

# Combined Large Language Models and Ontology Approach for Energy Consumption Analysis Software\*

Oleksandr Vyshnevskyy<sup>1,\*</sup> and Liubov Zhuravchak<sup>1,†</sup>

<sup>1</sup> Lviv Polytechnic National University, Bandery street 12, Lviv, Ukraine

## Abstract

The paper explores the potential of leveraging Large Language Models for human interaction with a digital twin representing a building's energy consumption system. A novel method is proposed for constructing a digital representation of building energy consumption. This approach integrates a previously developed domain ontology, a graph database, and Large Language Models to enable intuitive, natural language access to energy performance data via a knowledge-driven interface.

Within the scope of this research, a domain-specific knowledge base was developed using the Neo4j graph database and an ontological framework. This knowledge base encodes relationships between buildings, meters, consumption indicators, weather stations, and climate data. A prototype chatbot was implemented using the LangChain framework, an agent-based architecture, and Retrieval-Augmented Generation. The study evaluates the ability of the Gpt-4o-mini model to interpret energy-related queries, leverage semantic relations within the knowledge graph, generate Cypher queries for Neo4j, execute them, and provide context-aware responses based on custom-designed agent prompts.

The proposed ontology-based framework for energy analysis formalizes domain knowledge into a machine-readable structure, enhancing human understanding and interaction through natural language interfaces. This facilitates more informed planning and decision-making. The results confirm that digital twins can support system management by detecting anomalies, forecasting trends, and identifying operational issues.

## Keywords

Retrieval Augmented Generation, Large Language Model, ontology, knowledge graph, energy

## 1. Introduction

Ontologies have become foundational in modeling processes across various industries, fundamentally transforming the way physical systems are described. Through structured representation, ontologies simplify interaction, data integration, and automation of various building management tasks. Semantic ontologies provide a formalized structure for representing knowledge in a machine-readable format, enabling the structured description of complex systems [1]. In the field of building energy consumption, the implementation of ontologies has changed the way processes are modeled in building systems, ensuring a standardized interoperable language. The complexity of learning involved in developing semantic models and queries remains a significant challenge due to the labor-intensive process and the need for specialized expertise. Semantic models simplify the development and deployment of applications by providing a standardized structure to understand spatial and functional relationships between equipment, allowing generalizations to detect faults or diagnose and optimize control systems. Some researchers have utilized time series measurements for more accurate classification of specific sensor data. These approaches work well for determining the class of each sensor and providing information on the relationships between sensors and associated equipment. However, much less research has been done on data-driven approaches to determining spatial and functional relationships between equipment to provide contextual information for control elements and analytics.

\*International Workshop on Computational Intelligence, co-located with the IV International Scientific Symposium "Intelligent Solutions" (IntSol-2025), May 01-05, 2025, Kyiv-Uzhhorod, Ukraine

<sup>1\*</sup> Corresponding author.

<sup>†</sup> These authors contributed equally.

✉ oleksandr.k.vyshnevskyy@lpnu.ua (O. Vyshnevskyy); liubov.m.zhuravchak@lpnu.ua (L. Zhuravchak);

id 0009-0005-4857-9669 (O. Vyshnevskyy); 000-0002-1444-5882 (L. Zhuravchak);



© 2025 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

Generative Artificial Intelligence refers to an uncontrolled or partially controlled machine learning structure that creates content using statistical data derived from training on existing digital content (e.g., text, video, images, and audio). A Large Language Model (LLM) is a statistical mathematical model of token distribution in a large publicly available volume of human-created text, which, after training, can generate human-like language. A Generative Pre-trained Transformer (GPT) is a system based on LLM, designed to generate or statistically predict sequences of words, code, or other data, starting from input known as a prompt. GPT is based on transformer architecture, which processes large volumes of publicly available data in parallel. Recent studies have shown that the potential of LLMs can be utilized to create ontologies from unstructured text, extend existing ontologies to represent new concepts, and validate ontologies.

The Retrieval-Augmented Generation (RAG) approach has proven promising for enhancing LLM performance in domain-specific tasks. RAG combines the broad knowledge of pre-trained language models with the ability to retrieve and integrate relevant external information. In building semantic models, RAG offers advantages such as improved accuracy, domain adaptation, inclusion of up-to-date knowledge, and a reduction in "hallucinations" (responses containing factually incorrect information). However, its success depends on the relevance of the retrieved information, maintaining coherence, and contextual understanding in ontologies. Intellectual chatbots based on LLMs have gained tremendous popularity worldwide for a wide range of industrial applications, especially since the end of 2022.

Before the release of ChatGPT with integrated LLM, conversational chatbots were already widely used in various sectors such as education, manufacturing, healthcare, and government services. There are three main types of conversational chatbots: rule-based chatbots, live chatbots, and basic AI-based intelligent chatbots. The first two types interact by following predefined rules and, accordingly, involve chatbot software and human conversations to provide customer services. The third type, basic AI-based chatbots, facilitate communication beyond predefined commands without human interaction, laying the foundation for advanced intelligent chatbots. LLM-based intelligent chatbots, including ChatGPT, have an immense number of model parameters [2].

LLMs based on GPT architecture have shown considerable promise in interpreting energy sector tasks using input data in natural language format [3]. This study explored the capabilities and limitations of LLMs in applying them to the electricity sector. The effectiveness of LLMs in answering general energy system queries, generating code, and analyzing data was discussed. Additionally, through the RAG approach, LLMs can serve as knowledge bases for documentation and assist in tasks such as training operators. The multimodal capabilities of LLMs can be beneficial for diagnosing equipment malfunctions and remote monitoring. LLMs have demonstrated strong capabilities in detecting correlations between objects (text, images, data), but they are still weak in solving problems related to physics, which typically involve complex mathematical principles.

The aim of the work is to model the building energy consumption system using a digital twin based on an ontological approach, a graph database, and LLMs. This will increase the energy efficiency of the building and provide an easily accessible interface for direct communication between the user and the knowledge base.

The object of the study is the energy consumption of buildings.

The subject of the study is the ontological approach and the agent-based method of retrieval augmented generation using LLMs.

## 2. Related Works

To achieve the goals of limiting global warming, it is necessary to reduce energy consumption in buildings, which account for about 30% of global greenhouse gas emissions. Evaluating energy consumption during building design using performance simulations is crucial. However, manually enriching missing semantic information is still a very labor-intensive process. In the study [4], a new methodology was proposed for automatically enriching missing information using Semantic Textual Similarity and improved tuning of LLM. The authors matched room types and structures with

missing thermal properties using the most semantically similar pairs in the BIM model and corresponding databases. Three practical examples were used for the improved tuning of LLM. Various enhanced tuning strategies (using different loss functions, adding opposite pairs of words, or domain-related abbreviations) significantly increased matching accuracy.

The work [5] explores how LLMs can assist in solving problems related to the development of semantic models, focusing on their role in query construction for semantic models, specifically using the Brick Schema ontology. It is noted that although much effort has been made to capture the nuances of system construction, tools that enable building managers and application developers to efficiently create these models and make queries have not been properly developed. Therefore, the lack of necessary, user-friendly tools limits users with advanced knowledge of programming and information systems, as they also need to have a deep understanding of energy consumption systems, their components, and modeling options.

Query generation allows users to obtain structured information from the semantic model, which is crucial for fault detection in diagnostics and analytics. LLM studies have shown the potential for creating SPARQL (Protocol and RDF Query Language) queries with minimal training, offering greater adaptability across domains. A method called SGPT (SPARQL GPT) was introduced, which bypasses the need for manual SPARQL creation by using LLMs to learn graph patterns and generate queries. Based on this, the SPARQLGEN (SPARQL Generation) approach used GPT-3 in a one-shot SPARQL generation structure, where providing relevant context significantly improved the quality of generated queries [6].

Existing research indicates that while the use of LLMs in building energy efficiency remains relatively unexplored, interest in this domain is growing quickly, with a rising number of studies emerging. This work reflects the approach developed by the authors, using modern tools, including ontologies [7] and LLM, for creating semantic queries for digital building models.

Study [8] explored how LLMs can acquire domain-specific knowledge in Heating, Ventilation, and Air Conditioning (HVAC). The results showed that LLMs possess sufficient understanding and expertise to help users successfully complete the ASHRAE Certified HVAC Designer exam. The models demonstrated human-level performance in tackling complex problems, reasoning, and learning, similar to skilled HVAC professionals.. Three key knowledge properties were examined: recall, analysis, and application – on twelve typical models. Additionally, the GPT-3.5 model passed the exam twice out of five attempts. This demonstrated that some LLMs, such as GPT-4 and GPT-3.5, have great potential for assisting or replacing humans in the design and operation of HVAC systems.

In the work [9], the potential of generative pre-trained transformers in building energy management based on data analysis was investigated. The potential of the GPT-4 model was explored in three energy consumption data analysis scenarios for buildings: energy load forecasting, fault diagnosis, and anomaly detection. An approach was proposed to assess the performance of GPT-4's capabilities in creating software code for energy load forecasting, device fault diagnosis, and detecting patterns of system anomaly behavior. It was shown that GPT-4 can automatically solve most data analysis tasks in the HVAC field.

The work [10] presents a study of a prototype virtual university support agent, built on an LLM model, to resolve queries from students, teachers, and staff. The integration of generative artificial intelligence and the natural language features inherent in LLMs was explored to overcome customer service shortcomings. As a result, the university support agent provided a viable question-and-answer interface for students, teachers, and administrators to learn about the university's guidelines and policies.

### 3. Experiments

The research was conducted on a laptop with the following characteristics: an 8-core AMD Ryzen 7 6800H 3.20 GHz processor and a 12-core AMD Radeon 680M 2200 MHz graphics processor. A Neo4j graph database, PyCharm development environment, and Python programming language with LangChain, FastApi, and Streamlit libraries were used.

The ontology, previously developed in our work [7], describes a range of energy and construction terms and relationships. Ontology is a form of knowledge representation in a domain model that makes data smarter, transfers part of the program logic, and generates new data.

Let us consider the relationship between ontologies and graph databases. As of 2025, according to [11], Neo4j holds the top position in popularity among other graph data stores. Neo4j implements following ontology concepts: interoperability (defining a shared vocabulary) and logical inference (inferencing), which is the result of knowing fragments of the vocabulary.

### 3.1. Neo4j graph database and ontology

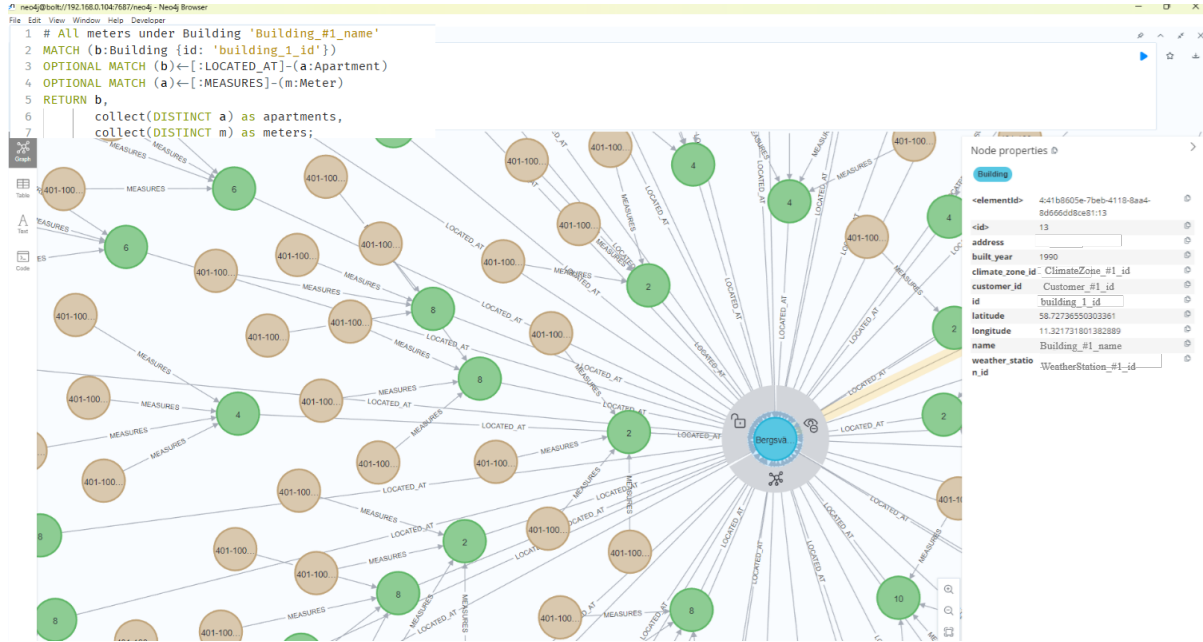
Neo4j is a native graph database that implements a true graph model all the way to the storage level. It uses Cypher query language, designed specifically for working with graphs. Since 2007, it has grown into a versatile platform with support for multiple programming languages and frameworks. Its ecosystem includes the Graph Data Science library, which enables advanced analytics and machine learning on graph data. It has the cloud service Neo4j AuraDB and can be run in a Docker container or a Kubernetes cluster. In this study, we worked with a local deployment and used Neo4j Desktop version 1.6.1.

Neo4j can load and write an ontology in RDF format. Therefore, as shown in Figure 1: Representation of the OWL ontology in the Neo4j knowledge graph database the ontology we developed can be loaded into the Neo4j knowledge base using the Neosemantics (n10s) extension. Until now, reasoning in RDF and OWL (Web Ontology Language) has been applied only to full-fledged triple stores or special reasoning mechanisms. Neo4j can be enhanced with reasoning capabilities for RDF data, even though it isn't a triple store and doesn't support SPARQL directly. Instead, it uses the Cypher language, and many SPARQL queries can be converted into Cypher for semantic querying. From the user's perspective, Neo4j combines two technologies into one platform: a transactional analytical graph system and an RDF/OWL reasoning mechanism capable of providing complex semantic Cypher queries through the materialized graph in Neo4j. Studies have shown that obtaining all derived facts (so-called materialization) using Neo4j demonstrates a much lower reasoning time growth rate as the amount of data increases compared to any other system. Therefore, materialization is the key to efficient real-world queries [12]. Without prior materialization, a triple store with reasoning must temporarily fetch all answers and relevant facts for each individual query on demand.



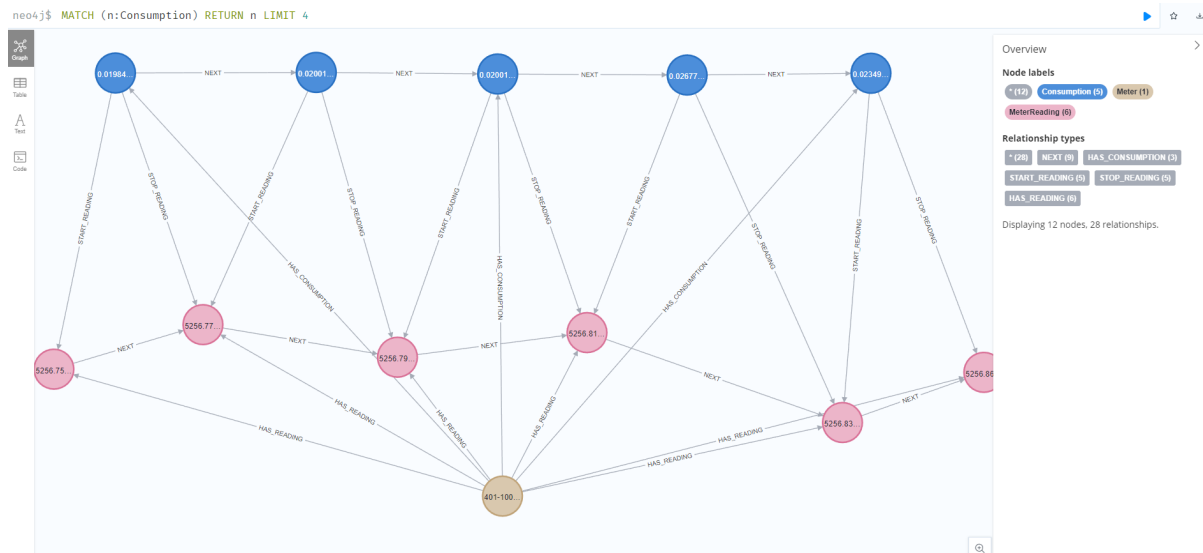
**Figure 1:** Representation of the OWL ontology in the Neo4j knowledge graph database.

According to the ontology we developed earlier [7], the following nodes and relationships were created in the graph database. The Customer (tenant) is connected to the Building via the HAS relationship, the Building is connected to the Apartment via the LOCATED\_AT relationship, and the Meter is connected to the Apartment via the MEASURES relationship. A sample of the relationships between a building, the apartments it contains, and the meters that measure their energy consumption is shown in Figure 2. Here, a Cypher query is provided to find all meters that take measurements for all apartments located in a building.



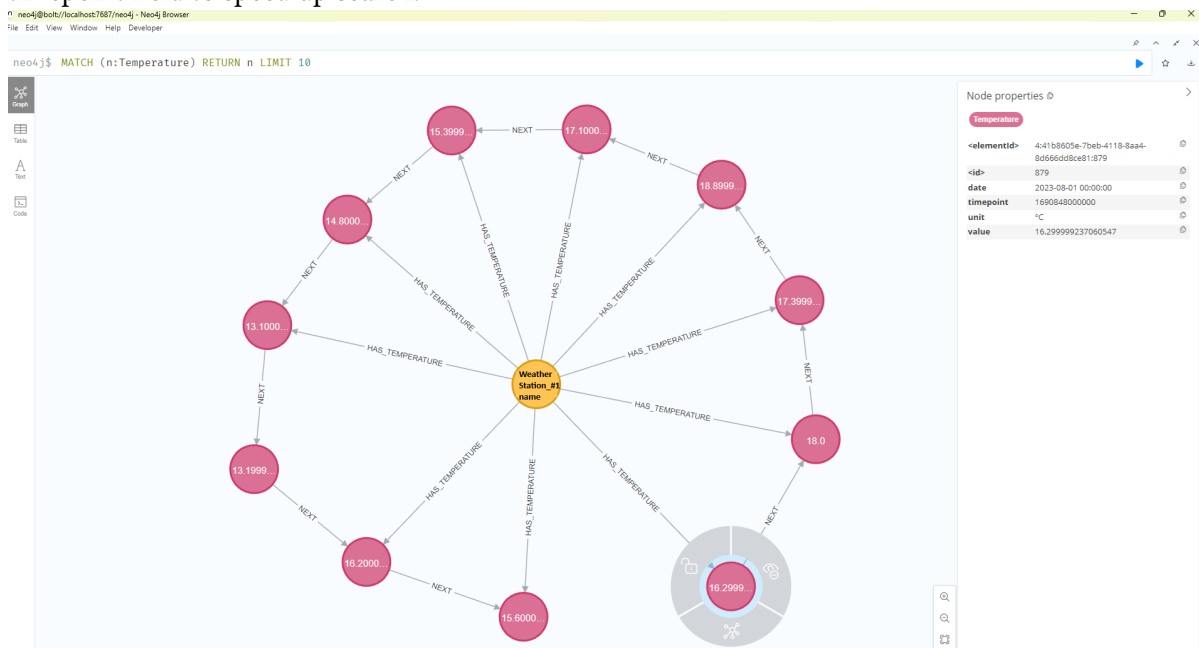
**Figure 2:** Representation of buildings, meters, and relationships in the graph.

The Meter node is connected to the Consumption node via the HAS\_CONSUMPTION relationship and to the MeterReading node via the HAS\_READING relationship. Since the process of collecting meter readings is continuous, with specific intervals (e.g., new values are read every hour), it is important to semantically represent the sequential connection of readings: the MeterReading node at time  $t$  is connected to the MeterReading node at time  $t+1$  via the NEXT relationship. Therefore, the problem of time series becomes a graph problem and can be effectively solved using existing graph algorithms. Since consumption per unit of time is the difference between the meter reading at time  $t$  and at time  $t+1$ , this is reflected in the semantic relationship of the readings. The Consumption node at time  $t$  is connected to the MeterReading node at time  $t$  via the START\_READING relationship and to the MeterReading node at time  $t+1$  via the STOP\_READING relationship. Consumption is also a sequential process, so the Consumption node at time  $t$  is connected to the Consumption node at time  $t+1$  via the NEXT relationship. A sample of sequential relationships between meter readings and consumption over several consecutive hours is shown in Figure 3.



**Figure 3:** Representation of meter readings/consumption data as a sequence of nodes in a graph.

Node WeatherStation (weather forecast station) is connected to the node Temperature (ambient temperature) by the HAS\_TEMPERATURE relationship. Weather station readings are a continuous process with a specific interval (e.g., new values are received every 1 day), so the node Temperature at time  $t$  is connected by the NEXT relationship to the next node Temperature at time  $t+1$ . A sample of sequential connections between ambient temperature readings over several consecutive days is shown in Figure 4. The nodes Consumption, MeterReading, and Temperature have an index for the timepoint field to speed up search.



**Figure 4:** Representation of ambient temperature data as a sequence of nodes in the graph.

Storing data in a graph structure has the following advantages:

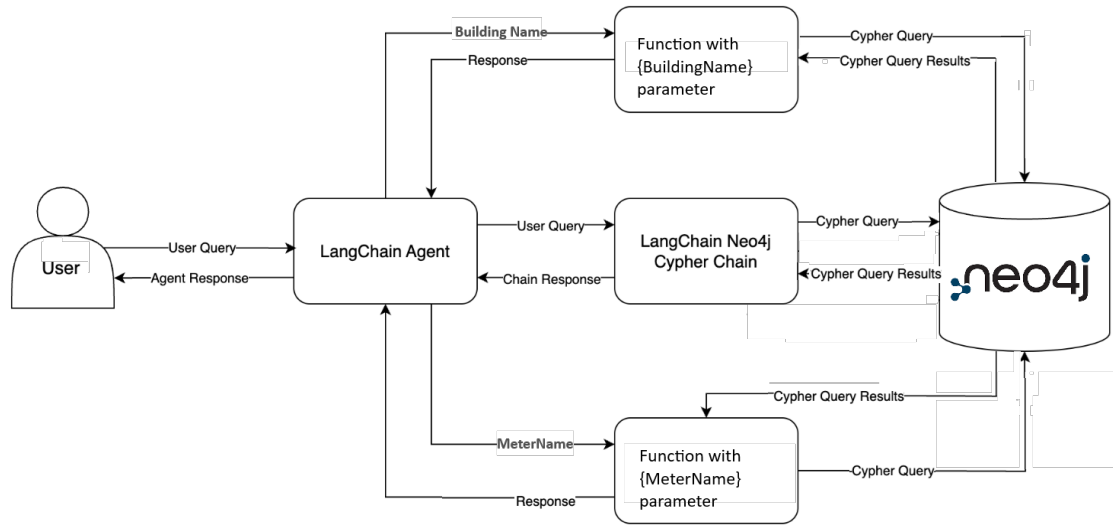
1. Natural and simple modeling of real-world relationships without complex joins.
2. Efficient processing of connected data via graph traversal.
3. Flexible structure.
4. Faster performance for complex, multi-relationship queries compared to relational databases.

5. Pattern discovery – support for pattern matching queries facilitates the search for specific structures in data.

Graph databases simplify design and querying for complex data by avoiding intricate table joins. Their flexible structure also helps LLMs generate more accurate queries with fewer errors. Compared to relational databases, LLMs must handle schema knowledge and relationships between keys, which can increase errors in generating SQL queries.

### 3.2. Retrieval Augmented Generation

This study utilized an ontological approach to build a knowledge model for the domain of energy consumption in buildings. An agent-based Retrieval Augmented Generation method was applied, and a set of tools was developed to interact with the knowledge base and the end-user, using LLMs. The diagram shown in Figure 6 illustrates the working principle of the developed chatbot application and demonstrates how a user query in natural language is transformed into a Cypher query understandable by the Neo4j graph database; the execution result is passed to the agent, which then returns the final response to the user. The GPT-4o-mini model from OpenAI was used, which efficiently solves tasks due to its low cost and latency.



**Figure 5:** User interaction scheme with the knowledge base using Large Language Models.

Here is a brief description of each component:

1. **LangChain agent** –upon receiving a user request, the agent decides which tool to call and what input data to provide to the tool. The agent then monitors the tool's response and decides what to return to the user (agent response).
2. **Neo4j graph database** stores structured data about buildings, apartments, residents, meters, and consumption.
3. **LangChain Neo4j Cypher Chain tool** translates user queries into Cypher, executes them on the Neo4j database, and returns the results through a question-answering chain using the GraphCypherQChain class..
4. **Tool function with building name parameter** is used when the LangChain agent successfully extracts the building name from the user query. The building name is passed as an input parameter for a Python function containing logic specific to the building. For example, it could make a request to an external resource providing metadata about the building, and this result is returned to the agent.



5. **Tool function with meter name parameter** is used when the LangChain agent successfully extracts the meter name from the user query. The meter name is passed as an input parameter for a Python function that provides a consumption forecast for that specific meter for the next 24 hours, and the result is returned to the agent.

The open source code for developed building energy agent is provided in Figure 6.

```
1 import os
2 from langchain.chains import GraphCypherQAChain
3 from langchain.prompts import PromptTemplate
4 from langchain_community.graphs import Neo4jGraph
5 from langchain_openai import ChatOpenAI
6 from langchain import hub
7 from langchain.agents import AgentExecutor, Tool, create_openai_functions_agent
8
9 graph = Neo4jGraph(url=os.getenv("NEO4J_URI"), username=os.getenv("NEO4J_USERNAME"), password=os.getenv("NEO4J_PASSWORD"))
10 graph.refresh_schema()
11
12 cypher_generation_prompt = PromptTemplate(
13     input_variables=["schema", "question"], template=cypher_generation_template
14 )
15
16 qa_generation_prompt = PromptTemplate(
17     input_variables=["context", "question"], template=qa_generation_template
18 )
19
20 energybot_cypher_chain = GraphCypherQAChain.from_llm(
21     cypher_llm=ChatOpenAI(model="gpt-4.0-mini", temperature=0),
22     qa_llm=ChatOpenAI(model="gpt-4.0-mini", temperature=0),
23     graph=graph,
24     verbose=True,
25     qa_prompt=qa_generation_prompt,
26     cypher_prompt=cypher_generation_prompt,
27     validate_cypher=True,
28     top_k=100,
29 )
30
31 energybot_agent_prompt = hub.pull("hwchase17/openai-functions-agent")
32
33 tools = [
34     Tool(
35         name="Graph",
36         func=energybot_cypher_chain.invoke,
37         description="Useful for answering questions about estates, buildings, apartments, meters, climate zones. For instance, if the prompt is 'How many meters exist?', the input should be 'How many meters exist?'.",
38     ),
39     Tool(
40         name="Meter Prediction",
41         func=get_meter_consumption_prediction,
42         description="Useful for answering questions about meters consumption prediction".
43     ),
44     ...
45 ]
46
47 chat_model = ChatOpenAI(model="gpt-4.0-mini", temperature=0)
48
49 energybot_rag_agent = create_openai_functions_agent(
50     llm=chat_model,
51     prompt=hospital_agent_prompt,
52     tools=tools,
53 )
54
55 energybot_rag_agent_executor = AgentExecutor(
56     agent=energybot_rag_agent,
57     tools=tools,
58     return_intermediate_steps=True,
59     verbose=True,
60 )
```

**Figure 6:** Open source code for developed building energy agent.

For communication with the agent, an API endpoint with the POST method is added. It accepts a single parameter with the user's question, runs the agent, and returns the result from the LLM using one of the defined tools. FastAPI, a Python web framework, was used to build the API. Figure 7 shows the documentation page for the developed API methods.

To create a graphical web interface for interaction with the end user, the Streamlit library was used. The deployment of the developed application was implemented using the Docker Compose container manager with two services: one for the API and one for the graphical interface. The user interface accepts the user's queries and sends a POST request to the agent's API endpoint. The user is also provided with an explanation of how the agent generated the response. This can be used during testing to check if the agent invoked the correct tool and provided the correct response. The result of searching for data about buildings and meters in the developed application is shown in Figure 8, where the user asked several questions: "Which meters were recently connected and when exactly?",



“Which meters are connected to the building {Building Name}?”, “Which building is the oldest and how many years old is it?”.

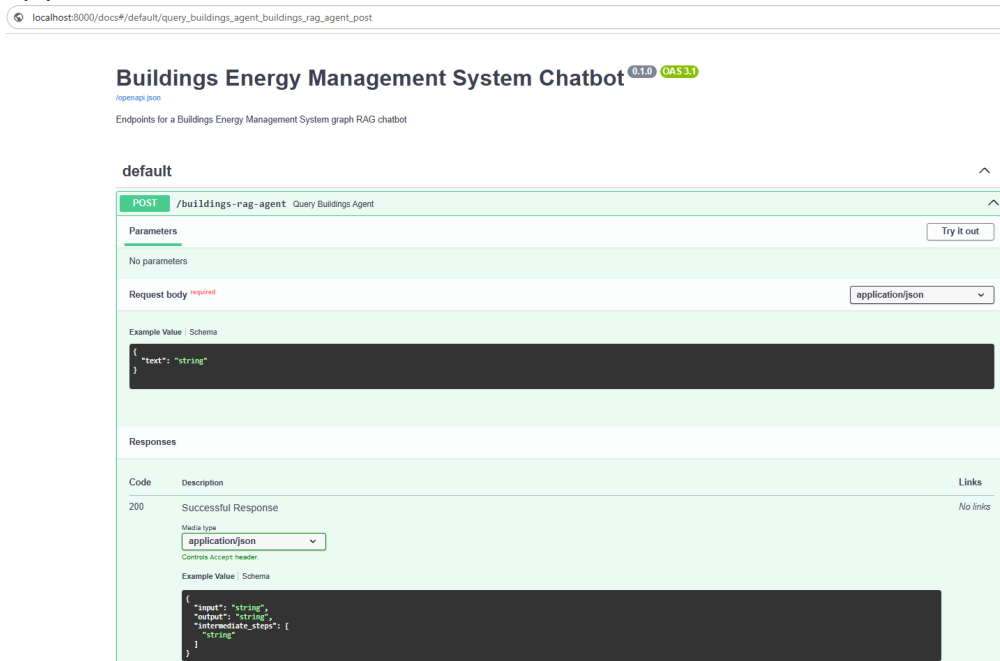


Figure 7: Chatbot API for communication with LLM agent.

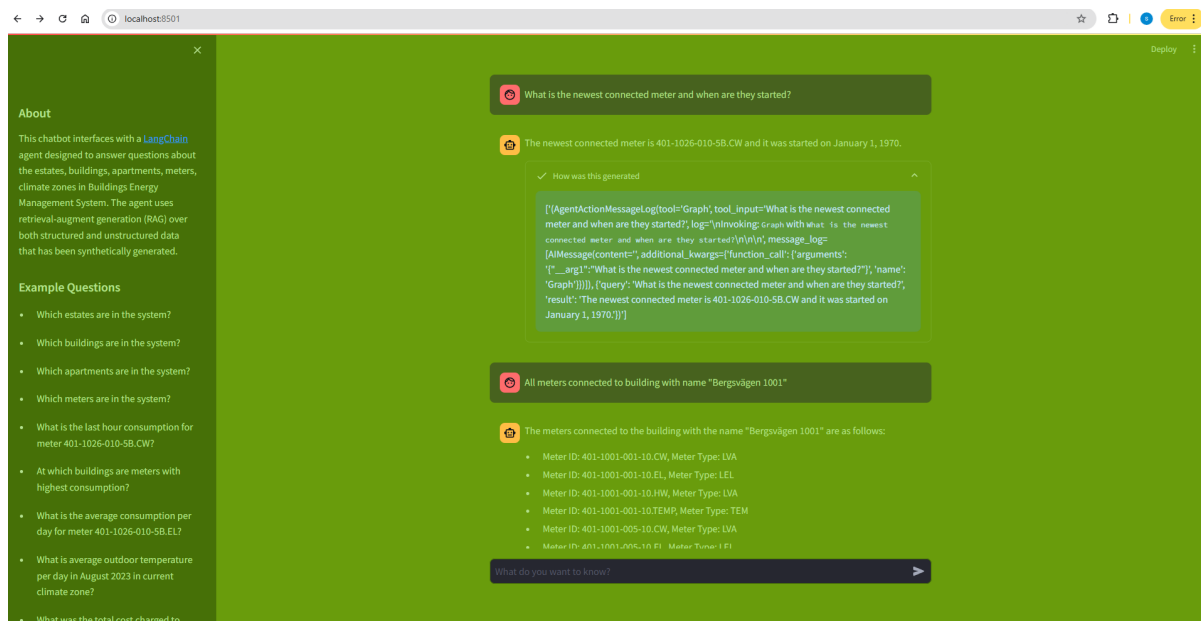


Figure 8: Result of searching for building and meter data in the developed application.

Details of the query 'Which building is the oldest and how old is it?' can be seen in the API service console. It is noted that the agent selected the correct tool, GraphCypherQACChain, and generated the Cypher query shown in Figure 9, which, after execution, returned the value '34'. This value was then passed to the agent, which formulated the response: 'The oldest building is {Building Name}, and it is 34 years old,' with the agent knowing that the unit of measurement is indeed a year, not a day or an hour, due to the generated prompts.

```
Select Administrator: C:\Windows\System32\cmd.exe - docker-compose up --build chatbot_api

chatbot_api-1 | > Entering new AgentExecutor chain...
chatbot_api-1 | / Invoking: `Graph` with `What is the oldest building and how old are they?`
chatbot_api-1 | > Entering new GraphCypherQAChain chain...
chatbot_api-1 | Generated Cypher:
chatbot_api-1 | MATCH (b:Building)
chatbot_api-1 | / RETURN b.name AS oldest_building,
chatbot_api-1 |         toInteger(date().year) - toInteger(b.built_year) AS age
chatbot_api-1 | ORDER BY age DESC
chatbot_api-1 | LIMIT 1
chatbot_api-1 | Full Context:
chatbot_api-1 | [{"oldest_building": "Bergsvägen 1001", "age": 34}]
chatbot_api-1 | > Finished chain.
chatbot_api-1 | {'query': 'What is the oldest building and how old are they?', 'result': 'The oldest building is Bergsvägen 1001, and it is 34 years old.'}
chatbot_api-1 | > Finished chain.
```

**Figure 9:** Result of the LLM agent's operation, generation of the Cypher query and response.

Let's consider the query 'Are there anomalies for the meter {MeterName}?. In the API service console, we can see that the agent, using the GraphCypherQAChain tool, generated the Cypher query shown in Figure 10. This query checks if there is any consumption where the ratio of the difference between consecutive meter readings to the time difference is greater than 10. This query to the Neo4j knowledge base returned an empty array '[]', which was passed to the agent, which then formulated the final response: 'No anomalies found for the meter {MeterName}'. It should be noted that there were no predefined prompts for anomaly detection; the GraphCypherQAChain chain automatically generated this Cypher query based on the structure of the built knowledge base. Thanks to modifications in the prompts, an instruction can be added to define an anomalous value differently, considering the characteristics of the meter type. This way, different parameters will be applied for meters of different utilities.

**About**

This chatbot interfaces with a [LangChain](#) agent designed to answer questions about the estates, buildings, apartments, meters, climate zones in Buildings Energy Management System. The agent uses retrieval-augment generation (RAG) over both structured and unstructured data that has been synthetically generated.

Is there any anomalies for meter {MeterName} ?

There are no anomalies found for meter {MeterName}

✓ Explain answer?

What do you want to know?

```
Administrator: C:\Windows\System32\cmd.exe - docker-compose up --build chatbot_api

chatbot_api-1 | > Entering new AgentExecutor chain...
chatbot_api-1 | / Invoking: `Graph` with `Anomalies for meter {MeterName}`
chatbot_api-1 | > Entering new GraphCypherQAChain chain...
chatbot_api-1 | Generated Cypher:
chatbot_api-1 | MATCH (m:Meter {id: {MeterName}})-[:HAS_READING]->(mr:MeterReading)
chatbot_api-1 | / WITH mr.reading AS current_reading, mr.timepoint AS current_timepoint
chatbot_api-1 | MATCH (m)-[:HAS_CONSUMPTION]->(c:Consumption)-[:START_READING]->(mr)
chatbot_api-1 | WITH current_reading, current_timepoint, c.value AS start_value, c.timepoint AS start_timepoint
chatbot_api-1 | MATCH (m)-[:HAS_CONSUMPTION]->(c)-[:STOP_READING]->(mr)
chatbot_api-1 | WITH current_reading, current_timepoint, start_value, start_timepoint, c.value AS stop_value, c.timepoint AS stop_timepoint
chatbot_api-1 | WHERE (stop_value - start_value) / (stop_timepoint - start_timepoint) > 10
chatbot_api-1 | RETURN current_reading, current_timepoint, start_value, start_timepoint, stop_value, stop_timepoint
chatbot_api-1 | Full Context:
chatbot_api-1 | []
chatbot_api-1 | > Finished chain.
chatbot_api-1 | {'query': 'Anomalies for {MeterName}', 'result': 'I'm sorry, but there are no anomalies found for meter 401-1001-10.EL.'}
chatbot_api-1 | > Finished chain.
```

**Figure 10:** Generation of a Cypher query to search for anomalies for the meter.

Consider the query “What is the consumption profile of the meter {MeterName} during the summer period?”. In the API service console, we can see that the agent, using the GraphCypherQACHain tool, generated the Cypher query shown in Figure 11. This query searches for the meter consumption where the date falls within the range from June 1st to September 1st. This query to the Neo4j knowledge base returned an array of values, which was passed to the agent, which then generated the final response in the form of a list of consumption values for specific dates. It should be noted that no prior prompts were provided to generate the consumption profile, and the GraphCypherQACHain chain automatically generated this Cypher query based on the structure of the constructed knowledge base, automatically filtering the data and establishing the condition for the beginning and end of the summer period. Thanks to prompt modifications, it is possible to add instructions to display the date in the format "2023-08-01 04:00:00" (instead of the epoch format) and to ensure that electricity consumption is displayed in kilowatts.

**Chatbot Interface:**

What is Consumption usage profile for meter with meterId={MeterName} in summer?

The consumption usage profile for the meter with meterId {MeterName} in summer is as follows:

- Timepoint: 1690848000000, Consumption: 0.01984850535042763
- Timepoint: 1690851600000, Consumption: 0.020019005984067917
- Timepoint: 1690855200000, Consumption: 0.020018380135297775
- Timepoint: 1690858800000, Consumption: 0.026774289086461067
- Timepoint: 1690862400000, Consumption: 0.023498333990573883

These are the recorded consumption values for the specified meter during the summer period.

**Terminal Output:**

```

chatbot_api-1 > Entering new GraphCypherQACHain...
chatbot_api-1 Generated Cypher:
chatbot_api-1 MATCH (m:Meter {meterId: {MeterName}})-[:HAS_CONSUMPTION]->(c:Consumption)
chatbot_api-1 / WHERE c.date >= '2023-06-01' AND c.date < '2023-09-01'
chatbot_api-1 RETURN c.timepoint AS timepoint, c.value AS consumption
chatbot_api-1 ORDER BY c.timepoint
chatbot_api-1 Full Context:
chatbot_api-1 [{"timepoint": 1690848000000, "consumption": 0.01984850535042763}, {"timepoint": 1690851600000, "consumption": 0.020019005984067917}, {"timepoint": 1690855200000, "consumption": 0.020018380135297775}, {"timepoint": 1690858800000, "consumption": 0.026774289086461067}, {"timepoint": 1690862400000, "consumption": 0.023498333990573883}]
chatbot_api-1 > Finished chain.
chatbot_api-1 {'query': 'Consumption usage profile for meter with meterId= {MeterName} in summer', 'result': 'The consumption usage profile for the meter with meterId 401-1001-001-10.EL in summer is as follows:\n\nTimepoint: 1690848000000, Consumption: 0.01984850535042763\nTimepoint: 1690851600000, Consumption: 0.020019005984067917\nTimepoint: 1690855200000, Consumption: 0.020018380135297775\nTimepoint: 1690858800000, Consumption: 0.026774289086461067\nTimepoint: 1690862400000, Consumption: 0.023498333990573883\n\nThese are the recorded consumption values for the specified meter during the summer period.'}
chatbot_api-1 > Finished chain.
  
```

Figure 11: Cypher-query generation for the meter consumption summer profile.

## 4. Discussion

Small and medium enterprises currently lack an integrated system that combines data from various media sources into a centralized information system, limiting their ability to effectively navigate energy sustainability initiatives according to [13]. To address this gap, they introduced Energy Chatbot, a system using LLM integrated with a multi-source search add-on. This system covers various media sources, including news, government reports, industry publications, scientific research, and social networks. The chatbot delivers real-time data, assisting in identifying long-term sustainability plans and preserving a competitive edge in the evolving energy industry. This approach reduces costs through the use of open-source models. The Energy Chatbot provides access to up-to-date information, allowing the identification of long-term sustainability strategies and maintaining a competitive advantage in the evolving energy sector. The authors selected Llama

models (Llama2, Llama3, and Llama3.1) developed by Meta, which are updated versions trained on large datasets and known for their high performance compared to other open-source models.

The work [14] investigates the improvement of chatbot resources for medical services using LLM, effective boundary-efficient fine-tuning, and effective information processing strategies. For the chatbot structure development, the Lang Chain library and a two-dimensional data search methodology were used to engage the chatbot with a large medical service information repository, including record stacking and piecing close using vector stores. BLEU and ROUGE metrics, based on accuracy evaluations, customer satisfaction research, and clinical expert assessments, were used. The study showed promising results regarding the chatbot's potential as an important tool for training patients and internal organization resources. It was noted that unlocking LLM potential for chatbot development requires careful consideration of resource constraints, particularly in environments such as Google Colab. The TinyLlama project focuses on creating compact and efficient LLMs by pretraining them with 1.1 billion parameters. Attention mechanisms allow LLM to selectively focus on relevant parts of the input sequence, capturing long-term dependencies and improving context understanding. The model stack and memory were optimized by applying quantization, deactivation of reserve storage, and thoughtful tokeniser configuration. These methods enable the development of effective chatbots even with limited computational resources.

The agent helps users understand what energy optimization procedures can be created and applied to make their household appliances more eco-friendly, reduce overall energy consumption, and simplify routine tasks through smart technologies. In the work [15], the development and implementation of the GreenIFTTT (Green If This Then That) application, based on the GPT4 model, for creating and controlling home automation procedures was presented. The system focuses on creating a sequence of automation procedures in the home environment based on the sequential execution of certain actions triggered by various conditions. The main interaction paradigm is conversation, using LLM capabilities to help users find and control their smart devices. The system also integrates data from connected sensors, providing users with real-time information about their daily activities.

The work [16] proposes an approach that utilizes the unique capabilities of LLM for generating SQL code for data transformation based on prompts with domain knowledge and historical data templates. The SQLMorpher library was developed, demonstrating the effectiveness of LLM in complex tasks related to specific domains, emphasizing their potential to drive sustainable solutions in energy efficiency.

The potential use of augmented search generation technology to answer questions regarding electricity consumption measurement in buildings, considering aspects of energy digital twin based on domain knowledge was investigated in [17][18]. The authors used ChatGPT, Gemini, and Llama models to answer questions based on a knowledge graph concerning the building's electricity consumption. Their knowledge graph was created in RDF (Resource Description Framework) format, stored in a Blazegraph database, and can accept queries via the SPARQL language. The authors compared answers generated by LLM and RAG methods using the existing digital twin based on electricity knowledge. Their conclusions showed that the RAG approach not only reduces the amount of incorrect information generated by LLM but also significantly improves the quality of the result, justifying answers with verifiable data.

## 5. Conclusion

The research conducted in this paper highlights the significant potential of using large language models for human interaction with building energy consumption systems by building a digital twin of the system based on an ontological approach.

Through experimental investigation, it was explored how LLMs can help transform a human language query about energy consumption into an appropriate Cypher query to a knowledge base. By learning from a massive dataset, LLMs have the potential to generalize across various domains without requiring extensive domain-specific training. The application of a hybrid approach with

retrieval augmented generation and ontology allowed for improved query generation accuracy by performing a prior search for relevant information using semantic relationships in the constructed Neo4j knowledge base schema.

The approach using the proposed ontological model for building a knowledge base about buildings has shown that it can be an effective tool for interacting with LLMs due to its formalized structure, suitable for machine reading, and system description for representing knowledge about domain-specific features. The interactive interface makes these approaches more adaptive and accessible for people since they are not limited by predefined rules or models, which allows for broader application in the field of buildings and energy. By utilizing LLMs, dependence on specialized knowledge can be reduced, and the creation of models based on ontology promotes their widespread use by end users.

Future research may focus on applying fine-tuning approaches for compact and efficient LLM models such as TinyLlama, which will save costs and provide the possibility of using these models for embedded devices in offline mode.

## Declaration on Generative AI

During the preparation of this work, the authors used X-GPT-4 in order to: Grammar and spelling check. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

## References

- [1] O. Vyshnevskyy, L. Zhuravchak, Semantic models for buildings energy management, in: Proceedings of the 2023 IEEE 18th International Conference on Computer Science and Information Technologies (CSIT), Lviv, Ukraine, 19–21 October 2023, IEEE, 2023. doi:10.1109/csit61576.2023.10324108.
- [2] P. Jiang et al., Preventing the immense increase in the life-cycle energy and carbon footprints of LLM-powered intelligent chatbots, *Engineering* (2024). doi:10.1016/j.eng.2024.04.002.
- [3] S. Majumder et al., Exploring the capabilities and limitations of large language models in the electric energy sector, *Joule* 8 (2024) 1544–1549. doi:10.1016/j.joule.2024.05.009.
- [4] K. Forth, A. Borrmann, Semantic enrichment for BIM-based building energy performance simulations using semantic textual similarity and fine-tuning multilingual LLM, *Journal of Building Engineering* (2024) 110312. doi:10.1016/j.jobee.2024.110312.
- [5] O. B. Mulayim et al., 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 (BuildSys '24), Hangzhou, China, ACM Press, New York, NY, 2024, pp. 312–317. doi:10.1145/3671127.3698792.
- [6] C. V. S. Avila et al., Experiments with text-to-SPARQL based on ChatGPT, in: Proceedings of the 2024 IEEE 18th International Conference on Semantic Computing (ICSC), Laguna Hills, CA, USA, 5–7 February 2024, IEEE, 2024. doi:10.1109/icsc59802.2024.00050.
- [7] O. Vyshnevskyy, L. Zhuravchak, Machine learning methods to increase the energy efficiency of buildings, *Visnik Nacionalnogo universitetu "Lvivska politehnika". Ser. Informacijni sistemi ta merezi* 14 (2023) 189–209. doi:10.23939/sisn2023.14.189.
- [8] J. Lu et al., Evaluation of large language models (LLMs) on the mastery of knowledge and skills in the heating, ventilation and air conditioning (HVAC) industry, *Energy and Built Environment* (2024). doi:10.1016/j.enbenv.2024.03.010.
- [9] C. Zhang, J. Lu, Y. Zhao, Generative pre-trained transformers (GPT)-based automated data mining for building energy management: Advantages, limitations and the future, *Energy and Built Environment* (2023). doi:10.1016/j.enbenv.2023.06.005.

- [10] J. B. Ilagan, J. R. Ilagan, A prototype of a conversational virtual university support agent powered by a large language model that addresses inquiries about policies in the student handbook, *Procedia Computer Science* 239 (2024) 1124–1131. doi:10.1016/j.procs.2024.06.278.
- [11] Historical trend of graph DBMS popularity, DB-Engines – Knowledge Base of Relational and NoSQL Database Management Systems. URL: [https://db-engines.com/en/ranking\\_trend/graph+dbms](https://db-engines.com/en/ranking_trend/graph+dbms).
- [12] T. Liebig, GraphScale: Adding expressive reasoning to semantic data stores, in: T. Liebig, V. Vialard, M. Opitz, S. Metzl, *Proceedings of the 14th International Semantic Web Conference (ISWC 2015)*, vol. 1486, 2015. URL: [https://ceur-ws.org/Vol-1486/paper\\_117.pdf](https://ceur-ws.org/Vol-1486/paper_117.pdf)
- [13] M. Arslan, L. Mahdjoubi, S. Munawar, Driving sustainable energy transitions with a multi-source RAG-LLM system, *Energy and Buildings* 324 (2024) 114827. doi:10.1016/j.enbuild.2024.114827.
- [14] S. Vidivelli, M. Ramachandran, A. Dharunbalaji, Efficiency-driven custom chatbot development: Unleashing LangChain, RAG, and performance-optimized LLM fusion, *Computers, Materials & Continua* (2024) 1–10. doi:10.32604/cmc.2024.054360.
- [15] M. Giudici et al., Designing home automation routines using an LLM-based chatbot, *Designs* 8 (2024) 43. doi:10.3390/designs8030043.
- [16] A. Sharma et al., Automatic data transformation using large language model – An experimental study on building energy data, in: *Proceedings of the 2023 IEEE International Conference on Big Data (BigData)*, Sorrento, Italy, 15–18 December 2023, IEEE, 2023. doi:10.1109/bigdata59044.2023.10386931.
- [17] C. Fortuna, V. Hanžel, B. Bertalanič, Natural language interaction with a household electricity knowledge-based digital twin, in: *Proceedings of the 2024 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, Oslo, Norway, 17–20 September 2024, IEEE, 2024, pp. 8–14. doi:10.1109/smartgridcomm60555.2024.10738062.
- [18] C. Fortuna, V. Hanžel, B. Bertalanič, Towards data-driven electricity management: Multi-region harmonized data and knowledge graph, 2024. URL: <http://arxiv.org/abs/2405.18869>.