2025 European Conference on Computing in Construction

CIB W78 Conference on IT in Construction

Porto, Portugal
July 14-17, 2025

NATURAL LANGUAGE INFORMATION RETRIEVAL FROM BIM MODELS: AN LLM-BASED AGENTIC WORKFLOW APPROACH

Sylvain Hellin[1,2], Stavros Nousias[1,2], and André Borrmann[1,2]
[1]Technical University of Munich, Munich, Germany
[2]TUM Georg Nemetschek Institute, Munich, Germany

Abstract

While Building Information Models (BIM) effectively store building-related information, accessing it requires specialized software and expertise. Natural Language (NL) interfaces for BIM data retrieval can mitigate this challenge, but existing approaches are limited by rigid ontological frameworks or extensive pre-processing requirements. We present a Large Language Model-based agentic workflow that processes NL queries and automatically interacts with IFC-encoded BIM models without ontological or pre-processing constraints. In tests across architectural, structural, and MEP domains, our approach achieves 80% overall accuracy. We provide open access to IFC-Bench-v1, our evaluation dataset containing various queries, answers, and reference BIM models.

Introduction

The Architecture, Engineering, Construction, and Operations (AECO) industry is experiencing an unprecedented digital transformation, with Building Information Modeling (BIM) emerging as its cornerstone technology. BIM has fundamentally revolutionized design processes and project management throughout built asset life-cycles (Borrmann et al., 2018b), yet significant adoption barriers persist despite its demonstrated capacity to integrate multi-dimensional data and facilitate stakeholder collaboration. Research indicates that high initial investment requirements—encompassing software, hardware, and training expenditures—constitute primary adoption barriers, particularly affecting small and medium-sized firms (Chowdhury et al., 2024; Hosseini et al., 2016). Moreover, the inherent complexity of BIM systems creates persistent challenges even post-implementation, preventing the full realization of BIM's benefits globally.

Natural language interfaces for BIM can help address these challenges by providing an intuitive access to building information for stakeholders who lack specialized BIM software expertise, aligning with their established practice of direct information queries (Paredes-Valverde et al., 2016). Furthermore, such interfaces are not only helpful for humans, but can also enable the development of sophisticated NL-based automated systems that can interpret and act upon BIM data, such as:

1. Automated maintenance systems enabling natural language defect reporting while retrieving relevant BIM data for facility managers (Jeon et al., 2024)

2. Intelligent assistants supporting BIM authoring and updates through natural language interaction (Du et al., 2024)

3. Automated code compliance systems that interpret building regulations and verify compliance through targeted BIM queries (Fuchs, 2021)

While recent research has explored various approaches to natural language information retrieval from BIM models, current solutions face significant limitations. Traditional ontology-based approaches (Yin et al., 2023; Nabavi et al., 2023) struggle with scalability, and newer Language Model methods (Zheng and Fischer, 2023) require extensive pre-processing and manual schema development.

This paper presents the first method for retrieving information from BIM models encoded using the Industry Foundation Classes (Borrmann et al., 2018a) schema that: (1) does not necessitate manual data pre-processing, (2) is not bound by ontological or domain constraints, and (3) handles complex multi-hop queries. Additionally, we introduce IFC-Bench-v1, an openly accessible dataset enabling reproducible experimentation and systematic benchmarking of future approaches.

Related work

The challenge of accessing BIM information through natural language has garnered increasing attention over the past decade, producing diverse methodologies including ontology-based systems, machine learning approaches, and prompt-based solutions.

Lin et al. (2016) developed a natural language processing approach using tokenization, tagging and parsing techniques, combined with mapping to IFC entities via a MongoDB database. Their system is significantly limited in that it can only process simple sentences without verbs and pronouns, relies on a predefined keyword mapping between accepted words and IFC entities, and does not

support multi-hop queries.

Wang et al. (2021) developed a hierarchical tree-based system (BIH-Tree) for BIM that processes multi-scale queries by combining word segmentation, semantic disambiguation, and syntactic analysis with IFC schema and IFD for ontology mapping. However, the approach requires extensive data pre-processing and relies heavily on ontology/keyword mapping—specifically using IFC schema to construct the BIH-Tree and IFD for semantic disambiguation. It is further limited by rigid dependency rules that primarily support parent-child relationships, restricting its flexibility for modeling complex relationships.

Yin et al. (2023) developed an ontology-aided semantic parser that transforms natural language questions into SPARQL queries. While demonstrating the potential of ontological approaches, its scope encompasses only nineteen building element types (including IfcWall, IfcRoof, and IfcBeam) and two spatial elements (IfcSpace and IfcBuildingStorey). This constraint, along with the inability to process quantitative queries, illustrates the challenge of developing comprehensive ontologies for diverse BIM models.

Nabavi et al. (2023) proposed a machine learning approach, combining support vector machines (SVM) for query classification with natural language processing (NLP) for keyword extraction. Their implementation is limited by the requirement for predefined query types in SVM training, the reliance on pre-processed element information lists rather than direct model access, as well as relying on ontology databases (like IfcOWL), and the pre-defined set of function to retrieve information from the BIM model (only 6 in total in their experiment). This approach also does not support the multi-step reasoning necessary for complex queries.

Zheng and Fischer (2023) introduced a dynamic prompt-based methodology utilizing LLMs for BIM information retrieval. Their approach implements a sequential workflow where an LLM analyzes queries to identify types, parameters, and values, then generates appropriate database queries to retrieve information from a MongoDB database. While this approach eliminates the need for predefined ontologies, it necessitates manually converting the BIM model into a document database first, and is limited to queries whose answers can be directly retrieved without logical inference or multi-hop reasoning.

Based on our extensive literature review, we found that existing solutions suffer from at least one of these critical limitations: (1) dependency on rigid ontologies or keyword mapping systems, (2) requirement for extensive manual data pre-processing for each BIM model to be queried, or (3) inability to handle multi-hop queries. Our research bridges this gap through a novel LLM-based agentic workflow approach that operates independently of ontological frameworks, interfaces directly with IFC-encoded BIM models without pre-processing requirements, and supports both flexible query processing and sophisticated multi-stage reasoning capabilities.

Methodology

System Overview

The proposed system for answering NL queries related to a BIM model is an LLM-based agentic workflow. In this context, agentic workflows are AI-driven systems where intelligent agents execute predefined tasks and autonomously plan, adapt, and make decisions in real-time. The system takes NL queries as input and provides reality-grounded answers by accessing and interpreting BIM model data. Figure 1 presents an overview of the system architecture.

External Resources

The system relies on two primary external resources: the tools and the BIM model. The BIM model, stored in IFC format, serves as the source data for all information retrieval operations. The tools are Python functions that can directly interact with any valid IFC file. Each function enables the retrieval or computation of information from the BIM model. Table 1 lists all 29 functions implemented for this experiment, categorized by their type of operation.

Agentic workflow

The proposed agentic workflow comprises two steps:

1. A tool selection step where an LLM prompted using Chain of Thought selects the tools necessary to answer the question.

2. A tool execution step where a ReAct agent uses the selected tools (Python functions) to interact with the BIM model, retrieve and process relevant information, and answer the question.

This approach is motivated by the need to: (1) minimize the context length of the input provided to the LLM, and (2) decompose the main task into two smaller, more manageable tasks.

Levy et al. (2024) demonstrated that LLMs exhibit notable degradation in reasoning performance at input lengths far shorter than their technical maximum. Limiting the context length to include only relevant information is key to improving the system's performance. For this reason, we include only the tool names and full descriptions of the selected tools in the context window of the tool execution step. This approach avoids overloading the input with unnecessary tokens from tools that are irrelevant to answering the question.

Khot et al. (2022) demonstrated that decomposing a complex task into multiple smaller tasks improves

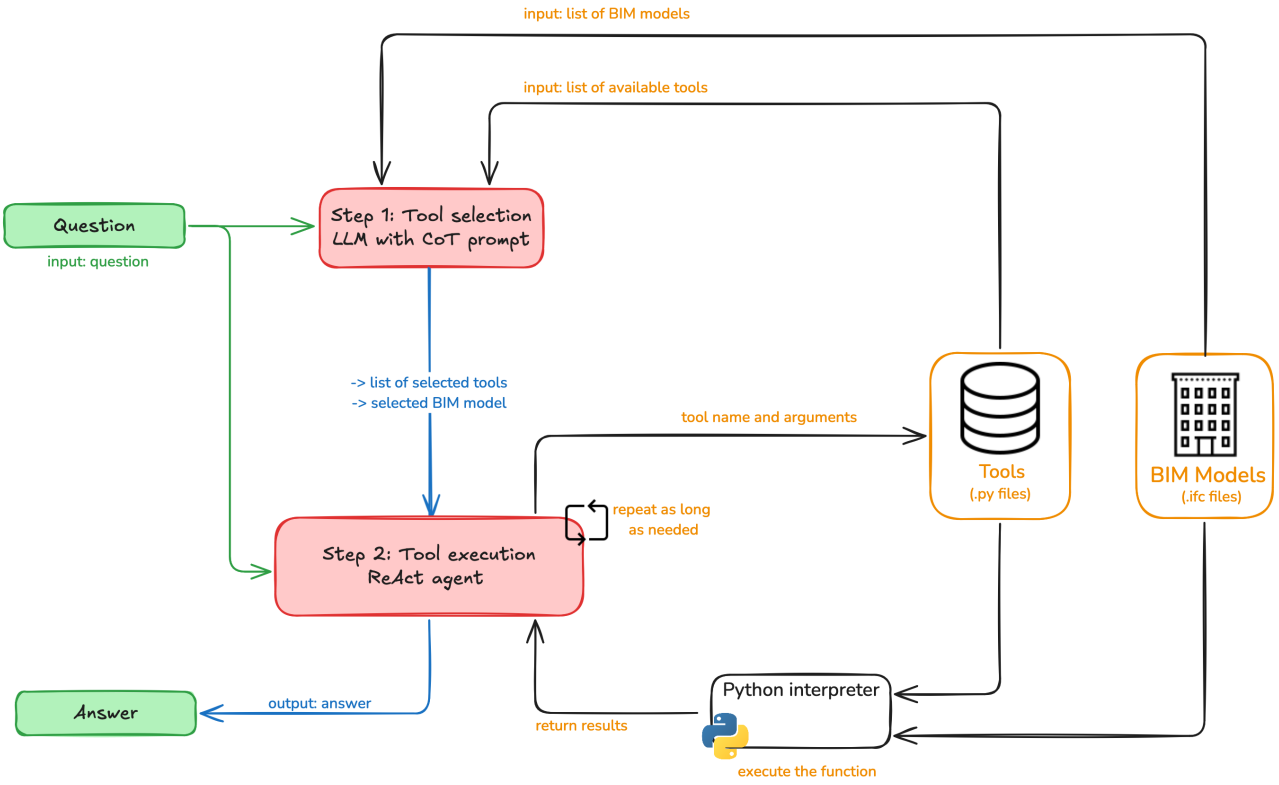

Figure 1: Overview of the system architecture

the performance of LLM-based systems compared to attempting to solve the complex task all at once.

Thus, the proposed workflow is divided into two steps (tool selection and tool execution), which simultaneously limits the context length for the challenging task and decomposes the main task into two smaller, more manageable tasks in order to maximize the system's performance.

Step 1: Tools and BIM model selection

In this first step, an LLM is prompted using the Chain-of-Thought (CoT) prompting strategy (Wei et al., 2023) to function as an intelligent query interpreter, model selector, and strategy planner. This implementation leverages the CoT technique's systematic reasoning capabilities to decompose complex queries into sequential analytical steps, thereby enhancing the accuracy of tool and model selection processes. The LLM processes three primary inputs:

1. The available IFC models encompassing architectural, mechanical, electrical, plumbing (MEP), and structural domains

2. A comprehensive catalog of tools with their corresponding functionality specifications

3. The user's natural language query

Based on its analysis of the query's requirements, the LLM identifies: (1) the specific BIM model to query, and (2) the list of relevant tools to be passed to the next step.

Step 2: Information retrieval and answer generation

In the second step of the workflow, a ReAct agent (Yao et al., 2022) is employed to retrieve information from the BIM model. A ReAct agent alternates between "reasoning," "action," and "observation" phases to achieve its objective, with the "observation" being the result of executing the requested "action". This strategy allows the agent to adapt its plan based on the outcomes of its interactions with the environment (provided as "observations").

In our implementation, actions correspond to the use of tools (Python functions) for retrieving information from the BIM model. The agent receives the user's question and detailed descriptions of the tools selected in the initial step. In the reasoning phase, the agent determines which combination of tools to use and with what arguments to formulate a response. During the action phase, the agent invokes specific functions with appropriate arguments to retrieve information from the BIM model. The output of each function is appended to the agent's conversation history, informing subsequent reasoning cycles. This enables the agent to dynamically adjust its planning based on new insights gained from the tools. The process continues until the agent has gathered sufficient information to answer the user's query.

The ReAct architecture was selected for its ability to handle queries requiring variable-length reasoning and information

retrieval chains. The number of steps required depends on both the query complexity and the characteristics of the tool implementation. This adaptive approach enables iterative information gathering and strategy refinement, increasing the likelihood of accurate query resolution.

Table 1: List of available functions categorized by operation type

Category	Functions
Information Retrieval	• find_elements_by_ifc_class • get_element_properties • get_available_models • get_model_path • list_object_types_for_ifc_entity • list_rooms • get_storeys_names • get_type_definitions_and_instances • get_state • is_georeferenced
Quantity Computation	• calculate_glazing_area • calculate_gross_floor_area • calculate_usable_floor_area • count_windows_on_facade • extract_quantity_from_property_sets • get_elements_area • get_elements_volume • get_pipe_length_by_type
Geometric Processing	• get_element_bounding_box • bounding_box_intersect • get_containing_rooms_for_entities_type • get_containing_rooms_for_entity_guids • get_containing_storey • get_door_dimensions • get_elements_in_room • get_floor_to_floor_height • get_room_ceiling_height • get_rooms_with_outdoor_access • check_door_accessibility

Python Interpreter

The Python Interpreter serves as the execution engine for the ReAct agent's actions, bridging the gap between the LLM's decision-making and the actual information retrieval from BIM models. Its primary functions are:

1. Action Execution: When the ReAct agent selects an action, the interpreter parses this command using regular expressions to extract the function name and arguments.
2. Function Matching: The interpreter matches the extracted function name against the available tool library and executes the corresponding Python function with the provided arguments.
3. Observation Generation: Results from executed functions are formatted and returned to the agent as "observations," providing the factual data needed for subsequent reasoning steps.
4. Control Flow: This cycle continues until the agent issues a "finished" action, signaling that sufficient information has been gathered to answer the user's query.

The Python Interpreter functions as the deterministic counterpart to the stochastic nature of the LLM. While LLM reasoning contains inherent variability, Python functions consistently produce identical outputs for given inputs. This determinism constrains the space of potential answers by introducing factual patterns into the token sequence, effectively grounding the LLM's responses in reality through precise information retrieval from the BIM model.

Experiment

Overview

To evaluate the effectiveness of the proposed approach quantitatively, we conducted an experiment comprising the following steps:

- Implemented the system using Python
- Developed a dataset containing 99 question/answer pairs
- Selected the best performing LLM from a selection of seven LLMs of varying sizes, types, and origins
- Evaluated system performance using the compiled dataset and selected LLMs

Each of these steps is described in more detail in the following sections.

Implementation

The proposed architecture was implemented in Python, leveraging two key libraries: DSPy for streamlining LLM interactions and IfcOpenShell for BIM model manipulation. DSPy (Khattab et al., 2023) provides a systematic approach to creating LLM pipelines. The framework's core concept emphasizes declarative syntax for specifying system inputs and outputs, eliminating the need for manually crafted prompts developed through time-consuming trial and error processes.
IfcOpenShell[1], an open-source (LGPL 3) software library developed by Thomas Krijnen, enables developers to work with Industry Foundation Classes (IFC) file formats. It provides comprehensive parsing support for various IFC

[1] https://github.com/IfcOpenShell/IfcOpenShell

schemas and offers both C++ and Python APIs for reading, writing, and manipulating Building Information Modeling (BIM) data.

Dataset

To the authors' knowledge, as of 2024, there are currently no publicly available datasets of question-answer pairs related to open BIM models. While the BINLQ dataset (Wang et al., 2022) is referenced in several scientific publications (Zheng et al., 2023), it remains inaccessible to the public. For the validation of our proposed approach, we developed **IFC-Bench-v1**, a comprehensive dataset comprising 99 domain-specific question-answer pairs for BIM information retrieval. The dataset incorporates four BIM models derived from two reference projects: the Duplex House Project and the Dental Clinic Project. To facilitate reproducibility and further research in BIM information retrieval, we have made the dataset publicly accessible through a version-controlled repository[2].

BIM Models

The source BIM models used in this study are standardized reference models provided by BuildingSMART International, with detailed provenance information included in the dataset documentation.
The Duplex Project (Figure 2) represents a small housing development featuring two attached houses (A and B), with a Gross Floor Area (GFA) of approximately 400 m². For this project, we utilized both architectural and Mechanical, Electrical and Plumbing (MEP) models.

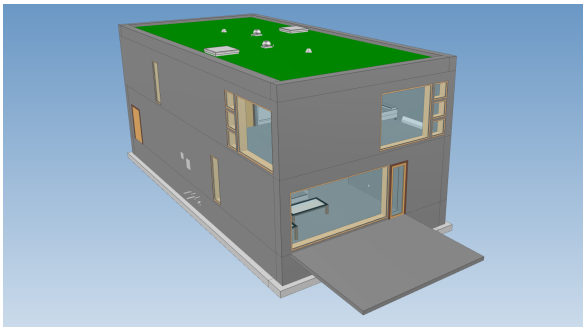

Figure 2: BIM model of the Duplex House project

The Dental Clinic Project (Figure 3), on the other hand, comprises a larger medical facility with a GFA of approximately 4,500 m², represented through architectural and structural models.

Question-Answer pairs

We categorized questions based on how readily their answers can be found in the BIM model:

- **Directly accessible information** (44% of queries): Inquiries where required data is explicitly encoded in model properties (e.g., "What is the depth of the foundation?").

[2]https://github.com/sylvainHellin/ifc-bench

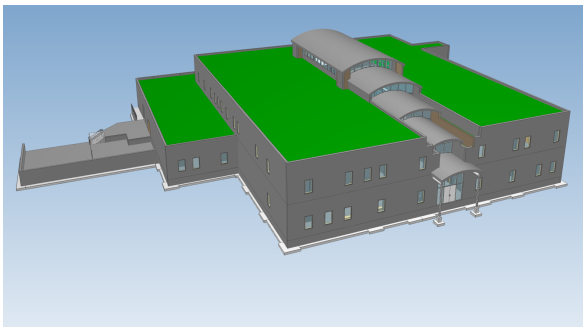

Figure 3: BIM model of the Duplex House project

- **Indirectly accessible information** (29% of queries): Inquiries necessitating computational processing or relational inference across multiple model elements (e.g., "What type of roof structure is used?").

- **Insufficient information** (26% of queries): Complex inquiries requiring integration of model data with domain expertise not encoded within the model itself (e.g., "What is the expected lifespan of the roofing material?").

The IFC-Bench-v1 dataset deliberately encompasses a broader range of query complexities compared to established benchmarks such as BINQL (Wang et al., 2022). While BINLQ primarily focuses on direct queries utilizing specific component identifiers or exact component names, our dataset incorporates more complex queries that require multi-step reasoning, indirect information extraction, and general nomenclature that may not align with BIM model conventions. This deliberate dataset design better reflects queries posed by real-world professionals, providing a more realistic assessment framework. Unlike datasets limited to directly answerable questions (comprising only 44% of our dataset), our comprehensive approach evaluates performance across the full spectrum of real-world query types. While this naturally results in lower aggregate performance metrics compared to more limited benchmarks, it offers a more honest and practical assessment of how systems would perform in actual professional contexts.

For each question-answer pair in the dataset, we specify the associated project and required BIM model for analysis. Given the complexity of building information queries, the ground truth answers are provided in natural language format to accommodate responses that go beyond simple value extraction. To illustrate this complexity, consider the query "What type of heating system is used in the bathrooms?" The corresponding ground truth answer in our dataset is: "The bathrooms are heated by wall-mounted hydronic radiators connected to a central boiler system." This example demonstrates the inherent complexity of BIM queries and illustrates our natural language approach to ground truth representation. The dataset and associated BIM models are available through our public repository, enabling quantitative benchmarking for future research. To enhance the

utility of the dataset, we encourage community contributions for its continued expansion and refinement.

Language Model

Our system architecture is designed to be model-agnostic, making it easy to swap different LLMs as needed. This flexibility allows users to select models based on specific requirements such as accuracy, response time, or cost considerations. The modular design ensures compatibility with future LLMs without requiring architectural changes or retraining. This approach offers a significant advantage over systems that depend on specialized model training or fine-tuning, making our solution more adaptable to the rapidly evolving landscape of LLM technology. Importantly, this design allows our system to automatically benefit from improvements in LLM capabilities—rather than becoming obsolete, our results would only improve as more capable models emerge.

For this experiment, we report the results obtained by using Claude-3.5-sonnet (Anthropic, 2024), as this model delivered the highest accuracy compared to the other language models we tested.

Evaluation Methodology

Given the natural language format of our ground truth answers, traditional exact-match evaluation metrics are not applicable for assessing response accuracy. We therefore adopted the "LLM-as-Judge" evaluation framework (Zheng et al., 2023) to determine the semantic equivalence between generated responses and ground truth answers.

Results

Overall Performance

Information	Count	Correct	Accuracy
Directly available	44	42	95%
Indirectly available	29	18	62%
Insufficient	26	19	73%
Total	99	79	80%

Table 2: Accuracy by information availability in the BIM model

The system achieved an overall accuracy of 80% across 99 test queries, and 95% for the queries for which the information is directly available in the BIM model. Table 2 presents the accuracy breakdown by information accessibility.

The system performed exceptionally well when information was directly accessible in the BIM model (95% accuracy), clearly demonstrating the effectiveness of our approach. Performance decreased for indirectly available information (62%), where multiple processing steps or inferences were required. Interestingly, for queries where information was not explicitly available in the model, the system still achieved 73% accuracy, indicating effective reasoning capabilities.

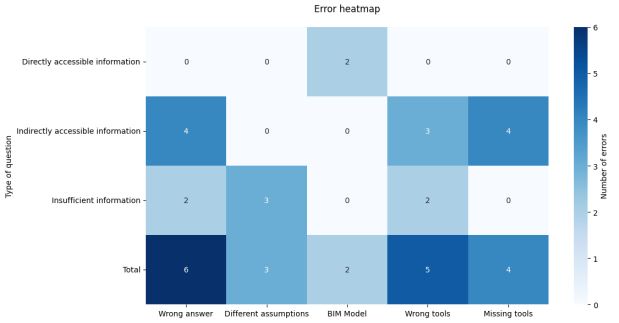

Figure 4: Error Heatmap

Error Analysis

To better understand system limitations and potential improvements, we conducted a detailed error analysis of the 20 incorrect responses. Errors were classified into 2 group: human-led errors and machine-led errors. Furthermore, we analyzed the relationship between error types and the type of question, providing deeper insight into the system's performance characteristics, as shown in Figure 4.

Human-Led Errors (55%)

Human-led errors accounted for 55% of all errors and can be attributed to limitations in the implementation rather than fundamental flaws in the LLM-based approach:

1. **BIM Model Errors (10%)**: These errors stemmed from inconsistencies or inaccuracies in the source BIM models themselves. For example, unit conversion errors from imperial to metric units in the waste pipe diameter of the Duplex House project resulted in incorrect information despite proper information retrieval.

2. **Wrong Tool Implementation (25%)**: Some tools returned incorrect information under specific circumstances due to implementation limitations. For example, our Gross Floor Area (GFA) calculation tool failed to handle certain edge cases in the duplex house project, resulting in inaccurate measurements.

3. **Missing Tools (20%)**: Several queries required functionalities that were not implemented in our current toolset, highlighting opportunities for expanding the system's capabilities. These errors occurred exclusively with indirectly available information, where specialized extraction tools would be necessary.

Machine-Led Errors (45%)

Machine-led errors, accounting for 45% of all errors, were directly related to LLM limitations:

1. **Different Assumptions (15%)**: In some cases, the model made assumptions that, while reasonable, differed from the ground truth in our dataset. This error type occurred exclusively with queries where information was not directly available in the model, requiring more interpretive reasoning.

2. **Wrong Reasoning/Tool Usage (30%)**: The most common error type involved incorrect reasoning paths or inappropriate tool selection, indicating areas where the LLM's decision-making could be improved. These errors were distributed between indirectly available (4 instances) and unavailable information (2 instances) queries.

Error Patterns by Information Accessibility

The cross-analysis of error types and information accessibility reveals clear patterns:

1. **Directly Available Information**: The only errors (2) in this category resulted from inaccuracies in the BIM models themselves, not from the system's processing logic. This explains the high 95% accuracy for these queries and suggests that with perfect BIM models, near-perfect accuracy is achievable for direct information retrieval.

2. **Indirectly Available Information**: Errors in this category were evenly distributed across wrong answers (4), missing tools (4), and wrong tool selection (3). This balance suggests that improvements in tool availability and implementation would have the greatest impact on system performance for this query type.

3. **Insufficient Information**: Errors in this category were primarily due to different assumptions (3), followed by wrong answers (2) and faulty tool implementation (2). As with the previous category, developing additional specialized tools could provide more factual grounding to prevent the model from making assumptions.

Limitations

While the overall accuracy rate of 80% and up to 95% when the information is directly stored in the BIM model demonstrates the viability of our approach, it remains insufficient for integration into complex workflows or user-facing applications. Our analysis has identified several critical factors that contribute to these performance limitations.

Prompt Optimization

The current implementation lacks systematic optimization of LLM prompts, a crucial component for enhancing system performance. Following established methodologies (Khattab et al., 2023), future work should incorporate systematic dataset partitioning into training and testing sets, enabling robust prompt engineering and optimization procedures. This should be particularly effective in reducing machine-led errors.

Tool Implementation Constraints

Proper tool implementation represents the primary bottleneck of the current system, as tool creation is currently labor-intensive and error-prone. It is impossible to implement the comprehensive toolset required for all potential queries, and as the implementation of some tools requires complex geometry processing and information handling, it is very difficult to create perfect tools that never return the wrong answer (as discussed above in the error analysis, wrong tools and missing tools are the direct cause of almost half of the errors). For example, creating a tool that could reliably assess the distance from the furthest point to the nearest emergency exit is quite a challenge. A promising solution would be extending the system to add a tool-making agent, where a new agent could design and test new tools on demand. This would both reduce human-led errors and expand the system's capabilities without requiring manual implementation of every possible tool. This approach would directly address the fundamental implementation bottleneck while significantly enhancing the system's practical versatility.

Conclusions

This research has demonstrated the viability of our LLM-based agentic workflow approach for BIM information retrieval, achieving an overall accuracy rate of 80% and up to 95% when information is directly stored in the BIM model. The system successfully processes complex natural language queries across architectural, structural, and MEP domains while addressing key challenges in BIM information accessibility.

Our primary contributions include: (1) a novel LLM-based agentic workflow for BIM information retrieval, (2) an open-source dataset enabling experimental reproduction and benchmarking, and (3) identification of critical factors affecting system performance in real-world applications. These establish a foundation for future research in natural language interfaces for BIM information retrieval.

The practical implications are significant, offering a promising solution for stakeholders who require BIM information but lack proficiency in specialized authoring tools. The system architecture provides a framework for integrating natural language processing into broader automation workflows, potentially transforming how building information is accessed across the construction industry.

Future research should address limitations through: (1) prompt optimization, (2) improvement of available tools, and (3) integration of tool generation capabilities. While the current implementation demonstrates the feasibility of natural language-based BIM information retrieval, the open-source nature of our dataset and methodology provides a foundation for collaborative advancement in this critical area of construction information management.

Acknowledgments

This research is financially supported by the Leonhard Obermeyer Center (LOC)[3] at the Technical University of

[3]https://www.ed.tum.de/loc/home/

Munich.

The authors gratefully acknowledge the valuable contributions of the IfcOpenShell and DSPy open-source communities, whose software frameworks were instrumental in the implementation of this research.

References

Anthropic (2024). Claude 3.5 sonnet. https://www.anthropic.com/news/claude-3-5-sonnet.

Borrmann, A., Beetz, J., Koch, C., Liebich, T., and Muhic, S. (2018a). Industry Foundation Classes: A Standardized Data Model for the Vendor-Neutral Exchange of Digital Building Models. In Borrmann, A., König, M., Koch, C., and Beetz, J., editors, Building Information Modeling, pages 81–126. Springer International Publishing, Cham.

Borrmann, A., König, M., Koch, C., and Beetz, J., editors (2018b). Building Information Modeling: Technology Foundations and Industry Practice. Springer International Publishing, Cham.

Chowdhury, M., Hosseini, M. R., Edwards, D. J., Martek, I., and Shuchi, S. (2024). Comprehensive analysis of BIM adoption: From narrow focus to holistic understanding. Automation in Construction, 160:105301.

Du, C., Esser, S., Nousias, S., and Borrmann, A. (2024). Text2BIM: Generating Building Models Using a Large Language Model-based Multi-Agent Framework.

Fuchs, S. (2021). Natural Language Processing for Building Code Interpretation: Systematic Literature Review Report.

Hosseini, M. R., Banihashemi, S., Chileshe, N., Namzadi, M. O., Udaeja, C., Rameezdeen, R. (2016). BIM adoption within Australian Small and Medium-sized Enterprises (SMEs): An innovation diffusion model. Construction Economics and Building, 16(3):71–86.

Jeon, K., Lee, G., Kim, Y., Kim, Y., and Lee, J. (2024). Prompt Defect Response Via Machine Reading Comprehension Using a Hybrid Large Language Model Approach.

Khattab, O., Singhvi, A., Maheshwari, P., Zhang, Z., Santhanam, K., Vardhamanan, S. (2023). DSPy: Compiling Declarative Language Model Calls into Self-Improving Pipelines.

Khot, T., Trivedi, H., Finlayson, M., Fu, Y., Richardson, K., Clark, P. (2022). Decomposed Prompting: A Modular Approach for Solving Complex Tasks.

Levy, M., Jacoby, A., and Goldberg, Y. (2024). Same Task, More Tokens: The Impact of Input Length on the Reasoning Performance of Large Language Models.

Lin, J.-R., Hu, Z.-Z., Zhang, J.-P., and Yu, F.-Q. (2016). A Natural-Language-Based Approach to Intelligent Data Retrieval and Representation for Cloud BIM. Computer-Aided Civil and Infrastructure Engineering, 31(1):18–33.

Nabavi, A., Ramaji, I., Sadeghi, N., and Anderson, A. (2023). Leveraging Natural Language Processing for Automated Information Inquiry from Building Information Models. Journal of Information Technology in Construction, 28:266–285.

Paredes-Valverde, M. A., Valencia-García, R., Rodríguez-García, M. Á., Colomo-Palacios, R., and Alor-Hernández, G. (2016). A semantic-based approach for querying linked data using natural language. Journal of Information Science, 42(6):851–862.

Wang, J., Gao, X., Zhou, X., and Xie, Q. (2021). Multiscale Information Retrieval for BIM using Hierarchical Structure Modelling and Natural Language Processing. Journal of Information Technology in Construction, 26:409–426.

Wang, N., Issa, R. R., and Anumba, C. J. (2022). Transfer learning-based query classification for intelligent building information spoken dialogue. Automation in Construction, 141:104403.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F. (2023). Chain-of-Thought Prompting Elicits Reasoning in Large Language Models.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K. (2022). ReAct: Synergizing Reasoning and Acting in Language Models.

Yin, M., Tang, L., Webster, C., Xu, S., Li, X., and Ying, H. (2023). An ontology-aided, natural language-based approach for multi-constraint BIM model querying.

Zheng, J. and Fischer, M. (2023). Dynamic prompt-based virtual assistant framework for BIM information search. Automation in Construction, 155:105067.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y. (2023). Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena.