Creating a Retrieval-Augmented Generation Pipeline for the Guidelines of the Dutch College of General Practitioners

Leander Bindt[1]
Supervisor(s): Dr.ir. J. Yang[1]**, Yannick ter Heerdt**
[1]EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 21, 2026

Name of the student: Leander Bindt
Final project course: CSE3000 Research Project
Thesis committee: Dr.ir. J. Yang, Yannick ter Heerdt, Dr. P.K. Murukannaiah

An electronic version of this thesis is available at http://repository.tudelft.nl/.

Abstract

As general practitioners currently experience high workloads, Large Language Models (LLMs) offer a promising opportunity to relieve some of this work by enabling faster searching of medical guidelines, saving doctors time and allowing them to deliver better care. This research aimed to answer the primary research question: How can a Retrieval-Augmented Generation (RAG) pipeline be constructed for Dutch NHG guidelines? By breaking this problem down into four distinct sub-questions focused on data processing, retrieval optimization, storage scalability, and model grounding, the research successfully demonstrates a complete, factually correct, and scalable system for general practitioners in The Netherlands. Specifically, the findings show that context-aware data splitting with minimized block sizes optimally preserves clinical cohesion while keeping costs low. For retrieval optimization, combining a traditional BM25 keyword search with an AI meaning-based vector search via Reciprocal Rank Fusion captures edge-case guidelines more effectively than either method alone. Storage scalability is achieved by pairing a Hierarchical Navigable Small World graph with memory-mapped storage, allowing the system to offload data to the disk while maintaining high throughput and low latency. Finally, the application of prompt instructions successfully enforces grounded refusal, preventing the AI from falling back on internal training data when valid clinical context is missing.

1 Introduction

General practitioners currently experience high workloads. A recent study shows that caring for an average-sized patient population requires over 25 hours per day, even when working in a team where tasks are delegated, it requires over 9 hours, resulting in an undesirably high workload [6]. Large Language Models (LLMs) offer a promising opportunity to relieve some of this work by enabling faster searching of medical guidelines, saving doctors time and allowing them to deliver better care [10]. However, integrating this technology introduces a critical problem: namely, how to ensure that these (often) American or Chinese-trained LLMs return valid advice and factually correct data based on the Dutch General Practitioners Group (NHG) guidelines and do not hallucinate wrong information. Giving a patient incorrect medication or a diagnosis based on false information can be dangerous.

A solution could be grounding the LLM with a Retrieval-Augmented Generation- (RAG-) pipeline [12]. While such RAG-pipelines do exist, the NHG guidelines represent a uniquely complex data type. There are three distinct structural hurdles: highly dense medical terminology, deeply nested sub-headers, and the often the necessary information is fragmented across multiple guidelines. Currently available pipelines are unequipped to resolve these multi-document clinical dependencies [3] and thus more research is required. The main research question is defined as: *How can a RAG pipeline be constructed for Dutch NHG-guidelines?*.

To make the project feasible within the available time it is broken up into smaller, more manageable pieces, translating into four sub-questions. First, the research explores how the choice of chunking algorithm (for example: fixed-size vs context-aware) affects the system's ability to retrieve complete and unbroken clinical context from the NHG guidelines. Second, it investigates how much combining a traditional keyword search with an AI meaning-based (vector) search improves the pipeline's ability to find the right guidelines when the prompt has typos, abbreviations and/or medicine-specific terms compared to just using either search alone. Third, splitting the guidelines into smaller chunks and embedding them results in a large database. Because storing these as static JSON files or executing iterative queries over

a standard relational database lacks scalability, the research evaluates better alternatives for storing and retrieving this data. Finally, the research determines how much giving the LLM strict rules and some good examples reduces the number of hallucinations, compared to just asking the same question with no rules and no examples.

When evaluating the overall goodness of a system, there are many possible quality dimensions, such as: query latency, (financial) token consumption, long-term database maintainability, and user-interface experiences. However, because the system is fundamentally unusable if it is too slow to use during a standard ten-minute consultation or invents false clinical claims, this research focusses on clinical factuality and storage scalability. So answering the sub-questions results in a factually correct and scalable AI system for general practitioners based in The Netherlands.

Chapter 2 Related Work explores the already available solutions and past research. Understanding, learning and combining these papers enables this research to push beyond what is currently possible. It describes the existing issues and research gaps. Chapter 3 Methodology explains the different experiments needed to solve these issues, fill in the gaps and answer the research questions. The results and possible limitations are discussed in Chapter 4 Experimental Setup and Results. Chapter 5 Responsible Research presents different topics and discussions on how this research aligns with the moral values of society. The conclusions and description of possible future works are proposed in Chapter 6 Conclusions and Future Work.

2 Related Work

Recent work with LLMs has demonstrated the use case for these models in the medical field. A good example of this, is the work on Med-PaLM 2 by Google which achieves state-of-the-art performance on various medical exams, with a high score of 86.5% on the United States Medical Licensing Examination (USMLE)-style questions [10]. This score is not achieved by a general purpose LLM but rather a model (PaLM 2 [1]) that went through targeted instruction fine-tuning using a dense mixture of domain-specific medical datasets.

Fine tuning such a base model is not a small update. For example: the instruction fine-tuning data required over 182,000 examples from the MedMCQA dataset and over 10,000 examples from the MedQA dataset. Updating a model on this scale relies on a big computing infrastructure, which Google has access to, but which is infeasible for this project. And even if the computer power was available, sourcing such a large dataset of Dutch question and answer examples would be out of the scope of this project.

2.1 RAG pipeline

To overcome the limitations of base models but without relying on large amounts of computing power and examples a different method is needed. Recent research has pivoted towards retrieval augmented generation (RAG) systems. An example of this is Almanac [12], a framework designed to safely use LLMs in a clinical decision making setting by adding external retrieval tools to off the shelf models. So rather than using the internal parameters to generate answers, Almanac utilizes a text encoder combined with a high performance vector database to retrieve relevant context chunks from predefined medical repositories, a web browser and clinical calculator. The model is then strictly used for its language understanding, using the retrieved context blocks for generating a factually correct response accompanied by the proper citations clinical verification.

Because standard multiple choice benchmarks often fail to representatively score a model on the complexities of actual medical environments, Almanac is evaluated using ClinicalQA, a dataset consisting of 130 open clinical scenarios for five different specializations. Checked by a panel of doctors, Almanac demonstrated a significant increase of 18% in factuality compared to the base model. Even reaching 91% factuality for the cardiology specialty.

Next to retrieving information Almanac also uses the RAG system to act as a safeguard against hallucinations. If the retriever cannot find any context blocks for the given prompt it will not try to generate an answer using the internal parameters from its training data.

The structure presented by the Almanac framework provides a robust starting point for this research. Using a RAG pipeline that retrieves context blocks from the NHG guidelines, the system will correctly localize to the Netherlands. It reduces the need for enormous computation power or sourcing large question and answer datasets.

While various RAG systems already exist, it might not be as easy as just dropping in the NHG guidelines. A recent study investigating the implementation of RAG systems of different fields such as biomedical, educational and research domains, highlights that these architectures suffer from limitations in their information retrieval [3]. In their analysis, it is found that RAG systems cannot be treated as plug-and-play mechanisms. Just giving the guidelines to an off-the-shelf pipeline results in multiple failure points: arbitrary chunking destroys the medical context, standard embedding models might fail to map Dutch phrasing, and ungrounded generation allows the model to fill information gaps with its training data. Therefore a new system is needed, custom made for the NHG guidelines.

2.2 Searching for Context Blocks

An important part of the RAG system is the retrieval of the different context blocks. A recent study shows that combining two types of algorithms results in a high quality searching system [2]. They combine a traditional sparse keyword search with a newer dense AI meaning based vector search.

A traditional search algorithm operates as a sparse retriever, finding exact matching word combinations in the prompt and context blocks. While the recent AI advancements have introduced a new type of algorithm which uses a model to map prompts and context blocks to a coordinate space, then these blocks are retrieved based on a distance score from the prompt point. This method of mapping information pieces to the vector space is called embedding.

2.2.1 Vocabulary Mismatches

While these new retrievers provide great improvements in general by understanding meaning and thus bridging vocabulary mismatches like synonyms, they have their limitations. Because this research uses very specific medical terminology and is based in the Dutch language, general purpose embedding models might struggle with certain parts of the guidelines.

2.2.2 Hybrid Approach

Because these sparse and dense retrievers operate on fundamentally different matching principles, they act as mutual complements The research by Arabzadeh investigates the trade offs between the two retrieval systems and demonstrates that using just either one results in missing relevant context blocks that otherwise were easily retrieved. They call this system

a hybrid search, by deploying such a system, it is ensured that the final model is grounded using the most relevant context blocks.

2.3 Best Match 25

The selection of the specific sparse retrieval algorithm is crucial to the system's overall correctness and efficiency. For the exact match keyword search, best match 25 (BM25) is the industry standard [9]. It is an algorithm that is recognized as one of the best text retrieval algorithms. Unlike simpler models that rely on linear term frequency, BM25 introduces a more complex probabilistic model that makes it perform better on complex information like the NHG guidelines. Specifically, it offers four structural mechanisms tailored to clinical extraction:

Non-linear Term Frequency (Saturation) In traditional models, a context block's score scales without a ceiling with the frequency of a query term. For BM25 the difference between the frequency of 50 and 100 is small, while the difference between 1 and 2 is large. This ensures that single, highly repeated clinical terms do not dominate the retrieval scores, allowing the system to find context blocks that balance matches across all terms in a complex prompt.

Pruning via RDF BM25 implements a reverse document frequency (RDF), meaning that widely used terms are removed from the algorithm before the search happens. A good example of this, is the use of articles. The word 'the' will be removed from the prompt 'the medicine'. Making sure only the relevant terms are used for the retrieval.

Document Length Normalization The context blocks will vary in length, some are complex tables about medicine prescription and others are short explanations of terms. It works by comparing the length of a given document against the average document length of the entire collection. This ensures that shorter blocks are not 'forgotten' and never retrieved.

Hierarchical Header Weighting It allows for the extension where titles, or subtitles are deemed more important than the regular text. This mechanism is super useful for the NHG guidelines as most of the text has at least three or four (sub) headers. A match with a context blocks header provides significantly stronger evidence of relevance than an equivalent match buried deep in the body.

By adjusting the weights of these functionalities, BM25 results in a highly tuned retrieval layer that respects the structure of the NHG guidelines.

2.4 Retrieval of Embedded Blocks using HNSW

The integration of a dense, AI meaning based retriever is essential for the hybrid search method defined before. However, the NHG guidelines dataset consist of more than 100 guidelines, each of those resulting in approximately 250 context blocks, and every block gets embedded to a vector space with a dimensionality of 3072. A standard approach would be to calculate some sort of distance score between the prompt and every one of those blocks, and retrieving the K-nearest neighbors. That would result in more than 75 million calculations per prompt.

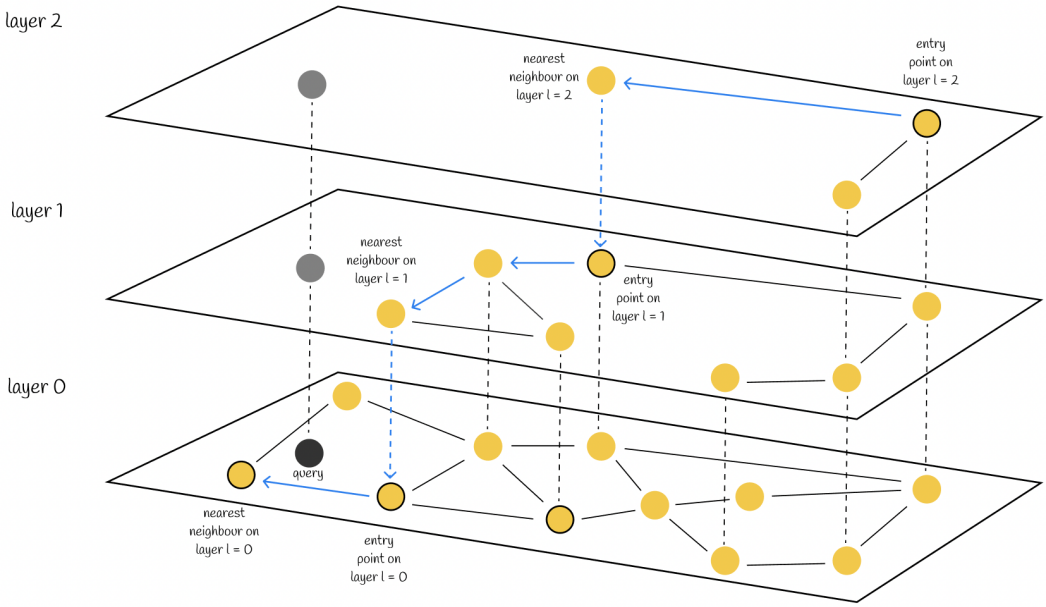

Figure 1: A representation of the ANN algorithm using the HNSW graph [4]

2.4.1 Navigating the Vector Space via HNSW Graphs

A recent study solves this problem by not looking for the exact nearest neighbors but for the approximate nearest neighbors (ANN) while keeping a very high accuracy [5]. They greatly improve performance by only calculating the distance for a small subset of the context blocks using a hierarchical navigable small world graph algorithm.

It works by dividing the block points into different layers, each more dense than the next. If a point exists in a certain layer, it must exist in all denser layers. In every layer a point exists, it has k neighbors saved. Then starting at the most sparse layer a random entry point is chosen. The algorithm moves towards the neighbor that results in the smallest distance to the prompt point. When the distance is smaller than a chosen threshold for that layer, the point moves to the next. Here the process repeats until the last layer is reached. At this point the algorithm searches for the k closest points. Figure 1 shows an instance of such an algorithm.

While the HNSW algorithm is primarily used to improve the performance of the dense retrieval, it is critical that this approximate method does not compromise the correctness required for clinical systems. Despite it being an approximate algorithm, the study shows that HNSW is capable of maintaining a high accuracy score. Which for the ANN search, is measured by the ratio of the true nearest neighbors that the algorithm finds. Another big advantage is that the accuracy score can be fine tuned by modifying the 'ef' parameter which is the maximum size of the dynamic list of closest points that the algorithm keeps track of during the search process.

2.4.2 Overcoming the RAM Bottleneck via Memory-Mapped Storage

The HNSW algorithm's standard implementation introduces a new bottleneck, namely the memory consumption of the algorithm. To achieve a logarithmic search complexity, the standard implementation requires that both the neighbor layers and the entire vector database are loaded into memory.

To overcome this limitation without sacrificing the correctness and efficiency of HNSW, this research utilizes Qdrant [7], a vector database shown in recent research to have superior memory efficiency when compared to other leading systems like Milvus and Weaviate. Rather than relying on pure in-memory storage, Qdrant allows the vector database to operate using Memmap (memory-mapped) storage. This creates a virtual address space linked to files stored directly on the disk. Instead of fitting everything into RAM, the Memmap storage leverages the operating system's page cache. By configuring the vectors to be stored on disk, only the most frequently accessed nodes and vectors are dynamically cached in available RAM, while the bulk of the database rests safely on the drive.

Normally, moving database indexes from RAM to SSDs resulted in a large latency. However, a recent study invalidated that [8]. In the research it is proved that a storage based vector search is not necessarily worse than a memory based search. In fact, it is demonstrated that using disk optimization graphs can even outperform traditional memory based indexes, achieving 3.2 times higher throughput and 44.5% lower latency.

2.5 Grounding Models

Assuming the correct context blocks are retrieved, how does the RAG system restrict the final model from hallucinating, or from utilizing its internal parameters to generate content not valid for the Netherlands? Most general purpose models are trained to always generate a response, even when it is not sure about its correctness. A recent study on measuring and enhancing trustworthiness of models using RAG systems, introduces the trust-score, which grades a model in a RAG pipeline on three things:

1. Grounded Refusal: Does the model correctly refuse to answer if there are no available context blocks?
2. Claim Recall: If the answer is in the blocks, does the model extract the correct facts?
3. Attribution Groundedness: Does the model provide accurate inline citations to prove exactly from which block it got each fact?

Using this trust-score, the study experiments with both positive and negative examples. Where giving the model a dataset of full of examples improved refusal ability by 45% [11].

3 Methodology

The design choices for this methodology use the takeaways from the literature as the starting point. To test whether these hypotheses actually hold up for the NHG guidelines, the setup is split into two main parts.

3.1 Searching Algorithm

Before the model can generate a response, the correct context blocks need to be retrieved. Chapter 2 Related Work describes three possible state-of-the-art retrieval algorithms: BM25 search, AI meaning-based (vector) search, and the hybrid search.

Using sets of different types of questions with their respective required context blocks, an experiment can be performed. In this research, three types are used: direct questions using the same keywords as the guidelines, synonym questions where the medical related words are replaced by synonyms, and descriptive questions that simulate a general practitioner asking about their patient. These three question categories were selected to systematically test the retrieval layer with different real world medical prompts. The direct questions serve as a good baseline. They use the exact same keywords as the guidelines, so they test the keyword search to catch the direct word overlaps. The synonym questions, where these medical terms are replaced with synonyms, resulting in removing the exact matches to test the meaning based search. It should map these synonyms to the same space as the original keywords used. Then finally, the descriptive questions simulate a general practitioner asking about a patient in a natural, conversational format, testing the algorithm against a real world example.

In the guideline about asthma for adults the specific (Dutch) keywords used are 'lichamelijke onderzoek', '1e consult' and 'astma bij volwassenen'. So to test the systems ability to find the correct blocks even when rephrasing, these are rewritten to 'fysieke evaluatie', 'eerste bezoek' and 'astma bij iemand ouder dan 18' respectively.

Direct Wat is het lichamelijke onderzoek bij het 1e consult voor astma bij volwassenen?

Synonym Welke fysieke evaluatie voer ik uit bij een eerste bezoek voor astma bij iemand ouder dan 18?

Descriptive Een oudere patiënt komt voor het eerst met vermoedelijke astma. Wat moet ik onderzoeken?

Furthermore, these searching algorithms have parameters that can be tweaked. Per algorithm there is a score cut-off and a maximum amount of context blocks they are allowed to find. The goal of this tuning is to maximize the retrieval of correctly found context blocks while minimizing the total block count. While passing the entire guidelines dataset with every prompt would guarantee high recall, it results in enormous prompts. For the base models used for RAG-pipelines, larger prompts directly translate to higher token consumption resulting in increased costs per query and slower generation. Therefore, the algorithm must find the lowest possible cut-off threshold that preserves clinical correctness. To find the best fitting parameters, an experiment will be conducted. Testing different parameter configurations on the validation set to evaluate which specific settings yield the highest retrieval scores.

3.1.1 Implementation

The traditional retrieval part is built upon BM25 described in Chapter 2 Related Work, it is recognized as an industry-standard probabilistic text retrieval algorithm. The implementation loads all guideline context blocks and applies standard alphanumeric tokenization. To prevent highly repeated clinical terms from heavily dominating the score, it utilizes term frequency saturation. Furthermore, it incorporates document length normalization to ensure that shorter context blocks are not forgotten or penalized against lengthy medical tables.

Finally, to adhere to the natural formatting of the guidelines, it dynamically prepends document titles to the section paths to build a header list. It ensures that matches hitting a title or subtitle carry significantly more weight than matches in the body text.

The AI meaning search layer uses the gemini-embedding-2 model from Google to map prompts and context blocks to a 3072-dimensional space. The choice to use this model stems from its retrieval performance across diverse language benchmarks and its cost-effective, free accessibility through the Google API. This allows the embedding algorithm to map medical-specific Dutch terminology to a high dimensional vector space, without resulting in computational or financial constraints. To reduce the computational bottleneck of calculating exact distance scores against every single block in the database, the algorithm implements an Approximate Nearest Neighbor (ANN) search. This is executed inside Qdrant using a Hierarchical Navigable Small World (HNSW) graph as described in Chapter 2 Related Work.

The hybrid search combines the traditional and meaning based retrievers into a single complementary system. Because BM25 and the vector embeddings operate on fundamentally different matching principles, their raw scores cannot be directly compared. To resolve this, the implementation uses Reciprocal Rank Fusion (RRF). The hybrid script executes both queries independently, fuses the candidates based strictly on their reciprocal position in the sorted result lists, and applies a strict upper limit.

3.2 Grounding and Restricting the AI

After the context blocks are correctly retrieved, they should be used to ground the model. It is only allowed to use information from the guidelines and not hallucinate or rely on its training data. In this experiment, three levels of grounding are tested:

1. Just giving the model the retrieved context blocks with the prompt.
2. Giving the retrieved context blocks and modifying the prompt to specifically ask for information from the guidelines.
3. Giving the context blocks, some examples on how to answer and the modified instruction prompt.

In addition to the three question categories for the retrieval experiment, two new types are introduced, a question with no available context blocks, and a question with only invalid context blocks. The model should in both cases refuse to answer.

Due to the time-constraint for this research the pipeline lacks inline references to the used context blocks. That means the trust-score from Chapter 2 Related Work is slightly modified to just the grounded refusal and claim recall.

It should be noted that both the instruction and the examples used in this experiment were manually created. While this serves as an effective proof-of-concept, they are not clinically validated. Chapter 6 proposes future work, in which certified medical experts refine and expand these instructions to match the exact phrasing and structural requirements of the professional medical practice.

3.2.1 Implementation

To ground a model from a standard generative language model into a strict clinical tool, the retrieved context blocks are directly added to the prompt alongside a system instruction.

This instruction includes a grounded refusal as it explicitly commands the model to act as a careful medical assistant and refuse to answer if the provided context blocks lack the necessary facts.

The three methods will run on the test set consisting of ten questions and are evaluated by hand for the two trust-score metrics.

4 Experimental Setup and Results

4.1 Splitting the Guidelines

It is not desirable to give the entire guidelines dataset with every prompt as that will result in enormous prompts. So splitting the dataset into smaller context blocks is needed. An initial exploratory evaluation revealed fixed-size chunking to be ineffective. It arbitrarily split bodies destroying the clinical context. An attempt to split strictly before headers (context-based splitting) while keeping the blocks as large as possible also failed as the resulting blocks became too big. The ideal solution proved to be context-based splitting while minimizing the block sizes. This resulted in small but usable context blocks. No experiment was needed.

4.2 Retrieval

To make sure useful context blocks are retrieved but to not give too many and increase the prompt size again, this experiment focusses on limiting the amount of blocks while keeping the correctness high. A final limit is introduced after combining and reranking the two different searching algorithms. For this to properly work it is important that both the traditional and meaning search have some spare room to accurately rerank the results. Therefore, in this experiment, the score thresholds for these algorithms are kept low to increase the amount of found blocks.

To find the optimal boundary, an automated step-function experiment was conducted. Looping iteratively over the upper limit parameter, scaling the allowed number of retrieved blocks upward, the algorithm tests the reranked search results against the ground-truth validation set. Figure 2 shows the results in a clear graph. At 44 blocks, the combination search reaches 100% recall accuracy, while at only 26 blocks already 86% accuracy is reached. If budget is a large constraint for a user, this may be a valid trade-off. When a high accuracy is needed, scaling up might be the best way to go.

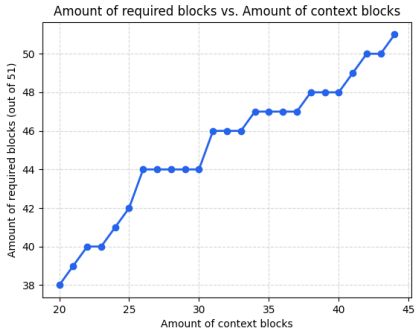

Figure 2: The amount of correctly returned context blocks vs the total amount of returned context blocks by the hybrid search algorithm.

For every run the used settings are saved together with diagrams showing the resulting scores. For every question category a diagram is saved. Figure 3 displays these distributions across final limits set to 20 context blocks, 32 blocks and finally 44 blocks.

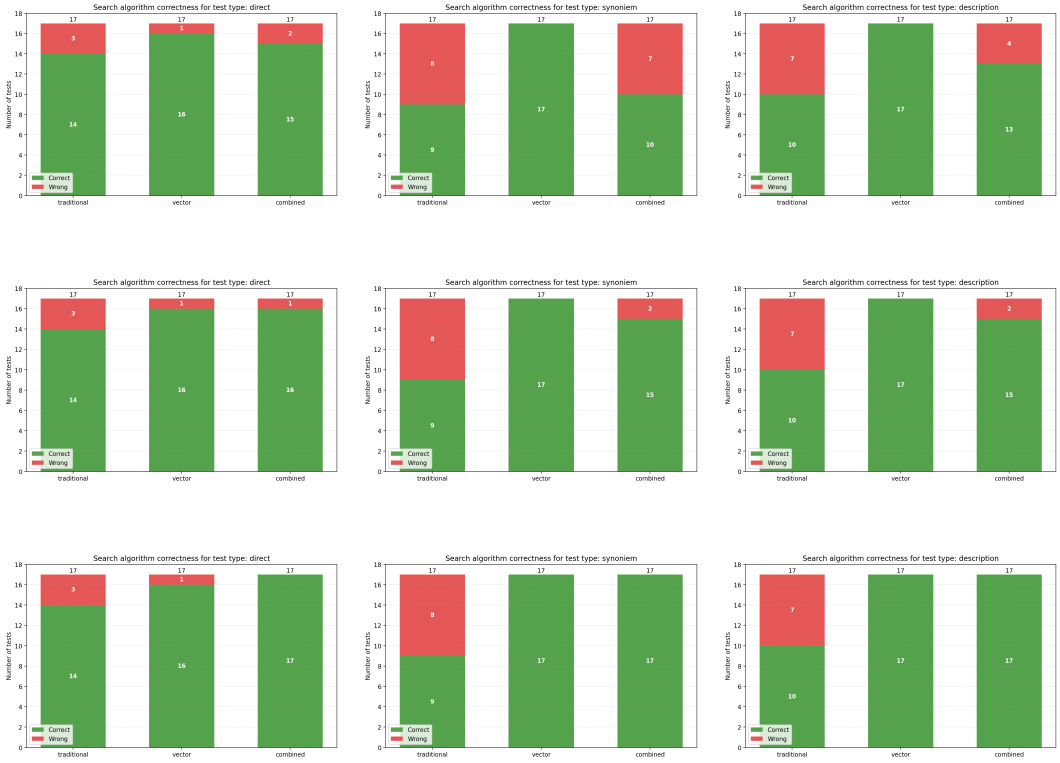

Figure 3: The amount of correctly returned context blocks for three different limits after reranking. 20, 32 and 44 from top to bottom. The three diagrams per row represent the three searching algorithms which are the traditional, AI meaning, and hybrid search from left to right. Across three categories: direct, synonym, and description.

Note that the meaning based search always outperforms the traditional search, even for the direct category where the hypothesis was that the traditional search would do the heavy lifting. This unexpected outcome may be attributed to the increasing understanding of the embedding models. However, it is noteworthy that even though the vector search outperforms the traditional search, when the limit is set high enough, combining them results in an even better score. In figure 4, the overall scores of the three different limits are presented. Because the threshold and top K for the two individual searches are the same, these parts of the diagram will always be the same. That the combining of these two algorithms results in a better score than both of them individually is precisely the strength of RRF. By fusing their ranked lists it leverages the complementary strengths of both methods, successfully retrieving edge-case context blocks that either algorithm operating independently would miss.

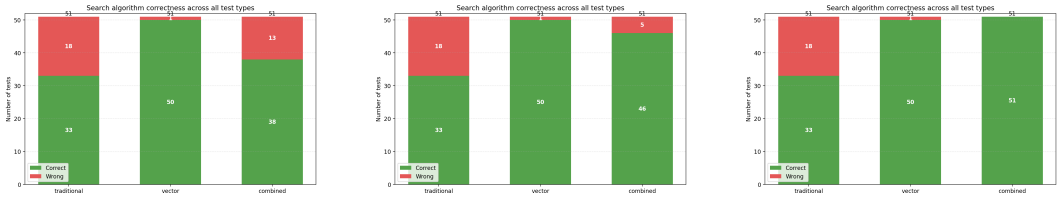

Figure 4: The overall test results when limiting the amount of context blocks to 20, 32 and 44 blocks from left to right.

4.3 Grounding

To make sure the model only uses information from the context blocks it should be grounded. Chapter 3 Methodology presents three ways of modifying the prompt to attempt this.

	Just blocks	Blocks + instruction	Blocks + instruction + examples
Grounded refusal	0/4	4/4	3/4
Claim recall	6/6	6/6	6/6

Table 1: The performance of the three grounding methods on grounded refusal and claim recall.

As seen in table 1, for questions where the correct context blocks are found, all three results find the answer. However, when context is missing or invalid context is given, the prompt with no instruction or examples still answers using its internal training data. What is surprising, is that giving the examples results in a lower grounded refusal score. This can be explained by an unintended information overlap within the human made examples and the test question. Because both are in the medical domain, useable information is found in the example context blocks. While in a live setting, correctly answering the question using valid information from the, by medical experts created, positive examples would be acceptable. In this experiment, it results in a failure.

5 Responsible Research

The integration of Large Language Models into the medical domain introduces significant ethical challenges, primarily concerning patient privacy. General practitioners might input specific patient symptoms or sensitive medical histories into the prompt to retrieve relevant guidelines. Because the proposed system relies on external base models it is imperative to guarantee that the data processed by these models is not stored externally or utilized for future training purposes, as it constitutes strictly private medical information.

While the RAG pipeline is designed to ground the model and reduce hallucinations, it still may generate factually incorrect medical advice. However, because this tool is developed solely for general practitioners rather than direct patient use, a small disclaimer stating that information can in fact be wrong will suffice. They have the medical expertise to critically evaluate the output before it is used for any clinical application.

6 Conclusions and Future Work

This research aimed to answer the primary research question: How can a Retrieval-Augmented Generation (RAG) pipeline be constructed for Dutch NHG-guidelines? By breaking this problem down into four distinct sub-questions focused on data processing, retrieval optimization, storage scalability, and model grounding, the research successfully demonstrates a complete, factually correct, and scalable system for general practitioners in The Netherlands.

First, evaluating guideline splitting algorithms revealed that fixed-size splitting destroyed the context. Splitting before headers (context-aware) and making context blocks as large as possible resulted in excessively large blocks. Undesirable for a system that requires high precision while keeping token costs low. The optimal approach proved to be context-aware splitting while minimizing the block size. Resulting in cohesive blocks, while keeping the token costs per block limited. Second, the retrieval layer experiment, demonstrated that combining an industry-standard keyword search, BM25, with an AI meaning-based vector search using Reciprocal Rank Fusion (RRF) achieves better retrieval performance compared to either method individually. Fusing these independent ranked lists successfully captures edge case guidelines that a standalone retriever would miss.

Third, embedding the context blocks into a 3072-dimensional vector space, created a potential computational bottleneck where exhaustive distance calculations across the entire database became infeasible. To solve this, the system uses an Approximate Nearest Neighbor (ANN) search utilizing a Hierarchical Navigable Small World (HNSW) graph. While the standard HNSW implementations require keeping the entire database, and graph indexes in memory, the Qdrant database utilizes a memory-mapped storage to offload parts directly to the SSD drive while keeping a high throughput and low latency. This makes the system scalable, even on lesser computers.

Finally, the experiment on grounding the generative model, showed that providing retrieved context blocks successfully informs the model when the correct blocks are present. However, when relevant context is missing or invalid, without additional instructions, the model falls back on its internal training data. Giving strict instructions successfully enforces grounded refusal. While adding examples could theoretically improve this refusal behavior further, the time constraints of this project limited the test set size, leaving this unproven. Furthermore, giving the model positive examples, even resulted in a test failure. The specific test question had overlap with the context blocks of an example question. Although in a

real-world setting, where the context blocks are proven to be correct, and the examples are made by medical experts this would be acceptable, in this experiment it counts as a failure.

Combining these optimized chunking, hybrid retrieval, scalable storage, and instruction-based grounding mechanisms results in a fully functional and reliable RAG pipeline tailored to Dutch primary care.

Despite the positive results, there are several areas where it could be improved. The instructions and examples for the generative model should be developed and validated by certified medical experts to make sure they align with the professional standards. Alongside a more extensive test set on the grounding.

As generating embeddings is computationally expensive and time-consuming, future research should explore integrating newer, more cost-effective embedding models as they become available. Finally, because the NHG guidelines are continuously updated, long-term database maintenance strategies must be established. Further research is required to determine whether the entire database must be re-embedded, or if a smarter indexing algorithm can identify and re-embed only the modified sections. Developing an automated method to track modifications (possibly, across multiple context blocks) represents a logical next step in deploying this system in active clinical practice.

References

[1] Rohan Anil et al. *PaLM 2 Technical Report*. May 2023. DOI: 10.48550/arXiv.2305.10403.

[2] Negar Arabzadeh, Xinyi Yan, and Charles Clarke. *Predicting Efficiency/Effectiveness Trade-offs for Dense vs. Sparse Retrieval Strategy Selection*. Sept. 2021. DOI: 10.48550/arXiv.2109.10739.

[3] Scott Barnett et al. *Seven Failure Points When Engineering a Retrieval Augmented Generation System*. 2024. arXiv: 2401.05856 [cs.SE]. URL: https://arxiv.org/abs/2401.05856.

[4] Vyacheslav Efimov. *Similarity Search, Part 4: Hierarchical Navigable Small World (HNSW)*. Towards Data Science. Image retrieved from https://towardsdatascience.com/wp-content/uploads/2023/06/1ziU6_KIDqfmaDXKA1cMa8w.png; part of the article series "Similarity Search, Part 4". June 2023. URL: https://towardsdatascience.com/similarity-search-part-4-hierarchical-navigable-small-world-hnsw-2aad4fe87d37/.

[5] Yu Malkov and Dmitry Yashunin. "Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PP (Mar. 2016). DOI: 10.1109/TPAMI.2018.2889473.

[6] Skandari Porter Boyd and Laiteerapong. *Revisiting the Time Needed to Provide Adult Primary Care*. 2023. DOI: 10.1007/s11606-022-07707-x. URL: https://doi.org/10.1007/s11606-022-07707-x.

[7] *Qdrant: High-Performance Vector Search at Scale*. https://qdrant.tech. Accessed: 2026-06-02.

[8] Zebin Ren et al. *Storage-Based Approximate Nearest Neighbor Search: What are the Performance, Cost, and I/O Characteristics?* 2025.

[9] Stephen Robertson and Hugo Zaragoza. "The Probabilistic Relevance Framework: BM25 and Beyond". In: *Foundations and Trends in Information Retrieval* 3 (Sept. 2009), pp. 333–389. DOI: 10.1561/1500000019.

[10] K. Singhal, T. Tu, and J. et al. Gottweis. "Toward expert-level medical question answering with large language models". In: *Nature Medicine* 31.3 (2025), pp. 943–950.

[11] Maojia Song et al. *Measuring and Enhancing Trustworthiness of LLMs in RAG through Grounded Attributions and Learning to Refuse*. Sept. 2024. DOI: 10.48550/arXiv.2409.11242.

[12] Cyril Zakka et al. *Almanac: Retrieval-Augmented Language Models for Clinical Medicine*. 2023. arXiv: 2303.01229 [cs.CL]. URL: https://arxiv.org/abs/2303.01229.