

MASTER'S THESIS 2025

Closed Loop Retrieval-Augmented Generation (RAG) for Content-based Recommendations in E-commerce

Eric Martell, Axel Prellner

Elektrotechnik
Datatechnik

ISSN 1650-2884

LU-CS-EX: 2025-48

DEPARTMENT OF COMPUTER SCIENCE

LTH | LUND UNIVERSITY



EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2025-48

**Closed Loop Retrieval-Augmented
Generation (RAG) for Content-based
Recommendations in E-commerce**

Retrieval-Augmented Generation (RAG) i
sluten krets för innehållsbaserade
produkt-rekommendationer inom e-handel

Eric Martell, Axel Prellner

Closed Loop Retrieval-Augmented Generation (RAG) for Content-based Recommendations in E-commerce

Eric Martell
em.ericmartell@gmail.com

Axel Prellner
ax8205pr-s@student.lu.se

July 3, 2025

Master's thesis work carried out at Theca AB, in collaboration with the
Department of Computer Science, Lund University.

Supervisors: Per Runeson, per.runeson@cs.lth.se
Rasmus Ros, rasmus.ros@theca.com

Examiner: Emelie Engström, emelie.engstrom@cs.lth.se

Abstract

This thesis investigates using a closed-loop Retrieval-Augmented Generation (RAG) system for content-based product recommendations in e-commerce. Traditional recommendation systems often struggle to capture the nuanced context of long-form content and the evolving intent of the reader, resulting in generic suggestions that miss the mark in terms of relevance and engagement. To address this, we propose an iterative refinement approach that leverages the language understanding capabilities of Large Language Models (LLMs) and incorporates feedback from previous retrieval cycles to produce more contextually aligned product recommendations. The system was evaluated across multiple Swedish e-commerce platforms using both offline metrics and a user-based interleaving test study. Results demonstrate consistent performance improvements across iterations and a significant advantage over baseline methods. These results highlight the potential of iterative RAG for recommendation tasks and suggest broad applicability beyond e-commerce.

Keywords: Retrieval-Augmented Generation (RAG), Closed-Loop, Recommendations, Prompt Engineering, Iterative Query Refinement, Iterative Retrieval, Large Language Models (LLMs)

Acknowledgements

We would like to thank our supervisor Per Runeson for his steady guidance, constructive feedback, and support throughout the duration of this thesis. We would also like thank our examiner, Emelie Engström whose early advice kept us on the right track.

We extend a special thanks to our industry supervisor Rasmus Ros for his practical insights, technical advice, and for making time to help us navigate the challenges of this project.

Finally, we would like to thank all the employees at Theca Systems AB for welcoming us to the team, answering questions, and contributing to a supportive and stimulating work environment.

Contents

1	Introduction	7
2	Background	9
2.1	Embeddings	9
2.2	Large Language Models (LLMs)	10
2.3	Retrieval Augmented Generation (RAG)	11
2.4	Prompt Engineering	12
2.5	Hyperparameters	13
3	Methodology	15
3.1	Design Science Research Paradigm	16
3.2	Problem Conceptualization	17
3.3	Solution Design	17
3.4	Evaluation & Validation	18
3.4.1	Evaluation Criteria and Benchmarks	18
4	Problem Conceptualization	21
4.1	Problem Constructs for Closed-Loop RAG in E-commerce Recommendations	22
5	Solution Design	23
5.1	Design Constructs	23
5.2	Initial Prototype: Validating the Problem Space	24
5.3	Data Collection & Preprocessing	25
5.3.1	Data Acquisition	25
5.3.2	Data Preprocessing	25
5.4	Closed-Loop Implementation	27
5.4.1	System Architecture	27
5.4.2	Query Refinement	28
5.4.3	Product Retrieval:	29
5.4.4	Reranker	29

5.4.5	Stopping Criteria	30
5.4.6	Benefits of the Closed-Loop Approach	31
5.5	Optimization & Enhancements	31
5.5.1	Fine Tuning	31
5.5.2	Maximal Marginal Relevance	31
6	Evaluation & Validation	33
6.1	Precision, Recall and F1-Score	33
6.2	Categorical Precision, Recall and F1-Score	34
6.3	Embedding-Based Evaluation	34
6.4	Relevance Scoring Metric	35
6.5	Overall Evaluation Strategy	36
6.6	User-Based Evaluation via Interleaving Test	36
6.6.1	Evaluation Metrics and Statistics	38
7	Experimental Evaluation	41
7.1	Experiment Setup	41
7.2	Tuning Hyperparameters	42
7.2.1	Grid Search	43
7.3	Offline Evaluation	45
7.3.1	Naturkompaniet	45
7.3.2	Bygghemma	46
7.3.3	Conclusion	46
7.4	Iterative Convergence	47
7.5	User Study – Interleaving Test Results	52
8	Discussion	53
8.1	Interpretation of Results	53
8.2	Reflection on Design Constructs	54
8.3	Hyperparameter Sensitivity	56
8.4	Threats to Validity	57
8.5	Implications for Future Research and Practice	59
8.6	Further Work	59
9	Conclusion	61
Appendix A		69
A.1	Author Contribution Statement	69

Chapter 1

Introduction

Within the context of e-commerce, effective product recommendations increasingly depend on more than just keyword matching or user browsing history. Traditional recommendation systems primarily rely on static embeddings of product information and predefined user profiles, which cannot easily adapt to rapidly changing user interests or highly context-specific information. As a result, they frequently produce generic or misaligned suggestions—negatively impacting user satisfaction and conversion rates. This is because they fail to adapt to nuanced contextual content such as blog posts, editorial picks, or user-generated reviews, where the relevance of recommendations depends on the evolving preferences of the reader.

To address this, this thesis proposes a **closed-loop RAG system** to enhance the precision and relevance of content-based recommendations in e-commerce. Unlike conventional approaches, which retrieve once and recommend based on pre-computed embeddings, our system introduces an **iterative refinement process**. Each cycle enhances both retrieval and generation by dynamically adjusting search queries and their embeddings based on previous iterations. This allows for product recommendations that evolve with the context of the inspirational content, leading to more precise, intent-aligned advertisements.

To realise this adaptive loop in practice, this approach relies on *Large Language Models* (LLMs)—models that have recently gained significant attention for their ability to generate coherent, context-aware text. Stand-alone LLMs, however, face challenges in domains that demand up-to-date or highly specialised knowledge because they rely on static, pre-trained embeddings. *Retrieval-Augmented Generation* (RAG) overcomes this limitation by coupling an LLM with a live retrieval module, enabling the model to inject external knowledge at inference time and to reformulate queries on the fly [1].

By iteratively refining search queries and embeddings in response to contextual cues and user engagement, the closed-loop RAG system produces recommendations that remain meaningfully aligned with evolving reader intent. This leads to more personalized, relevant, and engaging product suggestions, ultimately enhancing user experience and conversion rates.

This research will be conducted in collaboration with **Theca Systems AB** and **Lund University**, leveraging a design science approach to ensure a systematic cycle of design, implementation, and evaluation of the iterative RAG models [2, 3] in an e-commerce setting.

E-commerce Context

Modern e-commerce platforms do not only rely on search queries and product filters – they increasingly use *inspirational content* to engage users and drive product discovery. This content includes e.g., blog posts, guides, and editorial picks. Such articles are typically authored by editorial teams and embedded throughout the website to promote specific themes or products.

Complementing this content, sites often displays a set of curated product recommendations alongside or embedded within the inspirational content. These product lists are usually assembled by *human curators* who manually selects items that fit the theme, target audience, or intent of the content.

These platforms maintain a large product database, each entry describing a unique product and its attributes. Recommendation systems aim to rank or retrieve relevant products from this database given some form of input (e.g., search query, content page, or user profile).

In traditional systems, this process is either:

- **Manual** (human-curated, as described above)
- **Static** (relying on keyword search or fixed rules)
- Or **non-contextual** (showing the same product regardless of the specific article content)

This thesis focuses on automating and improving this recommendation process using Retrieval-Augmented Generation (RAG), where a large language model (LLM) refines product queries based on inspirational content. Our goal is to create a system that can understand the context and intent of long-form articles and retrieve relevant products from a large catalogue – iteratively improving results until alignment is achieved.

Chapter 2

Background

In this chapter, we introduce key concepts, terminology, and foundational technologies that are essential for the work presented in this thesis. These include embeddings, Large Language Models (LLMs), Retrieval-Augmented Generation (RAG), and Prompt Engineering. The aim is to establish a foundational understanding that supports the design and evaluation of our proposed system.

2.1 Embeddings

Embeddings are a fundamental component in modern natural language processing systems, enabling words, phrases, or entire documents to be represented as numerical vectors. These vectors capture semantic relationships in a high-dimensional space, allowing for efficient similarity comparisons between textual inputs.

The core idea behind embeddings is rooted in distributional semantics — that is, words that appear in similar contexts tend to have similar meanings. Early methods like Word2Vec and GloVe [4] introduced vector representations for individual words based on co-occurrence statistics, enabling simple word-level similarity tasks. However, these static embeddings assign the same vector to a word regardless of context, which limits their expressiveness.

This limitation was addressed by contextual embedding models, such as *BERT* [5], which generate representations dynamically based on surrounding text. These models enable far more nuanced representations, particularly for longer units of text, such as sentences or paragraphs.

Building on this idea, *Sentence-BERT (SBERT)* [6] maintains the same transformer architecture as BERT, but fine-tunes it as a bi-encoder on a large collection of retrieval-style sentence pairs (for example: query vs clicked product). During training, contrastive loss pulls the embeddings of each positive pair together while pushing randomly sampled negatives apart. Thus, with this fine-tuning step, each sentence can be encoded independently, and the resulting sentence embedding can be compared using simple *cosine similarity*, making SBERT

particularly suitable for large-scale retrieval applications.

In the context of Retrieval-Augmented Generation (RAG), embeddings are crucial for both encoding the query and indexing the candidate products or documents. By transforming both into the same embedding space, a *retrieval module* can retrieve items based on semantic similarity (typically using cosine similarity), rather than simple keyword overlap. The module containing all of these embeddings is called a *vector-database* or *vector-store*.

In the recommendation system introduced in Section 1, embeddings were generated using a Swedish BERT-based sentence transformer ¹, enabling nuanced comparisons across long-form blog content and product descriptions. Later in the project, we also experimented with hierarchical embeddings, combining product-level vectors with category-level summaries to introduce structural context into the search process (see Section 5.3.2).

2.2 Large Language Models (LLMs)

A *Large Language Model (LLM)* is a type of program that excels at generating human-like text by using vast amounts of written data to train itself. To do this, it utilizes the concept of a *decoder-only transformer*. The basis for this technique is the transformer.

Transformers

The transformer architecture is a concept introduced by Vaswani et al. [7] in 2017. Transformers utilize embeddings (detailed in Section 2.1) to handle sequential data through a concept called self-attention. Utilizing self-attention means that each token in a sequence is processed by assessing the impact of all other tokens upon it. This makes transformers suitable for handling text since the semantic meaning of a word depends heavily on contextual clues from surrounding text. The original transformer uses an encoder-decoder structure: the encoder processes and contextualizes the input, and the decoder generates the output based on both the encoder's output and previously generated tokens.

Encoder-only Models: BERT Family

One of the first transformer models that gained widespread recognition for its groundbreaking language understanding capabilities was the Bidirectional Encoder Representations from Transformers (BERT) model [5]. BERT is an encoder-only transformer, meaning it is designed to deeply understand input text by generating contextual embeddings.

Decoder-only Models: GPT Family

Around the same time that encoder-only models like BERT gained popularity, *Generative Pretrained Transformer-1* (GPT-1) emerged as the first major decoder-only transformer [8]. A decoder-only transformer bypasses the need for a separate encoder by using its own *masked self-attention mechanism* to both process input and generate output. In encoder models like BERT, embeddings are generated by considering both earlier and later words in the sequence

¹<https://huggingface.co/KBLab/sentence-bert-swedish-cased>

(i.e., bidirectional context). In contrast, decoder-only models only attend to earlier tokens, intentionally masking the context that the later words provide. This constraint during training allows them to learn to generate coherent text by predicting the next token based solely on preceding context.

Capabilities and Limitations

The decoder-only transformer serves as the basis for most modern LLMs, such as ChatGPT and Gemini. Scaling this architecture has led to remarkable language generation capabilities; however, there are some limitations. First, the internal knowledge base of an LLM is static – meaning it can only reflect information present during its pre-training. This means the model may provide outdated or incorrect information when asked about events after its knowledge cut-off or when the information was never included in training [9]. On a small scale, this can be addressed by manually appending relevant information to each query. However, this approach does not scale well. That is where Retrieval-Augmented Generation can be of use.

LLM Temperature

One important factor when optimizing the output of a Large Language Model (LLM) is the *temperature* setting [10]. Temperature controls the level of randomness in the model's responses. When generating text, the LLM assigns probabilities to a range of possible next words based on the given context. The temperature determines how strictly the model follows those probabilities.

A low temperature makes the model more confident and deterministic, favoring the most likely next word. A high temperature introduces more variation by increasing the chance of selecting less probable words, often resulting in more diverse or creative outputs. The choice of temperature can significantly influence the tone, style, and consistency of the response, making it a crucial parameter when tailoring the model's behaviour to a specific task [10].

2.3 Retrieval Augmented Generation (RAG)

Retrieval Augmented Generation (RAG) is a hybrid framework that integrates information retrieval with LLMs to enhance their ability to generate contextually relevant and up-to-date responses. Traditional LLMs rely solely on pre-trained embeddings, which – while effective for general linguistic tasks – suffer from inherent limitations such as knowledge staleness, hallucinations, and difficulty in handling domain-specific queries [1]. By incorporating an external retrieval component, RAG mitigates these issues by dynamically fetching and incorporating relevant knowledge while generating responses, ensuring that the output remains factually accurate and up-to-date [11].

At its core, RAG systems consist of two key modules:

1. **Retriever:** Identifies relevant documents or "knowledge snippets" from an external knowledge base.
2. **Generator:** After retrieving the most relevant information, the LLM conditions its output on both the input query and the retrieved context before producing a response.

The standard RAG approach uses a single retrieval-generation cycle [1], but our proposed system extends this capability by introducing iterative refinement. In this process, we continuously adjust retrieval based on previous outputs, refining search queries and their embeddings in multiple cycles until some iteration stopping criteria are met. Our approach builds on the idea that retrieval should not be static, but rather an evolving process where each step informs the next. Similarly to the iterative retrieval strategies explored by Lee et al. [12], our system continuously refines search queries based on prior retrievals, allowing product recommendations to become increasingly precise and contextually aligned with the inspirational content.

2.4 Prompt Engineering

The term prompt engineering refers to the practice of prompting an LLM in the optimal way to get the sought after response. The quality and relevance of an answer given by an LLM largely depends on how the user formulates the question (or *prompt*) to the LLM, this has given rise to the field of *prompt engineering*.

Manual Prompt Engineering

In its simplest form, prompt engineering involves a human manually refining prompts to improve model outputs. In this area, there are several promising techniques that have been shown to improve responses [10]:

- **Zero-shot prompting:** The simplest form of prompting, where a query is posed without providing any examples or prior context.
- **One-shot prompting:** Involves providing a single example in the prompt to guide the model's response.
- **Few-shot prompting:** Extends the one-shot approach by including multiple examples to better illustrate the desired output format or behaviour.
- **System prompting:** Specifies high-level instructions, such as expected output structure or behavioural traits (e.g. tone, style, personality), before presenting the actual query.
- **Role prompting:** Assigns the model a role to simulate (e.g. expert chemist, software engineer), allowing it to adopt a perspective that guides its response.
- **Chain-of-thought prompting:** Encourages the model to reason step by step before producing a final answer.

These techniques can be applied individually or in combination to significantly enhance prompt effectiveness [10].

Automatic Prompt Engineering

Beyond manual prompt design, there have emerged promising ways to automate this process by utilizing the LLMs themselves [13]. Given their high-level language comprehension, LLMs can be used to generate prompts for themselves through self-prompting techniques. To optimize this process, different strategies can be employed:

- **Positive examples:** The model is given several input-output pairs and tasked with identifying the underlying prompt that would generate those outputs [13].
- **Negative examples:** The model is provided with suboptimal prompts and their corresponding outputs, then asked to suggest a better prompt to improve the responses [14].
- **Iterative improvement:** The model is used to refine its own prompts through an iterative process of self-improvement [15]. This involves analysing prompt failures, generating feedback, and proposing refined prompts in multiple steps.

These techniques have been shown to achieve similar or superior results compared to human-crafted prompt engineering [13–15].

2.5 Hyperparameters

The term *hyperparameters* refers to configurable values in a program that significantly affect its performance. Identifying the optimal hyperparameters is a challenging problem in machine learning and software engineering as a whole [16].

One of the most widely used methods for hyperparameter optimization is called *grid search*. The first stage of a grid search involves selecting a set of promising values for each individual hyperparameter; this forms the grid that will be searched. The next stage consists of testing the different combinations of hyperparameter values. This can be done either exhaustively – testing every possible combination – or in stages, keeping some parameters constant while adjusting others. An exhaustive search is guaranteed to find the optimal combination within the defined grid, but it can be very computationally expensive. In contrast, a staged grid search is not guaranteed to find the best combination. However, it requires less compute, and with a good strategy for which parameters to adjust when, a satisfactory result can often be found.

Chapter 3

Methodology

This section outlines the approach used for the design, implementation, and evaluation phases of the thesis. We adopt a *Design Science Research (DSR)* approach [2, 3] to iteratively develop and refine an artifact – namely a closed-loop RAG system for e-commerce product recommendations, guided by the technological rule: *To achieve improved product recommendations accuracy and relevance in inspirational content, apply a closed-loop RAG approach that iteratively refines retrieval* – and evaluate how well it addresses identified challenges in accuracy, scalability, and reliability.

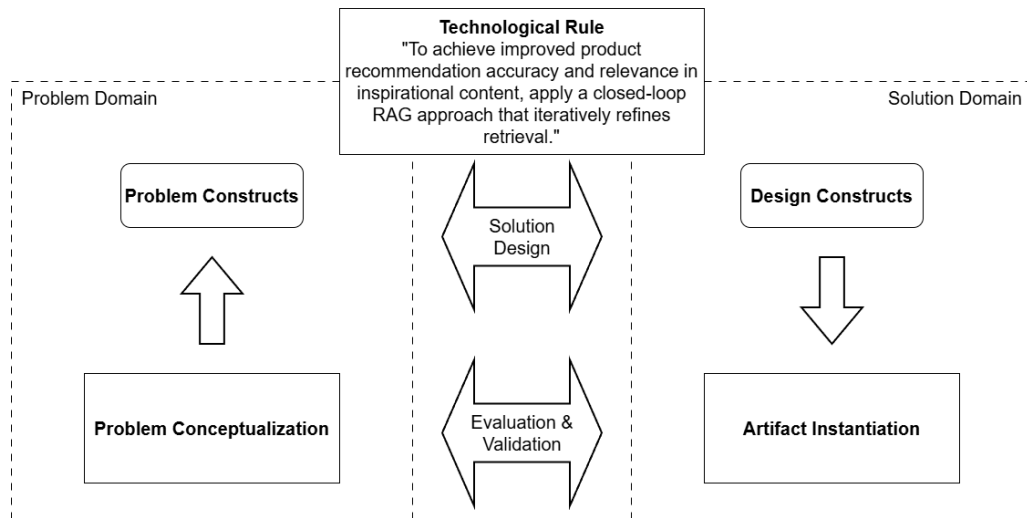


Figure 3.1: The Design Science Research (DSR) process applied in this thesis, adapted from Runeson et al [2, 3]. The model is centered around our key technological rule, and the problem and solution domains are explicitly separated, with corresponding constructs and activities grounded in the e-commerce context.

This research follows the *DSR cycle* as depicted in figure 3.1. In our application of the

DSR paradigm, the problem domain concerns static, non-adaptive recommendation systems commonly found in e-commerce platforms, where product suggestions are manual or lack contextual awareness. This is conceptualized through domain-specific problem constructs (see Section 4.1).

Our solution domain involves a closed-loop Retrieval-Augmented Generation (RAG) system tailored to inspirational content in the e-commerce domain. Key design constructs include an LLM-based query rewriter, a vector-based retrieval module, and a stopping criterion for iterative refinement (see Section 5.1).

3.1 Design Science Research Paradigm

The Design Science Research (DSR) paradigm focuses on the construction and evaluation of artifacts intended to solve identified problems [3, 17]. In this thesis, we apply the DSR paradigm to iteratively develop and refine a closed-loop RAG system tailored for e-commerce product recommendations.

As illustrated in Figure 3.1, adapted from Runeson et al. [3], DSR proceeds through iterative cycles of:

1. **Problem Conceptualization:** We analyze real-world problem instances in e-commerce — specifically the limitations of static, non-adaptive recommendation systems. Through literature review and empirical observations, we abstract these into problem constructs, such as the need for dynamic retrieval and iterative refinement.
2. **Solution Design:** Based on the problem constructs, we propose and develop a design construct – a closed-loop RAG system – guided by a technological rule: “Iteratively refine retrieval queries to improve recommendation relevance.” The artifact integrates prompt engineering, iterative query generation, stopping criteria, and evaluation modules.
3. **Artifact Instantiation:** The designed artifact is instantiated in practice by applying it to real-world datasets (e-commerce products and inspirational content) to create a solution instance. This instance can be abstracted and its attributes can be communicated as guidelines and principles for future iterative RAG system development beyond the e-commerce domain.
4. **Evaluation & Validation:** We conduct empirical validation through both quantitative relevance scoring and qualitative analysis of system outputs, assessing how well the system addresses the original problem constructs. Insights from empirical validation are generalized into a refined technological rule and abstracted to design knowledge.

By grounding our solution in DSR, incorporating both *theory-based inquiry* (prompt engineering, stopping criteria) and *applied experimentation*, we align practical implementation with systematic knowledge creation. This ensures our final artifact is both functionally robust for the immediate problem domain (i.e., e-commerce recommendations) and broadly instructive for future RAG-based systems.

3.2 Problem Conceptualization

The first stage involved conceptualizing the challenges associated with content-based product recommendation in e-commerce. Traditional single-pass systems often fail to align closely with evolving or nuanced user contexts. This motivated the need for a retrieval-centred, closed-loop system capable of iteratively refining outputs.

We conducted a targeted literature review on Retrieval-Augmented Generation, prompt engineering, and stopping criteria design to identify gaps and opportunities, forming the foundation for our design constructs. These constructs are instantiated in our system, as detailed in the next section.

3.3 Solution Design

To manage complexity and ensure systematic refinement, we structure the project into distinct, yet interconnected phases:

- **Initial Prototype (Proof of Concept):** A minimal implementation demonstrated feasibility using limited data. This prototype laid the groundwork for testing basic RAG functionality (retrieval & generation), allowing us to explore key libraries and LLM APIs early on.
- **Data Collection & Preprocessing:** The gathering of product catalogues and associated inspirational content from multiple e-commerce platforms. This data was pre-processed to standardize and structure it.
- **Closed-Loop Implementation:** Involves integrating the iterative feedback loop, which includes writing custom query-building algorithms, refined prompts, and implementing iterative logic for each step (retrieve → generate → refine).
- **Optimization & Enhancements:** Explores additional techniques and optimizations to enhance system efficiency and accuracy.
- **Validation & Refinement:** Employs both *qualitative* and *quantitative* methods to evaluate recommendation relevance, the feedback from which directly informs subsequent adjustments to prompts, retrieval parameters, stopping criteria, etc.

Reflecting DSR principles, each of these phases produces data and insights that feed into the next cycle of iteration [2, 3]. E.g., the system is first implemented, then evaluated, and results are used to refine either the retrieval logic, LLM prompting strategy, or the stopping criteria. Throughout the project we document how design decisions affect recommendation quality and generalizability, hoping to gradually extract broader guidelines for iterative RAG systems.

Through following this structured approach, we seek not only to deliver an effective recommendation system, but also to articulate **design constructs** (see Section 5.1) that can be transferred to other contexts where iterative retrieval and generation strategies are beneficial. Through multiple cycles of empirical validation, we plan to refine prompts, improve retrieval quality, and establish robust stopping conditions. The outcome of this work is a

closed-loop RAG artifact that addresses real-world e-commerce needs while contributing a generalizable template for iterative LLM-based recommendations.

3.4 Evaluation & Validation

In the Design Science Research (DSR) paradigm, the purpose of validation is to determine whether the developed artifact addresses the problem constructs defined in Section 4.1. In our case, this means evaluating whether the closed-loop RAG system generates more contextually aligned, relevant, and accurate product recommendations as compared to non-iterative alternatives.

Evaluation throughout this project was not limited to a final validation phase, but was rather conducted throughout the design process. As design constructs (see Section 5.1) were instantiated, we used empirical insights to refine e.g., prompting strategies, retrieval logic, and stopping conditions. This continuous feedback loop supports the DSR cycle of design → evaluation → refinement.

To guide this process, we define clear evaluation criteria and introduce benchmarks against which the performance of the system is measured. These are detailed in the next section.

3.4.1 Evaluation Criteria and Benchmarks

To evaluate whether our design constructs (see Section 5.1) effectively addresses the problem domain, we define what characterizes a “good” solution in this context. A *successful* closed-loop RAG system should:

- Generate product recommendations that are contextually relevant to inspirational content.
- Adaptively refine its recommendations through iteration, improving performance over time.
- Outperform static baselines (e.g., single-pass retrieval or keyword search) across multiple evaluation metrics.
- Exhibit robustness across product domains (e.g., outdoor gear vs. home renovation).

These criteria reflect the dual nature of our evaluation: (1) assessing the absolute quality of the recommendations, and (2) assessing whether the closed-loop design improves performance over baseline strategies.

To operationalize these criteria, we define and use several benchmarks throughout the thesis. These include:

- **Human-curated product lists**, embedded within the inspirational content on e-commerce sites — used as a form of reference ground truth for offline evaluation.
- **A non-iterative prototype baseline**, which uses static embedding-based search with no refinement or feedback.

- **User preference signals** collected through an interleaving test — used to evaluate real-world recommendation effectiveness.

Together, these benchmarks allow us to triangulate system performance using both reference labels and end-user feedback, in line with DSR's emphasis on multi-perspective validation[18].

Chapter 4

Problem Conceptualization

Modern e-commerce platforms increasingly rely on inspirational content – blog posts, style guides, and long-form editorials – to position products in a narrative that resonates with shoppers. While this approach boosts user engagement, the product recommendations process that accompany such content is predominantly manual, which introduces e.g., subjectivity, inconsistent recommendations, and lack of scalability. Consequently, users are regularly presented with generic or misaligned suggestions, leading to lower click-through rates and lost revenue opportunities since user intent within the world of e-commerce is highly context-dependent and rapidly shifting.

We conducted a targeted literature review of Retrieval-Augmented Generation (RAG), prompt engineering, and stopping criteria to identify current limitations and motivate our system design. This review highlighted that RAG mitigates factual drift by injecting external knowledge into LLM prompts [11]. However, most RAG pipelines terminate after a single retrieval-generation pass [12], forfeiting the opportunity to refine results. To address these limitations, we propose a shift in architecture. Specifically, we outline a **closed-loop system**, where outputs from each retrieval-generation cycle inform the next iteration. This allows the system to iteratively adjust and align retrieval with contextual intent.

Furthermore, empirical gaps remain in how to systematically guide such iterative refinement. Literature on prompt engineering highlights the significant impact of prompt structure on LLM behaviour, yet best practices for multi-iteration scenarios are underexplored. Similarly, while stopping criteria have been discussed in algorithmic contexts (e.g., convergence metrics in optimization), their role in iterative generation-and-retrieval loops within RAG systems lacks formalization.

The findings from this review informed the design construct in Section 5.1 and the evaluation criteria defined in Section 3.4.1.

4.1 Problem Constructs for Closed-Loop RAG in E-commerce Recommendations

Based on our empirical observations of existing recommendation pipelines (see Section 1.2), and the targeted literature review (see Section 4) we identified the problem constructs defined below. These define recurring barriers to content-based product recommendations and motivate our design work within the Design Science Research Framework.

- **PC1 - Manual, non-repeatable curation:** Current practice depends on human editors, leading to inconsistency and limited scalability.
- **PC2 - Static, non-adaptive Retrieval:** Traditional RAG approaches perform a single round of retrieval based on the initial query. This fails to incorporate feedback from previously retrieved results, even when they are partially relevant or clearly misaligned.
- **PC3 - Context misalignment:** Existing systems frequently miss the nuanced context embedded in long-form content. For example, a blog post about "eco-friendly home renovation" may lead to generic home improvement suggestions instead of focusing on sustainable or locally sourced materials, unless the system is able to iteratively align with the full scope of the content.
- **PC4 - Prompt Sensitivity and Instability:** The performance of the retrieval process is highly sensitive to how the prompt is phrased [19], particularly in single-pass systems. Minor changes to phrasing or focus often produce widely divergent results, even when the core intent remains the same. This creates instability and undermines reproducibility—especially when moving from one content domain to another.

In response, we conceptualize a closed-loop Retrieval-Augmented Generation (RAG) system that iteratively refines its recommendations based on both the original content and the results of previous iterations. This approach transforms the retrieval process from a one-time action into an evolving feedback loop, where each cycle informs the next. The problem construct guiding this thesis can be summarized as:

“How can we design a retrieval-centered, closed-loop recommendation system that dynamically adapts to evolving context and improves product relevance in long-form, inspirational e-commerce content?”

To answer this overarching question, we address three interrelated design challenges:

1. **Prompt Engineering:** How to systematically generate and refine queries that align with nuanced content and retrieved product attributes.
2. **Iterative Refinement Mechanism:** How to structure the system so that feedback from previous iterations improves both search and generation quality.
3. **Stopping Criteria:** How to determine, with measurable confidence, when the iterative process has reached a satisfactory level of recommendation alignment.

These challenges define the design requirements for the artifact and guide the subsequent phases of system development, evaluation, and refinement. By framing the problem through DSR constructs, we ensure alignment with both the practical needs of e-commerce platforms and the methodological rigor of the Design Science Research Paradigm [2, 3].

Chapter 5

Solution Design

Following the DSR cycle outlined in Section 3.3, this section specifies the design construct – a closed-loop RAG artifact – instantiates it, and justifies design decisions guided by the key technological rule "to achieve improved product recommendation accuracy and relevance in inspirational content, *apply* a closed-loop RAG approach that iteratively refines retrieval."

5.1 Design Constructs

Guided by the technological rule stated above, we translate the problem constructs in Section 4.1 into mutually reinforcing **design constructs**. Each construct represents a purposeful choice about *what* the artefact must contain so that later subsections can focus on *how* those choices are instantiated and empirically refined.

- **DC1 – Closed-Loop RAG Architecture.**
The core artefact involves an iterative pipeline where outputs from each retrieval-generation cycle inform subsequent iterations, progressively aligning results with contextual nuances.
- **DC2 – Product Relevance Score**
The retrieved products are given a score based on how relevant they are to the text from which they were recommended.
- **DC3 – Query Rewriter.**
A component that stores rationales and retrieval statistics from previous iterations and uses them to generate refined queries.
- **DC4 – Stopping Criteria.**
Iteration halts when further refinement no longer yields material gains.

- **DC5 – Data Foundation & Embedding Store.**
A preprocessed, embedded product database enables semantic similarity search across descriptive fields and keywords.
- **DC6 – Evaluation Instrumentation.**
Tools that collect metrics for each iteration, used to feed evidence into subsequent design loops and lay the groundwork for Section 6.

Each construct will be revisited after its initial implementation based on empirical evidence that will either confirm the current design choice or trigger refinement in the subsequent design loop, closing the DSR cycle with every design loop 3.

5.2 Initial Prototype: Validating the Problem Space

To assess the feasibility of applying Retrieval-Augmented Generation (RAG) to content-based product recommendation, we developed a minimal viable prototype implementing a single-pass RAG system. This initial instantiation did not include any iterative refinement logic and used basic prompts without tuning. The purpose of this prototype was not to optimize recommendation quality, but rather to validate the core assumptions underlying our problem constructs.

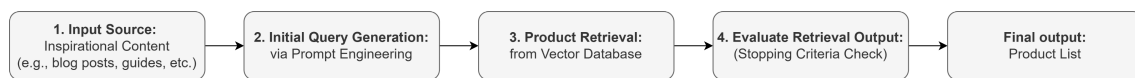


Figure 5.1: Basic overview of the initial prototype pipeline

As illustrated in Figure 5.1, this prototype took inspirational content – such as blog posts – and used an LLM to generate a single retrieval query, which was then executed over a hastily assembled embedded product database. Retrieved products were returned directly without feedback or re-querying. While primitive by design, this implementation highlighted several key insights:

- **Static retrieval** frequently misaligned with the nuanced intent expressed in the inspirational content.
- **Prompt quality** had significant impact on retrieval outcomes, even in single-pass setup.
- **Partial relevance** in many outputs suggested that incorporating feedback across iterations could help significantly improve retrieval outcomes.

These observations provided early empirical support for the problem constructs identified in Section 4.1. This validated the need for a closed-loop architecture that could iteratively refine queries and recommendations.

Thus, while limited in scope, the initial prototype served as a proof of concept that confirmed the viability of our proposed direction and justified investment in more advanced, feedback-driven iterations. The insights gathered here directly informed the design constructs introduced in Section 5.1 and guided the subsequent phases of system development under the DSR paradigm.

5.3 Data Collection & Preprocessing

Before instantiating the initial closed-loop RAG system, we collected data from a diverse set of major Swedish e-commerce platforms that differ significantly in both product range and target audience. Our goal was to construct a representative testbed for exploring recommendation relevance across varied domains, and to ensure sufficient contextual richness for iterative refinement.

We targeted two primary data sources:

- **Product listings:** Structured product information such as names, descriptions and categories.
- **Inspirational content:** Blog posts, editorials, or guides authored by the platform themselves, each associated with hand-picked product suggestions by a site administrator.

This pairing of narrative content and curated recommendations enabled system testing and evaluation: inspirational content served as input to our RAG system, while the associated hand-picked products offered a benchmark for assessing output quality.

5.3.1 Data Acquisition

To collect data on products and their associated inspirational content, we developed custom web-crawlers¹ for each platform to automatically extract structured data. The web-crawling process involved iteratively traversing each site's URLs, parsing relevant HTML sections, and storing the output in a unified format across domains. Table 5.1 summarizes the volume of content that was retrieved from each site.

Platform	Product Count	Blog Posts
Naturkompaniet	7,568	188
Rusta	2,896	82
Apoteket	33,090	210
Bygghemma	114,410	1,063

Table 5.1: Lists different e-commerce platforms and the their total product- and blog (inspirational content) count. The product count does not include variations of the same product (e.g., size or colour).

5.3.2 Data Preprocessing

Given significant structural differences across platforms, a standardization and enrichment process was required to make the collected data suitable for semantic retrieval and prompt

¹A web-crawler essentially simulates the action of a real user on the web, loading and parsing HTML content so that relevant information can be extracted.

generation. This preprocessing phase was essential to operationalizing DC5 (Data Foundation & Embedding Store) and ensuring downstream tasks – like vector search and query formulation – could operate over a uniform, high-quality corpus.

Key preprocessing steps included:

- **Category Extraction:** We identified relevant categories either from the product URLs (e.g., using regex patterns on URLs `'https://www.site.com/category-1/category-2/...'`) or from specific metadata embedded within the HTML that we extracted with our crawlers. This step ensured products could be grouped by their main- or sub-category, which improved our ability to generate accurate search and recommendation queries later on.
- **Deduplication:** Only a single *version* of each product was preserved – removing variations of the same product (e.g., size or colour). Thereby simplifying subsequent indexing and retrieval tasks.
- **Keyword extraction:** For each product, an LLM extracted 5–10 key terms (brand, use case, features). These served as an auxiliary embedding input, allowing us to later compare retrieval based on full descriptions versus attribute-based representations.
- **Length Normalization:** For products whose description were too short, we used an LLM to expand the text, thereby adding useful contextual details that could help improve retrieval. Conversely, if the product description exceeded the limit for the vector store we used (512 tokens²), a LLM was tasked with summarizing it to preserve key information while reducing its length. Exceeding the limit would lose potentially important information since the model would only use the first 512 tokens in the text.
- **Enriching Descriptions:** As the project evolved, we observed that many product descriptions – regardless of length – lacked product-specific focus. They often included comparisons, marketing language, or general category context that diluted the semantic representation of the actual product. In response, we introduced a second-stage preprocessing step in which all descriptions were rewritten using a prompt-guided LLM. This prompt instructed the model to:
 - Focus solely on the product in question.
 - Discard comparative or contextual references to other items.
 - Emphasize key attributes (e.g., use case, materials, brand, distinctive features).
 - Limit the output to a target token range that would not exceed our embedding model's limit (512 tokens).

This universal rewriting approach helped ensure that resulting embeddings captured descriptive, product-level semantics, thereby improving retrieval precision. This shift – from length-based normalization to universal semantic enrichment – represents a refinement in our data design, informed by empirical observations and aligned with the DSR principle of iterative artifact improvement.

²As a basic guideline 1 word $\approx$ 1.33 tokens.

After this standardization and enrichment process, each product record was embedded using a Swedish sentence-transformer model [20]³ based on *BERT* [5]. We stored the vector representations for both the main product descriptions and the extracted keywords separately, maintaining consistent product indices in a unified database. This setup supports more granular retrieval since the search engine can query the product description vectors and keyword vectors independently, or in combination, to produce refined matches.

However, as retrieval scenarios became more complex, we observed that semantic overlap across products in separate, yet similar, categories could dilute result diversity or lead to unintended generalization. To address this, we later extended the embedding scheme by introducing another representation: hierarchical category embeddings. In this approach every category was embedded separately (e.g., Clothes, Jackets), and each product was tagged with a hierarchical category path (e.g., Home → Clothes → Jackets). The final embedding was then a weighted mean between the category embedding(s) and the standard product embedding.

This hierarchical embedding structure allows the retrieval module to consider not only fine-grained product semantics but also higher-level category alignment—supporting more context-aware and diverse recommendation outputs. It reflects a key DSR iteration: early design decisions (flat embeddings) were empirically revised to better align with complex recommendation requirements.

5.4 Closed-Loop Implementation

The core idea behind our closed-loop RAG system is to iterate between retrieval and generation until a sufficiently relevant set of product recommendations emerges. This is in contrast to the traditional single-pass RAG approach, where the input text is used to perform a single retrieval of a static set of documents which the LLM uses to generate its answer. By repeatedly refining the search query in several passes, our system aims to yield more accurate results that closely align with the user intent.

This design directly instantiates DC1-DC4 and was guided by early observations that single-pass systems fail to capture nuanced user intent or adjust based on partially relevant retrievals.

5.4.1 System Architecture

As a high-level overview, our implementation follows the pipeline illustrated in figure 5.2:

1. **Input:** The inspirational user-facing content is fed into the system.
2. **Initial Query Generation:** The LLM is prompted to generate an initial product search query based on the text.
3. **Product Retrieval:** A retrieval module performs a vector-based search over a product database.
4. **Relevance Scoring:** The system evaluates the retrieved products based on their contextual relevance.

³<https://huggingface.co/KBLab/sentence-bert-swedish-cased>

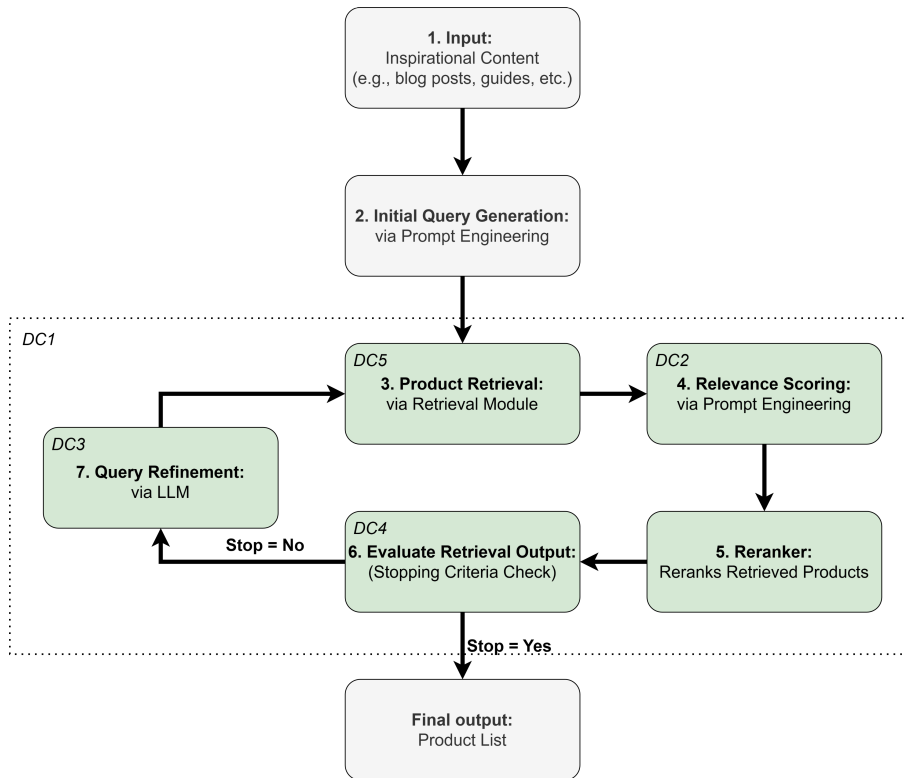


Figure 5.2: Basic overview of the closed-loop pipeline, with the corresponding *Design Constructs*

5. **Reranker:** The relevance scores are used to choose the most relevant subset of the retrieved products. This is detailed in Section 5.4.4.
6. **Evaluate Retrieval Output:** Evaluates the retrieved products, includes checking the iteration stopping criteria.
7. **Query Refinement:** The text, previous search query, and scored products are fed back into the LLM to generate a refined, improved search query. This is detailed in Section 5.4.2.

The system then repeats steps 3-7 in a loop until the stopping criteria (see Section 5.4.5) are met. This iterative refinement enables the system to gradually converge on products that best align both with the tone of the original content and with insights gained from earlier cycles.

5.4.2 Query Refinement

During each iteration, the LLM performs two key steps:

1. **Contextual Integration:** The text from the previous step – comprising the blog post, the current list of scored candidate products, and system-generated context (such as a short rationales for product ratings) – is passed to the model as input.
2. **Query Refinement:** The LLM outputs a refined search query based on the newly provided context. This query is later fed into a retrieval engine.

By incorporating both the original context and the data from previous iterations, the system continuously improves on its previous outputs. When the LLM receives its data from the previous iterations, it can use the information provided by the product ratings to generate a better search query. If a product has received a low score, then the search query should be changed so that the retrieval process no longer yields that product in its search. If, on the other hand, a product has received a high score, the LLM should try to make sure that the improved search query would still find that same product in the retrieval process of the next iteration. This iterative refinement will then continue until some stopping criteria (see Section 5.4.5) are met.

This behaviour embodies the iterative refinement mechanism central to our artifact design and echoes best practices in interactive information retrieval.

Feedback and Refinement. Each iteration also records short explanations – often referred to as "rationales" or "justification" – for why each product might be relevant. These rationales serve two purposes: (a) they are appended to the prompt in the next iterations, helping the LLM learn from prior retrievals; and (b) they give developers and end-users insights into how the system arrived at its recommendations, improving transparency. This aligns with design-science guidelines that emphasize iterative development and evaluation in research artifacts [2].

5.4.3 Product Retrieval:

The retrieval engine forms the core of the system's iterative search capability, see Section 5.3.2 for more details. Initially, we implemented a dual-embedding scheme where each product was represented with two separate embeddings – one based on its description and another based on extracted keywords. These were stored in a vector database and queried either independently or in combination using a weighted mean.

However, empirical testing revealed limitations in this structure. In particular, non-product specific semantics cluttered many product descriptions which in many cases led to very different products having very similar embeddings – which is why we transitioned to a hierarchical embedding model instead.

This modular, adaptive retrieval pipeline operationalizes DC1 (Closed-Loop Architecture) and DC5 (Embedding Store), while illustrating a clear instance of artifact evolution — where design changes were driven by observed limitations and refined through iteration in line with DSR methodology.

5.4.4 Reranker

When the retrieval engine has been used to retrieve the best matching products, based on embedding distance, we need a way to determine which of these products are most relevant to recommend to a potential customer. To do this we consider each product separately and give it a relevance score – a process described in detail later in this section. After all products have been scored we rank them according to their relevance and select the top-performing ones to use as our final recommendations for that iteration.

Product Ratings Obtaining accurate relevance scores for the different products recommended by our system is a crucial part of our RAG implementation. However, assessing the relevance of a certain product in relation to a text is inherently complex. It requires not only an understanding of the text’s surface content, but also its underlying themes and the perspective of a potential customer engaging with it.

Simply comparing the text embeddings with the product embeddings often fails to capture this nuance. High embedding similarity does not necessarily equate true contextual relevance and can even lead to very poor recommendations. For instance, consider that you are reading a text discussing how to properly wash waterproof clothing items. Simply comparing the embeddings would likely result in all different types of detergent getting a high relevance score, due to shared terminology. However, this misses the critical nuance that only certain detergents are suitable for waterproof garments – others may degrade the fabric and thus be inappropriate to recommend.

To address this challenge, we introduce a design construct for rating product relevance that supports context-aware evaluation and that can account for the complex themes present in natural language. To generate accurate scores, we use an LLM, guided through carefully designed prompt engineering techniques as described in the next Section.

Automatic Prompt Engineering: To receive an accurate relevance score for a product, we propose that an LLM can be employed as long as a sufficiently satisfactory prompt is used to guide it. The works of Ye *et al.* [15] acted as a rough template in our efforts to find this prompt.

We adopted an iterative method to determine the optimal prompt, beginning with a manually crafted one designed to clearly communicate the criteria for relevance scoring to the model. This initial prompt was tested by sending it, along with different texts and products to another model for it to determine the relevancy of the different products. The resulting relevance scores were then compared with our own relevance assessments to judge how well the prompt performed.

The products that had the biggest discrepancy in relevance scoring were then identified and returned back to the original model along with their corresponding text and the original prompt. Using this feedback, the model then tried to refine the original prompt by addressing the shortcomings displayed by the example given.

For each input prompt the model gave three different refined prompts for which the process described above was repeated. The three best-performing prompts (including prompts from previous iterations), determined by the lowest mean squared error compared to our relevance assessments, were selected to act as the base input prompts for the subsequent iteration. This process was repeated until a satisfactory prompt was found.

5.4.5 Stopping Criteria

Since we are using an iterative RAG process, it is crucial to determine the right point at which to terminate the loop. For each blog post, the process should terminate when we determine that the product suggestions are sufficiently relevant to the text. To do this we are using the rating system described in Section 5.4.4. When all products that are being recommended get a relevance score that is above a certain threshold the product recommendation is deemed to be of sufficient quality and the process terminates.

5.4.6 Benefits of the Closed-Loop Approach

The chief advantage of adopting a closed-loop design is its reactivity: the system does not rely on one-time retrieval, but rather adjusts its queries as it uncovers partial evidence of what is or is not relevant in the set of recommended products. This iterative refinement can be especially useful in an e-commerce setting, where blog posts often contain complex, nuanced descriptions of user preferences. By treating the LLM-generated query as an evolving entity, the system becomes better equipped to capture diverse or ambiguous user requirements.

5.5 Optimization & Enhancements

We explored several different optimization approaches.

5.5.1 Fine Tuning

As part of our effort to optimize retrieval and recommendation precision, we experimented with fine-tuning a language model to enhance the system's ability to assess product relevance in relation to long-form inspirational content. The goal was to improve our prompt-based scoring of product relevance using a task-specific fine-tuned model from OpenAI.

Method: We built a fine-tuning dataset from our collection of blog posts and their associated product recommendations. For each blog:

- **Positive examples** consisted of products that were manually recommended in the blog post.
- **Negative examples** were randomly selected from our product database.

However, we observed that random negative sampling could produce false negatives — i.e., products that were not listed but were nonetheless relevant. To mitigate this, we implemented a rudimentary category overlap scoring heuristic where products with the same or closely related categories to positive examples were given a higher score. Each training example thus included the blog post text, a product description, and a label in the interval $[0, 1]$ indicating relevance (0 = not relevant, 1 = relevant).

Results: Despite initial promise, the fine-tuned model underperformed compared to our solely prompt-based scoring system. As a result, we chose not to incorporate the fine-tuned model into the final artifact. This negative result reinforces the viability of prompt-based approaches for context-heavy tasks.

5.5.2 Maximal Marginal Relevance

As an attempt to improve the reranking step in our process we tried to take measures not to recommend products that were similar to poorly ranked products from previous iterations. With our base reranking approach, we simply chose some predefined number of products, based on how close those product embeddings were to our search query embedding. Those

candidates were then scored for relevance, and the top-ranked products were recommended in that iteration.

To prevent the model from recommending products that had already been processed and deemed irrelevant we maintained a record of the worst rated product recommendations from each iteration. In subsequent iterations, we used this set of negatively rated products to influence reranking by applying the *Maximal Marginal Relevance* (MMR) criterion [21]:

$$\text{MMR}(p_i) = \lambda \cdot \text{sim}(p_i, q) - (1 - \lambda) \cdot \max_{p_j \in S} \text{sim}(p_i, p_j) \quad (5.1)$$

Where:

d_i	a candidate product
q	the search query embedding
S	the set of previously low-rated products
$\text{sim}(a, b)$	cosine similarity between embeddings a and b
λ	a trade-off parameter between relevance and dissimilarity

MMR is designed to balance two competing goals: recommending products that are relevant to the current query (high similarity with q) while avoiding products that are too similar to known poor candidates (low similarity with any $p_j \in S$). By adjusting the trade-off parameter λ , we can control the degree to which the system favours avoidance over raw relevance.

Our first retrieval step – selecting the most query-relevant product embeddings – remained unchanged. However, we increased the number of initial candidates and applied MMR to rerank them, penalizing embeddings similar to previously rejected ones.

Chapter 6

Evaluation & Validation

As defined in Section 1 and further formalized in Section 3.4.1, a successful implementation of our closed-loop RAG system in this context must generate contextually relevant product recommendations, demonstrate improvement over iterations, and outperform non-iterative baselines across multiple evaluation criteria.

This chapter presents the concrete evaluation strategies used to assess whether our system meets these criteria. These include both offline and user-based methods, designed to cover different aspects of performance: semantic similarity to curated references, categorical and exact-match accuracy, user-perceived relevance, and convergence across iterations.

The strategies used correspond to the following evaluation goals:

- **Contextual Relevance:** Measured using LLM-based relevance scoring and embedding similarity metrics.
- **Improvement over Iterations:** Assessed by tracking metric progression over repeated RAG cycles.
- **Comparison to Baselines:** Evaluated through controlled offline benchmarks and user preference via interleaving tests.
- **Robustness across Domains:** Measured by comparing results across multiple e-commerce platforms with distinct content and product characteristics.

Together, these evaluation strategies allow us to validate the system against the criteria established in Section 3.4.1, ensuring both rigor and relevance in our findings.

6.1 Precision, Recall and F1-Score

As a foundational evaluation method we compute the precision, recall and f1-score [22] of the product recommendations by comparing them to the human-curated ones on the website.

- **Precision** is defined as the percentage of the model recommendations that were also a part of the website recommendations. This tests to see the correctness of the model recommendations.
- **Recall** measures the percentage of the website recommended products that were also recommended by our model. This tests whether our model can recover the relevant products that should be recommended for the text.
- **F1-score** is the harmonic mean of precision and recall. This provides a balanced metric that take both factors into account. This score gives us an indication of how effectively the model can replicate the recommendations made by humans.

Drawbacks of Precision, Recall and F1-score: A key limitation of this evaluation method is that it assumes the human-curated recommendations are exhaustive, that they cover all possible relevant products. As a result, a product is only counted as a true positive for precision if it appears in the curated list, and similarly, only products in that list can contribute to recall – even if other relevant products exist but were not included.

6.2 Categorical Precision, Recall and F1-Score

To address the limitations of exact product matching, we introduced category-based evaluation. In this approach, precision, recall, and F1-score are computed as before, but a product is considered a true positive if it belongs to the same category as a product in the opposing recommendation set, rather than requiring an exact match.

This helped to mitigate the problem with using exact product matching, but it did not resolve it entirely. Products that can be considered relevant for a text do not necessarily share a category with other relevant products. This can either be because some categories are very niche and only contain a few number of highly specialized products, or it can be because the text is very broad and can reasonably be used to promote very different kinds of products.

6.3 Embedding-Based Evaluation

To further address the limitations of exact and category-based matching, we introduce a custom embedding-based evaluation approach inspired by traditional precision and recall. Instead of requiring discrete matches, this method compares the semantic similarity between product embeddings from our model's recommendations and those from the human-curated lists.

While the general structure resembles precision and recall, there are key differences. Traditional precision and recall assign a binary score to each product: it either matches (score of 1) or it does not (score of 0). In contrast, our custom embedding-based method produces a continuous similarity score based on vector closeness. We define two complementary metrics: *precision distance* and *recall distance*.

Precision Distance is calculated by comparing each product recommended by our model to all products in the human-curated list. For each model-recommended product, we com-

pute the similarity to the closest human-curated product and take the average over all model outputs:

$$\text{Precision distance} = \frac{1}{N} \sum_{r_m} \max_{r_w} \text{sim}(r_m, r_w) \quad (6.1)$$

where r_m are products recommended by the model, r_w are products from the curated list, and $\text{sim}(\cdot, \cdot)$ is a similarity function in embedding space. This metric reflects the semantic closeness of the model's output to human recommendations, analogous to standard precision.

Recall Distance inverts this relationship. Here, each product from the curated list is compared to the closest match in the model's recommendations:

$$\text{Recall distance} = \frac{1}{N} \sum_{r_w} \max_{r_m} \text{sim}(r_w, r_m) \quad (6.2)$$

This mirrors recall by measuring how well the model captures the semantics of the human-curated set. Together, these metrics provide a more flexible and gradient-based alternative to exact matching, making relevant evaluation possible even when no direct matches exist.

This method is the best metric we have for comparing our model recommendations with the human-curated ones. However, there are still issues with it.

Drawbacks of Embedding-Based Evaluation: The main limitation of our embedding-based approach mirrors the broader challenge of objectively scoring product relevance based solely on the text. Two product embeddings can be very close to each other, even if one product is highly relevant to a text and the other one is not.

Consider, once again, the example of a text containing information about washing waterproof clothing items. In the human-curated list there would most likely be a recommendation for a detergent specifically formulated for waterproof materials. The embeddings for that product would likely be very similar to the embeddings for a detergent that did not share that specific functionality. As a result, our evaluation metric would give that product a high score.

This misalignment undermines the objective of relevance scoring. In this case, recommending a non-specialized detergent is not only irrelevant but potentially harmful to the garments and, by extension, detrimental to the credibility of the recommending system.

6.4 Relevance Scoring Metric

One other evaluation metric we used – that was independent of the human-curated recommendations – was the relevance score generated by the LLM during the RAG iteration loop. By using the technique of simply prompting the LLM in the optimal way we have been able to establish that a highly accurate relevance score can be computed. We can use this relevance score to accurately determine whether a final product recommendation is relevant or not.

Drawback of the Relevance Scoring Metric: The main issue with this approach is that it introduces an inherent evaluation bias. Since the same relevance score is used both to guide the RAG loop and to assess its performance, there is a risk of circular

reasoning. The RAG process is designed to terminate once the relevance scores surpass a predefined threshold, meaning that later iterations are, by design, optimized to yield higher scores. As a result, even if the scoring mechanism is flawed or misaligned with true user relevance, the evaluation metric will likely still report favorable outcomes. This makes it difficult to detect or diagnose weaknesses in the relevance scoring itself.

6.5 Overall Evaluation Strategy

We assess the effectiveness of our product recommendation method based on two core criteria: *Overall Relevance Score* and *Improvement Over Iterations*. These dimensions enable us to evaluate both the absolute quality of the final recommendations and the value of the iterative RAG process itself.

Overall Relevance Score: To determine whether the final set of recommended products is relevant to the source text, we aggregate insights from the multiple different evaluation metrics discussed in previous sections. Since some of these metrics are complementary, we consider strong performance across all of them to be a reliable sign of high relevance.

Improvement Over Iterations: A core feature of our system is its iterative RAG loop, which is designed to progressively refine the recommended products. For this design to be justified, we expect the relevance score to improve across iterations. The recommendations from later iterations – particularly the terminating one – should perform better than those from earlier stages.

To evaluate this, we track the selected evaluation metrics at each iteration and analyse their progression over time. If our system is effective, that would result in a general upward trend in these scores, which would indicate that the model is learning from previous iterations and is converging towards proper, contextually relevant recommendations.

However, if the relevance of the products remains flat or declines across iterations, that would suggest that the iterative process is unnecessary. As such, this dimension of evaluation is not only a measure of performance but also a validation of our system architecture.

By combining both the final recommendation quality and the trajectory of improvement, our evaluation strategy supports a nuanced understanding of the system's strengths, weaknesses, and potential areas for refinement.

6.6 User-Based Evaluation via Interleaving Test

To complement our offline evaluation strategies and mitigate the limitations associated with relying solely on human-curated recommendation lists or automated scoring metrics, we implemented a *user-based evaluation* method in the form of an *interleaving test* [23]. This approach introduces real users into the evaluation loop and enables us to directly compare two or more recommendation strategies based on user preferences.

Interleaving Methodology: We adopted a variation of *team draft interleaving*, a commonly used method in information retrieval to compare two ranked lists of recommendations [24]. In our case, users were presented with the original html-webpage of the inspirational content but with the products in the recommendations replaced with a product list with products from two recommendation strategies interleaved (baseline, closed-loop RAG, website curations).

Data Collection Platform: To enable user participation, we developed a simple lightweight Flask-based web application¹. The system:

- Randomly selects inspirational content and their corresponding product sets
- Interleaves products from two different strategies
- Collects user relevance ratings (0 or 1) and metadata
- Associates all ratings with a user identifier

This setup allows us to collect human-centred feedback over time. It also ensures distributed and randomized exposure of different recommendation strategies, reducing systematic biases and providing empirical evidence for model preference.

Benefits and Limitations: This method addresses key drawbacks of previously mentioned metrics:

- It provides direct evidence of user satisfaction
- It accounts for subjective relevance, which cannot always be inferred from embedding similarity or category overlap.
- It enables side-by-side evaluation of alternative system configurations in a natural setting

However, new challenges are also introduced:

- Ratings can vary due to subjective interpretation and user inconsistency.
- Sample size and rater engagement levels influence statistical robustness.

Despite these challenges, the interleaving test represents a crucial evaluation layer. It strengthens the empirical foundation for system refinement and supports our broader design science research goals by directly validation how well the closed-loop RAG system aligns with real-world user intent [2].

¹Flask is a micro web framework written in Python

6.6.1 Evaluation Metrics and Statistics

To analyze the results of our user-based interleaving tests, we adopt and adapt the team draft interleaving evaluation framework proposed by Kharitonov et al. [24] to accommodate comparisons between more than two recommendation systems.

We treat a *good* binary rating like a click, a *bad* rating like "no click":

$$r(c) = \begin{cases} 1 & \text{if the user marked the product as good} \\ 0 & \text{otherwise} \end{cases} \quad (6.3)$$

We define the team indicator

$$T(c) = \begin{cases} +1 & \text{product belongs to team Y} \\ -1 & \text{product belongs to team X} \end{cases} \quad (6.4)$$

A single *session* is defined as one user-facing page and its interleaved product recommendations. The win count for strategy *Y* over strategy *X* for session *q* is the familiar *binary win count*:

$$S_{XY}(q) = \sum_{c \in q} r(c)T(c) \quad (6.5)$$

Across *N* sessions the average advantage of *Y* over *X* is:

$$\Delta_{XY} = \frac{1}{N} \sum_{q=1}^N S_{XY}(q) \quad (6.6)$$

The variance of the session scores is:

$$\text{var}[S_{XY}] = \frac{1}{N-1} \sum_{q=1}^N (S_{XY}(q) - \Delta_{XY})^2 \quad (6.7)$$

The resulting non-stratified *z-score* is calculated with:

$$Z_{XY} = \frac{\Delta_{XY}}{\sqrt{\text{var}[S_{XY}]/N}} \quad (6.8)$$

This *z-score* allows us to interpret statistical significance which follows from the standard normal table, e.g., $|Z| \geq 1.96 \rightarrow 95\%$ confidence.

Because we compare multiple system pairs (i.e., baseline vs RAG, and RAG vs website), multiple hypothesis testing is necessary. Conducting each test independently increases the chance of Type I errors (false positives). To address this, we apply the Holm–Bonferroni correction [25] to control the family-wise error rate (FWER).

The procedure is as follows:

1. For each comparison, compute the two-sided p-value:

$$p_{XY} = 2(1 - \phi(|Z_{XY}|))$$

where ϕ is the cumulative distribution function of the standard normal distribution.

2. Rank the $C = \binom{K}{2}$ p-values in ascending order:

$$p_{(1)} \leq p_{(2)} \leq \dots \leq p_{(C)}$$

3. For each rank $k = 1, \dots, C$ compare $p_{(k)}$ with $\alpha/(C - k + 1)$
- If $p_{(k)}$ is larger than this threshold, stop testing and accept the remaining null hypothesis.
 - Otherwise reject $H_{(k)}$ (the null) and proceed to the next.

This method guarantees $\text{FWER} \leq \alpha$ while being uniformly more powerful than traditional Bonferroni correction.

For every pairwise comparison, we report:

- Mean user preference score Δ_{XY} (sign shows the preferred system)
- Corresponding z-score Z_{XY}
- Holm-Bonferroni corrected p-value
- Sample-size target for 95% confidence:

$$N_{95\%} \geq \left(\frac{1.96 \sqrt{\text{var}[S_{XY}]}}{|\Delta_{XY}|} \right)$$

This evaluation pipeline, adapted from Kharitonov et al. [24] provides a stable framework for analysing user preferences, enabling fair and statistically sound comparisons between competing recommendation strategies.

Chapter 7

Experimental Evaluation

In this chapter, we validate the performance of the closed-loop RAG system through a series of experiments conducted on real-world e-commerce data. Following the methodology outlined in Section 6, we evaluate how well our system aligns with the original design objectives, namely improving the relevance, adaptability, and semantic precision of content-based product recommendations.

We begin by detailing the experimental setup, including datasets, evaluation metrics, and baseline configurations. We then present the results of hyperparameter tuning, which was used to determine optimal system configurations for each e-commerce platform. The core of this section focuses on the outcomes of our offline evaluation, with both aggregate and per-site analysis that examine semantic, categorical, and exact-match performance. We also analyse the behaviour of the system across iterations to assess whether the closed-loop design leads to a convergence for the different blogs. The final subsection presents results from our user-based evaluation, offering additional insights into how the system’s recommendations are perceived in practice.

Together, these results provide a comprehensive understanding of both the strengths and limitations of the closed-loop architecture in supporting context-aware product recommendations.

7.1 Experiment Setup

Here we outline the evaluation datasets, recommendation strategies, and the metrics used to validate the performance of our closed-loop RAG system.

Datasets: We used blog posts and product catalogues collected from four Swedish e-commerce platforms (see section 5.3). For this evaluation, we focus on two platforms – Naturkompaniet & Bygghemma – due to their contrasting product catalogues and sizes.

Recommendation Strategies: Each site was evaluated using one or more of the following:

- A closed-loop RAG model using the optimal hyperparameters found in 7.2.
- A non-iterative baseline that uses basic vector search.
- Human curated recommendations (used as ground truth in offline evaluation)

Metrics: We employ a variety of evaluation methods to assess recommendation quality (see Chapter 6):

- Precision, Recall, F1-score (exact product match)
- Categorical Precision/Recall/F1-score (inexact product match)
- Embedding similarity (precision/recall distance)
- LLM-generated relevance scores
- Convergence behaviour over iterations
- User study – interleaving test

Results were tracked across multiple iterations to determine both absolute performance and trends over time.

7.2 Tuning Hyperparameters

To identify the best set of hyperparameters for our model, we conducted a staged grid search. Rather than exhaustively testing every possible combination of parameters – which would be both computationally expensive and time-consuming – we adopted an incremental approach to narrow the search space efficiently.

In the initial stage, we kept a subset of the hyperparameters fixed while systematically varying the others. This allowed us to identify poorly performing configurations early on. Based on these results, we were able to eliminate underperforming values for specific parameters.

In subsequent stages, we focused only on the more promising parameter values identified in the previous rounds. By progressively refining the search space in this manner, we reduced the computational cost while still exploring a diverse range of potentially optimal configurations.

This staged approach to grid search enabled us to balance efficiency with thoroughness, ensuring that we could find high-performing hyperparameter settings without the prohibitive overhead of a full combinatorial search.

Table 7.1 shows the hyperparameters that were identified as essential for tuning our model's performance:

Parameter	Description
emb	The type of embedding and preprocessing used during retrieval. ss denotes standard embeddings with minimally processed descriptions, hs uses hierarchical embeddings with minimal preprocessing, sf uses standard embeddings with fully preprocessed descriptions, and hf combines hierarchical embeddings with fully preprocessed descriptions.
nr	Number of products retrieved in the initial retrieval stage.
rr	Number of products selected for final recommendation after reranking.
temp	Temperature setting used in the LLM for both product rating and query rewriting.
sc	Stopping criterion: the minimum rating that the lowest-ranked product must achieve for the loop to terminate.
lam	Controls the influence of negative examples in subsequent retrievals. It takes a value between 0 and 1 where $\lambda = 1$ means negative products have no effect and a value of 0 means the search query has no effect.
mi	Maximum number of iterations allowed before forced termination of the recommendation loop.

Table 7.1: Hyperparameters used in the staged grid search

7.2.1 Grid Search

In the first stage of our search, we aimed to isolate the effects of a few key parameters by fixing the others to reasonable baseline values. Specifically, we set $nr = 10$, $rr = 5$, $mi = 5$ and $sc = 0.8$. We also noted from a smaller previous grid search that using hierarchical embeddings led to significantly better performance than using standard embeddings. As a result, we only considered hierarchical embeddings at this stage.

With these fixed, we varied the remaining parameters across a range of values. This allowed us to quickly filter out ineffective combinations and focus subsequent stages of the grid search on the most promising areas of the hyperparameter space. The results of this are depicted in Table 7.2.

From this set of results, we were able to identify some hyperparameter configurations that consistently led to lower performance. As a result, these were deemed not to require further investigation. In particular, fully preprocessing the products always resulted in a higher score than not doing so. Furthermore, we found that varying the model temperature did not affect the performance in any statistically significant way. Consequently, we set the temperature value at $temp = 0.8$ in subsequent stages.

Beyond these findings, the impact of the hyperparameters appeared to be very website-specific. No consistent patterns emerged among the optimal hyperparameters for the different sites, which is why we chose to do the subsequent stages of the grid search for each website separately. The results of this are detailed in the next section.

Parameter Values	Precision Distance	Recall Distance	Average Rating	F1 Score	Category F1 Score
emb: hs lam: 0.65 temp: 0.5	0.9379	0.9053	0.9193	0.2017	0.5309
emb: hs lam: 0.65 temp: 0.8	0.9453	0.9097	0.9210	0.2191	0.5493
emb: hs lam: 0.65 temp: 1.0	0.9350	0.8990	0.9160	0.2147	0.5467
emb: hs lam: 0.8 temp: 0.5	0.9426	0.9046	0.9221	0.2117	0.5545
emb: hs lam: 0.8 temp: 0.8	0.9429	0.9069	0.9224	0.2168	0.5778
emb: hs lam: 0.8 temp: 1.0	0.9391	0.9077	0.9238	0.2031	0.5383
emb: hs lam: 1.0 temp: 0.5	0.9392	0.9029	0.9074	0.2057	0.5633
emb: hs lam: 1.0 temp: 0.8	0.9428	0.9070	0.9193	0.2178	0.5677
emb: hs lam: 1.0 temp: 1.0	0.9367	0.8990	0.9066	0.2070	0.5306
emb: hf lam: 0.65 temp: 0.5	0.9508	0.9209	0.9340	0.2206	0.5735
emb: hf lam: 0.65 temp: 0.8	0.9504	0.9160	0.9381	0.2084	0.6037
emb: hf lam: 0.65 temp: 1.0	0.9477	0.9157	0.9241	0.2108	0.5675
emb: hf lam: 0.8 temp: 0.5	0.9473	0.9163	0.9200	0.2106	0.5807
emb: hf lam: 0.8 temp: 0.8	0.9494	0.9165	0.9174	0.2098	0.5919
emb: hf lam: 0.8 temp: 1.0	0.9488	0.9173	0.9341	0.2043	0.5829
emb: hf lam: 1.0 temp: 0.5	0.9468	0.9123	0.9039	0.1992	0.5678
emb: hf lam: 1.0 temp: 0.8	0.9416	0.9120	0.9074	0.2147	0.5799
emb: hf lam: 1.0 temp: 1.0	0.9494	0.9169	0.9169	0.2140	0.5785

Table 7.2: Effect of type of embedding (**emb**), model temperature (**temp**) and negative product effect (**lam**) on different metrics with number of retrieved products ($nr = 10$), number of recommended products ($rr = 5$), maximum iterations ($mi = 5$) and minimum rating ($sc = 0.8$) held constant.

7.3 Offline Evaluation

This section presents the results from our offline evaluation strategies outlined in section 6. For each site, we report performance across multiple metrics using the best-performing hyperparameter configuration. These configurations were identified through the initial stage of the grid search (section 7.2) followed by subsequent website-specific stages.

To assess the effectiveness of our approach, we compared the performance of each site-specific model to our baseline product recommendation system. The baseline did not incorporate any RAG-based components; instead, it relied on a simple vector search using the blog text and the original product descriptions. This comparison allowed us to evaluate whether our iterative, RAG-enhanced approach offered any meaningful improvements over a conventional retrieval setup.

7.3.1 Naturkompaniet

Optimal Hyperparameters

The most promising hyperparameters found in the grid search for Naturkompaniet are presented in Table 7.3. Their corresponding results when run on all blogs on Naturkompaniet’s website are presented in Table 7.4.

Parameter	Value
emb	hf
nr	10
rr	5
temp	0.8
sc	0.9
lam	0.65
mi	10

Table 7.3: Best hyperparameters found in grid search for Naturkompaniet.

Metric	Result
Average Rating	0.9676
Precision Distance	0.9476
Recall Distance	0.8959
F1 Score	0.2190
Category F1 Score	0.5677

Table 7.4: Results for the best hyperparameters for Naturkompaniet.

Baseline Results

The results for the baseline performance are presented in Table 7.5.

Metric	Result
Average Rating	0.7647
Precision Distance	0.8643
Recall Distance	0.8349
F1 Score	0.1362
Category F1 Score	0.3968

Table 7.5: Baseline results for Naturkompaniet.

7.3.2 Bygghemma

Optimal Hyperparameters

The inspirational content on *Bygghemma* is generally diverse and includes broader editorial content (e.g., home renovation, bathroom guides), and has by far the largest product catalogue of the sites we investigated. The best hyperparameters found during grid search are listed in table 7.6.

Parameter	Value
emb	hf
nr	10
rr	5
temp	0.8
sc	0.9
lam	0.65
mi	10

Table 7.6: Best hyperparameters found in grid search for Bygghemma.

Running with the best hyperparameters identified in Table 7.6 on a larger volume of inspirational content yielded the results shown in Table 7.7.

Baseline Results

The results for the baseline performance are presented in Table 7.8.

7.3.3 Conclusion

Across all offline metrics, the closed-loop RAG system outperforms the baseline. This supports the hypothesis that our closed-loop RAG system can be used to improve product rec-

Metric	Result
Average Rating	0.9737
Precision Distance	0.9150
Recall Distance	0.8943
F1 Score	0.1146
Category F1 Score	0.6747

Table 7.7: Results from using the best hyperparameters identified in Table 7.6.

Metric	Result
Average Rating	0.8380
Precision Distance	0.8968
Recall Distance	0.8830
F1 Score	0.0890
Category F1 Score	0.5849

Table 7.8: Baseline results for Bygghemma.

ommendations. The performance gains against the baseline are shown in Table 7.9.

Metric	Improvement Naturkompaniet	Improvement Bygghemma
Average Rating	0.2029	0.1357
Precision Distance	0.0833	0.0182
Recall Distance	0.061	0.0113
F1 Score	0.0828	0.0256
Category F1 Score	0.1709	0.0898

Table 7.9: Improvement over baseline for Naturkompaniet and Bygghemma.

7.4 Iterative Convergence

A key advantage of the closed-loop architecture is its ability to iteratively improve recommendation quality. We tracked system behaviour over multiple iterations and analysed trends across our different evaluation metrics.

Convergence Behaviour: With the optimal hyperparameters detailed in the previous sections, Table 7.10 lists the percentage of cases that converged within 1 iteration. These cases are deemed by our system to be good enough after the first iteration, as according to the stopping criteria (see 5.4.5), and therefore do not require any iterative improvements. As a result, these cases are not considered when evaluating the effectiveness of the iterative convergence.

Platform	%
Naturkompaniet	53%
Bygghemma	71%

Table 7.10: percentage of cases that converged within 1 iteration using the hyperparameters detailed in Section 7.2 & 7.3.

Trend Analysis: When analysing the cases that needed more than one iteration to converge, several patterns emerged. The metrics that were based on the human-curated product recommendation lists showed no significant improvement over the iterations. However, the average rating metric consistently increased from the first to the final iteration. The final iteration refers to the last iteration before termination. This means that in that iteration, either the stopping criteria were met or the maximum number of iterations were reached. The results are presented in Table 7.11.

The upward trend in average ratings supports our hypothesis that feedback loops improve recommendation quality by discarding irrelevant candidates and reinforcing relevant ones. However, the stability of the other metrics requires further investigation.

Metric	Improvement Naturkompaniet	Improvement Bygghemma
Average Rating	0.1223	0.0552
Precision Distance	0.0001	0.0014
Recall Distance	-0.0081	-0.0003
F1 Score	0.0127	0.0089
Category F1 Score	0.0019	0.0129

Table 7.11: Average improvement from first to last iteration for different metrics.

Figures 7.1, 7.2, 7.3, and 7.4 illustrate how key evaluation metrics evolve throughout the refinement process. Each line in the plots groups the cases that concluded after the same total number of iterations. For instance, the line with five data points represents cases that stopped after five iterations, with each point showing the average score at that stage. The label " $N = \dots$ " next to each iteration count indicates how many cases terminated after that number of iterations. This structure allows us to observe how performance trends vary depending on how many refinement steps were completed. See Section 8.1 for interpretation of these results.

Figures 7.6 & 7.5 show an average rating regression trendline for Naturkompaniet and Bygghemma.

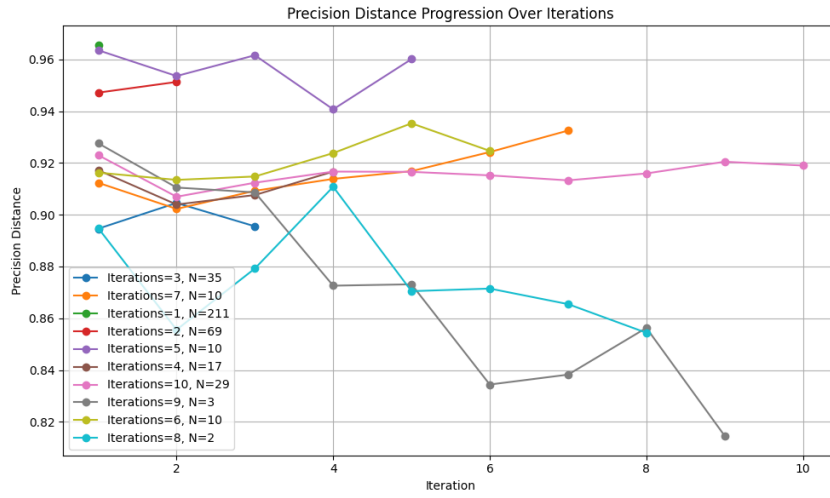


Figure 7.1: Precision distance progression over iterations for Naturkompaniet.

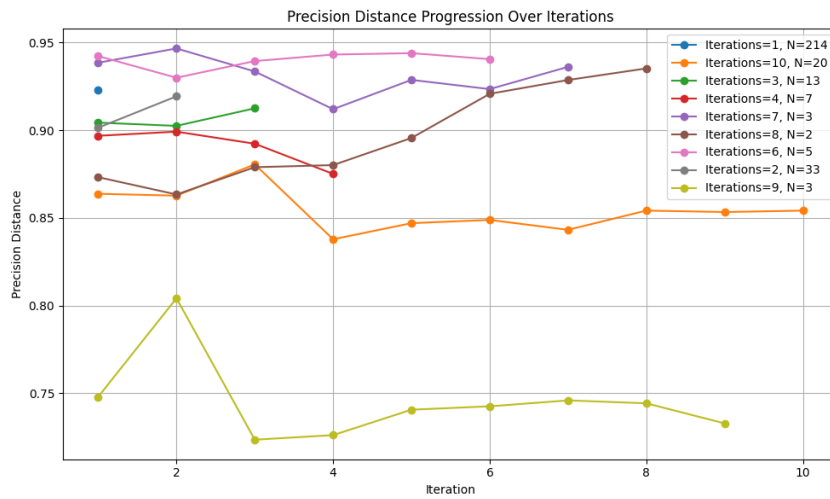


Figure 7.2: Precision distance progression over iterations for Bygghemma.

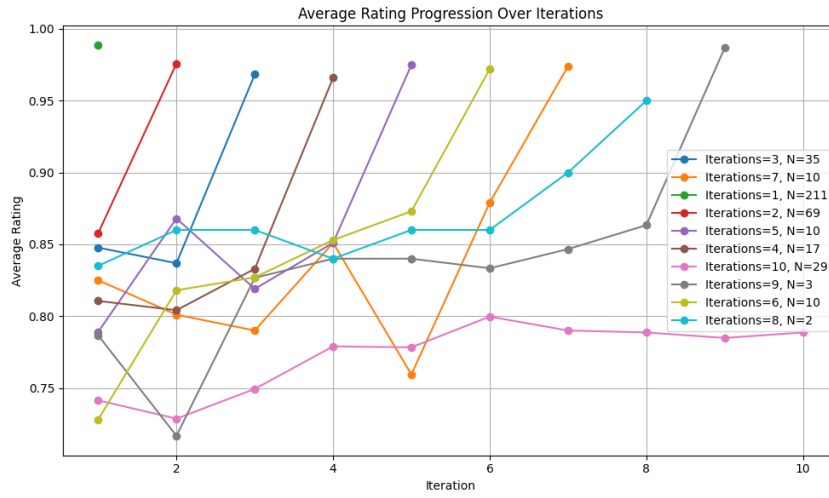


Figure 7.3: Average rating progression over iterations for Naturkompaniet.

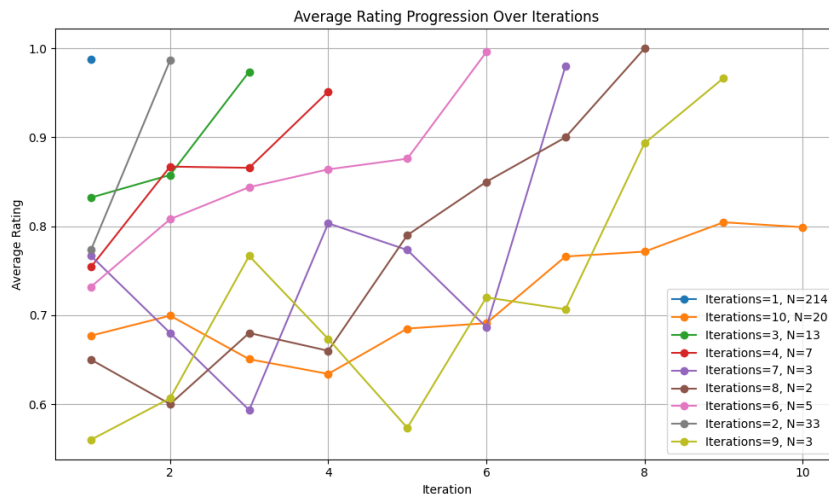


Figure 7.4: Average rating progression over iterations for Bygghemma.

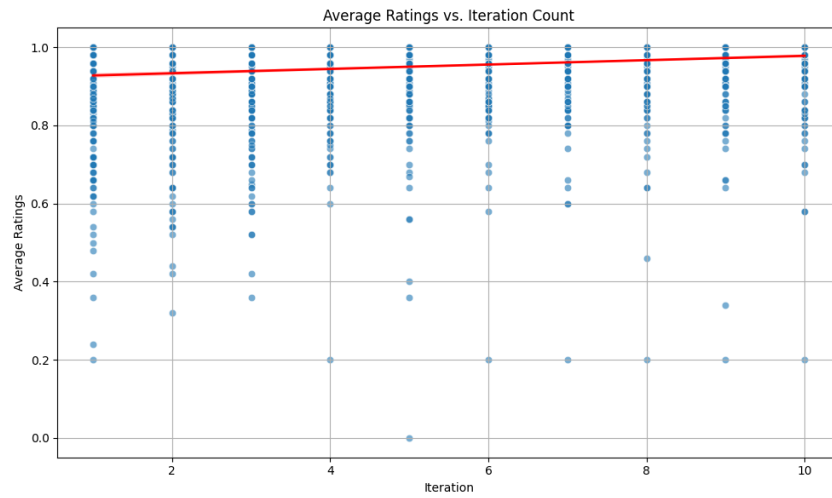


Figure 7.5: Regression trendline over the average ratings for Naturkompaniet.

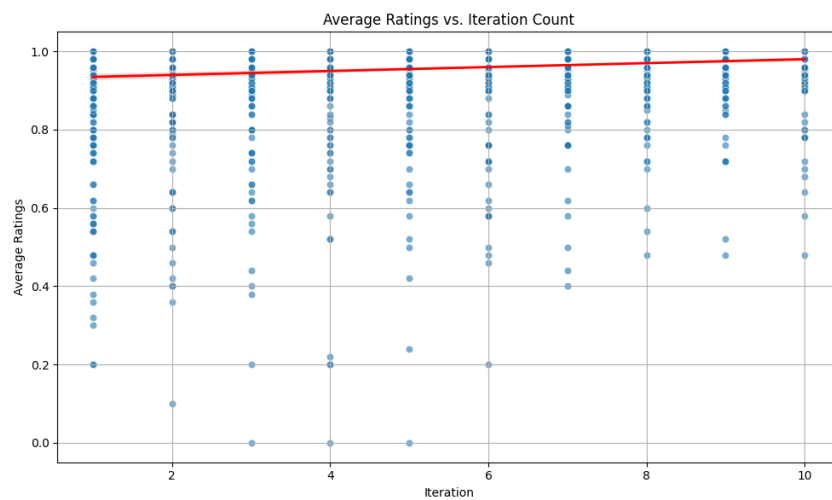


Figure 7.6: Regression trendline over the average ratings for Bygghemma.

7.5 User Study – Interleaving Test Results

To complement our offline evaluation metrics and better capture real-world user preferences, we performed a team-draft interleaving user study comparing the closed-loop RAG system against a non-iterative baseline and the product recommendations curated by the website. Participants for the study were mainly friends and family.

Metric	RAG vs Baseline	RAG vs Website
Δ	0.441	-0.082
z-score	2.256	-0.176
p-value	0.0241	0.8600

Table 7.12: Team draft interleaving results of the three systems *baseline*, *closed-loop RAG*, and *website*.

Table 7.12 presents the results from this test. Recall that we defined a *session* as a user-facing page (i.e., one *url* and its interleaved products). The mean per-session advantage for *RAG vs baseline* of $\Delta = 0.441$ indicates that, on average, the RAG system obtained roughly half a click (or *positive interactions*) more than the baseline system in each session. A two-tailed z-test yielded $z = 2.256$, corresponding to $p = 0.0241$. Hence, at the conventional $\alpha = 0.05$ level we can reject the null hypothesis that the two ranking systems are equally effective; our RAG variant exhibits a statistically significant advantage over the baseline.

Similarly, *RAG vs website* yielded $\Delta = -0.082$ with a corresponding p-value of $p = 0.8600$, meaning that although the website curated products received more clicks as compared to the RAG system, the advantage is **not** statistically significant.

These results validate the core premise of the closed-loop architecture: that incorporating feedback and refining queries across iterations yields recommendations better aligned with user expectations. Furthermore, the interleaving test offers an important counterbalance to offline metrics by incorporating subjective human judgments, which are particularly relevant in the context-heavy domain of inspirational e-commerce content.

Chapter 8

Discussion

This chapter critically reflects on the key outcomes of our work, assessing how well the developed system addresses the original problem constructs and contributes to the broader research landscape. Throughout the thesis, we have adhered to the principles of the Design Science Research (DSR) paradigm, iteratively building and refining an artifact – a closed-loop RAG system – aimed at improving the contextual alignment and relevance of content-based recommendations.

In line with the DSR process, we evaluate the artifact across both quantitative and qualitative metrics: offline metrics, iterative convergence behaviour, and user-based assessments. This chapter revisits the main findings, examines the effectiveness of our design constructs, analyses system sensitivities and limitations, and highlights broader implications for both research and practical deployment. Finally, we outline promising directions for future work.

8.1 Interpretation of Results

The overall findings of this thesis is that our closed-loop RAG system has been successful in improving automatic product recommendations. When we consider the offline methods that we used to evaluate the performance of our system, we saw an improvement over the baseline vector search system across every single metric. Furthermore, our interleaving test user study shows that our system recommendations are preferred to the ones made by the baseline vector search model, and that there is no statistically significant difference between our recommendations and the website's. This reinforces our hypothesis that our closed-loop RAG system can be used to improve product recommendations.

Limitations of Human-Curated Benchmarks

Although these results show promise, the findings raise important questions about the continued usefulness of some of our evaluation metrics. Many of the offline metrics assess perfor-

mance by comparing our recommendations to the products manually curated by the websites. This is a reasonable metric to use at first, but it becomes less relevant when our system consistently achieves the same level of relevance as the websites themselves. There is a natural upper bound: if the system suggests equally relevant but different items, it will never achieve a perfect match score. This suggests that such overlap-based metrics may not fully capture true recommendation quality.

Iterative Improvement

A central feature of the closed-loop RAG system is its iterative feedback mechanism, which is designed to refine recommendations over multiple cycles. Ideally, the relevance of the recommended products should improve with each successive pass. Our results support this for the average rating metric, where we observe a clear upward trend across iterations, indicating that the system effectively learns from its previous outputs.

However, this pattern is not reflected in the other offline metrics that rely on overlap with the human-curated product sets. These metrics show little to no improvement across iterations. As discussed earlier, we think this can be attributed to the diminishing relevance of the human-curated benchmarks once a sufficient level of relevance has been achieved. When the system proposes high-quality alternatives that differ from the original selections, improvement in overlap-based metrics becomes both statistically constrained and conceptually less meaningful. In this light, the steady rise in internally assessed relevance (via the rating metric) is a more reliable indicator of iterative improvement.

8.2 Reflection on Design Constructs

In this section, we revisit the design constructs that were set up in Section 5.1 and evaluate their effectiveness in practice. For each construct, we assess whether it achieved its intended purpose and whether the implementation aligned with our original design goals.

DC1 - Closed-Loop RAG Architecture

The closed-loop RAG architecture was developed to address not only the limitations of basic product retrieval using vector-search, but also the shortcomings of the single-pass RAG architecture. The goal was to utilize feedback from previous iterations to improve product recommendations dynamically, until a satisfactory recommendation set was achieved.

In practice, the architecture fulfilled its goal. The system's iterative nature enabled it to capture contextual nuances in the input texts and improve recommendation quality over successive iterations.

While this process was not universally necessary – for instance, some cases converged after the first iteration – it did improve performance overall. As such, we consider DC1 a successful design construct that effectively addressed several of the key problem constructs identified in the conceptualization phase.

DC2 - Product Relevance Score

The product relevance score is a central component of our closed-loop RAG system. It underpins key processes such as the reranking, the stopping criteria, and parts of the evaluation framework.

As discussed in Section 6.4, evaluating the success of the relevance score by rating the final recommendations with the same relevance score introduces bias. However, the consistent improvement of all other independent performance metrics - compared to the baseline - leads us to consider the relevance scoring implementation to be successful and DC2 to be valid.

DC3 - Query Rewriter

The query rewriter is the primary way in which the system incorporates feedback across iterations. It is responsible for refining the search query based on the results from previous iterations.

The only other part of the program that takes user feedback from prior iterations into account is the Maximal Marginal Relevance (MMR) construct. That system penalizes candidates that are embedding-wise similar to previous low-rated products. However, we can see from the results that even with $lam = 1$ - which means that negative products have no effect - the performance is still improving. That means that our query rewriter alone is improving performance and that DC3 can be considered impactful.

DC4 - Stopping Criteria

The stopping criteria for our RAG system consisted of measuring whether all products in the recommended product set received a relevance score that was above a certain threshold (0.9 for both Naturkompaniet and Bygghemma). There was also an additional safeguard that prevented the system from doing more than a preset number of iterations to prevent unnecessary computational overhead.

A significant percentage of cases (53% for Naturkompaniet and 71% for Bygghemma) terminated after a single iteration. This indicates that the stopping criteria were not overly conservative. In more complex cases, where our system determined that more than one iteration was necessary, we can see that, at least on average, the performance improved. This shows that the stopping criteria were not too permissive.

Overall, these outcomes demonstrate that the stopping criteria worked as intended, and at least in most cases, this led to an improvement. We therefore consider DC4 to be a successful design construct.

DC5 - Data Preprocessing and Embeddings

Preprocessing and embedding strategies were crucial for ensuring meaningful semantic similarity during retrieval. Our initial use of unmodified product descriptions proved insufficient, as many of these were overly marketing-focused or lacked detail. Through LLM-guided rewriting, we restructured product descriptions to emphasize core attributes and use cases, which significantly improved embedding quality.

The addition of hierarchical category embeddings further enhanced retrieval precision, especially in large and semantically overlapping product catalogues as is the case with Bygghemma. Empirical findings confirmed that fully preprocessed and hierarchically embedded representations consistently outperformed simpler baselines across key evaluation metrics.

DC6 - Evaluation Instrumentation

The inclusion of multiple evaluation methods – including embedding distance metrics, overlap with human curations, and user-based interleaving – enabled a robust validation of the system.

Importantly, these evaluations were not only post-hoc, but also informed iterative refinement loops. For example, convergence tracking across iterations helped us confirm the utility of the closed-loop mechanism itself. In that sense, DC6 served both evaluative and generative role within the broader DSR cycle.

8.3 Hyperparameter Sensitivity

A central component in refining the closed-loop RAG system was identifying and tuning hyperparameters that influence performance across both retrieval and iteration behaviour. The grid search we performed in Section 7.2 revealed that the system is sensitive to several parameters – particularly those governing embedding structure – while others exhibited less impact.

Embedding configuration (emb) was shown to be one of the most influential parameters. Across all sites, configurations that used fully preprocessed product descriptions consistently outperformed their minimally preprocessed counterparts. Similarly, hierarchical embeddings (which combined product-description embeddings with category embeddings) produces better results in all product spaces. These findings confirm that representation quality – both in content and structure – plays a central role in retrieval accuracy.

The influence on λ (lam) which govern the weight of negative feedback in subsequent iterations, was also substantial. Lower values of λ (i.e., heavier penalization of poorly rated products) tended to improve convergence and final recommendation quality. This suggests that the system benefited from actively learning to avoid certain types of products – reinforcing the value of incorporating prior iteration history into reranking logic.

Model temperature (temp) had a relatively minor impact. Although small differences were observed across tasks, there was no consistent pattern linking temperature to any metric. As such, we settled on a mid-range value (0.8) for most configurations without observing degradation in output quality.

Stopping criteria (sc) and maximum iteration count (mi) governed the loop dynamics. While a high stopping threshold ensured that only highly relevant recommendations were accepted, it also increased the number iterations needed for convergence – while increasing the number of cases where convergence was never reached. We found that a

threshold of 0.9 balanced convergence speed with recommendation quality effectively across sites. Likewise, a higher maximum iteration value prevented premature termination in more difficult cases while keeping the computation bounded.

Overall, while some hyperparameters (like temperature) showed little sensitivity, others – especially embedding configuration, feedback penalization, and stopping criteria – significantly shaped system performance. These findings underscore the importance of site-specific tuning in RAG systems and suggest that generalization across domains may require adaptive or learned hyperparameter strategies.

8.4 Threats to Validity

As with any empirical study, our work is subject to several *Threats to Validity* (TTV) factors that may influence the interpretation and generalizability of the results. Recent studies have pointed out that TTVs in software engineering are often treated as an afterthought [26]. To avoid this pitfall and ensure a more rigorous analysis, we adopt the taxonomy proposed by Lago et al. [27], which categorizes threats into internal, construct, external, and conclusion validity. For each category, we describe the threat and suggest possible mitigation strategies.

Internal Validity

Internal validity describes how confidently causal conclusions can be attributed to the study [27]. Because the closed-loop system contains several interacting modules, there is a possibility that observed improvements arise from undocumented implementation choices rather than the closed-loop logic itself. Moreover, using LLM-generated product rewrites introduces an uncontrolled source of variance, which can make it difficult to determine whether changes in final quality are caused by the retrieval strategy or the rewrites themselves. Conducting careful ablation studies would help isolate these effects and improve transparency.

Construct Validity

Construct validity refers to the extent to which the metrics accurately capture the theoretical concepts they are intended to measure [27]. A primary threat to construct validity in our study arises from evaluation bias. Specifically, the same relevance-scoring mechanism is used both to guide the iterative RAG loop and to assess its final output. This dual role risks inflating perceived performance gains, as the system is effectively optimized against its own evaluation function. In addition, both prompt tuning and embedding configurations were optimized on the same dataset used for evaluation. While this improves performance within the current domain, it may lead to an overly narrow operationalization of the concept of “relevance,” limiting the generality of our construct beyond the evaluation corpus. To enhance construct validity in future work, we recommend decoupling the optimization and evaluation pipelines, and incorporating triangulation through human judgment or alternative external benchmarks.

External Validity

External validity addresses the generalizability of the study’s findings beyond the specific context in which the research was conducted [27]. Our system was evaluated using data from a limited number of Swedish e-commerce platforms, which may not represent the diversity of platforms or user behaviours in other markets. The prompts used also rely on Swedish-centric linguistic cues, further limiting transfer. Extending the evaluation to multilingual catalogues or to non-retail recommendation tasks (e.g., news or academic papers) is required before broader claims can be made.

Conclusion (statistical) Validity

Conclusion validity relates to the reliability and significance of the study’s conclusions [27]. The statistical analyses employed in our evaluation may be susceptible to Type I or Type II errors, i.e., a false positive or negative, especially given the sample size and potential measurement noise. Moreover, the iterative nature of our system introduces complexities in interpreting performance metrics over time. To strengthen conclusion validity, future studies should employ robust statistical methods, ensure adequate sample sizes, and consider longitudinal analyses to capture the system’s performance dynamics accurately.

Summary of Key Limitations

To conclude this section, we summarize the most practically impactful limitations identified across the four validity categories above. These are presented here as a quick reference for readers and to highlight areas of concern for future work.

- **Evaluation Bias:** The same relevance scoring mechanism was used both for guiding the RAG loop and evaluating performance, risking positive bias.
- **Prompt Bias:** Prompts were optimized for the evaluated datasets, their effectiveness in new domains or languages remain untested.
- **Dataset Bias:** Data was collected from a small set of Swedish e-commerce sites, which may limit generalizability to other markets, languages, or content types.
- **Subjectivity in Ground Truth:** Human-curated product recommendations are not exhaustive nor objective – making offline metrics potentially under-represent true relevance.
- **Dependence on Embeddings:** System relies heavily on high-quality embeddings. Performance may degrade quickly with noisy, sparse, or less semantically rich data.
- **User Study Scale:** For statistically significant results the scale of the user study would need to be expanded.

8.5 Implications for Future Research and Practice

This work has both theoretical and practical implications for the development of automatic recommendation systems, not only in the immediate e-commerce setting in which our closed-loop RAG system was evaluated. Because the architecture separates retrieval, reranking, and relevance-scoring into loosely coupled components, the same framework can be applied to other domains without requiring a complete overhaul. The closed-loop refinement mechanism, in particular, shows promise for applications that are context-heavy and where user intent evolves over time. By iteratively adjusting retrieval based on context and feedback, the system is able to better align recommendations with content, which may translate to increased user engagement and conversion rates in domains such as news recommendation, education, or healthcare.

Our results also highlight the potential of using LLMs not only for query reformulation, but also as agents for data preprocessing. The use of prompt-guided rewriting to enhance product descriptions demonstrated measurable improvements in retrieval quality. This suggests that similar LLM-assisted preprocessing strategies could be employed in other domains to improve semantic representations, especially when dealing with sparse, noisy, or inconsistently structured data.

From a **research perspective**, this work contributes to the developing literature on the use of Retrieval-Augmented Generation beyond the scope of "simple" question and answering. By framing RAG as a feedback-driven recommendation engine, we extend its applicability and present a use case that demands iterative, context-sensitive refinement. This opens the door to a broader set of research questions regarding how RAG systems can be optimized for other tasks such as legal precedent retrieval or multimedia recommendations.

Additionally, our mixed-method evaluation strategy – combining offline benchmarks, iterative performance tracking, and user-based interleaving tests – illustrates a balanced framework for judging future RAG variants in both quantitative and qualitative dimensions.

In summary, the architecture, techniques, and evaluation methodologies developed in this work offer a foundation for both academic exploration and practical innovation. As LLMs continue to evolve, integrating them into interactive, retrieval-centric systems will likely remain a rich and impactful area of future research.

8.6 Further Work

While the closed-loop RAG system developed in this thesis shows promising results, several avenues remain open for future exploration and improvement.

First, one important direction is the refinement of prompt engineering through dynamic or automated approaches. Although our system relied on manually designed prompts that were iteratively refined, future work could extend this scope to investigate methods such as reinforcement learning to adapt prompts automatically based on e.g., performance feedback. Such approaches could make the system more adaptive to different domains and less reliant on manual trial-and-error.

A second promising area is the development of adaptive stopping criteria. In this thesis,

we used fixed thresholds to determine when the iterative loop should terminate. However, more flexible alternatives – such as confidence-based or entropy-based thresholds – could improve convergence decisions and reduce unnecessary computation without sacrificing relevance.

Third, enriching vector space with *multimodal embeddings* – e.g., incorporating product images in the embedding space – to aid in cases where product information is sparse could help improve retrieval, especially in visually-driven domains such as fashion. Recent cross-modal encoders make it feasible to index visual and textual cues side-by-side, allowing the reranker to preference items whose look and feel match the inspirational narrative.

In addition, future work may extend the objective beyond contextual relevance. The system could be modified to encompass various business metrics such as profit margins, sustainability, or stock availability. Re-framing the recommendations to a multi-objective optimization task would allow the system to balance commercial goals with user satisfaction.

Finally, integrating real-time implicit user feedback presents an exciting opportunity. Metrics such as click-through rates, dwell time, or scroll behaviour could provide dynamic signals to inform retrieval and reranking during live deployments. Incorporating such feedback would help the system adapt over time, personalizing recommendations based on actual user behaviour rather than relying solely on static content.

Together, these directions offer meaningful opportunities to enhance the flexibility, performance, and applicability of closed-loop RAG systems. Pursuing them could lead to more intelligent, context-aware, and commercially viable recommendation engines across domains.

Chapter 9

Conclusion

This thesis set out to design, implement, and evaluate a closed-loop Retrieval-Augmented Generation (RAG) system for content-based product recommendations in e-commerce. Through iterative refinement, the system aimed to improve recommendation quality by adapting to the contextual nuances of inspirational content.

Principal Findings

The central finding of this thesis is that the proposed closed-loop Retrieval-Augmented Generation (RAG) system outperforms the non-iterative baseline across all evaluated metrics. Through iterative refinement, the system was able to generate product recommendations that were better aligned with the contextual intent of inspirational content.

Across offline evaluations and user studies, the RAG system consistently demonstrated:

- **Offline Evaluation:** Across both Naturkompaniet and Bygghemma, the closed-loop RAG system achieved higher precision-/recall distances, F1-scores, and ratings as compared to the non-iterative baseline. For instance, the average relevance rating for Naturkompaniet increased from 0.79 to 0.96 when using the closed-loop system. Similar patterns were observed across other sites, indicating robustness across diverse product and content spaces.
- **Iterative Improvement:** When examining cases that required multiple iterations, we observed a consistent upward trend or stability across all evaluated metrics. None of the metrics deteriorated over successive iterations – each either improved or exhibited no notable change, see Section 8.1. Notably, all platforms saw a notable increase in the average product rating by the final iteration. This result validates the underlying premise of the closed-loop approach: that incorporating feedback and refining queries leads to more contextually relevant recommendations.

- **User Study:** The interleaving test further confirmed the system’s effectiveness in real-world scenarios. With a mean user preference advantage of $\Delta = 0.441$ and a statistically significant p-value of ($p = 0.0241$), participants showed a clear preference for products recommended by the closed-loop system over those from the non-iterative baseline. Confirming that the system delivers value even in subjective, real-world contexts.

Together, these results validate the technological rule proposed in Chapter 5: *"to achieve improved product recommendation accuracy and relevance in inspirational content, apply a closed-loop RAG approach that iteratively refines retrieval..* I.e., product recommendations demonstrably benefit from iterative refinement when applied to nuanced, context-rich content. This closed-loop design enables the system to course-correct based on prior retrievals, leading to higher precision and thus ultimately produces recommendations that better align with user expectations.

References

- [1] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, and H. Wang, *Retrieval-augmented generation for large language models: A survey*, 2024. arXiv: 2312 . 10997 [cs.CL]. [Online]. Available: <https://arxiv.org/abs/2312.10997>.
- [2] E. Engström, M.-A. Storey, P. Runeson, M. Höst, and M. T. Baldassarre, “How software engineering research aligns with design science: A review,” *Empirical Software Engineering*, vol. 25, no. 4, pp. 2630–2660, Jul. 2020, ISSN: 1573-7616. DOI: 10 . 1007/s10664-020-09818-7.
- [3] P. Runeson, E. Engström, and M.-A. Storey, “The design science paradigm as a frame for empirical software engineering,” in *Contemporary Empirical Methods in Software Engineering*, M. Felderer and G. H. Travassos, Eds. Cham: Springer International Publishing, 2020, pp. 127–147, ISBN: 978-3-030-32489-6. DOI: 10 . 1007/978-3-030-32489-6_5.
- [4] T. Mikolov, K. Chen, G. Corrado, and J. Dean, *Efficient estimation of word representations in vector space*, 2013. arXiv: 1301 . 3781 [cs.CL]. [Online]. Available: <https://arxiv.org/abs/1301.3781>.
- [5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran, and T. Solorio, Eds., Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. DOI: 10 . 18653/v1/N19-1423. [Online]. Available: <https://aclanthology.org/N19-1423/>.
- [6] N. Reimers and I. Gurevych, “Sentence-BERT: Sentence embeddings using Siamese BERT-networks,” in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, K. Inui, J. Jiang, V. Ng, and X. Wan, Eds., Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 3982–3992. DOI: 10 . 18653/v1/D19-1410.

- [7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30, Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.
- [8] T. S. I. S. Alec Radford Karthik Narasimhan, "Improving language understanding by generative pre-training," 2018. [Online]. Available: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf.
- [9] H. Zhang, Y. Zhang, X. Li, W. Shi, H. Xu, H. Liu, Y. Wang, L. Shang, Q. Liu, Y. Liu, *et al.*, "Evaluating the external and parametric knowledge fusion of large language models," *arXiv preprint arXiv:2405.19010*, 2024.
- [10] L. Boonstra, *Prompt engineering*, 2024. [Online]. Available: <https://www.onlinebrandambassadors.com/stable/Google-Prompt-Engineering-Boonstra.pdf>.
- [11] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive nlp tasks," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., 2020, pp. 9459–9474. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf.
- [12] Z. Shao, Y. Gong, Y. Shen, M. Huang, N. Duan, and W. Chen, "Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, H. Bouamor, J. Pino, and K. Bali, Eds., Singapore: Association for Computational Linguistics, Dec. 2023, pp. 9248–9274. DOI: 10.18653/v1/2023.findings-emnlp.620.
- [13] Y. Zhou, A. I. Muresanu, Z. Han, K. Paster, S. Pitis, H. Chan, and J. Ba, "Large language models are human-level prompt engineers," in *The Eleventh International Conference on Learning Representations*, 2022. [Online]. Available: <http://arxiv.org/abs/2211.01910>.
- [14] R. Pryzant, D. Iter, J. Li, Y. T. Lee, C. Zhu, and M. Zeng, "Automatic prompt optimization with" gradient descent" and beam search," *arXiv preprint arXiv:2305.03495*, 2023. [Online]. Available: <https://doi.org/10.48550/arXiv.2305.03495>.
- [15] Q. Ye, M. Axmed, R. Pryzant, and F. Khani, "Prompt engineering a prompt engineer," *arXiv preprint arXiv:2311.05661*, 2023. [Online]. Available: [https://aclanthology.org/2024.findings-acl.21/..](https://aclanthology.org/2024.findings-acl.21/)
- [16] B. Bischl, M. Binder, M. Lang, T. Pielok, J. Richter, S. Coors, J. Thomas, T. Ullmann, M. Becker, A.-L. Boulesteix, *et al.*, "Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 13, no. 2, e1484, 2023.

-
- [17] E. Knauss, “Constructive master’s thesis work in industry: Guidelines for applying design science research,” in *Proceedings of the 43rd International Conference on Software Engineering: Joint Track on Software Engineering Education and Training*, ser. ICSE-JSEET ’21, Virtual Event, Spain: IEEE Press, 2021, 110–121, ISBN: 9780738133201. DOI: 10.1109/ICSE-SEET52601.2021.00021.
 - [18] P. Offermann, O. Levina, M. Schönherr, and U. Bub, “Outline of a design science research process,” Jan. 2009. DOI: 10.1145/1555619.1555629.
 - [19] A. Chatterjee, H. S. V. N. S. K. Renduchintala, S. Bhatia, and T. Chakraborty, *Posix: A prompt sensitivity index for large language models*, 2024. arXiv: 2410.02185 [cs.CL]. [Online]. Available: <https://arxiv.org/abs/2410.02185>.
 - [20] F. Rekathati. “Introducing a Swedish Sentence Transformer.” (Aug. 23, 2021), [Online]. Available: <https://kb-labb.github.io/posts/2021-08-23-a-swedish-sentence-transformer/>.
 - [21] J. Carbonell and J. Goldstein, “The use of mmr, diversity-based reranking for reordering documents and producing summaries,” in *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, 1998, pp. 335–336.
 - [22] D. M. W. Powers, *Evaluation: From precision, recall and f-measure to roc, informedness, markedness and correlation*, 2020. [Online]. Available: <https://arxiv.org/abs/2010.16061>.
 - [23] F. Radlinski, M. Kurup, and T. Joachims, “How does clickthrough data reflect retrieval quality?” In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, ser. CIKM ’08, Napa Valley, California, USA: Association for Computing Machinery, 2008, 43–52, ISBN: 9781595939913. DOI: 10.1145/1458082.1458092.
 - [24] E. Kharitonov, C. Macdonald, P. Serdyukov, and I. Ounis, “Generalized team draft interleaving,” in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, ser. CIKM ’15, Melbourne, Australia: Association for Computing Machinery, 2015, 773–782, ISBN: 9781450337946. DOI: 10.1145/2806416.2806477.
 - [25] S. Holm, “A simple sequentially rejective multiple test procedure,” *Scandinavian Journal of Statistics*, vol. 6, no. 2, pp. 65–70, 1979, ISSN: 03036898, 14679469. [Online]. Available: <http://www.jstor.org/stable/4615733> (visited on 05/16/2025).
 - [26] R. Verdecchia, E. Engström, P. Lago, P. Runeson, and Q. Song, “Threats to validity in software engineering research: A critical reflection,” *Information and Software Technology*, vol. 164, p. 107329, 2023, ISSN: 0950-5849. DOI: 10.1016/j.infsof.2023.107329.
 - [27] P. Lago, P. Runeson, Q. Song, and R. Verdecchia, “Threats to validity in software engineering – hypocritical paper section or essential analysis?” In *18th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, M. Deneva and S. Martinez-Fernandez, Eds., ser. ESEM ’24, Barcelona, Spain: ACM, 2024, pp. 314–324. DOI: 10.1145/3674805.3686691.
-

Appendices

Appendix A

A.1 Author Contribution Statement

The authors, Eric Martell & Axel Prellner, contributed equally to the completion of this thesis. While our primary focus areas occasionally differed – e.g., Eric focused on the closed-loop architecture while Axel delved deep into prompt engineering – we collaborated closely throughout the project. We continually helped each other in both the practical- and academic work. All parts of this project were discussed, reviewed, and approved by both authors.

EXAMENSARBETE Closed Loop Retrieval-Augmented Generation (RAG) for Content-based

Recommendations in E-commerce

STUDENTER Eric Martell, Axel Prellner**HANDLEDARE** Per Runeson (LTH), Rasmus Ros (Theca Systems AB)**EXAMINATOR** Emelie Engström (LTH)

Smartare produktrekommendationer med hjälp av generativ AI

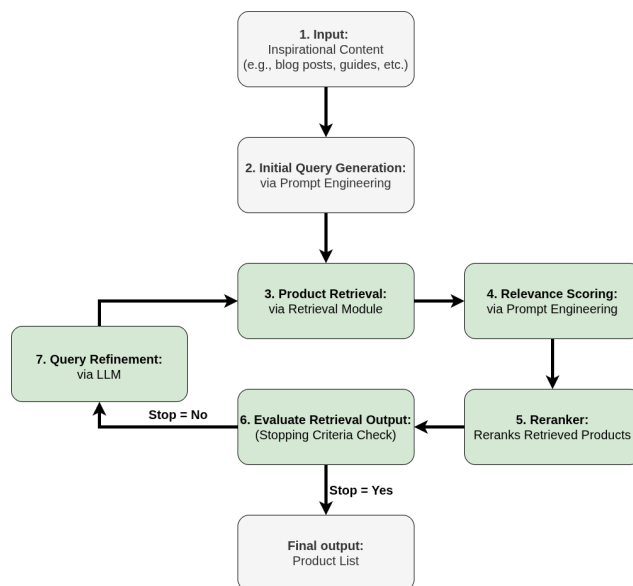
POPULÄRVETENSKAPLIG SAMMANFATTNING **Eric Martell, Axel Prellner**

Kan man använda ChatGPT för att göra en bra produktrekommendation? Det har vi undersökt i vårt examensarbete, där vi har utvecklat ett system som ska ge träffsäkra produktförslag genom att läsa och förstå texter såsom blogginlägg eller guider.

Att rekommendera relevanta produkter till en text på internet kräver vanligtvis både ämneskunskap och en djup förståelse för innehållet. Automatiska system misslyckas ofta med det senare, vilket gör att det ofta krävs en manuell process för att få passande produktförslag. Detta kräver dock mycket resurser och därför är intresset för ett automatiskt system - som faktiskt fungerar - stort.

För att lösa detta problem har vi utvecklat ett system som använder sig av språkförmågan hos så kallade *Large Language Models* (LLMs), såsom ChatGPT, för att förstå den underliggande meningen i en given text. Där ett traditionellt system bara skulle undersöka den semantiska likheten mellan texten och produkterna kan vårt system istället analysera innehållet från ett marknadsföringsperspektiv och utifrån det generera relevanta produktrekommendationer. Det kan sedan iterativt förbättra sina förslag genom att använda sig av information från tidigare rekommenderade produkter. Sättet detta fungerar på kan illustreras med en text som handlar om fjällvandring. Ett traditionellt system rekommenderar ofta fjällvandringsguider eller böcker, medan vårt system istället förstår att läsaren troligen är ute efter utrustning till vandringen.

Figuren visar hur vårt system fungerar i praktiken. Först läser systemet in en text, därefter använder den sin språkförståelse för att lista ut vilka typer av produkter som kan vara relevanta och söker efter dessa i en databas. Systemet kan förbättra sina förslag genom att analysera tidigare rekommendationer och justera sökningarna tills resultatet bedöms vara bra nog.



Systemet testades på data från olika svenska nätbutiker och visade tydliga förbättringar jämfört med traditionella, automatiska, rekommendationssystem. Detta kunde ses genom att våra rekommendationer var mer lika de manuellt rekommenderade produkterna som används på nätbutikernas officiella hemsidor. Det kunde också bekräftas med en studie där användare konsekvent föredrog vårt system.

Detta arbete visar på att vår användning av LLMs kan förbättra produktrekommendationer. Utöver det ser vi även att det kan användas inom andra områden där kontextuellt innehåll ska matchas med relevanta förslag, som till exempel nyheter eller videorekommendationer.