



Article

Context-Aware Few-Shot Learning SPARQL Query Generation from Natural Language on an Aviation Knowledge Graph

Ines-Virginia Hernandez-Camero, Eva Garcia-Lopez , Antonio Garcia-Cabot * and Sergio Caro-Alvaro

Departamento de Ciencias de la Computación, Universidad de Alcalá, 28801 Alcalá de Henares, Madrid, Spain; inesvirginia.hernand@uah.es (I.-V.H.-C.); eva.garcial@uah.es (E.G.-L.); sergio.caro@uah.es (S.C.-A.)

* Correspondence: a.garcia@uah.es

Abstract: Question answering over domain-specific knowledge graphs implies several challenges. It requires sufficient knowledge of the world and the domain to understand what is being asked, familiarity with the knowledge graph's structure to build a correct query, and knowledge of the query language. However, mastering all of these is a time-consuming task. This work proposes a prompt-based approach that enables natural language to generate SPARQL queries. By leveraging the advanced language capabilities of large language models (LLMs), we constructed prompts that include a natural-language question, relevant contextual information from the domain-specific knowledge graph, and several examples of how the task should be executed. To evaluate our method, we applied it to an aviation knowledge graph containing accident report data. Our approach improved the results of the original work—in which the aviation knowledge graph was first introduced—by 6%, demonstrating its potential for enhancing SPARQL query generation for domain-specific knowledge graphs.

Keywords: knowledge graphs; SPARQL; query generation; prompting; large language models



Academic Editors: Mourad Abbas, Najim Dehak and Gérard Chollet

Received: 25 April 2025

Revised: 2 June 2025

Accepted: 11 June 2025

Published: 13 June 2025

Citation: Hernandez-Camero, I.-V.; Garcia-Lopez, E.; Garcia-Cabot, A.; Caro-Alvaro, S. Context-Aware Few-Shot Learning SPARQL Query Generation from Natural Language on an Aviation Knowledge Graph. *Mach. Learn. Knowl. Extr.* **2025**, *7*, 52. <https://doi.org/10.3390/make7020052>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Knowledge base question answering (KBQA) is a classical natural language processing (NLP) task that has been widely studied. Unlike methods that rely on parametric knowledge stored within models, KBQA utilizes structured knowledge bases to extract answers [1]. With the recent advances in large language models (LLMs), their outstanding capabilities in language understanding and generation are being leveraged to solve KBQA tasks. However, this involves understanding the schema of a given knowledge base (KB) and grasping the diverse structures of KBs, which represents a challenge for LLMs, given that schemas are generally heterogeneous due to being user-created as well as very large in size [2].

Commonly used KBs are knowledge graphs (KGs), which are effective structures to model real-world knowledge. They represent relationships between entities in the form of triples (head entity, relation, tail entity) [3]. A common framework for publishing KGs on the Internet is the Resource Description Framework (RDF), a graph-based data model [4]. To access and retrieve information from RDF-based KGs, SPARQL has become the standard query language [5].

Knowledge graph question answering (KGQA) specifically attempts to find answers to natural language (NL) questions from a knowledge graph [6]. These answers can be used as direct answers or can be further passed to an LLM as an external knowledge source that can be combined with the parametric knowledge of the LLM. The knowledge of the

LLM combined with the information from the KG has been used to address traditional issues related to LLMs such as hallucinations [7].

Current KBQA systems can be divided into two categories: information extraction and semantic parsing [8]. While information extraction methods generally construct a question-specific graph that gathers all relevant entities and relationships from the KB, semantic parsing methods are centered on transforming a natural-language question into logical forms (e.g., SPARQL, the standard query language) that are then executed against the KB [9]. Furthermore, LLMs have recently shown great capabilities in code generation tasks [10]. Hence, as the SPARQL query language can be considered a type of code, the incorporation of LLM-based approaches into KBQA can offer an improvement in the performance of such semantic parsing models. However, some studies highlight their limitations in generating accurate SPARQL queries. For instance, while being syntactically correct, these queries often contain semantic errors [11,12].

Additionally, most current KBQA systems depend on supervised training, which requires a large amount of labeled data [13]. However, as mentioned before, since different KBs are large in size and have unique schemas and structures, each KB requires a customized training process. These challenges highlight the need for a more flexible approach that can adapt to different KBs without extensive retraining. In response, recent advancements in LLMs have introduced new possibilities for enhancing KBQA systems. These models have shown promising results in reasoning [14] and few-shot learning capabilities [15,16]. These developments suggest the potential for more flexible frameworks that may generalize across diverse KBs with reduced reliance on task-specific supervision.

This study introduces a flexible three-step methodology for generating SPARQL queries from natural-language questions within a domain-specific knowledge graph. The proposed approach integrates relevant triples and task examples into a single prompt in an attempt to guide the translation of natural language into executable SPARQL queries by an LLM. The primary objective was to develop an adaptable method for SPARQL query generation tailored to domain-specific knowledge graphs. To test this method, a test set that comprised 150 questions over a domain-specific knowledge graph was utilized, specifically an aviation knowledge graph [17].

The rest of this study is organized as follows. In Section 2, several related works are introduced. In Section 3, the methodology of this work is explained alongside an introduction to the knowledge graph used to test our method. Section 4 presents the results of the proposed method on the aviation knowledge graph. In Section 5, the findings of this work are discussed and certain issues related to the test set are raised. Finally, in Section 6, the conclusions of this study are presented.

2. Related Works

Enhancing SPARQL query generation has wide-ranging implications for domains where information is stored in structured and semantically rich knowledge graphs. For instance, in scientific research, LLMs often struggle to provide accurate answers due to hallucinations and difficulty reasoning over rare or complex concepts. Integrating LLMs with domain-specific knowledge graphs, such as the Open Research Knowledge Graph (ORKG) [18] or the CS-KG (Computer Science Knowledge Graph <https://scholkg.kmi.open.ac.uk>, accessed on 8 June 2025), offers a promising solution by grounding natural-language questions in structured, verifiable data [19,20]. An effective method to integrate LLMs with KGs involves translating questions in natural language into SPARQL queries [21]. This facilitates the retrieval of relevant information from the KG, which can be further passed to an LLM or presented directly to the user [20].

Similarly, modern information retrieval systems have turned to KGs to address limitations of traditional information retrieval, such as limited outputs in keyword search-based systems [22,23]. Knowledge graph-based retrieval enables semantic representation, higher search efficiency, and more accurate results by leveraging entities and relationships rather than raw text [24]. For instance, Jia et al. [25] constructed SPARQL queries to retrieve information from the ANU (Australian National University) scholarly knowledge graph (ASKG), which encompasses knowledge from academic papers produced at ANU in the computer science field. This empowers researchers to access knowledge from documents easily.

Given the enormous success of pretrained models to solve sequence-to-sequence problems, recent studies have proposed using these models for SPARQL query generation [11]. Huang et al. [26] proposed an approach based on fine-tuning large-scale pretrained encoder–decoder models to generate SPARQL queries from questions.

In the study of Das et al. [27], a neuro-symbolic approach called CBR-KBQA (case-based reasoning for knowledge-based question answering) was proposed. This method consists of three modules or phases: in the first module, dense representations of a query are computed using a RoBERTa-based (Robustly Optimized BERT Pretraining Approach) encoder [28]; then, they are used to retrieve other similar query representations from the training set. In the second module, Big Bird is used [29], which is a sparse-attention architecture that is initialized with pretrained BART-base weights to create a logical form of a natural-language query, leveraging the similar query representations extracted in the previous phase. Finally, in the third phase, the logical form is refined by aligning it to the relationships present in the knowledge graph. This model outperforms previous models regarding complex queries. In the same way, in the work of Wu et al. [30], the task of KGQA was also divided into three steps: hop prediction, where a classification model predicts the number of hops for a given query; relation path prediction, where a model is trained to predict the triples more related to the query considering the number of hops predicted in the first step; and triple sampling, where the triples are ranked.

For domain-specific data, Sima et al. [31] proposed a system based on several steps. Firstly, tokens of length one from the NL question are matched against the data stored in the database, the URI is retrieved alongside the class and property names, then these candidates are ranked and used to build a query graph that represents the possible answer to the NL question. This approach outperforms other systems in the biomedical domain. Additionally, Rangel et al. [32] proposed a data augmentation strategy to extend an existing set of queries to enable fine-tuning an LLM for translating NL to SPARQL queries in the life sciences domain, which shows an improvement in the performance of OpenLlama (https://github.com/openlm-research/open_llama, accessed on 8 June 2025).

Most of the works presented above require training data or a certain type of labeled data, which are not always available, especially for domain-specific knowledge graphs. For this reason, several approaches have explored prompting LLMs for KBQA. For instance, Baek et al. [33] proposed a knowledge augmentation approach for KGQA. This method first retrieves triples that match entities present in the natural-language questions and then includes this information in a prompt and asks an LLM to generate a new answer for the NL question. Liu et al. [2] proposed a contextual learning-based pipeline: first, the question in NL is vectorized, and similar questions along with the SPARQL query from the training set are extracted; then, these pairs are passed in a prompt along with the original question in NL and an instruction that provides an overview of the task. This approach achieves competitive results compared to other state-of-the-art LLM-based methods. However, a limitation of this work is the entity-linking prerequisite, which involves having the entity linking between the natural language and the KG conducted in the preliminary step. Therefore, if no such annotation or tool is available, the performance of the method can

decrease. A different approach was introduced by Tian et al. [34], encoding a subgraph of relevant information to a given question with a graph neural network, which helps to capture the structural and semantic components of the natural-language query. Then, the output is converted into a format the LLM can understand and passed as a prompt to it to guide its answering process.

In prompt-based methods, it is crucial to retrieve relevant information to answer a given question, and even these approaches may require the usage of a training or standard set [2]. In this work, we attempted to include contextual knowledge in a prompt similarly to previous approaches, but without involving labeled data. We proposed to leverage LLMs to first extract the information from the NL question. Then, we looked for relevant triples in the KG that could aid in the SPARQL query generation. Next, we generated a few examples of input and expected output and combined them in a new prompt, alongside a general overview of the KG and a description of the task. Unlike previous approaches, we attempted to combine in a single prompt (a) a specific context, giving triples that mention entities from the original question, (b) the general context explaining how the KG is organized (depending on the entities extracted from the query), and (c) pairs of contexts with the expected answer to exemplify the task. This approach enables the LLM to gain a high-level understanding of the target task by observing the expected outputs, while also learning about the general structure of the KG. As a result, this method remains flexible and adaptable to other domain-specific KGs.

3. Methodology

This work proposes a method consisting of three main phases. In Figure 1, a general overview of the approach is shown. In the first phase, given a question in natural language, the main entities are extracted through few-shot prompting of an LLM. Then, in the second phase, these entities are used to get the triples from the knowledge graph where these entities are mentioned. Finally, in phase three, the LLM is prompted to build a SPARQL query given a natural-language question, the relevant context retrieved in phase two, and several examples of the task.

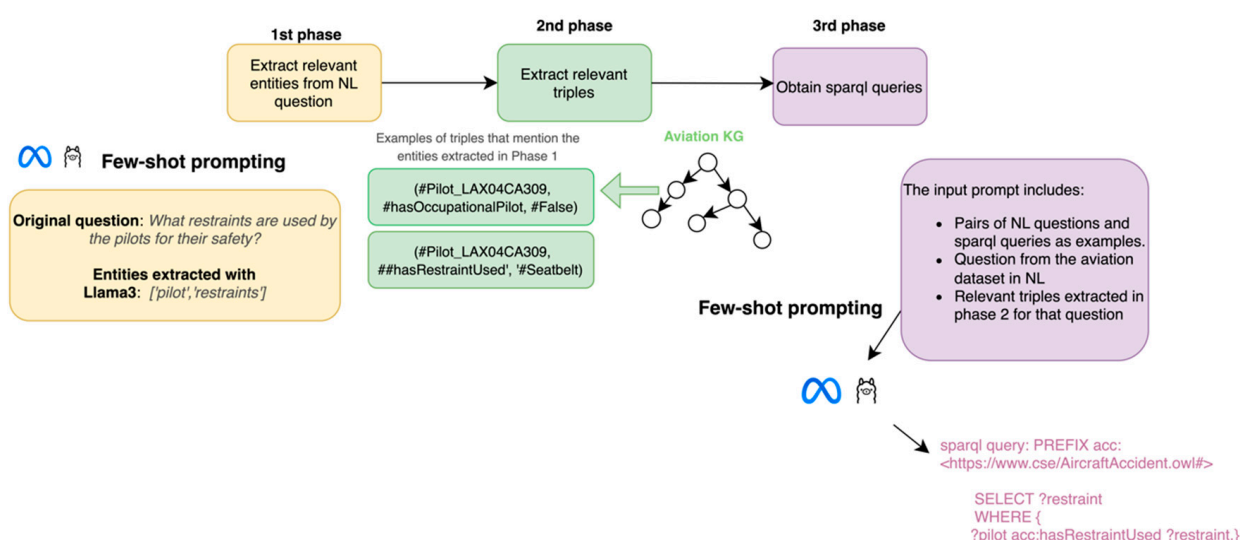


Figure 1. General description of the proposed approach.

For the first and third phases, we prompted a 4-bit AWQ-quantized LLaMA 3.1-70B Instruct model (<https://huggingface.co/hugging-quants/Meta-Llama-3.1-70B-Instruct-AWQ-INT4>, accessed on 8 June 2025) [35] accessed through the HuggingFace library [36]. This quantized variant was selected primarily due to computational resource constraints.

The Llama3 series was chosen due to its remarkable performance in natural language tasks as well as its open-source availability, which supports transparency and reproducibility. These series have recently demonstrated superior capabilities compared to other open-source alternatives in tasks involving prompt-based SPARQL query generation [37,38]. All the setup configurations for our experiments are available in our GitHub repository (<https://github.com/Ineshercam/PromptBasedQueryGeneration.git>, accessed on 8 June 2025).

In this section, first, an introduction to the knowledge graph used in this work is presented, then the three phases of the proposed methodology are explained.

3.1. Knowledge Graph and Test Set

According to the International Civil Aviation Organization (ICAO), aviation incidents fluctuated between 1.87 and 2.94 per million flights between 2019 and 2023 [39]. Given the complexity and critical nature of aviation safety, accident reports represent crucial sources of knowledge for identifying causes of accidents, although these reports are generally unstructured texts [40]. To address this, Agarwal et al. [17] proposed an aviation knowledge graph constructed from several accident reports. The ontology was built with the aid of experts and the Accident Data Reporting (ADREP) taxonomy; then, they used regular expressions to extract triples from the reports. Table 1 shows a description of the final KG that the authors made available (https://github.com/RajGite/KG-assisted-DL-based-QA-in-Aviation-Domain/blob/main/Aviation%20Knowledge%20Graph/Aviation_KG.owl, accessed on 8 June 2025).

Table 1. Properties of the aviation knowledge graph.

Metrics	Count
Entity class	138
Individual	8894
Object property	88
Data property	70
Axioms	96,699

The authors additionally provided a dataset of 150 questions along with their answers for 50 incident reports. This test set was manually curated by searching for every query in the 50 reports and documenting the desired answer and paragraph where it was found.

3.2. Extracting Relevant Entities

The first step of our methodology is to extract relevant entities from the natural-language question. The goal of this step is to identify triples in the KG that can guide a model in building a coherent SPARQL query. Given that this is an approach that targets domain-specific KGs, we determined that traditional named-entity recognition (NER) systems would not be sufficient. This is because such systems are typically trained to identify standard named entities (e.g., people, locations, organizations) and would fail at recognizing relevant but non-traditional entities, such as “injury” or “restraint,” which are crucial for our task. Furthermore, to ensure that the approach proposed in this work is adaptable across different domain-specific KGs, it was essential to adopt an entity extraction method that does not rely on domain-specific training data.

For these reasons, we employed an LLM-based approach using few-shot prompting. This approach was selected specifically for its adaptability and flexibility: by providing a small number of crafted examples, we were able to guide the LLM to extract the types of entities relevant to our task, including non-traditional entities that could be overlooked

by conventional NER systems. When compared to other methods, few-shot prompting offers a domain-agnostic solution that can generalize to new domains without the need for retraining, just by adapting the examples included in the prompt to a different domain.

Specifically, Llama3-70b-Instruct was leveraged through few-shot prompting to extract relevant entities from all the questions on the test set. The context window of this specific model is 128,000 tokens, which allows for sufficient information to be added.

In Figure 2, an example of the prompt presented for each question is shown. First, an instruction of the task is inserted; then, three examples of the expected output are shown to the model to guide it in understanding the task; and finally, a new question is added for the model to extract the entities. Although several tools exist for performing traditional named-entity recognition, they typically focus on standard entities, such as persons, organizations, etc. However, our task required the identification of all the words or entities that could aid in finding relevant triples.

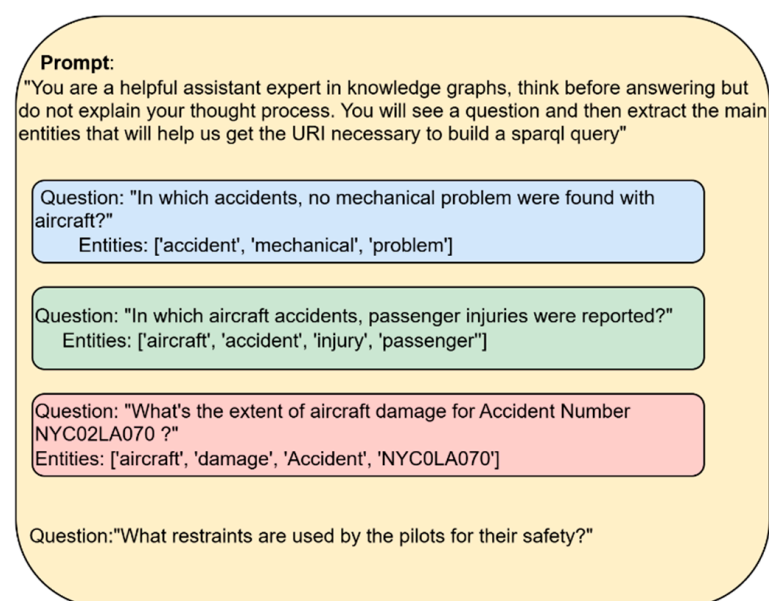


Figure 2. Example of the prompt used to extract relevant entities from an NL question.

We selected three examples that could represent some common cases present in the dataset, which should approximate real-life natural-language questions. Given the presence of multiple types of entities within these questions, we aimed to choose examples that include a diverse set of frequently occurring entity types, such as “accident,” “number of accident” (e.g., NYC0LA070), “aircraft,” “passenger,” etc. Rather than adhering to a rigid selection protocol, our goal was to ensure that the examples collectively captured both the structural and semantic variability commonly found in the dataset for the model to generalize more effectively.

After obtaining all the entities for each question, we lemmatized them since there were certain cases where the suffix of the word differed from the ending of those in the knowledge graph. For instance, in the question “Were any injuries reported for registration N7878N?”, Llama3 extracted the word “injuries” as an entity; however, in the aviation KG, there are relations such as “HasPassangerInjury,” which would not match with “injuries” if we did not lemmatize it. By lemmatizing the word, it was reduced to the form “injur,” which aligned with the relation’s name (“HasPassangerInjury” (please note that “Passanger” is a typo included in the original work [17]; since we wanted to be faithful to the original work where the KG was introduced, we present the entities and relations as they are expressed in the original work)) and allowed for a successful match. For multiword entities, we

lemmatized each word of the entity individually and then combined them in the same entity again.

3.3. Extracting Triples

This phase can be divided into two parts. The first part consists of building a general view of the KG. The second part encompasses the process of extracting the triples where the entities obtained in the previous phase are mentioned.

To provide the LLM with a general understanding of the knowledge graph, it is important to assess its content focus, and the most straightforward way to do this is to look at the size of its classes and the level of detail of representation of the instances of these classes [41]. Therefore, in the first part of this phase, we observed the classes with the most instances and which classes had the instances with the most connections. Table 2 displays the five classes with the most instances and the average number of relations per instance for each class. As observed, the classes with the highest average number of unique relations that also had a high number of instances were “Accident_number,” “Pilot,” and “AircraftRegistrationNumber.”

Table 2. Classes with the most instances and average number of relations of the instances of that class on the aviation knowledge graph.

Classes with the Most Instances	No. of Instances	Average No. of Relations	Min. No. of Relations	Max. No. of Relations
Accident_number	3775	13.74	2	78
AirportName	1088	1.31	1	2
Cause	639	2.01	1	3
Pilot	500	16	16	16
AircraftRegistrationNumber	498	28	28	28

We took an example of an instance of each of these three classes that covered all of the unique relations and registered them in a dictionary-like structure, which we called a “premade context”, where each class is a key and the values are (a) a general description of the class and (b) the list of triples that the example instance has. Thus, we first checked if, for a given question, any of the entities retrieved in the first phase matched the keys of the premade context dictionary. If they did, then we directly retrieved the information related to that entity from there, joining the description and the triples into a single string. Then, if an entity did not match the keys, we checked if it matched one of the elements of the triples of the whole premade context. If it did, then we added the general description of the key class and the triples that the entity matched. If the entity did not match any information from the premade context, we checked in the general graph for triples that matched the entity.

The second part of this phase is checking for entities in the whole graph. We built a SPARQL query to retrieve all those triples from the KG that mention the retrieved entities either in the predicate or in the object and grouped them by predicate and object, respectively. In this way, we limited the results to one example per object or predicate. When the entity is a multiword, the process becomes more complex. The process for such cases is exemplified in Figure 3. In this knowledge graph, some multiword expressions appear separated by underscores, such as “PILOT_IN_COMMAND.” However, in other cases, there is no separation, such as in “WEATHERCONDITION.” For this reason, it was necessary to check both possibilities for all multiword entities, as shown in Figure 3. If this did not match any predicates or objects, we checked if the entity could be partially matched, and if this failed, then we checked word by word.

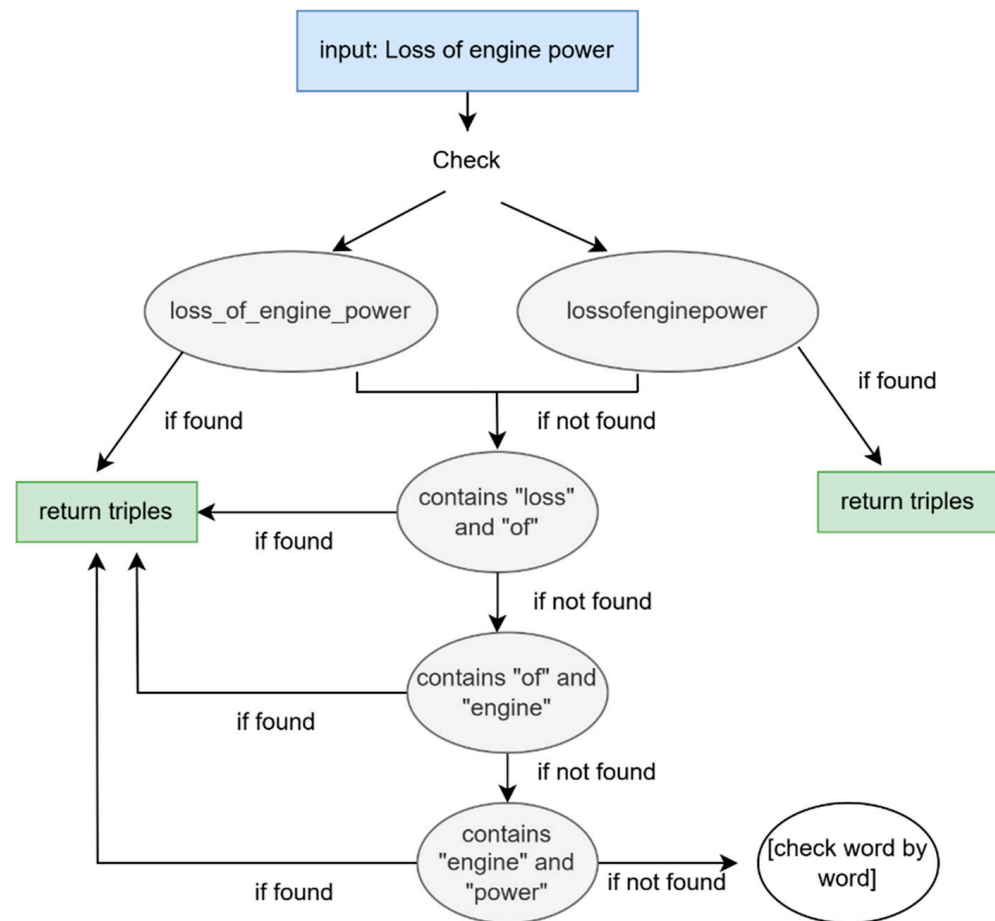


Figure 3. Example of getting relevant triples for multiword entities.

To illustrate the whole process of this phase, we provide the workflow in Figure 4. Consider the question “Which accidents caused substantial damage to aircraft?”, which would have the entities “accident,” “substantial damage,” and “aircraft.” Since “accident” is a key in the premade context dictionary, we extract the triples from there. Then, if “substantial damage” is not found in the dictionary keys, we check if it is present in any triples of the three entities that are keys in the dictionary. If this is not the case, then we proceed with the process explained in Figure 3, because it is a multiword entity. Finally, we check for the last entity (“aircraft”), which is a key in the premade queries dictionary, so we directly retrieve its associated triples.

Now, consider a hypothetical variation if the last entity was “injury” instead of “aircraft.” The entity “injury” does not appear as a key in the premade dictionary. However, it does match a subset of triples under the “accident” key. In this case, we retrieve only those “accident” triples that mention “injury” and discard the rest (what was added for the first entity “accident” at the beginning). The general description for “accident” is kept, but we narrow the result to the most specific context related to “injury”. This is done to ensure that irrelevant or overly broad information (i.e., noise) is excluded from the prompt, focusing instead on the most relevant and precise context extracted from the knowledge graph.

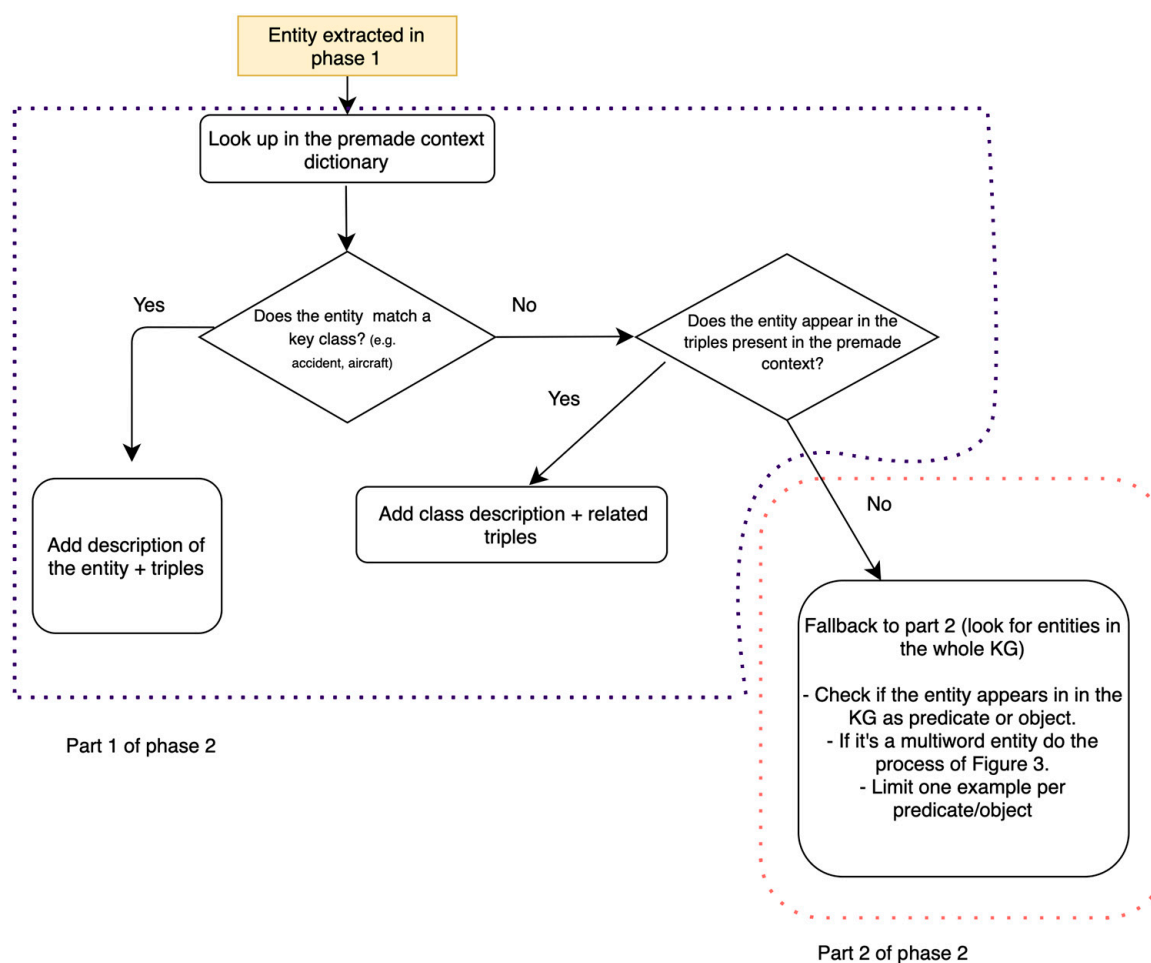


Figure 4. Workflow diagram of phase 2 (extracting triples) of the proposed method.

3.4. Obtaining SPARQL Queries

This step consisted of obtaining a SPARQL query that would retrieve the necessary information from a KG to answer a natural-language question. In the previous two phases, the goal was to extract relevant information from the question that could guide the model in building a correct query.

For this phase, as explained before, we prompted a 4-bit AWQ-quantized LLaMA 3.1-70B Instruct model. The prompt included (a) a description of the task, (b) four examples of the task (few-shot), (c) the question in NL, and (d) the context retrieved in phase 2 for this question.

In Figure 5, an example of the prompt is presented, with the first two boxes representing the expected input and output to illustrate the task. We manually crafted three examples that were similar to those present in the test set (in the image, only two are shown); then, a corresponding SPARQL query was built to exemplify the expected output and guide the model into building coherent sequences. These prompt examples can be found in the provided repository.

Following the presentation of three input–output examples, a new NL question is inserted. At this stage, additional contextual information, extracted during phase 2, is also provided (depicted in the box on the right in Figure 5). This supplementary information comprises relevant elements required to construct a SPARQL query capable of answering the newly introduced NL question. The effectiveness of this step is dependent on the entities identified during phase 1, as they serve as the foundation for linking the NL input to the corresponding knowledge graph triples.

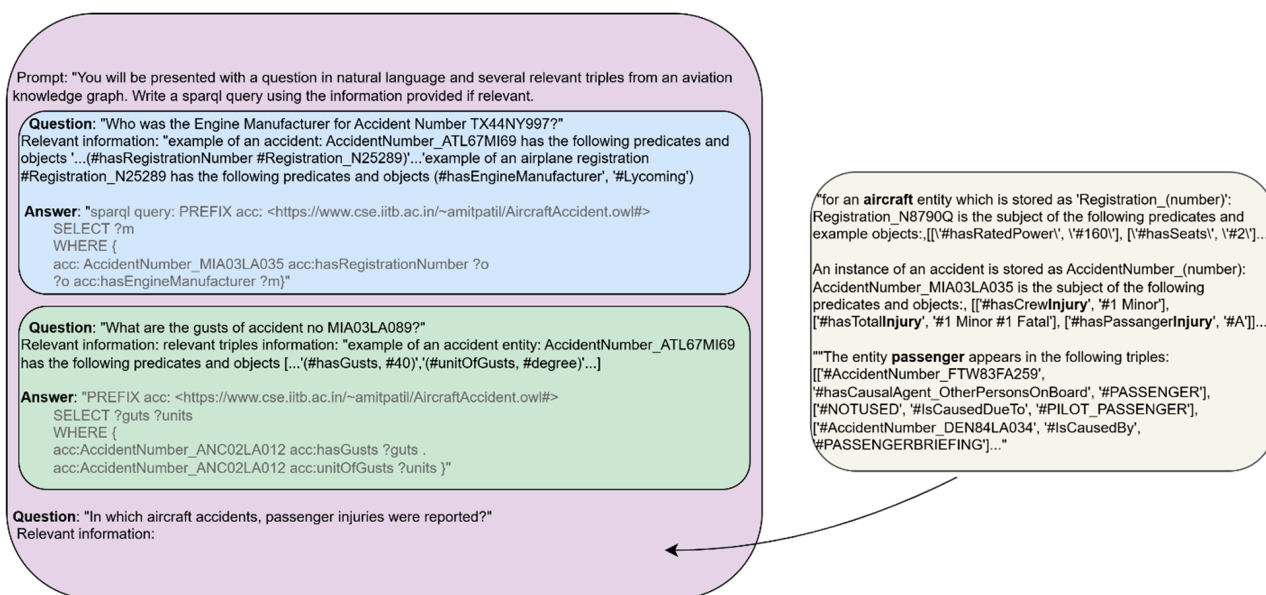


Figure 5. Example of the prompt used to obtain SPARQL queries.

After obtaining the SPARQL queries, minor postprocessing steps were applied before executing the queries. A space after the definition of each variable was added when there was not already one there (e.g., correcting and changing "SELECT?x" to "SELECT ?x"), and, if present, additional text that did not belong to the query was removed (e.g., "sparql query:" before the actual query or any additional explanation of the reasoning steps). The total of 150 natural-language queries in the test set (described in Section 3.1) were transformed into SPARQL queries and executed against the aviation knowledge graph using the RDFlib (<https://github.com/RDFLib/rdfliib>, accessed on 8 June 2025) library. We provide the results of the evaluation in the following section.

4. Results

In Table 3, the average running time of phase 1 (entity extraction) and phase 3 (query generation) per question of our method is displayed. All prompting experiments were conducted using a system equipped with three NVIDIA Quadro RTX 5000 GPUs (NVIDIA Corporation, Santa Clara, CA, USA).

Table 3. Average processing time (in seconds) per question for each prompting experiment. The results indicate that the query generation phase is substantially more time-consuming when compared to the entity extraction phase.

Task	Average Time (seconds)
Entity extraction	8.2
SPARQL query generation	30.0

After running the SPARQL queries obtained, we compared the results of the KG retrieval with the answers in the test set. It is crucial to highlight that the test set was created by extracting the information from the original documents and not from the KG. Therefore, as the KG is built upon the same collection of documents, we expected it to contain sufficient information to answer the test set questions. However, Agarwal et al. [17] stated that not all questions in the test set can be answered using the knowledge graph (KG), making it impossible to achieve an accuracy or exact match of 1.0. Following that, we manually annotated how many of the questions were based only on the facts present

in the KG. Thus, we additionally display the results considering only this part of the test set in Table 4. In line with the work of Agarwal et al. [17], we provide the value of the exact match metric, which calculates the number of answers predicted that fully match the golden answer. Additionally, since their system outputs multiple possible answers, they also introduced semantic accuracy, which considers an answer as correct if any of the top 10 predicted answers are semantically similar to the gold standard answer. However, as our system outputs only one predicted answer, we did not calculate the semantic accuracy in our work.

Table 4. Exact match of SPARQL query generation using the baseline KGQA method and our proposed context-aware prompting approach. The results are reported both for the full exact match and for the exact match based solely on the answers that are possible to answer with the KG.

Method	Exact Match	Exact Match (Only on KG Answers)
KGQA (baseline)	0.35	-
Context-aware prompting (our method)	0.41	0.55

Similarly to the exact match explained before, we calculated this metric by counting when our method retrieved the exact node(-s) necessary to answer the question, and the nodes matched the reference answer of the test set. Since the test set was created only considering a part of the documents, many answers would include more information than the answer in the test set. For instance, for the test set query “Accidents which have pilots older than 50 years,” the test set answer suggested a smaller number than the one found in the KG since the target answer only comprised a subset of documents. For this reason, we counted the prediction as correct if all the instances of the target answer were present in the predicted answer.

Additionally, questions involving numerical data (e.g., dew point, temperature) required normalization procedures due to substantial variation of the representations found in the test set, where more than one variation could be found, and in the knowledge graph. For instance, references to “nautical miles” appeared in different forms, such as “Nm” and “Nautical Miles,” while in the KG, it is represented as “nauticalMile.” To ensure an accurate evaluation of the model outputs, these variants were normalized to a consistent form in the test set. A table listing these variants and the normalized version is presented in Table A1.

As shown in Table 4, our method outperforms the approach used by Agarwal et al. [17], in which the aviation KG was introduced. We compared our results only with their KGQA module, since in this work, only the KG was used, and we wanted to assess the query generation capabilities of this method. In Table 4, we additionally present the results excluding unanswerable queries displayed in the third column “Exact match (only on KG answers).”

They also introduced other techniques that combine both the KG and the original documents to answer the questions; however, this includes additional external knowledge besides query generation.

Out of the 150 natural-language questions of the test, Llama generated syntactically incorrect queries only for three of them. Two of these errors were caused by introducing special characters in the instances using a namespace prefix. In Figure 6, an example is displayed: first, the prefix “acc” was inserted to avoid repeating the full URIs in the query; then, in the “FILTER” step (at the end of the query), the entity “UNSUITABLETERRAIN-ORTAKEOFF/LANDING/TAXIAREA” was introduced, since that was the name that was shown to the LLM in the triples examples included in the prompt and it was extracted from the KG. However, SPARQL does not accept the character “/,” which leads to a query

parsing error. The other malformed query inserted spaces in the definition of the entity, which is only possible for literals; therefore, it was incorrect.

Which terrain conditions could result in aircraft accident?

```
PREFIX acc:
<https://www.cse.iitb.ac.in/~amitpatil/AircraftAccident.owl#>

SELECT ?terrain

WHERE { ?acc a acc:Accident.

?acc acc:IsCausedBy ?terrain.

FILTER (strstarts(str(?terrain),
str(acc:UNSUITABLETERRAINORTAKEOFF/LANDING/TAXIAREA))) }
```

Figure 6. Example of a query syntax error produced with our method.

Nevertheless, most of the queries generated by the model were syntactically correct, although not all of them managed to retrieve the correct information. Furthermore, 58 queries did not return anything after execution because the query included non-existent relations or non-existent literals. Notably, out of these 58 empty outputs, 34 corresponded to questions that could not be answered with just the information in the KG.

5. Discussion

Our method outperforms the proposed approach of the original paper where the aviation KG was introduced [17].

Some of the questions cannot be answered based only on the information from the knowledge graph. For instance, more general questions, not related to an accident, such as “Who handles airspeed in aircraft?” are based on facts not included in the KG. Additionally, there are answers in the test set that do not match the answers found in the knowledge graph; therefore, even if the query generated by our method is correct, the answer does not match the test set answer, so it is counted as incorrect. Table 5 displays some examples where the target answer did not match the answer found in the KG. The first example introduces the question “Find the dew point for accident number ANC02LA012,” and the test set indicates “−2 °C” as the answer. However, in the knowledge graph, the answer is “1000,” and the answer retrieved by our method is [‘1000’, ‘degreeCelsius’]. Thus, our method retrieved the correct nodes to answer, but the nodes contained incorrect information, probably generated during the KG-building process. This could explain low values in the exact match, along with the several questions that are not possible to answer only with the KG.

We conducted a manual error analysis and classification to gain deeper insight into the specific challenges faced by our method. Table 6 presents the distribution of errors across the defined categories.

Table 5. Comparison between the selected gold standard answers and the corresponding information retrieved from the KG. The table highlights discrepancies that may arise due to inconsistencies in the KG relative to the reference (golden) answers.

Test Set Question	Test Set Answer	KG Information	Our Method's Output
Find the dew point for accident number ANC02LA012	−2 °C	1000	['1000 degreeCelsius']
When was the FAA exam conducted for Accident Number ANC02LA012?	37,155	2001-09-21T00:00:00	'2001-09-21T00:00:00'
Find the dew point for accident number NYC02LA070	−3 °C	3 degreeCelcius	'3 degreeCelsius'
What was the cause of accident in Accident Number ANC02LA012?	unsuitable terrain	<IsCausedBy #TERRAINCONDITION"/> <IsCausedBy #AIRPORTFACILITIESRUNWAY/LANDINGAREACONDITION"/> <IsCausedBy UNSUITABLETERRAINORTAKEOFF/LANDING/TAXIAREA"/>	'unsuitableterrainortakeoff/landing/taxiarea', 'airportfacilitiesrunway/landingareacondition', 'terraincondition'

Table 6. Error analysis and categorization of the outputs generated by our method. The asterisk in number 3 indicates that two of the three syntax errors are also found in the unanswerable queries; as a result, the total sum of the percentages is not 100%.

Type of Error	Quantity	Percentage of Total Errors
Syntax errors	3 *	3.4
Unanswerable queries	39	44.3
Incorrect information in the KG	9	10.2
Semantic–lexical errors	26	29.5
Logical errors	11	12.5
Incorrect information retrieval	2	2.3

- Syntax errors refer to those queries that are syntactically incorrect and cannot be executed.
- Unanswerable queries refer to those questions that are impossible to answer only with the information in the KG. As previously discussed, the KG contains less information than the original accident reports, resulting in a lack of coverage for certain questions.
- Incorrect information in the KG includes cases where the generated query correctly retrieves the intended node(s), but the factual content in the KG does not correspond to the ground-truth answer in the test set (as shown in Table 5). These errors can be attributed to inaccuracies or inconsistencies during the KG construction process.
- Semantic–lexical errors refer to cases where a query contains terms (typically predicates, classes, or individuals) that are lexically invalid or not present in the underlying ontology or vocabulary, resulting in semantic invalidity, even though the syntax of the query itself is correct.
- Logical errors occur when even if the terms used and syntax are correct, the logic or structure does not align with the structure of the KG due to reasoning issues.
- Incorrect information retrieval refers to queries that are syntactically correct and use valid ontology terms but fail to select the precise elements required by the question.

As mentioned before, only three queries produced by our method were syntactically incorrect, two of which also fell into the unanswerable queries category. In total, 39 queries could not be answered only with the information contained in the KG. Moreover, nine queries, despite being semantically and lexically correct, retrieved information that did not match the test set due to inaccuracies in the KG itself. These outputs, although we did not count them as valid for the exact match calculation, are still considered valid from the perspective of query generation, as they demonstrate proper structure and intent. This mismatch arises from data quality issues in the KG.

The most common errors regarding the performance of our method were semantic–lexical errors. Within this category, we identified that approximately 65% of this type of errors occurred because the model confused literals and entities. For instance, in the case illustrated in Figure 7, the model misinterpreted the aircraft model “R172E” as a string literal instead of recognizing it as a defined entity in the KG, leading to an empty result, even though an example showing “R172E” as an entity was included in the prompt. The other 35% of these errors encompassed the use of non-existing terms, either invalid predicates or undefined entities.

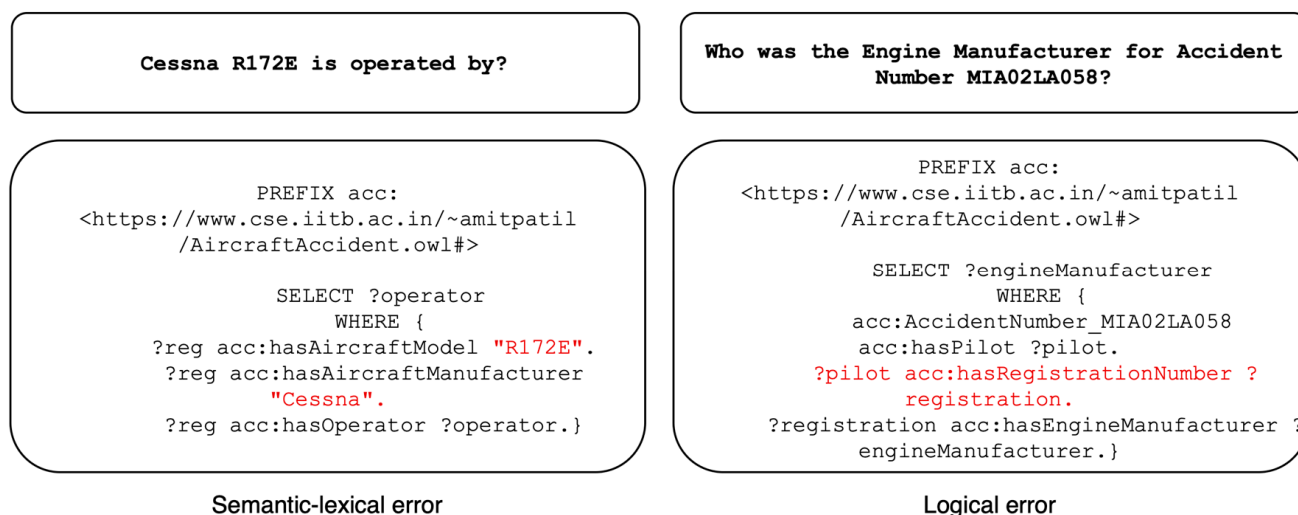


Figure 7. Examples of semantical–lexical and logical errors.

Logical errors represented approximately 12.5% of the total errors. These included the use of semantically correct predicates in contexts where they were logically inappropriate, incorrect selection of relationships to retrieve the desired information, and the use of regular expression matching without case sensitivity when it was needed. As shown in Figure 7, one such error involved using the relation “hasRegistrationNumber” to find the registration number; however, it is accidents that have a registration number and not pilots.

Despite these challenges, this proposed approach managed to successfully retrieve multiple answers, including answers where multistep reasoning was needed, such as those including a metric and the unit of the metric, or answers that required multi-hop search queries.

Figure 8 presents two examples of correctly generated queries. In the leftmost example, the NL question inquired about the type of landing gear involved in a specific accident. The model successfully identified the need to first retrieve the registration number associated with the accident, followed by querying the landing gear via the “hasLandingGear” relation. This reasoning process appears to be influenced by the exemplar triples provided in the prompt, which included the relation [“#Registration_N633KF,” “#hasLandingGear,” “#Tailwheel”]. In the initial phase, the phrase “landing gear” was extracted as an entity and subsequently matched to relevant triples, including the aforementioned example, thus guiding the generation of the correct query. Furthermore, the general context prompt included patterns indicating that accidents are linked to registration numbers, reinforcing the appropriate relation chain.

The second example in Figure 8 (on the right) involves an NL query about the wind speed during a specific accident. Here, the model accurately retrieved both the speed value and its units. Notably, while the term “speed” does not explicitly appear in the question, the model effectively inferred that the phrase “how fast” semantically corresponds to “wind speed,” which matched the necessary triples to retrieve the correct information.

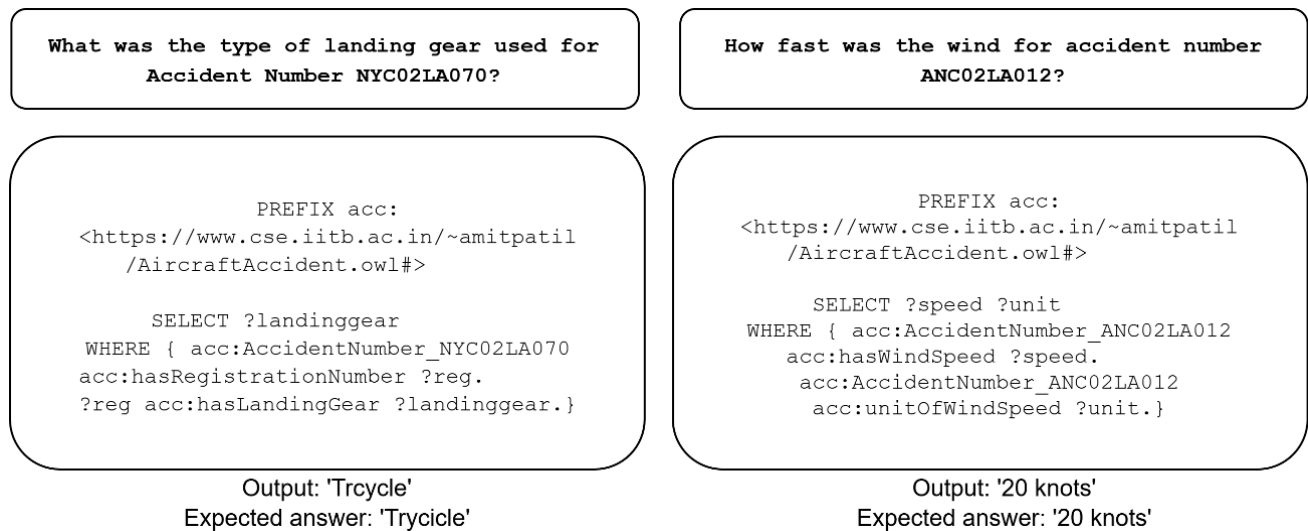


Figure 8. Examples of NL questions, the correctly generated queries by our method, and their output after execution and normalization.

It is important to note that several errors were attributable not to the model or the method itself but to the flaws of the underlying KG. Therefore, we advise against using this particular KG as a benchmark for domain-specific query generation without substantial refinement. Considerable manual intervention was required to properly assess and validate our method's performance. Even with this intervention, several correct queries produced by our method were not recognized as correct, complicating a fair assessment of its true capabilities.

Unlike previous approaches, the method proposed in this work does not rely on training or labeled data. Liu et al. [2] proposed a prompt-based method to generate SPARQL queries; however, a labeled set is needed to find similar question–SPARQL query pairs for a new NL question to include them in a prompt. Our method, unlike the aforementioned work, proposes leveraging the general reasoning capabilities of LLMs to extract relevant entities and match these with triples present in the KG for domain-specific scenarios where labeled data are sparse. The introduction of general and contextual knowledge allows the model to generate correct SPARQL queries tailored to each NL question.

Injecting KG-grounded information has proven to enhance KBQA, such as in the work of Baek et al. [33]. The method proposed by these authors injects triples that match the entities present in the NL directly into the LLM to generate an answer. However, as indicated in the limitations, this method relies heavily on the quality of the retrieval triples, struggles when multi-hop reasoning is necessary, and might retrieve information that is solely close to that entity. In our proposed approach, we leverage the reasoning capabilities of LLMs to generate a SPARQL query, which includes multi-hop reasoning if the question requires it. This query retrieves only the information necessary to answer the NL question, and this can be further used to enhance KBQA.

A key innovation of our approach compared to the previous baseline is the use of context-aware prompting: in the query generation phase, the LLM is not only given the question, but it is also enriched with relevant triples and a few carefully selected in-context examples extracted by analyzing the most relevant classes in the KG. This contextual guidance enables the model to better understand the intent, align with the KG schema, and generate structurally and semantically accurate queries. In our experiments, the token limit of the Llama3-70b model was never exceeded. However, if this is the case for other KGs, we recommend selecting the top-n most similar triples to the NL question. Furthermore, entity extraction is handled through few-shot prompting, enabling the model to identify

key concepts in complex phrasings without the need to use rule-based constraints present in the previous baseline. These advances lead to significantly improved performance of 6% in the exact match metric.

6. Conclusions

In this work, we proposed a three-step prompt-based approach for generating SPARQL queries from natural-language questions. Our method was evaluated on an aviation knowledge graph (KG), achieving a 6% improvement in the exact match metric when compared to prior results from the original KG study. It is important to acknowledge that this metric does not fully capture the overall effectiveness of our approach due to errors present in the test set. While our method remains susceptible to inaccuracies, as it relies on the language model's understanding of SPARQL, it offers significant potential for refinement. This method is adaptable to domain-specific KGs without requiring knowledge from experts, making it a promising direction for further research and development.

Certain limitations of this work and possibilities for future work include refining the triple-matching process. Questions asked by users may vary greatly; thus, it is necessary to search for semantically similar entities, given that the entities found in the question not always match elements from the knowledge graph or that there could be ambiguous entities. Contextual embeddings could be leveraged instead of lexical matching to extract relevant triples from the KG. Additionally, analyzing the impact of both the number and structure of different few-shot examples can be explored to understand how the results would be affected.

Similarly, since the design of our prompt forced the model to always extract entities and our triple-matching technique was flexible, our method extracted entities and retrieved triples for each given example. However, if the prompt is modified to suggest no entity extraction in the case where no relevant entities are found, several techniques, such as question rephrasing, might be needed.

Additionally, one potential strategy to address semantic–lexical errors (particularly, the most frequent type, which involves the confusion between literals and resources) is to enhance the prompt used in the third phase. This can be performed by explicitly specifying the nature of the terms in the example triples provided to the model, clearly indicating which elements are literals and which are resources. This clarification is especially important for the triples retrieved using the extracted entities, as it can help the model distinguish between different types of values and generate more accurate SPARQL queries.

Author Contributions: Conceptualization: I.-V.H.-C., E.G.-L., A.G.-C. and S.C.-A.; methodology: I.-V.H.-C. and E.G.-L.; software: I.-V.H.-C.; data curation: I.-V.H.-C.; investigation: I.-V.H.-C., E.G.-L., A.G.-C. and S.C.-A.; writing—original draft preparation: I.-V.H.-C., E.G.-L. and A.G.-C.; writing—review & editing: E.G.-L., A.G.-C., S.C.-A. and I.-V.H.-C.; supervision: A.G.-C. and E.G.-L.; funding acquisition: A.G.-C., E.G.-L. and S.C.-A. All authors have read and agreed to the published version of the manuscript.

Funding: This work was co-funded by the IA3: Research Chair of Artificial Intelligence in Aeronautics and Aerospace (Cátedra de Inteligencia Artificial en Aeronáutica y Aeroespacio, TSI-100920-2023-1) through the ENIA Research Chair Grants under the framework of the Spain Digital Agenda 2025 and the Recovery, Transformation and Resilience Plan funded by the European Union—Next Generation EU; and the project “Tecnologías Inteligentes para la Fabricación, el diseño y las Operaciones en entornos iNdustriales” (TIFON, PLEC2023-010251), through the call Proyectos de I+D+i en líneas estratégicas—Transmisiones 2023.

Data Availability Statement: Dataset available on request from the authors.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Table A1. Normalization process of the numeric terms found in the test set.

Variants Found in the Test Set	Variant of the KG	Normalized Version
Nm, Nautical Mile	nauticalMile	nm
ft	foot	ft
knots	knot	knots
°C	degreeCelsius	degrees celsius

References

- Huang, W.; Zhou, G.; Lapata, M.; Vougiouklis, P.; Montella, S.; Pan, J.Z. Prompting Large Language Models with Knowledge Graphs for Question Answering Involving Long-Tail Facts. *Knowl.-Based Syst.* **2024**, *324*, 113648. [\[CrossRef\]](#)
- Liu, Y.; Li, Z.; Jin, X.; Guo, Y.; Bai, L.; Guan, S.; Guo, J.; Cheng, X. An In-Context Schema Understanding Method for Knowledge Base Question Answering. *Int. Conf. Knowl. Sci. Eng. Manag.* **2024**, *14884*, 419–434.
- Wei, Y.; Huang, Q.; Zhang, Y.; Kwok, J. KICGPT: Large Language Model with Knowledge in Context for Knowledge Graph Completion. In *Findings of the Association for Computational Linguistics: EMNLP 2023*; Association for Computational Linguistics: Singapore, 2023; pp. 8667–8683.
- Ali, W.; Saleem, M.; Yao, B.; Hogan, A.; Ngomo, A.-C.N. A survey of RDF stores & SPARQL engines for querying knowledge graphs. *VLDB J.* **2022**, *31*, 1–26. [\[CrossRef\]](#)
- Papadaki, M.-E.; Tzitzikas, Y.; Mountantonakis, M. A Brief Survey of Methods for Analytics over RDF Knowledge Graphs. *Analytics* **2023**, *2*, 55–74. [\[CrossRef\]](#)
- Hu, N.; Wu, Y.; Qi, G.; Min, D.; Chen, J.; Pan, J.Z.; Ali, Z. An Empirical Study of Pre-Trained Language Models in Simple Knowledge Graph Question Answering. *World Wide Web* **2023**, *26*, 2855–2886. [\[CrossRef\]](#)
- Wu, X.; Tsioutsoulis, K. Thinking with Knowledge Graphs: Enhancing LLM Reasoning Through Structured Data. *arXiv* **2024**, arXiv:2412.10654.
- Zhang, C.; Lai, Y.; Feng, Y.; Zhao, D. A review of deep learning in question answering over knowledge bases. *AI Open* **2021**, *2*, 205–215. [\[CrossRef\]](#)
- Lan, Y.; He, G.; Jiang, J.; Jiang, J.; Zhao, W.X.; Wen, J.R. A Survey on Complex Knowledge Base Question Answering: Methods, Challenges and Solutions. *arXiv* **2021**, arXiv:2105.11644.
- Haluptzok, P.; Bowers, M.; Kalai, A.T. Language Models Can Teach Themselves to Program Better. *arXiv* **2023**, arXiv:2207.14502.
- Diallo, P.A.K.K.; Rey, S.; Zouaq, A. A Comprehensive Evaluation of Neural SPARQL Query Generation From Natural Language Questions. *IEEE Access* **2024**, *12*, 125057–125078. [\[CrossRef\]](#)
- Klager, G.G.; Polleres, A. Is GPT fit for KGQA?—Preliminary Results. In *Proceedings of the TEXT2KG 2023: Second International Workshop on Knowledge Graph Generation from Text*, Hersonissos, Greece, 28 May–1 June 2023.
- Li, Z.; Fan, S.; Gu, Y.; Li, X.; Duan, Z.; Dong, B.; Liu, N.; Wang, J. FlexKBQA: A Flexible LLM-Powered Framework for Few-Shot Knowledge Base Question Answering. *Proc. AAAI Conf. Artif. Intell.* **2024**, *38*, 18608–18616. [\[CrossRef\]](#)
- Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Ichter, B.; Xia, F.; Chi, E.; Le, Q.V.; Zhou, D. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 24824–24837.
- Chen, W. Large Language Models are few(1)-shot Table Reasoners. In *Proceedings of the Findings of the Association for Computational Linguistics: EACL 2023*, Dubrovnik, Croatia, 2–6 May 2023; pp. 1120–1130.
- Xiong, G.; Bao, J.; Zhao, W. Interactive-KBQA: Multi-Turn Interactions for Knowledge Base Question Answering with Large Language Models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, Bangkok, Thailand, 11–16 August 2024; pp. 10561–10582.
- Agarwal, A.; Gite, R.; Laddha, S.; Bhattacharyya, P.; Kar, S.; Ekbal, A.; Thind, P.; Zele, R.; Shankar, R. Knowledge Graph—Deep Learning: A Case Study in Question Answering in Aviation Safety Domain. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, Marseille, France, 20–25 June 2022; pp. 6260–6270.
- Jaradeh, M.Y.; Oelen, A.; Farfar, K.E.; Prinz, M.; D’Souza, J.; Kismihók, G.; Stocker, M.; Auer, S. Open Research Knowledge Graph: Next Generation Infrastructure for Semantic Scholarly Knowledge. In *Proceedings of the 10th International Conference on Knowledge Capture*, Marina Del Rey, CA, USA, 19–21 November 2019; pp. 243–246.
- Hogan, A.; Blomqvist, E.; Cochez, M.; D’Amato, C.; De Melo, G.; Gutierrez, C.; Kirrane, S.; Gayo, J.E.L.; Navigli, R.; Neumaier, S.; et al. Knowledge Graphs. *ACM Comput. Surv.* **2022**, *54*, 1–37. [\[CrossRef\]](#)

20. Meloni, A.; Recupero, D.R.; Osborne, F.; Salatino, A.; Motta, E.; Vahadati, S.; Lehmann, J. Assessing Large Language Models for SPARQL Query Generation in Scientific Question Answering. In Proceedings of the Special Session on Harmonising Generative AI and Semantic Web Technologies (HGAIS 2024), Baltimore, MD, USA, 13 November 2024.
21. Meyer, L.-P.; Frey, J.; Brei, F.; Arndt, N. Assessing SPARQL capabilities of Large Language Models. *arXiv* **2025**, arXiv:2409.05925.
22. Bounhas, I.; Soudani, N.; Slimani, Y. Building a morpho-semantic knowledge graph for Arabic information retrieval. *Inf. Process. Manag.* **2020**, *57*, 102124. [[CrossRef](#)]
23. Peng, C.; Xia, F.; Naseriparsa, M.; Osborne, F. Knowledge Graphs: Opportunities and Challenges. *Artif. Intell. Rev.* **2023**, *56*, 13071–13102. [[CrossRef](#)] [[PubMed](#)]
24. Zheng, D.; Song, X.; Ma, C.; Tan, Z.; Ye, Z.; Dong, J.; Xiong, H.; Zhang, Z.; Karypis, G. DGL-KE: Training Knowledge Graph Embeddings at Scale. *arXiv* **2020**, arXiv:2004.08532.
25. Jia, R.; Zhang, B.; Méndez, S.J.R.; Omran, P.G. Leveraging Large Language Models for Semantic Query Processing in a Scholarly Knowledge Graph. *arXiv* **2024**, arXiv:2405.15374.
26. Huang, X.; Kim, J.-J.; Zou, B. Unseen Entity Handling in Complex Question Answering over Knowledge Base via Language Generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*; Association for Computational Linguistics: Punta Cana, Dominican Republic, 2021; pp. 547–557.
27. Das, R.; Zaheer, M.; Thai, D.; Godbole, A.; Perez, E.; Lee, J.Y.; Tan, L.; Polymenakos, L.; McCallum, A. Case-based Reasoning for Natural Language Queries over Knowledge Bases. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Punta Cana, Dominican Republic, 7–11 November 2021; pp. 9594–9611.
28. Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; Stoyanov, V. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv* **2019**, arXiv:1907.11692.
29. Zaheer, M.; Guruganesh, G.; Dubey, K.A.; Ainslie, J.; Alberti, C.; Ontanon, S.; Pham, P.; Ravula, A.; Wang, Q.; Yang, L.; et al. Big Bird: Transformers for Longer Sequences. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 17283–17297.
30. Wu, Y.; Hu, N.; Bi, S.; Qi, G.; Ren, J.; Xie, A.; Song, W. Retrieve-Rewrite-Answer: A KG-to-Text Enhanced LLMs Framework for Knowledge Graph Question Answering. *arXiv* **2023**, arXiv:2309.11206.
31. Sima, A.C.; Mendes de Farias, T.; Anisimova, M.; Dessimoz, C.; Robinson-Rechavi, M.; Zbinden, E.; Stockinger, K. Bio-SODA: Enabling Natural Language Question Answering over Knowledge Graphs without Training Data. In Proceedings of the 33rd International Conference on Scientific and Statistical Database Management, Tampa, FL, USA, 6–7 July 2021; pp. 61–72.
32. Rangel, J.C.; Farias TMde Sima, A.C.; Kobayashi, N. SPARQL Generation: An analysis on fine-tuning OpenLLaMA for Question Answering over a Life Science Knowledge Graph. *arXiv* **2024**, arXiv:2402.04627.
33. Baek, J.; Aji, A.F.; Saffari, A. Knowledge-Augmented Language Model Prompting for Zero-Shot Knowledge Graph Question Answering. In Proceedings of the 1st Workshop on Natural Language Reasoning and Structured Explanations (@ACL 2023), Toronto, ON, Canada, 13 June 2023; pp. 78–106.
34. Tian, Y.; Song, H.; Wang, Z.; Wang, H.; Hu, Z.; Wang, F.; Chawla, N.V.; Xu, P. Graph Neural Prompting with Large Language Models. *Proc. AAAI Conf. Artif. Intell.* **2024**, *38*, 19080–19088. [[CrossRef](#)]
35. Grattafiori, A.; Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; Al-Dahle, A.; Letman, A.; Mathur, A.; Schelten, A.; Vaughan, A.; et al. The Llama 3 Herd of Models. *arXiv* **2024**, arXiv:2407.21783.
36. Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; et al. HuggingFace’s Transformers: State-of-the-art Natural Language Processing. *arXiv* **2020**, arXiv:1910.03771.
37. Ben Amor, M.; Strappazzon, A.; Granitzer, M.; Egyed-Zsigmond, E.; Mitrović, J. Instruct-to-SPARQL: A text-to-SPARQL dataset for training SPARQL Agents. In Proceedings of the 2025 ACM SIGIR Conference on Human Information Interaction and Retrieval, Melbourne, Australia, 24–28 March 2025; pp. 390–395.
38. D’Abramo, J.; Zugarini, A.; Torroni, P. Investigating Large Language Models for Text-to-SPARQL Generation. In Proceedings of the 4th International Workshop on Knowledge-Augmented Methods for Natural Language Processing, Albuquerque, NM, USA, 3 May 2025; pp. 66–88.
39. ICAO. *ICAO Safety Report*, 2024th ed.; International Civil Aviation Organization: Montreal, QC, Canada, 2024.
40. Ren, T.; Zhang, Z.; Jia, B.; Zhang, S. Retrieval-Augmented Generation-aided causal identification of aviation accidents: A large language model methodology. *Expert Syst. Appl.* **2025**, *278*, 127306. [[CrossRef](#)]
41. Heist, N.; Hertling, S.; Ringler, D.; Paulheim, H. Knowledge Graphs on the Web—An Overview. *arXiv* **2020**, arXiv:2003.00719.

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