BuildingQA: A Benchmark for Natural Language Question Answering over Building Knowledge Graphs

Ozan Baris Mulayim*
omulayim@andrew.cmu.edu
Carnegie Mellon University
Pittsburgh, PA, United States

Avia Anwar
avia_anwar@mines.edu
Colorado School of Mines
Golden, CO, United States

Mete Saka
saka@mines.edu
Colorado School of Mines
Golden, CO, United States

Lazlo Paul
LPaul@lbl.gov
Lawrence Berkeley National Laboratory
Berkeley, CA, United States

Anand Krishnan Prakash
akprakash@lbl.gov
Lawrence Berkeley National Laboratory / Carnegie Mellon University
Berkeley, CA, United States

Gabe Fierro
gtfierro@mines.edu
Colorado School of Mines
Golden, CO, United States

Marco Pritoni
mpritoni@lbl.gov
Lawrence Berkeley National Laboratory
Berkeley, CA, United States

Mario Bergés†
marioberges@cmu.edu
Carnegie Mellon University
Pittsburgh, PA, United States

ABSTRACT

Graph-based representations of building metadata using ontologies like Brick are vital for smart building applications, but querying them remains a challenge for practitioners. Knowledge Graph Question Answering (KGQA) systems, meant to retrieve answers from natural language questions, traditionally require large-scale training data, making them ill-suited for the specialized and data-scarce building domain. The advent of Large Language Models (LLMs) offers a paradigm shift, enabling zero-shot natural language querying without building/domain-specific training. Yet, there is no standardized benchmark for building-specific KGQA which can guide and validate research in this area.

To address this gap, our work makes three primary contributions. First, we introduce the BuildingQA Benchmark Dataset, constructed through a multi-stage process of collecting practitioner data, augmenting it with LLMs for linguistic diversity, and curating a final set of 188 questions across 4 buildings. Second, we characterize the benchmark's complexity and ambiguity, introducing a novel method to quantify its "lexical gap" and providing a four-stage diagnostic framework for analyzing how systems fail. Third, we benchmark zero-shot LLM-powered KGQA systems to establish baseline performance and analyze their failure modes.

Our evaluation reveals that top-performing systems achieve a maximum F1 score of only 0.38. This result does not indicate a failure of these powerful systems, but rather underscores the unique challenges posed by our benchmark. It demonstrates a critical performance gap, showing that current methods successful on general KGs struggle with the specific lexical and structural nuances of the building domain. BuildingQA[1] thus provides the benchmark dataset and foundational analysis needed to drive the development of novel, domain-aware methods required to unlock the use of semantic data in buildings.

CCS CONCEPTS

• **Information systems** → **Question answering**; **Ontologies**; • **Computing methodologies** → **Machine learning**.

KEYWORDS

semantic ontologies, knowledge graphs, large language models

ACM Reference Format:
Ozan Baris Mulayim, Avia Anwar, Mete Saka, Lazlo Paul, Anand Krishnan Prakash, Gabe Fierro, Marco Pritoni, and Mario Bergés. 2025. BuildingQA: A Benchmark for Natural Language Question Answering over Building Knowledge Graphs. In *The 12th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation (BUILDSYS '25), November 19–21, 2025, Golden, CO, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3736425.3770097

1 INTRODUCTION

Graph-based representations of building metadata, powered by semantic ontologies like Brick [3], are foundational for enabling the next generation of smart building applications. By creating a standardized, semantically rich, machine-readable format for data about equipment, spaces, and their interrelationships, these models

*Corresponding Author
†Mario Bergés holds concurrent appointments as a Professor of Civil and Environmental Engineering at Carnegie Mellon University and as an Amazon Scholar. This paper describes work at Carnegie Mellon University and is not associated with Amazon.

[1]Our code and the dataset are available in https://github.com/INFERLab/BuildingQA.

BUILDSYS '25, November 19–21, 2025, Golden, CO, USA

ACM ISBN 979-8-4007-1945-5/2025/11.
https://doi.org/10.1145/3736425.3770097

promise to break down data silos and foster interoperability [12]. However, a significant knowledge gap hinders their widespread adoption. Creating and using knowledge graphs (KGs) requires specific technical expertise, creating a barrier for building operators and other practitioners who stand to benefit most from them [5].

Empowering operators to query these complex data structures using natural language is a promising solution, a task central to the field of Knowledge Graph Question Answering (KGQA). While traditional KGQA methods require extensive training data, the recent advent of Large Language Models (LLMs) has allowed the field to develop "zero-shot" KGQA tools [2]. Zero-shot KGQA offers a direct path for non-specialists to interact with building models without requiring training data for each building. However, the development of robust, domain-specific tools is critically hampered by the absence of a standardized benchmark dataset [20]. While such benchmarks drive progress in the broader KGQA community [11] and similar efforts are observed for building information models [32], building ontologies lack a standard evaluation dataset and process, resulting in the use of small, ad-hoc datasets that hamper the development and validation of generalizable solutions [19].

To address this gap, we introduce a suite of resources designed to advance research into natural language interfaces for building semantic models. Our work makes three primary contributions following a clear pipeline from data collection to evaluation:

(1) **The `BuildingQA` Benchmark Dataset.** We introduce the `BuildingQA` Benchmark Dataset (§3), a resource specifically designed for evaluating natural language interfaces in the building domain. This benchmark was constructed through a multi-stage process of collecting real-world data from practitioners, augmenting these examples using LLMs, and curating the final, high-quality set of 188 questions across 4 buildings.
(2) **Benchmark Characterization & Diagnostic Framework.** We provide a detailed characterization of the `BuildingQA` benchmark, analyzing its complexity (§4.1.1) and ambiguity in comparison to existing KGQA datasets. As part of this, we develop a novel method (§4.1.2) to quantify the "lexical gap"—the vocabulary mismatch between questions and graph terminology—and show it is significantly wider in the building domain. Furthermore, we introduce a novel *four-stage diagnostic framework* (§4.2) that provides informative metrics to understand *how* a system fails, not just if it does.
(3) **A Baseline Evaluation of KGQA Systems.** Finally, we establish an initial performance benchmark by evaluating several modern KGQA systems and LLMs on the `BuildingQA` dataset (§5). This contribution provides the community with baseline metrics for future comparison and demonstrates the significant challenges that our benchmark poses to current state-of-the-art methods.

We observed that the top-performing KGQA systems [19, 28] achieve a maximum F1 score of only 0.38. This significant performance drop does not indicate a failure of these powerful KGQA systems, but rather underscores the unique and unsolved challenges posed by our benchmark. It reveals a critical gap: methods that excel on general-KGs do not transfer effectively to the specialized vocabularies and complex relational patterns found in building operations. This result firmly establishes BuildingQA as a challenging and necessary benchmark for the research community. It provides a standardized tool to measure progress on these specific problems and highlights the urgent need for novel, domain-aware methods that can bridge this performance gap to unlock the full potential of semantic data in the built environment.

Finally, while this work focuses on semantic metadata for buildings, the BuildingQA resource opens the door to empirically investigating a more fundamental question: the evolving relationship between symbolic knowledge representations (like Brick models) and neural representations derived from LLMs. As LLM context windows grow, it is critical to understand the role of structured, formalized information in specialized domains. Our work provides a concrete testbed for comparing the performance of these different knowledge paradigms, paving the way for a deeper understanding of how they can most effectively be combined to unlock the full potential of data in the built environment.

2 BACKGROUND

This section provides the necessary context for understanding the challenges of building natural language interfaces for building semantic models. We first provide background on KGs and clarify how KGs in buildings differ from those typical of existing KGQA research. This contextualizes our review of the KGQA state-of-the-art and the landscape of existing evaluation datasets.

Ontologies, Knowledge Graphs and SPARQL. To understand the challenges of applying KGQA methods to building data, it is crucial to clarify the difference between an ontology and a KG. An ontology provides a formal, abstract representation of domain knowledge, defining the types of entities, their properties, and the relationships between them. This ensures consistency and enables sophisticated reasoning. A KG encodes specific knowledge by using classes and relationships from a given ontology, representing the instances and their interconnections within a domain [15]. KGs and ontologies are commonly represented using the Resource Description Framework (RDF) standard.

In the buildings domain, this distinction is exemplified by prominent ontologies like the Brick schema [3], a widely adopted open-source ontology, and the ASHRAE 223p, an emerging industry-backed standard [1]. For example, Brick defines classes like `VAV` and `Temperature_Sensor` and their possible relationships (e.g., `feeds`). A Brick *model* of a building is therefore a KG that uses the ontology's vocabulary to encode knowledge about the specific equipment, data sources, and relationships in that building.

SPARQL [13] is the standard query language RDF graphs, which works by matching graph patterns, defined in a WHERE clause, against the KG's structure. These patterns are composed of triple patterns (?subject ?predicate ?object), which use variables (denoted by ?) to find entities and the relationships between them. A SELECT clause then specifies which variables to return. Text-to-SPARQL [29] systems translate natural language questions (e.g., "Which sensors are fed by VAVs?") into SPARQL queries which retrieve the answer from the KG.

Knowledge Graph Question Answering (KGQA). The primary goal of KGQA is to translate a user's natural language question

into a formal query, like SPARQL, that can be executed against a KG. KGQA methods largely fall into two categories: retrieval-based and semantic parsing-based approaches [7]. Retrieval-based methods (i.e., extracting the answer without using a query language) are efficient but often struggle with complex questions or large graphs. Semantic parsing methods, which convert natural language directly into a query language, are more powerful but have traditionally required large amounts of training data and domain-specific adaptation [26]. Given their better performance on multi-hop reasoning and emerging ability to quickly adapt to new domains using LLMs [2], our work focuses on semantic parsing.

Traditional KGQA methods are designed to work on a specific KG, such as Wikidata[2], which is large enough to train the KGQA method. While recent LLM-based KGQA methods have reduced the dependency on large training sets, many are still ill-suited for the building domain. Approaches like SGPT [22] still require question-query training pairs, which are unavailable for individual buildings. Others, like SPARQLGEN [17], rely on a few-shot framework that needs dataset-specific examples for context, limiting their ability to generalize to new, unseen building models. Methods requiring fine-tuning on components like entity extractors are also impractical, as this would be costly and ineffective on the dense, specialized KGs common in this domain [26]. These limitations highlight a critical requirement for practical deployment in buildings: methods must perform the task in a zero-shot setting. In simple terms, this means the system must work "out of the box" without being trained on or shown any hand-crafted examples for the specific building it is interacting with. Generating such examples for every new facility would demand domain expertise, limiting the scalability of the tools. Consequently, our work focuses exclusively on evaluating the *zero-shot capabilities of KGQA* systems.

However, the landscape of available zero-shot tools also presents challenges. Promising approaches like AutoKGQA provide the LLM with relevant ontology terms as context [2], but they still perform poorly on complex building models [4, 19]. To the best of the authors' knowledge, other existing zero-shot tools are either not designed to support arbitrary KGs [21] or their implementations were unavailable at the time of this submission [18].

Lack of a Benchmark Dataset for Buildings Domain. Progress in data-driven fields like KGQA is fundamentally driven by the availability of standardized benchmark datasets [11, 16]. Benchmark datasets let researchers meaningfully compare approaches, identify shortcomings, and track progress over time. While the broader KGQA community benefits from several such datasets, the building domain currently has no benchmark dataset for evaluating natural language interfaces for information retrieval [20].

A crucial question is whether KGQA systems trained on general-knowledge KGQA benchmarks perform well on building-specific KGQA tasks. While we empirically demonstrate the differences in §3 and previous work has indicated failures of generic KGQA tools for buildings [19], here we provide three fundamental reasons for why they are unsuitable for the buildings domain with general examples. First, the building domain relies on a highly specialized lexicon (e.g., "VAV," "economizer," "setpoint") that is absent in the entities used in benchmarks derived from general-knowledge sources like Wikidata. Second, the nature of information retrieval differs significantly: building-related queries often focus on discovering classes of entities by leveraging ontological hierarchies (e.g., finding all sensors of a specific type), rather than retrieving facts about a single, named entity (e.g., a person or place), which is the focus of many KGQA datasets (§4.1.1). Finally, the mapping between natural language and graph entities is far more complex (§4.1.2). In many existing datasets, mentions of entities are clean and map directly to graph nodes, partly due to how these datasets are constructed (automatically or semi-automatically). In contrast, human language regarding sensor metadata of buildings is inherently ambiguous and context-dependent, and may require interpretation of larger subgraphs within the KG.

This critical gap forced researchers to rely on small, ad-hoc, or private evaluation sets [2, 19], making it impossible to compare new methods effectively or to develop solutions that are truly generalizable across different buildings. Our work directly addresses the absence of a shared reference dataset, thereby encouraging future research into domain-specific KGQA tasks.

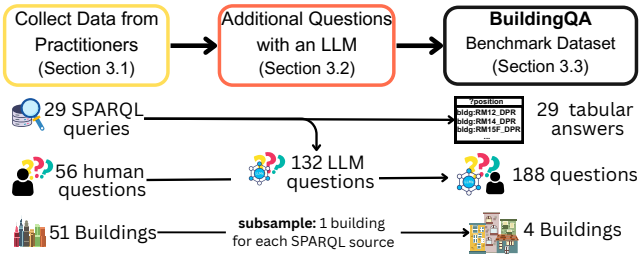

Figure 1: Process of developing the BuildingQA benchmark

3 CONSTRUCTING BUILDINGQA

Figure 1 illustrates our multi-stage process for constructing the BuildingQA benchmark, which involved collecting real-world practitioner queries, augmenting them with an LLM for linguistic diversity, and curating a final, focused dataset for evaluation.

3.1 Data Collection from Domain Experts

To ground our dataset in real-world use cases, we solicited contributions from domain experts in the building science community. Unlike general-purpose QA datasets that can leverage crowd-sourcing platforms like Amazon Mechanical Turk, the technical nature and complexity of the building domain necessitate contributors with a foundational understanding of building systems. We asked these experts to contribute SPARQL queries they had developed for their own applications, along with the corresponding building models. For each query, participants provided: (1) the building's name and characteristics, (2) building model (i.e., KG) (3) a description of the retrieved information, and (4) up to five natural language questions. This approach introduces a clear bias, as our contributors' expertise with KG structures and SPARQL likely leads to questions that are more technically precise than those from a novice building operator. We argue this expert bias makes our contributions more compelling. As we demonstrate in §4.1.2, even these expert-authored questions

[2]https://www.wikidata.org/wiki/Wikidata:Main_Page

exhibit considerable ambiguity, suggesting the difficulty of translating user intent into executable code is a fundamental problem that persists even for users with deep domain knowledge.

Our contributions came from authors of 4 sources:

- 9 SPARQL queries from Mortar's repository, for 45 buildings, all modeled in Brick [12].
- 8 SPARQL queries for four BOPTEST [6] buildings, modeled in Brick [9].
- 5 SPARQL queries for a Brick model of the student residence at the Technical University of Crete (TUC), Greece [8].
- 7 SPARQL queries for an ASHRAE 223p [1] model of commercial building model in Lawrence Berkeley National Laboratory (LBNL) [30].

In total, this collection effort yielded 56 human-authored questions corresponding to 29 unique SPARQL queries applied to 51 different buildings. Having multiple SPARQL queries for each building is an important characteristic of the initial dataset. Existing KGQA datasets (Table 2) only run queries on a single KG; our proposed dataset can be used to evaluate question answering across multiple heterogeneous KGs.

3.2 Data Augmentation with an LLM

To systematically enhance the linguistic diversity of our dataset, we employed an LLM-based approach to generate additional questions for each SPARQL query. We used GPT-o4-mini to create a spectrum of questions ranging from literal translations to abstract, conversational inquiries. For each SPARQL query, we instructed the LLM to generate five distinct phrasings. The first question is a direct, template-like mapping of the query, translating its classes and relationships into natural language while preserving the logical structure (e.g., `brick:Zone_Air_Temperature_Sensor` becomes "zone air temperature sensor"). The subsequent questions are designed to be progressively more abstract, first by paraphrasing terms and then by omitting parts of the logical chain. The final question was the most implicit, high-level inquiry a user might pose to find the same information (e.g., asking for "weather measurements" to find the outside air temperature sensors). This process creates a robust set of questions to test a system's tolerance for linguistic ambiguity. Importantly, it also establishes a vital benchmark for evaluating these systems as tools within larger agentic workflows, where AI agents can translate high-level goals into specific data queries.

3.3 BuildingQA Benchmark Dataset

Figure 1 describes the process of curating BuildingQA—a zero-shot KGQA evaluation benchmark—using the data collected from practitioners. The goal was to create a benchmark that is both comprehensive and efficient, testing a system's ability to generalize across different query types and linguistic styles rather than its performance on a query across many similar building models. Therefore, for SPARQL queries that were applicable to a large number of buildings, we selected the largest (in terms of triples) building. In addition, the complete, un-sampled dataset, containing all valid building-query pairs, will be released publicly to support broader

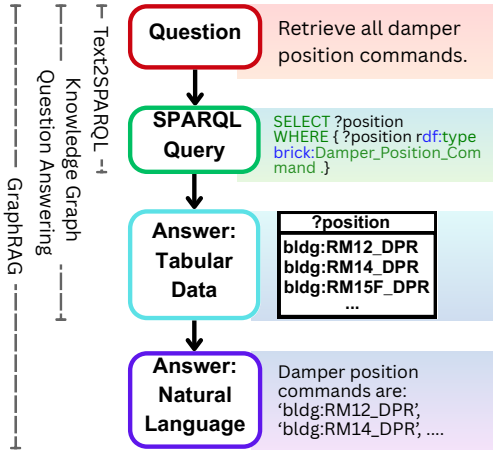

Figure 2: Essential Layers of QA pipeline. We focus on KGQA.

research and other use-cases. Nonetheless, investigating how performance on a question varies across different buildings is an important direction for future research. Our full dataset of 51 building models will be provided to support exactly this type of analysis.

The benchmark dataset we introduce is versatile, supporting evaluation at multiple stages of a question-answering pipeline (see Fig 2). These stages include: (1) SPARQL query generation, a Text-to-SPARQL task [22], which involves validating and comparing the generated queries; (2) tabular answer retrieval, a KGQA task that compares the structured results returned by executing the query on the KG; and (3) natural language response generation, a GraphRAG [18] task that evaluates the completeness and correctness of the system's final answer in plain language, similar to how LLM answers are evaluated. While all are valuable, this paper focuses specifically on the second step. We propose a novel diagnostic framework for this purpose in the next subsection, leaving the exploration of the other tasks to future work.

To develop a benchmark dataset for comparing the performance of various KGQA approaches, we processed the collected data into a final set of question-answer pairs and corresponding KGs. Unlike typical KGQA datasets that feature one large KG with thousands of automatically generated [14] or crowd-sourced questions [11] for SPARQL queries written via templates, our dataset is composed of multiple distinct KGs, where each building is a separate graph. Our curation process resulted in a benchmark of four buildings—three modeled in Brick and one in 223p—and a total of 188 question-answer pairs (with varying number of questions corresponding to each building) from both human and LLM-generated sources. Table 1 shows several examples of these questions. These buildings are:

- DFLEX (BOPTEST): A multi-zone building model from the BOPTEST [6] framework, modeled with Brick and deployed via DFLEXLIBS [9].
- B59 (LBNL Office): A medium-sized commercial office building in LBNL campus, modeled using ASHRAE 223P. It is a 67,000 sq. ft., two-floor building with an underfloor air distribution system and a DALI lighting network [30].
- BLDG11 (Mortar): A Brick-modeled Mortar building [12].
- TUC : A three-floor student residence with 60 rooms at the TUC, Greece, modeled with Brick [8].

Table 1: Example questions, with one from each building.

Building	Question
BLDG11 [12]	Find all Building electrical meters and outside air temperature sensors in my building.
TUC [8]	For each zone, what is the timeseries ID of its occupancy sensor, and what is the zone's IFC reference?
B59 [30]	Identify each entity that is an AHU, and either has a directly associated property or is indirectly connected to a property via a connectable.
DFLEX [9]	What are the timeseries IDs for the available electric power sensors and thermal power sensors in the building?

4 CHARACTERIZATION AND EVALUATION

4.1 Characterization of the Dataset

Existing KGQA benchmarks largely focus on answering factoid questions which are generated according to different methodologies. Some datasets, like LC-QuAD 2.0 [11], programmatically generate queries from templates and use crowdsourcing to create natural language questions. Others, like QALD-9 [25], rely on the manual curation of questions from previous challenges and real-world user logs to ensure linguistic diversity. A more recent approach taken by Spider4SPARQL [16] was to increase query complexity by automatically translating the large, cross-domain Spider Text-to-SQL [29] benchmark into a SPARQL format. BuildingQA distinguishes itself from these benchmarks by focusing on complex, task-driven queries within the specific domain of building operations. This presents unique structural and lexical challenges for KGQA.

4.1.1 Complexity. As shown in Table 2, the complexity of BuildingQA queries is evident in several metrics that were used to motivate Spider4SPARQL. A key differentiator is the number of projections (i.e., the columns of data returned by the query); while most benchmarks target single-entity answers and thus have few projections, BuildingQA features queries with up to 16 projections. This reflects a real-world need to extract wide, tabular datasets for operational analysis rather than retrieving simple facts.

Table 2: Comparison of SPARQL Query Complexity Metrics

Metric	LC-QuAD 2.0	QALD-9	Spider4SPARQL	BuildingQA
Max. #selections	1	2	2	1
Max. #projections	1	1	6	16
Set operations	No	No	Yes	Yes
Max. #triples	3	3	5	53
Uses aggregations	Yes	Yes	Yes	No
# of KGs	1	1	1	51[3]

Furthermore, the structural complexity of BuildingQA queries is significantly higher, with a maximum of 53 triple patterns in a single query. This metric indicates that the queries required for real-world building operations are considerably more intricate than those found in other datasets. Like the complex Spider4SPARQL benchmark, BuildingQA also employs set operations (UNION), further demonstrating its advanced nature. One characteristic of our current dataset is the absence of SPARQL aggregations (e.g., COUNT). We believe this reflects a common workflow where users perform such operations on the extracted tabular data rather than within the SPARQL query itself. However, incorporating aggregation examples is a clear avenue for future enhancements since new users might ask about the quantity of certain equipments or sensors. Overall, this analysis demonstrates that BuildingQA contributes a novel and domain-specific benchmark to the KGQA landscape, shifting the focus from factoid retrieval to the complex, multi-projection data extraction required for real-world operational tasks.

4.1.2 Lexical Gap. We develop a method for quantifying the *lexical gap* of a dataset; this is the difference between the natural language phrasings of concepts by domain experts, and the actual labels and classes found in the KG. The gap arises because users often paraphrase or indirectly refer to entities rather than using their formal names [23].

Our method first extracts and normalizes all referenced ontology terms from a given SPARQL query (e.g., `brick:Temperature_Sensor` becomes `Temperature Sensor`). In parallel, we extract potential entities from the corresponding natural language question using: (1) the GLiNER [31] zero-shot Named Entity Recognition (NER) model, and (2) prompt-based inference with GPT-4o. We also expand common abbreviations in the extracted terms (e.g., "ART" to "Room Temperature Sensor").

Finally, we measure the lexical gap by comparing the entities from the question to the terms from the query in two ways:

- Direct Match: An exact string match between the predicted entity and the SPARQL term (e.g., `brick:Room` and "room").
- Semantic Match: A high cosine similarity between the vector embeddings of the two terms (using all-mpnet-base-v2), which captures cases where the meaning is the same, even if the wording is different.

We performed the lexical gap analysis on BuildingQA and three standard KGQA datasets: Spider4SPARQL, LC-QuAD, and QALD-9. Table 3 summarizes the results, presenting the mean percentage of direct and semantic hits; a lower percentage indicates a greater lexical gap and thus higher question ambiguity. The results reveal that BuildingQA is significantly more lexically ambiguous than existing benchmarks. It registered only 9-13% direct hits and 32-42% semantic hits, which are much lower than those for the other datasets (11-58% direct hits and 32-84% semantic hits).

This increased ambiguity stems from two factors: the inherent complexity of the building domain's lexicon and the tendency for humans to describe entities more vaguely compared to template-based question generation methods. When analyzing the source of questions within BuildingQA, we observe that human-generated and LLM-generated questions yield nearly identical performance. Lastly, when we analyzed each LLM question type (i.e., LLM_1 to LLM_5) with GPT-4o, LLM_1 questions achieved a semantic hit rate of 66%, while more ambiguous LLM_5 questions achieved only 22%. These results further confirm that the intended design of varying question ambiguity, as described in §3.2, was successful.

4.2 Four-Stage Evaluation Framework

Evaluating structured tabular answers from a KG requires an informative approach that can tell us not just *if* a model is wrong, but *how* it failed. Traditional KGQA metrics like Hits@1 are designed for single-entity answers [24], while the Execution Accuracy metric from Text-to-SQL—which demands a perfect, set-based match of all

[3]Though our initial dataset has 51 building graphs, our benchmark KGQA dataset uses 4 of them.

Table 3: Summary of Lexical Gap Analysis across different datasets and models. Values are presented as mean percentage with standard deviation (± std).

Dataset	GLiNER		GPT-4o	
	Direct	Semantic	Direct	Semantic
QALD9Plus	26% (±24.2)	53% (±26.8)	33% (±23.8)	58% (±27.0)
Spider4SPARQL	20% (±26.3)	72% (±28.4)	34% (±29.4)	84% (±28.8)
LC_QUAD	50% (±33.2)	73% (±30.2)	58% (±25.9)	81% (±23.6)
BuildingQA(All)	13% (±17.3)	32% (±18.5)	11% (±15.8)	42% (±22.6)
BuildingQA (Human)	13% (±18.0)	32% (±16.2)	9% (±13.8)	38% (±16.4)
BuildingQA (LLM)	13% (±17.1)	32% (±19.3)	12% (±16.5)	43% (±24.5)

answer rows—is often too rigid, penalizing minor or partial errors excessively [29]. To this end, we introduce a diagnostic framework designed to pinpoint specific areas for model improvement. Analyzing a system's performance across our proposed four stages reveals distinct failure modes, from misunderstanding the required schema to incorrectly modeling entity relationships.

4.2.1 Core Mechanism: Column Alignment via Permutation Search. A key challenge in evaluating tabular results is that the projections returned in a generated SPARQL query, which become the column headers, are arbitrary. This is because they're based on the internal variable names an LLM uses to generate a query, which don't need to match the ground-truth query's variables even when the retrieved data is identical. To overcome this, our framework determines the optimal column mapping. It systematically evaluates all possible permutations of the predicted columns against the columns present in the ground-truth "oracle" table as long as the number of columns in the generated result is equal or more than the oracle set (e.g., by returning an additional intermediate variable, as we will show in §4.2.3). For each permutation, we calculate an F1-score (detailed below), and the permutation yielding the highest score is assumed to be the correct alignment. To manage computational cost, this search terminates as soon as a perfect match (`F1-score = 1.0`) is found, ensuring a robust mapping without relying on fragile, name-based heuristics. This alignment is foundational for the first three subsequent evaluation stages.

4.2.2 Foundational Concepts: Defining Correctness. To quantify performance at each stage, we adapt the standard classification metric F1, which is computed by comparing the system's predicted table against the oracle table. In this scenario, we define the metrics as follows:

- **True Positive (TP)**: A data point (e.g., a cell value or an entire row) in the predicted results that correctly matches one in the oracle.
- **False Negative (FN)**: A data point present in the oracle but missing from the predicted results, representing missed information.
- **False Positive (FP)**: A data point that appears in the predicted results but not in the oracle, representing extraneous information.

4.2.3 The Four Evaluation Stages. Using the concepts explained above, we calculate the F1-score in the following ways:

Stage 1: Arity Matching. *Did the system identify the correct number of concepts to return?* We simply compare the number of columns (the arity) in the predicted table to the oracle table. This metric diagnoses the most basic errors in understanding the query's required output dimensionality.

Stage 2: Entity Set F1. *Did the system retrieve the correct set of entities?* Using the aligned columns, we compute an F1-score on the set of unique values within each corresponding column. This isolates the model's performance on entity linking, independent of relational structure.

Stage 3: Row-Matching F1 (Relational Structure). *Are the relationships between entities correctly constructed?* After the column mapping, a predicted row is a TP only if an identical row exists in the oracle. This stricter, row-wise F1-score evaluates the structural correctness of the query's results.

Stage 4: Exact-Matching F1. *Did the results match the oracle perfectly?* Equivalent to `Execution Accuracy` used in Text-to-SQL[29] and Text-to-SPARQL[16], this metric performs a direct set-based comparison of rows without any column alignment, providing a final, stringent measure of overall correctness.

Interpreting Performance Metrics. Here, we will provide an example use case of our 4 stage evaluation metric. To illustrate, consider a scenario where the goal is to find all Air Handling Units (AHU) and their associated zone air temperature sensors, shown by the oracle SPARQL query (Figure 3). The correct query requires joining these entities through their defined relationships (`brick:feeds`, `brick:hasPoint`).

Now, imagine a model generates a query that correctly identifies the need to return AHUs and zone air temperature sensors, but completely omits the relational joins between them (see Figure 3). In this case, the query simply returns every AHU and every zone air temperature sensors in the database, unrelated to each other. Using strict metrics for tabular data comparison like exact-matching F1 would result in 0, since the returned columns are in the opposite order and have a different number of rows; we include it for completeness, but it is usually not helpful for understanding the failure modes. As arity matching F1=1, we can say that it returns the same number of columns as the oracle result. Then, subsequent evaluation of this common failure mode would yield a high entity-set F1, because the system successfully retrieved the correct *sets* of individual entities (most of the AHUs and most of the zone air temperature sensors). However, it would also yield a low row-matching F1, because the system failed to produce the correct *pairs* of related entities, as the crucial relational structure was missing from the query.

This result provides a clear diagnosis: the model is proficient at tabular `arity matching` (returning the right number of projections) and `entity retrieval` (identifying the "what"), but fails at `relational structuring` (defining the "how"). This insight allows a developer to focus on improving the model's ability to construct valid graph patterns for relationships, rather than incorrectly trying to fix its entity linking capabilities.

5 BASELINE EVALUATION RESULTS

In this section, we first explain the baseline methods we test using BuildingQA and then provide empirical results demonstrating performance across various criteria.

```
PREFIX brick: <https://brickschema.org/schema/Brick#>
-SELECT DISTINCT ?eqp ?sensor WHERE {
+SELECT DISTINCT ?s ?ahu ?vav WHERE {
-    ?eqp    a     brick:Air_Handling_Unit .
+    ?ahu    a     brick:Air_Handling_Unit .
     ?vav    a     brick:VAV .
-    ?sensor a     brick:Zone_Air_Temperature_Sensor .
-    ?eqp    brick:feeds   ?vav .
-    ?vav    brick:hasPoint  ?sensor .
+    ?s      a     brick:Zone_Air_Temperature_Sensor .
}
```

Figure 3: Git-style diff between the ground-truth query (red/−) and a flawed model-generated query (green/+). The model correctly selects relevant entity types but omits key relational joins (`brick:feeds`, `brick:hasPoint`) necessary for semantic structure.

5.1 Baselines

To evaluate performance on our benchmark, we selected baseline methods designed for a zero-shot setting. The vast majority of existing KGQA systems are unsuitable for the building domain because they require extensive, building-specific training data (i.e., question-query pairs) to be effective [20]. Creating such a dataset for every unique building is impractical, making zero-shot generalization a critical requirement. We therefore evaluate two distinct, state-of-the-art paradigms in LLM-powered zero-shot KGQA. These were chosen to represent the primary competing philosophies for this task: one based on retrieval-augmented generation and the other on iterative, agentic refinement.

DA-KGQA [19]: This is a recent KGQA approach designed specifically for building metadata. As DA-KGQA builds on top of a state-of-the-art KGQA tool (AutoKGQA [2]), its performance serves as an indication of how zero-shot KGQA tools (with slight prompt engineering) would perform in the buildings domain. To our knowledge, it is the only existing method tested in this domain. It consists of three steps:

(1) Subgraph Retrieval: It first identifies a relevant subgraph by performing a semantic similarity search (using FAISS[10] and WHOOSH[4]) against all triples in the KG, which have been converted into natural language sentences using labels.
(2) Query Generation: This retrieved subgraph is then injected into a prompt for an LLM, which uses a Tree-of-Thought (ToT) [27] process to generate five distinct SPARQL query candidates.
(3) Answer Selection: Finally, each of the five queries is executed. The queries and their corresponding results (or errors) are presented to another LLM instance, which selects and returns the most plausible option as the final answer.

The rationale behind DA-KGQA is to leverage semantic search to narrow the problem space while its multi-candidate approach provides solution diversity. However, its performance is highly dependent on the initial subgraph retrieval; if the retrieved triples are irrelevant, all subsequent query generations may fail. Furthermore, it lacks an iterative refinement mechanism if all five initial queries are unsuccessful.

[4]https://whoosh.readthedocs.io/en/latest/

ReAct (Reasoning and Acting) [28]: ReAct is a prominent agentic framework that enhances LLM performance through iterative self-correction. Our implementation uses a two-agent system: a `QueryWriter` and a `Critique` agent. In each iteration, the `QueryWriter` generates a SPARQL query. The `Critique` agent then observes the query and its execution result (or error message) and provides structured feedback with a decision of either *improve* or *final*. The loop continues until the `Critique` agent outputs *final* or a maximum of three iterations is reached. We tested this framework under three conditions to analyze the impact of context size:

(1) ReAct (No Graph Context): The QueryWriter agent is only provided with the necessary ontology prefixes extracted from the building graph and receives no other graph information.
(2) ReAct (Small Graph Context): To simulate a minimal, local view of the graph, the agent receives the first 100 triples. This configuration tests performance with very limited structural context.
(3) ReAct (Large Graph Context): To provide a large view of the graph, the agent receives 5,000 triples. This number was chosen to maximize the provided context while remaining within the operational token limits of the tested LLMs.

The motivation for these variants is to explore the trade-off between context size, performance, and cost. While providing more of the graph might seem beneficial, larger contexts increase token costs and can sometimes introduce noise that confuses the LLM, a known issue where models struggle with long contexts. Our results demonstrate these trade-offs.

5.2 Empirical Results

5.2.1 Performance on Correct Syntax and Execution. We evaluated three methods, ReAct and DA-KGQA, using three LLMs: GPT-o3-mini, Gemini-flash-2.5, and Llama-4-Scout-17B-16E, referred to as `o3-mini`, `g/flash`, and `llama4`. Figure 4 presents a preliminary analysis using three metrics designed to diagnose errors before the final F1 score evaluation.

First, we measure syntactic correctness to assess basic query formation ability. Success rates are high across the board (84%–100%), indicating this is not a major hurdle. Second, we evaluate if a query returns non-empty results, which serves as a proxy for whether the query can extract at least some entities, even if the final answer is flawed, potentially due to ambiguity. Here, a significant performance gap emerges: The iterative ReAct methods perform best, with ReAct (5000 Triples) using `g/flash` returning non-empty results 78% of the time. In contrast, DA-KGQA's non-iterative design achieves a 55% success rate, while the ReAct (No KG) method struggles at 24%. Third, we track the rate of queries that request fewer projections than the ground truth. This is a critical failure mode because these queries are subsequently excluded from our F1 evaluation, as complete column alignment with the ground truth is impossible. The ReAct (No KG) method is most susceptible to this error, failing this way 35% of the time. These preliminary metrics reveal that the primary challenges are not syntax, but rather retrieving the correct entities and ensuring the query requests the required columns. Overall, the results indicate that while basic syntax is not a major hurdle, successful query execution is. Iterative

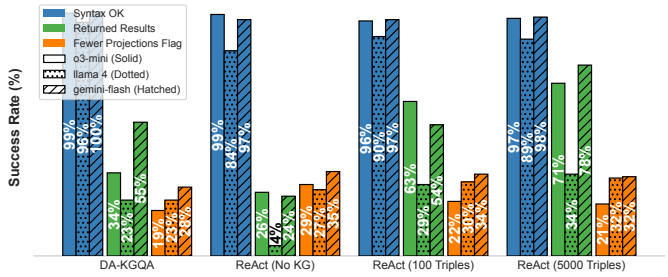

Figure 4: Performance comparison of `o3-mini`, `g/flash-2.5`, and `Llama 4`, across four methods on three quality metrics.

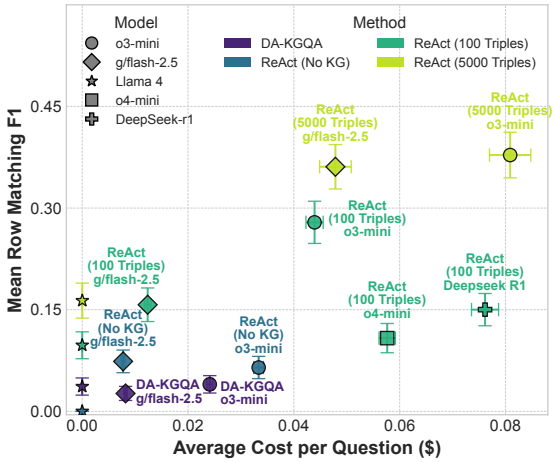

Figure 5: A scatter plot of the cost-performance trade-off, comparing the mean row matching F1 score against the average cost per question for various methods and LLMs.

methods with sufficient context (ReAct) outperform non-iterative (DA-KGQA) and no-context approaches, with `llama4` generally showing the weakest performance across these execution metrics.

5.2.2 Cost-Accuracy Trade-off. We analyze the cost versus accuracy (i.e., row matching F1 score) trade-off. All methods were tested with `o3-mini`, `g/flash`, and `llama4`. Additionally, the ReAct (100 triples) method was tested on the more powerful `o4-mini` and `deepseek-r1` models.

Figure 5 illustrates the trade-off between monetary cost per query and 'accuracy', measured by the row-matching F1 score. The ReAct (100 triples) configuration provides a controlled comparison across the five LLMs. In this low-context scenario, `o3-mini` is the most accurate, followed by `g/flash`, `deepseek-r1`, `o4-mini`, and `llama4`. For this analysis, the cost for the open-source `deepseek-r1` is based on its Azure API deployment due to its size, while the self-hostable `llama4` is assigned zero inference cost. The results reveal that investing in a larger context with an efficient model can be more effective than using a more expensive model with less context. For instance, ReAct (5000 triples) with `g/flash` achieves a higher F1 score than ReAct (100 triples) with the more expensive `o3-mini` for a comparable price. While the non-iterative DA-KGQA is the cheapest method, it also yields the lowest F1 scores. Ultimately, with the best-performing method reaching an F1 score of only 0.38, these results underscore the significant difficulty of the BuildingQA benchmark and establish a clear challenge for future research.

5.2.3 Performance across buildings. To conduct a more detailed, building-level analysis of different failure modes without the confounding variable of model capability, we now focus exclusively on results generated by `o3-mini`, as it was the most accurate LLM in the prior analysis.

Figure 6 shows the performance of DA-KGQA and three ReAct variations across our metrics for each building graph. In arity matching F1, all three ReAct variations clearly outperformed DA-KGQA, indicating they are better at matching the number of projections a user requests. The results for entity retrieval, however, are more nuanced and depend on the building's source and schema. For instance, DA-KGQA and ReAct(No KG) perform well on BLDG11 (Mortar), which we hypothesize is because Mortar is an older, open-source data, and `o3-mini` may have inherent knowledge of its common query patterns. In contrast, all methods struggle with B59, the only building modeled in the newer ASHRAE 223p standard, likely because the standard is too recent to have significant representation in the model's training data. This contrast highlights a key dependency of LLM-based KGQA: performance relies in part on the LLM's prior exposure to the domain ontology. The poor results on ASHRAE 223p therefore offer a unique insight into the baseline capabilities of these models when confronted with newer ontologies.

For other buildings like DFLEX and TUC, providing graph context proves crucial. This context enhances ReAct's ability to generate correct results, while DA-KGQA consistently struggles on these graphs, which we attribute to its ineffective subgraph retrieval and lack of an iterative query refinement process. A similar performance gap between methods is observed in the row matching F1 scores, which assess the final relational structure. These scores are expectedly lower across the board, as high row matching is impossible without first achieving high entity matching. Overall, the results indicate that ReAct, when provided with a large context of 5000 triples, demonstrates the most robust performance, particularly in extracting the correct entities across most buildings. Lastly, for exact match F1, all methods failed with the exception of ReAct(No KG). This outcome is not surprising given the metric's stringency. Exact Match performs no column mapping, requiring the output to have the same number of columns in the exact same order as the ground truth—a limitation that is exceptionally difficult to overcome when column order is not specified in the natural language question. Furthermore, the slightly lower row matching F1 scores compared to entity set F1 scores indicate that even when methods identify the correct entities, they sometimes fail to form the correct relational structures.

5.2.4 Impact of Question Ambiguity on Performance. Figure 7 shows the performance of each method across different question types, ranging from the most direct (LLM_1) to the most ambiguous (LLM_5), alongside human-authored questions (Human). As expected, performance on arity matching F1 is highest for the most direct questions, and for all methods, the ability to correctly infer the number of requested projections degrades as ambiguity increases. The Entity Matching results offer a deeper insight: while performance is highest with direct questions (LLM_1), having a

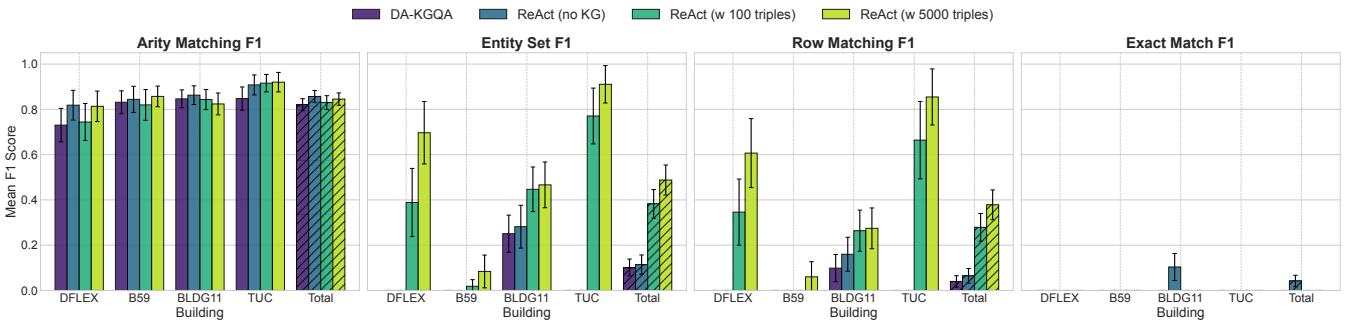

Figure 6: Mean F1 scores for four different methods across four buildings and a total summary, tested with `o3-mini`.

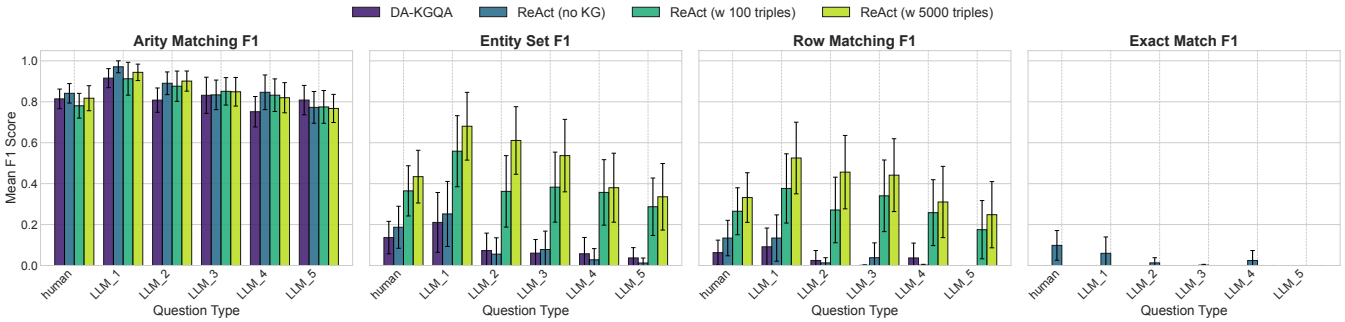

Figure 7: Mean F1 scores for four different methods (tested with `o3-mini`) across question types, from human-authored to LLM-generated questions of increasing ambiguity, where LLM_1 represents the most direct and LLM_5 the most ambiguous.

larger graph context (5000 triples) still provides a significant boost, showing that question clarity alone does not guarantee correct entity retrieval. We can also use this performance spectrum to gauge the effective directness of human questions as perceived by each system. For DA-KGQA, Human questions perform on par with LLM_1, suggesting they were interpreted as highly specific. However, for the ReAct agent with a large context, Human questions yielded performance similar to the more ambiguous LLM_4 questions. This indicates that the perceived "directness" of a question is not an absolute property but is dependent on the method being tested, and the context it was provided with. Similar patterns are observed for row matching F1, where forming the correct relational ties becomes more difficult as ambiguity increases. Finally, for exact match F1, only two methods achieved non-zero results, and on these, the Human questions performed best. This finding further highlights that exact match is a less practical metric, as its rigid dependency on column order—a detail rarely specified in a natural language question—makes it overly stringent.

5.3 Discussions

5.3.1 Common Failure Modes. Our analysis reveals several common failure modes, which we group into two main categories based on the nature of the error:

The first category involves schema mismatch errors, where the LLM fails to adhere to the provided knowledge graph structure. The most significant example is ontology hallucination, where the model invents relationships or classes that do not exist. This issue was particularly evident with the newer ASHRAE 223p [1] ontology, likely due to its limited presence in the models' training data, which reinforces the need for robust context retrieval.

The second category includes query scope and logic errors, where the model may identify correct entities but fails to capture the full user intent. This includes generating an incomplete projection (returning fewer columns than requested) or applying flawed query logic, such as unnecessary `UNION` or `FILTER` clauses. While one can iteratively refine the methods by changing prompts to be more careful against these failure modes, we did not do so, and designed each prompt without iterating based on results.

5.3.2 Ambiguity of the Ground Truth. Beyond these model-specific errors lies a more fundamental challenge: the ambiguity inherent in the ground truth itself. Many questions authored by humans are naturally underspecified and do not contain enough detail to map to a single, unique SPARQL query. Consequently, a "failure" on this benchmark does not always indicate a faulty system; it often reflects a valid mismatch between an ambiguous natural language question and the specific ground truth query it is paired with. For this reason, we do not expect any method to achieve a perfect F1 score. To help researchers analyze this effect, we have explicitly tagged questions as ambiguous if the natural language text lacked the specific details needed to uniquely construct the ground truth SPARQL query. We also include the highly specific LLM_1 questions to provide an upper-bound performance intuition.

This highlights a duality between scientific benchmarking and practical application. For a benchmark to be effective, it requires static, reproducible test cases. However, our findings suggest the most practical real-world applications will be interactive frameworks where users can iteratively clarify their intent. While such interactive systems are poorly suited for standardized benchmarking, their necessity is undeniable.

5.3.3 Lessons Learned for KGQA Benchmarking. The process of creating and evaluating the BuildingQA benchmark provided several key lessons that extend beyond our specific results and are vital for the broader community working on KGQA systems.

Embrace Ambiguity for Real-World Relevance. Existing KGQA benchmarks create questions through reverse-engineering a SPARQL query. While effective for testing raw generation, this sidesteps the reality that human language is often underspecified. Our lexical gap analysis (§4.1.2) quantitatively confirms this distinction, showing that our human-authored questions are deliberately less specific than those in comparable benchmarks. We contend that the "messiness" of human-authored questions is not a flaw to be eliminated, but a crucial feature for evaluating a system's true utility. We encourage future work to focus on developing systems that can navigate this ambiguity, rather than creating sterile benchmarks that avoid it.

The Correctness of the Graph is Not a Given. The performance of a KGQA system is fundamentally dependent on the quality of the underlying knowledge graph. During our work, we discovered that some building models, such as those from the Mortar dataset, contained outdated or incorrect schema elements. This required a non-trivial data validation and correction phase. The lesson is that the KG itself must be treated as a variable in the experiment; researchers should assume that data validation is a necessary precursor to any benchmarking, and contributions to community resources should include these corrected graphs.

Prompt Engineering Can Blur into Overfitting. There is a fine line between effective prompt engineering and overfitting to an evaluation set. It is easy to iteratively add specific instructions to a prompt to fix observed errors, but doing so risks creating a complex prompt that is brittle and does not generalize. Such a process inflates performance on a static benchmark without representing a truly robust method. For transparency and to combat this, future work must disclose the exact prompts used. We provide all of our prompts in the accompanying GitHub repository.

5.3.4 Future directions. The BuildingQA benchmark and our initial results point to several key directions for future research, ranging from dataset growth to new applications for the technology. A foundational direction for all the work described above is the continued growth of the underlying data resources. The value of the BuildingQA benchmark and its associated repository will be greatly amplified through community collaboration. We have established a public repository for this purpose and invite researchers and practitioners to contribute additional building models, expert queries, and questions. This collective effort is essential for enriching the benchmark's diversity and scale, creating a larger resource to develop the next generation of KGQA systems for buildings.

Building on this resource, our results highlight three promising avenues for advancing the core KGQA systems. First, enhancing context for the language model is paramount; future work should explore more robust, context-aware retrieval mechanisms beyond simple semantic similarity. Second, incorporating high-level building metadata, such as its primary use or archetype, could provide crucial context to disambiguate entities. Finally, a significant opportunity lies in expanding LLM agency with multi-tool, agentic frameworks that can iteratively explore the graph to construct more complex and precise queries.

While this benchmark provides a standardized measure of system performance, the next critical step involves studying real-world usability and human-computer interaction. Future experiments with human operators are essential for understanding how practitioners formulate questions and interact with system outputs, especially when faced with imperfect or empty results. Such studies would reveal opportunities for co-adaptation, where we can not only improve the tools but also guide users to phrase their requests more effectively, creating a more symbiotic relationship between the user and the system.

Finally, the utility of these text-to-SPARQL/KGQA/RAG systems extends beyond direct human use and into the domain of autonomous AI agents. We envision future AI systems tasked with high-level goals, such as "report on my building's energy efficiency last month" that must first query KGs to retrieve the necessary time-series data points for analysis. In this context, the tools we have benchmarked serve as a foundational capability, enabling AI agents to programmatically access and reason over the vast, structured data held within KGs.

6 CONCLUSIONS

In this work, we addressed a critical gap in the building science domain: the lack of a standardized benchmark for evaluating natural language interfaces for KGs. We introduced BuildingQA, a large-scale, diagnostically-rich benchmark constructed from real-world building models and expert-authored queries. Paired with a novel four-stage evaluation framework, BuildingQA provides the community with the necessary tools to rigorously measure and compare the performance of text-to-SPARQL systems.

Our evaluation of modern LLM-powered approaches on BuildingQA reveals that current methods still fall short, confirming that natural language querying in the building domain remains a challenging open problem. The agentic ReAct framework, when provided with a large portion of the graph as context, consistently outperformed the more complex, retrieval-focused DA-KGQA method.

Ultimately, this work establishes that future progress depends not only on more capable foundation models but, more importantly, on developing frameworks that can effectively equip them with the specific schema and instance data of a given building. By providing a benchmark, a diagnostic evaluation framework, and a clear analysis of current failure modes, BuildingQA offers a foundational resource to guide the development of the next generation of domain-aware methods needed to make semantic data truly accessible to building experts and autonomous systems.

ACKNOWLEDGMENTS

We would like to thank Flavia De Andrade Pereira, Ettore Zanetti, and Carlos Duarte for their valuable contributions of semantic models and questions during the data collection process. We also thank the Wilton E. Scott Institute for Energy Innovation for their support. This research was supported by the Assistant Secretary for Energy Efficiency and Renewable Energy, Office of Building Technologies of the U.S. Department of Energy, under Contract

No. DE-AC02-05CH11231, and by the California Energy Commission under EPC-19-301 and EPC-19-309. Additional support was provided by a NIST PREP fellowship and the National Alliance for Water Innovation (NAWI), funded by the U.S. Department of Energy, Office of Energy Efficiency and Renewable Energy (EERE), Industry Technology Office, under DE-FOA-0001905. It was also funded by the U.S. Department of Energy, Office of Energy Efficiency and Renewable Energy (EERE), Building Technologies Office. The views expressed in the article do not necessarily represent the views of the DOE or the U.S. Government. The U.S. Government retains and the publisher, by accepting the article for publication, acknowledges that the U.S. Government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this work, or allow others to do so, for U.S. Government purposes.

REFERENCES

[1] 2025. ASHRAE Standard 223P. https://www.energy.gov/eere/buildings/ashrae-standard-223p. Accessed: 2025-07-31.

[2] Caio Viktor S. Avila, Vânia M.P. Vidal, Wellington Franco, and Marco A. Casanova. 2024. Experiments with text-to-SPARQL based on ChatGPT. In *2024 IEEE 18th International Conference on Semantic Computing (ICSC)*. IEEE, Laguna Hills, CA, USA, 277–284. https://doi.org/10.1109/ICSC59802.2024.00050

[3] Bharathan Balaji, Arka Bhattacharya, Gabriel Fierro, Jingkun Gao, Joshua Gluck, Dezhi Hong, Aslak Johansen, Jason Koh, Joern Ploennigs, Yuvraj Agarwal, et al. 2016. Brick: Towards a unified metadata schema for buildings. In *Proceedings of the 3rd ACM International Conference on Systems for Energy-Efficient Built Environments*. 41–50.

[4] Ozan Baris, Yizhuo Chen, Gaofeng Dong, Liying Han, Tomoyoshi Kimura, Pengrui Quan, Ruijie Wang, Tianchen Wang, Tarek Abdelzaher, Mario Bergés, Paul Pu Liang, and Mani Srivastava. 2025. Foundation Models for CPS-IoT: Opportunities and Challenges. arXiv:2501.16368 [cs.LG] https://arxiv.org/abs/2501.16368

[5] Imane Lahmam Bennani, Anand Krishnan Prakash, Marina Zafiris, Lazlo Paul, Carlos Duarte Roa, Paul Raftery, Marco Pritoni, and Gabe Fierro. 2021. Query relaxation for portable brick-based applications. In *Proceedings of the 8th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*. 150–159.

[6] David Blum, Javier Arroyo, Sen Huang, Ján Drgoňa, Filip Jorissen, Harald Taxt Walnum, Yan Chen, Kyle Benne, Draguna Vrabie, Michael Wetter, et al. 2021. Building optimization testing framework (BOPTEST) for simulation-based benchmarking of control strategies in buildings. *Journal of Building Performance Simulation* 14, 5 (2021), 586–610.

[7] Abir Chakraborty. 2024. Multi-hop Question Answering over Knowledge Graphs using Large Language Models. arXiv:2404.19234 [cs.AI] https://arxiv.org/abs/2404.19234

[8] Flavia de Andrade Pereira, Kyriakos Katsigarakis, Dimitrios Rovas, Marco Pritoni, Conor Shaw, Lazlo Paul, Anand Prakash, Susana Martin-Toral, Donal Finn, and James O'Donnell. 2025. A semantics-driven framework to enable demand flexibility control applications in real buildings. *Advanced Engineering Informatics* 64 (2025), 103049.

[9] Flavia de Andrade Pereira, Lazlo Paul, Marco Pritoni, Armando Casillas, Anand Prakash, Weiping Huang, Conor Shaw, Susana Martin-Toral, Donal Finn, and James O'Donnell. 2024. Enabling portable demand flexibility control applications in virtual and real buildings. *Journal of Building Engineering* 86 (2024), 108645.

[10] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The Faiss library. (2024). arXiv:2401.08281 [cs.LG]

[11] Mohnish Dubey, Debayan Banerjee, Abdelrahman Abdelkawi, and Jens Lehmann. 2019. LC-QuAD 2.0: A Large Dataset for Complex Question Answering over Wikidata and DBpedia. In *The Semantic Web – ISWC 2019: 18th International Semantic Web Conference, Auckland, New Zealand, October 26–30, 2019, Proceedings, Part II* (Auckland, New Zealand). Springer-Verlag, Berlin, Heidelberg, 69–78. https://doi.org/10.1007/978-3-030-30796-7_5

[12] Gabe Fierro, Marco Pritoni, Moustafa Abdelbaky, Daniel Lengyel, John Leyden, Anand Prakash, Pranav Gupta, Paul Raftery, Therese Peffer, Greg Thomson, and David E. Culler. 2019. Mortar: An Open Testbed for Portable Building Analytics. *ACM Transactions on Sensor Networks* 16, 1 (Dec. 2019), 7:1–7:31. https://doi.org/10.1145/3366375

[13] Steve Harris and Andy Seaborne. 2013. *SPARQL 1.1 Query Language*. W3C Recommendation. W3C. https://www.w3.org/TR/sparql11-query/

[14] Ann-Kathrin Hartmann, Edgard Marx, and Tommaso Soru. 2018. Generating a large dataset for neural question answering over the DBpedia knowledge base. In *Workshop on Linked Data Management, co-located with the W3C WEBBR*, Vol. 2018.

[15] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and S Yu Philip. 2021. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE transactions on neural networks and learning systems* 33, 2 (2021), 494–514.

[16] Catherine Kosten, Philippe Cudré-Mauroux, and Kurt Stockinger. 2023. Spider4SPARQL: A Complex Benchmark for Evaluating Knowledge Graph Question Answering Systems. https://doi.org/10.1109/BigData59044.2023.10386182 arXiv:2309.16248 [cs].

[17] Liubov Kovriguina, Roman Teucher, Daniil Radyush, Dmitry Mouromtsev, N Keshan, S Neumaier, AL Gentile, and S Vahdati. 2023. SPARQLGEN: One-Shot Prompt-based Approach for SPARQL Query Generation.. In *SEMANTiCS (Posters & Demos)*.

[18] Meng-Chieh Lee, Qi Zhu, Costas Mavromatis, Zhen Han, Soji Adeshina, Vassilis N. Ioannidis, Huzefa Rangwala, and Christos Faloutsos. 2025. HybGRAG: Hybrid Retrieval-Augmented Generation on Textual and Relational Knowledge Bases. arXiv:2412.16311 [cs.LG] https://arxiv.org/abs/2412.16311

[19] Ozan Baris Mulayim, Gabe Fierro, Mario Bergés, and Marco Pritoni. 2025. Towards Zero-shot Question Answering in CPS-IoT: Large Language Models and Knowledge Graphs. In *Proceedings of the 2nd International Workshop on Foundation Models for Cyber-Physical Systems & Internet of Things*. 7–12.

[20] Ozan Baris Mulayim, Lazlo Paul, Marco Pritoni, Anand Krishnan Prakash, Malavikha Sudarshan, and Gabe Fierro. 2024. Large Language Models for the Creation and Use of Semantic Ontologies in Buildings: Requirements and Challenges. In *Proceedings of the 11th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*. 312–317.

[21] Md Rashad Al Hasan Rony, Debanjan Chaudhuri, Ricardo Usbeck, and Jens Lehmann. 2022. Tree-KGQA: an unsupervised approach for question answering over knowledge graphs. *IEEE Access* 10 (2022), 50467–50478.

[22] Md Rashad Al Hasan Rony, Uttam Kumar, Roman Teucher, Liubov Kovriguina, and Jens Lehmann. 2022. SGPT: A Generative Approach for SPARQL Query Generation From Natural Language Questions. *IEEE Access* 10 (2022), 70712–70723. https://doi.org/10.1109/ACCESS.2022.3188714

[23] Nadine Steinmetz and Kai-Uwe Sattler. 2021. What is in the KGQA benchmark datasets? Survey on challenges in datasets for question answering on knowledge graphs. *Journal on Data Semantics* 10, 3 (2021), 241–265.

[24] Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel M. Ni, Heung-Yeung Shum, and Jian Guo. 2024. Think-on-Graph: Deep and Responsible Reasoning of Large Language Model on Knowledge Graph. arXiv:2307.07697 [cs.CL] https://arxiv.org/abs/2307.07697

[25] Ricardo Usbeck, Ria Hari Gusmita, Axel-Cyrille Ngonga Ngomo, and Muhammad Saleem. 2018. 9th Challenge on Question Answering over Linked Data (QALD-9) (invited paper). In *Semdeep/NLIWoD@ISWC*. https://api.semanticscholar.org/CorpusID:53220210

[26] Shuangtao Yang, Mao Teng, Xiaozheng Dong, and Fu Bo. 2023. LLM-Based SPARQL Generation with Selected Schema from Large Scale Knowledge Base. In *Knowledge Graph and Semantic Computing: Knowledge Graph Empowers Artificial General Intelligence*, Haofen Wang, Xianpei Han, Ming Liu, Gong Cheng, Yongbin Liu, and Ningyu Zhang (Eds.). Springer Nature, Singapore, 304–316. https://doi.org/10.1007/978-981-99-7224-1_24

[27] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2024. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems* 36 (2024).

[28] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. arXiv:2210.03629 [cs.CL] https://arxiv.org/abs/2210.03629

[29] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2019. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. arXiv:1809.08887 [cs.CL] https://arxiv.org/abs/1809.08887

[30] Ettore Zanetti, David Blum, Hongxiang Fu, Chris Weyandt, Marco Pritoni, and Mary Ann Piette. 2025. Commercial Building HVAC Demand Flexibility with Model Predictive Control: Field Demonstration and Literature Insights. *Energy and Buildings* (2025), 116097.

[31] Urchade Zaratiana, Nadi Tomeh, Pierre Holat, and Thierry Charnois. 2023. GLiNER: Generalist Model for Named Entity Recognition using Bidirectional Transformer. arXiv:2311.08526 [cs.CL] https://arxiv.org/abs/2311.08526

[32] Junwen Zheng and Martin Fischer. 2023. Dynamic prompt-based virtual assistant framework for BIM information search. *Automation in Construction* 155 (2023), 105067.