

Lawrence Berkeley National Laboratory

LBL Publications

Title

Extraction and Analysis of Time Series Data from Building Automation Systems Using Large Language Models

Permalink

<https://escholarship.org/uc/item/4xh0900n>

Journal

ASHRAE Transactions, 131(Pt2)

ISSN

0001-2505

Authors

Mulayim, Ozan Baris

Prakash, Anand Krishnan

Paul, Lazlo

et al.

Publication Date

2025-08-04

DOI

10.63044/s25ext76

Copyright Information

This work is made available under the terms of a Creative Commons Attribution-NonCommercial License, available at <https://creativecommons.org/licenses/by-nc/4.0/>



Building Technologies & Urban Systems Division
Energy Technologies Area
Lawrence Berkeley National Laboratory

Extraction and Analysis of Time Series Data from Building Automation Systems Using Large Language Models

Ozan Baris Mulayim¹, Anand Krishnan Prakash^{1,2}, Lazlo Paul², Marco Pritoni²

¹Carnegie Mellon University, ² Lawrence Berkeley National Laboratory

Energy Technologies Area
June 2025

doi:10.20357/B73W4C



This work was supported by the Assistant Secretary for Energy Efficiency and Renewable Energy,
Building Technologies Office, of the US Department of Energy
under Contract No. DE-AC02-05CH11231.

Disclaimer:

This document was prepared as an account of work sponsored by the United States Government. While this document is believed to contain correct information, neither the United States Government nor any agency thereof, nor the Regents of the University of California, nor any of their employees, makes any warranty, express or implied, or assumes any legal responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by its trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof, or the Regents of the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof or the Regents of the University of California.

Extraction and Analysis of Time Series Data from Building Automation Systems Using Large Language Models

Ozan Baris Mulayim

Anand Krishnan Prakash

Lazlo Paul

Marco Pritoni

Student Member ASHRAE

Affiliate ASHRAE

Full Member ASHRAE

ABSTRACT

Semantic schemas like Haystack 4, Brick and ASHRAE standard 223 enable the structured, standardized, and machine-readable representation of building data, facilitating interoperability, data integration, and advanced analytics. However, extracting information from these models requires specialized expertise in SPARQL and other programming languages, skills that are not commonly found among building professionals. Recent advancements in Large Language Models (LLMs), such as ChatGPT, enable the construction of queries using natural language, making it easier for individuals to interact with these systems in a manner that resembles everyday speech. However, these methods have not yet been tested on building semantic ontologies. This paper introduces a novel workflow and tool for enabling users to ask questions about a specific building's data, using natural language and receive answers automatically generated by GPT-4o. Our approach integrates semantic ontologies with advanced LLM capabilities to automate three critical steps: (1) generating SPARQL queries to retrieve time series references from ontological models, (2) extracting the corresponding time series data from the Building Automation System, and (3) performing computations and visualizations tailored to the user's query. The proposed method simplifies access to BAS data, allowing both domain experts and non-specialists to conduct sophisticated analyses without needing extensive technical knowledge of semantic web technologies. By demonstrating this pipeline, we facilitate more accessible and scalable data-driven decision-making in building operations and management.

INTRODUCTION

The increasing connectivity and digitalization of modern buildings has led to the generation of enormous volumes of data from Building Automation Systems (BAS). Yet, unlocking the full potential of this data remains a significant challenge for facility and utility managers who typically lack the specialized expertise required to manipulate and analyze complex datasets. In the last decade, multiple semantic ontologies have been proposed as a solution for representing and understanding building metadata in a standard way. These ontologies, such as Brick (Balaji et al. 2016) and the proposed ASHRAE standard 223 (Pritoni et al. 2024), provide a way to represent buildings, their equipment topology and their data sources and actuators in a structured and machine-readable way. These semantic representations of buildings are built upon a standard definition of concepts within the ontologies and this allows application developers to create portable solutions based on semantic requirements and through semantic queries (Chiosa et al. 2024).

Ozan Baris Mulayim is a PhD student at Carnegie Mellon University. **Anand Krishnan Prakash** is a PhD student at Carnegie Mellon University and a Principal Engineering Associate at Lawrence Berkeley National Laboratory. **Lazlo Paul** is a Technology Researcher at Lawrence Berkeley National Laboratory. **Marco Pritoni** is a staff scientist at Lawrence Berkeley National Laboratory

However, in spite of data interoperability and the development of portable applications that semantic ontologies enable, the steep learning curve and the domain expertise required to use the ontologies for developing analytics and control applications has limited their practical adoption in the building industry (Bennani et al. 2021).

Recent breakthroughs in Large Language Models (LLMs) have opened new avenues for transforming data interaction (Bowen et al. 2023). LLMs are not only adept at understanding and generating natural language but are increasingly demonstrating emerging reasoning capabilities (Zhu et al. 2024). This evolving intelligence promises the ability to generate structured queries, understand context, and even perform preliminary analytical reasoning—all through simple, natural language instructions (Mulayim et al. 2024). By leveraging these capabilities, there is growing potential to develop interfaces that empower non-technical users to extract actionable insights from BAS data in the near future.

In this work, we present a novel framework designed to empower building practitioners to ask complex questions about their building operations in plain language and receive automated, data-driven answers. Our approach integrates LLMs into a workflow that orchestrates three critical steps:

1. Query generation: SPARQL (W3C 2008) is a query language for accessing and manipulating data in semantic models. We use LLMs to transform natural language queries into precise SPARQL commands to retrieve relevant time series references from semantic models.
2. Data extraction: Pulling the corresponding time series data from BAS and historical data sources.
3. Time series analysis and visualization: Processing and visualizing the data to address the specific inquiry, providing clear insights into building performance.

To our knowledge, this is the first work to demonstrate an end-to-end application of LLMs for workflows in building time series analytics. By combining the interpretive power of LLMs with the structured precision of semantic ontologies, our framework represents a significant step toward making sophisticated building data analytics accessible to domain experts without requiring deep technical knowledge.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 details the architecture and methodology of our proposed framework. Section 4 presents case studies that illustrate the practical benefits and challenges of our approach. Finally, Section 5 discusses future directions, and Section 6 concludes the paper.

BACKGROUND AND CONTRIBUTIONS

BAS data analysis requires access to sensor data, control commands and setpoints, as well as information about the building equipment and systems. Knowledge graphs (KGs) (Hogan et al. 2022) provide a powerful way to model these complex relationships, but interacting with them typically requires expertise in semantic technologies and query languages like SPARQL. This section introduces the key concepts behind KGs, the Brick Schema (Balaji et al. 2016), and SPARQL before discussing two complementary and promising research areas — text-to-SPARQL translation and language-to-code generation — that enable more intuitive access to BAS data and corresponding analytics through natural language interaction.

Knowledge Graphs and Semantic Representations

A KG is a structured representation of information where entities (e.g., rooms, sensors, HVAC systems) are connected through well-defined relationships, commonly referred to as a semantic model in the building's domain. KGs are typically built using the Resource Description Framework (RDF) (W3C 2014), a standard data model that represents information as triples in the form of a subject, a predicate and an object. While RDF provides the underlying structure, ontologies define the domain-specific vocabulary, relationships, and constraints that give meaning to the data. In other words, an ontology is a formal schema that organizes concepts and their interrelations within a knowledge graph, enabling consistent interpretation across different systems. One widely adopted ontology for buildings is Brick Schema, which provides a standardized representation of building components, their

relationships, and metadata. It describes physical, logical, and virtual assets, such as sensors and equipment, to enhance interoperability and data integration across building management systems. By providing a common vocabulary and structure, Brick facilitates consistent data annotation and querying, enabling more effective analysis, fault detection, and optimization of building operations. By leveraging SPARQL, a query language specifically designed for RDF-based KGs, users can retrieve building data in a flexible and structured manner. Despite these advantages, manually constructing SPARQL queries remains challenging for non-experts in many domains, which has motivated the development of tools to translate natural language into structured queries such as SPARQL.

Text-to-SPARQL

Text-to-SPARQL refers to the process of converting natural language questions into SPARQL queries, enabling structured access to KGs. This capability is central to Knowledge Graph Question Answering (KGQA) systems, which rely on semantic parsing to ensure precise query generation rather than relying on retrieval-based methods that may yield approximate answers (Chakraborty 2024). Early approaches to text-to-SPARQL relied on rule-based and template-driven methods (Bizer and Schultz 2009; Schmidt et al. 2009), which required significant manual effort and domain-specific knowledge. Machine learning-based approaches later emerged, leveraging supervised training on labeled SPARQL datasets (Zafar et al. 2018; Soru et al. 2018). However, these models often struggle with generalization across different schemas and domains due to their reliance on structured query pairs.

Recent advances in LLMs have introduced a new paradigm for text-to-SPARQL translation, enabling dynamic query generation without requiring explicit supervision. LLM-based methods such as SGPT have demonstrated the ability to learn graph structures and generate SPARQL queries without predefined templates (Rony et al. 2022). Other approaches, including SPARQLGEN and AUTO-KGQAGPT, enhance query accuracy by integrating schema information into LLM prompts (Kovriguina et al. 2023; Avila et al. 2024). Hybrid approaches further improve query generation by combining entity linking with structured prediction models before applying LLM-based query formulation (Yang et al. 2023). Preliminary results suggest that LLMs hold promise for generating SPARQL queries over KGs created with Brick, though challenges remain—particularly as these tools begin to fail when user questions become more ambiguous (Baris et al. 2025; Mulayim et al. 2025).

Language-to-Code

Language-to-code generation has emerged as a key area in leveraging LLMs for automation tasks. Early work, such as Codes as Policies (Liang et al. 2023), demonstrated the potential of LLMs to generate executable code for robotic applications. More recently, researchers have explored the use of LLMs for industrial control and automation, including automated test case generation (Koziolek et al. 2024) and retrieval-augmented generation (RAG) (Lewis et al. 2020) methods for industrial control programming.

One notable advancement is the introduction of interactive refinement techniques, such as flow engineering, where LLMs iteratively refine code through execution and validation cycles. This approach incorporates self-reflection mechanisms that allow models to analyze and improve generated code based on test results. Benchmarks such as CodeContests (Google DeepMind 2023), initially used in DeepMind’s AlphaCode (Google DeepMind 2022), have demonstrated the effectiveness of fine-tuned models for programming tasks, highlighting the growing capabilities of LLMs in structured code generation.

Contribution

The integration of text-to-SPARQL and language-to-code capabilities is essential for our proposed framework, which leverages LLMs to automate data retrieval and analysis in BAS environments. By building on these advancements, we aim to bridge the gap between complex semantic representations and intuitive, natural language interfaces for facility and utility managers. In this work, we combine the strategies from other domains into developing a novel framework for extraction and analysis of time series data for buildings. Though the development of this framework is still in its infancy, it demonstrates the promise LLMs hold for allowing nonexperts to interact with building data.

METHODOLOGY

Our framework is built upon a recent emerging paradigm called agentic workflows. Agentic workflows are “structured sequences of LLM invocations accompanied by detailed instructions” (Zhang et al. 2024). Our framework is implemented in Python and uses LLMs and Brick Schema to autonomously generate SPARQL queries, retrieve time series data, generate plots, and produce analytical outputs, which can be seen in Figure 1. For clarity, the prompts presented in this paper are shortened; full-length prompts can be found in our GitHub repository¹. In this work, we assume that the Brick model of the building already exists. Across all steps we have an iterative approach that reprompts LLMs with the observed error in case of generated code/query crashing. We also configure a parameter (“max_iterations”) to avoid excessive LLM calls.

The objective of the first step is to identify the data sources required to answer a prompt provided by a user and their corresponding time series references. The framework handles this by generating SPARQL queries using an example-based retrieval strategy for few-shot prompting. Since prior work (Baris et al. 2025; Mulayim et al. 2025) demonstrated considerable challenges in generating accurate queries when no examples are present in the prompt (i.e., zero-shot), this framework leverages a set of example queries and their descriptions. While the framework should contain generic query-description pairs, in practice, these examples can be generated by the practitioner who developed the Brick model of the building. Each example shall include a description of what the query retrieves (e.g., “Extracting time series of zone air temperature sensors and their corresponding VAVs”) and the corresponding SPARQL query to retrieve the necessary information from the Brick model of the specific building. Based on the query-description pairs and the prompt provided by a user, our framework computes a similarity metric between the user prompt and example descriptions. The similarity metric is computed by calculating the cosine similarity between the mpnet_v2 embeddings (Lahitani et al. 2016) of the question and each description. The three most similar queries are chosen to be included in the LLM prompt, improving accuracy over a zero-shot approach. This similarity-based retrieval ensures that the examples are contextually aligned with the user’s intent and the choice of “three” similar queries is a commonly used and configurable parameter. This approach is identical to the RAG methods commonly seen in LLM applications, but is applied to extract examples rather than general context. A successful completion of this step returns a SPARQL query, which is then executed on the building’s Brick model to retrieve the time series references of the required data sources and actuators.

In the second step, the references to time series data is used to prompt another LLM to write a custom function in Python that extracts the corresponding historical data streams from the database (or another source). The prompt shall include instructions on how to navigate the database once the time series references are provided.

The next step is responsible for using the retrieved data and generating interpretable charts and other visualizations. The LLM is prompted with the first five rows (configurable parameter) of the extracted time series data, and asked to write a function in Python to generate a single plot. This visualization step enhances interpretability, allowing users to observe trends and patterns relevant to their questions, without needing to look at numerical data and creating charts manually.

The goal of the fourth step is to perform computations on the retrieved data to answer the user’s prompt. It achieves this by writing a function in Python that will perform the necessary computations on the data. Depending on the question, these computations may involve statistical summarization, anomaly detection, or trend analysis.

EVALUATION AND RESULTS

We evaluated our framework using GPT-4o as the LLM, with max_iterations = 3 to limit excessive retries and temperature = 0 to minimize stochasticity in responses. To evaluate the framework, we asked it three questions (see Table 1) and compared the final result determined against a manually determined ground truth. Additionally, since LLMs are inherently probabilistic, we ran each question five times to assess consistency. As the KG, we utilized the Brick model of FLEXLAB², a well instrumented experimental test facility located at Lawrence Berkeley National Laboratory, Berkeley, California. FLEXLAB is a highly

¹ <https://github.com/ozanbarism/Talking2MyBAS/tree/main>

² <https://flexlab.lbl.gov/>

configurable facility, allowing users to test and evaluate different technologies and solutions for different types and configurations of buildings and equipment. For our evaluation, the Brick model of two FLEXLAB test cells that represent a commercial office space was used. Each cell is conditioned by its own air handling unit (AHU) and is also divided into 3 zones, each of which have their own variable air volume (VAV) box. The example description-query pairs used for few-shot prompting were generated by the practitioner who developed the KG, while the testing questions were formulated by our team. Further details about the framework, including the example description-query pairs and the testing questions, are available in our GitHub repository.

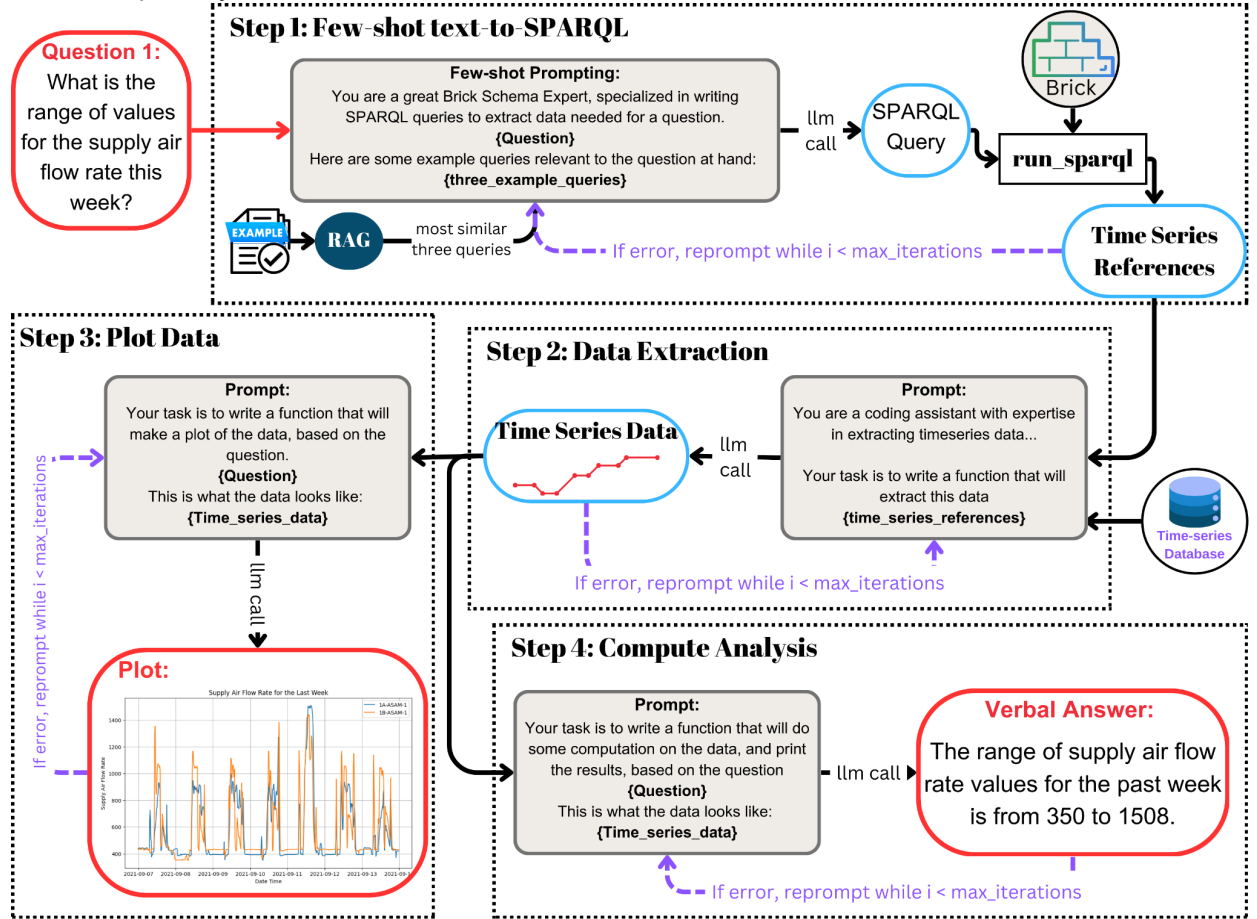


Figure 1 The Proposed Agentic Framework

Table 1 shows the questions provided to the framework and summarizes the results, showing success rates, cost per query, execution time, and token usage for different building analytics questions. The framework successfully answered most queries, with 100% accuracy in retrieving and analyzing VAV damper positions and supply air flow rate ranges (questions 1 and 2). However, for the third question on airflow drops, which are suggestive of blockages, two out of five attempts failed due to incorrect SPARQL queries and due to indexing errors when referencing the timestamp column respectively. The cost per query ranged from \$0.0229 to \$0.0338, with an average runtime of 15–17 seconds. Token usage varied based on query complexity, with higher token counts observed for questions requiring multi-step reasoning.

Table 1. Questions asked to the framework and summarized results (each asked five times).

Questions	Succe ss	Average Cost (\$)	Average Duration (s)	Average Tokens
-----------	-------------	----------------------	-------------------------	-------------------

How frequently do the VAV dampers reach their fully open or fully closed positions?	5/5	0.034	17	4350
What is the range of values for the supply air flow rate this past week?	5/5 ⁺	0.023	15	2890
Are there any significant drops in airflow that could indicate potential blockages?	3/5 [*]	0.032	17	3882

^{*} Two failure cases were due to (1) an error in the generated SPARQL query, and (2) the generated code checking the datetime column even though it was the index.

⁺ In 3 of these cases, though it made the correct plot, it only returned the air flow rate for one of the sensors in the verbal description.

Question 1. Our framework, when prompted with the first question: “How frequently do the VAV dampers reach their fully open or fully closed positions?”, generated clear and interpretable plots, one of which is shown in Figure 2. As a result of the last step, it also printed “Fully open positions (100%): 349, Fully closed positions (0%): 0”. These results provide very direct answers to the question and demonstrate the ability of this framework to generate relevant plots, without any input from the user. It also exhibits the ability to provide more than what was asked for as we can see with the verbal response, where it was able to separately provide the duration of fully closed dampers and fully open dampers.

Question 2. Figure 3 shows examples of a SPARQL query and a trend plot auto-generated by our framework for question 2: “What is the range of values for the supply air flow rate this past week?”. Across the five trials we conducted, two of them resulted in verbal answers as follows: “{‘1A-ASAM-1’: {‘min’: 350.26, ‘max’: 1508.58}, ‘1B-ASAM-1’: {‘min’: 350.71, ‘max’: 1442.084}}”. However, the remaining three trials produced the output: “The range of supply air flow rate values for the past week is from 350.26 to 1508.58”. This result is interesting as the first set of results concurs with the observation from question 1, where the framework provided more information than what was asked, as it provided the range of the flow rate for each AHU separately, but the result is not very interpretable unless the user knows what ‘1A-ASAM-1’ and ‘1B-ASAM-1’ mean. The next batch of outputs are much more human-understandable, but incomplete as they provide results for only one of the two AHUs.

Question 3. The third question that we used to evaluate our framework: “Are there any significant drops in airflow that could indicate potential blockages?” resulted in failures two out of the five times. An incorrect SPARQL query generated in step 1 caused one of the unsuccessful results and an incorrect reference to the timestamp column (used column ‘datetime’ instead of using the index) in the Python code that was analyzing the time series data, generated by in step 4 resulted in the second failure. In the remaining three runs, we were able to generate the right queries and charts as shown in Figure 4. To answer the question, the framework also makes an assumption that a 10% drop in airflow will be considered “significant”. The verbal answers that were generated were, however, quite difficult to interpret since it presented the rows of time series measurements of the two supply air flow rate sensors during the time periods where there were significant drops in airflow.

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX brick: <https://brickschema.org/schema/Brick#>
PREFIX ref: <https://brickschema.org/schema/Brick/ref#>
SELECT ?vav ?damper_pos ?ts WHERE {
  ?vav rdf:type brick:VAV .
  ?damper_pos brick:isPointOf ?vav .
  ?damper_pos rdf:type brick:Damper_Position_Sensor .
  ?damper_pos ref:TimeseriesReference ?ts .
}

```

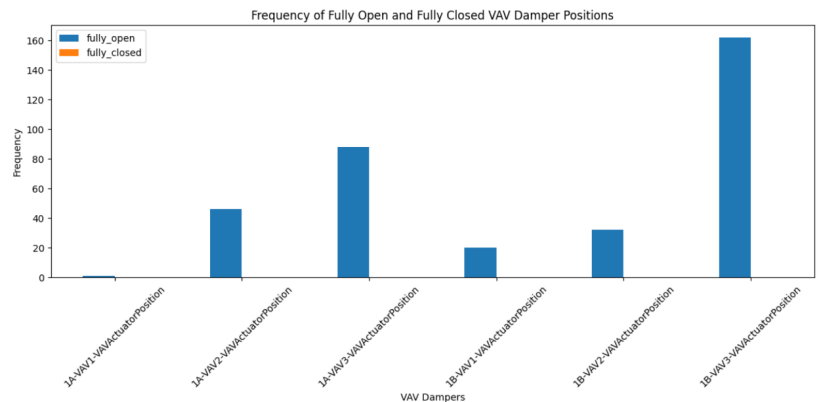


Figure 2 (a) Auto-generated SPARQL query for the question “How frequently do the VAV dampers reach their fully open or fully closed positions?” (step 1) (b) Auto-generated visualization generated by the framework (step 3). It shows how long each of the six dampers (1 in each VAV box, 3 VAV boxes in each cell and 2 cells in FLEXLAB Brick model) was fully open or closed (none of them were fully closed, so there are no orange bars).

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX brick: <https://brickschema.org/schema/Brick#>
PREFIX ref: <https://brickschema.org/schema/Brick/ref#>
SELECT ?sup ?ts WHERE {
  ?sup rdf:type brick:Supply_Air_Flow_Sensor .
  ?sup ref:TimeseriesReference ?ts .
}
```

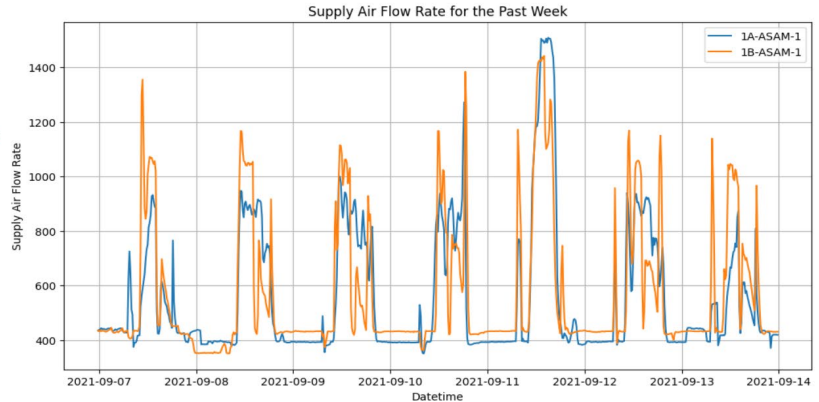


Figure 3 (a) Auto-generated SPARQL query for the question “What is the range of values for the supply air flow rate this past week?” (step 1) (b) Auto-generated visualization generated by the framework (step 3). It shows the trended values of each of the two supply air flow sensors for the two AHUs supplying conditioned air to the two cells in FLEXLAB. From this chart, we are able to see when the flow rate was the minimum and maximum, which is missing in the verbal output provided by the framework (step 4).

DISCUSSION

Our results indicate that agentic workflows can enhance interactions between practitioners and building automation systems, particularly for those without extensive computer science expertise. The framework demonstrates strong potential in generating visual representations to answer user queries, making building data more interpretable. However, while the framework provides meaningful verbal responses, it sometimes produces limited or incomplete answers, highlighting the necessity for clarification mechanisms.

A key example of this limitation is the handling of ambiguous queries, such as detecting "significant drops in airflow." The framework currently assumes a 10% threshold, which, while reasonable in some cases, may not align with user expectations. For such scenarios, incorporating an interactive clarification mechanism would allow the system to prompt users for additional input, improving response accuracy and relevance.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX brick: <https://brickschema.org/schema/Brick#>
PREFIX ref: <https://brickschema.org/schema/Brick/ref#>
SELECT ?ahu ?sup ?ts WHERE {
  ?ahu rdf:type brick:Air_Handling_Unit .
  ?sup brick:isPartOf ?ahu .
  ?sup ref:TimeseriesReference ?ts .
  ?sup rdf:type brick:Supply_Air_Flow_Sensor .
}
```

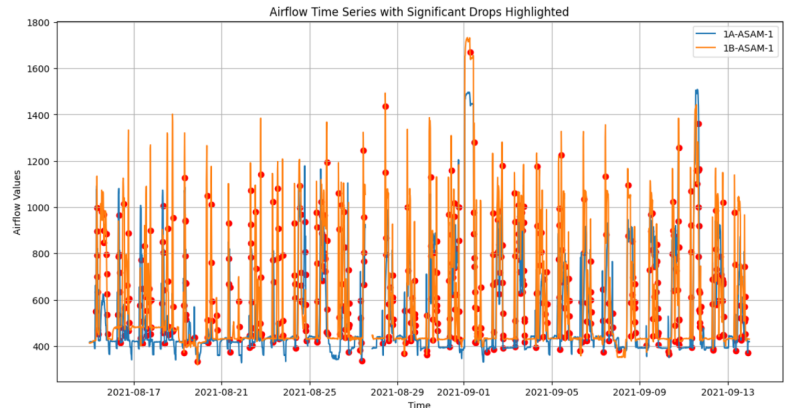


Figure 4 (a) Auto-generated SPARQL query for the question “Are there any significant drops in airflow that could indicate potential blockages?” (step 1) (b) Auto-generated visualization generated by the framework (step 3). It shows the trended values of each of the two supply air flow sensors for the two AHUs supplying conditioned air to the two cells in FLEXLAB. From this chart, we are able to see times when the supply air flow rate dropped significantly (the framework assumed 10% reduction as significant) and suddenly.

The current implementation has several constraints. First, our evaluation was conducted using only three sample questions on a single small commercial building testbed. Expanding the framework to larger datasets and more diverse building configurations is necessary to assess its generalizability. Second, while the system functions autonomously, fully eliminating human oversight is impractical. In real-world applications, an interactive feedback loop would be crucial for refining responses. Users should have the ability to (1) refine their queries based on initial results, (2) request clarifications when responses are vague, and (3) provide feedback when outputs deviate from their intended meaning. Integrating these feedback mechanisms would not only enhance usability but also improve the framework’s accuracy by leveraging user corrections to refine subsequent query and response generation. Future iterations should explore adaptive models that learn from user interactions, enabling a more robust and context-aware system for building analytics.

CONCLUSION

This paper introduces a novel agentic workflow that leverages LLMs to bridge the gap between natural language queries and structured semantic ontologies in BAS. By automating SPARQL query generation, time-series data retrieval, and analytical visualization, our framework shows promise in reducing the technical expertise required to interact with complex BAS data, empowering facility managers, engineers, and other non-specialists to derive actionable insights through intuitive, conversational interfaces.

The experimental results show that LLMs - when supported with RAG of building-specific query examples - can effectively convert natural language questions into accurate SPARQL queries while producing meaningful visualizations and summaries. However, several challenges emerged, including handling linguistic ambiguity, maintaining query consistency, and ensuring robust performance across different building configurations. While the system successfully interpreted queries about damper positions and airflow trends, it revealed limitations when processing subjective terms like “significant drops,” where it defaulted to a 10% threshold assumption. Furthermore, while the framework performed well in our controlled testbed environment, adapting it to new buildings would require adjustments to the prompts to reflect the differences in how time-series data is stored (e.g., obtained via BACnet, stored in a database, accessed via an API).

Looking ahead, future research should focus on scaling the framework to larger and more varied building datasets, ensuring generalizability across different ontologies and BAS implementations. Incorporating human-in-the-loop feedback—such as query disambiguation prompts and error correction loops—could further enhance reliability, while hybrid approaches combining LLMs with a knowledge base of domain-specific rules may improve precision in critical applications like fault detection. Additionally, exploring closed-loop automation, where the system not only analyzes data per user request but also suggests operational adjustments, could unlock new possibilities for intelligent building management by transforming passive analytics into active optimization.

Ultimately, this work represents a meaningful step toward democratizing BAS analytics, making BAS data accessible to a broader audience while maintaining the rigor of structured ontologies. As LLM capabilities continue to evolve, integrating these technologies with building-specific KGs and domain knowledge promises to transform how we monitor, analyze, and optimize the built environment—moving us closer to a future where anyone can interrogate their building’s data as effortlessly as asking a question.

ACKNOWLEDGMENTS

This work was supported by the Assistant Secretary for Energy Efficiency and Renewable Energy, Building Technologies Office, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

REFERENCES

- Avila, Caio Viktor S., 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)*, 277–84. IEEE.
- Balaji, Bharathan, Arka Bhattacharya, Gabriel Fierro, Jingkun Gao, Joshua Gluck, Dezhi Hong, Aslak Johansen, 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*. New York, NY, USA: ACM.
- Baris, Ozan, Yizhuo Chen, Gaofeng Dong, Liying Han, Tomoyoshi Kimura, Pengrui Quan, Ruijie Wang, et al. 2025. "Foundation Models for CPS-IoT: Opportunities and Challenges." *arXiv [Cs.LG]*. arXiv. <http://arxiv.org/abs/2501.16368>.
- Bennani, Imane Lahmam, 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*, pp. 150-159. New York, NY, USA: ACM.
- Bizer, Christian, and Andreas Schultz. 2009. "The Berlin SPARQL Benchmark." *International journal on semantic web and information systems* 5 (2): 1–24.
- Bowen Jin, Gang Liu, Chi Han, Meng Jiang, Heng Ji and Jiawei Han. 2023. "Large Language Models on Graphs: A Comprehensive Survey." *IEEE Transactions on Knowledge and Data Engineering*, 36: 8622-8642.
- Chakraborty, Abir. 2024. "Multi-Hop Question Answering over Knowledge Graphs Using Large Language Models." *arXiv [Cs.AI]*. arXiv. <http://arxiv.org/abs/2404.19234>.
- Chiosa, Roberto, Marco Savino Piscitelli, Marco Pritoni, and Alfonso Capozzoli. 2024. "A portable application framework for energy management and information systems (EMIS) solutions using Brick semantic schema." *Energy and Buildings* 323: 114802.
- Google DeepMind. 2022. "Competitive Programming with AlphaCode." 2022. <https://deepmind.google/discover/blog/competitive-programming-with-alphacode/>.
- Google DeepMind. 2023. *Code_contests*. Github. https://github.com/google-deepmind/code_contests.
- Hogan, Aidan, Eva Blomqvist, Michael Cochez, Claudia D'amato, Gerard De Melo, Claudio Gutierrez, Sabrina Kirrane, et al. 2022. "Knowledge Graphs." *ACM Computing Surveys* 54 (4): 1–37.
- Kovriguina, Liubov, Roman Teucher, Daniil Radyush, and Dmitry Mouromtsev. 2023. "SPARQLGEN: One-Shot Prompt-Based Approach for SPARQL Query Generation." In *SEMANTICS (Posters & Demos)*.
- Koziolek, Heiko, Virendra Ashiwal, Soumyadip Bandyopadhyay, and Chandrika K. 2024. "Automated Control Logic Test Case Generation Using Large Language Models." In *2024 IEEE 29th International Conference on Emerging Technologies and Factory Automation (ETFA)*, 166:1–8. IEEE.
- Lahitani, Alfirna Rizqi, Adhistya Erna Permanasari, and Noor Akhmad Setiawan. 2016. "Cosine Similarity to Determine Similarity Measure: Study Case in Online Essay Assessment." In *2016 4th International Conference on Cyber and IT Service Management*, 1–6. IEEE.
- Lewis, Patrick, Ethan Perez, Aleksandara Piktus, F. Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Kuttler, et al. 2020. "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks." Edited by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin. *Neural Information Processing Systems* abs/2005.11401 (May): 9459–74.
- Liang, Jacky, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. 2023. "Code as Policies: Language Model Programs for Embodied Control." In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 9493–9500. IEEE.
- Mulayim, Ozan Baris, 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*, 312–17. New York, NY, USA: ACM.
- Mulayim, Ozan Baris, Gabe Fierro, Mario Bergés, and Marco Pritoni. 2025. "Towards Zero-shot Question Answering in CPS-IoT: Large Language Models and Knowledge Graphs." In *The 2nd International*

Workshop on Foundation Models for Cyber-Physical Systems Internet of Things (FMSys '25), May 6–9, 2025, Irvine, CA, USA. ACM, New York, NY, USA.

- Pritoni, Marco, Anand Prakash, Weiping Huang, Steven Bushby, Parastoo Delgoshaei, Michael Poplawski, Avijit Saha et al. 2024. "Digital and Interoperable: The future of building automation is on the horizon. What's in it for me?." *ACEEE Summer Study on Energy Efficiency in Buildings*.
- Rony, Md Rashad Al Hasan, Uttam Kumar, Roman Teucher, Liubov Kovriguina, and Jens Lehmann. 2022. "SGPT: A Generative Approach for SPARQL Query Generation from Natural Language Questions." *IEEE Access: Practical Innovations, Open Solutions* 10: 70712–23.
- Schmidt, Michael, Thomas Hornung, Georg Lausen, and Christoph Pinkel. 2009. "SP²Bench: A SPARQL Performance Benchmark." In *2009 IEEE 25th International Conference on Data Engineering*, 222–33. IEEE.
- Siahaan, Steven Gianmarg Haposan, and Masayu Leylia Khodra. 2024. "Sentence Embedding-Based Soft Clustering for Analyzing Texts of Student Outcomes." In *2024 IEEE International Conference on Data and Software Engineering (ICoDSE)*, 205–10. IEEE.
- Soru, Tommaso, Edgard Marx, André Valdestilhas, Diego Esteves, Diego Moussallem, and Gustavo Publio. 2018. "Neural Machine Translation for Query Construction and Composition." *arXiv [Cs.CL]*. arXiv. <http://arxiv.org/abs/1806.10478>.
- W3C. 2008. "SPARQL 1.1 Overview." 2008. <https://www.w3.org/TR/sparql11-overview/>.
- W3C. 2014. "RDF - Semantic Web Standards." 2014. <https://www.w3.org/RDF/>.
- Yang, Shuangtao, Mao Teng, Xiaozheng Dong, and Fu Bo. 2023. "LLM-Based SPARQL Generation with Selected Schema from Large Scale Knowledge Base." In *Communications in Computer and Information Science*, 304–16. Communications in Computer and Information Science. Singapore: Springer Nature Singapore.
- Zafar, Hamid, Giulio Napolitano, and Jens Lehmann. 2018. "Formal Query Generation for Question Answering over Knowledge Bases." In *The Semantic Web*, 714–28. Lecture Notes in Computer Science. Cham: Springer International Publishing.
- Zhang, Jiayi, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, et al. 2024. "AFlow: Automating Agentic Workflow Generation." *arXiv [Cs.AI]*. arXiv. <http://arxiv.org/abs/2410.10762>.
- Zhu, Yuqi, Xiaohan Wang, Jing Chen, Shuofei Qiao, Yixin Ou, Yunzhi Yao, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2024. "Llms for knowledge graph construction and reasoning: Recent capabilities and future opportunities." *World Wide Web* 27, no. 5: 58.