

Jaakko Junttu

ENHANCING INFORMATION ACCESSIBILITY IN INFRASTRUCTURE CONSTRUCTION

Using retrieval-augmented generation for efficient parsing of
information requirements

Master of Science Thesis
Faculty of Built Environment
Examiners: Ph.D. Osku Torro
Asst. Prof. Antti Kurvinen
August 2025

ABSTRACT

Jaakko Junttu: Enhancing information accessibility in infrastructure construction
Master of Science Thesis
Tampere University
Master's Degree Programme in Department of Civil Engineering
August 2025

Information management plays a central role in modern infrastructure projects, requiring the efficient interpretation of various project documents such as requests for proposals and requirement specifications. Identifying and summarizing the information requirements within these documents can be time-consuming and prone to error. Retrieval-Augmented Generation (RAG) combines the generative capabilities of Large Language Models (LLMs) with vector-based retrieval, potentially improving information access and interpretation in the context of infrastructure projects.

This thesis investigates whether a simple hybrid RAG implementation can serve as a useful tool for the automatic retrieval and summarization of information requirements. A RAG system is implemented in which project documents are converted into plain text, divided into smaller chunks, vectorized using an embedding model, and stored in a database. Based on this data, the language model is used to answer user queries.

The study is guided by three research questions: (1) How effectively can a hybrid RAG system retrieve relevant information requirements from infrastructure project documents? (2) Can the system accurately summarize and interpret the retrieved content? (3) What are the main challenges and limitations of such a system?

The system is tested on real-world datasets from infrastructure projects, and its performance is evaluated accordingly. The results suggest that the RAG approach shows potential in supporting information modeling tasks, particularly in preprocessing and extracting relevant content from large documents. However, challenges remain, especially in retrieval accuracy and the language model's ability to consistently utilize the retrieved information. The findings offer a foundation for future research and the further development of RAG-based methods for information management in infrastructure projects.

Keywords: Information modeling, Retrieval augmented generation, Large language model

The originality of this thesis has been checked using the Turnitin OriginalityCheck service.

TIIVISTELMÄ

Jaakko Junntu: Tiedon saatavuuden parantaminen infrarakentamisessa
Diplomityö
Tampereen yliopisto
Rakennustekniikan diplomi-insinöörin tutkinto-ohjelma
Elokuu 2025

Tiedonhallinta on keskeinen osa nykyaikaisia infrahankkeita, ja sen onnistuminen edellyttää erilaisten projektidokumenttien, kuten tarjouspyyntöjen ja vaatimuserittelyjen, tehokasta tulkintaa. Näiden asiakirjojen sisältämien tietovaatimusten tunnistaminen ja yhteenveto voi kuitenkin olla aikaa vievää ja virhealtista. Retrieval-Augmented Generation (RAG) -menetelmä yhdistää suurten kielimallien (LLM) kyvyn tuottaa tekstiä vektoripohjaiseen tiedonhakuun, ja sen avulla voidaan mahdollisesti parantaa tiedon saavutettavuutta ja tulkintaa infrastruktuuriprojekteissa.

Tämän diplomityön tavoitteena on selvittää, voiko yksinkertaistettu ns. hybridi RAG -toteutus toimia hyödyllisenä työkaluna tietovaatimusten automaattisessa haussa ja tiivistämisessä. Tutkimuksessa toteutetaan RAG-järjestelmä, jossa dokumentit muunnetaan tekstimuotoon, jaetaan pienempiin osiin, vektoroidaan ja tallennetaan tietokantaan. Näiden tietojen perusteella kielimalli vastaa käyttäjän kyselyihin.

Tutkimus perustuu kolmen tutkimuskysymyksen tarkasteluun: (1) Kuinka tehokkaasti hybridi RAG -järjestelmä löytää infrahankkeiden dokumenteista olennaiset tietovaatimukset? (2) Onko järjestelmä kykenevä tiivistämään ja tulkitsemaan dokumenttien sisältöä tarkasti? (3) Mitkä ovat järjestelmän keskeisimmät haasteet ja rajoitukset?

Aineistona käytetään todellisia infraprojektien asiakirjoja, ja järjestelmän suorituskykyä arvioidaan niiden pohjalta. Tulosten perusteella RAG-menetelmällä on potentiaalia tukea tietomallintamista, erityisesti dokumenttien esikäsittelyssä ja olennaisen tiedon esiin nostamisessa. Haasteita liittyy kuitenkin erityisesti haun tarkkuuteen sekä kielimallin kykyyn hyödyntää haettua tietoa johdonmukaisesti. Tulokset tarjoavat suuntaa jatkotutkimukselle ja RAG-pohjaisten menetelmien kehittämiseksi infrahankkeiden tiedonhallinnassa.

Avainsanat: Tietomallinnus, RAG, Kielimalli

Tämän julkaisun alkuperäisyys on tarkastettu Turnitin OriginalityCheck -ohjelmalla.

PREFACE

Throughout my studies in construction management and machine learning, I've been interested in how modern AI-techniques can be applied to real-world problems in the built environment. With this thesis, I wanted to bring those two fields together by exploring how Large Language Models and especially Retrieval-Augmented Generation could support information management in infrastructure projects.

This work allowed me to apply technical concepts in a context that's closely tied to my academic background, while also learning new tools and approaches along the way. The process was both challenging and rewarding, and it deepened my interest in AI-assisted knowledge systems.

I want to sincerely thank my thesis instructor, Osku Torro (Ph.D.), for his consistent support, sharp feedback, and practical insights during the project. His guidance helped shape this work into its final form.

In Tampere, 18th August 2025

Jaakko Junttu

CONTENTS

1.	Introduction	1
1.1	Aims of the Study	1
1.2	Thesis Structure	2
2.	Background.	3
2.1	Information modeling in infrastructure projects	3
2.1.1	Parsing of information requirements today	4
2.2	Retrieval augmented generation	5
2.2.1	RAG process	6
2.2.2	Vectorization in RAG.	7
2.2.3	Retrieval in RAG	8
2.2.4	Evolution of RAG	9
2.3	Large language models	10
2.3.1	Development of large language models	10
2.3.2	Large language models today	11
2.3.3	Challenges with large language models	11
2.3.4	RAG and LLMs	12
3.	Materials and methods	14
3.1	Datasets	14
3.2	Data Preparation	15
3.2.1	Extracting the Documents	15
3.2.2	Chunking	16
3.2.3	Uploading chunked documents into indexes	17
3.3	RAG System Implementation	18
3.3.1	Retrieval	18
3.3.2	Generation	18
3.4	Evaluation	20
3.4.1	Retrieval Evaluation	21
3.5	Generation Evaluation.	22
4.	Results	26
4.1	Retrieval Phase	26
4.1.1	Normalized Relevance Scores	26
4.1.2	Search Latency.	27
4.2	Generation Evaluation.	28
4.3	System evaluation	31

5. Discussion	32
5.1 Retrieval findings	32
5.2 Generation findings	33
5.3 Future work and development	35
5.3.1 Technical enhancements	35
5.3.2 Implications for industry adoption	36
6. Conclusion	38
References	40

LIST OF SYMBOLS AND ABBREVIATIONS

AI	Artificial Intelligence
ANNS	Approximate Nearest Neighbor Search
CRAG	Corrective Retrieval Augmented Generation
DPR	Dense Passage Retrieval
IFC	Industry Foundation Class
IM-RAG	Inner Monologue Retrieval Augmented Generation
LLM	Large Language Model
LM	Language Modeling
LSH	Locally Sensitive Hashing
ML	Machine Learning
MPL	Multi Layer Perceptron
MoE	Mixture of Experts
NLM	Neural Language Model
NLP	Natural Language Processing
NN	Neural Network
NNS	Nearest Neighbor Search
PLM	Pre-trained Language Model
RAG	Retrieval Augmented Generation
RNN	Recurrent Neural Network
SLM	Statistical Language Model

1. INTRODUCTION

In modern infrastructure projects, compliance with information requirement standards and project-specific requirements is essential for ensuring interoperability, efficiency, and regulatory adherence. However, navigating and extracting relevant requirements from extensive documentation can be a complex and time-consuming task. Traditional search methods often struggle to provide precise results, requiring significant manual effort to interpret and apply the retrieved information. Large Language Models (LLMs) have demonstrated remarkable capabilities in processing and summarizing text, yet they are limited by their reliance on pre-trained knowledge, which may be outdated or incomplete. Retrieval-Augmented Generation (RAG) offers a promising solution by combining LLMs with real-time retrieval mechanisms, enabling more accurate and context-aware responses.

This study investigates how RAG can be used to query and process information requirements, addressing key challenges in retrieving and interpreting technical documents. The research focuses on developing a system that enhances LLM outputs by integrating retrieved content, thereby improving factual accuracy and domain relevance. The study is guided by the following research questions:

1. How effectively can a RAG system retrieve and summarize information requirements relevant to user queries?
2. What are the limitations and challenges in applying RAG for technical document retrieval?
3. How does the integration of RAG with LLMs impact the accuracy and relevance of generated responses?

By exploring these questions, this thesis aims to contribute to the development of AI-driven solutions for knowledge management in complex technical domains.

1.1 Aims of the Study

The primary objective of this study is to explore the integration of Retrieval-Augmented Generation (RAG) with Large Language Models (LLMs) for querying and infrastructure project files for information modeling and specifically parsing information requirements. The study investigates how RAG can enhance the retrieval and summarization of relevant

documents, improving the accuracy and efficiency of information access. Specifically, the research aims to evaluate RAG usage to retrieve relevant information requirements, assess the effectiveness of RAG-enhanced LLMs in summarizing and interpreting project documents, and identify the challenges and limitations of using RAG in this context. By addressing these objectives, the study contributes to the broader field of AI-driven information management and automation in infrastructure projects.

The research is guided by the following questions:

- How effectively can a hybrid RAG pipeline retrieve information requirements from infrastructure project documents?
- Can RAG-enhanced LLMs accurately summarize and interpret the retrieved content in the context information modeling?
- What are the main limitations and challenges of applying RAG in this domain?

1.2 Thesis Structure

This thesis is structured as follows:

Chapter 1 introduces the research problem, objectives, and structure of the study. Chapter 2 provides background information on information requirements in infrastructure projects, Retrieval-Augmented Generation (RAG), vectorization, retrieval methods, and Large Language Models (LLMs). Chapter 3 outlines the methodology used to integrate RAG with LLMs, including dataset selection, preprocessing techniques, and the evaluation methods applied. Chapter 4 presents the system performance results, which are further analyzed and discussed in Chapter 5. Finally, Chapter 6 summarizes the findings and proposes directions for future work.

2. BACKGROUND

In infrastructure projects, one of the critical challenges is managing and accessing the vast amount of information requirements each phase of the project demands. The complexity of these projects — spanning design, construction, and maintenance phases — often leads to fragmented data sources, inconsistent documentation, and difficulties in retrieving relevant information when needed. This issue creates inefficiencies, delays, and potential errors that can compromise project outcomes. Addressing this problem is essential to improving workflows and ensuring that stakeholders have timely access to accurate and actionable insights. This thesis aims to combat these challenges by exploring technological solutions such as large language models and retrieval augmented generation, which have the potential to streamline information retrieval and utilization in such contexts.

2.1 Information modeling in infrastructure projects

Infrastructure projects involve extensive information requirements across various phases, including planning, design, construction, and maintenance. The effective management of these information requirements is crucial for ensuring consistency, efficiency, and accuracy in project execution. However, current practices often lead to fragmented documentation, inconsistencies in data formats, and a lack of clarity in requirements, which can create significant challenges for project stakeholders.

One of the fundamental issues in information management within infrastructure projects is the variability and ambiguity of information requirements. The process of defining and enforcing requirements is often inconsistent, with different projects adopting different standards and specifications. While general frameworks for information modeling requirements exist, additional project-specific requirements may introduce deviations that complicate standardization. This lack of uniformity can lead to misunderstandings between project owners, designers, and contractors.

A significant challenge arises from the way requirements are documented and communicated. In many cases, requirements are scattered across various documents, including tendering materials, technical guidelines, and organization-specific modeling standards. This fragmentation makes it difficult for project participants to gain a holistic view of the

requirements applicable to a specific project (buildingSMART Finland 2021). A common pitfall is the assumption that adhering to general standards automatically ensures clarity and completeness, when in reality, additional project-specific requirements often need to be considered. An example of this is if a designer is working on a project commissioned by the Finnish Transport Infrastructure Agency (FTIA), the primary information modeling requirements are defined by FTIA's Infra Model Requirements. These requirements supplement and specify aspects of the general InfraBIM (YIV) guidelines, which remain applicable unless FTIA's requirements state otherwise (Finnish Transport Infrastructure Agency 2022).

Another issue is the interpretation of these requirements. Even when specifications are clearly stated, discrepancies can arise between the expectations of the client and the understanding of the information providers. If a project specifies that data must be delivered in compliance with a given framework, both parties may have different assumptions about what that entails (buildingSMART Finland 2021). Additionally, the evolution of standards means that some requirements may no longer align with current industry practices, further complicating compliance efforts.

Furthermore, the format and structure of the information requirements present challenges in accessibility and usability. Documents containing these requirements often include information in non-machine-readable formats, making automated processing difficult. Key specifications may be embedded in textual descriptions, color-coded formats, or diagrams, limiting their usability in digital workflows (buildingSMART Finland 2021). This results in increased manual effort in extracting, interpreting, and applying requirements, which can be error-prone and time-consuming.

These challenges highlight the need for improved methods of organizing, retrieving, and utilizing information requirements in infrastructure projects (buildingSMART Finland 2021). The potential of large language models (LLMs) and retrieval-augmented generation (RAG) to address these issues is particularly promising. By leveraging these technologies, it is possible to develop systems that can efficiently extract and synthesize relevant requirements from multiple sources, providing project participants with a clear and structured understanding of applicable standards. Such an approach could enhance transparency, reduce misinterpretations, and improve the overall efficiency of infrastructure project workflows.

2.1.1 Parsing of information requirements today

With practically no aid of AI, the current process of parsing information requirements in infrastructure projects is largely manual and labor-intensive. Project stakeholders, such as designers, modelers, and contractors, are typically required to review a wide range of documents, including general InfraBIM guidelines (YIV), client-specific modeling in-

structions, and project-specific tender documents. This review process relies heavily on individual interpretation and professional experience.

In practice, the task often involves manually scanning through lengthy documents to locate relevant requirements. These documents may use inconsistent terminology and formatting, further complicating the task. For example, a modeling requirement might be described in free-form text within one document, represented as a diagram in another, or implied indirectly through references to standards. As a result, identifying applicable requirements involves not only locating information but also interpreting its meaning and relevance in a specific project context.

The fragmentation of requirements across multiple unstructured sources leads to a high cognitive load for practitioners, and thus is also error-prone. Since requirements are rarely centralized or presented in a standardized, machine-readable format, project participants must invest considerable effort in aligning different requirement sources and resolving inconsistencies manually. In cases where general guidelines (e.g., YIV) are supplemented by organization-specific requirements (e.g., FTIA Infra Model Requirements), practitioners must carefully trace which rules override or extend others. This creates room for ambiguity and increases the risk of non-compliance or misunderstandings.

Furthermore, there is often limited support for cross-referencing or validation of requirements during project execution. For example, once a model is delivered, checking its compliance against all relevant information requirements is typically done via manual review rather than automated validation. This increases the likelihood of errors and omissions, especially in complex projects involving multiple disciplines and stakeholders.

Overall, the current approach to parsing information requirements is characterized by manual extraction, subjective interpretation, and fragmented documentation. The current process also relies heavily on the experience of the practitioners. These challenges highlight a clear opportunity for improvement through the adoption of AI-assisted methods, such as retrieval-augmented generation, to automate and standardize the way information requirements are identified, interpreted, and applied in infrastructure projects.

2.2 Retrieval augmented generation

Retrieval augmented generation (RAG) was first introduced in a 2020 paper by Meta (Facebook), in which they came up with a framework to give large language models (LLM) access to information that was not included in its training data (Martineau 2021). This process is used alongside LLMs to help them perform in knowledge intensive or domain specific tasks, in which they usually struggle due to not having data regarding the specific topic. RAG looks up relevant documents from a database and utilizing document retrieval techniques, finds the most relevant documents regarding the user query. Those

documents (or parts of documents, chunks) are then fed to an LLM as added context. (Gao et al. 2024; *Retrieval Augmented Generation (RAG) and Semantic Search for GPTs 2024*)

2.2.1 RAG process

For Retrieval-Augmented Generation (RAG) to function effectively, the raw data must first be converted into a plain text format from its original formats, such as PDF, XLSX, or HTML, and then divided into smaller chunks (*Retrieval Augmented Generation (RAG) and Semantic Search for GPTs 2024*). These chunks are passed through an embedding model, a neural network that transforms them into vector representations. These embeddings are stored in a vector database.

Once indexed, the database can be searched by comparing a user's query—also converted into a vector using the same embedding model—to the stored vectors. This similarity search retrieves the chunks most relevant to the query based on vector proximity (Gao et al. 2024; Microsoft Azure 2024). This entire process comprises the retrieval phase of RAG.

With the relevant documents retrieved, the RAG model can then generate a response by feeding both the user query and the selected context into a large language model (LLM). This allows the system to produce answers grounded in the retrieved information (Gao et al. 2024).

The difficulties in each step of the process include, but are not limited to:

- Challenges in retrieving
 - Precision and recall issues
 - Misaligned or irrelevant chunks of data Missing crucial information
- Challenges in generation
 - Hallucination¹
 - Irrelevance of information

In this work, a hybrid RAG (Retrieval-Augmented Generation) approach is implemented, meaning the system follows the basic two-step RAG architecture without any advanced components such as multi-step reasoning, agent frameworks, or advanced retrieval techniques. In this setup, the process is straightforward: the user query is embedded and used to retrieve relevant chunks from a preprocessed and vectorized database of information requirements. These retrieved chunks are then passed—along with the original query and an optional system prompt—to a large language model (LLM). The objective

¹Though hallucinations are mitigated significantly compared to only using LLMs.

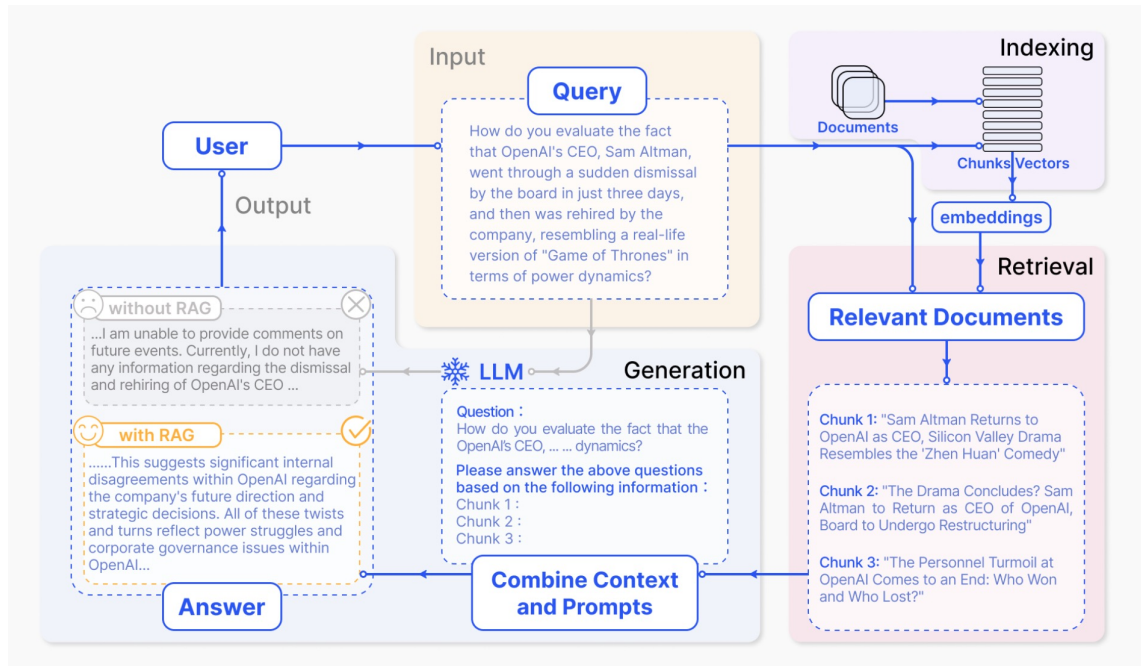


Figure 2.1. RAG visualized. (Gao et al. 2024, p. 3)

is to assess whether this simple configuration can provide sufficient context for the LLM to accurately respond to queries about the requirements.

2.2.2 Vectorization in RAG

Vectorization is a key process in RAG systems, enabling efficient retrieval of relevant information by converting textual data into numerical representations. This transformation allows for the storage and comparison of documents within a high-dimensional vector space, facilitating similarity-based searches that enhance the accuracy and efficiency of retrieval processes. (Microsoft Azure 2024) The effectiveness of vectorization is largely dependent on the choice of embedding models, vector databases, and search algorithms used in the retrieval pipeline. (Şakar and Emekci 2025)

Embedding models are responsible for generating vector representations of textual data by capturing semantic relationships between words, phrases, and documents. These models are often trained on large-scale corpora and project text into a dense, high-dimensional space where similar concepts are positioned closer together. (Neelakantan et al. 2022) Modern embedding models such as OpenAI's `text-embedding-v3-large`, BAAI's `bge-en-small`, and Cohere's `cohere-en-v3` have demonstrated varying degrees of effectiveness in different RAG scenarios. The selection of an appropriate embedding model is a crucial part in the process of RAG, as it directly impacts the quality of retrieved documents and the overall performance of the system. (Şakar and Emekci 2025)

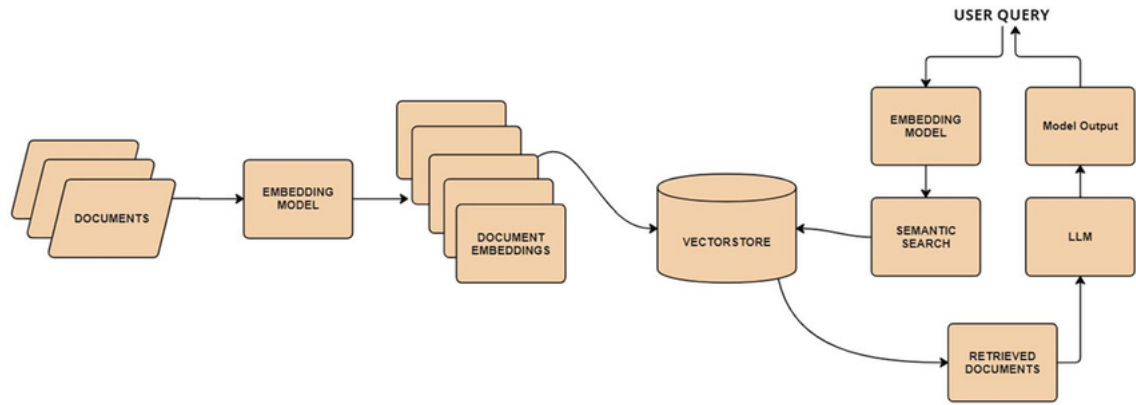


Figure 2.2. Illustration of the process of searching and matching similar contexts from a vector database based on a user's query. (Şakar and Emekci 2025, p. 4)

2.2.3 Retrieval in RAG

Once textual data has been transformed into vector embeddings, it must be efficiently stored and retrieved. Vector databases play an important role in managing these embeddings. (*Retrieval Augmented Generation (RAG) and Semantic Search for GPTs 2024*) Unlike traditional databases that rely on exact keyword matching, vector databases leverage similarity metrics such as cosine similarity (2.1), Euclidean distance, or dot product to retrieve the most relevant documents for a given query. Cosine similarity in particular is a widely used measure, calculated as

$$\cos(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n (A_i)^2} \cdot \sqrt{\sum_{i=1}^n (B_i)^2}}, \quad (2.1)$$

where \mathbf{A} and \mathbf{B} are the vector representations of two documents or smaller chunks (such as the user query). Cosine similarity score ranges from -1 to 1, with higher values indicating greater semantic similarity. The ability to retrieve documents based on semantic similarity rather than exact word-by-word matching has been shown to significantly improve the effectiveness of RAG systems, particularly in tasks where contextual understanding is essential. (Şakar and Emekci 2025)

Search algorithms play an important role in determining how effectively vectorized data is queried and retrieved, directly impacting the efficiency and accuracy of RAG systems. At the core of these algorithms is nearest neighbor search (NNS), which identifies the most relevant data points within a dataset based on a selected distance metric, such as the aforementioned cosine similarity. While exact nearest neighbor search techniques, like linear search or k-d trees, provide precise results, their computational complexity renders them impractical for large-scale, high-dimensional datasets. (Şakar and Emekci 2025)

To address these challenges, Approximate Nearest Neighbor Search (ANNS) methods have emerged as a practical alternative, balancing retrieval accuracy with computational efficiency. These methods employ various indexing techniques, such as locality-sensitive hashing (LSH) to facilitate faster searches while maintaining reasonable precision. By leveraging these indexing structures, ANNS enables rapid retrieval across large-scale vector databases without the need for exhaustive comparisons against all stored embeddings. (Şakar and Emekci 2025)

In a typical RAG pipeline, the similarity search process involves transmitting query vector embeddings to a vector store containing document embeddings. (*What is RAG?* 2025) The search algorithm identifies the most similar embeddings, retrieves top ranking relevant documents, ranks them using cosine similarity, and passes them to the generative model for response generation. This retrieval mechanism forms the foundation of RAG, highlighting the importance of efficient search algorithms to improve system performance. (Şakar and Emekci 2025)

2.2.4 Evolution of RAG

As AI continues to evolve rapidly, so do the architectures and techniques surrounding it. Since the introduction of the original, so-called vanilla RAG approach, numerous variations have emerged, building on the same core principle. While this study employs a basic hybrid RAG setup, it's important to acknowledge that the field has advanced significantly. The most notable developments and extensions of the basic RAG techniques are outlined below.

Graph RAG

Graph RAG builds on traditional RAG by using graph-based structures to improve reasoning and context. It connects entities through nodes, manages both structured and unstructured data with hierarchies, and enhances responses by leveraging relationships within the graph. This makes it ideal for tasks needing deep relational understanding, such as healthcare diagnostics or legal research. However, it faces challenges like limited scalability with large datasets, reliance on high-quality graph data, and complex integration with unstructured systems. (Singh et al. 2025)

Agentic RAG

Agentic RAG introduces autonomous agents that make real-time decisions and optimize workflows. Unlike static systems, it uses iterative feedback to refine retrieval and dynamically adjusts strategies for complex queries. This adaptability suits applications like customer support, financial analytics, or adaptive learning platforms. The challenges of

such a system include struggling with coordination between different agents, high computational demands, and scalability issues under heavy query loads. (Singh et al. 2025)

Modular RAG

Modular RAG focuses on flexibility by breaking down the retrieval and generation process into independent, reusable parts. It combines sparse and dense retrieval methods, like BM25 and Dense Passage Retrieval (DPR), to improve accuracy across various query types. It can also integrate external tools, such as APIs or external databases for tasks like real-time data analysis. Composable pipelines allow components to be swapped or adjusted for specific needs, making it highly adaptable. This makes Modular RAG ideal for high complexity work requiring strong precision and scalability. (Singh et al. 2025)

2.3 Large language models

Deciphering information requirements is in its core a natural language processing (NLP) problem which large language models (LLM) have proven to be a viable solution for. LLMs first came to widespread knowledge with the public release of ChatGPT in November 2022 (Minaee et al. 2024). LLMs such as GPT-4 from OpenAI, LLaMA-3 from Meta and Gemini from Google have only become more popular since then, and are widely used by many people today. Their rapid adoption has been driven by advancements in model architecture, increased computational power, and the availability of vast training datasets, enabling them to generate highly coherent and contextually relevant responses.

2.3.1 Development of large language models

The history and development of language modeling (LM) can be divided into four distinct steps of progress. The first of these is the development of statistical language models (SLM), which are models that try and predict the next word based on the previous context. The emergent problem with SLMs is that it is challenging to accurately estimate such models because of the need to estimate an exponential number of transitions (Zhao et al. 2024)

The next step in the development of LMs is the neural language model (NLM). NLMs refer to models that estimate the probability of sequences using neural networks (NN) composed of multi-layer perceptrons (MLP) and recurrent neural networks (RNN). This technology was then further refined by applying the idea of pre-training to the model. These pre-trained language models (PLM) were very effective in terms of generalizing semantic features. (Zhao et al. 2024)

The last step of progress was driven by the realization that PLMs are highly scalable and

the capability of the models at least in principle is correlated with the size of them (Kaplan et al. 2020). Thus tunable parameters have risen to hundreds of billions, and the models' performance has improved greatly. Today, large size PLMs are therefore called Large language models or LLMs. (Zhao et al. 2024)

2.3.2 Large language models today

Most prominent LLMs today are built on transformer technology, which leverages the self-attention mechanisms to calculate attention scores that capture how each word influences and relates to others in a sequence. The transformer also allows much more parallelization than conventional recurrent neural networks (RNN), which in turn allows for more efficient computing and thus the pre-training of very vast language models with a lot of data. (Kaplan et al. 2020; Minaee et al. 2024)

The question of whether just expanding the models' size or the amount of compute power is the most efficient way to improve these models has been present for some time now (Kaplan et al. 2020). Recently, this has been amplified in part due to the release of Deepseek's R1 model, which has demonstrated strong performance despite being significantly smaller than many of its counterparts. Deepseek's R1 model leverages more efficient training techniques, improved data selection, and architectural refinements to achieve competitive results without simply scaling up parameters and compute. (DeepSeek-AI et al. 2025)

This has sparked discussions within the AI research community about the diminishing returns of brute-force scaling. While increasing model size and compute has historically led to performance gains, recent advancements like those by DeepSeek-AI et al. suggest that algorithmic improvements, better tokenization, and smarter data curation may provide more efficient ways to enhance LLMs.

Techniques like those discussed in the paper *DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning*, including mixture of experts (MoE), and reinforcement learning (RL) are being explored as alternatives to pure scaling. These approaches aim to optimize how models process and retrieve information, reducing redundancy and improving contextual understanding without exponentially increasing computational costs. (DeepSeek-AI et al. 2025)

2.3.3 Challenges with large language models

The challenges of utilizing LLMs for tasks like the one discussed in this thesis, are boiled down to three main points: hallucination, non-transparent or untraceable reasoning processes and outdated knowledge. (Gao et al. 2024) In the context of LLMs the term *hallucination* is used to describe the event in which an LLM hasn't been trained on a piece

of knowledge, but when queried about it, it still tries to give an answer for said query. The answer is probably wrong, because the model doesn't have the required data to answer the question. (*What Are AI Hallucinations?* 2023)

The next problem is *non-transparent or untraceable reasoning*, which essentially means that the model arrives at a false conclusion, but the reason it got there is unclear, and therefore it can be difficult to correct the model's behavior. Unlike rule-based systems, which provide clear logical steps for their conclusions, LLMs operate as black boxes, relying on neural network weight distributions that obscure their reasoning processes (Castelvecchi 2016; Goodfellow, Bengio, and Courville 2016). Therefore, users cannot easily identify whether an error results from faulty training data, poor prompt formulation, or an inherent limitation of the model (Burrell 2016). This can be a serious problem in fields such as engineering, where incorrect or unverifiable outputs can have serious consequences (Rudin 2019).

The third major challenge is outdated knowledge, which stems from the fact that LLMs are trained on a fixed dataset and do not inherently update their knowledge post-training (Cheng et al. 2024). For example, an LLM trained before 2023 will not have access to the latest technological advancements, regulatory changes, or discoveries in scientific literature (Lewis et al. 2021). This limitation reduces their reliability for tasks which require up-to-date knowledge retrieval, forcing users to cross-check responses manually (Marcus 2020).

To deal with these issues, several strategies have been proposed. One approach is the RAG (discussed in chapter 2.2), which integrates LLM-generated responses with external document retrieval (Martineau 2021). By retrieving relevant documents from an external database before generating an answer, RAG can help reduce hallucinations and the other aforementioned problems, ensuring access to updated information.

2.3.4 RAG and LLMs

As the name "retrieval augmented generation" says, LLMs are always a part of RAG as the generation part of the pipeline. The RAG system works as follows:

1. The user asks a query from the system.
2. If vectorized search is used, the query is transformed into an embedding using a vectorization model.
3. The embedding is used to retrieve relevant documents from a vector database or other structured knowledge sources. Also keyword based or semantic searches can be used.
4. The retrieved documents are processed and formatted to be passed as context to

the LLM.

5. The LLM generates a response by combining the retrieved information with its pre-trained knowledge.
6. The final response is returned to the user.

The integration of RAG and LLMs allows for improved factual accuracy, as the language model can leverage up-to-date and domain-specific information rather than relying solely on its pre-trained knowledge . This approach mitigates common issues such as hallucinations and outdated responses. (*What is RAG? 2025*)

Advantages of RAG-enhanced LLMs

Using retrieval-augmented generation in conjunction with LLMs offers several benefits:

- **Improved factual accuracy:** By incorporating retrieved knowledge, the system reduces the likelihood of generating incorrect or outdated information.
- **Better domain adaptation:** RAG enables LLMs to specialize in specific industries or technical fields without requiring full retraining.
- **Efficient handling of large knowledge bases:** Rather than encoding vast amounts of knowledge within the model itself, RAG dynamically retrieves only the most relevant information.
- **Interpretability and traceability:** Since retrieved documents are included in responses, users can verify the source of information.

Despite these advantages, challenges remain in optimizing retrieval mechanisms, handling noisy or irrelevant retrieved documents, and ensuring a seamless fusion of retrieved content with the LLM's generative capabilities (*What is RAG? 2025*). Addressing these issues is an active area of research in AI and NLP.

3. MATERIALS AND METHODS

This study implements a hybrid Retrieval-Augmented Generation (RAG) system to evaluate its ability to parse information requirements and support infrastructure project designers. The system uses a hybrid retrieval strategy that combines vectorized search, keyword-based retrieval, and semantic reranking to extract relevant content from project documents.

A hybrid RAG approach was chosen for its modularity, simplicity, and clarity. Using off-the-shelf components, pretrained embedding models and a general-purpose LLM, the setup avoids complex fine-tuning or instruction training. This decision also aligns with the tools and infrastructure already in use within the target company, ensuring smoother integration and practical relevance. Furthermore, the hybrid setup serves as an effective baseline to assess whether such a system is suitable for the use case explored in this thesis.

3.1 Datasets

Datasets for this thesis consisted of different projects that the target company has been a part of in combination with some general information requirements from big parties, such as cities and the Finnish Transport and Infrastructure Agency. The data was in many different formats including XLSX, PDF, TXT, and DOCX and contained a diverse set of infrastructure project documents, including requirement specifications, design guidelines, calls for offers, and compliance checklists. These documents varied in structure and complexity, with some being highly structured (e.g., spreadsheets) and others more unstructured (e.g., PDFs and free-text reports).

By using a diverse dataset, the study can evaluate how well the RAG system can handle real-world infrastructure documentation as well as accurate and efficient information retrieval across different formats. On the other hand, the fact that the data is so vast for even a single project introduces challenges for the retrieval phase.

3.2 Data Preparation

To enable efficient retrieval, the collected documents were uploaded to Azure AI Search into separate indexes. This process involved several key steps, including preprocessing of the documents, generating embeddings, and uploading them into the indexes.

3.2.1 Extracting the Documents

The first step is to standardize and clean the documents stored in the directory. These documents exist in various formats such as DOCX, XLSX, TXT, and others, which means that they need to be converted to a plain text format. This preprocessing was carried out using Azure Document Intelligence, which applies a range of techniques to extract meaningful information from these diverse formats (Microsoft Azure 2025d). In this study, paragraphs, tables, and key-value pairs were extracted, and then sorted into their respective bins based on the project ID.

Document data was encapsulated in a dedicated Python class, `ParsedDocument` (Listing 3.1), which included document metadata, an `AnalyzeResult` object from the Azure Document Intelligence API, and a content field for plain text. The content field served as a fallback when the API could not process certain file formats. The `AnalyzeResult` class in the Azure Document Intelligence SDK stores all data returned from the document analysis process, including extracted text, tables, key-value pairs, metadata, and more (Microsoft Azure 2025a).

```

1 class ParsedDocument:
2     def __init__(
3         self,
4         metadata: Dict,
5         analyzerresult: Optional[AnalyzeResult] = None,
6         content: Optional[str] = None,
7     ):
8         self.analyzerresult = analyzerresult
9         self.metadata = metadata
10        self.content = content
11
12        def is_azure(self) -> bool:
13            return self.analyzerresult is not None
14
15        def __repr__(self) -> str:
16            return (
17                f"<ParsedDocument "
18                f"source={self.metadata.get('source', 'unknown')}, "
19                f"azure={self.is_azure()}>"
20            )

```

Listing 3.1. Class for storing parsed documents.

Unsupported formats included .xlsx, .docx, and .pptx. For .xlsx files, the open-source library pandas was used to extract and process the data. Each Excel sheet was read and converted to a string format, then saved to the content field of the Python class, alongside the metadata.

For .pptx files, the Presentation class from the python-pptx library was used to parse the content. The slides were iterated through, and text was extracted from each slide and saved to the content field. Metadata was appended at the end of the process, as with the .xlsx files.

For .docx files, the Document Intelligence API was applied. However, it was unable to extract key-value pairs from this format. This limitation was not explicitly handled; the decision was made to proceed without key-value pairs for .docx documents.

Once the contents of each file within a specified project directory were parsed, the entire data structure was exported to a JSON file. This approach eliminates the need to repeatedly call the Document Intelligence API for previously parsed documents, thereby reducing costs and time. In subsequent runs, the JSON file could simply be deserialized into the Python classes, reducing API traffic.

3.2.2 Chunking

The first step in preparing documents for embedding and retrieval was dividing the extracted content into manageable text chunks. Each document, stored as a parsed object, was processed to extract structured fields—paragraphs, key-value pairs, and tables—using. These fields were combined into a single string used for chunking. If the document was not processed via Azure Document Intelligence, its plain text content served as the fallback.

Chunking was performed using the RecursiveCharacterTextSplitter from LangChain, which splits text based on a character limit while respecting natural boundaries such as sentence and paragraph ends (Langchain 2023). A chunk size of 1000 characters with an overlap of 100 characters was used to preserve context between adjacent segments.

Unlike the earlier stage where documents were stored in a class that preserved all metadata and parsed objects, this phase transitioned to a more compact representation. Each resulting chunk was saved with essential metadata and a unique identifier that included the sanitized document ID and chunk index. To ensure retrievability and traceability, paragraph text, key-value pairs, and table content were stored only in the first chunk of each document instance when available.

Additionally, embeddings for each chunk were generated during this step and included in the output, enabling downstream search and retrieval functionalities. This format allowed

precise identification of source documents and their corresponding content segments in the responses generated by the language model.

3.2.3 Uploading chunked documents into indexes

After extracting and chunking the text, the next step was to upload the chunks into an Azure AI Search index. Each document chunk, along with its corresponding metadata (e.g., document name, project ID, chunk ID, and chunk type), was formatted into a structure compatible with Azure AI Search's indexing API. The metadata ensured that chunks could be filtered or prioritized during retrieval based on query context, such as project-specific or buyer-specific requirements.

In order to get the most out of the system and maintain context, separate indexes were created for project-specific, universally applicable, and buyer-specific documents. This structure ensured that queries targeting a specific project or orderer only retrieved chunks from the relevant index, while universally applicable documents, such as Yleiset inframalli vaatimukset (YIV, Common InfraBIM Requirements), were stored in a shared index for separate searching.

The upload process involved initializing an Azure AI Search client using the Python SDK, authenticating with API keys, and creating or updating an index with a schema that included fields for different bits of metadata and the content of the extracted documents, including embeddings generated for each chunk to enable vector-based similarity searches. The embeddings were generated using OpenAI's text-embedding-ada-002 model via Azure. To handle large datasets, batch uploads were implemented to optimize performance, processing up to 100 chunks per batch to minimize API call overhead.

Enabling vector and semantic search

To enable efficient retrieval of relevant document chunks, the system implemented hybrid and semantic search capabilities using Azure AI Search. Hybrid search combines vector-based and keyword-based retrieval to leverage the vector similarities and exact term matching, ensuring comprehensive results (Microsoft Azure 2025b). Semantic search, on the other hand, utilizing Azure AI Search's semantic ranking, prioritized contextually relevant chunks by reranking results based on query intent (Microsoft Azure 2025c). These approaches are essential for the Retrieval-Augmented Generation (RAG) system to deliver accurate and context-aware responses.

The hybrid search process generated a vectorized query using OpenAI's text-embedding-ada-002 model to retrieve the top ranking nearest neighbors based on embeddings stored in the index. At the same time, it performed a keyword-based search using the query text, selecting fields like id, source, type, and combined text chunk. Results included source,

text, and search scores for further processing.

The semantic search process used Azure AI Search's semantic query type with a default configuration, applying extractive query answering and captioning to enhance result relevance. It selected the same fields as the hybrid search, with results prioritized by a reranker score or fallback search score to focus on contextually significant chunks.

The retrieval process combined results from hybrid and semantic searches across the different indexes. Scores were normalized to ensure a fair comparison, and the top ranking chunks from each index were selected based on normalized scores.

3.3 RAG System Implementation

The RAG system combines document preprocessing, retrieval, and response generation to deliver contextually relevant answers to user queries. It uses the preprocessed document corpus, embeddings from OpenAI's text-embedding-ada-002 model, and large language models (LLMs) to generate responses based on retrieved information. The system relies on Azure AI Search's hybrid retrieval capabilities for both vector-based and keyword-based searches alongside the semantic retrieval, ensuring comprehensive access to information. The system architecture is outlined in Figure 3.1.

3.3.1 Retrieval

Relevant documents were retrieved using Azure AI Search's hybrid and semantic search capabilities. The dataset was organized into three indexes: project-specific, universally applicable, and buyer-specific, with documents sorted into their respective indexes based on their type. Each index was queried independently to ensure contextually relevant results and prevent cross-retrieval of unrelated documents.

For each query, hybrid search combined keyword-based and vector-based methods. The user's query was passed as text for keyword-based search to retrieve the best matching document chunks, while the query was also embedded using OpenAI's text-embedding-ada-002 model for vector-based similarity search, employing the HNSW (Hierarchical Navigable Small World) algorithm for efficient vector retrieval. Additionally, semantic search was applied using the default Azure AI Search semantic configuration, processing the query as text to capture contextual meaning. Results from both hybrid and semantic searches were combined for further processing or ranking.

3.3.2 Generation

After retrieving relevant document chunks using Azure AI Search's hybrid and semantic search, the generation phase synthesized these into coherent, contextually accurate

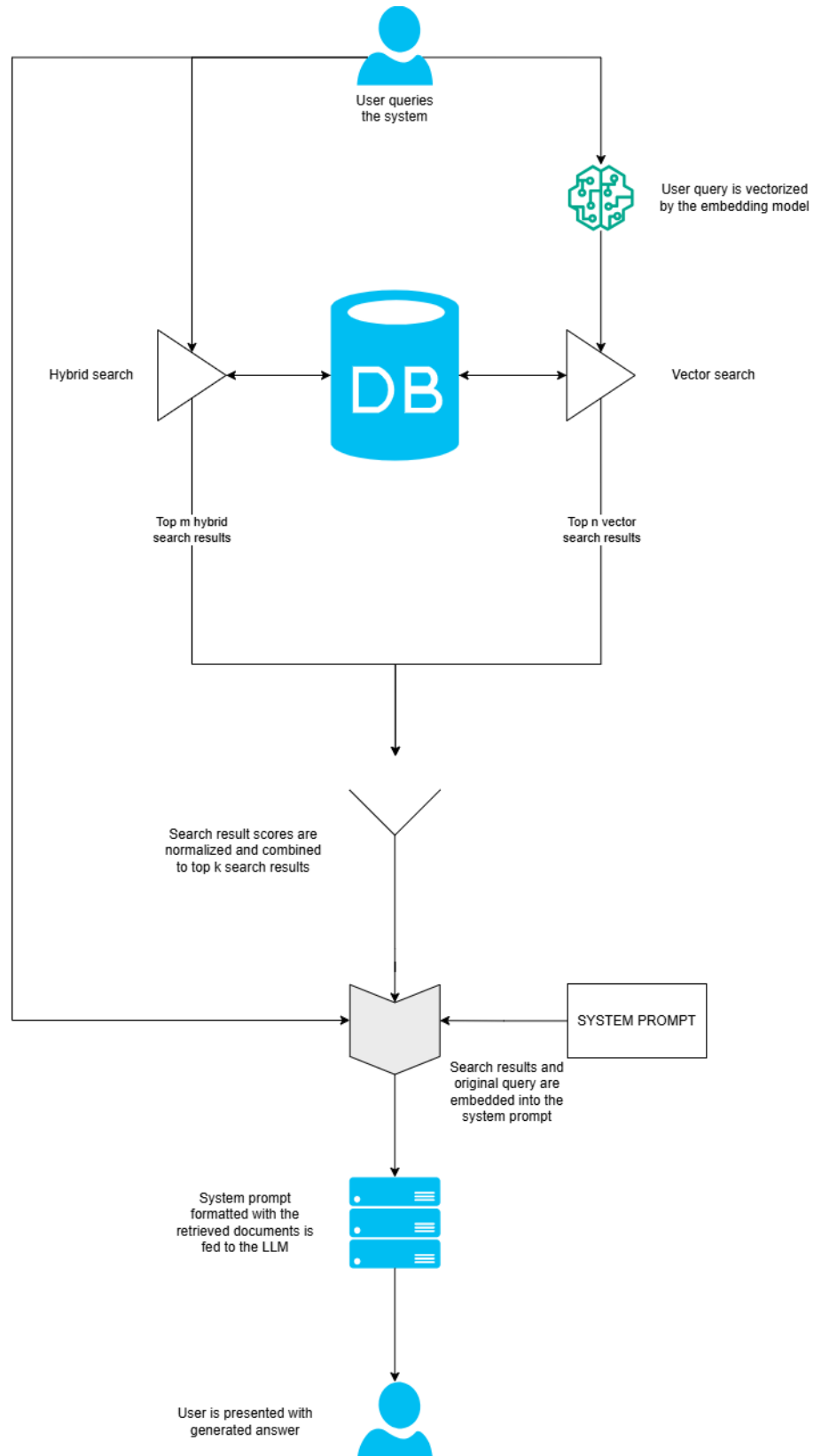


Figure 3.1. Outline of the RAG system used in the study.

responses tailored for Finnish infrastructure projects. The retrieved chunks, accompanied by metadata (e.g., document ID, chunk type, source), were passed to OpenAI's GPT

model via Microsoft Azure. A system prompt guided the model to prioritize information from the retrieved chunks while leveraging its knowledge of Finnish infrastructure project requirements to identify discrepancies or gaps in the data.

The system prompt was structured as follows to ensure precise and relevant responses:

- For general queries on information management or infrastructure project requirements, the model used both retrieved chunks and its knowledge of Finnish practices, clearly indicating when general knowledge was applied.
- For project-specific queries, the model identified and presented requirements in a structured format (e.g., lists or tables), distinguishing mandatory from optional requirements, citing specific documents (by ID and name), and highlighting any conflicts or ambiguities.
- The prompt instructed the model to ground responses in the provided document context, use general knowledge only when necessary, and explicitly state the source of information (document context, general knowledge, or inference).
- Metadata filters ensured only chunks from the relevant index (project-specific, buyer-specific, or universally applicable, such as YIV) were used, maintaining context and avoiding irrelevant data. For instance, a query about a specific project used only chunks from that project's index and the universal index, ensuring accuracy.

A key challenge was balancing fidelity to retrieved content with response fluency. Over-reliance on chunk text risked repetitive or rigid answers, while excessive creativity could introduce errors. The prompt was refined to promote concise, natural language while strictly adhering to the retrieved data. If no relevant chunks were found, the model returned a neutral response, such as “No relevant information found,” to ensure reliability. Additionally, responses included citations to document sources for transparency.

Performance was optimized by limiting the number of chunks (e.g., top-5 by relevance) passed to the model, reducing latency while maintaining quality. Caching frequent query-response pairs further improved efficiency. This approach ensured the RAG system delivered accurate, contextually relevant responses for Finnish infrastructure projects, effectively combining Azure AI Search's retrieval with OpenAI's generation capabilities.

3.4 Evaluation

The system was evaluated to ensure accurate, relevant, and efficient responses to user queries, with optimization efforts addressing identified limitations. The evaluation focused on retrieval accuracy, response quality, and system performance, relying on human metrics for assessing response generation.

As the query was executed, the data from it was simultaneously saved to a JSON file.

This data included

- Timestamp of the query
- The query itself
- The answer of the LLM
- Search results for hybrid and semantic search
 - Document ID
 - Document source
 - Retrieved chunk
 - Search score
 - Normalized score
 - Latency.

3.4.1 Retrieval Evaluation

Using Azure AI Search, two search methods, hybrid and semantic, were compared based on relevance scores. This evaluation aimed to determine which method better supported accurate and actionable responses, aligning with the thesis's focus on infrastructure requirements like Building Information Modeling (BIM) standards.

Relevance Scores

Each search method returned relevance scores for retrieved chunks, indicating how well they matched the query. The hybrid search, combining keyword and vector-based matching, produced `@search.score`, a positive float reflecting term frequency and embedding similarity. For example, a chunk from a project specification document might score 2.8. The semantic search, leveraging a reranking model for contextual understanding, yielded `@search.reranker_score` that ranged from 0 to 4. A chunk from a YIV guideline discussing BIM standards might score 3.5.

Direct comparison of raw scores was not feasible due to differing scales and calculations. To address this, a min-max scaling normalization was applied using equation mapping scores to $[0, 1]$ within each query. Normalized scores enabled relative comparison. A Python script analyzed these scores, generating box plots to visualize distributions. The normalization was done using this equation

$$S_{\text{norm}} = \frac{S - S_{\text{min}}}{S_{\text{max}} - S_{\text{min}}}, \quad (3.1)$$

where S_{norm} is the normalized score, scaled to $[0, 1]$, representing the relative relevance

of a retrieved document chunk, S is the raw score of a chunk, such as `@search.score` for hybrid search or `@search.reranker_score` for semantic search, S_{\min} is the minimum raw score among all chunks for a query, and S_{\max} is the maximum raw score among all chunks for a query. If $S_{\max} = S_{\min}$, S_{norm} was set to 0.0 to avoid division by zero.

Search Latency

In addition to evaluating the relevance of search results, the efficiency of the search operation was assessed by measuring the search latency. Latency, in this context, refers to the time delay, in seconds, between issuing a query and receiving the results. This is a critical performance metric, as it provides insight into the responsiveness of the search system under varying conditions and configurations.

Limitations and Considerations

Several limitations were identified. Although normalized scores facilitate comparison, they are query-specific, making cross-query analysis unreliable. For instance, a score of 0.8 may represent different raw scores across queries, because the values used for normalization across queries were different. Semantic search sometimes defaulted to `@search.score` when reranking failed, potentially distorting results. Moreover, the evaluation did not consider edge cases such as ambiguous queries or unsupported document formats, which could impact retrieval quality.

3.5 Generation Evaluation

The generation phase of the system was evaluated to assess the quality of responses for queries. Due to the absence of standardized ground-truth data, such as predefined correct answers for queries related to BIM standards or YIV guidelines, automated metrics like BLEU or ROUGE could not be reliably applied. Instead, expert grading was selected as the evaluation method. Three experts in information modeling, each with over five years of experience in BIM and infrastructure projects, were chosen to review the generated responses across 13 queries. This human-based approach ensured a nuanced assessment tailored to the specific context of infrastructure projects.

The evaluation involved grading each response on six criteria:

Table 3.1. Evaluation criteria for the generation phase of the RAG system, with descriptions, scoring guidelines, and examples.

Criterion	Explanation
Relevance	<p>Relevance to Query:</p> <p><i>Question:</i> How well does the response address the user's query?</p> <p><i>Description:</i> Assess whether the response directly answers the query (e.g., "What are the information management requirements of this project?") without irrelevant or off-topic information.</p> <p><i>Scoring Guidelines:</i></p> <p>1: Completely irrelevant or unrelated to the query.</p> <p>5: Partially relevant, addresses some aspects but includes irrelevant details.</p> <p>10: Fully relevant, directly and concisely answers the query.</p>
Accuracy	<p>Accuracy of Information:</p> <p><i>Question:</i> How accurate is the response based on the retrieved documents and domain knowledge?</p> <p><i>Description:</i> Check if the response correctly reflects information from project-specific, universal (e.g., YIV), or buyer-specific documents and aligns with Finnish infrastructure standards.</p> <p><i>Scoring Guidelines:</i></p> <p>1: Contains major factual errors or misinterpretations.</p> <p>5: Mostly accurate but includes minor errors or unsupported claims.</p> <p>10: Fully accurate, consistent with retrieved chunks and domain standards.</p>
Completeness	<p>Completeness of Response:</p>

Continued on next page

Table 3.1 continued from previous page

Criterion	Explanation
	<p><i>Question:</i> How comprehensively does the response cover the query's requirements?</p> <p><i>Description:</i> Evaluate whether the response includes all necessary details (e.g., mandatory vs. voluntary requirements, source citations) without omitting key aspects.</p> <p><i>Scoring Guidelines:</i></p> <p>1: Misses most required information, highly incomplete.</p> <p>5: Covers some aspects but omits important details or sources.</p> <p>10: Fully comprehensive, includes all relevant details and sources.</p>
Coherence/Clarity	<p>Coherence and Clarity:</p> <p><i>Question:</i> How clear and well-structured is the response?</p> <p><i>Description:</i> Assess the response's readability, logical flow, and organization (e.g., use of lists or tables as requested in the system prompt).</p> <p><i>Scoring Guidelines:</i></p> <p>1: Confusing, poorly structured, or unreadable.</p> <p>5: Moderately clear but lacks structure or has minor inconsistencies.</p> <p>10: Highly clear, logically structured, and easy to understand.</p>
Usability	<p>Usability for Stakeholders:</p> <p><i>Question:</i> How useful is the response for infrastructure project stakeholders (e.g., engineers, project managers)?</p> <p><i>Description:</i> Evaluate whether the response provides actionable information that stakeholders can apply (e.g., clear requirements for project implementation).</p> <p><i>Scoring Guidelines:</i></p> <p>1: Not useful, lacks actionable insights or context.</p>

Continued on next page

Table 3.1 continued from previous page

Criterion	Explanation
	<p>5: Moderately useful, provides some actionable information but lacks specificity.</p> <p>10: Highly useful, directly applicable to project needs with clear guidance.</p>
Source Integration	<p>Source Integration Quality:</p> <p><i>Question:</i> How well does the response integrate and prioritize information from retrieved sources (e.g., project-specific, universal)?</p> <p><i>Scoring Guidelines:</i></p> <p>1: Ignores retrieved sources or uses incorrect ones.</p> <p>5: Uses some retrieved sources but prioritizes incorrectly or misses key ones.</p> <p>10: Seamlessly integrates and prioritizes sources (e.g., project-specific over universal when appropriate).</p>

These criteria were selected to capture the practical applicability and technical correctness of the responses, reflecting the needs of infrastructure professionals. Experts provided both numerical scores and qualitative comments to offer detailed feedback on the system's strengths and weaknesses. This method was chosen to account for the complexity and variability of the queries, ensuring the evaluation reflected real-world usability and alignment with project-specific requirements, which automated tools might overlook.

4. RESULTS

The performance analysis of the system was divided into distinct components to provide a comprehensive evaluation. This approach involved analyzing the retrieval phase first, followed by the generation phase, and concluding with the system's overall performance. The results presented here focus on the system's capability to handle Finnish infrastructure project queries, particularly in parsing information requirements related to information modeling.

4.1 Retrieval Phase

The retrieval phase was evaluated to determine how effectively the system retrieved relevant document chunks using Azure AI Search's hybrid and semantic search methods. The evaluation focused on two metrics: normalized relevance scores and search latency derived from query data stored in a json file. These metrics were analyzed using a Python script to generate visualizations, providing insights into the comparative performance of hybrid and semantic searches.

4.1.1 Normalized Relevance Scores

Relevance scores indicated how well retrieved chunks matched each query. Although they don't give an exact measure of how well the search performed, they give an idea of how confident AI Search was about the results (Microsoft Azure 2025e). Hybrid searches, combining keyword and vector-based matching, produced `@search.score` [0, 1], while semantic searches, using a reranking model, yielded `@search.reranker_score` [0, 4]. Due to differing score scales, min-max scaling was applied to normalize all scores to [0, 1], as defined in Equation (3.1). Normalized scores were saved as `normalized_score` in the aforementioned JSON.

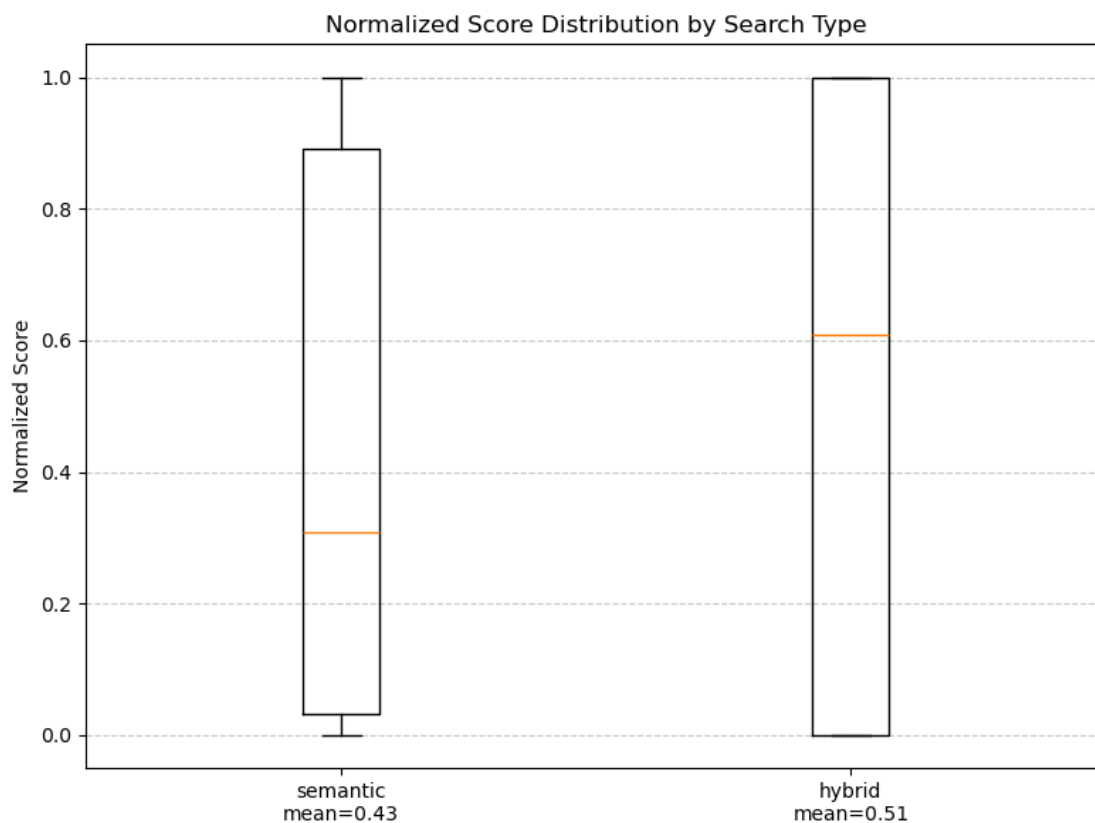


Figure 4.1. Normalized score distribution for hybrid and semantic searches.

Analysis of normalized scores across queries showed hybrid searches averaged 0.511, slightly higher than semantic searches at 0.433. This suggests hybrid searches were more precise for targeted queries, likely due to strong keyword matching. Semantic searches, while marginally less precise, excelled for more conceptual queries which needed leveraging of contextual understanding.

4.1.2 Search Latency

Search latency, measured in seconds, assessed the efficiency of each search method. Latency data captured the time taken to retrieve chunks for each query. Semantic searches averaged 0.111 seconds, faster than hybrid searches at 0.191 seconds, as shown in Figure 4.2.

This 42% reduction in latency for semantic searches indicates greater efficiency, likely due to the fact that hybrid search performs two searches, keyword- and vector-based, at once.

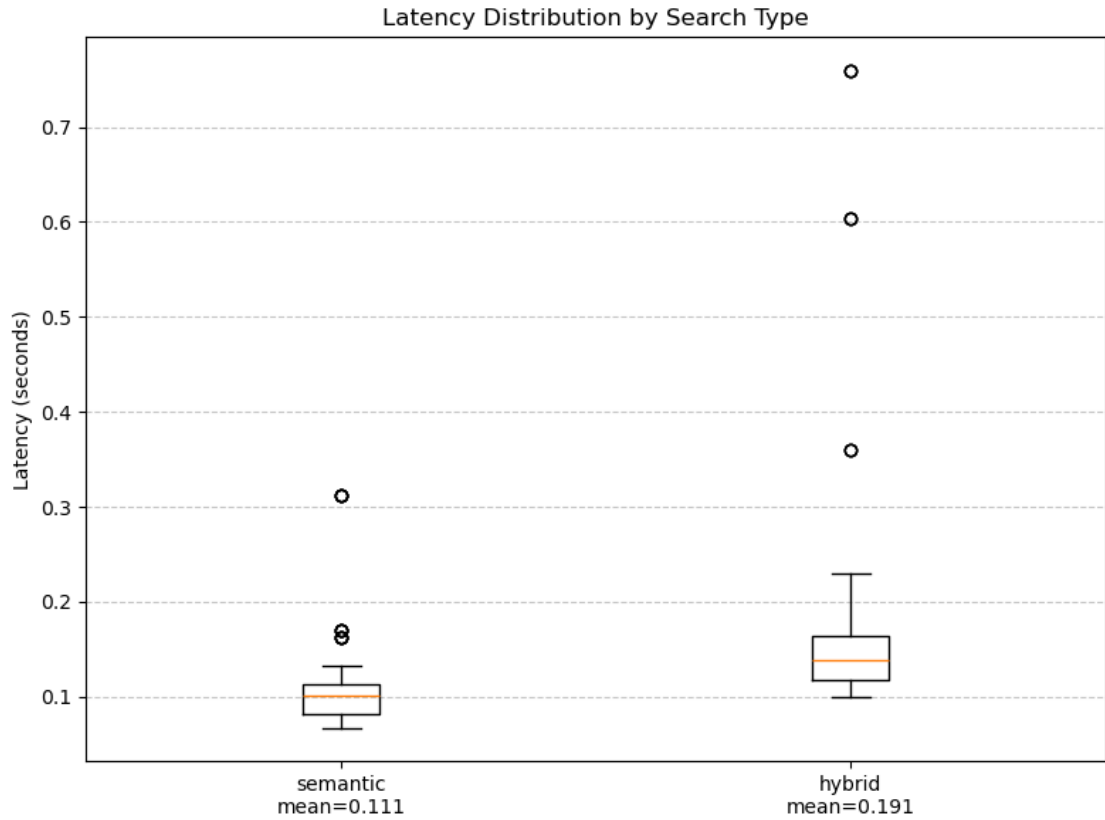


Figure 4.2. Latency for hybrid and semantic searches.

4.2 Generation Evaluation

The generation phase of the system was evaluated to assess the quality of responses using expert grading due to the lack of ground-truth data. Experts in information modeling assessed responses across queries. Each response was graded on six criteria: relevance, accuracy, completeness, coherence/clarity, usability, and source integration, scored from 1 to 10, where 1 indicates poor performance and 10 indicates excellence.

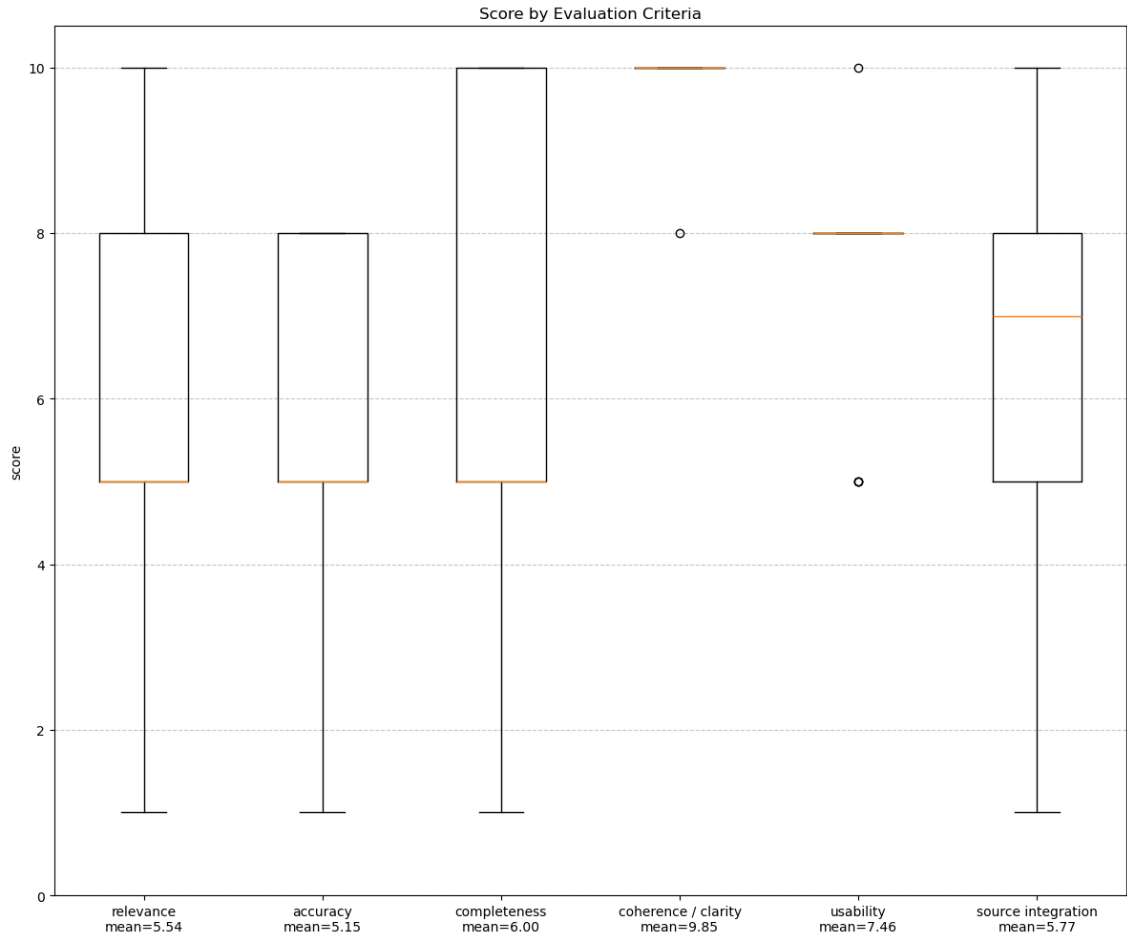


Figure 4.3. Scores by evaluation criteria for generations.

Average scores across the queries showed strengths and weaknesses. Coherence/clarity scored highest at 9.85, reflecting consistent readability. Relevance averaged 5.54, accuracy 5.15, and completeness 6.10, indicating challenges with hallucination and missing details. Usability scored 7.46, and source integration 5.77, showing moderate practical value and data integration. These results are visualized in Figure 4.3, highlighting the balance between clear structure and content accuracy.

Expert comments provided qualitative insights into the system’s performance:

Table 4.1. Summary of expert comments on LLM-generated answers, organized by evaluation criteria.

Criterion	Evaluation	Comments
Relevance	Positive	Good recognition of key points, such as quality assurance goals and information model coordinator reference requirements.

Continued on next page

Table 4.1 continued from previous page

Criterion	Evaluation	Comments
	Negative	Inclusion of irrelevant tasks or reduced relevance to the specific project scope.
Accuracy	Positive	Accurate identification of key details in some cases, such as database details and quality assurance targets.
	Negative	Frequent hallucination of content, which lowers the overall accuracy.
Completeness	Positive	Good task lists encouraged further exploration and identified hidden requirements.
	Negative	Missing critical details, including design phases, stakeholders and database details.
Coherence/Clarity	Positive	Structure and clarity were consistently praised, with responses being clear and easy to understand.
	Negative	No notable issues; responses were generally well-structured.
Usability	Positive	Identification of useful insights, such as database details, enhanced practical application.
	Negative	Lack of project-specific tailoring and inclusion of irrelevant tasks reduced practical relevance and trustworthiness.
Source Integration	Positive	Effective integration in some cases, such as identifying new database details and information model coordinator requirements

Continued on next page

Table 4.1 continued from previous page

Criterion	Evaluation	Comments
	Negative	Failure to address discrepancies, such as information management desing requirements, and to prioritize buyer's quality requirements over YIV, weakened source integration.

4.3 System evaluation

The whole system was evaluated by integrating retrieval and generation phases using data from queries and human-assessed generation metrics. This assessment considered retrieval averages, generation quality, and correlations between components to determine system effectiveness.

Correlations revealed a weak negative link between semantic and hybrid retrieval (-0.26), suggesting differing retrieval strategies, a moderate positive link between generation score and semantic retrieval score (0.40), indicating semantic's influence on output quality, and a weaker positive link with hybrid (0.28), reflecting hybrid's precision contribution. The system excels in producing coherent responses for time-sensitive queries with semantic retrieval and precise outputs with hybrid.

5. DISCUSSION

This chapter discusses the findings presented in Chapter 4, exploring their implications for the Retrieval-Augmented Generation system's application to Finnish infrastructure projects, with a focus on enhancing information modeling workflow.

5.1 Retrieval findings

The retrieval phase findings from the evaluation reveal key insights into the performance of the Retrieval-Augmented Generation (RAG) system for Finnish infrastructure projects. Hybrid searches demonstrated higher average normalized scores of 0.511 and a median of 0.608, compared to semantic searches with an average of 0.433 and a median of 0.308. This suggests that hybrid searches, combining keyword and vector-based matching, better prioritize relevant document chunks, particularly for targeted queries concerning specific projects and/or their phases. The higher median for hybrid searches indicates a stronger concentration of precise results, which could enhance the system's effectiveness for detailed infrastructure specifications.

Latency results further highlight a trade-off between speed and precision. Semantic searches completed in an average of 0.111 seconds, faster than the 0.191 seconds for hybrid searches, making them suitable for time-sensitive tasks such as rapid lookups of BIM standards. However, the slower hybrid searches' precision advantage suggests a need to balance speed and accuracy based on project demands. At this point, the latency was not considered in the research and more focus was directed to the quality of searches.

These findings imply that the choice of retrieval method should align with specific infrastructure needs. Semantic searches offer efficiency and wider coverage, supporting quick decision-making, whereas hybrid searches provide precision, crucial for compliance with detailed standards like YIV. This means that more broad queries about information modeling should be directed to prioritize semantic searches over keyword or vector-based searches. Future work could explore optimizing the choice of search method based on the query style.

5.2 Generation findings

The generation phase evaluation provides critical insights into the quality of responses produced by the RAG system for Finnish infrastructure project queries. Expert grading across queries revealed a wide range of performance across the six criteria, with average scores as follows:

- relevance at 5.38
- accuracy at 5.15
- coherence/clarity at 9.85
- completeness at 5.85
- usability at 7.15
- integration at 6.15

These results, while showcasing strengths in response structure, highlight significant challenges in content quality that impact the system's ability to support infrastructure professionals with reliable and actionable information.

The high coherence/clarity score of 9.85, with most queries scoring 10, underscores the system's ability to produce well-structured and readable responses. The clarity was consistently praised, noting that the responses were clear and easy to understand and well-structured to match the question. This strength suggests that the LLM effectively organizes retrieved information, making it accessible even for complex queries about BIM standards or YIV compliance. However, this clarity is undermined by lower scores in other criteria, particularly accuracy and relevance, which limit the overall utility of the responses.

Accuracy, averaging 5.15, was heavily impacted by frequent hallucination, where the system generated incorrect or fabricated content. It was noted that there were entirely hallucinated parts in some answers. This means that trusting the model output will be challenging even though it has been instructed to only refer to project details it can find in the retrieved chunks. These errors led to scores as low as 1 for some queries, indicating a critical flaw in the system's ability to align with retrieved documents and domain knowledge. For infrastructure projects, where accuracy is paramount to ensure compliance with requirements, such hallucinations could lead to costly misinterpretations, such as applying non-existent requirements or missing important ones.

Relevance, with an average of 5.38, was similarly affected by the inclusion of irrelevant tasks. For example, one of the query answers included a quality assurance plan that was not part of the project scope, and another answer introduced worksite tasks despite the absence of a general contractor at this stage of the project. These issues suggest that the system struggles to filter out extraneous information and to spot missing information, reducing its effectiveness for targeted queries. This is a challenging issue because the

system would need to place strong trust in the search component to confidently conclude that certain content is truly missing from the documents, rather than simply assuming that the search failed to locate it. It would also mean that every time the search is unable to return the relevant information, the LLM would report that information as missing in the documents. In contrast, when the system correctly identified key points, such as quality assurance goals in one query or model coordinator reference requirements in another, relevance scores improved, highlighting the potential for better performance with improved filtering mechanisms.

Completeness scored 5.85, reflecting frequent omissions of critical details. It was pointed out that missing elements in answers make it once again hard to trust the system. These gaps mean that responses often fail to provide a full picture, which is essential for stakeholders needing comprehensive guidance. However, the system showed promise in identifying hidden requirements and producing task lists that encouraged further exploration. This suggests that with better retrieval integration, completeness could improve.

Usability, at 7.15, indicates moderate practical value for stakeholders like engineers and project managers. Positive feedback included the identification of useful insights, such as new database details in one query, which enhanced applicability. Yet, the lack of project-specific tailoring, as noted in one query ("lacking fitting to the specific project"), and the inclusion of irrelevant tasks reduced usability. For Finnish infrastructure projects, where responses must directly support tasks like IFC format compliance, this lack of specificity limits the system's immediate utility.

Source integration, averaging 6.15, revealed inconsistent use of retrieved documents. While the system effectively integrated sources in some cases, it often failed to prioritize different sources appropriately. For instance, one query overlooked buyer's quality requirements despite their precedence over the YIV, and one answer missed a discrepancy in information management planning requirements between documents. This inconsistency suggests that the system needs better logic to weigh project-specific sources over universal ones, ensuring responses align with the most relevant standards. This behavior could be fine-tuned through adjustments to the system prompt, but more significant changes would require modifications to the underlying system architecture.

These findings highlight a disconnect between the system's retrieval and generation phases. Even when retrieval provides relevant chunks (e.g., hybrid's 0.511 normalized score), the generation phase struggles to accurately and completely synthesize this information into usable responses. Hallucination and lack of tailoring indicate a need for improved system architecture or fine-tuning, possibly incorporating more retrieved chunks to the model to enhance outputs. Additionally, enhancing source integration logic to prioritize buyer-specific requirements could address the gaps in accuracy and completeness, making the system more reliable for infrastructure applications like BIM standards compliance.

5.3 Future work and development

5.3.1 Technical enhancements

Based on the findings of the study, retrieval accuracy should be the main focus of system improvements. Despite hybrid retrieval methods achieving higher normalized scores, their effectiveness was limited by the downstream generator's failure to consistently utilize relevant information. Improvements in the retriever module, such as fine-tuning retrieval granularity, optimizing retrieval stride, or employing adaptive reranking, could enhance alignment between retrieved content and query intent (Sharma 2025; Birkins 2024).

One promising direction is the use of modular architectures that combine symbolic and neural retrieval methods, such as Re2G, which introduces a reranking layer to bridge traditional keyword-based methods (like BM25) and dense retrievers. This layered approach has been shown to improve retrieval precision and ranking quality without requiring retraining of retrievers or generators, making it a practical enhancement path for infrastructure-specific document corpora (Glass et al. 2022).

Complementary to this, filtering mechanisms that eliminate low-utility or noisy chunks before generation remain critical. Strategies like retrieval clues, sentence-level Focus Mode, and utility-based chunk scoring have been shown to significantly reduce hallucinations and improve contextual faithfulness (Birkins 2024). In the Finnish infrastructure domain, these would help the generator anchor responses more reliably to source material, especially when handling legally or technically significant clauses from standards such as YIV or project-specific IFC requirements.

Query optimization should also be prioritized. Techniques such as query intent recognition and query decomposition can be used to improve retrieval coverage and focus by breaking complex questions into simpler subqueries, a strategy that has been shown to improve robustness and reduce hallucinations (Birkins 2024). Similarly, contrastive in-context learning (C-ICL), presenting the model with both correct and misleading examples, can help reduce overgeneration of plausible-sounding but incorrect content, a common challenge identified in the evaluation.

From a system design perspective, incorporating elements of hybrid RAG architectures, where retrieval and generation interact iteratively, could address current shortcomings. For instance, a multi-step or retrieval-aware generation loop may allow the system to refine its understanding of missing or ambiguous requirements during generation. Models like inner monologue RAG (IM-RAG) and corrective RAG (CRAG) demonstrate this potential by dynamically re-triggering retrieval based on evolving generation context (Sharma 2025).

To further mitigate hallucination and domain misalignment, faithfulness-aware decod-

ing strategies such as critique-generate loops or self-verification (e.g., SELF-RAG) offer promising improvements. These techniques ensure that generation stays anchored to evidence found in retrieved content, which is essential for infrastructure use cases where factual errors could compromise regulatory compliance or contractual obligations (Asai et al. 2023).

Additionally, metadata-aware prompting and dynamic query reformulation, where retrieval metadata or domain priors guide generation, could improve response specificity. These strategies help the model remain on-topic and maintain contextual alignment, especially in multi-document or multilingual corpora (Sharma 2025).

Finally, evaluation methods should be formalized and scaled. While this thesis relied on manual expert scoring, automated retrieval-aware evaluation frameworks like RAGAS or ARES can support scalable, consistent assessment of response faithfulness and context alignment, especially useful in production pipelines (Sharma 2025).

In summary, the next development cycle should prioritize:

- enhancing retriever-generator alignment through hybrid retrieval, reranking, utility filtering, and query decomposition,
- reducing hallucination via critique-based generation and contrastive in-context learning,
- enabling retrieval-generation feedback loops through iterative or retrieval-aware generation,
- and formalizing evaluation through retrieval-aware automatic scoring frameworks.

These improvements could significantly increase the system's reliability and trustworthiness for Finnish infrastructure information modeling. However, they also introduce greater system complexity and computational demands, potentially reducing scalability and requiring careful cost-benefit evaluation in practical deployments.

5.3.2 Implications for industry adoption

Implementing these enhancements in a real-world company setting would present both opportunities and trade-offs. From a benefits standpoint, improved accuracy, relevance, and completeness of generated content could significantly reduce manual information retrieval efforts, improve compliance with domain-specific standards like YIV, and accelerate decision-making processes during project planning and coordination.

However, the introduction of more complex system components, such as iterative retrieval loops, reranking layers, and agentic RAG solutions, could raise operational costs (Sharma 2025; Singh et al. 2025). These include increased computational requirements, longer inference times, and the need for more robust maintenance workflows. Companies

may need to invest in additional infrastructure (e.g., GPU clusters or API credits), system integration, and model supervision pipelines to manage model behavior and quality.

Furthermore, domain-specific fine-tuning, prompt engineering, and continuous evaluation workflows would likely require a dedicated team or expertise, increasing the human resource burden. As a result, companies must carefully evaluate whether the improvements in quality and trustworthiness justify the added complexity and costs, particularly for use cases where fast iteration and scalability are essential.

That said, organizations already engaged in digital modeling workflows and information requirement management would be well-positioned to benefit from such a system, especially if the RAG component is integrated gradually as a decision-support module, rather than a fully autonomous solution. This modular integration approach could help manage risks while still delivering meaningful productivity gains.

6. CONCLUSION

The analysis of the Retrieval-Augmented Generation (RAG) system reveals its promising yet currently limited potential to support Finnish infrastructure projects, particularly in addressing BIM standards and YIV compliance. This thesis examined both retrieval and generation components, identifying strengths that align with infrastructure information workflows, while highlighting critical weaknesses that hinder dependable system performance.

The retrieval evaluation showed that hybrid search methods, which achieved a higher average normalized score, outperformed semantic-only searches in prioritizing relevant content, particularly for detailed, compliance-driven tasks like requirement analysis. However, these results should not be interpreted too literally as Azure AI Search's relevance scores are not absolute indicators of accuracy or correctness. They serve as internal heuristics rather than ground-truth metrics, and should be supplemented with expert judgment or external validation when assessing the reliability of retrieved content.

The generation phase presented more serious challenges. While expert evaluations praised the clarity and structure of the responses, issues with factual accuracy, relevance, and completeness were consistent. Hallucinations and critical omissions undermined the reliability of answers, making them potentially unsuitable for compliance-critical decisions. Although the system's outputs were moderately usable in early planning stages, their current limitations in grounding and source fidelity restrict practical application in high-stakes contexts.

The broader implications for infrastructure information modeling are mixed. The system's structured outputs can help standardize communication across project roles. However, factual inconsistencies and irrelevant content pose risks in tasks where accuracy and domain specificity are non-negotiable. While moderate usability offers a base for actionable outputs, improvements in hallucination control, source prioritization, and domain adaptation are necessary before operational deployment is feasible.

This study was constrained by the use of expert scoring in the absence of ground-truth reference answers, which introduced subjectivity. To improve future evaluations, expanding the query set and adopting retrieval-aware automatic metrics (e.g., ARES or RAGAS) is recommended. Enhancing retrieval fidelity through reranking or context filtering (e.g.,

Re2G, SEER), and integrating critique-based generation mechanisms (e.g., SELF-RAG), could significantly reduce hallucinations and increase source-grounded accuracy. These enhancements, while beneficial for reliability, may increase system complexity and inference costs, requiring a balance between precision and scalability.

In conclusion, this thesis provides a baseline assessment of RAG's suitability for infrastructure information modeling. It highlights where the system performs well, and where improvements are essential. With targeted architectural and process-level enhancements, RAG has the potential to evolve into a reliable support tool for managing and interpreting complex project documentation in infrastructure.

REFERENCES

- Asai, Akari et al. (Oct. 17, 2023). *Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection*. DOI: 10.48550/arXiv.2310.11511. arXiv: 2310.11511[cs]. URL: <http://arxiv.org/abs/2310.11511> (visited on 06/23/2025).
- Birkins, Joyce (2024). *6 Advanced RAG Optimization Strategies: Analysis of 14 Key Research Papers*. Accessed: 2025-06-23. URL: <https://medium.com/@joycebirkins/6-advanced-rag-optimization-strategies-analysis-of-14-key-research-papers-f12329975009>.
- buildingSMART Finland (Oct. 4, 2021). *YLEISET INFRAMALLIVAATIMUKSET YIV*. URL: https://wiki.buildingsmart.fi/fi/04_Julkaisut_ja_Standardit/YIV (visited on 01/30/2025).
- Burrell, Jenna (June 1, 2016). "How the machine 'thinks': Understanding opacity in machine learning algorithms". In: *Big Data & Society* 3.1. Publisher: SAGE Publications Ltd, p. 2053951715622512. ISSN: 2053-9517. DOI: 10.1177/2053951715622512. URL: <https://doi.org/10.1177/2053951715622512> (visited on 03/25/2025).
- Castelvecchi, Davide (Oct. 6, 2016). "Can we open the black box of AI?" In: *Nature News* 538.7623. Cg_type: Nature News Section: News Feature, p. 20. DOI: 10.1038/538020a. URL: <http://www.nature.com/news/can-we-open-the-black-box-of-ai-1.20731> (visited on 03/25/2025).
- Cheng, Jeffrey et al. (Sept. 17, 2024). *Dated Data: Tracing Knowledge Cutoffs in Large Language Models*. DOI: 10.48550/arXiv.2403.12958. arXiv: 2403.12958[cs]. URL: <http://arxiv.org/abs/2403.12958> (visited on 03/25/2025).
- DeepSeek-AI et al. (Jan. 22, 2025). *DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning*. DOI: 10.48550/arXiv.2501.12948. arXiv: 2501.12948[cs]. URL: <http://arxiv.org/abs/2501.12948> (visited on 02/07/2025).
- Finnish Transport Infrastructure Agency (2022). *Infra Model Requirements*. Accessed: March 14, 2025. URL: https://ava.vaylapilvi.fi/ava/Julkaisut/Vaylavirasto/vo_2022-32_inframallivaatimukset.pdf.
- Gao, Yunfan et al. (2024). *Retrieval-Augmented Generation for Large Language Models: A Survey*. arXiv: 2312.10997 [cs.CL]. URL: <https://arxiv.org/abs/2312.10997>.
- Glass, Michael R. et al. (2022). "Re2G: Retrieve, Rerank, Generate". In: *ArXiv abs/2207.06300*. URL: <https://api.semanticscholar.org/CorpusID:250391085>.
- Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. <http://www.deeplearningbook.org>. MIT Press.

Kaplan, Jared et al. (Jan. 23, 2020). *Scaling Laws for Neural Language Models*. DOI: 10.48550/arXiv.2001.08361. arXiv: 2001.08361[cs]. URL: <http://arxiv.org/abs/2001.08361> (visited on 03/25/2025).

Langchain (2023). *RecursiveCharacterTextSplitter*. URL: https://python.langchain.com/api_reference/text_splitters/character/langchain_text_splitters.character.RecursiveCharacterTextSplitter.html#langchain_text_splitters.character.RecursiveCharacterTextSplitter (visited on 04/24/2025).

Lewis, Patrick et al. (Apr. 12, 2021). *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*. DOI: 10.48550/arXiv.2005.11401. arXiv: 2005.11401[cs]. URL: <http://arxiv.org/abs/2005.11401> (visited on 03/25/2025).

Marcus, Gary (Feb. 19, 2020). *The Next Decade in AI: Four Steps Towards Robust Artificial Intelligence*. DOI: 10.48550/arXiv.2002.06177. arXiv: 2002.06177[cs]. URL: <http://arxiv.org/abs/2002.06177> (visited on 03/25/2025).

Martineau, Kim (Feb. 9, 2021). *What is retrieval-augmented generation (RAG)?* IBM Research. URL: <https://research.ibm.com/blog/retrieval-augmented-generation-RAG> (visited on 02/21/2025).

Microsoft Azure (Aug. 28, 2024). *Retrieval Augmented Generation using Azure Machine Learning prompt flow (preview) - Azure Machine Learning | Microsoft Learn*. URL: <https://learn.microsoft.com/en-us/azure/machine-learning/concept-retrieval-augmented-generation?view=azureml-api-2>.

– (2025a). *Azure.ai.documentintelligence.models.analyzeresult*. URL: <https://learn.microsoft.com/en-us/python/api/azure-ai-documentintelligence/azure.ai.documentintelligence.models.analyzeresult?view=azure-python> (visited on 05/26/2025).

– (May 27, 2025b). *Hybrid search in Azure AI Search*. URL: <https://learn.microsoft.com/en-us/azure/search/hybrid-search-overview> (visited on 06/04/2025).

– (May 8, 2025c). *Semantic search in Azure AI Search*. URL: <https://learn.microsoft.com/en-us/azure/search/semantic-search-overview> (visited on 06/04/2025).

– (Feb. 6, 2025d). *What is Azure AI Document Intelligence? - Azure AI services*. URL: <https://learn.microsoft.com/en-us/azure/ai-services/document-intelligence/overview?view=doc-intel-4.0.0> (visited on 04/24/2025).

– (2025e). *Index similarity and scoring in Azure AI Search*. URL: <https://learn.microsoft.com/en-us/azure/search/index-similarity-and-scoring> (visited on 06/02/2025).

Minaee, Shervin et al. (Feb. 20, 2024). *Large Language Models: A Survey*. DOI: 10.48550/arXiv.2402.06196. arXiv: 2402.06196[cs]. URL: <http://arxiv.org/abs/2402.06196> (visited on 01/14/2025).

Neelakantan, Arvind et al. (Jan. 25, 2022). *Introducing text and code embeddings*. URL: <https://openai.com/index/introducing-text-and-code-embeddings/> (visited on 03/25/2025).

- Retrieval Augmented Generation (RAG) and Semantic Search for GPTs* (2024). URL: <https://help.openai.com/en/articles/8868588-retrieval-augmented-generation-rag-and-semantic-search-for-gpts> (visited on 02/21/2025).
- Rudin, Cynthia (May 2019). “Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead”. In: *Nature Machine Intelligence* 1.5. Publisher: Nature Publishing Group, pp. 206–215. ISSN: 2522-5839. DOI: 10.1038/s42256-019-0048-x. URL: <https://www.nature.com/articles/s42256-019-0048-x> (visited on 03/25/2025).
- Şakar, Tolga and Hakan Emekci (Jan. 2025). “Maximizing RAG efficiency: A comparative analysis of RAG methods”. In: *Natural Language Processing* 31.1, pp. 1–25. ISSN: 2977-0424. DOI: 10.1017/nlp.2024.53. URL: https://www.cambridge.org/core/product/identifier/S2977042424000530/type/journal_article (visited on 02/14/2025).
- Sharma, Chaitanya (May 28, 2025). *Retrieval-Augmented Generation: A Comprehensive Survey of Architectures, Enhancements, and Robustness Frontiers*. DOI: 10.48550/arXiv.2506.00054. arXiv: 2506.00054[cs]. URL: <http://arxiv.org/abs/2506.00054> (visited on 06/23/2025).
- Singh, Aditi et al. (Feb. 4, 2025). *Agentic Retrieval-Augmented Generation: A Survey on Agentic RAG*. DOI: 10.48550/arXiv.2501.09136. arXiv: 2501.09136[cs]. URL: <http://arxiv.org/abs/2501.09136> (visited on 06/23/2025).
- What Are AI Hallucinations?* (Sept. 1, 2023). *What Are AI Hallucinations? | IBM*. URL: <https://www.ibm.com/think/topics/ai-hallucinations> (visited on 03/25/2025).
- What is RAG?* (2025). *What is RAG? - Retrieval-Augmented Generation AI Explained - AWS*. Amazon Web Services, Inc. URL: <https://aws.amazon.com/what-is/retrieval-augmented-generation/> (visited on 03/12/2025).
- Zhao, Wayne Xin et al. (Oct. 13, 2024). *A Survey of Large Language Models*. DOI: 10.48550/arXiv.2303.18223. arXiv: 2303.18223[cs]. URL: <http://arxiv.org/abs/2303.18223> (visited on 01/14/2025).