Stockholms universitet

Intelligent Requirement Review using Retrieval-Augmented Generation

Aristi Christoforou

Masteruppsats i matematisk statistik
Master Thesis in Mathematical Statistics

Masteruppsats 2026:11
Matematisk statistik
Juni 2026

www.math.su.se

Matematisk statistik
Matematiska institutionen
Stockholms universitet
106 91 Stockholm

Matematiska institutionen

Stockholm University

Mathematical Statistics
Stockholm University
Master Thesis **2026:11**
http://www.math.su.se

Intelligent Requirement Review using Retrieval-Augmented Generation

Aristi Christoforou*

June 2026

Abstract

This thesis studies the use of Retrieval-Augmented Generation (RAG) for requirement review in large-scale engineering systems. Requirement management platforms such as Polarion contain thousands of interconnected requirements, making manual review time-consuming, subjective, and difficult to scale.

In this work, requirements are converted into vector representations, and retrieval is treated as a ranking task based on similarity. Building on this, a RAG-based system is used to support requirement analysis by identifying relationships, inconsistencies, and missing constraints between requirements.

A central part of the thesis is the evaluation framework. Retrieval performance is treated as an estimation problem, where metrics such as Precision@k, Recall@k, and MRR@k are analyzed using bootstrap confidence intervals and uncertainty-aware comparison.

The results indicate that the retrieval component behaves as a high-recall candidate generator, while the RAG-based system extends this pipeline by adding structured interpretation of the retrieved requirements. At the same time, the findings show that evaluation design has an impact on the conclusions and that uncertainty must be taken into account when interpreting performance.

*Postal address: Mathematical Statistics, Stockholm University, SE-106 91, Sweden. E-mail: christoforouaristi1@gmail.com. Supervisor: Martina Favero.

Acknowledgments

I would like to express my sincere gratitude to my academic supervisor, Martina Favero, for her guidance and support throughout this thesis.

I would also like to thank my industrial supervisor at Hitachi Energy, Hasan Basri Celebi, for his valuable insights and for providing the industrial context for this work.

Finally, I would like to thank my family and close friends for their support during my studies.

Contents

1 Introduction

Modern industrial systems, such as automation platforms and power grids, rely on a large collection of requirements to characterize system behavior and the interactions between different components. These requirements are usually managed using specific tools such as Polarion. The repositories created using these tools have thousands of entries that are connected with each other and across multiple subsystems and products. Examples of requirements include statements such as safety constraints, communication interface specifications, synchronization requirements, or subsystem behavior descriptions.

One important challenge in large requirement repositories is ensuring the quality of these requirements. Even today, engineers need to identify issues such as inconsistencies, missing constraints, duplicated functionalities, and incorrect or incomplete links. This process is therefore still mostly performed manually. The problem is that, as the size and complexity of requirement repositories increase, it becomes more difficult to review them manually at scale. As a consequence, engineers also need to go through large volumes of text, often written by different people and at different levels of abstraction, which makes it hard to identify relationships between requirements.

In this context, requirement review refers to the process of systematically analyzing requirements to assess their quality and relationships.

Manual requirement review therefore has several important limitations. Firstly, it is time-consuming and can depend strongly on the experience level and focus of each individual engineer. As a result, the risk of missing important issues increases, and the quality and consistency of the review may depend on the expertise and experience of the individual engineer [1, 6]. Additionally, many relationships between requirements are not explicitly defined, which makes them more difficult to identify using traditional keyword-based search [9, 12]. This can cause relevant connections between requirements to remain hidden, which can make relevant requirement relationships more difficult to identify during review.

To address these challenges, recent studies have explored the use of ar-

tificial intelligence (AI) and large language models (LLM) in requirements engineering [1, 6, 11]. One important method in this area is semantic retrieval, where requirement texts are converted into vector representations and compared using similarity measures in order to retrieve relevant requirements beyond exact keyword overlap [9, 15, 3]. This is especially useful for requirement review, because semantically related requirements may use different terminology while still describing similar functions, constraints, or system behaviors.

Building on retrieval, retrieval-augmented generation (RAG) combines retrieved requirements with a generative language model in order to support more advanced analysis. Such approaches are commonly used in software and requirements-related settings to improve tasks by using retrieved information as input for generation [2, 19, 21]. In the context of requirement review, this makes it possible not only to retrieve relevant requirements, but also to support their interpretation and explain why they are related.

Importantly, in this thesis, RAG is not used to improve retrieval performance, but to support the review process by helping interpret and analyze the retrieved requirements. In other words, retrieval generates candidates, while RAG supports their interpretation. The goal is to assist engineers in understanding relationships between requirements, rather than to replace the retrieval component.

Unlike approaches based on training or fine-tuning task-specific models, RAG does not require large labeled datasets, which are often unavailable in industrial requirement repositories. Training such models is also computationally expensive and difficult to maintain as requirements evolve over time. In contrast, RAG allows the system to operate directly on existing requirement data through retrieval, making it potentially more flexible and easier to adapt to evolving industrial requirement repositories.

Existing work in requirements engineering shows that semantic retrieval and retrieval-augmented generation can support tasks such as traceability and consistency checking [2, 19, 21]. However, many existing studies often focus primarily on demonstrating retrieval performance, while less attention is given to the reliability and statistical validity of the evaluation process

itself [18, 16].

However, despite the growing interest in retrieval and RAG-based approaches, the way they are evaluated is still often limited or methodologically limited. Research has shown that statistical analysis, evaluation design, and the interpretation of metrics are critical for drawing reliable conclusions in both information retrieval and machine learning [17, 10, 14]. For example, metrics such as precision and recall are often reported as single values, even though they are computed from finite samples and depend on the queries that are selected and the relevance judgments that are made [8, 13]. This means that differences in performance are not always meaningful, especially when uncertainty and variability are not reported. As a result, it is often difficult to determine whether reported improvements reflect genuine retrieval quality or artifacts of the evaluation setup.

To address these challenges, this thesis develops and evaluates a retrieval-based and retrieval-augmented framework for requirement review using a controlled and statistically grounded evaluation methodology.

Research questions This thesis is guided by the following research questions (RQ):

- RQ1: How well does embedding-based semantic retrieval support requirement review tasks in terms of precision, recall, and ranking performance?
- RQ2: How does the use of retrieval-augmented generation (RAG) support the interpretation and analysis of retrieved requirements?
- RQ3: How does the evaluation design, including relevance labeling and statistical analysis, affect the interpretation of retrieval performance?

Together, these research questions aim to evaluate both the effectiveness of retrieval-based methods and the impact of evaluation design on the interpretation of results.

This thesis addresses these limitations by developing a controlled and reproducible evaluation framework for retrieval-based requirement review. A

key contribution is the application of pooled relevance judging within the proposed evaluation framework, which expands the ground truth beyond a single retrieval output and helps reduce evaluation bias. In addition, retrieval metrics such as precision, recall, and ranking performance are treated as statistical estimates, and bootstrap confidence intervals are used to quantify uncertainty. This provides a more reliable basis for interpreting performance differences.

Overall, this work contributes both a practical retrieval and retrieval-augmented pipeline for requirement review and a statistically grounded evaluation methodology that supports more interpretable and reliable evaluation of results.

2 Background

2.1 RAG and requirements engineering

Recent advances in large language models have led to growing interest in their use within requirements engineering [1, 6, 11]. In general, these models are used to support tasks that involve understanding, organizing, and analyzing requirement texts. This is especially relevant in modern engineering projects, where repositories are usually large and difficult to review manually.

One important approach in this area is retrieval-augmented generation (RAG). A RAG system combines two steps. Firstly, a retrieval component is used to identify documents or requirements that might be relevant to a given query. Secondly, the retrieved information is passed to a generative language model, which uses this context to produce a response. This means that the model uses the retrieved information when generating the output, instead of relying only on what it already knows [19, 21].

Why Retrieval-Augmented Generation RAG can provide a practical alternative to fully fine-tuned models for domain-specific tasks [19, 21]. One key advantage is that it does not require additional training. The model uses a pre-trained language model together with a retrieval system, so it can

access relevant information when producing an answer.

This approach has several practical advantages. First, it reduces the need for large labeled datasets, which are often expensive to obtain in industrial settings. Second, it allows the system to remain flexible, since updates to the knowledge base can be incorporated without retraining the model. Third, it can improve transparency, since outputs can be compared with the retrieved requirements used as evidence.

For requirement review tasks, this is particularly important, since correctness and traceability are essential. RAG therefore provides a potentially more flexible and interpretable approach for requirement review compared to fully fine-tuned approaches.

Combining these two steps is particularly useful for requirement review tasks. In many cases, the goal is not only to find related requirements, but also to understand how they are connected and whether they reveal potential issues. For example, an engineer may need to identify inconsistencies between parts of the system. A system that only retrieves can help find candidate requirements, while a RAG-based system can support a more structured interpretation of these results. In this thesis, the role of RAG is specifically to support interpretation and reasoning over retrieved requirements, rather than to improve retrieval performance itself.

Recent studies show that large language models and RAG systems are increasingly explored in requirements engineering. Studies also report applications such as requirement classification, traceability analysis, consistency checking, and review support [6, 11]. Other work has shown that retrieval-augmented language models can be used in industrial settings to support tasks such as generating test scenarios from natural-language requirements [2]. These developments suggest that large language model (LLM)-based systems may help reduce manual effort and make it easier to access relevant information in large requirement collections.

One of the limitations worth mentioning in the literature is that evaluation is often less developed than the system design itself. Many studies present promising tools or proof-of-concept systems, but the evaluation setups are often very specific in the tasks used, and they are based on small

datasets, or they are difficult to compare across studies [6, 21, 18]. As a result, it is often not clear how reliable the reported improvements are, and whether they can be generalized beyond the specific setting in which the system was tested.

These limitations highlight the need for more systematic and statistically grounded evaluation methods. Because of this, we look more carefully at semantic retrieval, evaluation methods, and uncertainty, which are discussed in the next sections. The overall pipeline used in this thesis is illustrated in Figure 1.

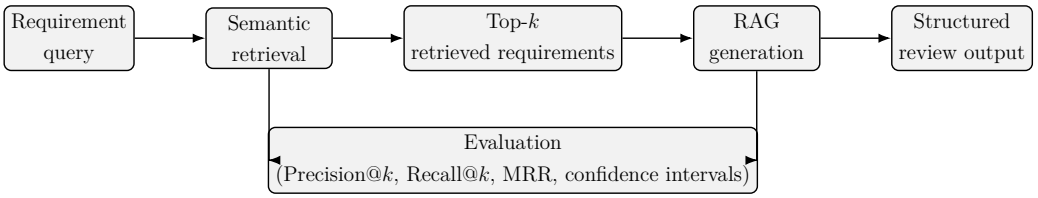


Figure 1: High-level pipeline used in this thesis: semantic retrieval, RAG-based analysis, and statistical evaluation.

2.2 Embeddings and similarity

Many approaches that use retrieval or RAG rely on representing text in a meaningful way. Instead of comparing requirements based on exact descriptions, many methods use embeddings. An embedding is a numerical vector that represents a piece of text. In practice, a description of a requirement is transformed into a point in a high-dimensional space, where similar requirements are expected to be located close to each other [15, 3].

Once requirements are represented as vectors, we can compute the similarity between them mathematically. In this thesis, we use cosine similarity as the main similarity measure. Cosine similarity measures how similar two vectors are based on the angle between them. A higher cosine similarity means that two requirements are more semantically similar in the embedding space.

Several similarity and distance measures can be used to compare embedding vectors. These metrics capture different geometric notions of similarity

in the embedding space and may lead to different retrieval behavior [22]. In this thesis, we focus on cosine similarity, but also consider alternative distance metrics for comparison.

Given two vectors x and y:

- Cosine similarity:
$$\text{cosine}(x, y) = \frac{x \cdot y}{\|x\|\|y\|}$$
- Euclidean distance:
$$d_{\text{euclidean}}(x, y) = \|x - y\|_2$$
- Manhattan distance:
$$d_{\text{manhattan}}(x, y) = \|x - y\|_1$$
- Chebyshev distance:
$$d_{\text{chebyshev}}(x, y) = \max_i |x_i - y_i|$$

Cosine similarity is commonly preferred in embedding-based retrieval, as it measures directional similarity and is less sensitive to vector magnitude [9, 15]. When embeddings are L2-normalized, cosine similarity and Euclidean distance become closely related, which explains their similar behavior in practice.

For this type of task, sentence-level embedding models are commonly used. They are designed to represent entire sentences or requirement descriptions as single vectors. This makes them suitable for semantic retrieval, since the goal is to find requirements that are similar in meaning to a given query requirement.

It is important to note that the quality of the retrieval results depends on both the embedding model and the similarity measure. This is because the embedding model determines how the text is represented, while the similarity measure determines how these representations are compared. Different

choices can lead to different retrieval results, which makes both components important when designing and evaluating a retrieval system.

2.3 Evaluation in IR and RAG

Evaluation has long been an important topic in information retrieval (IR), where the goal is to measure how well a system retrieves relevant documents for a given query [10, 17]. In classical IR, this is usually done using metrics such as precision and recall [10, 13]. Precision measures how many of the retrieved items are relevant, while recall measures how many of the relevant items are successfully retrieved. Additional metrics such as Mean Reciprocal Rank (MRR) can be used in ranked retrieval settings in order to capture how early a relevant item appears in the ranked list [10]. In this thesis, we focus on a ranked retrieval setting, so we use metrics such as Precision@k, Recall@k, and MRR to evaluate retrieval performance.

Most of these evaluation methods rely on a binary definition of relevance [10, 13]. This means each item is labeled as either relevant or not. This simplifies the evaluation, but it also introduces some limitations. One of them is that relevance is not always clear, especially in requirement engineering tasks, where relationships between requirements can be partial, indirect, or context-dependent. This is also the approach used in this thesis, where relevance is defined in a binary way for evaluation purposes.

Additionally, evaluation becomes more challenging with the introduction of semantic search and embedding-based retrieval. Semantic retrieval focuses on meaning rather than exact keyword overlap [7, 12]. This makes relevance harder to define in a consistent way because two requirements may be related in different ways depending on the task. For example, a requirement may be useful for detecting inconsistencies but not for traceability analysis. More generally, evaluating semantic search is difficult because results can depend a lot on the domain, the dataset used, and the fact that different studies are not always easy to compare.

In retrieval-augmented generation (RAG) systems, evaluation becomes even more complex. This is because a RAG system consists of two parts:

retrieval and generation, so the overall system performance depends on not only how well relevant documents are retrieved but also on how well the language model uses this information to generate useful outputs [21, 18]. As a result, evaluation is often split into retrieval quality and generation quality, which are not always easy to combine [21, 16].

Another limitation that can be found in the literature is the lack of unified evaluation standards. Different studies use different datasets, tasks, and metrics, which makes it difficult to compare results across papers [21, 18]. In many cases, evaluation setups are specific to a particular application or dataset, and it is not always clear whether the reported results could be generalized to other settings [6].

Overall, these challenges show that evaluation of retrieval and RAG systems is not straightforward. Differences in datasets, relevance definitions, and evaluation protocols can make it difficult to draw consistent conclusions across studies [17, 7, 21]. For this reason, evaluation should be designed and reported more carefully and systematically. This thesis therefore focuses on evaluation from a statistical perspective, which is discussed in the next section.

2.4 Statistical evaluation and uncertainty

Evaluation metrics such as precision, recall, and MRR are often reported as fixed values when retrieval systems are assessed. However, in practice these metrics are computed from a limited set of queries, so the reported values should be interpreted as estimates of system performance rather than exact fixed values [8, 14].

Since performance metrics are usually calculated over a finite number of queries, the reported values depend on the specific queries that are selected and also on the associated relevance labels. In requirement engineering tasks, this effect may be even stronger, since relevance is often defined manually and may depend on how we interpreted the results. For this reason, evaluation outcomes are subject to variability [8, 4].

This variability means that differences in metric values between systems

do not necessarily reflect true performance differences. Instead, they may arise because of the particular query set or the labeling process. It is therefore important to quantify the uncertainty associated with the reported metrics.

One common way to address this is to construct confidence intervals, which provide a range of plausible values for a performance metric and make it easier to assess how stable the results are. For instance, wide intervals suggest greater variability across queries, while narrower intervals may indicate more consistent observed behavior [20, 5].

Statistical comparison methods can also be used to assess whether observed differences may reflect variability in the evaluation setup. Instead of relying only on small differences between point estimates, statistical methods make it possible to assess whether observed differences might be meaningful or occurred by chance [4, 14].

Overall, we can conclude that evaluation should not rely only on single metric values. Performance should also be considered together with its associated uncertainty. This is important in requirement engineering settings, since datasets are often limited in size and heterogeneous.

For this reason, this thesis treats evaluation as a statistical estimation problem. Metrics are treated as estimates of performance, and uncertainty-aware methods are used to make the results more reliable and easier to interpret.

3 Methodology

3.1 Dataset

We conduct the evaluation on synthetic requirement datasets that are designed to resemble realistic industrial requirement repositories. The use of synthetic data is motivated by the lack of publicly available datasets that capture the structure, heterogeneity, and scale of industrial systems such as those managed in tools like Polarion. Since real-world requirement repositories are often incomplete, or difficult to access due to confidentiality constraints, two synthetic datasets were generated. This allows us to construct

a controlled and reproducible evaluation setting while preserving selected structural properties of real systems.

Although the dataset is synthetically generated, it is constructed using a controlled, LLM-assisted workflow to support structural realism. The generation process enforces predefined constraints such as hierarchical relationships (system, subsystem, component), multiple interacting products, and diverse requirement types. The generated requirements are then manually reviewed and cleaned to ensure consistency and coherence. The goal is not to exactly replicate real industrial data, but to approximate its structure closely enough to enable a meaningful and controlled evaluation of retrieval and RAG-based methods in this setting. As a result, the dataset should be seen as structurally realistic but not fully representative of real-world industrial repositories.

Dataset versions We use two dataset versions in this thesis.

- **Dataset v1** is the initial dataset used for controlled retrieval evaluation. It contains a moderate number of requirements (approximately 130) and is used for developing the evaluation protocol, defining queries, and performing the first experiments.
- **Dataset v2** extends v1 with additional requirements in order to increase dataset size (approximately 340 requirements), structural diversity, and subsystem coverage. It is used to analyze how the pipeline behaves on a larger requirement collection, final system evaluation, and demo scenarios.

Both datasets follow the same schema and generation principles, which makes sure that we have consistency across the experiments.

Requirement structure Each requirement in the dataset consists of:

- A unique identifier (ID)
- A title and a natural-language description
- A requirement type (e.g., system, software, interface, safety)

- A parent identifier encoding hierarchical relationships
- Product and subsystem annotations
- Additional metadata such as priority and verification method

Only the textual description is used as input to the embedding model for retrieval. The remaining fields are used for interpretation, relevance definition, and qualitative analysis.

The dataset follows a three-level hierarchy:

$$\text{System} \rightarrow \text{Subsystem} \rightarrow \text{Component}$$

This structure shows how requirements are typically organized in industrial systems and helps model relationships between requirements such as traceability and cross-level dependencies.

Products The dataset includes requirements from multiple interacting products:

- Protection Relay
- Intelligent Sensor
- Grid Automation Controller
- Protection Gateway
- Monitoring Software (human-machine interface, HMI)

These products represent different layers of a power system architecture, from field devices to supervisory systems, and allow evaluation on cross-product retrieval behavior.

Requirement types To reflect engineering diversity, the dataset includes multiple requirement types:

- Software requirements
- Hardware requirements
- Firmware requirements
- Interface requirements
- Safety requirements
- System-level requirements

The use of a variety of requirement types makes the retrieval task more diverse and closer to realistic requirement review scenarios.

Reproducibility During evaluation, the dataset is kept fixed. No requirements are added, removed, or modified after the evaluation setup is defined. This ensures that all experiments are reproducible and that the differences found in performance can be attributed to changes in the retrieval or RAG configuration rather than to changes in the underlying data.

Importantly, retrieval is based only on the requirement text, not on metadata fields. This means that the system must identify similarity from the requirement descriptions rather than from simple metadata matches.

3.2 Retrieval methodology

The retrieval system used in this thesis is based on semantic similarity between the description of the requirements [9]. Using the description of each requirement, we transform it into a dense vector embedding using a pretrained embedding model [15]. Specifically, the descriptions are encoded using the Azure OpenAI embedding model `text-embedding-3-small` to generate these representations.

In addition to the baseline embedding model `text-embedding-3-small`, we also consider a larger embedding model `text-embedding-3-large` for comparison.

The two models differ in representation capacity and embedding dimensionality. According to the model documentation, text-embedding-3-small produces 1536-dimensional embeddings, while text-embedding-3-large produces 3072-dimensional embeddings.

This comparison allows us to examine whether increased embedding capacity leads to improved retrieval performance in requirement-based semantic search. However, since retrieval performance also depends on the structure of the dataset and the similarity measure, larger models do not necessarily guarantee better results.

In the final pipeline, the embeddings are precomputed once and stored as matrix files, and are loaded during retrieval to avoid recomputation. As a result, the retrieval process is faster, computationally consistent, and easier to reproduce.

After we load the embeddings, we perform the retrieval using cosine similarity [9, 15]. For a query requirement, we compare its embedding with all other embeddings in the dataset. Since the vectors are L2-normalized, cosine similarity is equivalent to the dot product, which simplifies computation and ensures consistent similarity scores across requirements. In general, a higher similarity score means that the query and the compared requirement are considered to be more semantically related.

To better understand the structure of the embedding space, we visualize the requirements using a three-dimensional PCA projection, as shown in Figure 2.

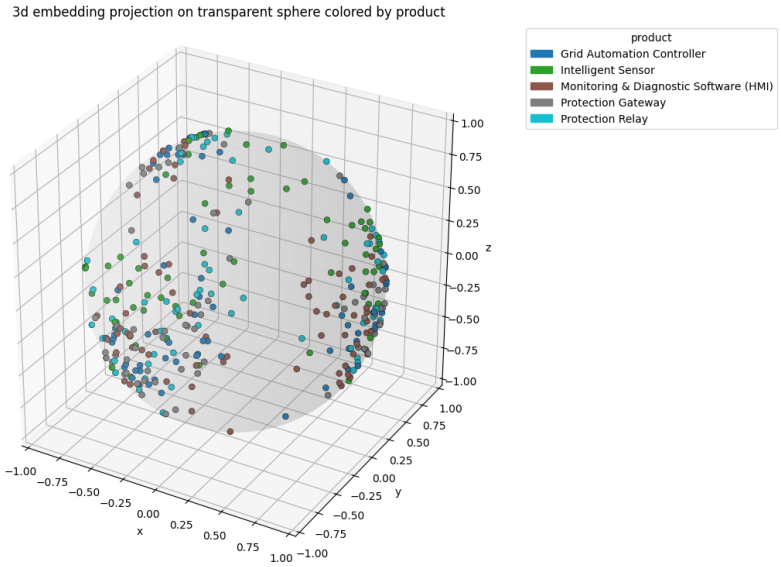


Figure 2: 3D PCA projection of requirement embeddings for dataset v2. Each point represents a requirement, colored by product category. The projection illustrates the structure of the embedding space and the overlap between different product groups. The first three principal components explain approximately 18.9% of the total variance. The principal components correspond to directions in the embedding space that capture the largest variation in the data. Since only a portion of the total variance is explained, the visualization provides an approximate view of the structure.

When we are done with computing the similarity scores, the requirements are ranked in descending order. In order to avoid retrieving the query requirement itself as the nearest neighbor, we exclude it from the ranking. At the end of this step, the system returns the top-k ranked requirements as the candidates of the query.

Throughout the thesis, we keep the retrieval depth fixed at $k = 10$, so the system returns the ten most similar requirements for each query. The reason for fixing k is to ensure consistency across all experiments and allow comparison between results. It also reflects a realistic review setting, where an engineer would usually check only a small number of top-ranked candidates.

The retrieved results are then stored in structured JSON (JavaScript Object Notation) files so that we can reuse the pipeline without rerunning the full retrieval step each time. This frozen retrieval setup is the basis for

both the retrieval-only baseline and the retrieval-augmented generation setup used later in this thesis.

3.3 Evaluation Protocol

In this thesis, we evaluate the retrieval system using a controlled query-based information retrieval setup [10, 17]. The goal is to evaluate how well the system retrieves requirements that are relevant for different review tasks, such as detecting duplicates, identifying possible conflicts, and finding related requirements that should be analyzed together.

3.3.1 Query setup

Before starting the evaluation, we manually define a fixed set of 20 evaluation queries. Each query corresponds to a selected requirement from dataset v1 and represents a plausible requirement review scenario. The queries that we chose are designed to cover different types of review situations, including:

- detection of similar or duplicate requirements
- identification of potential conflicts or inconsistencies
- discovery of missing constraints
- interface and integration analysis
- logging and fault-related behavior

The limited number of queries is considered when interpreting the statistical results.

For each query, the system retrieves the top-k most similar requirements using the fixed retrieval configuration with $k = 10$. The ranked outputs are stored and reused during evaluation.

3.3.2 Relevance definition

We define relevance using a binary semantic relevance model, which is common in information retrieval evaluation [10, 13]. For each query q, each retrieved requirement is manually labeled as either relevant or non-relevant. The set of requirements labeled as relevant forms the evaluation ground truth for that query. Since the labels are manually assigned by one annotator, the relevance judgments should be interpreted as task-specific evaluation labels rather than objective ground truth.

In general, a requirement is considered relevant if it supports the intended review objective of the query.

A requirement may be labeled as relevant to a query if it:

- expresses similar functional behavior
- refers to related subsystem or system-level logic
- represents a meaningful engineering dependency
- supports traceability or cross-product relationships

The relevance labels are stored in structured JSON files and reused across experiments.

It is important to note that we manually define the relevance labels because there is no existing ground truth for requirement similarity or consistency in this dataset. As a result, it is necessary to determine which requirements are relevant for each query. This approach is common in information retrieval evaluation, where relevance is often defined through human assessment [10, 17].

3.3.3 Metrics

We measure retrieval performance using Precision@k, Recall@k, and Mean Reciprocal Rank (MRR@k), which are standard metrics in retrieval evaluation [10, 8].

Using $R_k(q)$ for the top-k retrieved requirements and $G(q)$ for the set of relevant requirements for query q (the ground truth), Precision@k and Recall@k are defined as:

$$\text{Precision@}k(q) = \frac{|R_k(q) \cap G(q)|}{k}, \qquad \text{Recall@}k(q) = \frac{|R_k(q) \cap G(q)|}{|G(q)|}.$$

All evaluation queries are constructed with at least one relevant requirement, so Recall@k is well-defined.

Let rank_q be the rank of the first relevant requirement retrieved for query q. Then

$$\text{MRR@}k(q) = \begin{cases} \frac{1}{\text{rank}_q}, & \text{if at least one relevant requirement appears in the top-}k\text{ results} \\ 0, & \text{otherwise.} \end{cases}$$

All main metrics are first computed separately for each query. We then report the average across all queries. This means that the results describe average query performance, rather than one combined count over all retrieved requirements.

In addition to the standard metrics, we also compute Adjusted Precision@k as a secondary interpretation measure. This is motivated by the fact that many queries contain only a small number of relevant requirements. In these cases, raw Precision@10 can appear low even when the system retrieves most or all relevant items.

As a result, using adjusted precision helps by normalizing the number of retrieved relevant items by the maximum number of relevant items that could be retrieved at depth k. It is defined as:

$$\text{Adjusted Precision@}k(q) = \frac{|R_k(q) \cap G(q)|}{\min(k, |G(q)|)}$$

Adjusted Precision@10 takes values between 0 and 1 and reflects how close the system is to the best possible performance for that query at the given retrieval depth. Overall, we use this metric not as a replacement for the

standard metrics, but as an additional measure that helps interpret precision more fairly when relevance sets are small.

3.3.4 Candidate set size justification (k=20)

In order to construct a pooled relevance set that captures a sufficiently broad candidate set for manual relevance judging of relevant requirements, it is necessary to choose an appropriate candidate pool size.

In this thesis, we use $k = 20$ as the candidate set size for pooled relevance judging. This choice balances coverage of potentially relevant candidates with manageable manual labeling effort. A larger value of k may increase the number of retrieved relevant requirements, but also introduces additional non-relevant candidates, making manual labeling more time-consuming.

The choice of $k = 20$ is further supported by empirical analysis of retrieval coverage across different depths, as shown in Section 4.

3.3.5 Pooled relevance judging

In the initial evaluation setup, relevance labels were defined only within the top-10 retrieved results for each query. This introduces bias, since the evaluation would depend on the output of the retrieval system itself.

To address this issue, we apply pooled relevance judging, a common approach in information retrieval evaluation [10, 17]. Specifically, for each query, a candidate pool is constructed by combining:

- the top-20 retrieved requirements from the baseline system
- requirement identifiers referenced in the RAG outputs

We also removed any duplicate identifiers, and we manually labeled all candidates again using the same binary relevance criteria as before.

Finally, the pooled relevance labels are stored and used to recompute the evaluation metrics. It is important to note that the requirement identifiers extracted from the RAG outputs are only used to expand the candidate pool, and are not automatically considered relevant. All candidates are manually

labeled using the same binary relevance criteria. Duplicate identifiers are removed before labeling. In this thesis, the labeling is done by one person. This makes the process simpler, but it also means that the labels may be subjective. Therefore, no inter-annotator agreement is reported, which is a limitation of this work.

3.3.6 Statistical evaluation, uncertainty and baseline–pooled comparison

The evaluation metrics that we report are treated as statistical estimates rather than exact values. Since the evaluation is based on a finite set of $N = 20$ queries, the metrics may vary depending on the sampled queries [8, 4].

In order to quantify the uncertainty, we use a non-parametric bootstrap procedure. We resample the evaluation queries with replacement, and then recompute the mean metric for each resample. This process is repeated with $B = 5000$ bootstrap iterations. We choose $B = 5000$ to obtain stable bootstrap estimates while keeping the computation practical. A smaller number of resamples could make the confidence intervals less stable due to randomness in the resampling process, while a much larger number would mainly increase computation time without significantly changing the results.

In simple terms, the bootstrap estimates how much the results may vary by repeatedly resampling the query set. This helps avoid over-interpreting small observed differences.

The confidence intervals are obtained from the empirical bootstrap distribution, which provides a range of plausible values for each metric [20, 5]. The bootstrap intervals reflect variability across the sampled queries, but they do not capture all sources of uncertainty, such as subjective relevance labels or the use of synthetic data.

In this thesis, we compare two evaluation settings. The first uses the original relevance labels that were defined from the baseline retrieval outputs, while the second uses the pooled relevance labels obtained after we expanded the candidate set. In both settings, the retrieval outputs are kept identical,

which allows us to isolate the effect of the ground-truth construction.

We also apply the same bootstrap-based comparison framework to the Adjusted Precision@10 in order to analyze how the change in relevance labeling affects how this metric is interpreted.

Dataset-specific evaluation setup It is important to clarify that the baseline and pooled evaluation comparison is conducted only on dataset v1. This dataset is used to develop and analyze the evaluation protocol in a controlled setting.

For dataset v2, the evaluation is performed exclusively using the pooled relevance protocol. This is because v2 is used for extended evaluation on a larger dataset, where a broader and less system-dependent evaluation ground truth is required.

As a result, all comparisons between baseline and pooled relevance labels are restricted to dataset v1, while dataset v2 reflects the final evaluation setup based on pooled relevance judging.

3.3.7 Reproducibility artifacts

To support reproducibility, all key components of the evaluation pipeline are stored and organized in a structured format. This includes:

- the synthetic datasets (v1 and v2)
- retrieval outputs for all queries
- manually defined relevance labels
- pooled candidate sets
- evaluation scripts and notebooks
- generated figures and result tables

These artifacts make it possible to reproduce the evaluation results without rerunning the full pipeline, and support transparency in how the results are obtained.

3.4 System setup: Baseline vs RAG

This section describes the two final system configurations used in the evaluation.

In this thesis, we compare two system configurations in order to study how adding a generation component changes the interpretation layer built on top of retrieval.

Baseline_v1 The first configuration, referred to as **Baseline_v1**, consists only of the retrieval component. For a given query requirement, the system retrieves the top-k most similar requirements using cosine similarity in the embedding space.

In this step, we do not use any generation algorithm. The output is therefore a ranked list of requirement identifiers, which is then used directly in the retrieval evaluation.

RAG_v1 The second configuration, referred to as **RAG_v1**, extends the baseline by adding a retrieval-augmented generation component. We use the same retrieval setup to obtain the top-k candidate requirements. These retrieved requirements are then provided as context to a large language model. In this thesis, the purpose of the RAG component is not to improve retrieval metrics directly, but to support interpretation of the retrieved requirements.

The overall pipeline is illustrated in Figure 3.

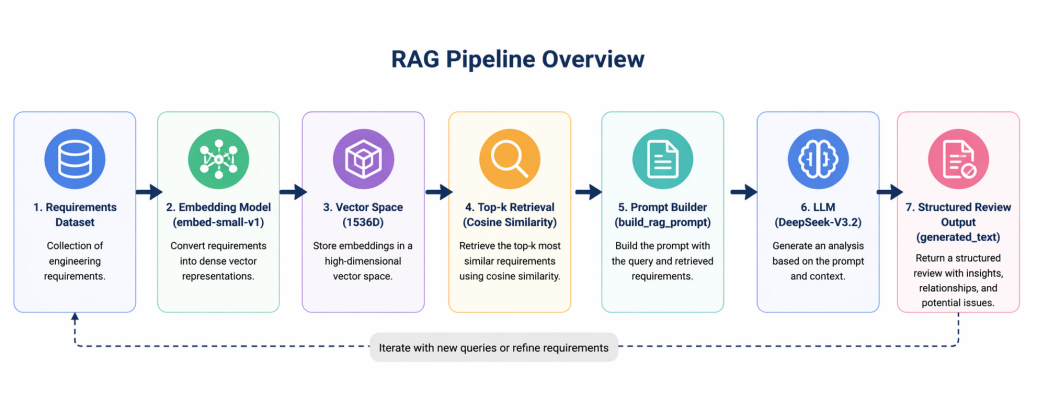


Figure 3: Overview of the RAG_v1 pipeline. Requirements are embedded into a vector space, retrieved using cosine similarity, and passed to a language model to generate structured requirement reviews.

To guide the generation process, we use a fixed prompt template (see Appendix A). The prompt is designed to support requirement review tasks such as identifying inconsistencies, missing constraints, and relationships between requirements.

Generation is performed with temperature set to 0.0, which helps reduce output variability across runs. The generated outputs follow a structured format, which makes them easier to interpret and analyze.

Frozen configuration We keep fixed the setup across both configurations in order to ensure a fair comparison between the baseline and the RAG-based system. In particular, the following components remain unchanged:

- the dataset
- the predefined evaluation queries
- the retrieval depth ($k = 10$)
- the embedding model and similarity measure
- the prompt template and its version
- the evaluation protocol

The fixed setup makes it easier to understand where any differences come from. Since everything else stays the same, this makes it easier to interpret differences as being related to the added generation component rather than changes in the retrieval setup. It also helps keep the evaluation setup consistent across configurations.

Role of dataset v2 In addition to the controlled experiments conducted on dataset v1, dataset v2 is used for extended evaluation and scalability analysis. In this setting, the system is evaluated using the pooled relevance protocol, and additional experiments are conducted, including embedding model comparison and evaluation of the RAG-based system.

This allows us to assess how the system behaves under a larger and more diverse dataset while maintaining consistency with the evaluation framework defined for dataset v1.

Free-text query support In addition to the predefined evaluation queries, we added free-text queries to the system for demonstration purposes. In this case, the same retrieval and generation pipeline is used, but the input is no longer restricted to requirements from the dataset.

Instead, a user can provide an arbitrary natural-language query. Apart from this difference in input type, the underlying system remains unchanged. This makes the demo setting technically consistent with the evaluated pipeline, although the free-text queries are not part of the quantitative evaluation.

3.5 Example of pipeline output

This example illustrates how the system behaves in a realistic usage scenario. The query is not part of the evaluation set. The goal is to show the full pipeline, from user input to retrieval and structured output.

The example below includes the query, the retrieved requirements, and the generated requirement review. For readability, only the first five retrieved requirements are shown, although the generation step uses the full retrieved context.

The generated review shown below is lightly formatted for readability, while preserving the structure and content of the original system output.

User Query

What requirements are related to time synchronization?

Top-5 retrieved requirements

ID	Description
PR-HW-005	The hardware shall support high-precision time synchronization.
PR-SW-004	The subsystem shall manage time synchronization with external sources.
PR-SW-008	The module shall synchronize time using external protocols.
PR-SW-017	The module shall detect time synchronization deviations.
PR-IF-003	The interface shall support Network Time Protocol (NTP) and Precision Time Protocol (PTP) synchronization.

Generated requirement review

Related requirements

- PR-HW-005: Hardware support for precise synchronization
- PR-SW-004: Synchronization management logic
- PR-SW-008: External protocol synchronization
- PR-SW-017: Drift detection
- PR-IF-003: Interface-level synchronization support

Potential conflicts

- GA-HW-005, PG-HW-005: inconsistent precision definitions without explicit tolerance

Missing constraints

- No explicit synchronization accuracy thresholds
- No explicit synchronization frequency requirements

Figure 4: Example of the end-to-end RAG pipeline using a free-text query. The system retrieves semantically related requirements and generates a structured requirement review, including relationships, potential conflicts, and missing constraints.

4 Results

4.1 Baseline retrieval results (Dataset v1)

In this section, we report the retrieval performance of the baseline system on dataset v1 using the evaluation protocol described in Section 3 [10, 17]. The results are computed over $N = 20$ evaluation queries with a fixed retrieval depth of $k = 10$.

The baseline achieves the following mean performance:

- Precision@10 = 0.235
- Recall@10 = 0.925
- MRR@10 = 0.679

In addition, uncertainty is estimated using bootstrap resampling ($B = 5000$), and the corresponding 95% confidence intervals are reported [20, 5]:

- Precision@10: [0.190, 0.270]
- Recall@10: [0.800, 1.000]
- MRR@10: [0.513, 0.838]

The results show a clear difference between precision and recall. The high recall indicates that the system is generally able to retrieve most of the relevant requirements within the top-10 results. In contrast, the relatively low Precision@10 suggests that many non-relevant requirements are also included in the retrieved set.

Since many queries contain only a small number of relevant requirements, the raw Precision@10 values can appear low even when most relevant items are retrieved. To support interpretation, Adjusted Precision@10 is also computed, which normalizes the number of retrieved relevant requirements by the maximum achievable number of relevant requirements at depth k. This metric is analyzed further in the adaptive-k analysis below.

In embedding-based semantic retrieval, this behavior is expected, since similarity captures broad semantic relations rather than strict relevance. As discussed in the literature, precision and recall should be interpreted as statistical estimates that depend on the evaluation setup and query sample [8, 9, 7].

The MRR@10 result further shows that even though the top-10 set contains some noise, relevant candidates are typically found among the top-ranked results.

4.2 Per-query variation in the baseline v1 evaluation

In addition to the aggregated baseline results for dataset v1, we analyze how retrieval performance varies across individual queries in the original baseline evaluation at $k = 10$. This provides a more detailed view of how the system behaves across different queries.

Figure 5 shows the Precision@10 values per query. The results vary across queries, with some achieving higher precision than others.

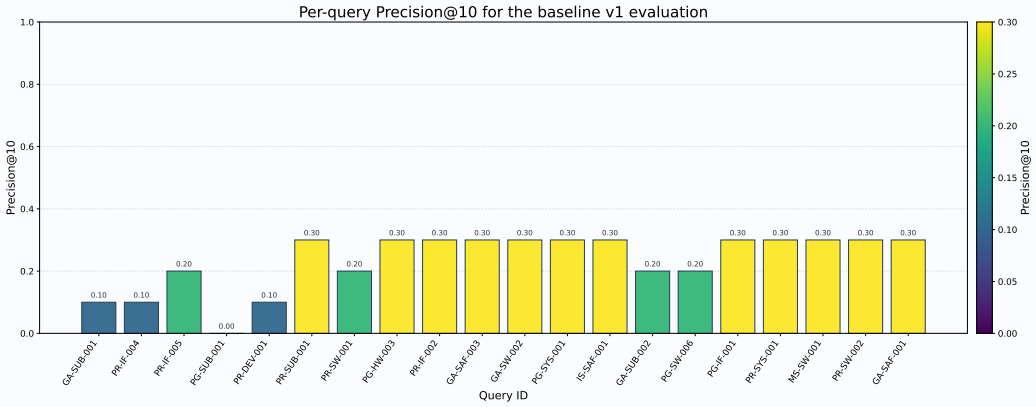


Figure 5: Per-query Precision@10 for the baseline v1 evaluation.

Figure 6 shows the Recall@10 values per query. In contrast to precision, recall remains high for most queries, although a small number of queries show lower recall, including one clear failure case, where no relevant requirements are retrieved.

Standard deviation measures how much the metric values vary across queries. The standard deviation is 0.093 for Precision@10 and 0.245 for

Recall@10, which indicates that recall varies more across queries, mainly because of a small number of difficult cases.

Finally, query PG-SUB-001 represents a clear failure case, where both precision and recall are zero. This means that the retrieved set contains mostly non-relevant requirements and misses relevant ones entirely.

Overall, the results confirm that recall is consistently high, while precision varies across queries. This supports the interpretation that the reported mean values should be seen as aggregated estimates rather than uniform performance across all queries [8, 14].

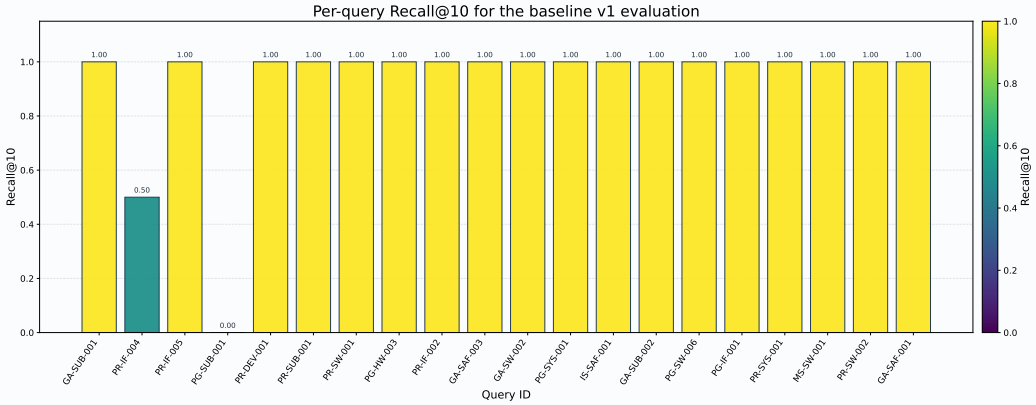


Figure 6: Per-query Recall@10 for the baseline v1 evaluation.

4.3 Pooled evaluation (v1 vs v2)

We use the pooled evaluation as the main evaluation protocol in this thesis. As described in Section 3, pooled relevance judging gives a broader evaluation ground truth than the original baseline-only labeling setup, since relevance is defined after combining candidates from retrieval and RAG outputs [10, 17].

For dataset v1, the pooled evaluation gives the following mean results:

- Precision@10 = 0.435
- Recall@10 = 0.794
- MRR@10 = 0.779

For dataset v2, the pooled evaluation gives the following mean results:

- Precision@10 = 0.440
- Recall@10 = 0.737
- MRR@10 = 0.802

The pooled results show that precision remains almost unchanged between the two datasets. In contrast, recall is slightly lower for dataset v2, while MRR is slightly higher. This means that under the pooled protocol, the system keeps a similar level of precision, retrieves a somewhat smaller share of all relevant requirements in v2, but tends to place the first relevant requirements slightly earlier in the ranked list.

To quantify how much the evaluation protocol changes the results, we also compare the original baseline labels with the pooled labels for dataset v1 while keeping the retrieval outputs fixed. For Precision@10, the mean difference between the pooled and baseline evaluation is 0.20, with a 95% confidence interval of $[0.15, 0.25]$. Since this bootstrap interval does not include zero, it suggests a consistent positive difference across the evaluated queries.

For Adjusted Precision@10, the mean difference between the pooled and baseline evaluation is -0.13, with a 95% confidence interval of $[-0.23, -0.03]$. This indicates that the pooled protocol changes the interpretation of performance, since the same retrieval outputs are evaluated against a broader ground truth.

4.4 Confidence intervals (v1 vs v2)

To quantify the uncertainty of the results reported, we estimated 95% bootstrap confidence intervals using $B = 5000$ resamples over the query set, as described in Section 3 [20, 5].

For dataset v1, the pooled confidence intervals are:

- Precision@10: [0.370, 0.500]

- Recall@10: [0.704, 0.877]
- MRR@10: [0.642, 0.908]

For dataset v2, the pooled confidence intervals are:

- Precision@10: [0.385, 0.495]
- Recall@10: [0.672, 0.807]
- MRR@10: [0.655, 0.929]

Overall, the intervals suggest relatively stable results across the evaluated queries, although some uncertainty and variability still remain, especially for recall and MRR.

4.5 Retrieval behavior results

To further analyze retrieval behavior beyond fixed $k = 10$ evaluation, we examine how performance changes with varying retrieval depth.

4.5.1 Adaptive-k

To better understand how retrieval depth affects ranking quality, we analyzed retrieval performance for different values of k.

Dataset v1: Precision@k and normalized precision For dataset v1, we computed mean Precision@k across different retrieval depths:

- Precision@1 = 0.650
- Precision@3 = 0.567
- Precision@5 = 0.570
- Precision@10 = 0.435
- Precision@20 = 0.275

As expected, Precision@k generally decreases as k increases. This reflects a typical ranked retrieval behavior, where the highest-ranked results are the most relevant, while lower-ranked results introduce additional non-relevant items [10, 13].

To further examine how ranking quality changes with depth, we define normalized precision as the ratio between Precision@k and Precision@1 for each query:

$$\text{Normalized Precision@}k(q) = \frac{\text{Precision@}k(q)}{\text{Precision@1}(q)}.$$

This measure captures how quickly precision decreases relative to the top-ranked result.

For queries where Precision@1 = 0, normalized precision is defined as zero for all k, which reflects complete retrieval failure. As a result, the mean normalized precision at $k = 1$ reflects the proportion of queries for which the top-ranked retrieved requirement is relevant.

The corresponding mean normalized precision values are:

- Normalized Precision@1 = 0.650
- Normalized Precision@3 = 0.467
- Normalized Precision@5 = 0.430
- Normalized Precision@10 = 0.310
- Normalized Precision@20 = 0.178

The normalized precision values show a consistent decrease as k increases, from 0.650 at $k = 1$ to 0.178 at $k = 20$. Normalized precision highlights that this decline is steep relative to the top-ranked result, indicating that most useful information is concentrated in the highest-ranked items.

This behavior is illustrated in Figure 7, where both Precision@k and normalized Precision@k are shown for comparison.

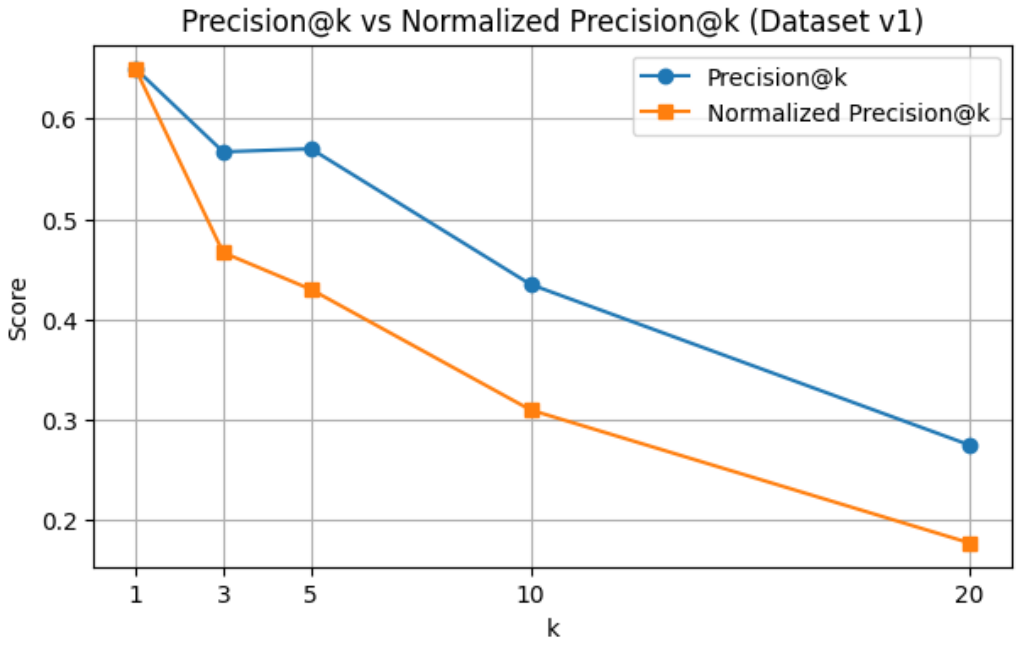


Figure 7: Comparison between Precision@k and normalized Precision@k for dataset v1. While Precision@k decreases with increasing k, normalized precision highlights how quickly ranking quality degrades relative to the top-ranked result.

Although this measure is useful for analyzing ranking behavior, it is not stable across all queries and does not provide a complete picture of retrieval performance. For this reason, it is used here primarily as a diagnostic tool.

Adjusted Precision@k Adjusted precision is computed using the pooled relevance labels, so the corresponding Precision@k values differ slightly from the original adaptive-k precision values reported above.

In addition to normalized precision, we also compute Adjusted Precision@k, which measures how close the system is to retrieving all relevant requirements that we set as ground truth within the top-k results. Adjusted precision is defined as the number of retrieved relevant requirements divided by the maximum possible number of relevant hits within the top-k, i.e., $\min(k, |G(q)|)$.

The corresponding mean adjusted precision values are:

- Adjusted Precision@1 = 0.650

- Adjusted Precision@3 = 0.567
- Adjusted Precision@5 = 0.600
- Adjusted Precision@10 = 0.794
- Adjusted Precision@20 = 0.794

Unlike normalized precision, adjusted precision increases with k and stabilizes around $k = 10$. This indicates that most relevant requirements are retrieved within the top-ranked results, and that increasing the retrieval depth beyond this point mainly introduces non-relevant items rather than additional relevant ones.

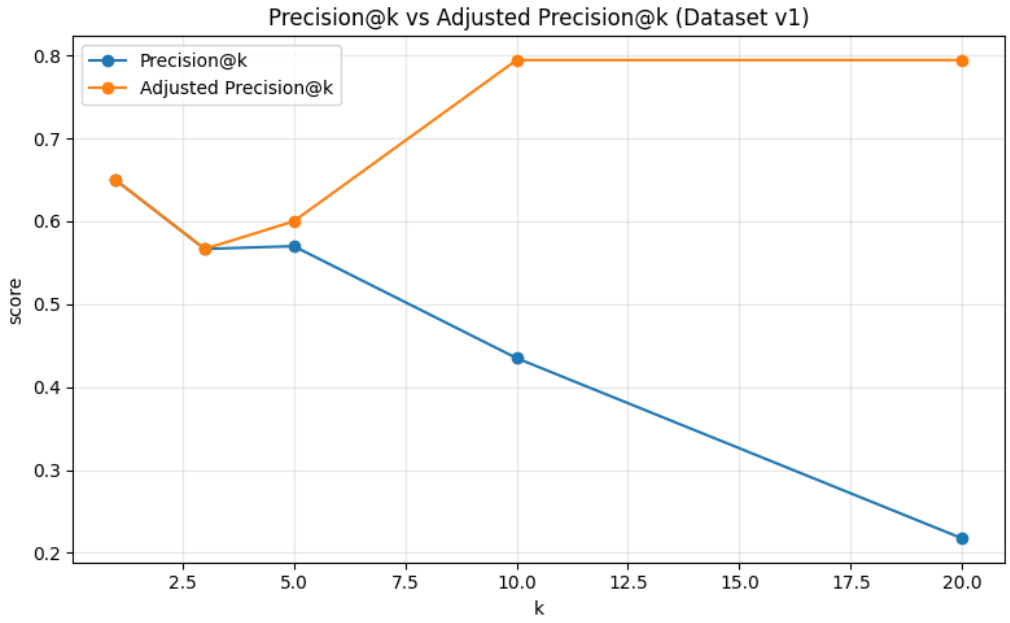


Figure 8: Comparison between Precision@k and Adjusted Precision@k for dataset v1. While Precision@k decreases with increasing k, adjusted precision increases and stabilizes, indicating that most relevant requirements are retrieved within the top-ranked results.

Dataset v2: Precision@k and adjusted precision For dataset v2, we compute both Precision@k and Adjusted Precision@k, where adjusted precision is defined as the number of retrieved relevant requirements divided

by the maximum possible number of relevant hits within the top-k, i.e., $\min(k, |G(q)|)$.

The results are summarized below:

- $k = 1$: Precision = 0.700, Adjusted = 0.700
- $k = 3$: Precision = 0.600, Adjusted = 0.600
- $k = 5$: Precision = 0.520, Adjusted = 0.530
- $k = 10$: Precision = 0.440, Adjusted = 0.737
- $k = 20$: Precision = 0.305, Adjusted = 1.000

We can observe the same Precision@k pattern as in dataset v1.

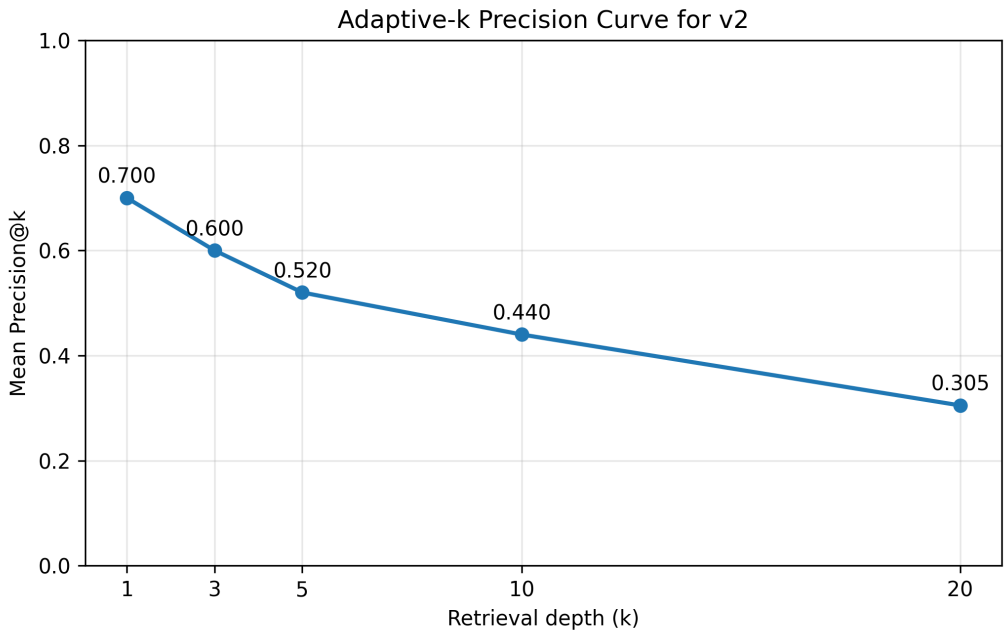


Figure 9: Adaptive-k Precision@k curve for dataset v2 under pooled evaluation.

Normalized precision was also computed for dataset v2 and shows the same decreasing trend as in dataset v1. For clarity, the discussion here

focuses on adjusted precision, which provides a more direct interpretation of retrieval coverage under the pooled evaluation protocol.

Precision@k decreases as k increases, which shows that less relevant items are introduced at deeper ranks. However, Adjusted Precision@k shows a different trend.

Figure 10 illustrates this difference between raw and adjusted precision.

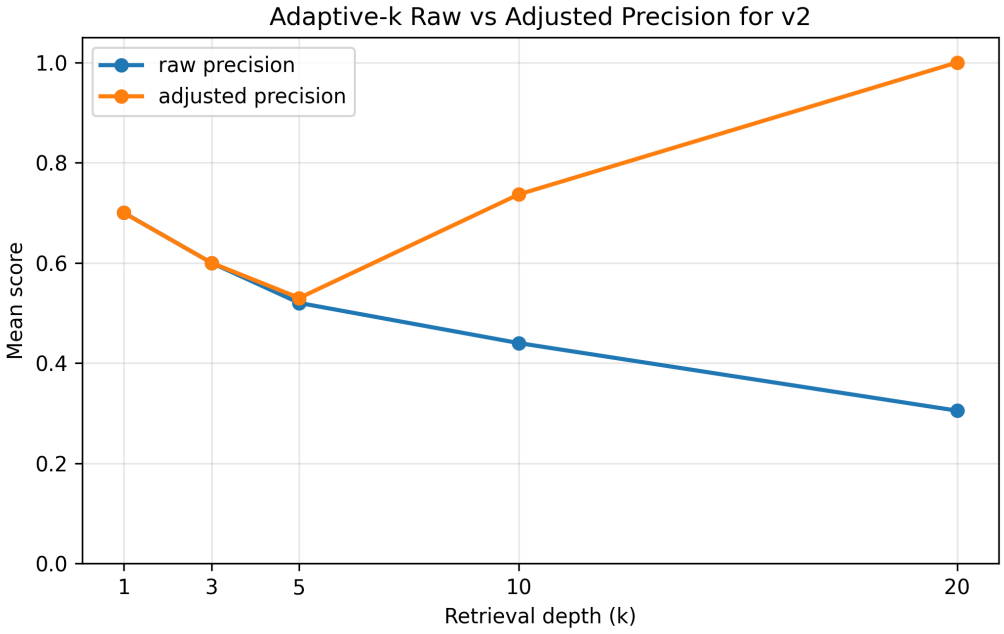


Figure 10: Comparison of Precision@k and Adjusted Precision@k for dataset v2, illustrating the trade-off between ranking quality and retrieval coverage.

It increases with k and reaches 1.0 at $k = 20$, which indicates that the system retrieves all judged relevant requirements within the top-20 results under the pooled relevance set.

This is particularly important in this setting, where each query has only a small number of relevant requirements and raw precision is therefore capped at relatively low values.

Interpretation The comparison between normalized precision and adjusted precision shows that the two measures support different interpretations. Nor-

malized precision is useful for analyzing ranking behavior, while adjusted precision provides a more interpretable estimate of retrieval coverage, especially when the ground truth sets are small.

4.5.2 Pooling depth analysis

In order to justify the choice of candidate pool size used in the pooled relevance judging, we analyze how retrieval depth affects the inclusion of additional relevant requirements.

Specifically, we compare retrieval results at depths $k = 20$, $k = 30$, and $k = 40$ for each query in dataset v1. The goal is to examine whether increasing the pooling depth beyond $k = 20$ provides a substantial number of additional relevant requirements. For each query, we identify the additional candidates that appear beyond the top-20 results (i.e., ranks 21-30 and 21-40), and we manually inspect whether the candidates are relevant according to the same binary relevance criteria.

The results show that increasing the depth from $k = 20$ to $k = 30$ introduces, on average, only 0.5 additional relevant requirements per query. Extending further to $k = 40$ increases this number to approximately 0.9 relevant requirements per query.

At the same time, each increase in depth introduces a fixed number of additional candidates (10 for top-30 and 20 for top-40), most of which are not relevant. This shows that deeper pooling adds many non-relevant candidates and only a small number of additional relevant items. This is further supported by the low average number of additional relevant requirements observed beyond rank 20.

Based on these results, we can say that the majority of judged relevant requirements in this evaluation setup are already captured within the top-20 results, while only a small number of additional relevant items appear at deeper ranks. Increasing the pooling depth only gives a small improvement in coverage, while requiring much more manual labeling effort.

As a conclusion, this analysis suggests that $k = 20$ provides a reasonable balance between coverage and efficiency, and is therefore used as the

candidate pool size for pooled relevance judging for both datasets v1 and v2 throughout the thesis.

4.5.3 Distance metric comparison

As an additional retrieval analysis, we compared cosine similarity with Euclidean, Manhattan and Chebyshev distance using the same embeddings, query set ($N = 20$), and retrieval depth ($k = 10$).

For dataset v1 (pooled evaluation), cosine and Euclidean distance produced identical results across all metrics (Precision@10 = 0.435, Recall@10 = 0.794, MRR@10 = 0.779). Manhattan distance showed very similar performance, with slightly higher Precision@10 (0.445) and Recall@10 (0.808), and a small improvement in MRR@10 (0.792). In contrast, Chebyshev distance resulted in substantially lower Precision@10 (0.330) and Recall@10 (0.600), although it produced a higher MRR@10 (0.875), which indicates that relevant items were sometimes ranked early despite retrieval quality being weaker overall.

For dataset v2, the same overall pattern was observed. Cosine and Euclidean distance again produced identical results (Precision@10 = 0.440, Recall@10 = 0.737, MRR@10 = 0.802). Manhattan distance remained close to cosine, with slightly lower performance (Precision@10 = 0.425, Recall@10 = 0.712, MRR@10 = 0.774). Chebyshev distance again showed a clear drop in Precision@10 (0.325) and Recall@10 (0.529), while MRR@10 remained relatively high (0.754).

The distance-metric results are summarized in Table 1.

Metric	Precision@10	Recall@10	MRR@10
Dataset v1 (pooled)			
Cosine	0.435	0.794	0.779
Euclidean	0.435	0.794	0.779
Manhattan	0.445	0.808	0.792
Chebyshev	0.330	0.600	0.875
Dataset v2			
Cosine	0.440	0.737	0.802
Euclidean	0.440	0.737	0.802
Manhattan	0.425	0.712	0.774
Chebyshev	0.325	0.529	0.754

Table 1: Comparison of retrieval performance across distance metrics for datasets v1 (pooled) and v2 ($k = 10$).

Overall, cosine similarity and Euclidean distance behave identically under normalized embeddings, which indicates that Euclidean distance does not provide a practical difference in this setting. Manhattan distance introduces only minor variations and remains close in performance. Chebyshev distance, however, consistently reduces Precision@10 and Recall@10, indicating weaker retrieval quality despite having occasionally high MRR values.

These results suggest that cosine similarity is a suitable default choice for the retrieval setup used in this thesis, especially since it behaves identically to Euclidean distance under normalized embeddings and remains competitive across both datasets [3, 22].

4.6 Embedding model comparison

To evaluate the impact of the embedding model on retrieval performance, we compared the baseline embedding model (text-embedding-3-small) with a larger model (text-embedding-3-large) under the pooled evaluation setup. The comparison is limited to the pooled setting, as the baseline results reported earlier in this section were computed only using the smaller embedding model. This comparison was included to examine whether a larger embed-

ding model improves retrieval quality in this requirement retrieval setting.

Dataset v1 (pooled)

- text-embedding-3-small: Precision@10 = 0.435, Recall@10 = 0.794, MRR@10 = 0.779
- text-embedding-3-large: Precision@10 = 0.425, Recall@10 = 0.778, MRR@10 = 0.779

For dataset v1, the larger embedding model leads to a small decrease in both Precision@10 and Recall@10, while MRR@10 remains unchanged. This indicates that the ranking of the first relevant requirement is not affected, but slightly fewer relevant requirements are retrieved overall.

Dataset v2

- text-embedding-3-small: Precision@10 = 0.440, Recall@10 = 0.737, MRR@10 = 0.802
- text-embedding-3-large: Precision@10 = 0.400, Recall@10 = 0.663, MRR@10 = 0.817

For dataset v2, the larger model shows a clearer trade-off. Precision@10 and Recall@10 both decrease, while MRR@10 increases. This means that the larger model tends to rank the first relevant requirement earlier, but retrieves fewer relevant requirements within the top-10 results.

Interpretation Across both datasets, the larger embedding model does not consistently improve overall retrieval quality in this setting. While it can slightly improve early ranking (as reflected by MRR in v2), it consistently reduces Precision@10 and Recall@10.

Overall, these results show that a larger model does not necessarily lead to better retrieval performance. In this setting, the smaller model provides a better observed balance between precision, recall and ranking performance. This suggests that, in this thesis, retrieval quality depends not only on model size but also on the dataset structure and the retrieval setup.

4.7 RAG evaluation (v1 vs v2)

In this section, we summarize the behavior of the RAG component for dataset v1 and dataset v2. The goal here is not to evaluate retrieval quality again, but to examine how the generated requirement reviews behave with respect to structure, grounding and consistency [21].

Dataset v1 For dataset v1, the RAG outputs are structured and follow the fixed prompt format introduced in Section 3. In general, the generated reviews are organized into the same main parts, including related requirements, potential conflicts, missing constraints or assumptions, and notes. The outputs are therefore easy to inspect across queries and they remain aligned with the review task.

At the same time, the explanations given by the LLM are largely grounded in the retrieved evidence, with only limited instances of unsupported references. In most cases, the model presents requirement IDs that are already present in the retrieved top-k set, which suggests that the generation step remains closely tied to the retrieval step [19]. However, it is important to mention that the outputs in v1 are not perfect. A small number of queries still contain unsupported references, mainly in the form of self-references or occasionally additional requirement IDs. For this reason, the v1 results still contain some grounding issues, but overall they remain largely grounded and consistent with the retrieved evidence.

Dataset v2 For dataset v2, the same overall output structure is preserved, but the generated responses appear more detailed. The sections remain consistent across queries, and the answers provide clearer reasoning for why retrieved requirements are related, for potential conflicts or useful for identifying missing constraints.

In addition, in comparison with the v1 outputs, the v2 outputs also appear more stable in terms of format. The structure remains the same across queries, but the content is usually more developed. In particular, the model provides complete sections more often and uses explicit statements such as

"Not enough information" when the evidence is insufficient. This may make the outputs easier to interpret and may help reduce unsupported conclusions.

Grounding and hallucinations Across both datasets, the RAG system is mostly grounded in the retrieved requirements, although some partial grounding issues remain. Grounding coverage measures the share of generated requirement references that are present in the retrieved context.

The grounding and hallucination results are summarized in Table 2.

Property	v1	v2
Generated IDs outside retrieved set (%)	30	40
True hallucinated IDs (not in dataset) (%)	0	0
Average grounding coverage (%)	–	88
Self-references (% of queries)	30	–

Table 2: RAG grounding and hallucination analysis across datasets

The quantitative results are consistent with the qualitative observations. In the first dataset, 30% of queries contain unsupported references, mainly in the form of self-references rather than true hallucinations outside the dataset. In the second dataset, generated IDs outside the retrieved set appear in approximately 40% of queries, but all generated IDs still correspond to valid requirements. This indicates that the model rarely introduces invalid requirement identifiers, but it sometimes refers to valid requirements outside the retrieved context. Furthermore, the average grounding coverage in the second dataset is approximately 88%, meaning that most generated references are directly supported by the retrieved results. Overall, these findings show that the RAG system remains mostly grounded in the retrieval step, with errors mainly related to partial grounding rather than true hallucination.

Overall, the RAG component behaves in a fairly consistent way. The outputs are structured, mostly grounded when it comes to the evidence they provide, and stable across queries. Taken together, these observations suggest that the RAG component can support structured review of retrieved

requirements, while still depending on the quality of the retrieved results.

5 Further Analysis

This section provides a deeper interpretation of the results presented in Section 4. While the results section reports the measured performance, this section discusses possible reasons for the observed patterns and what they mean for requirement review.

5.1 Adaptive-k analysis

The results suggest that there is no single best value of k for all queries in this setup. Many queries reach their highest precision at small values such as $k = 1$ or $k = 3$, while some queries benefit from a larger retrieval depth. This means that the most suitable retrieval depth depends on the query and the review objective.

Despite this, we keep $k = 10$ throughout the main evaluation. Although $k = 10$ does not give the highest precision, it provides a practical balance between ranking quality and retrieval coverage.

Overall, the adaptive-k analysis supports the interpretation that the retrieval system should be understood as a candidate generation step rather than a perfect ranking system.

5.2 Adjusted precision interpretation

Many queries in the evaluation contain only a small number of relevant requirements. This directly affects how Precision@k should be interpreted.

Precision@k measures how clean the retrieved set is, while Adjusted Precision@k measures how complete the retrieved set is compared with what is achievable for that query. For example, in dataset v2, Precision@10 is relatively low but Adjusted Precision@10 is higher, showing that the system retrieves most relevant requirements while still including some non-relevant items.

Overall, adjusted precision helps explain why the system can still retrieve many relevant requirements even when raw precision appears low. It is especially useful in this thesis because many relevance sets are small [8, 10].

5.3 Distance metric comparison

The distance metric comparison shows that cosine similarity is a suitable default choice for this retrieval setup. Cosine similarity and Euclidean distance behave identically because the embeddings are normalized. Manhattan distance produces only minor changes, while Chebyshev distance consistently reduces retrieval quality.

This suggests that changing the distance metric does not substantially improve retrieval performance in this setting. Instead, the main retrieval behavior is determined by the embedding representation and the structure of the requirement data [3, 22].

5.4 Embedding model comparison

The comparison between `text-embedding-3-small` and `text-embedding-3-large` shows that a larger embedding model does not necessarily improve retrieval quality. The larger model sometimes improves early ranking, as reflected by MRR, but this does not lead to better Precision@10 or Recall@10.

This suggests that model size alone is not enough to improve requirement retrieval. In this thesis, the smaller model provides a slightly better balance between precision, recall, and ranking performance. It is also expected to be more computationally efficient due to its smaller size, which makes it a reasonable default choice for the final retrieval pipeline.

5.5 Failure case analysis

The per-query analysis in Section 4 showed that retrieval performance is not uniform across queries. A representative failure case is PG-SUB-001, where no relevant requirements were retrieved within the top-10 results.

Even though Recall@10 is high on average, this does not mean that every query is handled well [8, 7]. These failure cases are important because they show limitations that are hidden by aggregated metrics.

One reason for these failures is semantic mismatch. The embedding model may perform poorly when the query and the relevant requirements describe the same concept using different wording. Another reason is domain-specific terminology, where similar concepts may be expressed differently across products or subsystems. A third reason is that requirements may differ only in small but important details, which embedding models do not always capture well.

These cases show why per-query analysis is necessary. Mean Precision@10 and Recall@10 provide useful summaries, but they can hide difficult queries where the retrieval performance becomes very poor.

6 Discussion

6.1 Interpretation of results

To sum up, the results show similar overall patterns across both datasets. The retrieval system achieves high recall, while precision remains lower. This means that the system retrieves most of the relevant requirements, but also returns non-relevant items.

As discussed earlier, this behavior is expected in embedding-based retrieval, since similarity captures broader semantic relationships rather than strict task-specific relevance [9, 7]. Its main role is to ensure that relevant requirements appear in the retrieved set so that they can be inspected further. In an industrial setting, this means that retrieval may support engineers by quickly surfacing relevant parts of large requirement repositories, even if additional filtering is still needed.

The RAG setup builds on this retrieval behavior. Instead of directly improving the retrieval stage, it adds a structured interpretation based on the top retrieved requirements. In this way, it helps organize the evidence, highlight potential conflicts, and suggest missing assumptions between re-

quirements. It also aligns with recent work that shows that RAG systems are often most useful when they support interpretation and reasoning over evidence rather than replacing the retrieval process completely [19, 21]. This is particularly useful in requirement engineering, where understanding relationships and inconsistencies between requirements is often more important than retrieving them individually.

From an engineering perspective, this behavior may still be practical for exploratory review tasks. In requirement review, the goal is usually to inspect a broader candidate set for tasks such as inconsistency detection, traceability analysis, and validation. A high-recall set therefore provides a reasonable starting point for manual inspection, even if it contains some noise [10].

In addition, the comparison between embedding models shows that increasing model size does not necessarily improve retrieval performance in this setting. While the larger model slightly improves early ranking in some cases, it consistently reduces Precision@10 and Recall@10. This suggests that retrieval performance depends not only on representation capacity, but also on the dataset structure and the retrieval setup. In this thesis, the smaller embedding model provides a more stable balance between ranking quality and coverage.

6.2 Why pooling matters

In the initial setup, relevance labels were only defined within the top-k results of the baseline retrieval system. This made the ground truth partially dependent on the retrieval output and introduced evaluation bias, since relevant requirements outside the top-k set were not labeled [10, 17].

To address this, we introduced pooled evaluation by combining retrieval outputs and RAG-generated references, and by increasing the candidate set from $k = 10$ to $k = 20$ for manual labeling. This provides a broader and less system-dependent approximation of relevance while keeping manual effort manageable.

Changing the evaluation protocol has a direct effect on the reported metrics. In our experiments, the same retrieval outputs led to different con-

clusions when relevance was redefined through pooled judging. The pooled setup therefore provides a less system-dependent estimate of performance within this evaluation framework.

Overall, this is one of the main methodological findings of this thesis. It shows that the evaluation protocol plays a critical role in how retrieval performance is interpreted, and that relying only on the top-k labeling can lead to incomplete or misleading conclusions.

6.3 Evaluation problems

Retrieval performance depends not only on the system itself, but also on the evaluation design. In our setup, query selection, relevance labeling, and the fixed retrieval depth $k = 10$ all influence the reported metrics. Because the query set is relatively small ($N = 20$), even small design changes can affect the results [10, 17].

This means that retrieval evaluation is not a neutral measurement. Instead, it depends on several design choices that must be made explicit, including query selection, relevance definition, candidate pooling, and metric reporting [4].

6.4 Limitations

Despite the controlled and reproducible evaluation setup used in this thesis, there are several limitations that should be considered when interpreting the results.

Limited query set The evaluation is based on a relatively small set of only $N = 20$ queries. While a small set allows for controlled analysis and manual relevance labeling, it limits how representative the statistical results are. As discussed in Section 3, the reported metrics should therefore be interpreted as estimates rather than measures of system performance [8]. A larger and more diverse query set would likely provide a more representative and generalizable evaluation. The bootstrap confidence intervals only capture variability across

the sampled queries and do not include uncertainty from manual relevance labeling or synthetic data generation.

Synthetic dataset The datasets used in this thesis are generated synthetically. Although they aim to reflect key structural properties of industrial requirement repositories, they do not fully capture the complexity, noise, and inconsistencies found in real-world data. As a result, the retrieval performance observed in this work may differ from what would be obtained in an actual industrial environment, which limits the external validity of the results [6].

Binary relevance definition We define relevance using a binary labeling scheme, where each requirement is labeled as either relevant or non-relevant. While this simplifies the evaluation and aligns with common information retrieval practices [10], it does not capture different degrees of relevance. Relevance can vary in strength depending on the context in requirement engineering tasks. This means that the evaluation may not fully reflect the relationships between requirements. It may also affect how precision and recall are interpreted, since partially relevant requirements are treated as non-relevant [17].

Lack of large-scale industrial validation The evaluation is conducted entirely in a controlled experimental setting, without validation on large-scale industrial datasets or with real engineering workflows. Although the system is designed to reflect industrial use cases, it is not tested in a real industrial working environment. This means that factors such as user interaction, tool integration, and domain-specific constraints are not evaluated, which limits the ability to directly assess the system's impact in practice.

RAG evaluation without expert study The RAG component is evaluated mainly based on its structure, such as grounding, output and format consistency. While this provides insight into how the system behaves, it does not include a systematic human expert evaluation of the generated require-

ment reviews. As discussed in recent work on RAG evaluation, assessing usefulness, correctness, and trustworthiness often requires domain experts [21, 18]. This means that the conclusions about the RAG component should be interpreted as preliminary.

Summary Overall, these limitations reflect the trade-offs between control, reproducibility, and realism. The setup supports systematic analysis, but stronger conclusions would likely require larger query sets, real industrial data and expert-based validation.

6.5 Contributions

In this thesis, we make several contributions to the evaluation of semantic retrieval and retrieval-augmented generation (RAG) for requirement review.

Semantic retrieval baseline Firstly, we design and evaluate a semantic retrieval baseline for requirement engineering data. Using dense embeddings and cosine similarity, the system behaves as a high-recall candidate generator, consistent with first-stage retrieval in multi-stage pipelines [9].

Controlled RAG comparison Secondly, we introduce a controlled comparison between a retrieval-only baseline and a RAG-based system. By keeping the dataset, queries, retrieval configuration, and prompt fixed, we study the effect of adding a generation component. The RAG system does not directly change retrieval performance, but adds structured interpretation on top of retrieved requirements [19, 21].

Pooled relevance evaluation Thirdly, we apply pooled relevance judging to reduce dependence on a single retrieval output during evaluation. By combining retrieval outputs and RAG references, we construct a larger candidate pool and show that ground-truth construction can change reported metrics even when retrieval outputs are fixed [10, 17]. We also justify the top-20 pool as a practical balance between coverage and labeling effort.

Statistically grounded evaluation framework Fourthly, we treat retrieval metrics as estimates rather than exact values. Using bootstrap confidence intervals, we quantify uncertainty in the reported evaluation results and avoid over-interpreting small differences under limited query counts and manual relevance labels [8, 4].

Reproducible evaluation pipeline Finally, we implement a reproducible evaluation pipeline in which embeddings, retrieval outputs, relevance labels, pooled candidate sets, and evaluation results are stored and reused across experiments. This supports transparent comparison across configurations and contributes to more consistent experimentation in AI-based requirement engineering [6]. Taken together, these contributions provide both a practical system for requirement review and a structured evaluation framework for this domain.

7 Conclusion

In this thesis, we studied how semantic retrieval and retrieval-augmented generation (RAG) can support requirement review in large-scale requirement repositories. The motivation comes from industrial settings, where engineers need to work with large collections of independent requirements, and manually reviewing them is often time-consuming and difficult to scale.

To solve this problem, we designed a retrieval-based system that represents requirements using dense embeddings and retrieves related requirements based on semantic similarity. Additionally, we implemented a RAG setup that uses the retrieved requirements as context for structured requirement review. In this way, the goal is to support the interpretation of the retrieval results by organizing and explaining relationships between requirements, rather than replacing the retrieval step.

A key part of the work is the evaluation framework. We defined a controlled query-based setup with manually labeled relevance, and we extended it using pooled relevance judging in order to reduce evaluation bias in the manually defined ground truth. We also treated evaluation metrics as sta-

tistical estimates and used bootstrap confidence intervals to quantify uncertainty.

The results show that the retrieval system generally behaves with high recall while retrieving both relevant and non-relevant requirements for the given queries. For example, the baseline evaluation shows a Precision@10 of 0.235 and a Recall@10 of 0.925, confirming the high-recall nature of the system. Under pooled evaluation, Precision@10 increases to approximately 0.44 while recall remains high, which highlights the impact of evaluation design on the interpretation of results. It is generally able to retrieve most of the relevant requirements defined in the ground truth, but it does not always produce a clean ranking, which means that additional interpretation or filtering is needed. Building on this behavior, the RAG component provides structured explanations of the retrieved results, without directly improving retrieval performance.

The main contributions of this work are threefold. First, it demonstrates that semantic retrieval can be used as a high-recall candidate generation mechanism for requirement review. Second, it shows that retrieval-augmented generation can support structured interpretation of retrieved requirements. Third, it provides a statistically grounded evaluation framework that highlights how evaluation design affects the interpretation of retrieval performance.

One of the main findings is that evaluation design has a strong impact on the reported results. This is observed when using pooled relevance instead of top-k labeling, where the same retrieval outputs lead to different metric conclusions.

From a statistical point of view, the results show that evaluation metrics should be treated as estimates and not as exact values. Since the evaluation is based only on a few queries, we should always take into account the uncertainty. Using bootstrap confidence intervals helps avoid over-interpreting small differences and provides a more reliable comparison between evaluation settings [8, 4].

In practice, pooled relevance judging provides a broader estimate of the retrieval performance compared to the original setup. This is particularly

important in settings where the full set of relevant requirements is not known in advance. We also observe that a candidate pool of size 20 provides a reasonable trade-off between capturing many of the relevant requirements and keeping the manual labeling effort manageable in this setting.

At the same time, this highlights that careful evaluation design is necessary for meaningful conclusions.

7.1 Research Questions Answered

RQ1: How well does embedding-based semantic retrieval support requirement review tasks in terms of precision, recall, and ranking performance? The results show that embedding-based retrieval is effective at finding relevant requirements, especially in terms of recall. At the same time, precision is lower, which means that non-relevant requirements are also retrieved. This indicates that the retrieval component is best understood as a high recall candidate generator rather than a precise ranking system.

RQ2: How does the use of retrieval-augmented generation (RAG) support the interpretation and analysis of retrieved requirements? The results show that RAG supports requirement review by producing structured and mostly grounded explanations based on retrieved requirements. It can help organize potential conflicts or missing constraints for interpretation. However, it does not directly improve retrieval performance, since its role is interpretation rather than retrieval optimization.

RQ3: How does the evaluation design, including relevance labeling and statistical analysis, affect the interpretation of retrieval performance? The results show that evaluation design has a clear impact on the conclusions that are reported. When pooled relevance is used instead of top-k labeling, the same retrieval outputs can lead to different metric values and different interpretations. In addition, treating metrics as statistical estimates with uncertainty analysis gives a more reliable basis for comparing evaluation settings.

7.2 Future work and outlook

Future work should focus on validating the approach on real industrial datasets and integrating the system into practical engineering workflows. In addition, it would be important to study how these systems are used in practice to better understand their impact.

Overall, this thesis shows that combining semantic retrieval, structured interpretation, and statistically grounded evaluation can support requirement review in large-scale requirement repositories.

References

[1] Chetan Arora, John Grundy, and Mohamed Abdelrazek. "Advancing Requirements Engineering Through Generative AI: Assessing the Role of LLMs". In: *Generative AI for Effective Software Development*. Ed. by Anh Nguyen-Duc, Pekka Abrahamsson, and Foutse Khomh. Cham: Springer Nature Switzerland, 2024, pp. 129–148. ISBN: 978-3-031-55641-8 978-3-031-55642-5. DOI: 10.1007/978-3-031-55642-5_6.

[2] Chetan Arora, Tomas Herda, and Verena Homm. "Generating Test Scenarios from NL Requirements Using Retrieval-Augmented LLMs: An Industrial Study". In: *2024 IEEE 32nd International Requirements Engineering Conference (RE)*. IEEE, 2024, pp. 240–251.

[3] Karlo Babić et al. "A Comparison of Approaches for Measuring the Semantic Similarity of Short Texts Based on Word Embeddings". In: *Journal of information and organizational sciences* 44.2 (2020), pp. 231–246.

[4] Anne-Laure Boulesteix et al. "A Statistical Framework for Hypothesis Testing in Real Data Comparison Studies". In: *The American Statistician* 69.3 (July 2015), pp. 201–212. ISSN: 0003-1305, 1537-2731. DOI: 10.1080/00031305.2015.1005128.

[5] Kendrick Boyd, Kevin H. Eng, and C. David Page. "Area under the Precision-Recall Curve: Point Estimates and Confidence Intervals". In: *Advanced Information Systems Engineering*. Ed. by David Hutchison et al. Vol. 7908. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 451–466. ISBN: 978-3-642-38708-1 978-3-642-38709-8. DOI: 10.1007/978-3-642-40994-3_29.

[6] Haowei Cheng et al. "Generative AI for Requirements Engineering: A Systematic Literature Review". In: *Software: Practice and Experience* 56.2 (Feb. 2026), pp. 141–170. ISSN: 0038-0644, 1097-024X. DOI: 10.1002/spe.70029.

[7] Khadija M. Elbedweihy et al. “An Overview of Semantic Search Evaluation Initiatives”. In: *Journal of Web Semantics* 30 (2015), pp. 82–105.

[8] Cyril Goutte and Eric Gaussier. “A Probabilistic Interpretation of Precision, Recall and F-Score, with Implication for Evaluation”. In: *Advances in Information Retrieval*. Ed. by David Hutchison et al. Vol. 3408. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 345–359. ISBN: 978-3-540-25295-5 978-3-540-31865-1. DOI: 10.1007/978-3-540-31865-1_25.

[9] Jiafeng Guo et al. “Semantic Models for the First-Stage Retrieval: A Comprehensive Review”. In: *ACM Transactions on Information Systems* 40.4 (Oct. 2022), pp. 1–42. ISSN: 1046-8188, 1558-2868. DOI: 10.1145/3486250.

[10] Donna Harman. *Information Retrieval Evaluation*. Morgan & Claypool Publishers, 2011.

[11] Arshia Hemmat et al. “Research Directions for Using LLM in Software Requirement Engineering: A Systematic Review”. In: *Frontiers in Computer Science* 7 (2025), p. 1519437.

[12] Angelos Hliaoutakis et al. “Information Retrieval by Semantic Similarity”. In: *International journal on semantic Web and information systems (IJSWIS)* 2.3 (2006), pp. 55–73.

[13] Vijay Raghavan, Peter Bollmann, and Gwang S. Jung. “A Critical Investigation of Recall and Precision as Measures of Retrieval System Performance”. In: *ACM Transactions on Information Systems* 7.3 (July 1989), pp. 205–229. ISSN: 1046-8188, 1558-2868. DOI: 10.1145/65943.65945.

[14] Oona Rainio, Jarmo Teuho, and Riku Klén. “Evaluation Metrics and Statistical Tests for Machine Learning”. In: *Scientific Reports* 14.1 (2024), p. 6086.

[15] Nils Reimers and Iryna Gurevych. *Sentence-BERT: Sentence Embeddings Using Siamese BERT-Networks*. Aug. 2019. DOI: 10.48550/arXiv.1908.10084. eprint: 1908.10084.

[16] Sujoy Roychowdhury et al. *Evaluation of RAG Metrics for Question Answering in the Telecom Domain*. July 2024. DOI: 10.48550/arXiv.2407.12873. eprint: 2407.12873.

[17] Tefko Saracevic. "Evaluation of Evaluation in Information Retrieval". In: *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR '95*. Seattle, Washington, United States: ACM Press, 1995, pp. 138–146. ISBN: 978-0-89791-714-8. DOI: 10.1145/215206.215351.

[18] Sebastian Simon et al. *A Methodology for Evaluating RAG Systems: A Case Study On Configuration Dependency Validation*. Oct. 2024. DOI: 10.48550/arXiv.2410.08801. eprint: 2410.08801.

[19] Xiaohua Wang et al. "Searching for Best Practices in Retrieval-Augmented Generation". In: *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. 2024, pp. 17716–17736.

[20] William Webber. *Approximate Recall Confidence Intervals*. Oct. 2012. DOI: 10.48550/arXiv.1202.2880. eprint: 1202.2880.

[21] Hao Yu et al. "Evaluation of Retrieval-Augmented Generation: A Survey". In: *Big Data*. Ed. by Wenwu Zhu et al. Vol. 2301. Singapore: Springer Nature Singapore, 2025, pp. 102–120. ISBN: 978-981-96-1023-5 978-981-96-1024-2. DOI: 10.1007/978-981-96-1024-2_8.

[22] Kaitlyn Zhou et al. *Problems with Cosine as a Measure of Embedding Similarity for High Frequency Words*. May 2022. DOI: 10.48550/arXiv.2205.05092. eprint: 2205.05092.

A Prompt Template

The following prompt template is used in the RAG-based system. The prompt is kept fixed across all experiments to ensure reproducibility.

```
Query requirement
ID: {query_id}
Title: {q_title}
Description: {q_desc}

Retrieved requirements (top-{k})
{retrieved_block}

Task
Using only the query requirement and the retrieved list:
1. Identify likely duplicates / near-duplicates
2. Identify potential conflicts
3. Identify missing constraints

Output format (must follow exactly)

=== Related requirements ===
• ID:  | Reason: <1-2 sentences>

=== Potential conflicts ===
• ID:  | Conflict: <1-2 sentences>
(or write "None")

=== Missing constraints / assumptions ===
• <bullet 1>
• <bullet 2>
(or write "None")

=== Notes ===
```

- `<optional short notes or "None">`

`Constraints`

- `Do not invent requirement IDs.`
- `If no items fit a section, write "None".`
- `Keep the output short and technical.`

B Dataset Structure

The datasets used in this thesis follow a structured requirements format with hierarchical parent-child relationships.

B.1 Schema

Field	**Description**
id	Unique requirement identifier
title	Short requirement title
description	Requirement statement in natural language
type	Requirement type (system, software, hardware, firmware, interface, safety)
parent_id	Parent requirement identifier (empty for top-level requirements)
product	Product family/category
subsystem	Functional subsystem label
priority	Requirement priority (e.g., MUST, SHOULD, MAY)
verification	Verification method (e.g., analysis, test, inspection, demonstration)

Table 3: Requirements dataset schema

B.2 Example Entries

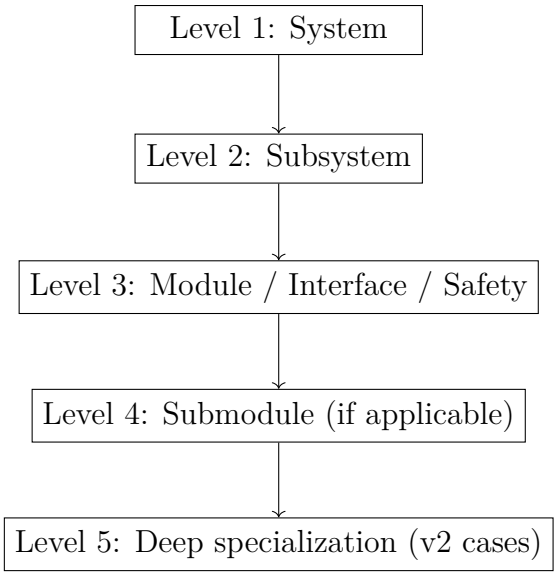
ID	Title	Description	Type	Parent	Product	Subsystem	Priority	Verification
PR-SYS-001	Protection Relay System Overview	The protection relay system shall provide real-time monitoring, protection, and control for medium- and high-voltage power networks.	system	-	Protection Relay	System	MUST	analysis
PR-SYS-002	Interoperability Requirement	The system shall interoperate with external devices using standardized protocols.	system	-	Protection Relay	System	SHOULD	analysis

Table 4: Sample entries from the requirements dataset (v2)

B.3 Dataset Overview

- Dataset v1: 130 requirements
- Dataset v2: 340 requirements
- Number of products: 5
- Requirement types: system, software, hardware, firmware, interface, safety
- Hierarchy depth: variable (up to 5 levels in v2)

B.4 Hierarchy Structure



The first three levels represent the main conceptual hierarchy used in the evaluation, while deeper levels appear only in some extended v2 cases.
Not all requirements use all levels; many entries stop at earlier levels depending on product and subsystem complexity.