entity extractors and materially improve the baseline tool without domain-specific training or fine-tuning. Because the injected RDF is not domain-exclusive, the approach should transfer to other domains, even if the underlying tool is not itself fully domain-agnostic. This architecture shifts effort from model training to lightweight schema indexing, a one-off step that supports incremental adoption at scale.

6 Experiment Evaluation for Use-Case 3

This section will explore the evaluation of Use Case 3, which involves assessing the effectiveness of LLMs to assist wildlife researchers with general inquiries. Unlike the first two use cases, this scenario presents unique challenges because the system entirely depends on the LLM's training to deliver accurate responses. The accuracy of these responses in practical settings necessitates human evaluation, specifically by subject-matter experts (SMEs) engaged in wildlife research.

6.1 Experiment Setup

The following steps outline the experimental setup employed for the evaluation of Use Case 3 (see Appendix B). All data related to these steps is available at (https://github.com/i3omar/LLM-Integration-Data).

Step 1: Question Formulation and Categorisation. We organised a session with a SME who possesses over 20 years of experience in wildlife research to formulate, validate, and categorise more than 40 questions. This process involved using colour-coding and tagging techniques with cards and labels. We identified approximately 13 categories for these questions, encompassing topics such as ethics, sensors, tracking, implications, trees, camera traps, and general technological inquiries (see Figure 8).

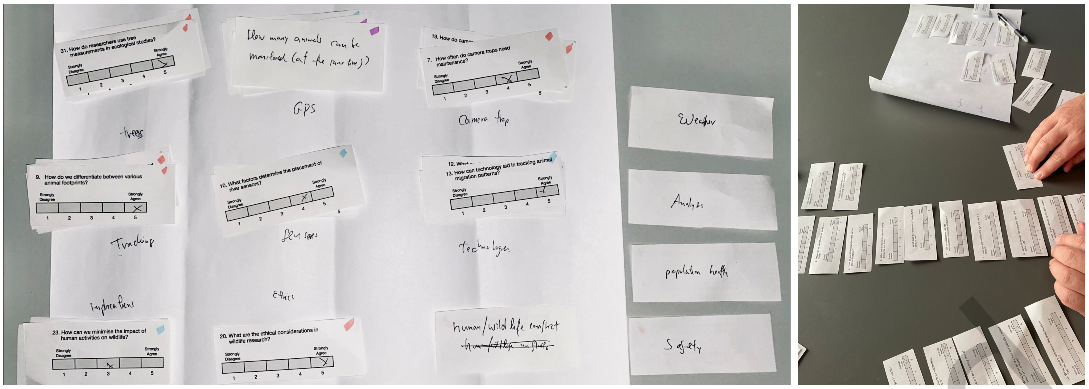

31. How do researchers use tree measurements in ecological studies?
Strongly Disagree
Strongly Agree
1
2
3
4
5
How many animals can be monitored (at the same time)?
GPS
19. How do came...
7. How often do camera traps need maintenance?
Camera trap
trees
Weather
9. How do we differentiate between various animal footprints?
10. What factors determine the placement of river sensors?
12. What ...
13. How can technology aid in tracking animal migration patterns?
Analysis
Tracking
Technologies
population health
Ethics
23. How can we minimise the impact of human activities on wildlife?
20. What are the ethical considerations in wildlife research?
human/wild life conflict
Safety

Fig. 8. Initial step in the experimental setup: Subject Matter Expert (SME) formulating and categorising the question for Use Case 3.

Step 2: Collection of Reference Responses. Subsequently, we selected the top 20 questions from the initial set and forwarded them to six experts, whose expertise spans from 2 to 30 years, in order to gather the most comprehensive answers. We now possess multiple responses for the same questions, which have been further validated by additional experts. Following an assessment by our initial SME, we have chosen the most accurate responses to serve as our reference answers for these questions.

Step 3: Generation of LLM Responses. In this phase, we utilised the tool to pose the selected 20 questions to the model, alternating among all the models we had previously employed. We meticulously logged all responses for comparison and for use in our subsequent human evaluation phase.

Step 4: Identifying LLMs for Human Evaluation. Since it is impractical to ask participants to evaluate nine responses (eight from LLMs and one Expert-crafted response) for comparison on the same questions, we selected three representative models from those used in the previous two use cases. To inform this selection, we conducted automatic evaluations to assess the overall performance of all models against our collected reference responses. Specifically, we computed the embedding-based cosine similarity between each model's responses and the reference responses, using the average cosine similarity score as a benchmark. Additionally, we calculated the BERTScore [41] for each response to provide additional insight.

As shown in Table 8, the evaluation results indicated a slight difference in performance among the models, with GPT-4 models demonstrating a marginally better overall performance. As a result, we selected `GPT-4-turbo-2024-04-09`, which achieved the highest performance in our prior experiment. Among the GPT-3.5 models, we chose `GPT-3.5-turbo-0125` due to its superior BERTScore. Lastly, from the open-source models, we selected `Llama-3-8B-Instruct`, as it demonstrated strong performance in the previous experiment and achieved slightly higher cosine similarity and BERTScore compared to the other open-source model, `Openchat`.

Step 5: Assessment of Responses. To evaluate these responses, we assessed four key dimensions of text quality: readability, correctness, coherence, and engagement. **Readability** required participants to assess how easily the text could be understood from their perspective [9, 20, 36]. **Correctness** involved evaluating the accuracy of the responses, specifically whether they reflected factual information and aligned with rational judgment based on the participants' experience [6, 9]. **Coherence** focused on the degree to which the responses remained consistent with and relevant to the original questions [36]. Lastly, **engagement** measured how compelling and interesting

Table 8. The results of the automatic evaluation, where LLMs responses were compared to expert responses, which served as the reference. (Models highlighted in bold are the ones selected for human evaluation)

Model Name	Cosine Similarity	BERTScore
GPT-4o-2024-05-13	0.706 ±0.12	0.867
GPT-4-turbo-2024-04-09	0.702 ±0.13	0.862
GPT-4-0125-preview	0.701 ±0.12	0.852
GPT-3.5-turbo-0125	0.700 ±0.12	0.882
GPT-3.5-turbo-0613	0.697 ±0.13	0.864
GPT-4-0613	0.696 ±0.14	0.873
Meta-Llama-3-8B-Instruct	0.682 ±0.13	0.858
Openchat-3.5-0106	0.665 ±0.11	0.859

participants found the text [36, 38]. These dimensions provided a comprehensive framework for assessing the overall effectiveness and quality of the responses.

We prepared two main questionnaires and selected the best 16 questions, eliminating four that had similar answers to enhance diversity. Each participant was assigned eight questions, with each question accompanied by four responses. This resulted in a total of 32 responses for evaluation by each participant. As participants were required to assess each response based on the four text quality dimensions outlined above, each participant provided a total of 128 ratings. To minimise potential bias, participants evaluated the responses without knowing whether they were generated by humans or LLMs. Additionally, the order of responses was randomised to prevent participants from detecting patterns or developing preferences.

6.2 Study Participants

In this study, we recruited 20 participants, all of whom hold a certificate in bioscience and have experience in wildlife research (see Table 9). The participants were evenly assigned to either Questionnaire A or B. Importantly, they were unaware of whether the responses they evaluated were generated by humans or large language models (LLMs), and they had no knowledge of the study's overall design. Prior to rating the responses, participants were introduced to the rating mechanism and provided with a detailed explanation of the meaning of each rating category. Furthermore, they were explicitly instructed not to compare responses to one another but rather to assess each response independently. This directive ensured that participants could freely assign ratings based on their individual judgements, regardless of whether they deemed all responses to be of high or low quality. Additionally, they were informed that the responses had been randomised to prevent any potential biases.

6.3 Use-Case 3: Results

The evaluation aimed to assess the performance of LLMs in responding to general wildlife research inquiries to acquire knowledge. Rather than determining whether LLMs outperform Expert-produced responses or vice versa, the study measured their performance relative to the ratings Expert-produced responses collected, which served as a baseline. The null hypothesis posited no significant difference in mean scores among the evaluated groups (GPT-3.5, GPT-4, LLAMA-3, and Expert-produced outputs) across four key evaluation metrics: Readability, Correctness, Coherence, and Engagement. An ANOVA test was conducted, followed by Tukey's Honest Significant Difference (Tukey HSD) post hoc test, to determine pairwise group differences (see Table 19 and Table 20 in Appendix B for further details).

The ANOVA revealed statistically significant differences across all evaluated categories. In addition, the subsequent Tukey HSD analyses provided robust evidence for rejecting the null hypothesis in multiple pairwise

Table 9. This table presents the demographic and technical background of participants involved in the Use-Case 3 evaluation, categorised by age group, questionnaire version used (A or B), language proficiency, educational qualification, academic major, years of experience in wildlife research, and self-reported technical skills. The technical skills were rated on a five-point scale, where a score of 1 indicated complete unfamiliarity, while a score of 5 denoted a high degree of proficiency.

#	Age	Version	Language	Edu	Major	Experience	Tech. Skills
P01	25-34	A	Native	MSc	Molecular Ecology	4-6 years	2
P02	25-34	A	Native	MSc	Biosciences	4-6 years	3
P03	18-24	A	Native	BSc	Zoology	1-3 years	2
P04	35-44	A	Native	MSc	Biosciences	4-6 years	2
P05	25-34	A	Native	MSc	Ecology	7-9 years	3
P06	25-34	A	Native	PhD	Molecular Ecology	4-6 years	1
P07	55-64	A	Fluent	PhD	Chemical Ecology	25+ years	2
P08	25-34	A	Native	PhD	Genetics	1-3 years	3
P09	25-34	A	Native	BSc	Zoology	1-3 years	2
P10	35-44	A	Fluent	MSc	Zoology	1-3 years	3
P11	25-34	B	Native	MSc	Zoology	4-6 years	2
P12	18-24	B	Native	MSc	Genetics	1-3 years	1
P13	18-24	B	Native	BSc	Biosciences	1-3 years	4
P14	25-34	B	Native	BSc	Biosciences	7-9 years	1
P15	25-34	B	Native	BSc	Bioveterinary Sciences	7-9 years	4
P16	18-24	B	Native	BSc	Biological sciences	1-3 years	2
P17	25-34	B	Native	BSc	Ecotoxicology	1-3 years	2
P18	45-54	B	Native	PhD	Ecology	25+ years	2
P19	18-24	B	Native	BSc	Biosciences	1-3 years	2
P20	18-24	B	Native	BSc	Zoology	1-3 years	2

comparisons, notably between responses produced by human experts and those generated by LLMs, as well as among specific LLM models (e.g., GPT-3.5 versus GPT-4). The use of adjusted p-values (p-adj) further ensured statistical rigour by controlling for the inflation of Type I error in multiple comparisons. Figure 9 shows the distribution of user ratings. The following paragraphs offer a detailed analysis for each category.

Readability: The readability ratings reveal a robust difference between LLM-generated and Expert-produced responses. The mean scores for GPT-4, GPT-3.5, and LLAMA-3 were 4.35, 3.92, and 4.38 respectively, compared to a significantly lower mean of 3.42 for Expert responses. Post-hoc comparisons using Tukey's HSD indicated that there was no significant difference between GPT-4 and LLAMA-3 (mean difference ≈ 0.0312, $p = 0.997$), whereas GPT-3.5 was rated significantly lower than both GPT-4 (mean difference = 0.4312, $p = 0.023$) and LLAMA-3 (mean difference = 0.4625, $p = 0.0127$). Moreover, Expert responses were significantly less readable than those produced by GPT-4 (mean difference = -0.9312, $p < 0.001$) and LLAMA-3 (mean difference = -0.9625, $p < 0.001$), thereby underscoring the superior clarity achieved by the language models.

Correctness: The evaluation of correctness scores exhibits a similar trend, with LLM responses outperforming Expert-produced content. Mean correctness ratings for GPT-4, GPT-3.5, and LLAMA-3 were 4.39, 4.03, and 4.44 respectively, while Expert responses attained a mean score of 3.30. Tukey's HSD tests revealed no significant difference between GPT-4 and LLAMA-3 (mean difference = 0.05, $p = 0.9865$). However, both GPT-4 and LLAMA-3 outperformed Expert responses significantly, with mean differences of -1.0875 ($p < 0.001$) and 1.1375 ($p < 0.001$)

respectively. Although the difference between GPT-3.5 and GPT-4 (mean difference = 0.3625) did not reach conventional significance (p = 0.0752), GPT-3.5 still demonstrated significantly better correctness than the Expert-produced responses.

Coherence: In terms of coherence, the language models again outperformed the Expert-produced responses markedly. The mean coherence scores for GPT-4, GPT-3.5, and LLAMA-3 were 4.31, 4.01, and 4.34 respectively, whereas the Expert responses achieved a lower mean of 3.39. Post-hoc comparisons showed that there was no significant difference between GPT-4 and LLAMA-3 (mean difference = 0.0312, p = 0.996), while Expert responses were significantly less coherent than those produced by GPT-4 (mean difference = −0.9188, $p < 0.001$) and GPT-3.5 (mean difference = −0.6188, p = 0.0002). The difference between GPT-3.5 and LLAMA-3 (mean difference = 0.3312, p = 0.0899) approached significance but did not meet the conventional threshold, thereby indicating that the top LLMs maintained comparably high levels of coherence.

Engagement: Engagement scores exhibited the most pronounced differences between LLM-generated and Expert-produced responses. The mean ratings were 4.05 for GPT-4, 3.53 for GPT-3.5, and 4.23 for LLAMA-3, while Expert responses scored a notably lower mean of 2.56. The ANOVA for engagement indicated highly significant differences among the groups ($p = 1.95\times10^{-15}$). Tukey's HSD analysis revealed that GPT-3.5 was rated significantly lower than both GPT-4 (mean difference = 0.5187, p = 0.0156) and LLAMA-3 (mean difference = 0.6937, p = 0.0006). Additionally, Expert responses were markedly less engaging than those from GPT-4 (mean difference = −1.4938, $p < 0.001$) and LLAMA-3 (mean difference = −1.6687, $p < 0.001$). These results highlight that, while GPT-4 and LLAMA-3 exhibit strong engagement, GPT-3.5, though superior to Expert responses, does not achieve the same level of engagement as the more advanced models.

Qualitative Feedback

The participants provided qualitative feedback that corroborates the quantitative findings and reinforces the patterns observed in the statistical analysis. Several participants commented on the difficulty of extracting information from dense blocks of text in contrast to structured responses. Bullet points, numbered lists, and clear formatting were frequently noted as factors that enhanced readability and comprehension. In particular, GPT-4 responses were often highlighted for their superior formatting. One participant stated, "The more block text formatted responses were much more difficult to read and extract the relevant information from. Whilst the listed/bulleted responses were much easier to gather information from at a glance."

Similarly, another remarked, "Providing the correct terminology in the answer is more relevant… Breaking down answers improves readability." Such comments align with the statistical results, which indicated that LLM-generated responses, especially those from GPT-4 and LLAMA-3, were rated significantly higher than Expert responses in terms of readability and coherence. Another participant noted, "…it is also nice to have an explanation of each point, especially when unfamiliar/technical terms are used." These insights suggest that the structured presentation of complex information facilitates easier scanning and comprehension, thereby supporting the models' superior readability and coherence scores.

In terms of correctness, several comments indicated that while some responses were highly informative, others contained unnecessary detail or omitted specific key terms. One participant commented, "Some of the answers were almost over-informative, and the question asked would not have required such a long and detailed answer but were still excellent/good in terms of the rank-able attributes." Other participants highlighted issues regarding completeness and clarity; for example, one noted, "The first response, whilst correct, was very basic. This may be useful in some applications but does not provide the reader with much information." Additionally, some participants found it challenging to evaluate correctness when responses were overly detailed.

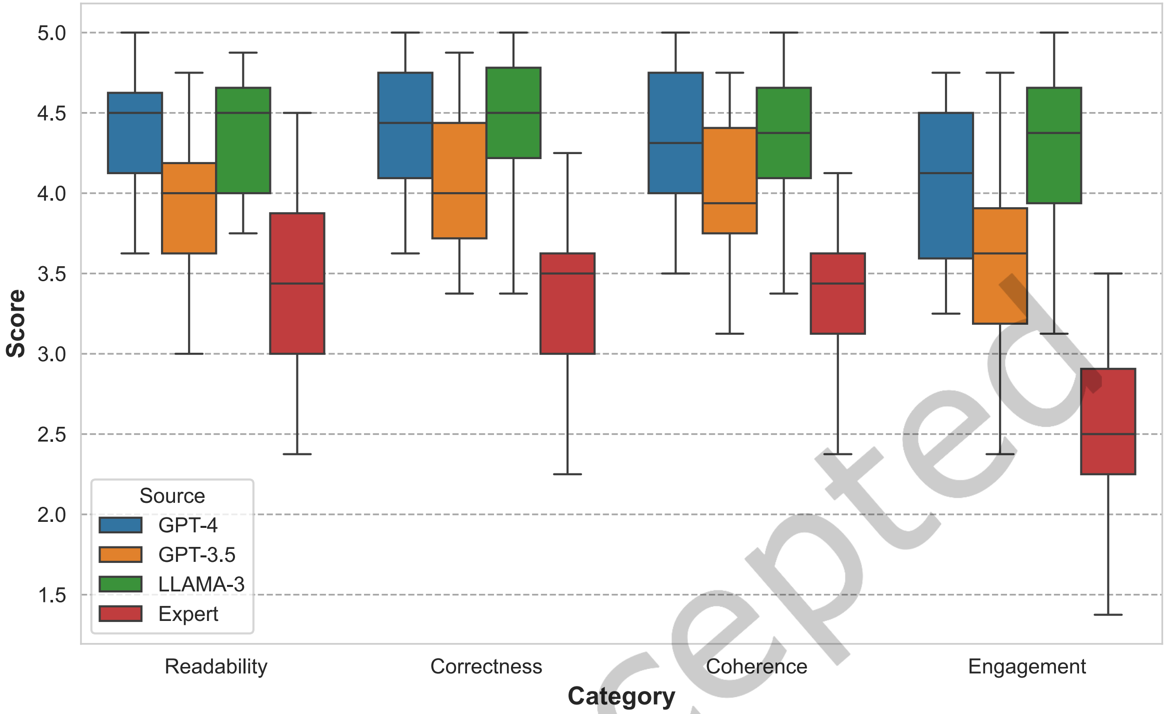


Fig. 9. This boxplot presents the distribution of user ratings for generated responses from GPT-4-turbo-2024-04-09, GPT-3.5-turbo-0125, Meta-Llama-3-8B-Instruct, and Human Experts across four evaluation categories: Readability, Correctness, Coherence, and Engagement. The central line within each box represents the median rating, while the interquartile range (IQR) denotes the middle 50% of the data. Whiskers extend to 1.5 times the IQR, summarising the variability within each category. This visualisation enables a comparative assessment of the perceived quality of responses across different evaluation criteria.

Engagement was another frequently discussed aspect. Many participants expressed a preference for responses that were both informative and engaging, with the inclusion of examples, explanations, and relevant elaborations contributing to sustained interest. One participant emphasised the importance of additional context, stating, “I found responses that gave some extra context (such as an example of something they were talking about) were more engaging than those that matter-of-factly stated their answer.” Another participant reinforced this point by asserting, “Responses where further resources were suggested are especially helpful.” Several participants also underscored the importance of formatting in enhancing engagement, with one remarking, “In my opinion, the responses were much easier to read when the main point was written in bold with additional information stated rather than being written in a paragraph, which sometimes lacked the extra information needed.” These qualitative insights support the statistical finding that LLAMA-3, which provided more resources and examples, achieved the highest engagement score.

Overall, the participant feedback strongly corroborates the quantitative results, offering robust qualitative validation of the observed statistical differences. The consistent preference for well-structured, readable, and engaging responses further substantiates the superior performance of GPT-4 and LLAMA-3 across all evaluated categories.

6.3.1 Final Remarks on Results. The most notable finding of this investigation is that all three LLMs surpassed human experts in addressing wildlife-related queries. This suggests that participants perceived the AI-generated responses as not only more refined but also more accurate and engaging compared to those produced by human experts. Among the LLMs evaluated, LLAMA-3 and GPT-4 received comparable ratings, whereas GPT-3.5 ranked slightly lower, albeit still outperforming the Expert-produced responses. This pattern was consistently observed across all metrics, with only minor variations in statistical significance. For example, although GPT-3.5 did not differ significantly from GPT-4 in terms of correctness and coherence, it was rated significantly lower in readability and engagement.

Furthermore, it was unexpected that LLAMA-3, a relatively small open-source model, achieved marginally higher mean scores than GPT-4 across all evaluated categories, with the most pronounced difference observed in engagement. Although this difference was not statistically significant, it nevertheless suggests that even a compact open-source model may offer considerable capability and practical utility in these scenarios.

Overall, these findings reveal a clear trend. The more advanced LLMs, particularly GPT-4 and LLAMA-3, were consistently rated more favourably than both GPT-3.5 and human experts across all measured dimensions. These results support the conclusion that for general wildlife enquiries, current LLMs are capable of providing high-quality, expert-level responses that effectively contribute to knowledge acquisition and data discovery.

7 Limitations

Although the proposed approach has yielded promising results across all three use cases, several limitations must be acknowledged, particularly with regard to the practical deployment and scalability of LLMs. A primary constraint lies in the computational and financial demands associated with LLM utilisation. Running open-source models locally requires substantial hardware resources, which may not be accessible to all users. To mitigate this, we selected the lightweight 8B model versions of the open-source `LLAMA-3` series, as well as `OpenChat-3.5-0106`, both of which offer a strong balance between performance and resource efficiency. These models are well-suited for deployment on modest hardware and still delivered robust results within our use cases. While larger models, such as `LLAMA-3`'s 70B variant, may provide enhanced performance, their high computational demands placed them beyond the scope of our available infrastructure. Our choice thus reflects a practical compromise, prioritising accessibility and feasibility without significantly compromising model quality.

In contrast, commercial APIs such as OpenAI's GPT series offer more accessible and integrated solutions, but they involve significant cost implications. These costs are not limited to monetary considerations alone; latency is also a notable concern. For example, in our experiments, the average response times for few-shot prompts using GPT-4o-2024-05-13 and GPT-4-turbo-2024-04-09 were 5.95 and 5.47 seconds, respectively. Such latency may impair the usability of real-time systems and negatively affect the user experience. Although latency optimisation lies outside the scope of the present study, it remains an important consideration for practical deployment and merits further exploration.

Another limitation involves the reliance on RDF ontology embeddings, particularly in Use Cases 1 and 2. The proposed technique depends on the availability of RDF schema data, which must be embedded in the prompt to provide the necessary context for the LLM. When such data is unavailable, as might be the case with private endpoints where the necessary descriptive data is not accessible, the system lacks the semantic structure required to generate meaningful or accurate outputs. This limitation restricts the flexibility and scalability of our approach, confining its utility to environments where RDF data is available and structured in a compatible format.

Furthermore, the deployment of LLMs in practical settings introduces the risk of inaccuracies, particularly in the form of what is commonly referred to as "model hallucination". As discussed in Section 2.1 and 5.6.1, hallucination occurs when the model produces information that is factually incorrect, contextually irrelevant, or entirely fabricated. In Use Case 1, given that the generated responses are structured and must be presented interactively via the user interface, we implement an automated post-hoc validation step comprising (i) format and schema checks to ensure that the outputs parse as valid JSON, conform to the specified JSON Schema, and include all required fields; and (ii) fact-checking by verifying the validity of URIs for all extracted entities. If hallucinations are detected in the produced JSON, the system discards the output and invokes a fallback process that bypasses the LLM and applies traditional entity extraction techniques identical to those used when the LLM is not employed.

However, responses for Use Cases 2 and 3 are delivered as unstructured natural language output, and users are permitted to issue open-ended or loosely constrained queries. Thus, users may be exposed directly to inaccurate or misleading information. While techniques such as few-shot prompting have demonstrated some success in steering model responses towards more accurate and context-aware outputs, they do not provide a definitive safeguard against hallucinations. In this work, we have focused on identifying and documenting these instances, as well as exploring mitigation strategies to minimise their frequency and potential impact. The design and evaluation of automated hallucination-detection mechanisms fall outside the present scope and remain a priority for future work. One promising avenue is to use a second model, or the same model in a subsequent pass, to critique responses, perform consistency checks, and present the findings in a structured format.

In addition, as explained in Section 5.5.2, one of the challenges in Use Case 1 was ensuring that the model produced valid JSON. We addressed this through few-shot prompting and, following generation, a client-side post-processing stage to parse the JSON and conduct post hoc validation. Although schema-constrained decoding can enforce the expected structure and types at generation time, we did not adopt it because support was limited to a subset of the LLMs under evaluation. To preserve comparability, we therefore kept the API request configuration consistent across models.

An alternative approach is function calling, which allows the model to select from predefined functions and return JSON arguments validated against a specified schema. The host application then decides whether to run the function, executes it if appropriate, and returns the result for the model to incorporate into subsequent reasoning or generation. Function calling does not move work from the model to the system. Rather, it performs tool and parameter selection for operations such as database queries, vector search or RAG retrieval, embedding generation, and external API calls, all of which are executed outside the model. Because function calling is not uniformly supported across all LLMs, we did not rely on it in order to preserve cross-model comparability in our experiments. When implemented well, it can improve factual accuracy and robustness and can reduce token cost and sometimes latency. It typically introduces one additional model round trip when explanations or specific formatting are required, as in Use Case 2.

8 Conclusion and Future Work

In this paper, we propose a technique for integrating LLMs into existing conversational UIs, enhancing their functionality without requiring retraining. Our approach leverages the intrinsic strengths of LLMs to complement traditional chatbot frameworks. Specifically, it improves entity extraction from Linked Data and RDF triplestores, facilitates advanced reasoning over RDF datasets, and supports exploratory wildlife queries. This integration aims to enhance information retrieval and data discovery, thereby optimising the accessibility and usability of semantic data. We initiated our discussion by outlining the system's limitations with the proposed use cases, followed by a critical analysis of why direct SPARQL generation or outright replacement of the existing chatbot model with LLMs was suboptimal. Our proposed solution includes embedding RDF schema into LLM prompts to ensure the question is contextually enriched with relevant RDF data. Furthermore, we enhanced the quality of LLM responses through optimised prompt templates, leading to more accurate and context-aware interactions.

Our experimental evaluations demonstrated that LLMs can effectively function as an entity extraction component, a reasoning tool, and a source of high-quality, expert-level responses within our system, significantly augmenting user experience and improving the overall information retrieval process. In addition to the practical evaluation conducted in Use Cases 1 and 2, we performed a human assessment to evaluate the quality of LLM-generated responses. The findings revealed that these responses were perceived as more polished, accurate, and engaging than those produced by human experts.

Future work will focus on further enhancing our system's performance. In particular, we plan to investigate whether fine-tuning, especially open-source models like LLAMA-3, can yield improved results in practical applications. Additionally, we intend to explore the potential for these models to dynamically control the GUI in response to user queries by incorporating few-shot examples. Such research aims to refine our current techniques, expand their applicability across diverse domains, and further demonstrate the effectiveness of LLMs in query handling and information retrieval within semantic applications.

Overall, our approach addresses key limitations of prior systems, such as scalability and contextual understanding, and establishes a new benchmark for advancements in conversational UIs and Linked Data retrieval. The integration of LLMs into these use cases not only enhances semantic query processing and entity recognition but also paves the way for a wide range of future applications, ultimately contributing to more informed and productive interactions with complex data.

Acknowledgments

We gratefully acknowledge the scholarship and continued support provided by Saudi Electronic University, as well as the support received from the Danau Girang Field Centre (DGFC).

References

[1] Sweta Agrawal, Chunting Zhou, Mike Lewis, Luke Zettlemoyer, and Marjan Ghazvininejad. 2023. In-context Examples Selection for Machine Translation. In *Findings of the Association for Computational Linguistics: ACL 2023*, Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, Toronto, Canada, 8857–8873. doi:10.18653/v1/2023.findings-acl.564

[2] AI@Meta. 2024. Llama 3 Model Card. (2024). https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md

[3] Tareq Al-Moslmi, Marc Gallofré Ocaña, Andreas L. Opdahl, and Csaba Veres. 2020. Named Entity Extraction for Knowledge Graphs: A Literature Overview. *IEEE Access* 8 (2020), 32862–32881. doi:10.1109/ACCESS.2020.2973928

[4] David Baidoo-anu and Leticia Owusu Ansah. 2023. Education in the Era of Generative Artificial Intelligence (AI): Understanding the Potential Benefits of ChatGPT in Promoting Teaching and Learning. *Journal of AI* 7, 1 (2023), 52–62. doi:10.61969/jai.1337500

[5] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models Are Few-Shot Learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems (NIPS'20)*. Curran Associates Inc., Red Hook, NY, USA.

[6] Asli Celikyilmaz, Elizabeth Clark, and Jianfeng Gao. 2021. Evaluation of Text Generation: A Survey. https://arxiv.org/abs/2006.14799

[7] Wenhu Chen. 2023. Large Language Models are few(1)-shot Table Reasoners. In *Findings of the Association for Computational Linguistics: EACL 2023*, Andreas Vlachos and Isabelle Augenstein (Eds.). Association for Computational Linguistics, Dubrovnik, Croatia, 1120–1130. doi:10.18653/v1/2023.findings-eacl.83

[8] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V Le, and Jason Wei. 2024. Scaling Instruction-Finetuned Language Models. *Journal of Machine Learning Research* 25, 70 (2024), 1–53. http://jmlr.org/papers/v25/23-0870.html

[9] Giuseppe Di Fabbrizio, Amanda Stent, and Robert Gaizauskas. 2014. A Hybrid Approach to Multi-document Summarization of Opinions in Reviews. In *Proceedings of the 8th International Natural Language Generation Conference (INLG)*, Margaret Mitchell, Kathleen McCoy, David McDonald, and Aoife Cahill (Eds.). Association for Computational Linguistics, Philadelphia, Pennsylvania, U.S.A., 54–63.

doi:10.3115/v1/W14-4408

[10] Eleftherios Dimitrakis, Konstantinos Sgontzos, and Yannis Tzitzikas. 2020. A survey on question answering systems over linked data and documents. *Journal of Intelligent Information Systems* 55, 2 (2020), 233–259. doi:10.1007/s10844-019-00584-7

[11] Michael Dowling and Brian Lucey. 2023. ChatGPT for (Finance) research: The Bananarama Conjecture. *Finance Research Letters* 53 (2023), 103662. doi:10.1016/j.frl.2023.103662

[12] Bruno Faria, Dylan Perdigão, and Hugo Gonçalo Oliveira. 2023. Question Answering over Linked Data with GPT-3. In *12th Symposium on Languages, Applications and Technologies (SLATE 2023) (Open Access Series in Informatics (OASIcs), Vol. 113)*, Alberto Simões, Mario Marcelo Berón, and Filipe Portela (Eds.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 1:1–1:15. doi:10.4230/OASIcs.SLATE.2023.1

[13] Yu Feng, Linfang Ding, and Guohui Xiao. 2023. GeoQAMap - Geographic Question Answering with Maps Leveraging LLM and Open Knowledge Base. In *12th International Conference on Geographic Information Science (GIScience 2023) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 277)*, Roger Beecham, Jed A Long, Dianna Smith, Qunshan Zhao, and Sarah Wise (Eds.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 28:1–28:7. doi:10.4230/LIPIcs.GIScience.2023.28

[14] Nan Hu, Yike Wu, Guilin Qi, Dehai Min, Jiaoyan Chen, Jeff Z Pan, and Zafar Ali. 2023. An empirical study of pre-trained language models in simple knowledge graph question answering. *World Wide Web* 26, 5 (2023), 2855–2886. doi:10.1007/s11280-023-01166-y

[15] Zhentao Hu, Wei Hou, and Xianxing Liu. 2024. Deep learning for named entity recognition: a survey. *Neural Computing and Applications* 36, 16 (2024), 8995–9022. doi:10.1007/s00521-024-09646-6

[16] Krzysztof Janowicz, Armin Haller, Simon J D Cox, Danh Le Phuoc, and Maxime Lefrançois. 2019. SOSA: A lightweight ontology for sensors, observations, samples, and actuators. *Journal of Web Semantics* 56 (2019), 1–10. doi:10.1016/j.websem.2018.06.003

[17] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of Hallucination in Natural Language Generation. *ACM Comput. Surv.* 55, 12 (3 2023). doi:10.1145/3571730

[18] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering.

[19] Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2022. What Makes Good In-Context Examples for GPT-3?. In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*. Association for Computational Linguistics, Dublin, Ireland and Online, 100–114. doi:10.18653/v1/2022.deelio-1.10

[20] Joy Mahapatra, Sudip Kumar Naskar, and Sivaji Bandyopadhyay. 2016. Statistical Natural Language Generation from Tabular Non-textual Data. In *Proceedings of the 9th International Natural Language Generation conference*, Amy Isard, Verena Rieser, and Dimitra Gkatzia (Eds.). Association for Computational Linguistics, Edinburgh, UK, 143–152. doi:10.18653/v1/W16-6624

[21] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

[22] Michalis Mountantonakis and Yannis Tzitzikas. 2022. Linking Entities from Text to Hundreds of RDF Datasets for Enabling Large Scale Entity Enrichment. *Knowledge* 2, 1 (2022), 1–25. doi:10.3390/knowledge2010001

[23] Michalis Mountantonakis and Yannis Tzitzikas. 2023. Using Multiple RDF Knowledge Graphs for Enriching ChatGPT Responses. In *Machine Learning and Knowledge Discovery in Databases: Applied Data Science and Demo Track*, Gianmarco De Francisci Morales, Claudia Perlich, Natali Ruchansky, Nicolas Kourtellis, Elena Baralis, and Francesco Bonchi (Eds.). Springer Nature Switzerland, Cham, 324–329.

[24] Michalis Mountantonakis and Yannis Tzitzikas. 2025. Generating SPARQL Queries over CIDOC-CRM Using a Two-Stage Ontology Path Patterns Method in LLM Prompts. *J. Comput. Cult. Herit.* 18, 1 (2 2025). doi:10.1145/3708326

[25] Omar Mussa, Omer Rana, Benoit Goossens, Pablo Orozco Ter wengel, and Charith Perera. 2024. ForestQB: Enhancing Linked Data Exploration through Graphical and Conversational UIs Integration. *ACM J. Comput. Sustain. Soc.* 2, 3 (9 2024). doi:10.1145/3675759

[26] Omar Mussa, Omer Rana, Benoit Goossens, Pablo Orozco Ter wengel, and Charith Perera. 2025. Towards Enhancing Linked Data Retrieval in Conversational UIs Using Large Language Models. In *Web Information Systems Engineering – WISE 2024*, Mahmoud Barhamgi, Hua Wang, and Xin Wang (Eds.). Springer Nature Singapore, Singapore, 246–261.

[27] Jianmo Ni, Gustavo Hernández Ábrego, Noah Constant, Ji Ma, Keith B Hall, Daniel Cer, and Yinfei Yang. 2021. Sentence-T5: Scalable Sentence Encoders from Pre-trained Text-to-Text Models.

[28] Italo L Oliveira, Renato Fileto, René Speck, Luís P F Garcia, Diego Moussallem, and Jens Lehmann. 2021. Towards holistic Entity Linking: Survey and directions. *Information Systems* 95 (2021), 101624. doi:10.1016/j.is.2020.101624

[29] Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. 2019. Language Models as Knowledge Bases? (9 2019). http://arxiv.org/abs/1909.01066

[30] André Gomes Regino and Julio Cesar dos Reis. 2025. Can LLMs be Knowledge Graph Curators for Validating Triple Insertions?. In *Proceedings of the Workshop on Generative AI and Knowledge Graphs (GenAIK)*, Genet Asefa Gesese, Harald Sack, Heiko Paulheim, Albert Merono-Penuela, and Lihu Chen (Eds.). International Committee on Computational Linguistics, Abu Dhabi, UAE, 87–99. https://aclanthology.org/2025.genaik-1.10/

[31] Nils Reimers and Iryna Gurevych. 2019. 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), Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (Eds.). Association for Computational Linguistics, Hong Kong, China, 3982–3992. doi:10.18653/v1/D19-1410

[32] Petar Ristoski and Heiko Paulheim. 2016. RDF2Vec: RDF Graph Embeddings for Data Mining. In *The Semantic Web – ISWC 2016*. Springer International Publishing, Cham, 498–514.

[33] Malik Sallam. 2023. ChatGPT Utility in Healthcare Education, Research, and Practice: Systematic Review on the Promising Perspectives and Valid Concerns. *Healthcare* 11, 6 (2023). doi:10.3390/healthcare11060887

[34] Timo Schick and Hinrich Schütze. 2021. It's Not Just Size That Matters: Small Language Models Are Also Few-Shot Learners. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (Eds.). Association for Computational Linguistics, Online, 2339–2352. doi:10.18653/v1/2021.naacl-main.185

[35] Yiming Tan, Dehai Min, Yu Li, Wenbo Li, Nan Hu, Yongrui Chen, and Guilin Qi. 2023. Can ChatGPT Replace Traditional KBQA Models? An In-Depth Analysis of the Question Answering Performance of the GPT LLM Family. In *The Semantic Web – ISWC 2023*, Terry R Payne, Valentina Presutti, Guilin Qi, María Poveda-Villalón, Giorgos Stoilos, Laura Hollink, Zoi Kaoudi, Gong Cheng, and Juanzi Li (Eds.). Springer Nature Switzerland, Cham, 348–367.

[36] Chris van der Lee, Albert Gatt, Emiel van Miltenburg, Sander Wubben, and Emiel Krahmer. 2019. Best practices for the human evaluation of automatically generated text. In *Proceedings of the 12th International Conference on Natural Language Generation*, Kees van Deemter, Chenghua Lin, and Hiroya Takamura (Eds.). Association for Computational Linguistics, Tokyo, Japan, 355–368. doi:10.18653/v1/W19-8643

[37] Guan Wang, Sijie Cheng, Xianyuan Zhan, Xiangang Li, Sen Song, and Yang Liu. 2023. OpenChat: Advancing Open-source Language Models with Mixed-Quality Data. *arXiv preprint arXiv:2309.11235* (2023).

[38] Changrong Xiao, Sean Xin Xu, Kunpeng Zhang, Yufang Wang, and Lei Xia. 2023. Evaluating Reading Comprehension Exercises Generated by LLMs: A Showcase of ChatGPT in Education Applications. In *Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023)*, Ekaterina Kochmar, Jill Burstein, Andrea Horbach, Ronja Laarmann-Quante, Nitin Madnani, Anaïs Tack, Victoria Yaneva, Zheng Yuan, and Torsten Zesch (Eds.). Association for Computational Linguistics, Toronto, Canada, 610–625. doi:10.18653/v1/2023.bea-1.52

[39] Ningyu Zhang, Luoqiu Li, Xiang Chen, Shumin Deng, Zhen Bi, Chuanqi Tan, Fei Huang, and Huajun Chen. 2022. Differentiable Prompt Makes Pre-trained Language Models Better Few-shot Learners. (8 2022).

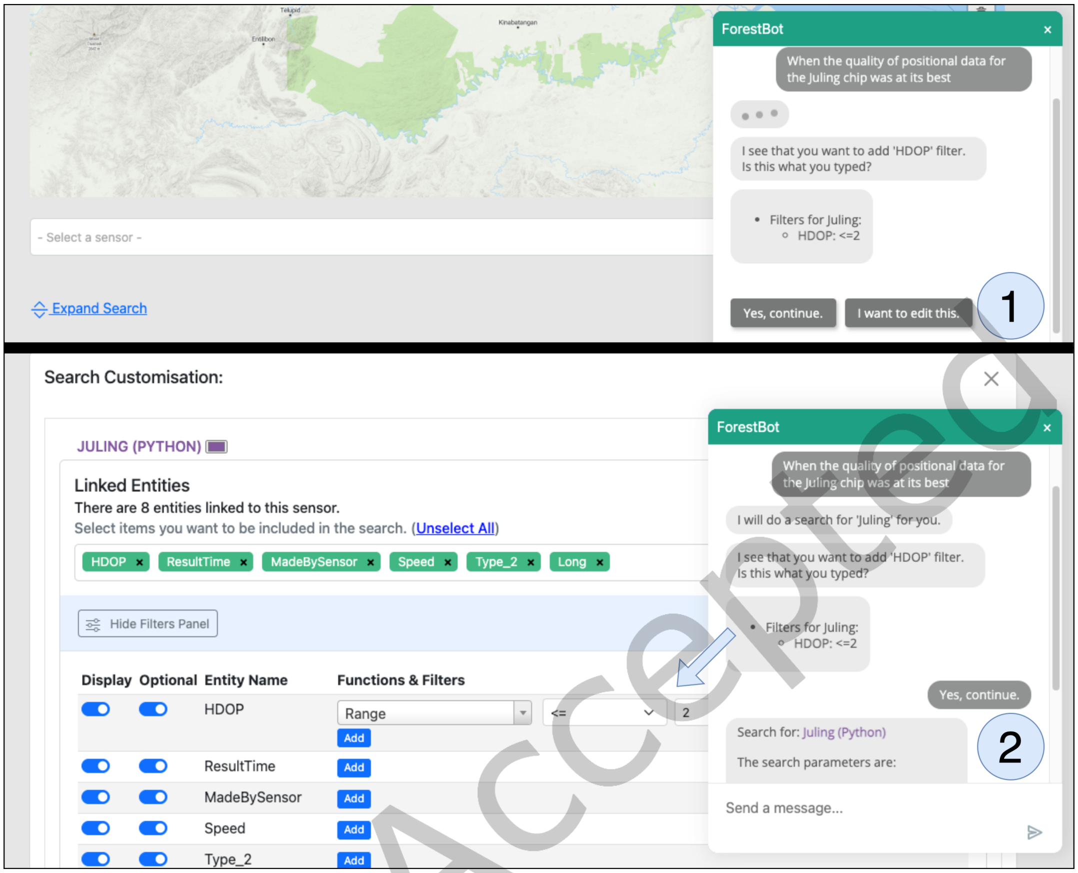
[40] Rongzhi Zhang, Yue Yu, Pranav Shetty, Le Song, and Chao Zhang. 2022. Prompt-Based Rule Discovery and Boosting for Interactive Weakly-Supervised Learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (Eds.). Association for Computational Linguistics, Dublin, Ireland, 745–758. doi:10.18653/v1/2022.acl-long.55

[41] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating Text Generation with BERT.

A Appendix: Experiment Questions for LLM Evaluation Across Use Cases 1 and 2

This appendix contains the full set of questions used to evaluate LLM performance across Use Case 1 and Use Case 2 in our experiment. The questions are grouped according to complexity and relevance to either sensor data (Use Case 1: Extracting Entities) or RDF and ontology-based reasoning (Use Case 2: Query about RDF and Ontology).

Each use case is divided into four distinct groups based on query type, from simple direct queries with single entities to complex queries that require understanding indirect references and reasoning with multiple entities. For each group, the questions are tailored to assess specific capabilities of the LLM, such as direct retrieval of sensor data, handling of synonyms, and the ability to make inferences based on contextual RDF information. The structure of the groups provides a comprehensive approach to measure LLM accuracy, inference skills, and robustness across different scenarios.



When the quality of positional data for the Juling chip was at its best

I see that you want to add 'HDOP' filter. Is this what you typed?

- Filters for Juling:
  - HDOP: <=2

Linked Entities

There are 8 entities linked to this sensor.

Select items you want to be included in the search. (Unselect All)

HDOP × ResultTime × MadeBySensor × Speed × Type_2 × Long ×

Display	Optional	Entity Name	Functions & Filters
●	●	HDOP	Range <= 2 Add
●	●	ResultTime	Add
●	●	MadeBySensor	Add
●	●	Speed	Add
●	●	Type_2	Add

When the quality of positional data for the Juling chip was at its best

I will do a search for 'Juling' for you.

I see that you want to add 'HDOP' filter. Is this what you typed?

- Filters for Juling:
  - HDOP: <=2

Yes, continue.

Search for: Juling (Python)

The search parameters are:

Fig. 10. This figure illustrates the testing approach for Use Case 1, wherein each query was manually input into the tool. Initially, as depicted in Screen 1, the accuracy of the extracted entities was verified. For example, when the user entered "When the quality of positional data for the Juling chip was at its best," the chatbot correctly identified and displayed "HDOP <= 2" as the relevant entity. Following successful verification, as shown in Screen 2, if the responses were accurate, the chatbot was prompted to proceed and automate the selection process. Here, the user confirmed the accuracy by typing "yes," prompting the chatbot to select the Juling sensor, apply the correct entities, and employ the appropriate filters without errors. This dual-validation process ensures the practicality and reliability of our approach for each utilised large language model (LLM).

Table 10. **Use Case 1 (Group 1)**: Questions targeting single, explicitly referenced entities within the RDF data, designed to test the model's precision in retrieving straightforward information without additional inference requirements.

Use Case 1: Simple Direct Queries (Group 1)	
Task	**Question**
1.1	What was Dara's readings value when Dara's speed exceeded 170?
1.2	Where was Aqeela headed when its PDOP value was below 10?
1.3	Are there any instances where Sandi's temperature measurements are below 28?
1.4	How often has Putut's sensor recorded a speed that is between 10 and 20?
1.5	Which readings from Juling have HDOP not bound to value?
1.6	What readings have been recorded by Juling when the sensor type contains GPS?
1.7	Show me the Liun readings from when its direction was towards the north.
1.8	Are there any readings from Kuma with an altitude greater than 1000?
1.9	Are there any readings from Kasih with a result time between January 5, 2022, and January 6, 2022?
1.10	When was the HDOP value <= 10 for the Juling chip?

Table 11. **Use Case 1 (Group 2)**: Questions involving multiple, directly mentioned entities in the RDF data, aimed at assessing the model's ability to process and integrate multiple data points within a single response.

Use Case 1: Complex Direct Queries (Group 2)	
Task	**Question**
2.1	What were the reading values for Dara and Aqeela when Dara's speed exceeded 170 and Aqeela's speed was less than 20?
2.2	Where was Aqeela headed when its Position Dilution of Precision value was poor?
2.3	Find readings for Kuma and Aqeela within a latitude range of 3 km from the point at latitude 6.160698, and a longitude range of 3 km from the point at longitude 116.756694.
2.4	Find readings for Liun and Merotai within area 3, where the direction of Merotai is 205 and Liun altitude is above 951.
2.5	When was the quality of positional data at its best for the Juling chip, specifically for the Horizontal Dilution of Precision?
2.6	Retrieve the readings from all sensors that include sensor type as part of their data.
2.7	What are the readings for Juling where its Horizontal Dilution of Precision is not empty and its sensor type contains 'GPS'?
2.8	What are the readings for Sandi, Kasih, and Puteri from the last year, during which Kasih and Puteri's speed > 160, and Sandi's temperature is bound to a value?
2.9	What are the readings for sensors that observe Horizontal Dilution of Precision?
2.10	Retrieve the top 30 readings of Jasmin on 2016-05-10 where temperature is below 27 and speed exceeds 3.
2.11	Get the latest 50 Readings of Sandi between 2015-08-11 and 2015-08-12 with Temperature > 32 and Speed above 1 within area 1.

Table 12. **Use Case 1 (Group 3)**: Queries related to a single entity but framed with indirect references, such as synonyms and varied wording, to evaluate the model's ability to infer information and recognize synonymous terms.

Use Case 1: Simple Indirect Queries (Group 3)	
Task	**Question**
3.1	What were Dara's readings when Dara's velocity surpassed 170?
3.2	Where was Aqeela headed when its location accuracy index was below 10?
3.3	Are there any instances where Sandi's heat readings are below 28?
3.4	How frequently has Putut's sensor registered a rate of movement between 10 and 20?
3.5	Which readings from Juling are missing a horizontal spatial accuracy value?
3.6	What readings have been captured by Juling when the data source contains GPS?
3.7	Show me the Liun readings from when it was oriented towards the north.
3.8	Are there any readings from Kuma at elevations exceeding 1000?
3.9	Are there any readings from Kasih recorded in the timeframe of January 5, 2022, to January 6, 2022?
3.10	When was the quality of positional data <= 10 for the Juling chip?

Table 13. **Use Case 1 (Group 4)**: Questions encompassing multiple entities referenced indirectly or with varied terminology, used to assess the model's comprehension, reasoning, and synthesis abilities when handling implicit information.

Use Case 1: Complex Indirect Queries (Group 4)	
Task	**Question**
4.1	What were the reading values for Dara and Aqeela when Dara's movement was over 170 and Aqeela's motion was lower than 20?
4.2	Where was Aqeela headed when its Signal Strength was weak?
4.3	Find readings for Liun and Merotai within area 3, where the heading of Merotai is 205 and Liun's height is above 951.
4.4	When the quality of positional data for the Juling chip was at its best?
4.5	Retrieve readings from all sensors that record the name of the sensor generating the data as part of their information.
4.6	What are the readings for Juling where the quality of positional data has a value and the source of the data is identified as 'GPS'?
4.7	What are the readings for Sandi, Kasih, and Puteri from the last year, during which Kasih and Puteri's movement was faster than 160, and Sandi's recorded heat was not empty?
4.8	What are the readings of sensors that monitor the quality of positional data?
4.9	Retrieve the top 30 readings of Jasmin on 2016-05-10 where heat is below 27 and velocity surpasses 3.
4.10	Get the latest 50 Readings of Sandi between 2015-08-11 and 2015-08-12 with heat higher than 32 and motion beyond 1 within area 1.

Table 14. **Use Case 2 (Group 5)**: Basic questions directly linked to RDF data properties, intended to determine the model's capacity to interpret core information about entities and relationships as explicitly stated in RDF.

Use Case 2: Simple Direct Queries (Group 5)	
Task	**Question**
5.1	Are there any sensor deployments related to camera traps?
5.2	Does Aqeela observe a property that determines its speed?
5.3	Can the GPS collar for the elephant named Abaw measure the elephant's altitude?
5.4	What observable properties does the GPS collar for the elephant named Aqeela monitor?
5.5	How is tree age estimated in the Meranti Tree Project?
5.6	What sensors are used to measure water level?
5.7	Do the camera traps capture latitude and longitude data?
5.8	Is airTemperature an observable property of any sensors?
5.9	Which animals are observed using a VHF chip?
5.10	What environmental factors do soil sensors measure?

Table 15. **Use Case 2 (Group 6)**: Questions that require reasoning across multiple properties or entities directly referenced in RDF, designed to test the model's ability to integrate various RDF data points into a cohesive response.

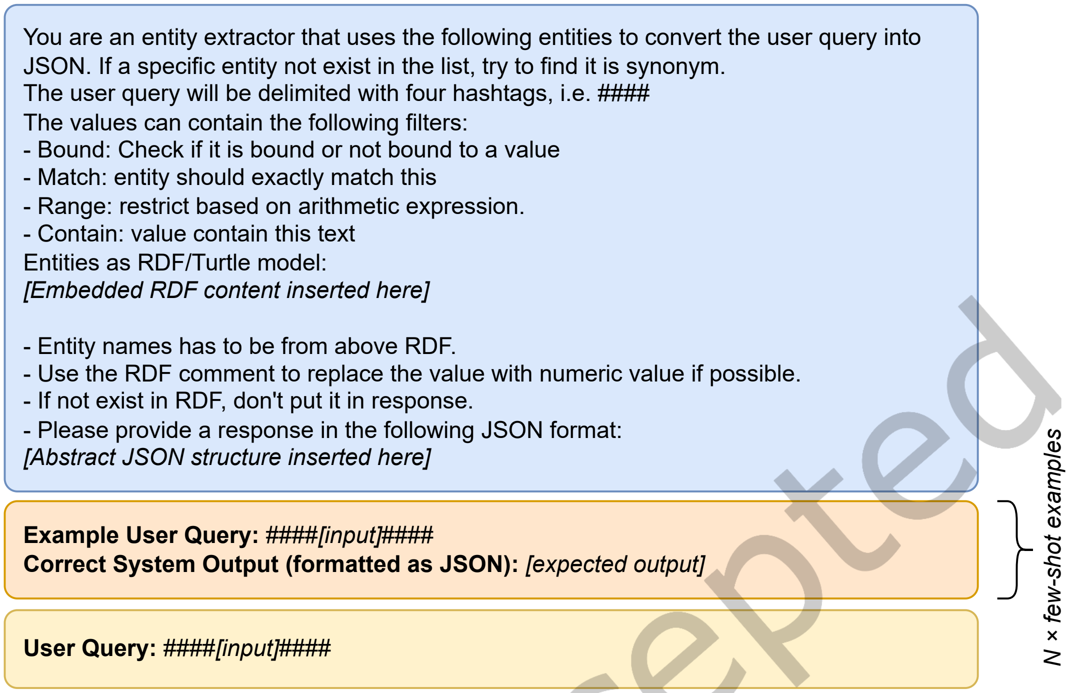
Use Case 2: Complex Direct Queries (Group 6)	
Task	**Question**
6.1	Are there differences in the observable properties of the GPS collars worn by elephants Ita and Dara?
6.2	Which elephant sensors observe the external temperature of the environment outside the sensor?
6.3	What kind of image format do the Camera Traps capture?
6.4	What tree measurements are observed in the Meranti Tree Project-1?
6.5	What is the significance of the HDOP observable property in the GPS collar worn by elephant Dara?
6.6	What types of weather conditions are monitored by Weather Sensor 1?
6.7	What is the role of the 'Result Time' observable property in the data collected by elephant GPS collars?
6.8	How does the External Temperature observable property of elephant Dara's GPS collar differ from the internal temperature measurement in other collars?
6.9	What data format does Camera Trap 3 capture, and how does it differ from Camera Trap 1?
6.10	Is an 8-meter reading on the River Sensor considered dangerous, and at what level should I become concerned?

Table 16. **Use Case 2 (Group 7)**: Basic queries involving indirectly referenced RDF properties, created to assess the model's capacity for RDF reasoning and the ability to interpret concepts implied through synonyms or indirect associations.

Use Case 2: Simple Indirect Queries (Group 7)	
Task	**Question**
7.1	Are there any sensor deployments related to outdoor cameras?
7.2	Does Aqeela maintain a feature that determines its motion?
7.3	Can the tracking device for the elephant named Abaw measure the elephant's elevation?
7.4	What parameters does the tracking device for the elephant named Aqeela monitor?
7.5	What technique is utilised to approximate the years of growth in the Meranti Tree Project?
7.6	What are the sensors used to determine the river's height?
7.7	Do the sensors that capture images record geographic coordinates?
7.8	Do any sensors have the capability to capture wind heat as a measurable feature?
7.9	Which animals are tracked using a Telemetry device?
7.10	What ecological variables do ground sensors measure?

Table 17. **Use Case 2 (Group 8)**: Advanced queries involving indirect RDF references and complex reasoning requirements, meant to evaluate the model's aptitude for nuanced RDF interpretation, inference, and relationship understanding in less explicit contexts.

Use Case 2: Complex Indirect Queries (Group 8)	
Task	**Question**
8.1	Are there differences in the characteristics of the tags worn by elephants Ita and Dara?
8.2	Which elephant sensors record the heat of the area surrounding the sensor?
8.3	What kind of image data type do the camera devices generate?
8.4	What are the characteristics monitored in the first Meranti Tree Project?
8.5	What is the significance of the Horizontal Dilution of Precision in the tag worn by elephant Dara?
8.6	What types of environmental conditions are monitored by the first Weather Sensor?
8.7	What is the role of the time recorded in the data collected by elephant tags?
8.8	How does the Outer Heat of elephant Dara's tag differ from the heat measurement in other tags?
8.9	What specific data does the third Camera record, and how does it differ from the first Camera?
8.10	Is an 8-meter reading on the water sensor considered risky, and at what threshold should I become worried?



You are an entity extractor that uses the following entities to convert the user query into JSON. If a specific entity not exist in the list, try to find it is synonym.
The user query will be delimited with four hashtags, i.e. ####
The values can contain the following filters:

- Bound: Check if it is bound or not bound to a value
- Match: entity should exactly match this
- Range: restrict based on arithmetic expression.
- Contain: value contain this text

Entities as RDF/Turtle model:
[Embedded RDF content inserted here]

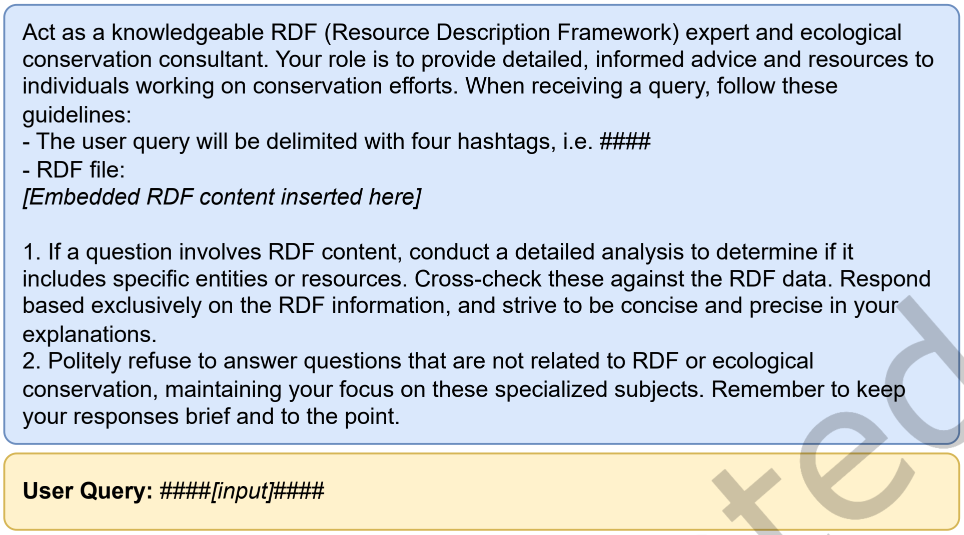
- Entity names has to be from above RDF.
- Use the RDF comment to replace the value with numeric value if possible.
- If not exist in RDF, don't put it in response.
- Please provide a response in the following JSON format:

[Abstract JSON structure inserted here]

Example User Query: ####*[input]*####
Correct System Output (formatted as JSON): *[expected output]*

User Query: ####*[input]*####

Fig. 11. This figure presents the prompt used in Use Case 1. The first block contains the complete instructions, incorporating the embedded RDF data, which is placed in the designated placeholder, with a JSON structure template positioned at the end of this block to help the model understand the expected output format. The second block follows, providing a series of few-shot examples where user queries are paired with their corresponding correct responses. These examples serve as guidance for the model and are presented as historical context to enhance its ability to generate more accurate responses. The final block contains the actual user query, instructing the model to generate an appropriate response.



Act as a knowledgeable RDF (Resource Description Framework) expert and ecological conservation consultant. Your role is to provide detailed, informed advice and resources to individuals working on conservation efforts. When receiving a query, follow these guidelines:
- The user query will be delimited with four hashtags, i.e. ####
- RDF file:
[Embedded RDF content inserted here]

1. If a question involves RDF content, conduct a detailed analysis to determine if it includes specific entities or resources. Cross-check these against the RDF data. Respond based exclusively on the RDF information, and strive to be concise and precise in your explanations.
2. Politely refuse to answer questions that are not related to RDF or ecological conservation, maintaining your focus on these specialized subjects. Remember to keep your responses brief and to the point.

User Query: *####[input]####*

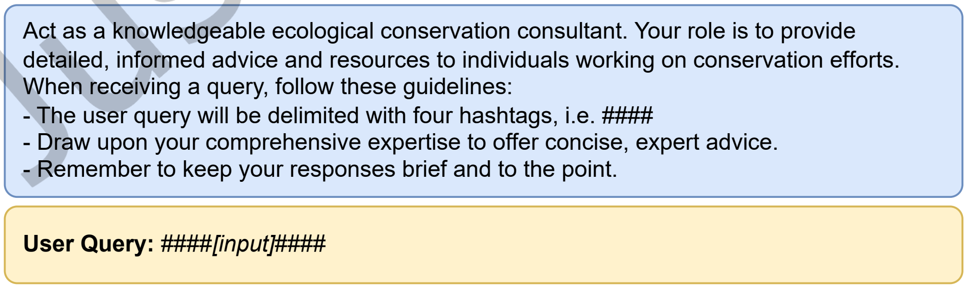
Fig. 12. This figure illustrates the prompt used in Use Case 2. The first block provides the full set of instructions, while the second block presents the user query, prompting the model to generate an appropriate response.

B Appendix: Questions and Results for Use Case 3 Experiment

This appendix provides a comprehensive overview of the materials and methods used in the user evaluation for Use Case 3. It includes the full set of questions presented in Survey A and Survey B, designed to evaluate user preferences and perceptions. Additionally, this section details the results obtained from the evaluation, along with an in-depth analysis of the statistical tests performed, including ANOVA and Tukey's Honest Significant Difference (HSD) tests. The appendix aims to offer transparency and reproducibility by presenting the complete dataset and the methodologies employed to assess model performance across the four evaluation categories: Readability, Correctness, Coherence, and Engagement.

Table 18. **Use Case 3**: Questions utilised in the human evaluation of this use case were developed in collaboration with experts in wildlife research to support wildlife researchers in data discovery and knowledge acquisition. The letter accompanying the question number indicates the survey in which it was used (Survey A or Survey B).

#	**Question**
A1	How do researchers use tree measurements in ecological studies?
A2	What information can we gather from GPS collar data?
A3	How do camera traps get activated?
A4	What are the methods for estimating wildlife populations?
A5	What are common challenges in wildlife data collection?
A6	What factors determine the placement of river sensors?
A7	How can technology aid in tracking animal migration patterns?
A8	What safety precautions should researchers take in the field?
B1	How can we estimate the age of a tree in a forest?
B2	What is the ideal location to set up a camera trap for bird monitoring?
B3	How can we ensure minimal disturbance to wildlife when setting up equipment?
B4	What types of animals are typically tracked with GPS collars?
B5	How can we use technology to monitor endangered species?
B6	What are the indicators of a healthy animal population?
B7	How do weather patterns affect wildlife behaviour?
B8	What are the ethical considerations in wildlife research?



Act as a knowledgeable ecological conservation consultant. Your role is to provide detailed, informed advice and resources to individuals working on conservation efforts. When receiving a query, follow these guidelines:
- The user query will be delimited with four hashtags, i.e. ####
- Draw upon your comprehensive expertise to offer concise, expert advice.
- Remember to keep your responses brief and to the point.

User Query: *####[input]####*

Fig. 13. This figure illustrates the prompt used in Use Case 3. The first block provides the full set of instructions, while the second block contains the user's query, directing the model to generate a suitable response.

Table 19. This table summarises the results of user ratings for Use Case 3, where participants rated their preferences for outputs generated by the three models (GPT-4, GPT-3.5, and LLAMA-3) and the Expert-produced outputs across four evaluation categories: Readability, Correctness, Coherence, and Engagement. Each row includes the statistical significance (indicating whether the differences between models are meaningful), mean ratings, standard deviation, variance, and ANOVA p-values. The ANOVA test was used to identify significant differences in user preferences across models for each category. Results demonstrate that GPT-4 and LLAMA-3 generally achieve higher user ratings compared to GPT-3.5 and expert-produced outputs across all categories, with notable differences in Engagement where expert-produced output ratings are substantially lower.

Category	**Significant**	**Source**	**Mean**	**StdDev**	**Variance**	**ANOVA p-value**
Readability	TRUE	GPT-4	4.35	0.3924	0.1539	3.38×10^{-9}
		GPT-3.5	3.9188	0.5102	0.2603	
		LLAMA-3	4.3813	0.3815	0.1455	
		Expert	3.4188	0.5565	0.3097	
Correctness	TRUE	GPT-4	4.3875	0.4193	0.1758	1.86×10^{-11}
		GPT-3.5	4.025	0.4635	0.2148	
		LLAMA-3	4.4375	0.4579	0.2097	
		Expert	3.3	0.52	0.2704	
Coherence	TRUE	GPT-4	4.3125	0.4507	0.2031	1.15×10^{-9}
		GPT-3.5	4.0125	0.4514	0.2038	
		LLAMA-3	4.3438	0.4309	0.1856	
		Expert	3.3938	0.4278	0.1830	
Engagement	TRUE	GPT-4	4.05	0.5152	0.2655	1.95×10^{-15}
		GPT-3.5	3.5313	0.6014	0.3616	
		LLAMA-3	4.225	0.5203	0.2707	
		Expert	2.5563	0.4975	0.2475	

Table 20. This table summarises the results of Tukey's Honest Significant Difference (HSD) test performed on user ratings for Use Case 3. Pairwise comparisons were conducted between the three models (GPT-4, GPT-3.5, and LLAMA-3) and Expert-produced outputs across the four evaluation categories: Readability, Correctness, Coherence, and Engagement. Each row reports the mean difference, the adjusted p-value (p-adj), the confidence interval bounds, and the null hypothesis outcome. Reject = TRUE means the null hypothesis is rejected and a significant user preference exists between the groups, while Reject = FALSE indicates no significant difference. A p-adj below 0.05 is considered significant. The confidence interval is considered significant if its lower and upper bounds agree in sign (i.e. the interval does not include zero). The preferred source is shown in **bold**: if the Mean Diff is negative (indicating that Group1's mean is higher), then Group 1 is preferred; if the Mean Diff is positive (indicating that Group2's mean is higher), then Group 2 is preferred. The results highlight significant differences in user ratings between most pairs, especially in Engagement and Correctness, where Expert-produced output scores are markedly lower compared to the models.

Category	**Group 1**	**Group 2**	**Mean Diff**	**p-adj**	**Lower**	**Upper**	**Reject**
Readability	GPT-3.5	**GPT-4**	0.4312	0.023	0.044	0.8185	TRUE
	GPT-3.5	Expert	-0.5	0.006	-0.8873	-0.1127	TRUE
	GPT-3.5	**LLAMA-3**	0.4625	0.0127	0.0752	0.8498	TRUE
	GPT-4	Expert	-0.9312	0	-1.3185	-0.544	TRUE
	GPT-4	**LLAMA-3**	0.0312	0.9966	-0.356	0.4185	FALSE
	Expert	**LLAMA-3**	0.9625	0	0.5752	1.3498	TRUE
Correctness	GPT-3.5	**GPT-4**	0.3625	0.0752	-0.0251	0.7501	FALSE
	GPT-3.5	Expert	-0.725	0	-1.1126	-0.3374	TRUE
	GPT-3.5	**LLAMA-3**	0.4125	0.0325	0.0249	0.8001	TRUE
	GPT-4	Expert	-1.0875	0	-1.4751	-0.6999	TRUE
	GPT-4	**LLAMA-3**	0.05	0.9865	-0.3376	0.4376	FALSE
	Expert	**LLAMA-3**	1.1375	0	0.7499	1.5251	TRUE
Coherence	GPT-3.5	**GPT-4**	0.3	0.1456	-0.0658	0.6658	FALSE
	GPT-3.5	Expert	-0.6188	0.0002	-0.9845	-0.253	TRUE
	GPT-3.5	**LLAMA-3**	0.3312	0.0899	-0.0345	0.697	FALSE
	GPT-4	Expert	-0.9188	0	-1.2845	-0.553	TRUE
	GPT-4	**LLAMA-3**	0.0312	0.996	-0.3345	0.397	FALSE
	Expert	**LLAMA-3**	0.95	0	0.5842	1.3158	TRUE
Engagement	GPT-3.5	**GPT-4**	0.5187	0.0156	0.0743	0.9632	TRUE
	GPT-3.5	Expert	-0.975	0	-1.4195	-0.5305	TRUE
	GPT-3.5	**LLAMA-3**	0.6937	0.0006	0.2493	1.1382	TRUE
	GPT-4	Expert	-1.4938	0	-1.9382	-1.0493	TRUE
	GPT-4	**LLAMA-3**	0.175	0.73	-0.2695	0.6195	FALSE
	Expert	**LLAMA-3**	1.6687	0	1.2243	2.1132	TRUE

Received 28 April 2025; revised 11 November 2025; accepted 10 January 2026