Chair of Computational Modeling and Simulation
TUM School of Engineering and Design
Technical University of Munich

# Enhancing Project Knowledge Management in Construction: Integrating Generative AI and Large Language Models

Scientific work to obtain the degree

**Master of Science (M.Sc.)**

at the TUM School of Engineering and Design
of the Technical University of Munich.

**Supervised by** Prof. Dr.-Ing. André Borrmann
Dr.-Ing. Sebastian Esser
Chair of Computational Modeling and Simulation

**Submitted by** Vinzenz Trimborn (1122233)

e-Mail: vinzenz.trimborn@tum.de

**Submitted on** September 20, 2024

## Abstract

This thesis explores the application of Knowledge Graphs (KGs) and Large Language Models (LLMs) to enhance project knowledge management in the construction industry. The research addresses long-standing challenges in knowledge retrieval, organization, and analysis within the sector by proposing a novel framework that leverages the semantic understanding capabilities of LLMs to construct KGs from project communication data. The methodology was tested using an open-source BCF file dataset, representing standardized communication data in BIM projects.

The methodology involves a multi-step process: first, setting themes of interest for analysis; second, using LLMs to construct a knowledge graph representing both explicit and implied content from BCF comments; and finally, exploring various querying methods to gain insights from the dataset. The framework demonstrates the ability to transform unstructured project communication data into a structured, queryable format, enabling detailed post-project analysis at a scale previously unfeasible with manual work.

Results show that the framework successfully indexed BCF comments according to predefined themes, such as components, materials, and request types while discovering additional relevant categories. The natural modularity of the graph structure proved effective in identifying trends within the dataset. Agentic retrieval methods, including custom tool-based approaches and Cypher query generation, showed promise for intuitive graph querying and analysis through natural language interactions.

## Zusammenfassung

Diese Arbeit untersucht die Anwendung von Wissensgraphen (KGs) und Large Language Models (LLMs) zur Verbesserung des Projektwissensmanagements in der Bauindustrie. Die Forschung adressiert langjährige Herausforderungen bei der Wissensabfrage, -organisation und -analyse innerhalb des Sektors, indem sie ein neuartiges Framework vorschlägt, das die semantischen Verständnisfähigkeiten von LLMs nutzt, um KGs aus Projektkommunikationsdaten zu konstruieren. Die Methodik wurde anhand eines Open-Source-Datensatzes von BCF-Dateien getestet, die standardisierte Kommunikationsdaten in BIM-Projekten darstellen. Die Methodik umfasst einen mehrstufigen Prozess: zunächst die Festlegung von Analysethemen, dann die Verwendung von LLMs zur Konstruktion eines Wissensgraphen, der sowohl explizite als auch implizite Inhalte aus BCF-Kommentaren darstellt, und schließlich die Erforschung verschiedener Abfragemethoden, um Erkenntnisse aus dem Datensatz zu gewinnen. Das Framework zeigt die Fähigkeit, unstrukturierte Projektkommunikationsdaten in ein strukturiertes, abfragbares Format zu transformieren und ermöglicht damit detaillierte Nachprojektanalysen in einem Umfang, der zuvor mit manueller Arbeit nicht realisierbar war. Die Ergebnisse zeigen, dass das Framework BCF-Kommentare erfolgreich nach vordefinierten Themen wie Komponenten, Materialien und Anfragetypen indizierte und dabei zusätzliche relevante Kategorien entdeckte. Die natürliche Modularität der Graphenstruktur erwies sich als effektiv bei der Identifizierung von Trends innerhalb des Datensatzes. Agenten-basierte Abrufmethoden, einschließlich angepasster werkzeugbasierter Ansätze und CYPHER-Abfragegenerierung, zeigten vielversprechende Ansätze für intuitive Graphenabfragen und -analysen durch natürlichsprachliche Interaktionen.

# Contents

# Abbreviations

| | |
|---|---|
| **AEC** | Architecture, Engineering and Construction |
| **BCF** | BIM Collaboration Format |
| **BIM** | Building Information Modeling |
| **BOT** | Building Topology Ontology |
| **CDE** | Common Data Environment |
| **DOT** | Damage Topology Ontology |
| **ICT** | Information and Communication Technology |
| **IFC** | Industry Foundation Classes |
| **KG** | Knowledge Graph |
| **KM** | Knowledge Management |
| **LCEL** | LangChain Expression Language |
| **LLM** | Large Language Model |
| **NER** | Named Entity Recognition |
| **NLP** | Natural Language Processing |
| **RAG** | Retrieval Augmented Generation |
| **RE** | Relation Extraction |

# Chapter 1

# Introduction

## 1.1 Knowledge Management

Knowledge management (KM) is a critical factor in the success and competitiveness of organizations across various industries. It serves as a primary driving force for social and economic development, as well as a key determinant of an enterprise's competitive strength (Deng et al., 2022). As a competitive strategy, KM focuses on learning from previous experiences, avoiding mistakes, and replicating successes (Yepes & López, 2021). Consequently, knowledge has become the most valuable resource for organizations seeking to maintain their competitive edge (H. Wang et al., 2022).

While KM is widely recognized as crucial, its precise definition remains subject to debate and evolution (Deng et al., 2022). Broadly, KM can be understood as a systematic approach to creating, capturing, organizing, and utilizing an organization's knowledge assets. It encompasses a range of activities, including knowledge capture, sharing, storage, retrieval, and reuse (H. Wang et al., 2022). This multifaceted nature of KM reflects its complexity and the challenges organizations face in implementing effective KM strategies.

Implementing effective KM practices presents numerous challenges for organizations. These challenges stem from the diverse nature of knowledge itself, which exists in both explicit and tacit forms. Explicit knowledge, which is easily codified and shared, still poses difficulties in terms of information overload and efficient retrieval (Anshari & Hamdan, 2022). However, the management of tacit knowledge often presents the greatest challenge. Tacit knowledge, comprising the unwritten experiences and insights that circulate through an organization's daily activities, is particularly difficult to capture and share. Individuals or teams may be unaware of the valuable tacit knowledge they possess, making its identification and utilization a complex task (Vaz-Serra & Edwards, 2021).

## 1.2 Challenges of Knowledge Management in the Construction Industry

While Knowledge Management (KM) challenges are universal across industries, the construction sector faces additional, industry-specific hurdles that significantly complicate KM implementation and effectiveness. These challenges come from the unique nature of construction projects, which are characterized by their temporariness, uniqueness, and

the involvement of multiple stakeholders. Understanding these challenges is crucial for developing effective KM strategies in the construction industry.

**Fragmented and Temporary Nature of Projects** One major obstacle is the fragmented structure and project-centric orientation of the industry. Construction project parties come from different organizations and most of the time only work together on single projects, moving on to other projects immediately after they finish the current tasks (H. Wang et al., 2022) (Vaz-Serra & Edwards, 2021). This inherent fragmentation and short-term collaboration hinder effective knowledge continuity across different projects (Tan et al., 2007). In addition, projects usually have a rather short-term orientation with a focus on immediate deliverables. In contrast, knowledge management requires a long-term perspective as there is often a time lag between the initial investment in knowledge management systems and the return on investment (Lindner & Wald, 2011).

**Uniqueness and Complexity of Projects** Another important factor that distinguishes the requirements of the construction industry is that each construction project is unique concerning its constituent elements, requiring particular forms of professional inputs and management (Vaz-Serra & Edwards, 2021). The various procedures involved in context with the large project volume and the long construction period, make it complex to manage knowledge (Deng et al., 2022). This uniqueness often means that knowledge gained in one project may not be directly applicable to another, complicating efforts to standardize KM practices across the industry.

**High Staff Turnover and Team Dissolution** Knowledge derived from construction projects is often lost, due to the short time between capturing the knowledge, the turnover of technical staff, and the dismantling of teams engaged for a specific project at the project's end (Vaz-Serra & Edwards, 2021). This issue is exacerbated by the numerous highly specialized roles in the construction industry, with many actors joining projects only for individual tasks or short durations. Consequently, strategies and techniques for knowledge capture, like for example post-project review to capture lessons learned can't be conducted effectively because the relevant staff already would have moved to the next project (H. Wang et al., 2022). This constant change of personnel makes it difficult to maintain organizational memory and ensure that valuable insights and experiences are retained and shared across projects and teams.

**Data and Information Challenges** Moreover, the construction industry generates vast amounts of data and information throughout project lifecycles. However, managing this data effectively for knowledge creation and sharing presents unique challenges. The unstructured data collected in the construction industry is more difficult to recognise and process by computers, which increases the difficulty of sharing knowledge among different projects and enterprises (Deng et al., 2022). Consequently, storing and retrieving the massive and complex knowledge generated during construction projects is challenging.

The difficulty in managing the knowledge generated, imposed through the project conditions mentioned above leads to a high likelihood of repetitive errors and a constant duplication of existing solutions (Vaz-Serra & Edwards, 2021) (Tan et al., 2007). This issue is amplified when considering that most of the knowledge of the industry is generated during the progress of delivering a custom-built facility in accordance with the client's requirements (Tan et al., 2007). Therefore, the problem is not the generation of knowledge in the construction sector, but the waste of valuable information gained within projects (Yepes & López, 2021).

Given these challenges and the wealth of untapped knowledge, it becomes crucial for the construction industry to find effective ways to reuse and analyze existing information. The key lies not just in capturing knowledge, but in developing systems and processes that can efficiently extract insights from the vast amount of data generated across projects. By doing so, construction firms can potentially avoid repeating mistakes, capitalize on past successes, and drive continuous improvement in their operations.
One promising avenue for addressing these knowledge management challenges is the emerging field of Artificial Intelligence, particularly Large Language Model (LLM)s. These novel AI systems have shown remarkable capabilities in processing and understanding large volumes of unstructured data, including the type of complex and diverse information generated in construction projects.

## 1.3 Research Question

The emergence of LLMs as powerful tools for processing and understanding vast amounts of unstructured data has paved the way for innovative approaches in various fields. In parallel, Knowledge Graph (KG)s offer a structured methodology for representing and connecting information. This research focuses on harnessing the synergy between LLMs' semantic understanding and KGs' structured representation within a framework to analyze existing data produced during construction projects. This study will be based on construction data, with a particular emphasis on the unstructured text within BIM Collaboration Format (BCF)s. The goal is to explore how LLMs can leverage their semantic understanding capabilities to build KGs from a data format that combines structured information and unstructured text comments. By creating a comprehensive and enriched KG, I aim to provide a robust basis for project analysis, which can improve decision-making processes and enhance the efficiency of future construction projects. Furthermore, the research will investigate how this enriched KG can be utilized for information retrieval. Different retrieval methods, including Cypher queries facilitated by LLMs, will be utilized to ensure comprehensive and intuitive access to the information. Furthermore, I will propose and test different approaches using agentic systems for reasoning over the graph structure.

The primary research question guiding this thesis is: How can LLMs be used to build KGs from semi-structured construction data, and which KG retrieval methods can be leveraged, to enhance project analysis in construction projects?

# Chapter 2

# Background and Related Work

The construction industry, despite its wealth of knowledge generation, faces significant challenges in effectively managing and utilizing this knowledge due to its project-centric nature, high staff turnover, and complex data structures. LLMs have emerged as powerful tools for processing and understanding vast amounts of unstructured data, while KGs offer a structured approach to representing and connecting information. There is considerable potential in applying a framework that combines the strengths of both LLMs and KGs tailored to construction industry data, to address these unique challenges.

To explore this potential, this work will first examine promising approaches leveraging Information and Communication Technology (ICT) for knowledge management in construction. Following this, an overview of LLMs and associated techniques are given, including prompt engineering and LLM based agents. Next, a comprehensive review of LLMs in Ontology and KG creation is presented, considering their potential adaptations for the construction sector. The last section is dedicated to LLM based query-focused summarization. This literature review will lead to the proposal of a methodology, aiming to bridge the gap between current practices and the potential of integrated LLM and KG solutions in construction knowledge management.

## 2.1 Leveraging ICT for Knowledge Management in Construction: Current Approaches and Emerging Trends

ICT systems play a crucial role in supporting and facilitating the activities of Knowledge Management (KM). They enable the agile collection, storage, and exchange of knowledge, which is essential for effective KM practices in the construction industry. One of the primary advantages of ICT in KM is its ability to facilitate the codification of knowledge, allowing it to be stored in databases. This capability makes the scaled and agile collection, storage, and exchange of information possible (Yepes & López, 2021).

### 2.1.1 Web Applications as KM Systems

Web applications exemplify this advantage by providing platforms where users can input, store, and share knowledge efficiently. Therefore, various researchers have proposed specific ICT solutions to address KM challenges in construction. Tan et al. (2007) introduced a web-based knowledge base with an integrated workflow system, designed to capture knowledge in time during projects while minimizing overhead costs. The system

was applied in learning situations, including weekly site meetings and project reviews. It provided participants with a structured form to capture their knowledge, which could then be searched. A similar web-based approach is introduced by Vaz-Serra and Edwards (2021). Therefore, ICT e.g. a web application, can be viewed as both a tool and a means to realize knowledge management. This is particularly beneficial as it lowers the threshold for implementing KM strategies since practitioners don't necessarily need to understand the intricacies of KM theory but can instead focus on learning to use relevant ICT software (Deng et al., 2022).

### 2.1.2 IFC and BIM as KM Systems

Building upon these specialized ICT applications, the construction industry has further developed and adopted tools tailored to its specific processes and data structures. In this landscape of construction-specific ICT, certain developments stand out in facilitating KM activities. Product data modelling initiatives such as Building Information Modeling (BIM) and Industry Foundation Classes (IFC) have significantly promoted KM activities (Yu & Yang, 2018).

BIM not only serves as a data model but also as a dynamic methodology that integrates data-driven use cases throughout the project's lifecycle (Borrmann et al., 2021). In this sense BIM encompasses a range of activities from project initiation through planning, feasibility, design, and construction to completion, handover, and ongoing management, including maintenance and eventual demolition (Razali et al., 2019) (Borrmann et al., 2021). Therefore, BIM in general, in combination with the open IFC format, which was recognized as an ISO standard in 2013, caused a paradigm shift in the industry about collaboration, data exchange and the software landscape, enabling the development of new KM methods.

BCF, a data format augmenting BIM which enables workflow communication in BIM processes (buildingSMART Technical, 2023), enhances KM in terms of traceability and collaboration. In BIM-based collaboration projects, BCFs act as a specialized communication tool, facilitating issue tracking and resolution, including clash detection, similar to ticketing systems used in project management (Borrmann et al., 2018).

BIM, in particular as a data model, with its object-oriented nature and parameter-driven approach, contributes to the improvement of knowledge capture (H. Wang et al., 2022). Beyond its traditional uses, BIM is also further expanded for KM. Exemplary for such methods, H. Wang et al. (2022) proposed a framework that uses BIM as the main tool for capturing knowledge, storing information as parameters within the BIM model and later transferring it to a central database for retrieval using Natural Language Processing (NLP) and case-based reasoning methods.

### 2.1.3 Semantic Technologies for KM

The advent of the semantic web, underpinned by ontologies, has widely boosted the development of IT solutions for KM in construction (Yu & Yang, 2018). This progress was further expanded in 2012 when Google proposed the concept of knowledge graphs, which combine ontology and semantic network concepts (Singhal, 2012). These advanced technologies have enabled more sophisticated approaches to KM in construction, since they enabled the combination of deep learning with the fusion of multi-source heterogeneous data, helping to address the miscellaneous information from various project activities (Deng et al., 2022).

Ontologies have significantly advanced KM in the construction industry by providing structured frameworks for representing and connecting diverse information. As demonstrated by Schulz et al. (2023), ontologies like bcfOWL, the Damage Topology Ontology (DOT) (Hamdan et al., 2019), and the Building Topology Ontology (BOT) (Rasmussen et al., 2021) serve as crucial bridges between different aspects of construction data. For instance, bcfOWL enables effective communication between BCF Issues and Linked Building Data concepts, where DOT facilitates the representation of damage to constructions, and BOT describes the core components of a building. These ontologies enhance the ability to capture, retrieve, and share knowledge efficiently by improving querying capabilities and connectivity within the Linked Building Data domain.

Building on these concepts, researchers have developed frameworks that leverage ontologies and semantic technologies for specific construction KM challenges. For instance, Kim and Chi (2019) proposed an approach for managing knowledge contained in construction accident case documents using NLP enhanced by an ontology. This ontology represented unique expressions used in accident cases and common terms in the general construction industry, improving the system's ability to retrieve and analyze relevant cases. Similarly, Zou et al. (2017) developed a framework to retrieve similar risk cases from construction accident databases using NLP and ontology-based techniques. Al Qady and Kandil (2010) proposed an NLP system to manage knowledge in contract documents, facilitating quick access and efficient use of such knowledge for project management and contract administration tasks. These approaches demonstrate how advanced semantic technologies can be applied to extract and manage domain-specific knowledge in construction.

The knowledge graph is considered one of the most advanced knowledge management technologies in the Architecture, Engineering and Construction (AEC) sector (Deng et al., 2022) since it offers promising solutions for managing the complex and diverse knowledge generated in construction projects.

## 2.2 Large Language Models

The field of NLP has been dramatically transformed in recent years by the rapid advancements in LLMs. At their core, language models are designed to estimate the probability

distribution over text. However, recent scaling improvements have pushed these models to new heights of capability and versatility (Kojima et al., 2022).

LLMs typically employ a pre-training approach, where expansive foundational models with hundreds of billions of parameters are initially trained using massive datasets. These models can be subsequently fine-tuned or adapted during inference, offering considerable advantages over conventional NLP models that often require extensive supervised learning on smaller, task-specific datasets (Manning, 2022).

The emergence of these large pre-trained models, particularly those scaled to over 100 billion parameters, has initiated a paradigm shift in the field. These models exhibit properties conducive to few-shot learning, enabling a technique known as in-context learning (Brown et al., 2020). This approach utilizes text or templates, referred to as prompts, to guide the model in generating outputs for desired tasks (Kojima et al., 2022).

This transition marks the beginning of the "pre-train and prompt" era, where pre-trained models are adapted to new tasks through strategic prompting rather than extensive retraining (Kojima et al., 2022). As a result, LLMs have demonstrated remarkable capabilities across a wide spectrum of NLP tasks and real-world applications, ranging from language understanding to text generation. Their potential to address various NLP challenges and practical use cases continues to expand (Yang et al., 2024).

In the following subsection 2.2.1, I will shortly introduce Prompt Engineering and Prompt Chaining. This will be the basis for more advanced concepts like LLM based Agents (section 2.2.2) and Retrieval Augmented Generation (RAG) (section 2.4.1).

### 2.2.1 Prompt Engineering and Chaining

Prompt engineering is a rapidly evolving discipline that focuses on crafting effective instructions to guide LLMs in performing various tasks without extensive retraining (P. Liu et al., 2023). It involves designing prompts that leverage the models' pre-trained knowledge to generate desired outputs (Amatriain, 2024). Techniques range from standard prompting, where users create customized instructions, to more advanced methods like few-shot prompting, which enable LLMs to tackle new tasks with minimal examples (Brown et al., 2020) (Radford et al., 2019). In the literature, prompts that explicitly condition on a few task examples are called few-shot prompts. This method is similar to teaching by example, while template-only prompts without any examples or exposure to similar tasks are termed zero-shot prompts (Kojima et al., 2022).

Prompt chaining, also known as LLM chaining, is an advanced technique that breaks complex tasks into a series of interconnected prompts or LLM calls (Wu et al., 2022). This approach, exemplified by methods like Chain-of-Thought and Tree-of-Thoughts, allows LLMs to tackle intricate problems by decomposing them into manageable steps (Wei et al., 2022) (Yao et al., 2024). By guiding the model through a sequence of reasoning stages, prompt chaining enhances the LLM's ability to handle complex reasoning tasks, improve transparency, and provide more reliable and controllable outputs (Kojima et al., 2022).

### 2.2.2 LLM-Based Agents

LLM based autonomous agents combine the powerful language understanding and generation capabilities of LLMs with the ability to interact with and manipulate their environment. LLM agents are characterized by their ability to plan, reason, and take actions over multiple iterations to execute goals (Masterman et al., 2024).

According to L. Wang et al. (2024) in their comprehensive survey on LLM based autonomous agents, LLM agents typically consist of four main components: Profiling, Memory, Planning, and Action. These modules work together to create an autonomous system capable of understanding its role, remembering past experiences, planning future actions, and executing tasks effectively.

**Profiling Module** The profiling module is responsible for defining the agent's identity and characteristics. It aims to establish the agent's role, personality, and background, which are then used to influence the LLM's behaviour. This module helps tailor the agent's responses and actions to align with its designated persona, making interactions more consistent and contextually appropriate (Qian et al., 2023) (Chen et al., 2023).

**Memory Module** The memory module allows the agent to store and retrieve information from its interactions with the environment. This component is crucial for enabling the agent to maintain consistency in its behaviour and make informed decisions based on accumulated knowledge. The memory module typically includes mechanisms for both short-term and long-term information storage and retrieval (Park et al., 2023) (Z. Huang et al., 2023).

**Planning Module** The planning module empowers the agent to approach complex tasks by breaking them down into manageable steps. It enables the agent to formulate strategies, consider potential outcomes, and adapt its approach based on feedback or changing circumstances. This module is key to the agent's ability to handle multi-step problems and exhibit goal-directed behaviour (Yao et al., 2022) (W. Huang et al., 2022).

**Action Module** The action module is responsible for translating the agent's decisions into specific outcomes or behaviours. It serves as the interface between the agent's internal processes and the external environment. This module determines how the agent interacts with its surroundings, whether through generating text responses, utilizing internal knowledge, or leveraging external tools. Tools are external resources that expand the agent's capabilities, including APIs for specific functions, databases, or knowledge bases for information retrieval. For example, the agent might use a calculator API for mathematical operations or query a database for specific information. By integrating both internal capabilities and these external tools, the action module enables the agent to perform a wide range of tasks and adapt to various environments (Qian et al., 2023)

(Nakano et al., 2021). As outlined, according to the authors L. Wang et al. (2024), these four modules work in combination to create a comprehensive LLM based agent architecture.

## 2.3 LLMs in Ontology and Knowledge Graph Creation: A Comprehensive Review

The following chapter reviews recent applications of LLMs within the Ontology and Knowledge Graph creation process, detailing different protocols and methods of how this process can be (semi-) automated.

### 2.3.1 NER and RE Approaches with Provided Ontology

An essential component in scientific NLP is Named Entity Recognition (NER), where entity labels such as “material" or “property", which are usually predefined, are applied to words from the text. NER identifies text spans that mention relevant items or concepts like for example ingredients in a recipe text (Caufield et al., 2024) (Dagdelen et al., 2024). These tagged sequences can be mapped to persistent identifiers in ontologies within a process called grounding (Caufield et al., 2024).

Another key component is the development of Relation Extraction (RE) techniques to accurately extract the relationships between named entities. RE models have been developed and trained to determine which entities are linked by a predefined set of relations (Deng et al., 2022). For example, in the sentence "Reinforced concrete is used in building foundations", the material entity "reinforced concrete" is linked to the application entity "building foundations". Through RE named entities are connected with predicates, such as ’used in’, forming simple triple statements (Caufield et al., 2024).

Deep Learning methods such as LLMs (Vaswani et al., 2017) have made gains in all these applications. Whereas the former generation of methods relied heavily on task-specific training data, LLMs such as GPT-3 and GPT-4 can generalize on these tasks by reframing them as prompt-completion tasks (Brown et al., 2020) (Achiam et al., 2023).

In various recent approaches, an ontology is provided and LLMs are used to transform inputted unstructured text to conform to this schema. These LLM based approaches allow to fill out schemas bypassing a need for training examples. Early experiments levering LLMs started by custom designing prompts for entity-relationship diagrams creation. Approaches conducted by Fill et al. (2023) start these prompts with an explanation of Entity Relationship diagrams, including an example in JSON, following a natural language description of the task. The LLM is therefore prompted to extract the JSON from unstructured text. Although the results are satisfying for the authors, they did not find any benchmarks to evaluate their results. Furthermore, only small diagrams are extracted from a relatively small quantity of text.

Caufield et al. (2024) propose SPIRES, a method for populating a knowledge base using the zero-shot learning capabilities of LLMs. Prompts are recursively derived from an ontology for a schema-driven prompting approach. LLMs are used to recursively interrogate the heterogeneous information according to the hierarchy levels of the ontology to obtain a set of responses matching the provided schema. Their results for NER while leveraging GPT-4 show a precision of 0.85 with a recall of 0.65 concluding with an F-score of 73.69, showing promising results without the need for any specific training data.

Dagdelen et al. (2024) investigate an approach where a LLM is fine-tuned to simultaneously extract named entities and their relationships. Instead of providing an ontology, one has to define the desired output structure, for example, a list of JSON objects with a predefined set of keys—and annotate 100–500 text passages using this format. The LLM is then fine-tuned on these examples, and the resulting model can accurately output extracted information in the same structured representation. In comparison to BERT-based models which require fine-tuning on a large corpus of domain-specific data (e.g., millions of article abstracts or paragraphs), the comprehensive pretraining of the LLMs along with the user-provided annotations are sufficient to accomplish a broad array of complex tasks.

In all the approaches mentioned above some sort of ontology or schema is used as an input. To automate the process of ontology creation, a lot of NLP methods can be leveraged in the area of ontology learning.

### 2.3.2 Ontology Learning Approaches

Ontology Learning is a major field of research in AI, NLP, and knowledge engineering (Babaei Giglou et al., 2023). Its goal is to provide a cost-effective and scalable solution for knowledge acquisition and representation, enabling more efficient and effective decision-making in a range of domains (Babaei Giglou et al., 2023).

Ontology Learning is based on automatically discovering and extracting knowledge structures from textual information to construct or extend an ontology. This process typically employs techniques such as automated type discovery, type taxonomy recognition and non-taxonomic relations discovery between types. Moreover, it includes methods to infer potential axioms and other ontological structures (Konys, 2019). Earlier approaches for Ontology Learning include linguistics-based approaches and statistic-based, as well as logic-based machine learning approaches (Al-Aswadi et al., 2020).

Since LLMs are pre-trained on vast amounts of text data, which gives them a broad understanding of language and entities, they can also be used to define categories for the NER and RE tasks. Therefore, they can automate the task of researchers or domain experts defining a set of categories based on the needs of the application or the domain of interest. Different approaches have been explored in literature. Mateiu and Groza (2023) enriches ontologies by translating natural language sentences into OWL Function Syntax (axioms), by fine-tuning a GPT-3 model accordingly.

Bikeyev (2023) proposes an approach where ontologies are synthetically generated, introducing a method which uses two prompts to generate a hierarchy of elements and another to determine possible relationships between them. Thereafter, an iterative approach is conducted to add more detail to the ontology. Therefore, the knowledge model becomes purely machine-generated.

Babaei Giglou et al. (2023) test and propose LLMs for Ontology Learning (LLMs4OL). Three main questions are evaluated: How effective are LLMs for automated type discovery, to recognize a type taxonomy and to discover non-taxonomic relations between types? A zero-shot prompting method is applied with a variety of different LLMs including GPT-4. Babaei Giglou et al. (2023), however, concluded that the models alone are not yet sufficiently suitable for ontology construction since the process entails a high degree of reasoning skills and domain expertise.

### 2.3.3 LLM Supported Frameworks for the Ontology Engineering Process

Instead of leveraging LLMs to fully automate the ontology engineering process, there have also been approaches to facilitate the ontology engineering process, which is often a laborious collaborative task.

B. Zhang et al. (2024) proposes a framework that focuses on requirement elicitation, analysis and ontology testing. Through a chat interface, domain experts are supported in creating the user story, which is used to extract the Competency Questions in a collaborative approach between an Ontology Engineer and a LLM Agent. After Ontology creation, the LLM Agent also provides testing support in a SPARQL-free approach, through ontology verbalisation and prompt driven Competency Questions unit testing, against the verbalissed ontology.

## 2.4 LLM-Based Query-Focused Summarization

Given the challenges in knowledge management within the construction industry outlined in section 1.2, there is a need for methods to effectively process and derive meaning from large document collections. This section provides a short review of recent LLM-based systems designed to read and reason about extensive document collections. In particular, systems that can reach conclusions that extend beyond the explicit content of the source texts.

Such approaches are valuable for supporting human-led sensemaking across entire text corpora. They empower individuals to both apply and refine their mental models of data by facilitating the asking and answering of questions with a global scope (Klein et al., 2006) (Edge et al., 2024). This capability is crucial in the context of construction knowledge management, where professionals must synthesize information from diverse project documents to make informed decisions and avoid repeating past mistakes, as outlined before.

In this context, Query-focused Summarization emerges as a more appropriate task framing (Dang, 2006). While techniques like NER and RE, discussed earlier in section 2.3.1, focus on identifying specific elements within text, Query-focused Summarization aims to synthesize information across larger text collections. Specifically, query-focused abstractive summarization generates coherent natural language summaries that capture the essence of the content, rather than simply selecting and combining existing sentences or passages from the original text (Dang, 2006) (Edge et al., 2024).

Recent advancements in LLMs have significantly simplified these tasks. These models utilize in-context learning to summarize any content provided within their context window, directly processing source text to create summaries tailored to specific information needs (Edge et al., 2024), without requiring pre-identified entities or relationships. However, challenges persist in query-focused abstractive summarization over entire data sets, as such volumes of text exceed LLM context window limits (Edge et al., 2024).

The following subsections will introduce RAG, before focusing on Graph-RAG, an approach for Query-Focused Summarization recently published by Microsoft Research Edge et al. (2024), which forms the basis for the methodology presented in this thesis.

### 2.4.1 Retrieval Augmented Generation

Retrieval Augmented Generation (RAG) has emerged as an established approach to answering user questions over entire datasets (Lewis et al., 2020). It enables LLMs to answer questions about private unseen document collections by retrieving relevant information from an external knowledge source.

It addresses key limitations of traditional LLMs, including hallucination (Y. Zhang et al., 2023), outdated information (He et al., 2022), and lack of transparency in reasoning (Gao et al., 2023). RAG framework consists of three primary components: retrieval, which involves fetching relevant information from external databases; generation, where the LLM utilizes both its inherent knowledge and the retrieved information to produce responses; and augmentation, which refers to the process of integrating the retrieved information into the LLM's workflow (Gao et al., 2023). This approach allows for more accurate, up-to-date, and verifiable responses, particularly in knowledge-intensive tasks, while also enabling continuous knowledge updates and integration of domain-specific information (Gao et al., 2023).

As described in detail by Gao et al. (2023), the retrieval process in RAG, heavily relies on vector stores and semantic similarity calculations. During the indexing phase, documents are split into smaller chunks, which are then encoded into vector representations using an embedding model. These vector representations are stored in a vector database, creating an efficient searchable index. When a user submits a query, it is similarly transformed into a vector representation using the same embedding model. The system then computes similarity scores between the query vector and the vectors of the indexed chunks, typically using metrics like cosine similarity. This allows the retrieval system to identify and retrieve

the most semantically relevant chunks of information from the vector store (Gao et al., 2023).

However, RAG is designed for situations where answers are contained locally within retrievable text regions that provide sufficient grounding for the generation task. Despite its usefulness, RAG falls short when dealing with global questions directed at an entire text corpus, such as "What are the main themes in the dataset?" This is because such queries inherently constitute a Query-focused Summarization task rather than an explicit retrieval task (Edge et al., 2024).

### 2.4.2 Graph-RAG for Query-Focused Summarization

Edge et al. (2024) propose a framework called "Graph RAG", tailored to the Query-focused Summarization task for large data sets of text described above. The core concept leverages the power of LLMs to create a self-generated entity knowledge graph index of the source documents.

After splitting the source documents into text chunks, a LLM is used to extract entities and relationships from these chunks and summarise these instances into concise element descriptions. Within the resulting constructed graph, community structures are leveraged to generate summaries for each community at various hierarchical levels. These summaries are then used in return to answer global queries via a map-reduce approach.

The overall approach leverages the modularity of the graph, which refers to the tendency of graphs to form communities or clusters of nodes that are more densely connected than nodes in other parts of the graph. Therefore, in contrast with other approaches, the focus is not on using the graph for structured retrieval and traversal, leveraging the graph structure to efficiently search for and navigate between pieces of information but to leverage the graph for information structuring in terms of its modularity (Edge et al., 2024).

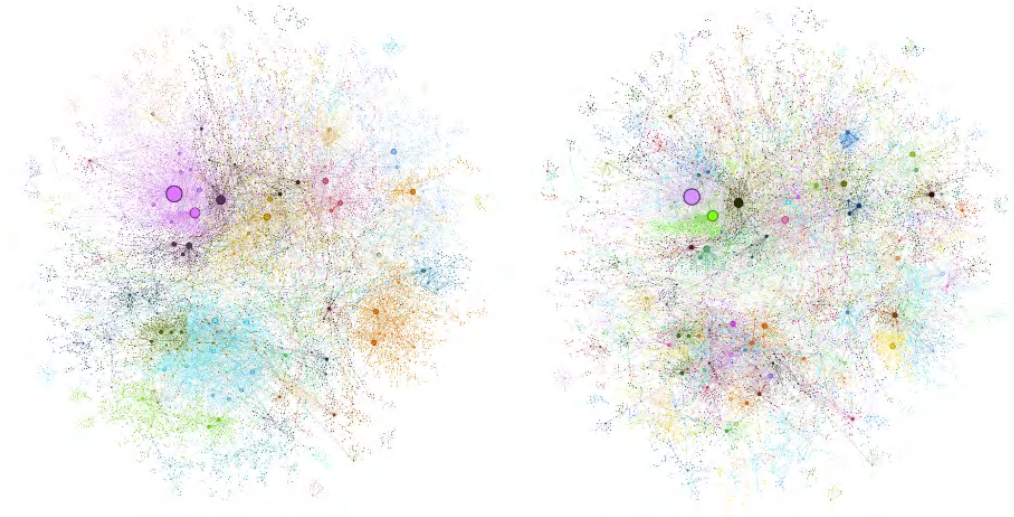

Figure 2.1: Graph communities detected on different levels, showcasing the modularity of a graph. Figure taken from (Edge et al., 2024).

### 2.4.3 Summary and Identified Research Gap

The literature review presented in this chapter highlights the developments in knowledge management within the construction industry, particularly through the application of ICT, semantic technologies and NLP. Traditional approaches have evolved from web-based knowledge management systems to more sophisticated BIM and IFC-based solutions. The emergence of semantic technologies, including ontologies and KGs, has further enhanced the industry's ability to capture, organize, and retrieve complex project knowledge.

While the construction industry has made advancements in implementing various NLP techniques, the application of novel technologies like LLMs remains largely unexplored in this domain. The review highlights a clear gap between the potential capabilities of LLMs, as demonstrated in other fields, and their current application in construction knowledge management.

Recent advancements in LLMs, such as their ability to perform complex reasoning tasks, suggest significant potential for addressing long-standing challenges in construction knowledge management. These challenges include processing unstructured data, inferring implicit knowledge, and providing context-aware information retrieval. However, the literature review reveals a lack of research specifically applying LLMs to construction project analysis and knowledge management.

The research question posed in Section 1.3 seeks to address this gap by investigating how LLMs can be used to build KGs from semi-structured construction data, and which KG retrieval methods can be leveraged to enhance project analysis. Therefore, this research aims to explore the potential of combining the advanced natural language processing capabilities of LLMs with the structured representation offered by KGs. By applying this novel combination to semi-structured construction data this work seeks to develop a novel approach to capture, organize, and retrieve complex project knowledge.

# Chapter 3

# Methods

The proposed methodology was designed to semi-automate analysis over a dataset of communication data from a construction project. First, a method is introduced to specify themes of interest relevant to the data analysis. After that, I used the reasoning capabilities of LLMs to construct a KG representing the explicit and implied content of the communication data under the angle of different themes. Subsequently, by leveraging the graph's modularity, I explored common trends and themes in the data. Finally, I investigated different querying methods to gain insights from the dataset.

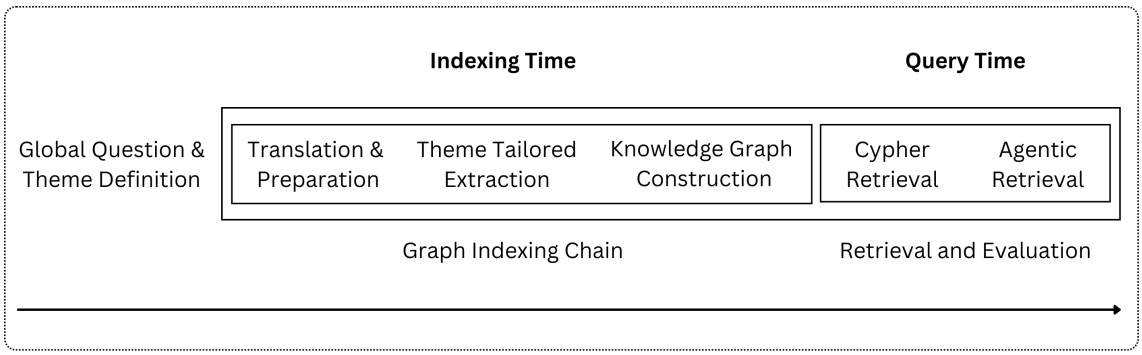


Figure 3.1: Methology Overview.

As displayed in Figure 3.1, the workflow involves several key steps:

1. **Setting Themes:** To guide the analysis, a set of global themes was developed, corresponding to key questions to be answered about the dataset. These themes emerged through the analysis of a subset of the data to identify recurring topics and through consideration of broader research objectives. The themes effectively framed the inquiries to be pursued across the entire dataset. Additionally, during the automated analysis process, the LLM explored and uncovered further themes of interest, expanding the range of questions that could be addressed.
2. **Graph Indexing Chain** To construct a KG, I applied a chain of prompts to every text in the dataset. This involved the following steps:
    a) Translation and preparation of each text, in a more descriptive format.
    b) Extraction of explicit and implied information based on the content of each text under the angle of different themes.
    c) Construction of a KG containing all derived information and reasoning of the previous steps.
3. **Retrieval and Evaluation:** The KG was used as a tool for evaluation and analysis. Unlike probabilistic methods, the graph allows for precise counting and quantitative

analysis of relationships, occurrences, and patterns within the data. This enabled me to make definitive statements about the frequency and distribution of various elements within the dataset. I leverage and explore two approaches which build up on each other:

a) Deterministic retrieval or querying of information from the graph using Cypher.

b) Natural language communication with the graph leveraging agentic frameworks.

This methodology provided a framework for storing, representing, and quantitatively analyzing derived knowledge about the unstructured text in the dataset. It allowed thematic aspects to be captured while also facilitating a broader, deterministic analysis of the entire dataset.

In the first section 3.1 of this chapter I outline the data source for my methodology and describe its required pre-processing steps. The next section 3.2 focuses on the graph indexing, detailing the usage of LLMs for KG construction, the definition of my graph schema and a detailed description of my prompt chain used to construct the KG. In section 3.3 I propose different methods I used for the retrieval and analysis of information from the KG, including agentic approaches.

## 3.1 Data Source and Data Preprocessing

Building upon the challenges and opportunities identified in the previous chapters, this study leveraged data sources that are inherently created during the planning phase of construction projects. This approach addressed one of the key issues in construction knowledge management: the need for efficient data collection without additional overhead (as discussed in Chapter 2). By utilizing existing data, I ensured that the method could be implemented without disrupting current workflows or requiring extra effort from project teams.

Construction projects generate a diverse array of information artefacts throughout their lifecycle. These include 2D plans, 3D models, timetables, cost estimates, and various forms of communication data such as meeting protocols, email correspondence, and chat logs. While technical documents like plans and models represent the final results of design and planning processes, they often fail to capture the rationale behind decisions, the evolution of ideas, and the challenges encountered during the project. In contrast, communication data offers unique insights into the decision-making processes and the chronology of changes that occur throughout a project's lifecycle.

The core idea of this work was to extract and structure the implicit knowledge created in construction projects. As a starting point, this thesis focused specifically on communication data, particularly BCF files. The BCF file format is used as it is an open-source format and can be easily parsed.

Communication data and particularly BCF files are beneficial for several reasons:

1. **Rich Contextual Information:** Communication data often contains discussions about problems, solutions, and decision rationales that are not evident in final technical documents. Therefore, they lay the basis for the analysis of decisions through the project.

2. **Change History:** Unlike static plans or models, communication records can reveal how and why certain decisions were made, providing valuable insights for improving future planning processes.

3. **Industry Adoption:** The global construction industry is increasingly adopting BIM, driven by government mandates (Borrmann et al., 2018). As BIM adoption grows, the utilization of standardized data exchange and collaboration formats spearheaded by IFC and BCF is likely to rise.

4. **Standardization:** BCFs, in particular, offer a standardized format for communication within BIM projects, providing a structured yet rich source of project-related discussions and issue tracking, which can be easily parsed.

Focusing on this data source enabled analyses that can lead to improvements in the planning process, decision-making, and overall project management in the construction industry.

**IFC and BCF Project Dataset** The data for this thesis was obtained from an open-source project published on GitHub (buildingSMART International, 2023) by buildingSMART (buildingSMART Technical, 2022), which is the organization behind both the IFC and BCF standards. By leveraging this project, this thesis aimed to develop methodologies that can be directly applicable to real-world construction projects, as a post-analysis method.

BCF is an already structured data format that combines XML files with additional resources like images, all compressed into a single package. Every BCF includes multiple files like the markup.bcf, the viewpoint.bcfv and snapshot images. The markup file contains the core information of the issue represented as XML. Furthermore, the viewpoint.bcfv contains the IFCGuid of related model components and the viewpoint of the camera in the model. Through this file, the BCF directly links discussion topics to specific elements within the BIM model, while also providing a viewpoint of the model (Borrmann et al., 2018).

As shown in Figure 3.2 each BCF XML contains a header with project and file information, a topic section describing the main issue, and one comment section detailing discussions related to the topic.

In this thesis, I did not only convert the structured data of the BCF into the KG, but also the text-based unstructured data. The focus was specifically on extracting and analyzing the semantic content within BCF files. By analyzing this semantic content, I aimed to uncover deeper insights that may not be immediately apparent in the structured data fields, like the data, author or referenced BIM objects. Through this approach, this information became available for further analysis. While traditional BCF analysis often relies on

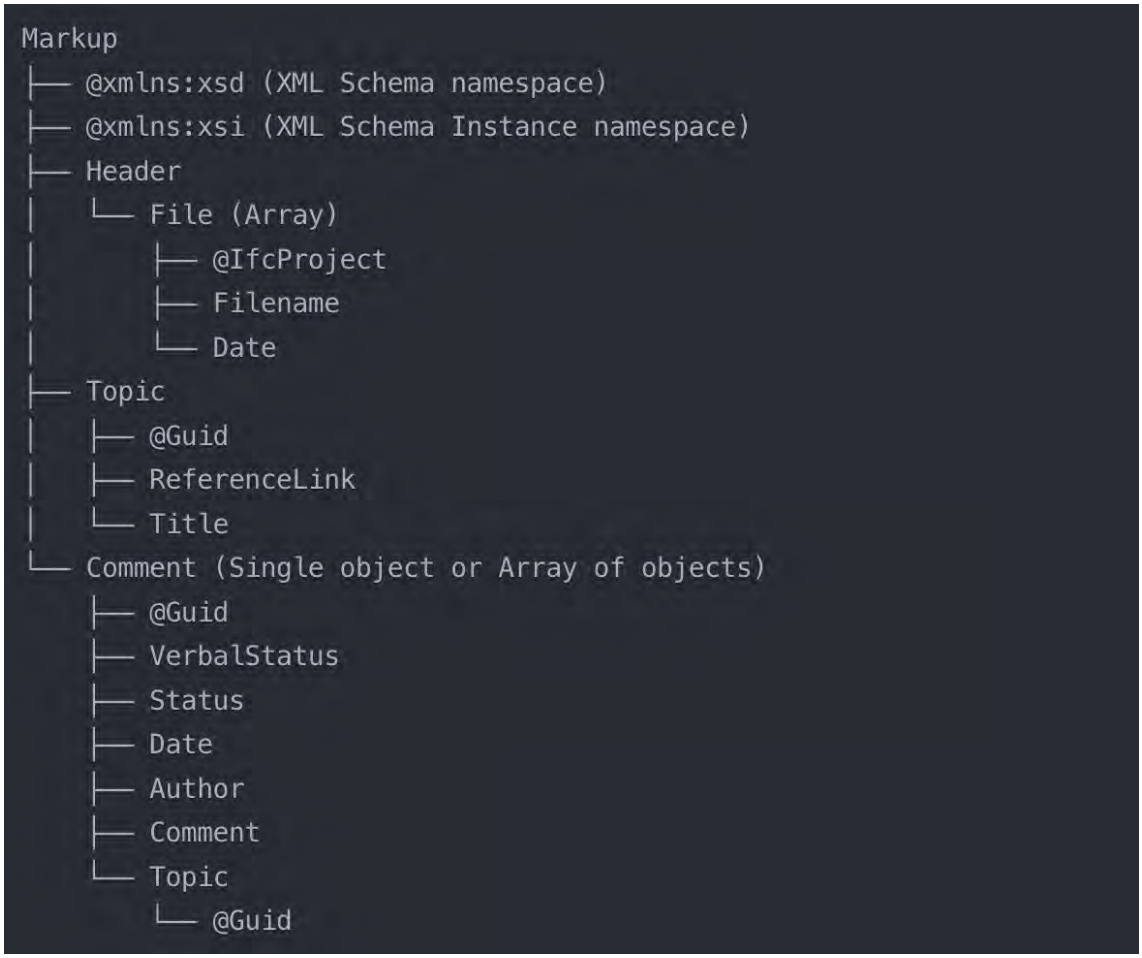


Figure 3.2: The structure of the BCF XML Markup (Version 1) of the dataset.

dashboards in tools like PowerBI or industry-specific software, these methods typically focus on quantitative metrics and predefined categories, from the classified or numeric fields of the BCF. My approach, in contrast, sought to delve into the thematic and semantic details of the discussions, potentially revealing nuanced patterns and knowledge that might be overlooked in standard evaluations.

Therefore, my aim was to extract further information from the comment text, which had been created by industry professionals during the project and is the only larger source of unstructured text in the BCF XML. Other already structured information, like the date or author, was directly converted into the KG. I used a LLM to interpret, classify and reason about the comment, enriching the KG for later analysis.

While the implemented approach focuses solely on comment extraction and indexation, this research also proposes a more comprehensive concept for BCF data analysis. This broader concept recognizes and suggests methods to leverage additional data points contained within or referenced by BCF files. As part of this expanded concept, I propose the use of Multimodal LLMs with vision capabilities to analyze images included in BCFs. This approach could provide the LLM with additional visual context, potentially enabling a deeper understanding of the described issues. Furthermore, the concept includes methods to derive and utilize component information from the IFC model through the component IDs

provided in the viewpoint.bcfv file. This information could either be integrated directly into the graph or provided to the LLM as additional context during the comment interpretation process.

Figure 3.3 illustrates both the implemented approach (represented by solid arrows) and these conceptually developed, but not yet implemented, elements (represented by dotted arrows). While the implementation of these additional elements is beyond the scope of the current work, they represent potential avenues building upon this thesis.

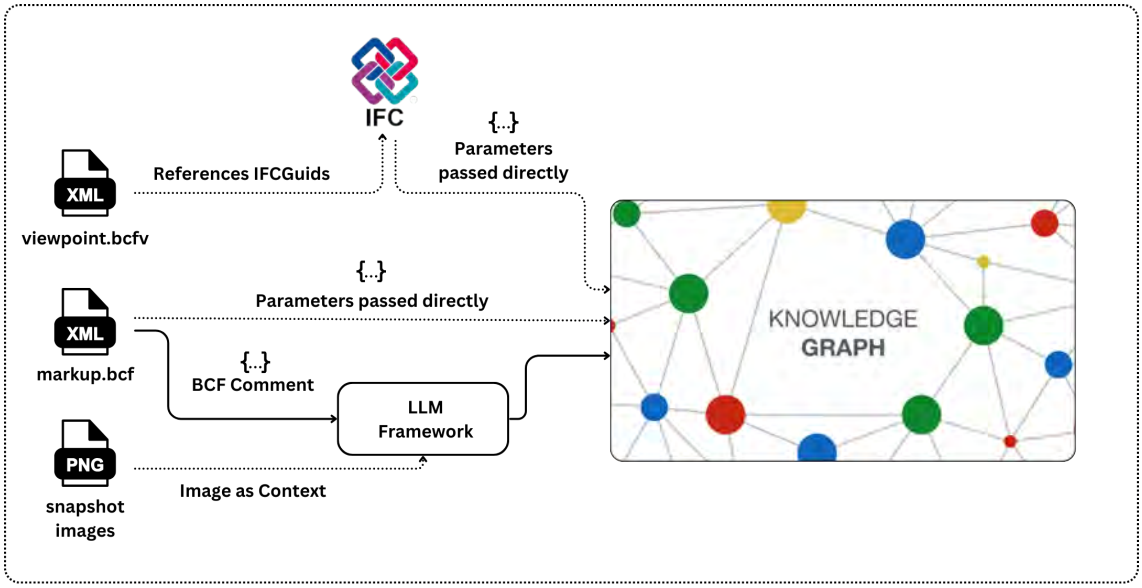


Figure 3.3: The overall concept of restructuring and indexing the BCF dataset into a KG.

## 3.2 Graph Indexing

I constructed a KG to systematically organize the information extracted from BCF comments. I applied an LLM to create descriptions of entities and relationships about a BCF comment text, which is a form of abstractive summarization, relying on the LLM to create independently meaningful derivations that may be implied but not stated by the text itself, similar to the approach by Edge et al. (2024). Therefore, the main task performed by the LLM was query-focused summarization (Dang, 2006). In the following subsections, I first explain how LLMs are leveraged to construct the KG (subsection 3.2.1). I then describe the graph structure in detail (subsection 3.2.2), before explaining the prompt chain used within my methodology to construct the KG (subsection 3.2.3).

### 3.2.1 Leveraging LLM for Knowledge Graph Construction

For KG construction from BCF comments, I leveraged the generalization capabilities of LLMs. Recent advancements in LLM technology, including models from the GPT (Brown et al., 2020), Llama (Touvron et al., 2023), and Gemini (Team et al., 2023) series, have demonstrated the ability to perform complex tasks such as query-focused summarization

without task-specific training (Edge et al., 2024). This is particularly advantageous for the work with BCFs, as it eliminates the need for large amounts of BCF-specific training data. Unlike earlier approaches that required extensive domain-specific datasets for tasks like query-focused text summarization (Laskar et al., 2022), modern LLMs can effectively process and summarize content provided within their context window through in-context learning. This capability allowed me to apply sophisticated natural language processing techniques to BCF comments, without the constraints of limited domain-specific training data.

### 3.2.2 Graph Structure Definition

In defining the graph structure for the BCF knowledge representation, I employed a two-level architecture with individual BCF trees and one merged BCF graph, as illustrated in Figure 3.4 and 3.5, similar to Xu et al. (2024) proposal in another context.

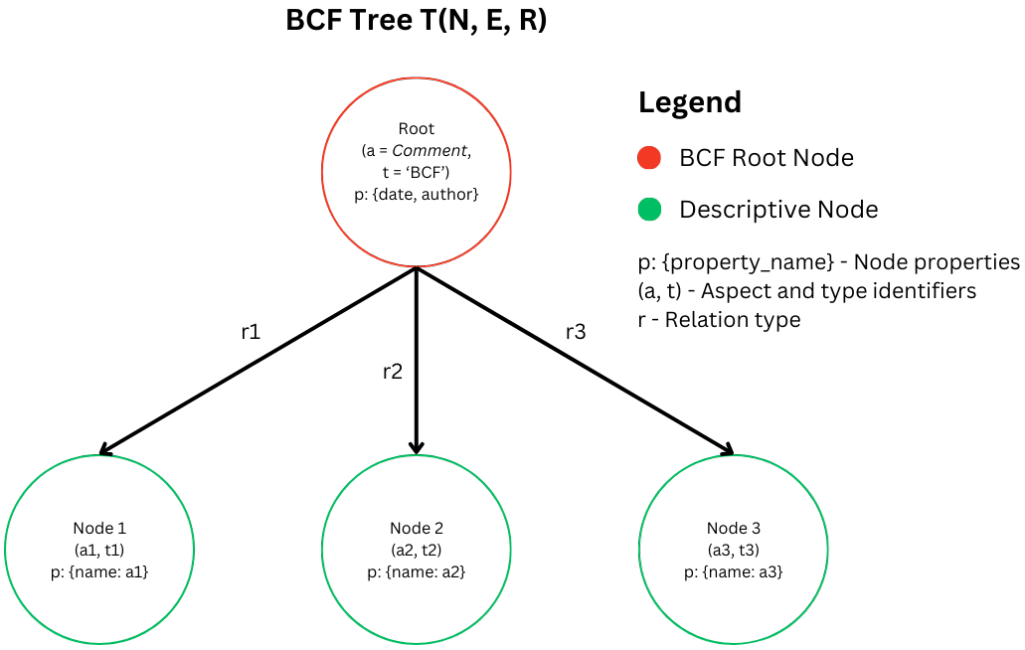


Figure 3.4: Graph Structure of the BCF Tree.

The BCF Tree $T(N, E, R)$ models each BCF comment as a shallow tree. In this structure, $N$ represents the set of nodes, where each node $n \in N$ is uniquely identified by $(a, t)$. Here, $a$ denotes a specific aspect of the BCF comment, and $t$ represents the node type. The root node of each tree, denoting the whole comment of the BCF as $a$, has the type $t$ = *BCF*. Each BCF comment node keeps the date of the BCF and the author of the comment as properties $p$. More metadata of the BCF can be included through the framework but is avoided here for the simplicity of the overall schema. All other nodes have types $t$ and aspects $a$ derived during the semantic analysis process by the LLM. For readability and schema clarification during KG retrieval, $a$ is also appended as a node property, with the key-name *name*. $E$ represents the set of edges connecting these nodes, and $R$ is the set of relation types that label these edges, which are also derived during the semantic analysis process by the LLM. The tree has a path length of one, meaning that each

descriptive node is directly connected to the root BCF comment node, without intermediate levels.

The Merged BCF Graph $G(N', E', R')$ is created by combining nodes with the same aspect and type $(a, t)$ across different BCF Trees. In this structure, $N'$ represents the set of unique nodes after merging, $E'$ is the set of edges that now connect these merged nodes to their respective BCF comment nodes, and $R'$ is the set of relation types preserved from the original trees. Nodes are combined only if they share the same aspect and type $(a, t)$.

This merging process creates connections between previously isolated BCF trees. The resulting graph structure never directly connects BCF comments, but rather through their shared descriptive nodes.

The significance of a descriptive node in this merged graph is determined by the number of connections it has to different BCF comment nodes. Nodes with numerous connections represent aspects or issues that are prevalent across multiple BCFs, indicating their importance in the overall dataset. This structure allows for the identification of common themes and recurring concepts across the entire set of BCF comments. Figure 3.5 provides a visual representation of this merged graph structure, highlighting how individual BCF comment nodes are interconnected through shared nodes of the same aspect and type $(a, t)$.

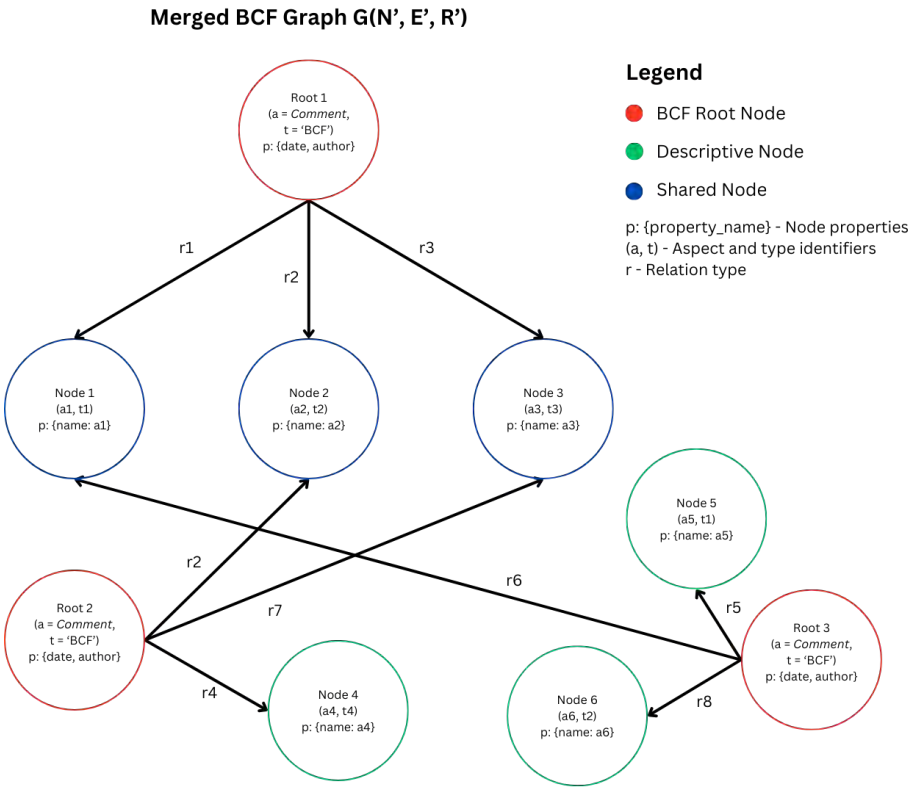


Figure 3.5: Graph Structure of the Merged BCF Graph.

### 3.2.3 Graph Chain

This subsection describes every step of the KG construction, starting with the translation and preparation, continuing with the theme tailored extraction and ending with the post processing of the extracted information for graph creation.

#### Translation and Preparation

As LLMs perform better when complex tasks are decomposed into manageable steps, I adopted a similar approach in my methodology. To analyze the given BCF dataset in English, the comment text first needed to be translated, taking into account the specific jargon of the construction industry. This task constituted the initial prompt in the chain. The text was translated in an exploratory manner, with the LLM being guided towards the domain-specific language of the construction industry.

When using a LLM, its performance can be improved by using a system parameter to give it a role. This technique, known as role prompting or role-play prompting consistently surpasses the standard zero-shot prompting approach across most datasets. The reasoning capabilities of an LLM can be directed to a specific domain and used to emulate a system conducting a specific task (Kong et al., 2023). In this context, I assigned the LLM the role of an engineer in the construction industry translating the comments.

```
system_prompt_template = (
    "You are an engineer in the construction industry. Please explain "
    "the issue of the following comment in a clear and concise manner "
    "in English. Do not mention any words in the original language of "
    "the comment. Moreover, do not infer or assume any additional "
    "information that is not explicitly mentioned in the comment, "
    "even if it's likely, be brief if the comment is short."
    "After explaining the comment just append a 1 to 1 translation of "
    "the comment in English at the end of the explanation. Just say: "
    "'Original Comment: -- Put 1-to-1 translation here --'\n\n"
    "Here is the comment:\n"
    "{comment}"
)
```

Normally, the chunk size of the input text for processing is an important design decision, when working with LLMs. As the used dataset is naturally split into separate BCFs with individual comments and each comment did not contain more than a few sentences. This resulted in a natural chunk size of around 60 tokens, with a maximum of not more than 150 tokens. Therefore, the LLM calls had no risk of suffering from recall degradation of longer LLM context windows (N. F. Liu et al., 2024).

### Theme Tailored Extraction

This step aimed to extract instances of graph nodes and edges from each BCF comment. When invoking the LLMs' interface, a specific output format is required to facilitate the code processing for the creation of the graph. In this case, the LLM was required to return a relation type, a tail node, and a tail node type in response to the prompt which specifies the head node and the head node type as the corresponding BCF comment.

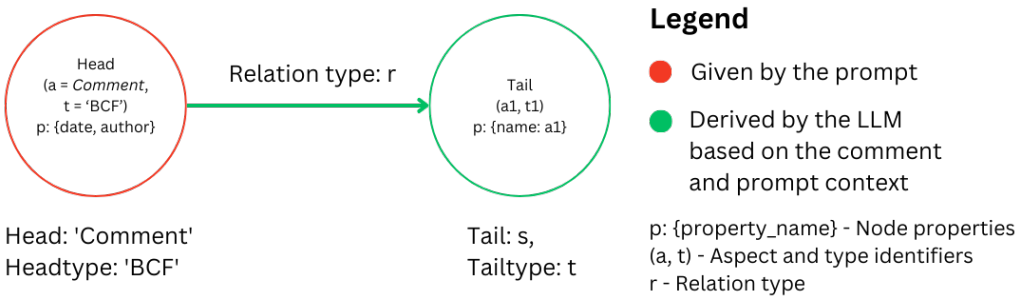


Figure 3.6: Components extracted through the LLM.

**Structured Extraction** To achieve a structured output from a LLM, I utilized the function calling capabilities of LLMs. Function calling, also known as tool calling, allows a model to respond to a given prompt by generating output that matches a user-defined schema (LangChain, 2024b). It is important to note that while the term "tool calling" might imply that the model is performing an action by itself, this is not the case. Instead, the model generates the arguments for a tool, and the execution of the tool, the creation of the graph, is left to the code framework implementing the LLM call.

Figure 3.6 displays the arguments that needed to be provided by the LLM to construct the graph. The head entity and type of the relationship were already provided within the prompt as the comment of the BCF. The LLM needed to generate the argument for the relation type, as a short descriptive string. The tail entity and the tail type also needed to be provided by the LLM as a short descriptive string. The parameters of all nodes could be determined after the graph construction, since the heads date and author parameter were contained within the corresponding BCF comment and the tail parameter name was $a$ and could be derived from the tail node tuple $(a, t)$.

**Prompting Techniques** The overall architecture of the prompt is inspired by an approach provided by Neo4j (Neo4j and LangChain, 2024). My approach leveraged similar prompting techniques, while adapting and expanding the prompt to specific needs. The whole prompt is written in markdown and has a clear enumerated structure with clear tasks. The system prompt began with role-based prompting (Kong et al., 2023), directing the LLM to assume the role of an expert engineer constructing knowledge trees within the construction industry domain.

```
"## 1. Overview\n"
"You are a state-of-the-art AI designed to extract structured
    information from comments "
"in the construction and engineering domain. Your task is to build a
    knowledge graph "
"that accurately represents both the explicit and implied content of
    each comment.\n\n"
```

The next part of the prompt was to provide the LLM with clear extraction guidelines: It described the graph schema to provide the LLM with a clear guide to organizing and extract information about the BCF comment. It is important to highlight that the LLM was prompted to return clear and concise labels. This is a key part of the indexing approach, as those precise labels and relationships helped to interconnect the graph and to find similarities during query time.

```
"## 2. Extraction Guidelines\n"
"- Extract information that is stated in the comment.\n"
"- Include information that is implied by the comment in context with
    the theme-specific focus.\n"
"- Maintain accuracy: if in doubt, err on the side of caution.\n\n"
"## 3. Knowledge Graph Structure\n"
"- For every extraction reason only about the comment text and
    therefore always fill the head with:\n"
"  - head: 'Comment'\n"
"  - head_type: 'BCF'\n"
"- Nodes: Represent entities, concepts and themes mentioned or implied
    by the comment in context with the theme-specific focus.\n"
"- Relationships: Capture connections to entities, concepts and themes
    identified.\n"
"- Use clear, concise, and consistent labels for node types and
    relationships.\n\n"
```

**Themes** A key step was the inclusion of special themes within the prompt. These themes were based on analysis questions and were formulated through different approaches, either by analyzing a subset of the dataset to identify recurring themes or by deriving them from the overarching analysis objectives I aimed to address across the entire dataset. Themes allowed me to focus the LLM's attention on particularly relevant or recurring topics. For each BCF comment, the LLM was directed to reason about these specific themes, with the results of this reasoning process being incorporated into the KG. Thereby, I indexed the dataset under different thematic angles that can be customized.

The themes were important when answering global questions to ensure that the higher-level understanding and reasoning perspective required to address these questions is embedded within the KG structure. This approach provided consistency while reasoning about each BCF comment, while also not limiting the LLM and encouraging it to explore other themes not specifically mentioned. Figure 3.7, shows exemplary themes of interest

for the BCF dataset. As an example, I decided to analyze which components were a recurring topic within the BCF comments. By specifying this theme *Components* during the indexing process, my goal was to answer questions like "Which components are mentioned the most?", during query time. I also partly specified, examples for categories of themes, like different request types. This allowed the LLM to index according to the specified categorization.

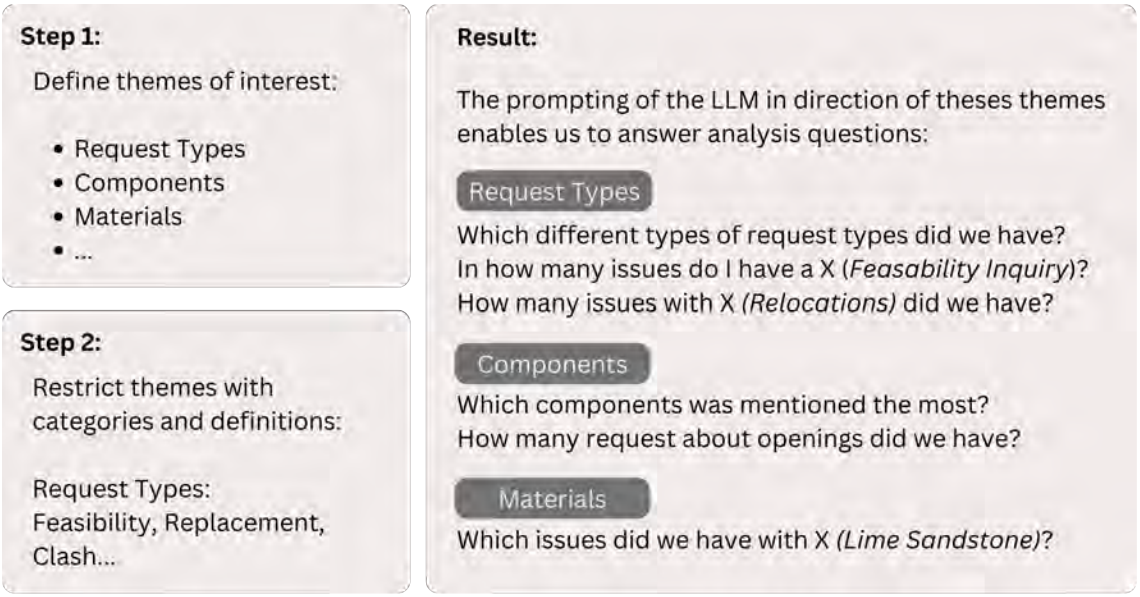


Figure 3.7: Thematic definition and corresponding questions with examples.

The prompt as a whole has the ability to be applied several times with different themes. This allows for a scalable level of detail in the KG – the more often this prompt is applied to reason about and explore different themes, the more detailed and comprehensive the graph can become. This means one can iteratively enrich the knowledge representation by repeatedly querying the LLM about various aspects of each BCF comment.

```
"## 4. Theme-Specific Focus\n"
"Identify and categorize the following:\n"
"- Request types (Feasibility, Replacement, Clash...)\n"
"- Components \n"
"- Materials \n\n"
```

The listing, displays the part of the system prompt that declared the theme-specific focus, showcasing three exemplary themes including one theme categorization. In the chain, I applied this whole prompt, displayed in Figure 3.8, once for each BCF comment, as this was sufficient to extract several nodes and relationships around one BCF comment.

To tailor this prompt to the domain and to enhance the overall quality of its output, I incorporated few-shot examples in the prompt for in-context learning (Brown et al., 2020). It is important to mention that these few-shot examples also needed to contain thematic examples to enhance the LLMs output in perspective to the themes specified.

**System Prompt**

#Prompt
## 1. Overview
You are a state-of-the-art AI designed to extract structured information from comments in the construction and engineering domain. Your task is to build a knowledge graph that accurately represents both the explicit and implied content of each comment.

## 2. Extraction Guidelines
- Extract information that is stated in the comment.
- Include information that is implied by the comment in context with the theme-specific focus.
- Maintain accuracy: if in doubt, err on the side of caution.

## 3. Knowledge Graph Structure
- For every extraction reason only about the comment text and therefor always fill the head with:
  - head: 'Comment'
  - head_type: 'BCF'
- Nodes: Represent entities, concepts and themes mentioned or strongly implied by the comment in context with the theme-specific focus.
- Relationships: Capture connections to entities, concepts and themes identified.
- Use clear, concise, and consistent labels for node types and relationships.

## 4. Theme-Specific Focus
Identify and categorize the following:
- Request types (Feasibility, Replacement, Clash...)
- Components
- Materials

## 5. Quality Control
- Maintain consistency in terminology and relationship types across extractions.
- Prioritize accuracy over comprehensiveness, but aim to capture all relevant information.
- Do not infer complex technical details or relationships that require specialized knowledge.

## 6. Strict Compliance
Adhere to these guidelines strictly. Always use 'Comment' as the head and 'BCF' as the head_type.

Figure 3.8: The combined system prompt, excluding few-shot examples.

#### Post Processing for Graph Creation

The next step was to use the structured output provided by the LLM to construct the KG. Since there is a chance that the LLM returned invalid results that do not adhere to the schema, I checked for their validity and dropped invalid results before constructing the graph. I combined each BCF Tree created, described in section 3.2.2, into the Merged BCF Graph by merging the descriptive nodes with the same aspect and type $(a, t)$.

## 3.3 Graph Retrieval

After constructing the knowledge graph from BCF comments, the next crucial step was to effectively retrieve and analyze the stored information. This section describes various methods used for querying and extracting insights from the graph structure. I will propose

different approaches, each offering unique advantages in accessing and interpreting the knowledge embedded within the graph.

The proposed methods range from direct querying using the Cypher (Neo4j, 2024b) language to more advanced techniques involving prompt chains, and AI agents. Each approach provides different capabilities in terms of deterministic evaluation, semantic search, and complex reasoning. By leveraging these diverse methods, I aimed to demonstrate the flexibility and power of the graph-based knowledge representation.

### 3.3.1 Cypher Query

The foundation of my graph retrieval methods is Cypher (Neo4j, 2024b), a declarative query language specifically designed for interacting with graph databases. Originally developed as Neo4j's native query language, Cypher has become a standard in graph querying due to its intuitive syntax and powerful capabilities in graph traversal (Holzschuher & Peinl, 2013). Cypher's syntax has some similarities to SQL, making it accessible to those familiar with relational database querying. The language's design philosophy emphasizes readability, which significantly reduces the complexity of working with graph-centric applications (Holzschuher & Peinl, 2013).

Cypher offered several key advantages for querying the BCF-derived KG:

1. **Schema Exploration:** Cypher provides commands to return the graph schema, facilitating dynamic exploration and understanding of the knowledge structure.
2. **Deterministic Evaluation:** Cypher allows for precise, deterministic querying of the graph structure. This enabled exact counts, aggregations, and pattern matching across the KG.
3. **Metadata Filtering:** Cypher supports filtering based on node and relationship properties and labels, respectively. This approach allowed me to query based on BCF metadata such as dates, authors, or any other attributes incorporated into my graph structure.
4. **Full-Text Indexing:** In combination with Cypher graph databases can support full-text indexes on string properties. This feature is particularly valuable for my KG, as it enables semantic similarity searches on node aspects and tolerance for misspellings in queries (Neo4j, 2024a).

### 3.3.2 A Semantic Layer through Agent with Tools

The created graph can be challenging to interpret and analyze for non-technical users. Therefore, I developed an agent-based approach that enables users to interact in a conversational manner. This allows users unfamiliar with graph databases or technical query languages to explore the graph in detail.

As outlined, LLM based agents can understand natural language queries, make decisions, and execute actions using a set of predefined tools. Within this framework, these tools form a semantic layer between the user's natural language input and the graph database. The semantic layer acts as a bridge, translating user intent into specific, actionable queries that the agent can execute with accuracy and reliability. Each tool in the semantic layer can be thought of as a function with a specific purpose. For my approach, I equipped the agent with the following three tools and tested its capabilities to communicate with the user based on information retrieved from the graph.

**Tool 1: Find Major Aspects** This tool identifies and returns the first three nodes of a certain type $t$, which have the most connections to BCF comment nodes. The node type $t$ could be for example *Component* or *Request Type*.

```
def find_major_aspects(node_type: str) -> str:
    query = f"""
    MATCH (t:BCF)-[r]->(c:{node_type})
    RETURN c.id AS node_id, COUNT(r) AS connections
    ORDER BY connections DESC
    LIMIT 3
    """
    result = graph.query(query)
    output = f"Top 3 {node_type}s with the most relations to BCF
        comments:\n\n"
    for i, item in enumerate(result, 1):
        output += f"{i}. {item['node_id']}: {item['connections']}
            connections\n"
    return output
```

**Tool 2: Aspect Analyzer Tool** This tool counts the number of BCF comments which are connected to a node with a certain aspect $a$. In the beginning, the entity chain extracts all entities mentioned in the question through a LLM. Next, the extracted entities are matched against the node aspects $a$ of the graph. This matching is conducted through a full-text index, which allows for small spelling mistakes and therefore minor deviations in entities. The same entities with minor deviations in spelling within the graph can therefore all be found and evaluated together.

Moreover, the tool provides the content of the BCF comments alongside the total count, offering a brief overview. Examples of aspects include: *Feasibility Inquiry*, *Relocations*, *Openings*, *Lime Sandstone*

```
def aspect_analyzer_tool(question: str) -> str:
    result = f"Queried graph for entities related to: '{question}'\n\n"
    entities = entity_chain.invoke({"question": question})

    def generate_entity_resolution_full_text_query(input: str) -> str:
        words = [el for el in remove_lucene_chars(input).split() if el]
```

```
        query = " AND ".join(f"{word}~2" for word in words[:-1])
        query += f" {words[-1]}~2" if words else ""
        return query.strip()

    for i, entity in enumerate(entities.names, 1):
        result += f"Entity {i}: {entity}\n{'=' * (len(entity) + 10)}\n"
        response = graph.query(
            """CALL db.index.fulltext.queryNodes('indexForThemes',
               query, {limit:2})
            YIELD node, score
            WHERE score >= 1.2
            MATCH (node)<-[r]-(neighbor:BCF)
            RETURN node, neighbor.id AS output, score
            """,
            {"query":
                generate_entity_resolution_full_text_query(entity)})

    {... Formatting and counting of the result before returning ...}

    return result
```

**Tool 3 and 4: Query Node Types and Aspects** Tool 3 returns all the different node types $t$ present in the graph. Tool 4 returns all the different node aspects $a$ of a certain type $t$ in the graph.

**Agent Tool Choice** The LLM agent selects the appropriate tools based on its understanding of the user's query. Each tool is passed with a description to the LLM agent, enabling the LLM agent to determine when to use which tool. By leveraging these tools, the agent gains the ability to interact with the KG and perform queries according to the user's request. Furthermore, through the use of the tools' capabilities like counting or pattern matching via Cypher, the LLM agent acquires new capabilities. Since the tools that do not leverage an LLM internally always return the same results, the trustworthiness and reliability of the system are enhanced, mitigating the risk of incorrect and irrelevant outputs.

Furthermore, the agent can interpret the results returned by these tools, providing explanations and insights in natural language. This capability allowed for a more interactive and informative exploration of the KG.

### 3.3.3 Agent-Writing Cypher

While this tool-based agent approach provides an interactive way of engaging with the KG, there are scenarios where more flexibility is needed. An agent leveraging tools is only as capable as the tools provided. Because these tools rely on static query templates,

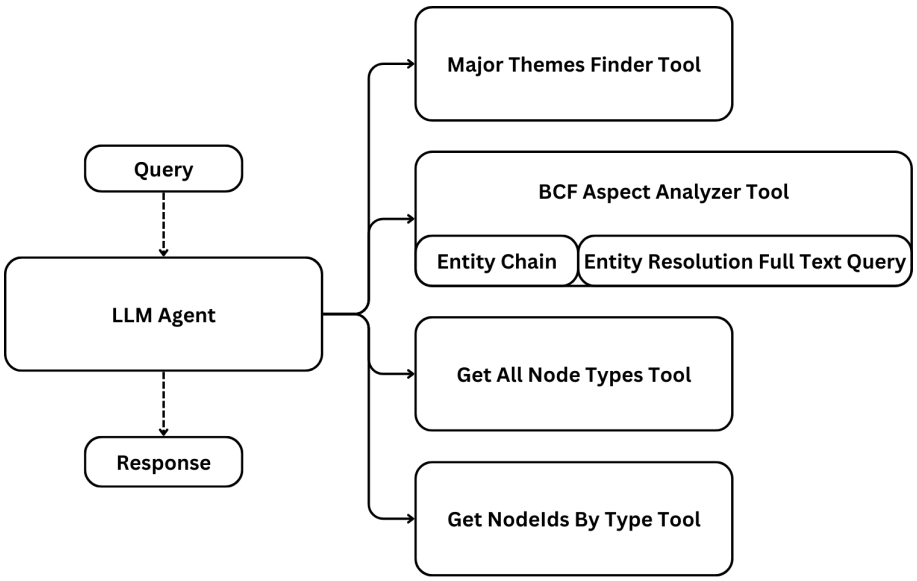


Figure 3.9: The LLM agent leveraging different tools to interact with the knowledge graph to answer the query.

the agent is limited in the interaction with the graph through those templates. Manual engineering effort would be needed to create new tools. Moreover, tools can never sufficiently cover the wide complexity and variety of questions, a user can pose towards the KG.

To avoid this effort, one option is to use a LLM-based agent to generate Cypher queries. The agent can dynamically construct Cypher queries tailored to specific user requests, potentially creating more complex or customized queries than what is possible with predefined tools. However, this flexibility comes with challenges as the agent must generate syntactically correct Cypher queries that accurately reflect the user's intent. Moreover, the agent needs to be aware of the KG's schema to generate valid queries. For my approach, I tested an implementation provided in collaboration by LangChain and Neo4j (2024), which can generate and execute Cypher queries based on the created graph schema and prompt.

# Chapter 4

# Proof of Concept and Implementation

This chapter describes the proof of concept and implementation of the framework proposed in my methodology. To test my approach I used an open-source project buildingSMART published on GitHub (buildingSMART International, 2023). In the first section, I will describe the obtained dataset of BCF files in detail, with a focus on the abnormalities of this dataset. In the next section, I will shortly describe how I parsed and flattened the obtained BCF files for further processing. Following, I will argue why I used LangChain as a framework to apply LLMs to my data and present other framework alternatives. In the section after, I will argue about the choice of LLM used. The next section focuses on Neo4j, comparing it to other database systems and lining out why I chose a graph database.

## 4.1 Public buildingSMART Dataset

The published dataset from buildingSMART consists of 93 BCF files (version 1) in the Dutch language. It exhibits some unique characteristics, which need to be considered in the analysis process.

All BCF comments are in Dutch, with the topic title always being "Opmerking" (Remark). In a wider sense, this topic title always describes the BCF correctly, since every BCF is a remark. However, no further conclusions can be made on the actual content of the remark which prohibits further analysis of this property. The same holds true for the status field of each comment in each BCF file, which is always set to "Unknown". Either the software used to create the BCF provided no option to set this field or the authors just decided to not make use of it, which leaves less room for the analysis.

Notably, 91 of the 93 BCFs files were created and edited by a single author, while the other two files are from one different author. Moreover, four BCF files contain more than one comment. All these four BCFs contain two comments from the same author. Therefore, no discussions of different authors exist in the comments of the BCF dataset. The BCF file format seems to have been used as a one-way communication format, possibly only notifying different stakeholders of inquiries or requests. This restricts the scope of analysis in the dataset since two-way communication or discussions would provide more opportunities for further semantic analysis and knowledge derived from the engineering and planning process.

Since the BCFs are file-based, the entire file is duplicated, when editing or replying to a comment. This further limits the number of BCF comments available for analysis. Only

84 BCFs have a unique ID and topic. From the 93 BCF files five contain an edited comment with the same ID and minor changes, mainly in numbering or small details. Four files contain one additional comment. Therefore, in total the dataset contains 88 unique comments to analyze.

A deviation from typical BCF usage can be observed in the *VerbalStatus* field, which, in the dataset contains company or group names rather than the descriptive text of the BCF statuses. It appears the field has been used to label the addressees of the BCF.

Using the BCF format provides both opportunities and challenges for knowledge extraction and analysis in my research. Given the topic titles in the dataset are generic, the comment status is set to unknown, and the communication flow is unidirectional, the BCF XML file (Figure 3.2) does not provide additional context valuable for the LLM when analyzing the comment text for graph creation. Therefore, only the comment text is processed directly by the LLM within the graph creation pipeline, while selected other parameters can be assigned directly to their corresponding BCF comment node within the graph for later analysis and reasoning.

## 4.2 BCF File Acquisition and Processing

My approach to BCF file acquisition recognized two primary technical methods: accessing BCF files directly and utilizing the BCF API (buildingSMART, 2023) for server retrieval. For this research, I used BCF data published on GitHub by buildingSMART. As this data is not hosted in a Common Data Environment (CDE) with a BCF API, I had to clone the GitHub repository to my local storage to process the data. It is worth noting that in scenarios where data is provided from a CDE, the open BCF API could be leveraged to fetch entire BCF contents, streamlining the data acquisition process in applications. This does have several advantages, one of them being that there would be no need to store large amounts of files locally. While my current implementation involved storing and processing BCF files locally, my methodology was designed to be adaptable to API-based retrieval in future applications.

The initial step of the preprocessing pipeline was to decompress BCFZIP files. BCFZIP is a compressed format commonly used to combine and distribute multiple BCF files efficiently. This decompression step is necessary to access the individual BCF files contained within the BCFZIP package (Figure4.1.

Once extracted, I processed each BCF, parsing each BCF XML file to read its content. The XML structure was then flattened to create easily accessible properties, a process that involves de-nesting all information from the XML structure and organizing it into a flat format where each piece of information is directly accessible (Figure 4.2. In cases where a BCF contains multiple comments, the step involved separating each comment while maintaining the overall BCF context properties. This separation ensured that each comment was treated as a distinct entity in the graph creation process while retaining

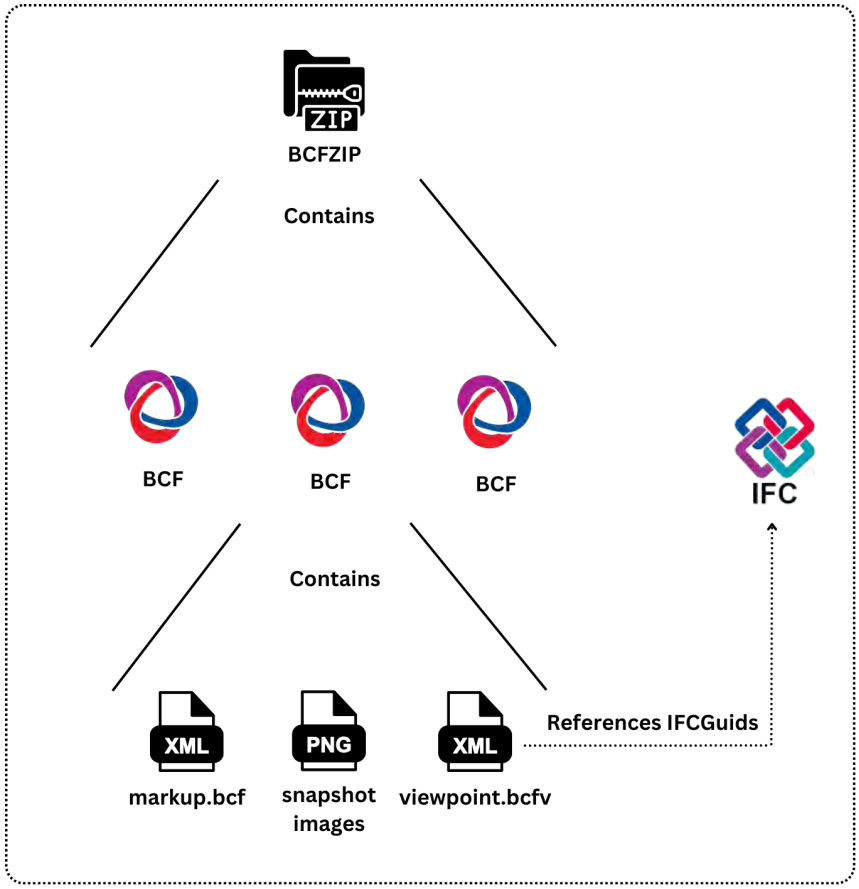


Figure 4.1: A BCFZIP decompressed in its parts. A BCFZIP contains several BCFs, which each, in turn, contain a markup.bcf, a viewpoint.bcfv, and potentially several snapshot images.

the properties of its parent BCF for context. This approach allowed for a more granular analysis of each comment text while preserving the relationship between comments and their originating BCFs.

I also considered concatenating all comments of a BCF and providing them as a single prompt to the LLM. However, this approach offers less granular analysis and is only eligible when all comments are from the same author. If comments come from different authors, the LLM would need to be prompted differently, requiring a structured prompt to denote all authors in the conversation thread. Consequently, a BCF with multiple comments would need to be treated differently in the indexing pipeline and within the created graph. However, when processing each comment separately while preserving the relationship between the comments within the graph, I applied the same indexing process to each BCF no matter the number of comments while still being able to analyze discussions within the graph, through their shared properties. Therefore, treating each comment separately provides more benefits.

Through these preprocessing steps, I transformed the raw dataset into a structure optimized for the subsequent graph construction and analysis phases. This preparation ensured that I could efficiently extract and represent the knowledge contained within each BCF in the graph structure.


Figure 4.2: Restructured BCF XML file parsed into JSON. All properties are de-nested making them accessible from the top level.

## 4.3 LangChain: A Framework for LLM-Powered Applications

In my implementation of the methodology, I utilized LangChain, 2024a, an open-source framework designed for developing applications powered by LLMs. Central to LangChain is the Runnable interface, which forms the foundation for the entire framework. This interface defines a standard set of invocation methods for various components within a LLM powered application, enabling the creation of modular and interchangeable components. In my implementation, I employed two key methods of this interface: `invoke()`, which executes the components synchronously with a single input, and `stream()`, which allows for streaming output generation, useful for real-time or incremental processing.

This standardization allows to easily combine and reconfigure different components of LLM applications. Building upon this core interface, LangChain offers a wide variety of different key components, within its ecosystem, most of them developed by the open-source community. The most basic components include:

1. Prompt Components: These define the input for the LLM, allowing for dynamic template generation.
2. LLM Components: These wrap around various language models from different providers (OpenAI, Anthropic, Hugging Face, etc.), providing a consistent interface for interaction.
3. Output Parsers: Tools for processing and formatting the output from LLMs. Can be used to convert raw text to more structured formats.

4. Chains: Sequences of operations that combine multiple components implementing the Runnable interface, allowing for complex workflows to be constructed from simpler building blocks.

This modular architecture allowed me to easily construct sophisticated LLM pipelines by combining these components in various ways. Chains, in particular, serve as a powerful abstraction, enabling the creation of complex, multi-step processes.

In this simple example, I demonstrate how LangChain's components work together to create a functional LLM application with minimal code:

```
from langchain import PromptTemplate, LLMChain
from langchain.llms import OpenAI

# Define a prompt template
template = "What is a good name for a company that makes {product}?"
prompt = PromptTemplate(template=template, input_variables=["product"])

# Create an LLM chain
llm = OpenAI()
chain = LLMChain(llm=llm, prompt=prompt)

# Run the chain
result = chain.run("eco-friendly bricks")
print(result)
```

### 4.3.1 LangChain Expression Language (LCEL)

Building upon the Runnable interface, LangChain introduces the LangChain Expression Language (LCEL), which further simplifies the process of constructing complex chains. LCEL utilizes the pipe operator (|) to create more readable chain compositions. This approach makes the creation and modification of LLM chains even more straightforward. Here is an example of how LCEL can be used to create a chain:

```
from langchain import PromptTemplate, LLMChain
from langchain.llms import OpenAI
from langchain.schema import StrOutputParser

# Define components
prompt = PromptTemplate.from_template("What is a good name for a
    company that makes {product}?")
model = OpenAI()
output_parser = StrOutputParser()

# Compose the chain using LCEL
chain = prompt | model | output_parser
```

```
# Run the chain
result = chain.invoke({"product": "eco-friendly bricks"})
print(result)
```

In this example, the pipe operator (|) is used to connect the prompt, model, and output parser into a cohesive chain. This syntax not only makes the code more readable but also allows for easy modification and extension of the chain.

LCEL's expressive power extends beyond simple linear chains. It provides built-in support for more complex operations such as parallel processing, conditional branching, and error handling. These features, combined with LangChain's modular architecture, enable developers to create sophisticated LLM-powered applications while maintaining clean and maintainable code structures.

### 4.3.2 Agent Component in LangChain

A key feature of LangChain that I utilized in my methodology during graph retrieval is its Agent component. It provides an abstraction layer for creating autonomous, tool-using LLM applications. At the core of this functionality is LangChain's Agent Executor, which manages the agent's tool-calling capabilities and serves as the runtime of the agent.

The Agent Executor orchestrates a complex process:

1. It calls the agent to determine the next action.
2. It executes the chosen action using the appropriate tool.
3. It returns the action outputs to the agent.
4. It iteratively repeats this process until the task is complete.

One of the most significant advantages of using LangChain's Agent component is its ability to abstract numerous runtime complexities. The Agent Executor handles various edge cases and potential issues, including, managing scenarios where the agent selects a non-existent tool, handling errors that occur during tool execution or dealing with unparseable outputs from the LLM.

### 4.3.3 LangChain Alternatives

While LangChain was my framework of choice for this research due to its flexibility and comprehensive feature set, it is important to acknowledge that there are other frameworks available for LLM application development. Alternatives such as LlamaIndex (LlamaIndex, 2024), and Haystack (Deepset, 2024) offer their own unique features and approaches to working with LLMs. LlamaIndex, for instance, specializes in data ingestion and indexing, making it particularly useful for applications that require efficient querying of large datasets.

Haystack, on the other hand, focuses on question-answering systems and offers strong support for document retrieval tasks. The choice of framework ultimately depends on the specific requirements of the project, the developer's familiarity with the tools, and the particular strengths of each framework. While I found LangChain to be the most suitable for my needs, other researchers or developers might find that alternative frameworks better align with their project goals or workflows.

## 4.4 Choice of LLM

The landscape of LLMs is dominated by several key providers, each offering unique models with varying capabilities and access methods. In this section, I will provide a brief overview of different models, as well as providers and shortly argue about the rationale behind my choice of LLM for my case study.

**Overview of LLM Providers** LLM providers are organizations that develop, maintain, and often provide access to LLMs, offering various services ranging from API access to open-source distributions. The field of LLMs is rapidly evolving, with several major players:

- **OpenAI**: Pioneers in the field, offering the GPT (Generative Pre-trained Transformer) series, including the widely-used GPT-3.5, GPT-4 and GPT-4o. These models are easily accessible through API calls and are known for their versatility and strong general performance (Radford et al., 2019) (Brown et al., 2020) (OpenAI, 2024).
- **Anthropic**: Creators of Claude, focusing on developing AI systems with enhanced safety features and ethical considerations. Anthropic regularly publishes research on AI safety, scalable oversight, and improving AI systems. They're exploring ways to make AI systems more reliable and interpretable (Templeton, 2024) (Anil et al., 2024).
- **Google**: Provides access to models like PaLM (Pathways Language Model) through its VertexAI platform. These models leverage Google's vast computational resources and are integrated with other Google Cloud services (Chowdhery et al., 2023).
- **Microsoft**: In partnership with OpenAI, Microsoft offers Azure OpenAI Service, providing access to models like GPT-3.5, GPT-4 and GPT-4o through the Azure cloud platform.
- **Meta**: Released open-source models like LLaMA (Large Language Model Meta AI) (Dubey et al., 2024), which researchers and developers can download and run locally or fine-tune for specific applications.
- **Mistral AI**: Offers both open-source models and API access, positioning them as an alternative to both fully closed commercial providers and purely open-source initiatives. (Jiang et al., 2023)

**Selected Model: GPT-4o** For the case study, I selected GPT-4o, as it performs at the cutting edge in LLM technology. GPT-4o's advanced understanding is an important ground for the domain-specific analysis of construction project communication data. The model's robust performance across various tasks, among others in complex reasoning and analysis, made it the most suitable. Nevertheless, through leveraging the LangChains framework it is easily possible to change the LLM used through a few lines of code. Therefore, in further case studies one can use different LLMs at different parts of the pipeline. LLMs can also be fine-tuned through services provided by their providers. Going deeper it is also possible to leverage open-source models hosting them locally or through a provider like Google's VertexAI platform. At this stage of research, where I tested the general feasibility of my approach, fine-tuning and locally hosting an open-source LLM would introduce misleading complexity. Additionally, GPT-4o's accessibility through API calls facilitated seamless integration into my research pipeline, streamlining the implementation process.

**Configuration Details**

- **Model**: GPT-4o
- **Temperature**: Set to 0 for maximum consistency
- **Access Method**: OpenAI API

## 4.5 Choice of Database System

When selecting a database for my implementation, I considered mainly two types of database systems: MySQL and NoSQL databases.

Traditional relational databases, such as MySQL, offer structured data storage with predefined schemas, while NoSQL databases provide more flexibility in data representation. NoSQL databases can be schema-less or have flexible schemas (Li & Manoharan, 2013). This characteristic is particularly beneficial for my approach, as it enables the LLM to explore and generate new node types in my graph structure. A database that can easily accommodate schema changes or expansions is crucial for dynamic knowledge representation.

Graph databases, a subset of NoSQL databases, are specifically designed to store and query graph-like structures (Vicknair et al., 2010). They excel at representing and traversing relationships between data points, which aligns with my methodology's graph-based knowledge representation. The schema described in my methodology naturally maps to a graph structure, making graph databases an ideal choice for my implementation.

One of the primary advantages of graph databases over traditional relational databases is their ability to perform complex joins - akin to joins in SQL databases over multiple tables - more efficiently. Graph databases can traverse relationships (edges) between nodes with minimal computational overhead, allowing for quick and efficient querying of

interconnected data (Hogan et al., 2021). Moreover, the overall graph structure provides benefits in terms of data representation and query flexibility. Cypher, a declarative query language specifically designed for property graphs, facilitates these benefits (Francis et al., 2018). It provides powerful capabilities for querying and modifying data and its pattern-matching capabilities are intuitive and efficient for querying the interconnected structure of my graph representation of BCF comments. Several graph database options are available, including Neo4j (Neo4j, Inc., 2024), ArangoDB (ArangoDB GmbH, 2024), and NebulaGraph (vesoft inc., 2024).

I chose Neo4j for my approach in this thesis due to its alignment with my data indexing structure, as a graph. Moreover, Neo4j supports property assignment to both nodes and relationships, which was also a key requirement for my implementation. This feature allows to keep the graph schema clear while still enabling the addition of supplementary information to nodes as needed (Guia et al., 2017). Additionally, Neo4j natively uses Cypher for querying and navigation, which was crucial for the analysis phase of my methodology.

# Chapter 5

# Results

In this chapter, I present the results of the framework implemented. I will first introduce the results of the graph indexing chain based on the described BCF dataset, including the selection of three different themes contained in the prompt. In the subsequent section, I will evaluate the different retrieval methods. Both sections include different case studies that analyze the results of my methodology.

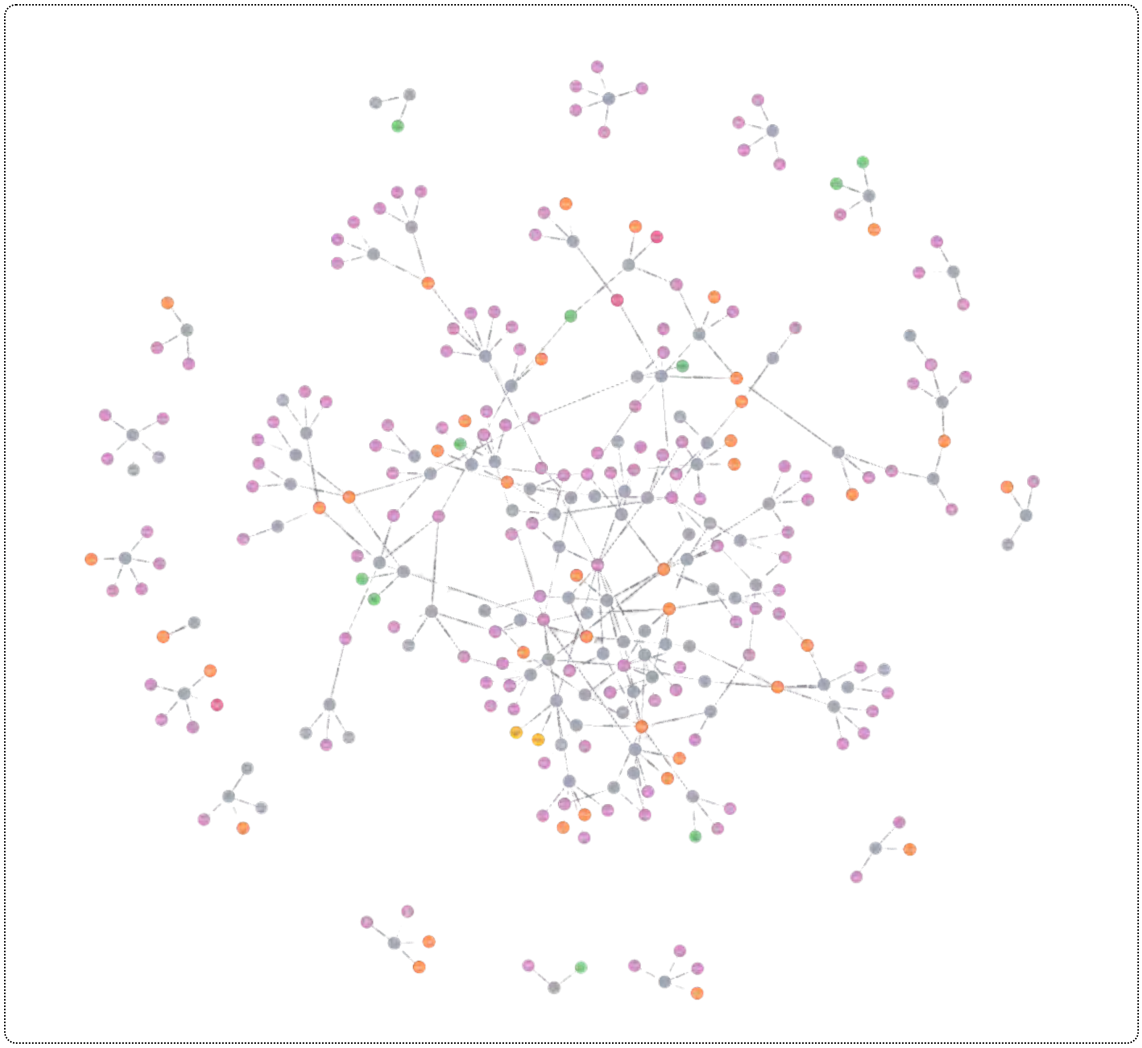

Figure 5.1: Graph created by the Graph Indexing Chain.

## 5.1 Graph Indexing Process

For the graph indexing process, I selected the themes *Request Types*, *Components* and *Materials*, which were also given as an example in the methodology. Within this chapter, I

conduct case studies to qualitatively analyze how well the graph was indexed according to those different themes. Before doing so I will analyze the overall graph creation process.

**Analysis of the Graph Creation Process** The graph was successfully created by the LLM prompting chain, adhering to the structure specified. Only one relation did not adhere to the schema specified and had to be removed automatically.

Algorithm 5.1: Created relationship not adhering to the rules specified in the prompt

```
Relationship(source=Node(
                id='Timber Frame Extension',
                type='Component'),
             target=Node(
                id='Sand-Lime Bricks',
                type='Material'),
             type='INTERSECTS')
```

Listing 5.1 shows the relation that connects a node of type *Component* to a node of type *Material*. The connection in itself is reasonable and adheres to the information contained in the BCF comment. However, as specified in the graph schema, each relation must specify the origin, the BCF comment node, as its head so that all indexed information is about the BCF comment directly. Therefore, the relation was removed from the graph.

As shown in Figure 5.2, between one and seven nodes with relationships were extracted in relation to one BCF comment. Most of the time, the LLM prompting chain extracted four nodes with relationships for one BCF comment node.

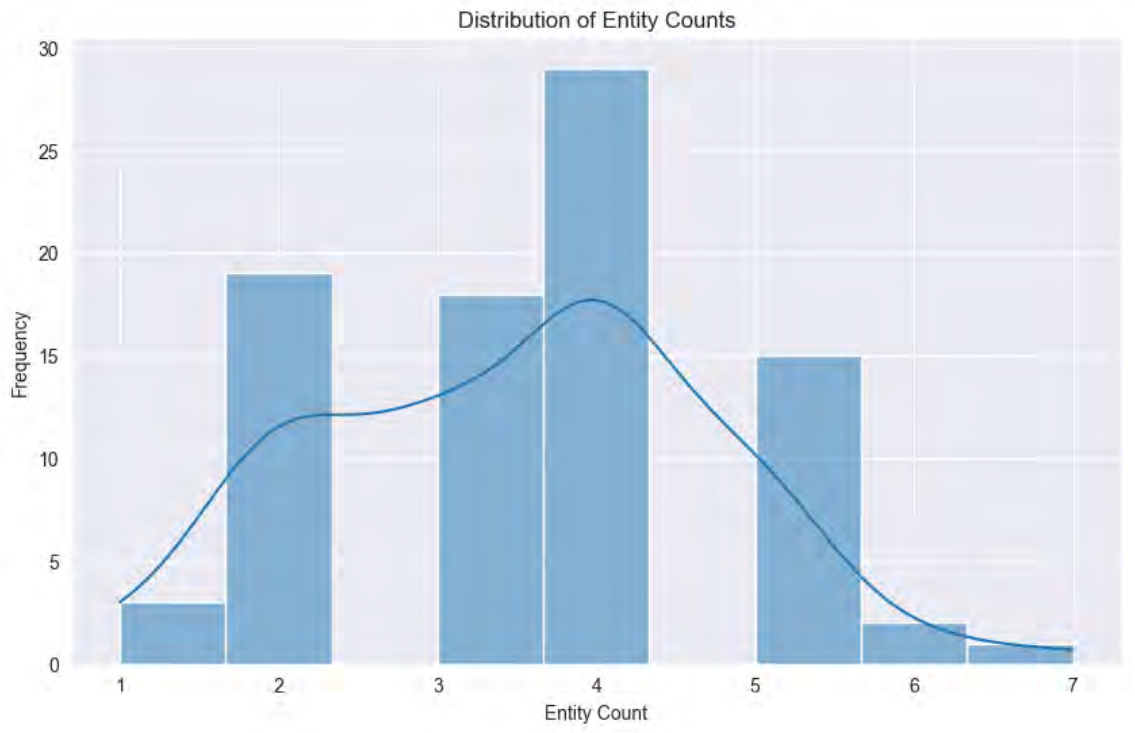


Figure 5.2: Count of nodes with relationships extracted to one BCF comment.

As displayed in Table 5.1a the graph contains 297 notes of 16 different types. Most common are the *Component* nodes, followed by nodes of type *BCFComment*, *RequestType* and *Material*. The LLM was also able to index several node types without specifying the corresponding theme explicitly in the prompt. The various node types display aspects about the BCF comment which the LLM indexed without direct prompting. Table 5.1b shows all the relationship types indexed by the LLM. In total, there are 303 relationships with 12 distinct relation types.

| Node Type | Count |
|---|---|
| Component | 135 |
| BCFComment | 87 |
| RequestType | 39 |
| Material | 10 |
| Specification | 5 |
| Document | 4 |
| Issue | 4 |
| Person | 3 |
| Discipline | 2 |
| Process | 2 |
| Software | 1 |
| Company | 1 |
| Model | 1 |
| Organization | 1 |
| Service | 1 |
| Measurement | 1 |
| **Total** | **297** |

(a) Distribution of node types.

| Relationship Type | Count |
|---|---|
| MENTIONS | 178 |
| REQUESTS | 78 |
| DISCUSSES | 22 |
| SPECIFIES | 11 |
| REFERS_TO | 4 |
| INVOLVES | 3 |
| INDICATES | 2 |
| SUGGESTS | 2 |
| IMPLIES | 2 |
| INQUIRES | 1 |
| ASSUMES | 1 |
| REFERENCES | 1 |
| **Total** | **303** |

(b) Distribution of relationship types.

Table 5.1: Distribution of node types and relationship types in the graph.

**Case Study on the Indexing Theme *Components*** Within my knowledge graph, 135 different components were identified. The most prominent components discussed in the BCF comments were 'Opening' and 'Openings', followed by the 'Floor' and the 'Roof', as shown in Table 5.5. Since there are more nodes of type component than BCF comment nodes, it becomes apparent that frequently the combination of several components is discussed in one comment. Up to a maximum of six components were identified within one BCF comment, as shown in Table 5.2c. However, most BCF comments discuss between one to three components. Figure 5.3 displays two examples of different components being correctly identified and indexed from two different BCF comments.

From this case study of indexing the BCF comments by *Components*, several key points emerged:

1. The proposed LLM framework demonstrated a robust capability to identify and index a diverse range of construction-specific components with high precision. Despite the absence of labelled data and ground truth for this NLP task, preliminary results indicated promising performance in component extraction and classification.

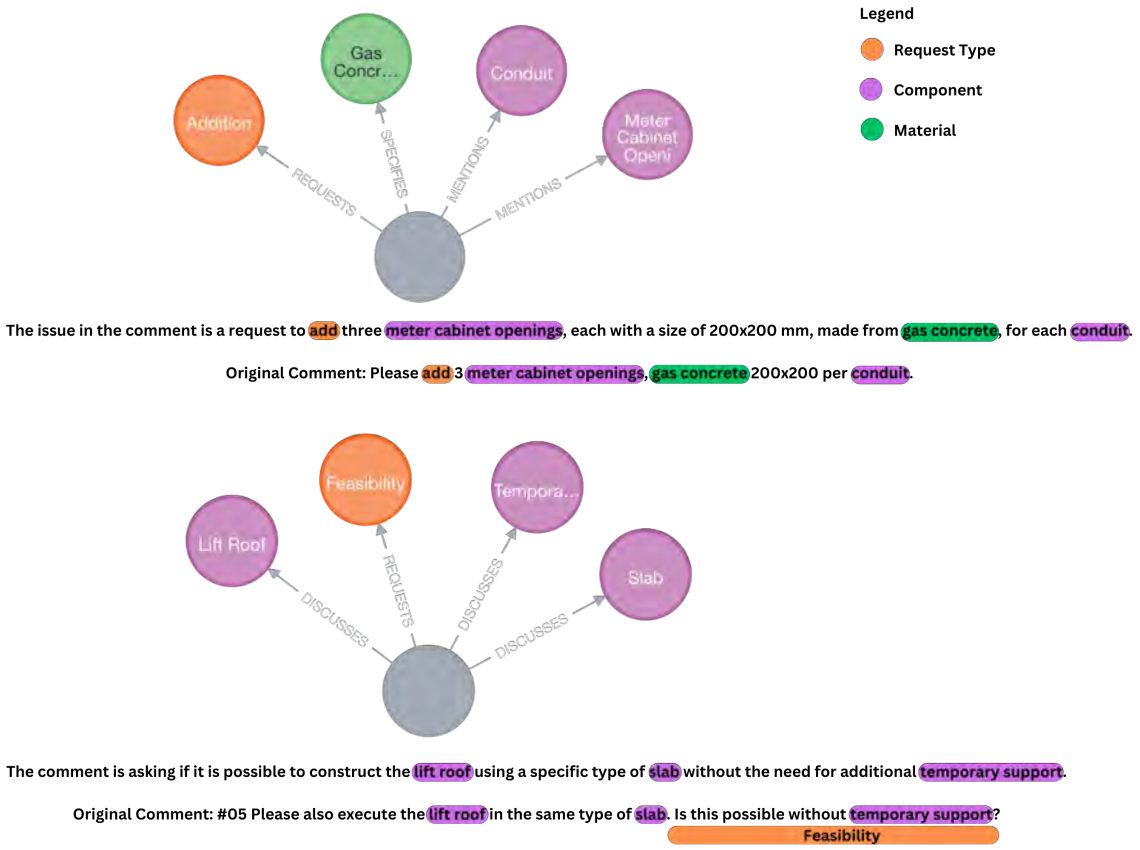


Figure 5.3: Intermediate and end results of the promting chain indexing a BCF comment with three different themes.

2. It becomes apparent that the framework lacks entity resolution. Components in plural and singular are separated into different nodes. An obvious example is 'opening' and 'openings' in my dataset, but there are several more occurrences of this problem like 'Side Wall' and 'Side Walls' and 'Shaft' and 'Shafts'.

3. The framework identified various hierarchical components, such as groupings of 'Floor', 'First Floor', and 'Second Floor'. These groupings only emerged after the different components were identified and cannot be predicted in advance. Without proper grouping, the graph's modularity revealed only major trends for specific components, potentially obscuring higher-level patterns across categories like 'Window' or 'Beam' variants.

4. The lack of a predefined ontology led to ambiguity in the component classification. For instance, elements identified by the LLM framework, such as 'Gap', may not be universally considered as components in traditional construction terminology. Here the implementation of a post-processing filtering mechanism that matches the LLM outputs against a predefined ontology could help to eliminate ambiguous classifications.

5. The proposed LLM framework occasionally indexed elements that are artefacts of the planning process rather than physical components of the construction or building. Examples include 'Drawing', 'IFC File', or 'Axis 3'.

6. In some cases, particularly with BCF comments of limited token size, the framework misinterpreted comment enumerations as component identifiers. This resulted in non-component elements such as '#10 Sparing' being included in the aspect list.

7. While the majority of identified components are plausible, outliers such as 'Issue' and 'Method of Attachment' demonstrate that some aspects classified as *Component* do not represent physical building elements.

8. There are cases of over-specification, where aspects like 'Gutter Location' and 'Selected Columns' should be simplified to 'Gutter' and 'Columns' respectively.

| Material | |
|---|---|
| Number of Nodes | Relation Count |
| 7 | 1 |
| 2 | 2 |

(a) Number of *BCFComment* nodes connected to node/s of type *Material*.

| RequestType | |
|---|---|
| Number of Nodes | Relation Count |
| 61 | 1 |
| 11 | 2 |

(b) Number of *BCFComment* nodes connected to node/s of type *RequestType*.

| Component | |
|---|---|
| Number of Nodes | Relation Count |
| 28 | 1 |
| 28 | 2 |
| 19 | 3 |
| 8 | 4 |
| 1 | 5 |
| 1 | 6 |

(c) Number of *BCFComment* nodes connected to node/s of type *Component*.

Table 5.2: Distribution of *BCFComment* Connections by Node Type.

**Case Study on the Indexing Theme *Material*** The analysis of materials mentioned within the BCF comment dataset revealed a relatively low frequency of material references. My indexing chain identified only ten distinct aspects of type *Material*, with 'Sand-Lime Brick' being the sole material associated with multiple BCF comments. This low number of material mentions suggests that material-specific information is less common in the BCF comments analyzed. This indicates that BCF communications in this dataset focus more on component-level issues rather than material-specific concerns.

From this case study of indexing the BCF comments by *Material*, several key observations emerged:

1. Consistent with earlier findings, my framework exhibited a limitation in entity resolution. Materials in singular and plural forms were categorized as separate entities, as evidenced by the distinct entries for 'Sand-Lime Brick' and 'Sand-Lime Bricks' in Table 5.3.

2. The LLM framework demonstrated the capability to identify and index a range of construction materials. Notably, it successfully recognized brand names such as

'Ytong', which are often used interchangeably with generic material names in the BCF comments. This suggests the framework's potential to capture industry-specific terminology and proprietary product names.

3. Cases of over-specification can be observed here, as well as aspects, like 'Manageable Blocks', should be simplified to 'Blocks'.

4. The classification of certain material-related aspects revealed inconsistencies in type assignment. For instance, 'Backing Wood' was classified as a *Component* rather than a *Material*. Conversely, 'Rock Panel' was identified as a *Material*, while its plural form 'Rock Panels' was categorized as a *Component*. This observation indicates that the LLM framework tends to index certain aspects under a single type category, even when they could potentially be classified under multiple node types.

| **Material Nodes** | | |
|---|---|---|
| **Group** | **Aspects** | **Relation Count** |
| 1 | Sand-Lime Brick | 2 |
| 2 | Sand-Lime Bricks, Gas Concrete, Concrete Cover, Ytong, Manageable Blocks, Chipboard, Limestone, Rock Panel, Sheet | 1 |

Table 5.3: Grouped *Material* aspects by count.

**Case Study on the Indexing Theme *Request Type*** In total, the LLM framework indexed 39 distinct aspects of the type *RequestType*. This indexing process demonstrated my approach in which the LLM not only identifies explicitly mentioned request types within the BCF comments but also infers and assigns request types through contextual reasoning. For instance, as illustrated in Figure 5.3, a comment was indexed with the *RequestType* 'Feasibility', despite the absence of this specific term in the text. This showcases the LLM's capability to perform semantic analysis beyond keyword extraction.

Table 5.4 shows that a high number of BCF comments are associated with the aspects *addition*, *extension*, and *move*, indicating prevalent themes in the dataset. The framework allowed for multiple *RequestType* aspects to be assigned to a single BCF comment, as demonstrated in Figure 5.4 and proven in Table 5.2b.

From my case study of indexing the BCF comment by *RequestType*, several key points emerged:

1. Despite the relatively small dataset of BCFs, the LLM effectively identified common request type aspects that recur across various BCF comments. The natural

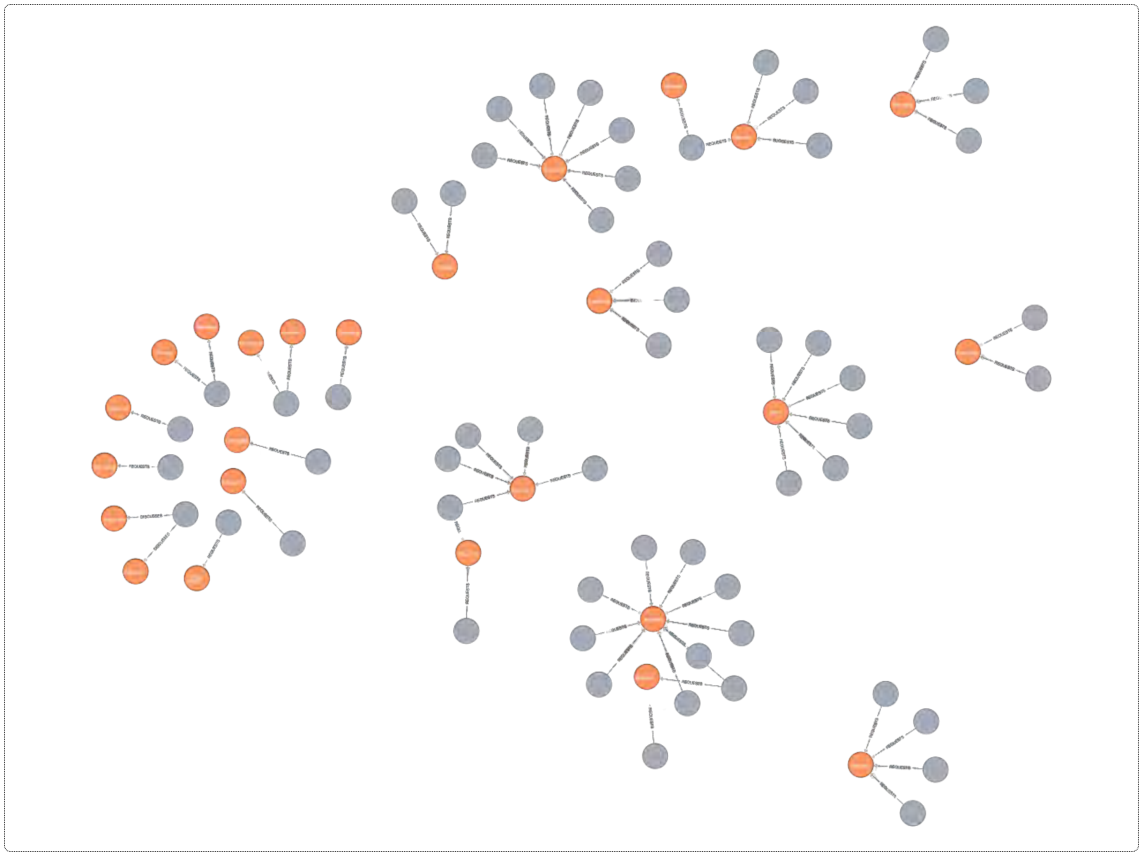

Figure 5.4: Request Type Graph.

modularity of the graph structure facilitated the identification of these common aspects, allowing efficient indexing and exploration of different themes within the BCF comments.

2. Many *RequestType* aspects occurred only once in the graph, failing to create connections between multiple BCF comments. This suggests a long tail of unique or highly specific request types in the dataset. The lack of pre-specified request type aspects led to a wide range of classifications. Some *RequestType* aspects are highly specific and unlikely to create multiple connections. The current graph structure did not capture hierarchical relationships between aspects (e.g. when one aspect is a subgroup of another).

3. Not all BCF comment nodes were linked to a *RequestType* aspect, indicating potential gaps in classification coverage.

4. While the relationship labels contributed significantly to the graph's readability, they sometimes lacked semantic depth, adding limited detail or meaning.

**Case study on Indexing without a Theme** Within my LLM framework, I also allowed the LLM to index notes and declare node types which are not specified by any theme. As shown in Table 5.1a a total of 26 node instances and relationships were indexed in this way, with 12 distinct node types.

From this case study of indexing the BCF comment without specifying a theme, several key points emerged:

1. Similar node types like *Organization* and *Company* were indexed separately, resulting in duplicate or similar types in the graph.
2. Some node types only declare one instance within the graph. Node types like *Software* and *Company* might be reasonable indices but based on the small size of my dataset, it is unclear if these types are usable for analysis or retrieval on a larger scale.
3. The LLM framework detected and indexed several novel node types with two to five instances within the graph. Thereby, it explored node types which could reasonably be included in further analysis. An example is the node type *Document*, under which several refereed documents mentioned in the BCF comments were indexed. However, some identified node types, such as *Specification*, seem unreasonable since they only specify different concrete measurement nodes mentioned in the BCF comments, like '60mm' and '300mm'.

| **Request Type Nodes** | | |
|---|---|---|
| **Group** | **Aspects** | **Relation Count** |
| 1 | Addition | 10 |
| 2 | Extension | 7 |
| 3 | Move | 6 |
| 4 | Replacement<br>Shortening | 5 |
| 5 | Adjustment<br>Request | 4 |
| 6 | Modeling<br>Feasibility<br>Relocation | 3 |
| 7 | Removal<br>Reduction<br>Shorten<br>Modification | 2 |
| 8 | Slope Planning, Placement, Alignment, Correction, Beveling, Change, Setback, Centering, Adherence, Movement, Question, Narrowing, Installation, Support, Moving, Enlargement, Rebate, Incorporation, Cutout, Forgotten Or Overlooked, Revised Layout, Coordination, Clash, Omission, Continuation | 1 |
| **Total** | | **83** |

Table 5.4: Grouped *RequestType* aspects by count.

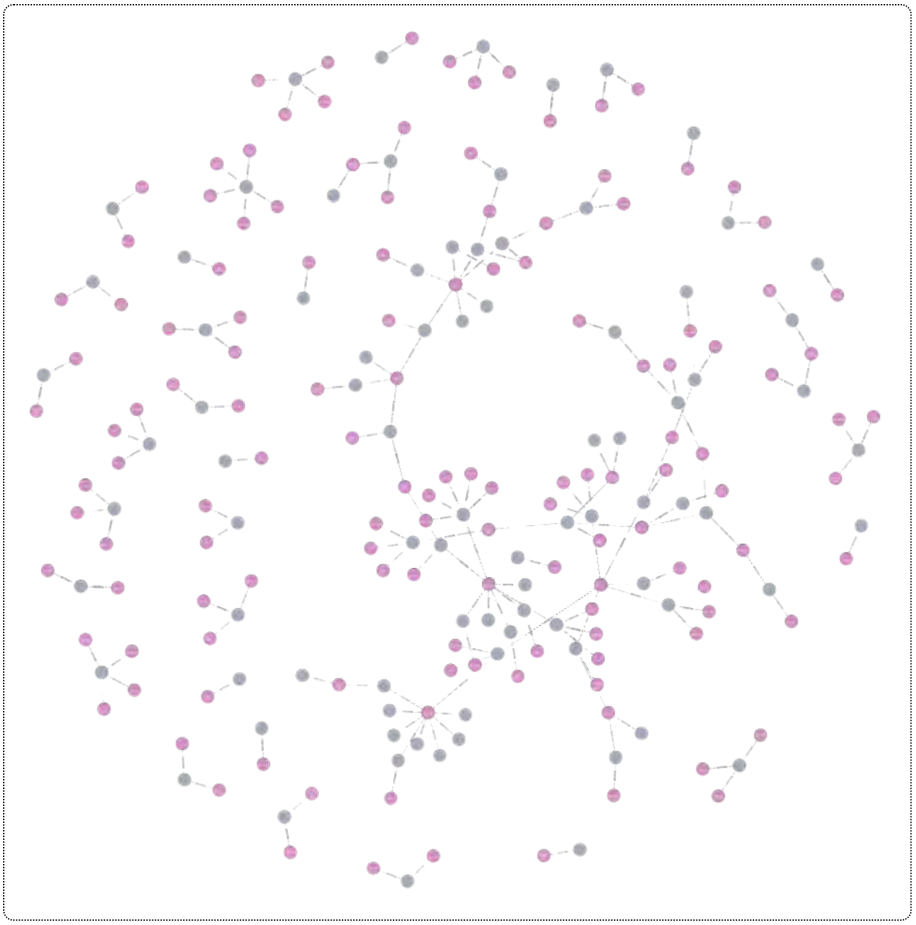

Figure 5.5: Component Type Graph.

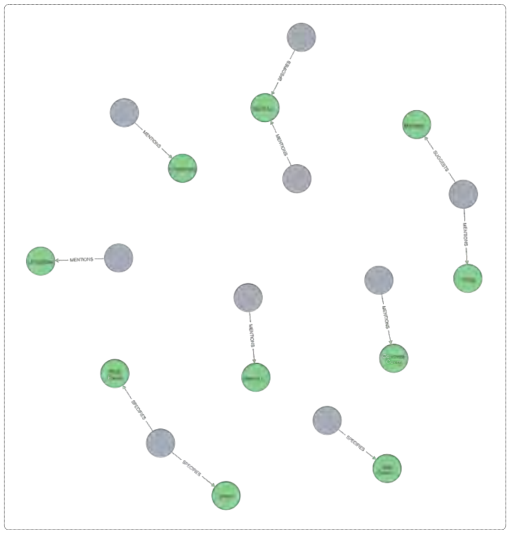

Figure 5.6: Material Type Graph.

| Component Nodes | | |
|---|---|---|
| **Group** | **Aspects** | **Relation Count** |
| 1 | Opening | 9 |
| 2 | Openings | 9 |
| 3 | Floor | 7 |
| 4 | Roof | 6 |
| 5 | Beam | 4 |
| 6 | Fencing, Railings, Window Frames | 3 |
| 7 | Cover Strips, Embedded Provisions, Cavity Batten, Brickwork, Balconies, Bay Window, Support, Overhang, Base Model, Wire Sleeves, Window Frame, Steel Beam, Cavity Side | 2 |
| 8 | M16 Nuts, Timber Frame Extension, Drawing, Slope, Wedges, Shafts, Dormer Windows, Side Wall, Roof Tiles, Issue, Green Panels, Rock Panels, Side Walls, Steel Angle Lines, #10 Sparing, Rainwater Drainage Pipes, Shaft, Sewer Ventilation, Meter Cabinet Openings, Conduit, Method Of Attachment, Ridge, Cover Strip, Roof Structure, Gap, Outlet, Rainwater Drainage System, Sides, Gutter Location, He200B, He280B, White Panels, Parapet, Finishing, Wall, Finishing Plate, Insulation, Stair Landings, First Floor, Second Floor, 190Mm Bearing, Component #13, Beams, Red Lines, Plates, Screws, Utilities, Inner Cladding, Skylights, Basic Model, Pipe-In-Pipe Connection, Angle Profile, Floor Support, Wide Slab, #08 Sparing, Ytong Walls, U-Profile, Backing Wood, Selected Columns, Through-Holes, Stair Railings, Handrails, Top Bottle, Flange, Concrete Beam, Window Opening, Gutter Construction, Allowances, Bracket, L150/18 Angle Profile, Cantilevered Slab, Front Facade, Pluvetta, Bend, Balcony, Facade Support, #04 Nuts, Lift Roof, Slab, Temporary Support, Rafters, Holes, Provisions, Fek, Emergency Drainage Outlet, Gutter, Window Frame Sill, Meter Cabinets, Facade Carriers, Flat Roof, Ifc File, Onderleggers, Spare Part, Plot 1, Transom Windows, Threaded Ends, Bay Windows, Dimension, Sloped Roof, Gutter Racks, Corner, Steel Strip, Red Walls, Bolts, Steel Beams, Strips, Axis 3, Crawl Space Hatches, Hollow Core Slabs, Bands, Masonry, Hollow Core Slab, Inner Wall, Red Markings | 1 |
| **Total** | | **83** |

Table 5.5: Grouped *Component* aspects by count.

**Summary** Overall, the framework was able to create a knowledge graph which not only represented the content contained in the BCF comments, but it also contained abstractions or classifications of the BCF comments that provide additional metadata about the BCF comment in a structured format. This structured format within the graph database provides a good starting point for further evaluation on a larger scale. The framework demonstrated promising capabilities in semantic analysis and flexible categorization. Moreover, indexing without themes also provides an opportunity to identify novel themes that can be explicitly included in further analysis. However, there are clear opportunities for improvement, which will be discussed in the following chapter.

## 5.2 Agentic Retrieval and Evaluation

For agentic graph retrieval, I conducted several case studies. First, I will introduce the general setup of the questions posed to the knowledge graph within the case study. Then, I will describe the results of the different retrieval tools used to answer these questions based on the knowledge contained within the graph.

### 5.2.1 Case Study Questions

Each question asked from the knowledge graph, focuses on the issues and contents of several BCF comments, aiming to identify common occurrences of themes and topics. Due to the graph schema, trends can be identified through the number of connections a certain node instance has towards BCF comment nodes. By querying which node of a certain type has the most connections towards BCF comment nodes, recurring discussion topics can be identified. The graph contains the relevant knowledge to answer these questions, as corresponding themes were defined before the indexing of the dataset. The following questions were addressed in the case studies:

1. Which different node types are present in the graph?

   This question aims to retrieve all different node types from the graph.

2. Which *component* was most discussed in the BCF comments?

   This question asks for the instance of a certain node type, in this case 'Component', which has the most relations towards a BCF comment nodes, therefore being frequently mentioned within the BCF comments. The same question can be asked for other themes and node types, like 'Material' or 'Request Type'.

3. How many BCF comments with *Relocations* did we have?

   This question asks about a specific node instance with the aspect 'Relocation'. It requests a relation count of connections to BCF comment nodes. In this case for the query, the node type does not matter and the question can be directed to every aspect existing within the graph. 'Relocation' is a 'Request Type'.

4. How many issues did include a *Move* and what were they about?

   This question is a variation of the previous one. 'Issue' is interchangeably used and 'Move' is a 'Request Type'.

5. How many requests about *openings* did we have?

   This question is a variation of the previous question as well. 'Request' is interchangeably used and 'openings' is of type 'Component'.

6. Which issues did we have with *Lime Sandstone*?

   This question asks for the content of BCF comments in relation to a node with the aspect 'Lime Sandstone' which is of type material.

### 5.2.2 Case Study Evaluation

Each case study question was posed at the LLM agent with custom-implemented tools and at the agent with the ability to write Cypher.

1. Which different node types are present in the graph?

After the question was invoked to the LLM agent with custom tools, the agent called a tool to retrieve all the node types from the graph. It then used the result of the tool to generate its answer.

```
> Entering new AgentExecutor chain...
Invoking: `Get_All_Node_Types` with `{}`
Node types present in the graph:
1. BCF          2. Company      3. Component    4. Discipline
5. Document     6. Issue        7. Material     8. Measurement
9. Model        10. Organization 11. Person     12. Process
13. Requesttype 14. Service     15. Software    16. Specification
The graph contains the following node types:
1. BCF          2. Company      3. Component    4. Discipline
5. Document     6. Issue        7. Material     8. Measurement
9. Model        10. Organization 11. Person     12. Process
13. Requesttype 14. Service     15. Software    16. Specification
> Finished chain.
```

In comparison, the Cypher agent generated a correct Cypher query and retrieved the right results from the graph. The context is then used to generate an answer as a text.

```
> Entering new GraphCypherQAChain chain...
Generated Cypher:
MATCH (n) RETURN DISTINCT labels(n)
Full Context:
[{'labels(n)': ['BCF']}, {'labels(n)': ['Component']}, {'labels(n)': ['
   Requesttype']}, {'labels(n)': ['Material']},
 {'labels(n)': ['Discipline']}, {'labels(n)': ['Software']}, {'labels(n
    )': ['Document']}, {'labels(n)': ['Specification']},
 {'labels(n)': ['Person']}, {'labels(n)': ['Company']}, {'labels(n)':
    ['Process']}, {'labels(n)': ['Issue']},
 {'labels(n)': ['Model']}, {'labels(n)': ['Organization']}, {'labels(n)
    ': ['Service']}, {'labels(n)': ['Measurement']}]
Result:
The graph contains the following node types:
- BCF          - Component     - Requesttype   - Material
- Discipline   - Software      - Document      - Specification
- Person       - Company       - Process       - Issue
- Model        - Organization  - Service       - Measurement
> Finished chain.
```

The next question was used to test if the agents can also count relationships within the graph:

2. Which component was most discussed in the BCF comments?

The custom agent was able to call the according tool for this question and generate its answer based on the context of the tool. It kept the distinction between, entities in singular and plural.

```
> Entering new AgentExecutor chain...
Invoking: `Major_Themes_Finder` with !*{'node_type': 'Component'
Top 3 Components with the most relations to BCF comments:
1. Opening: 9 connections    2. Openings: 9 connections
3. Floor: 7 connections
The most discussed components in the BCF comments are:
1. Opening (9 connections)    2. Openings (9 connections)
3. Floor (7 connections)
It appears that "Opening" and "Openings" are the most frequently
   mentioned components.
> Finished chain.
```

When adding the prompt 'Please combine the count of similar nodes', to the question the LLM was able to resolute and combine the entities and their count, although no calculator tool was leveraged.

```
{... Same chain as before ...}
The most discussed component in the BCF comments, when combining
   similar nodes, is "Opening/Openings" with a total of 18 connections
    (9 for "Opening" and 9 for "Openings"). The next most discussed
   component is "Floor" with 7 connections.
> Finished chain.
```

The Cypher agent could understand the question, given the context of the graph schema, and query the graph for the right relationship. However, it did interpret the posed question differently. Since all relationships within the graph were labelled, the Cypher agent only queried for the relationships with the type 'discusses', which in turn yielded a different result. When prompted to consider all relationship types, the Cypher agent retrieved a similar result as the custom agent. However, since the Cypher agent did not leverage pre-coded tools, but generated the Cypher query on demand, the result of the tool call just included one node 'Opening', and not the three nodes with the highest relation count, as returned by the custom tool. The Cypher agent limited the retrieval to one, not considering the possibility of several nodes with the same relation count.

```
> Entering new GraphCypherQAChain chain...
Generated Cypher:
cypher
MATCH (b:BCF)-[:DISCUSSES]->(c:Component)
RETURN c.name, COUNT(*) AS discussionCount
```

```
ORDER BY discussionCount DESC
LIMIT 1

Full Context:
[{'c.name': 'Openings', 'discussionCount': 2}]
Result:
The most discussed component in the BCF comments was "Openings," with 2
    discussions.
> Finished chain.
```

The next question was used to test, if the agent can query the graph for certain nodes of interest and evaluate their importance and therefore relation count within the knowledge graph.

3. How many BCF comments with *Relocations* did we have?

The custom agent called the BCF Aspect Analyzer Tool, which recognized two entities within the question: 'BCF comments' and 'Relocations'. It then searched the graph for both aspects using the full-text index. Through the index, it was able to retrieve the node with the name 'Relocation' from the graph in combination with the connected BCF comments and calculate the corresponding count, deriving the correct answer.

```
> Entering new AgentExecutor chain...
Invoking: `BCF_Aspect_Analyzer` with `{'question': 'How many BCF
    comments with Relocations did we have?'}`
Queried graph for entities related to: 'How many BCF comments with
    Relocations did we have?'
Entity 1: BCF comments
======================
Total 'BCF comments' relations: 0
Comments:
----------------------------------------
Entity 2: Relocations
=====================
Total 'Relocations' relations: 3
Comments:
{...}
Prompt for information reuse: Based on the above comments, answer
    concisely: How many BCF comments with Relocations did we have? Use
    only comment information. Note any information gaps.
We had 3 BCF comments with Relocations.
> Finished chain.
```

In comparison, the Cypher agent was not able to retrieve any results, since, it restricted its query through a relationship type. Moreover, since it did not use the full-text index, it did only search for 'Relocations' not finding any matching with 'Relocation'. Only after reformulating the question for 'Relocation' and prompting to consider all relation types the same results as with the custom agent were achieved.

```
> Entering new GraphCypherQAChain chain...
Generated Cypher:
cypher
MATCH (b:BCF)-[:MENTIONS]->(r:Requesttype {name: "Relocation"})
RETURN COUNT(b) AS RelocationBCFCount

Full Context:
[{'RelocationBCFCount': 0}]
Result:
There were no BCF comments with a Relocation.
> Finished chain.
```

The last three questions were similar to question 3 as they also retrieved a node by name from the graph and counted its relations to BCF comments. However, they had slight variations in wording and retrieved different node types.

4. How many issues did include a *Move* and what were they about?
5. How many request about *openings* did we have?
6. Which issues did we have with *Lime Sandstone*?

The custom agent was able to answer all questions according to the knowledge contained in the graph. Moreover, the BCF Aspect Analyzer Tool was also able to allow minor errors in spelling and matched entities in singular and plural. Therefore, question number five about openings was answered with 18 openings, combining the count of 'opening' and 'openings'.

The Cypher writing agent, in comparison, showed several shortcomings. Within the graph, it searched for complex graph patterns, related to the structure of the question. Therefore, the Cypher agent searched for patterns that were too complicated, as it had no explanation of the overall graph schema.

```
Generated Cypher:
cypher
MATCH (b:BCF)-[:DISCUSSES]->(i:Issue), (b)-[:DISCUSSES]->(r:Requesttype
    {name: "Move"})
RETURN COUNT(i) AS numberOfIssues, COLLECT(i.id) AS issueIds

Full Context:
[{'numberOfIssues': 0, 'issueIds': []}]
Result:
There were no issues that included a Move.
> Finished chain.
```

Especially, the node type *Issue*, which is contained within the graph and contained as a word within the question, caused the Cypher agent to search for complex patterns not

applying to the definition behind the graph schema. Moreover, as already apparent in earlier questions, the Cypher agent matches node aspects directly, ignoring capitalization and plural forms.

In summary, the custom agent was superior to the Cypher agent in the limited set of questions. However, it is important to note that the custom agent was provided with tools specifically designed to answer a similar question set. Therefore, the main task of the custom agent was to select the right tool and generate a matching answer based on the result of the tool. The tasks required by the Cyper agent are much more complex as it had to generate Cypher queries matching the question and the knowledge graph schema. All generated Cypher statements were syntactically correct, as well as the generated answers based on the information retrieved from the graph were satisfying.

# Chapter 6

# Discussion

The application of LLMs and KGs to enhance project knowledge management in construction presents a novel approach to addressing the long-standing challenges of knowledge retrieval, organization, and analysis in the industry. This thesis tested the semantic understanding capabilities of LLM and leveraged them to build a KG from unstructured conversational text data, created during a construction project. The proposed framework enabled detailed post-project analysis on a scale prior unfeasible with manual work. Furthermore, by creating a comprehensive and enriched KG, the framework provided a robust basis for project analysis.

The results of this work showed that, based on the KG, it is possible to identify components frequently discussed throughout the project files. Moreover, it was shown that trends like the most frequent discussion of components in comparison to materials, which was a far less discussed topic, can be identified and evaluated. Based on insights like these, stakeholders can train their employees on topics identified through post-project analysis provided by my framework.

The natural modularity of the graph provided an excellent data structure to identify trends within the dataset. LLMs, when prompted accordingly, leveraging their few-shot learning abilities, were able to construct this KG. The results of this work also included case studies of different LLM based agentic retrieval methods. These methods showed great potential for graph retrieval and analysis, being able to answer a variety of questions posed to the used dataset through natural language.

In the following sections 6.1, I will critically discuss the outcomes of my graph indexing process, examining its strengths and limitations. I will go into the effectiveness of my theme-based indexing approach. Additionally, I will discuss the insights gained from allowing the LLM to index information without predefined themes. Next, the strengths and limitations of the agentic retrieval process will be highlighted (Section 6.2). Afterwards, this discussion will focus on the scalability of my framework, and contextualize the findings (Section 6.3).

## 6.1 Discussion of Graph Indexing Process

In summary, the LLM based framework demonstrated promising capabilities in transforming unstructured BCF comments into a structured, queryable knowledge graph. The approach showed advanced inference abilities beyond simple keyword extraction, including classification and adherence to a predefined schema. The rich diversity of extracted

information and high adherence to the specified structure further underscore the method's value.

Moreover, it is also important to acknowledge several limitations, including the lack of entity resolution during indexing, the absence of hierarchical grouping, and ambiguities due to the lack of a predefined ontology. These limitations and potential improvements will be explored in the following paragraphs to fully discuss the potential of this method for construction knowledge management.

**Adherence to Graph Structure** One notable success of the approach was the LLM's ability to adhere to the specified graph structure with high consistency. Out of all the relationships extracted, only one did not conform to the rules specified in the prompt, which could be automatically excluded. This high level of adherence showed that LLMs can be effectively guided to produce structured outputs, which is crucial for creating consistent and analyzable knowledge graphs.

**Diversity and Inference Capabilities** The identification of 16 different node types and 12 distinct relationship types indicated a rich and diverse extraction of information from the BCF comments. This diversity showed that the approach can capture a wide range of concepts and relationships within construction project communications, offering a more comprehensive view of project knowledge. Moreover, the LLM demonstrated the ability to infer request types even when not explicitly stated in the text. This capability showed that the LLM can perform semantic analysis beyond simple keyword extraction, uncovering implicit information in BCF comments by classifying them. To further enhance the inference capabilities, the prompting change could be split into different subtasks, spending more computing on indexing different aspects separately.

**Relationship Labeling** The variety of relationship types extracted (e.g., 'mentions', 'requests', 'discusses') provides semantic context to the connections between nodes, which enhances the readability of the graph. However, upon closer examination, these relationship labels sometimes leave room for semantic depth, adding limited detail or meaning to the connections they represent.

This observation suggests an opportunity for enhancing the quality and informativeness of these labels in future iterations of the framework. One potential improvement is to differentiate between extraction and classification in the relationships. For example, a relationship can be labelled differently when a request type is directly mentioned in the text, as when the LLM infers that something is a specific request type based on context. Another approach can involve adding parameters to the relationships to provide more detailed information. Implementing such enhancements improves the graph's ability for downstream analysis processes.

However, as shown during agent retrieval, it is important to balance the desire for more detailed relationships with the need to maintain simplicity and clarity in the graph structure.

Since the relationship types are not resoluted during indexing time, they can be misleading to agents and users during the analysis of the graph, as shown by the results.

**Emergence of Node Types and Ontology Learning** The frequency of certain node types, such as 'Component', 'RequestType', and 'Material', aligned well with the intended focus themes, which were chosen in advance as an area of interest for dataset investigation. Moreover, the emergence of additional node types like 'Document', and 'Person' suggests that the LLM framework is capable of identifying and categorizing information beyond the explicitly defined themes. This capability demonstrates a form of ontology learning, where the LLM infers and generates new categories and relationships from the data without explicit pre-definition.

This emergent ontology learning has several important implications for knowledge representation. It enables the discovery of unforeseen patterns or topics within project communications that might have been overlooked in a more rigid, predefined categorization system. The approach allows for a more adaptive and comprehensive representation of project knowledge, potentially capturing nuances specific to individual projects or organizations.

Therefore, the methodology and structure of the knowledge graph facilitate both focused analyses on specific themes and broader exploration of interconnections between different aspects of project communications. As more data is processed, the ontology of the graph can potentially evolve, reflecting changing project dynamics or industry trends over time.

However, this flexibility also presents challenges in maintaining consistency and managing the growth of the ontology, especially as the dataset expands. As already demonstrated through the results, some node types can be duplications or contain misleading node instances. The absence of pre-specified node types and aspects leads to a widespread of both. Some node types and aspects can become highly specific and are unlikely to create multiple connections.

Furthermore, this flexibility in ontology discovery potentially limits comprehensive retrieval and evaluation, as not all relevant BCF comments for a certain aspect might be identified. This highlights a trade-off between classification flexibility and completeness.

**Entity Resolution and Hierarchical Relationship Challenges** A significant limitation observed was the lack of entity resolution, particularly evident in the separate indexing of singular and plural forms (e.g., 'Opening' and 'Openings', 'Side Wall' and 'Side Walls') and in over specifications (e.g. 'Gutter Location' instead of simply 'Gutter'). This issue could lead to fragmentation of information and hinder the identification of trends related to specific node aspects. During graph retrieval, I addressed this limitation through an indexed matching process. However, the refinement of the LLM framework, by adding a module for entity resolution, can enhance the graph's utility for analysis, without the need for entity resolution during retrieval. The same holds true for hierarchical relationships hindering the identification of trends.

## 6.2 Discussion of the Agentic Graph Retrieval Process

Overall, this thesis demonstrated that a LLM-based agent can provide intuitive ways to communicate with the information contained within a knowledge graph. Through the implementation of a custom tool set, an agentic system can extend the original functionality of a LLM to deterministically retrieve information from the graph using pre-programmed tools. This approach bridged the gap between natural language queries and structured data retrieval, making knowledge graphs more accessible to users without expertise in query languages like Cypher.

The agentic graph retrieval process demonstrated in this thesis offers several advantages over traditional query methods. Firstly, it allows for more natural and conversational interactions with complex data structures. Secondly, it reduces the technical barriers to entry for users seeking to extract information from knowledge graphs. Lastly, the deterministic nature of the tool-based approach ensures consistency and reliability in the retrieval process.

The performance of the proposed agents strongly relied on the capabilities of their tools and can therefore be enabling in certain areas while being limiting in others. As evident from the results of this work, the agent can answer questions with high accuracy when provided with the appropriate tools. To address the limitations of tools, a Cypher-generating agent was tested. This agent showed promising capabilities in overcoming the limitations of custom tools. However, due to the requirement of capabilities in a wide area of questions towards the knowledge graph, accuracy, and performance decreased.

In the following two paragraphs, I will discuss the limitations of the tested retrieval methods, including both the tool-based approach and the Cypher writing agent.

**Entity Resolution through Agent Tools** The agent equipped with custom tools demonstrated the ability to answer specific questions based on information contained within the knowledge graph. However, its entity resolution capabilities remain limited, primarily due to constraints in the BCF Aspect Analyzing Tool. While the tool's full-text index can match queried words with entities in the graph, this approach could be significantly enhanced by incorporating an embeddings-based matching system, which would allow for improved semantic matching.

Furthermore, the current implementation of the tool's entity extraction chain, while effective for a range of predefined questions, restricts the flexibility and diversity of queries that can be effectively processed. Although the results indicate that extracting entities from questions and matching them with the knowledge graph can successfully answer various queries, this method lacks the generalizability required for comprehensive knowledge graph retrieval. Nevertheless, this approach has demonstrated the potential for natural language interaction with knowledge graphs, paving the way for more sophisticated implementations in the future.

**Cypher Generating Agent** By providing the LLM (GPT-4o) utilized within the Cypher agent with the schema of the knowledge graph, consistent generation of syntactically correct Cypher queries was achieved. The LLM successfully generated Cypher queries based on the entities and relations stated in the input questions, adhering to the provided graph schema. Although this approach was only tested experimentally within the scope of this thesis, it demonstrates significant potential for simplifying knowledge graph retrieval for non-technical users.

There are promising avenues for further improvement through advanced prompt engineering techniques. Specifically, incorporating an explanation of the reasoning behind the knowledge graph's schema into the system prompt can enhance the LLM's understanding and query generation capabilities.

Moreover, the studies showed that it is important to keep a simple schema of the graph. Therefore, removing relationship types or note types and instances with minor occurrences can improve the accuracy of the Cypher generation.

## 6.3 Scalability and Consistency

The LLM framework successfully processed and indexed a dataset of BCF comments, generating a graph with 297 nodes and 305 relationships. While this dataset is relatively small in the context of large-scale graph databases, it represents a feasible volume of project communication data for the testing of my approach. The framework's ability to handle this dataset suggests a potential for scalability, though further testing with larger datasets is necessary to confirm this. The graph structure, with its ratio of nodes and relationships per comment, indicates a rich representation of the underlying data, capturing multiple aspects within each comment.

The potential for scalability in this framework is further enhanced by the graph's inherent extensibility. As new data becomes available or project analysis requirements evolve, the graph can be dynamically enriched and expanded. For example, new nodes can be added to represent emerging themes in the project. This open-ended structure provides a flexible framework for representing the project knowledge contained in conversational data. Moreover, with increasing complexity and the need to search for more complex graph patterns to identify trends, the advantages of a graph database come more and more into play.

However, as the dataset expands, managing the growing complexity becomes a critical consideration. It becomes necessary to focus on specific themes to prevent information overload, especially for agentic retrieval. To enhance the framework's utility, a more focused approach to graph construction and analysis is necessary. The current exploratory indexing method, while flexible, can limit certain analysis approaches. A key limitation of the current framework is that it does not mandate the assignment of node types to every BCF comment. While this provides flexibility during the exploratory phase, it can

hinder comprehensive retrieval and evaluation in larger datasets. Specifically, relevant BCF comments for a particular aspect may be overlooked if they lack proper classification.

Therefore I propose a two-phase approach for future implementations:

1. Exploratory Phase: Initially focus on a subset of the dataset to identify and explore different themes and topics. This allows for a broad understanding of the data's structure and content.
2. Structured Phase: Based on insights from the exploratory phase, construct a more focused graph with a predefined set of themes and include the option for mandatory classification. This approach enables more accurate analysis.

# Chapter 7

# Conclusion

This thesis proposed a framework, leveraging the abilities of LLMs and KGs, to automate construction data analysis and thereby contribute to the area of knowledge management. The research demonstrated a novel approach to addressing long-standing challenges of knowledge retrieval, organization, and analysis within the construction sector. The proposed framework successfully transformed unstructured project communication data into a structured, queryable KG, enabling detailed post-project analysis on a scale previously unfeasible with manual work. The natural modularity of the graph structure proved highly effective in identifying trends within the dataset. Moreover, the graph's inherent extensibility lays a strong foundation for the scalability of the framework. The LLM within the framework demonstrated advanced inference abilities beyond simple keyword extraction, including classification and adherence to a predefined schema. This capability allowed for the capture of both explicit and implicit information from communication.
Agentic retrieval methods showed great potential for intuitive graph querying and analysis, enabling users to interact with the KG through natural language queries. The framework's flexibility in ontology discovery allowed for the identification of unforeseen patterns or topics within project communications, potentially capturing nuances specific to individual projects or organizations.

This research primarily leveraged a BCF communication dataset due to its open-source availability and structured, parsable format. While this choice provided a solid starting point, it also highlighted the need for more diverse and comprehensive datasets in future research. The limited availability of additional communication data constrained further testing of the framework. Consequently, due to the absence of a comprehensive labelled dataset, this thesis focused on qualitative analysis based on a small dataset of BCF communication data, rather than providing quantitative benchmarking.
Despite these limitations, this study lays a promising foundation for future research in applying LLMs and KGs to construction project communication analysis. The developed framework demonstrates the potential to transform unstructured project data into valuable, queryable knowledge, opening new avenues for project analysis and knowledge transfer in the construction industry.

# Chapter 8

# Outlook

The framework developed in this thesis presents numerous opportunities for extension and application in future research. As the construction industry continues to digitize and generate more diverse data, several promising directions emerge for further development and refinement of this approach.

**Expanding Data Sources:** Future iterations of the framework could incorporate a wider range of data sources beyond BCF files. Meeting protocols and minutes often contain richer information about project decisions and issues. Integrating these sources could provide a more comprehensive view of project knowledge and decision-making processes. Additionally, exploring the integration of data from BIM models could further enrich the knowledge graph with technical and spatial information.

**Real-time Analysis:** Adapting the framework for real-time analysis of ongoing projects represents a significant opportunity. By processing communication data as it is generated, the system could provide valuable insights during the construction process, not just in post-project review. This real-time capability could enable proactive decision-making and issue resolution, potentially improving project outcomes and efficiency.

**Risk Analysis Enhancement:** With enhanced classifications and real-time processing capabilities, this framework could be adapted for dynamic risk analysis during projects. By monitoring communication data and identifying patterns associated with potential issues, the system could alert project managers to emerging risks before they escalate. This proactive approach to risk management could significantly reduce project delays and cost overruns.

**Cross-Project Learning:** Future research could explore methods for aggregating knowledge across multiple projects. This could lead to the development of company-wide knowledge bases, facilitating learning and best practice sharing across different construction projects.

**Human-AI Collaboration:** Investigating how human experts can most effectively interact with and augment AI-driven analysis will be an important area of study. This could involve developing more sophisticated user interfaces and explaining AI decisions to continuously improve the system's performance.

# Appendix A

# Developed Source Code

This appendix provides an overview of the code files developed as part of this thesis. The code files consists of a series of Jupyter notebooks designed to analyze BCF files. A detailed README for setup and execution also exists within the files.

The project is structured into four main notebooks, each focusing on a specific stage of the data processing and analysis pipeline:

1. **BCF Analysis (01_BcfAnalysis.ipynb):** This notebook is dedicated to the initial analysis of the BCF files stored in the designated BCF folder.
2. **Preprocessing and AI Prompt (02_PreProcessing.ipynb):** This notebook handles the parsing and flattening of BCF files, transforming them into a list of dictionaries for each comment. It also generates AI prompts to explain the issues within each comment clearly and concisely in English. The processed comments are then run through a LLM, with the results saved as JSON files.
3. **Graph Creation and Analysis (03_GraphCreationAnalysis.ipynb):** This notebook focuses on creating a graph from the previously explained BCF comments. It then conducts an analysis of this graph to extract insights about the BCF and relationships identified within the data.
4. **Agented Retrieval (04_AgenticRetrieval.ipynb):** The final notebook in the series is designed for retrieving information from the graph database using agent-based approaches.

Together, these notebooks create a framework for processing, analyzing, and extracting valuable insights from BCF data, forming the core of the research conducted in this thesis.

# Bibliography


Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. (2023). Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Al Qady, M., & Kandil, A. (2010). Concept relation extraction from construction documents using natural language processing. *Journal of construction engineering and management*, *136*(3), 294–302.

Al-Aswadi, F. N., Chan, H. Y., & Gan, K. H. (2020). Automatic ontology construction from text: A review from shallow to deep learning trend. *Artificial Intelligence Review*, *53*(6), 3901–3928.

Amatriain, X. (2024). Prompt design and engineering: Introduction and advanced methods. *arXiv preprint arXiv:2401.14423*.

Anil, C., Durmus, E., Sharma, M., Benton, J., Kundu, S., Batson, J., Rimsky, N., Tong, M., Mu, J., Ford, D., et al. (2024). Many-shot jailbreaking. *Anthropic, April*.

Anshari, M., & Hamdan, M. (2022). Understanding knowledge management and upskilling in fourth industrial revolution: Transformational shift and seci model. *VINE Journal of Information and Knowledge Management Systems*, *52*(3), 373–393.

ArangoDB GmbH. (2024). Arangodb documentation [Accessed: 30.08.2024].

Babaei Giglou, H., D'Souza, J., & Auer, S. (2023). Llms4ol: Large language models for ontology learning. *International Semantic Web Conference*, 408–427.

Bikeyev, A. (2023). Synthetic ontologies: A hypothesis. *Available at SSRN 4373537*.

Borrmann, A., König, M., Koch, C., & Beetz, J. (2018). Building information modeling technology foundations and industry practice: Technology foundations and industry practice.

Borrmann, A., König, M., Koch, C., & Beetz, J. (2021). *Building information modeling-technologische grundlagen und industrielle praxis* (A. Borrmann, M. König, C. Koch, & J. Beetz, Eds.). Springer Fachmedien Wiesbaden. https://doi.org/10.1007/978-3-658-33361-4

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, *33*, 1877–1901.

buildingSMART. (2023). Bcf-api [Accessed: 2024-08-28].

buildingSMART International. (2023). Sample test files [Last commit: 2023; Accessed: 2024-08-06]. https://github.com/buildingSMART/Sample-Test-Files

buildingSMART Technical. (2022). *Software implementations* (Technical Specification). buildingSMART International. https://technical.buildingsmart.org/

buildingSMART Technical. (2023). *Bim collaboration format (bcf)* (Technical Specification). buildingSMART International. https://technical.buildingsmart.org/standards/bcf/

Caufield, J. H., Hegde, H., Emonet, V., Harris, N. L., Joachimiak, M. P., Matentzoglu, N., Kim, H., Moxon, S., Reese, J. T., Haendel, M. A., et al. (2024). Structured

prompt interrogation and recursive extraction of semantics (spires): A method for populating knowledge bases using zero-shot learning. *Bioinformatics*, *40*(3), btae104.

Chen, W., Su, Y., Zuo, J., Yang, C., Yuan, C., Qian, C., Chan, C.-M., Qin, Y., Lu, Y., Xie, R., et al. (2023). Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors in agents. *arXiv preprint arXiv:2308.10848*, *2*(4), 6.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al. (2023). Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, *24*(240), 1–113.

Dagdelen, J., Dunn, A., Lee, S., Walker, N., Rosen, A. S., Ceder, G., Persson, K. A., & Jain, A. (2024). Structured information extraction from scientific text with large language models. *Nature Communications*, *15*(1), 1418.

Dang, H. T. (2006). Duc 2005: Evaluation of question-focused summarization systems. *Proceedings of the Workshop on Task-Focused Summarization and Question Answering*, 48–55.

Deepset. (2024). Haystack [[Online; accessed 8-August-2024]]. https://github.com/deepset-ai/haystack

Deng, H., Xu, Y., Deng, Y., & Lin, J. (2022). Transforming knowledge management in the construction industry through information and communications technology: A 15-year review. *Automation in Construction*, *142*, 104530.

Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., et al. (2024). The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Edge, D., Trinh, H., Cheng, N., Bradley, J., Chao, A., Mody, A., Truitt, S., & Larson, J. (2024). From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Fill, H.-G., Fettke, P., & Köpke, J. (2023). Conceptual modeling and large language models: Impressions from first experiments with chatgpt. *Enterprise Modelling and Information Systems Architectures (EMISAJ)*, *18*, 1–15.

Francis, N., Green, A., Guagliardo, P., Libkin, L., Lindaaker, T., Marsault, V., Plantikow, S., Rydberg, M., Selmer, P., & Taylor, A. (2018). Cypher: An evolving query language for property graphs. *Proceedings of the 2018 international conference on management of data*, 1433–1445.

Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., & Wang, H. (2023). Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

Guia, J., Soares, V. G., & Bernardino, J. (2017). Graph databases: Neo4j analysis. *ICEIS (1)*, 351–356.

Hamdan, A.-H., Bonduel, M., & Scherer, R. J. (2019). An ontological model for the representation of damage to constructions. *LDAC*, 64–77.

He, H., Zhang, H., & Roth, D. (2022). Rethinking with retrieval: Faithful large language model inference. *arXiv preprint arXiv:2301.00303*.

Hogan, A., Blomqvist, E., Cochez, M., d'Amato, C., Melo, G. D., Gutierrez, C., Kirrane, S., Gayo, J. E. L., Navigli, R., Neumaier, S., et al. (2021). Knowledge graphs. *ACM Computing Surveys (Csur)*, *54*(4), 1–37.

Holzschuher, F., & Peinl, R. (2013). Performance of graph query languages: Comparison of cypher, gremlin and native access in neo4j. *Proceedings of the Joint EDBT/ICDT 2013 Workshops*, 195–204.

Huang, W., Xia, F., Xiao, T., Chan, H., Liang, J., Florence, P., Zeng, A., Tompson, J., Mordatch, I., Chebotar, Y., et al. (2022). Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*.

Huang, Z., Gutierrez, S., Kamana, H., & MacNeil, S. (2023). Memory sandbox: Transparent and interactive memory management for conversational agents. *Adjunct Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, 1–3.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., Casas, D. d. l., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., et al. (2023). Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Kim, T., & Chi, S. (2019). Accident case retrieval and analyses: Using natural language processing in the construction industry. *Journal of Construction Engineering and Management*, *145*(3), 04019004.

Klein, G., Moon, B., & Hoffman, R. R. (2006). Making sense of sensemaking 1: Alternative perspectives. *IEEE intelligent systems*, *21*(4), 70–73.

Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., & Iwasawa, Y. (2022). Large language models are zero-shot reasoners. *Advances in neural information processing systems*, *35*, 22199–22213.

Kong, A., Zhao, S., Chen, H., Li, Q., Qin, Y., Sun, R., & Zhou, X. (2023). Better zero-shot reasoning with role-play prompting. *arXiv preprint arXiv:2308.07702*.

Konys, A. (2019). Knowledge repository of ontology learning tools from text. *Procedia Computer Science*, *159*, 1614–1628.

LangChain. (2024a). Langchain [[Online; accessed 8-August-2024]]. https://github.com/langchain-ai/langchain

LangChain. (2024b). Langchain documentation [[Online; accessed 8-August-2024]]. https://python.langchain.com/v0.2/docs/introduction/

LangChain and Neo4j. (2024). Graphcypherqachain [Accessed: [Insert access date here]].

Laskar, M. T. R., Hoque, E., & Huang, J. X. (2022). Domain adaptation with pre-trained transformers for query-focused abstractive text summarization. *Computational Linguistics*, *48*(2), 279–320.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., et al. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, *33*, 9459–9474.

Li, Y., & Manoharan, S. (2013). A performance comparison of sql and nosql databases. *2013 IEEE Pacific Rim conference on communications, computers and signal processing (PACRIM)*, 15–19.

Lindner, F., & Wald, A. (2011). Success factors of knowledge management in temporary organizations. *International Journal of project management*, *29*(7), 877–888.

Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., & Liang, P. (2024). Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, *12*, 157–173.

Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., & Neubig, G. (2023). Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, *55*(9), 1–35.

LlamaIndex. (2024). Llamaindex [[Online; accessed 8-August-2024]]. https://github.com/run-llama/llama_index

Manning, C. D. (2022). Human language understanding & reasoning. *Daedalus*, *151*(2), 127–138.

Masterman, T., Besen, S., Sawtell, M., & Chao, A. (2024). The landscape of emerging ai agent architectures for reasoning, planning, and tool calling: A survey. *arXiv preprint arXiv:2404.11584*.

Mateiu, P., & Groza, A. (2023). Ontology engineering with large language models. *arXiv preprint arXiv:2307.16699*.

Nakano, R., Hilton, J., Balaji, S., Wu, J., Ouyang, L., Kim, C., Hesse, C., Jain, S., Kosaraju, V., Saunders, W., et al. (2021). Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*.

Neo4j. (2024a). *Cypher full text indexes* [Accessed: 2024-08-15]. Neo4j, Inc. https://neo4j.com/docs/cypher-manual/current/indexes/semantic-indexes/full-text-indexes/

Neo4j. (2024b). *Cypher manual* [Accessed: 2024-08-15]. Neo4j, Inc. https://neo4j.com/docs/cypher-manual/current/introduction/

Neo4j and LangChain. (2024). Constructing knowledge graphs [Accessed on August 12, 2024]. *LangChain*. https://python.langchain.com/v0.1/docs/use_cases/graph/constructing/

Neo4j, Inc. (2024). Neo4j graph database [Accessed: 30.08.2024].

OpenAI. (2024). Hello, gpt-4o [Accessed: 30.09.24].

Park, J. S., O'Brien, J., Cai, C. J., Morris, M. R., Liang, P., & Bernstein, M. S. (2023). Generative agents: Interactive simulacra of human behavior. *Proceedings of the 36th annual acm symposium on user interface software and technology*, 1–22.

Qian, C., Cong, X., Yang, C., Chen, W., Su, Y., Xu, J., Liu, Z., & Sun, M. (2023). Communicative agents for software development. *arXiv preprint arXiv:2307.07924*, *6*.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, *1*(8), 9.

Rasmussen, M. H., Lefrançois, M., Schneider, G. F., & Pauwels, P. (2021). Bot: The building topology ontology of the w3c linked building data group. *Semantic Web*, *12*(1), 143–161.

Razali, M. F., Haron, N. A., Hassim, S., Alias, A. H., Harun, A. N., & Abubakar, A. S. (2019). A review: Application of building information modelling (bim) over building

life cycles. *IOP Conference Series: Earth and Environmental Science*, *357*(1), 012028.

Schulz, O., Oraskari, J., & Beetz, J. (2023). Lessons learned from designing and using bcfowl. *LDAC*, 23–34.

Singhal, A. (2012). *Introducing the knowledge graph: Things, not strings* [Accessed: August 23, 2024]. Retrieved August 23, 2024, from https://googleblog.blogspot.com/2012/05/introducing-knowledge-graph-things-not.html

Tan, H. C., Carrillo, P. M., Anumba, C. J., Bouchlaghem, N., Kamara, J. M., & Udeaja, C. E. (2007). Development of a methodology for live capture and reuse of project knowledge in construction. *Journal of management in engineering*, *23*(1), 18–26.

Team, G., Anil, R., Borgeaud, S., Wu, Y., Alayrac, J.-B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A. M., Hauth, A., et al. (2023). Gemini: A family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Templeton, A. (2024). *Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet*. Anthropic.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. (2023). Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

Vaz-Serra, P., & Edwards, P. (2021). Addressing the knowledge management "nightmare" for construction companies. *Construction Innovation*, *21*(2), 300–320.

vesoft inc. (2024). Nebulagraph: Distributed graph database [Accessed: 30.08.2024].

Vicknair, C., Macias, M., Zhao, Z., Nan, X., Chen, Y., & Wilkins, D. (2010). A comparison of a graph database and a relational database: A data provenance perspective. *Proceedings of the 48th annual Southeast regional conference*, 1–6.

Wang, H., Meng, X., & Zhu, X. (2022). Improving knowledge capture and retrieval in the bim environment: Combining case-based reasoning and natural language processing. *Automation in Construction*, *139*, 104317.

Wang, L., Ma, C., Feng, X., Zhang, Z., Yang, H., Zhang, J., Chen, Z., Tang, J., Chen, X., Lin, Y., et al. (2024). A survey on large language model based autonomous agents. *Frontiers of Computer Science*, *18*(6), 186345.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. (2022). Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, *35*, 24824–24837.

Wu, T., Terry, M., & Cai, C. J. (2022). Ai chains: Transparent and controllable human-ai interaction by chaining large language model prompts. *Proceedings of the 2022 CHI conference on human factors in computing systems*, 1–22.

Xu, Z., Cruz, M. J., Guevara, M., Wang, T., Deshpande, M., Wang, X., & Li, Z. (2024). Retrieval-augmented generation with knowledge graphs for customer service question answering. *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2905–2909.

Yang, J., Jin, H., Tang, R., Han, X., Feng, Q., Jiang, H., Zhong, S., Yin, B., & Hu, X. (2024). Harnessing the power of llms in practice: A survey on chatgpt and beyond. *ACM Transactions on Knowledge Discovery from Data*, *18*(6), 1–32.

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T., Cao, Y., & Narasimhan, K. (2024). Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, *36*.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2022). React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Yepes, V., & López, S. (2021). Knowledge management in the construction industry: Current state of knowledge and future research. *Journal of Civil Engineering and Management*, *27*(8), 671–680.

Yu, D., & Yang, J. (2018). Knowledge management research in the construction industry: A review. *Journal of the Knowledge Economy*, *9*, 782–803.

Zhang, B., Carriero, V. A., Schreiberhuber, K., Tsaneva, S., González, L. S., Kim, J., & de Berardinis, J. (2024). Ontochat: A framework for conversational ontology engineering using language models. *arXiv preprint arXiv:2403.05921*.

Zhang, Y., Li, Y., Cui, L., Cai, D., Liu, L., Fu, T., Huang, X., Zhao, E., Zhang, Y., Chen, Y., et al. (2023). Siren's song in the ai ocean: A survey on hallucination in large language models. *arXiv preprint arXiv:2309.01219*.

Zou, Y., Kiviniemi, A., & Jones, S. W. (2017). Retrieving similar cases for construction project risk management using natural language processing techniques. *Automation in construction*, *80*, 66–76.