Retrieval-Augmented Pipeline for Clinical Pharmacology: From MeSH-Based Queries to Local LLM Answers

THERESE ANDERSSON & MARTIN LYSÉN
BACHELOR´S THESIS
DEPARTMENT OF ELECTRICAL AND INFORMATION TECHNOLOGY
FACULTY OF ENGINEERING | LTH | LUND UNIVERSITY

Retrieval-Augmented Pipeline for Clinical Pharmacology: From MeSH-Based Queries to Local LLM Answers

Therese Andersson, Martin Lysén
thpoan@hotmail.com, martin.lysen@hotmail.com

Department of Electrical and Information Technology
Lund University

Supervisor: Christin Swahn

Examiner: Christian Nyberg

January 6, 2026

Printed in Sweden
Tryckeriet i E-huset, Lund

Abstract

This thesis investigates the extent to which literature search and summarisation in clinical pharmacology can be partially automated by utilising a locally deployed Large Language Model (LLM) coupled with a Retrieval-Augmented Generation (RAG) component, designed as a free-text clinical decision support tool for medical experts. The thesis aims to develop a prototype system consisting of a locally deployed LLM coupled with an API to PubMed. The prototype system successfully converted free text clinical questions into structured search queries, retrieving relevant articles, to support clinical decision-making. The workflow analysis enabled the development of partly automated search strategies that reflect the clinical workflow. The locally deployed RAG developed by Morphik, coupled with ColPali embeddings improved multimodal processing, enhancing retrieval quality. Clear guidelines must be established regarding when and how it should be disclosed that an AI system contributed to a patient's treatment plan, either for individual patients or for broader clinical recommendations. This work addresses a real and pressing need to enhance workflows with AI-assisted tools, without compromising on quality or accountability.

Keywords: ColPali, Information Retrieval, Large Language Models, Morphik, Retrieval Augmented Generation, PubMed

Sammanfattning

Denna kandidatuppsats undersöker i vilken utsträckning litteratursökning och sammanfattning inom klinisk farmakologi kan delvis automatiseras genom användning av en lokalt implementerad språkmodell (LLM) i kombination med en Hämtningsförbättrad generering (RAG)-komponent, utformad som ett fritextbaserat kliniskt beslutsstöd för medicinska experter. Kandidatuppsatsen syftar till att utveckla en prototyp bestående av en lokalt implementerad språkmodell kopplad till ett gränssnitt mot PubMed. Prototypen kunde framgångsrikt omvandla kliniska frågor i fritext till strukturerade sökfrågor och hämta relevanta artiklar för att stödja kliniskt beslutsfattande. Arbetsflödesanalysen möjliggjorde utvecklingen av delvis automatiserade sökstrategier som speglar det kliniska arbetsflödet. Den lokalt implementerade RAG-komponenten, utvecklad av Morphik och kombinerad med ColPali-inbäddningar, förbättrade den multimodala bearbetningen och kvaliteten på hämtningen. Det krävs tydliga riktlinjer för när och hur det ska redovisas att ett AI-system har bidragit till en patients behandlingsplan, antingen för enskilda patienter eller för bredare kliniska rekommendationer. Detta arbete tillmötesgår ett verkligt och angeläget behov av att effektivisera arbetsflöden med AI-assisterade verktyg utan att kompromissa med kvalitet eller ansvarstagande.

Nyckelord: ColPali, Informationsinhämtning, Morphik, PubMed, Retrieval Augmented Generation, Stora språkmodeller (LLM)

Preface

We would like to express our gratitude to Dr. Andersson and the department of Clinical Chemistry and Pharmacology for providing this opportunity, to our families for their endless support, and to our supervisor Swahn and examiner Nyberg for their guidance throughout the course of this thesis.

Contents

1 Introduction **1**
- 1.1 Background . . . 1
- 1.2 Purpose . . . 6
- 1.3 Objectives . . . 7
- 1.4 Problem Statement . . . 7
- 1.5 Justification for the Thesis . . . 7
- 1.6 Delimitations . . . 8
- 1.7 Division of labour . . . 8

2 Technical Background **9**
- 2.1 Terminology . . . 9
- 2.2 Large Language Model (LLM) . . . 11
- 2.3 Retrieval Augmented Generation . . . 13
- 2.4 PubMed article retrieval . . . 14
- 2.5 Backend . . . 16
- 2.6 Report creation . . . 18

3 Method and Analysis **19**
- 3.1 Overview of the work process . . . 19
- 3.2 Use of AI in thesis . . . 20
- 3.3 Elicitation and research . . . 20
- 3.4 Prototype development . . . 23
- 3.5 Evaluation and analysis of results . . . 27
- 3.6 Evaluation of sources . . . 27

4 Results **29**
- 4.1 Visual representation of the prototype . . . 29
- 4.2 Limitations . . . 29
- 4.3 Reports . . . 31
- 4.4 Showcased report . . . 32
- 4.5 Mail Follow-up . . . 39

5 Conclusion **41**
- 5.1 Evaluation of System Requirements . . . 41

5.2 Reflection on Research Questions 42
5.3 Future development 43
5.4 Ethical Considerations 43

A Screenshots of Prototype. 48

List of Figures

1.1 Manual workflow 3
1.2 Part-automated workflow 5

2.1 Illustrates how `MetapubError` and its subclasses inherit from Python's built-in `Exception` class. 16

3.1 The work process 20

4.1 Simplified overview of how the prototype works 29
4.2 The start page of a generated report. Reports were produced as HTML books according to the formatting guidelines described in the Quarto documentation. 33
4.3 The question page shows the prompt provided by the user. All questions are machine-translated from the detected language into English to facilitate processing by the LLM. 33
4.4 PubMed queries generated by the system, including variations automatically produced by the LLM. 35
4.5 The first half of the Morphik response page. It displays a warning informing the reader that the LLM will generate an answer based on the retrieved sources, and that it may occasionally fail to recognise when the provided context is insufficient. 36
4.6 This section presents the included source chunks along with their PMID, retrieval score, publication type, document type, and a Vancouver-style reference that can be copied for further use. 37

A.1 Start button to launch the protomain script in the web UI. 49
A.2 Choice to run with or without checkpoint facilitated with Yes No prompt to the user to help them navigate trough the program flow. . 49
A.3 Text box with prefilled answer to show user that freetext enquires are viable. 50
A.4 Output of machine translated query to English and prompt to validate the the meaning was preserved. 50
A.5 The translated question is prefilled in the input box for faster user editing. 51

A.6 Prompt to name the project of the query which will be the folder name and the title of the generated report, default option to just store the results in general purpose folder abstracts. 52
A.7 List of generated queries, presented with the condition and intervention terms the LLM identified in the question and the final query to send to PubMed, the user can opt to modify any of the queries in the list. 52
A.8 Button based prompt for each query in the list, DONE button escapes the prompts and continues the pipeline. 53
A.9 Prefilled prompt of query chosen to edit. 53
A.10 Prompt to search using displayed query or skip to next in list. 53
A.11 Decpits PMIDS of every article found using the selected query. . . . 53
A.12 Streaming list of accessibility check for every found article and prompt on whether to search using expanded query and fallback methods. . . 54
A.13 Result from expanded query and fallback methods, typically fails if query is not of the exact format condition AND intervention. 54
A.14 Streaming list of ingestion progression for every found article. 55
A.15 Answer from LLM after ingestion and choosing chunks with highest semantic score as sources. 56
A.16 One of four source chunks used in LLM answer, source chunks are formated and presented in easier to read form in the generated report. 57
A.17 Final step of protomain pipeline, displays runtime in seconds and progress of report creation as well as location of created report on the Computer. 58

List of Tables

1.1 Division of labor between Andersson T. and Lysén 8

2.1 ColPali training data, taken from Faysse et al. [29] 18

Chapter 1
Introduction

This chapter outlines the background and motivation for the thesis. It discusses the context of the problem, the objectives of the thesis, and provides information about the Department of Clinical Pharmacology, where the thesis was carried out.

1.1 Background

An important responsibility at the Department of Clinical Chemistry and Pharmacology at Lunds hospital is to answer patient-specific questions related to drug interactions, safety during pregnancy or breastfeeding, and other aspects of pharmacological safety. Answering such questions requires interpreting and summarising complex data from multiple sources, including:

- Peer-reviewed articles (e.g., PubMed, Embase)
- Key textbooks and reference materials
- Other online resources and databases (e.g., clinical guidelines, trial registries, statistical databases)

Manual searches are time-consuming and risk overlooking critical information. A semi-automated system could streamline the process and ensure more comprehensive coverage.

1.1.1 Examples of current workflow

The 10 physicians at the Department of Clinical Chemistry and Pharmacology have several different questions that they answer, here are two notable and recurring tasks that they have to tackle. Figure 1.1 illustrates two of the main workflows of physicians.

Consulting Colleagues on Alternative Treatments

A colleague at a healthcare unit posts a question on about switching a patient from one medication (a) to another (b) for treating OCD.
Check Relis database – Has prior evaluation been made on whether there is evidence that substance b can be used to treat OCD. The criteria for a successful search are based on:

- Results are based on an AND check for the affliction and the proposed substance, additionally the evaluation must answer the question.
- The evaluation is not too old, subjective but generally < 2 years.

Search PubMed – again using an AND search, applying filters such as clinical trials, randomized controlled studies, reviews and systematic reviews.

The results that are found can then be printed and used as the basis of a conclusion.

Evaluating Medication Risks for Vulnerable Groups

A pregnant woman asks if she can continue taking a medication.

1. Check Janusmed (janusmed.se) – provides a rating of safety for different patient groups, a 1 is deemed as safe and a 3 is considered very dangerous.
2. If the result is 2 , they proceed to PubMed and search for meta-studies (since no clinical trials are conducted on healthy pregnant women).
3. Adjusting search strategy – direct queries like pregnancy AND substance could fail, then alternative MeSH-terms like teratogenesis can be applied.
4. The retrieved results are printed and used to form an evidence-based response to the patient and an opinion on the preferred course for them.

Publication of reports

Following the completion of a report, it is typically disseminated through Relis or its Swedish counterpart, Svelic, when the inquiry pertains to treatment options. Reports concerning the assessment of drug-related risks are instead published on Janusmed.

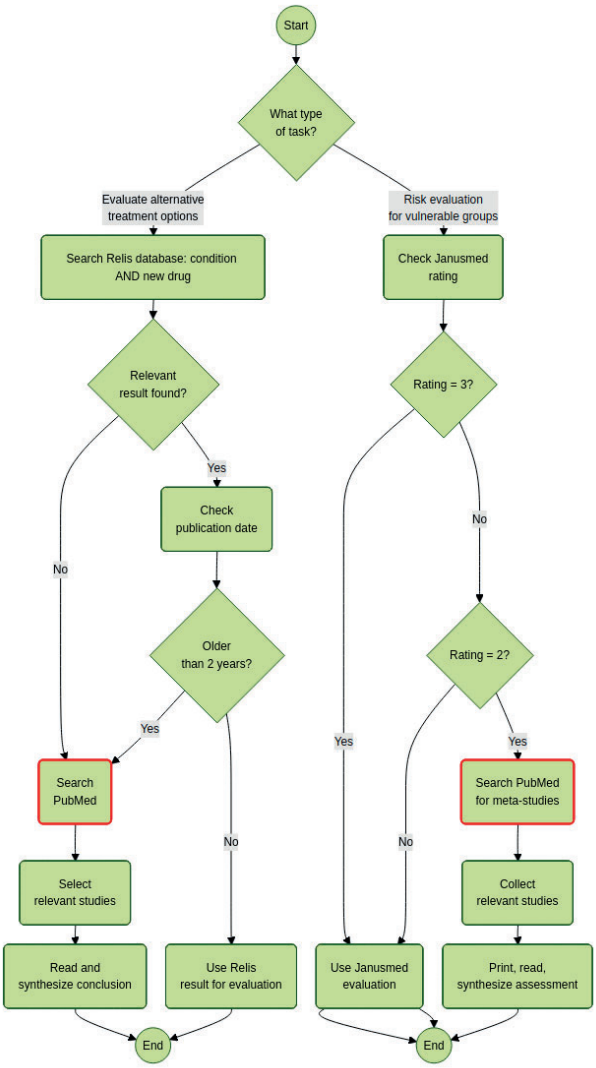


Figure 1.1: Flowchart illustrating two main workflows in medical information retrieval. The left branch shows the process of evaluating alternative treatment options, starting by searching on Relis and falling back on PubMed if needed. The right branch shows the evaluation of medication risks for vulnerable groups, starting with Janusmed ratings and extending to PubMed searches. PubMed-related steps are highlighted in red.

1.1.2 Part-automated Process

For this thesis, the thesis workers will investigate whether the retrieval of relevant articles from PubMed can be partly automated by setting up an LLM–RAG system with open-access APIs. The goal is to search for relevant articles, download them, and present the results to the physician.

The proposed process flow can be seen in Figure 1.2 and is as follows: a physician first performs an initial search on their own and concludes that previous evaluations are either based on too weak evidence or are too old to be used.

The physician then submits the request they received in a chat interface to the local LLM. The LLM checks whether it can provide an answer from its internal database, based on the semantic value and age of the articles. If this is found to be insufficient, the system queries the PubMed API to search for new data using MeSH terms. It then generates a PDF report of the retrieved sources, including clickable citation links, which the physician can use to synthesize a conclusion.

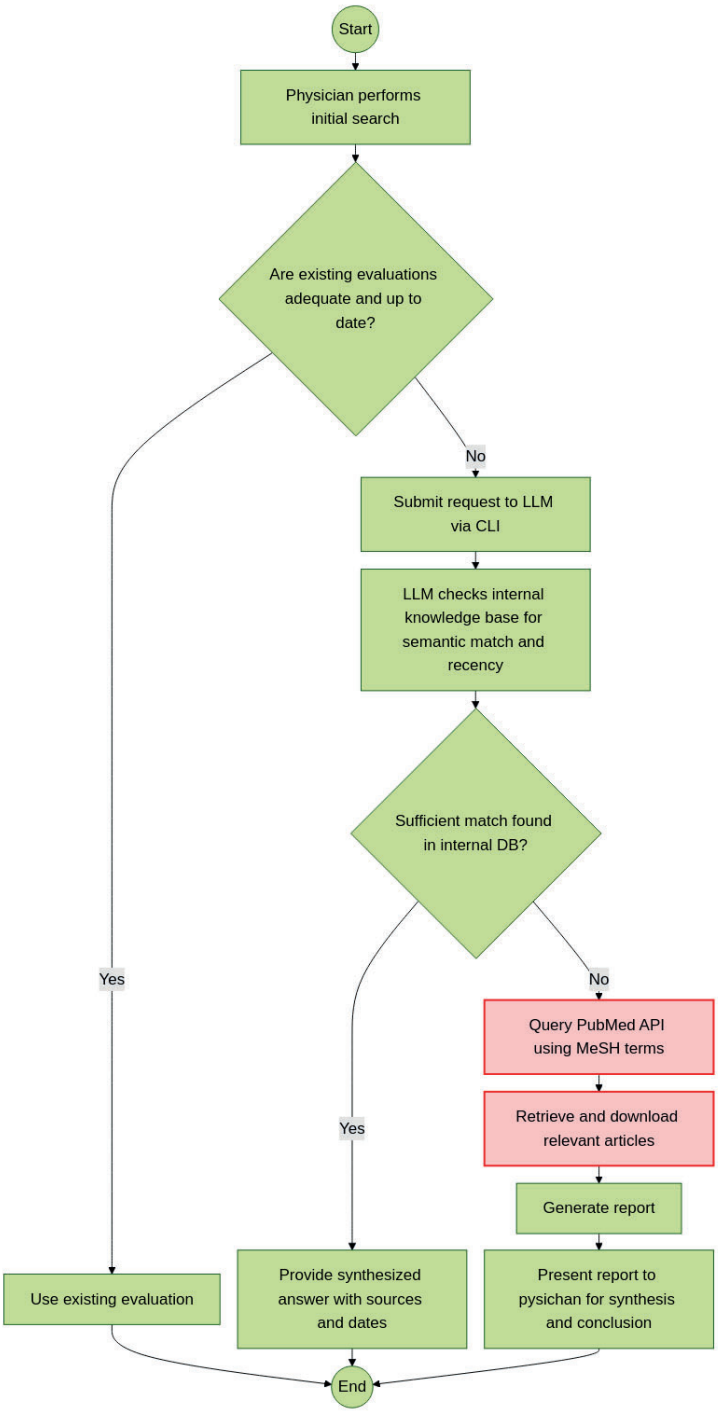


Figure 1.2: Flowchart illustrating a partly automated workflow utilising a RAG-LLM to search for relevant articles compile a report for the physician to read with the found relevant articles with links. PubMed-related steps are highlighted in red.

1.1.3 Client Requirements

The thesis is commissioned by Dr. Peder Andersson, a medical doctor in clinical pharmacology at Region Skåne, who is actively engaged in clinical research. Region Skåne is the public healthcare provider for the Skåne region in southern Sweden.

According to requirements provided by Dr. Andersson, the solution must support the following.

1. **Automated Query Transformation** – Convert free-text questions into MeSH terms or similar ontologies.
2. **Tag Approval** - Allow users approve/reject MeSH terms.
3. **Tag Adding** - Allow users to add custom MeSH terms.
4. **Tag Removal** - Allow users to remove custom MeSH
5. **Article Retrieval and Semi-Automation** – Use PubMed/Embase APIs (or scraping if necessary) to retrieve abstracts/articles, with user validation before download.
6. **Multimodal RAG Pipeline** – Develop a pipeline that processes not only abstract text but also figures, tables, and supplements to maximize the model's understanding of complex biomedical articles and improve the factual depth of generated answers.
7. **Create a reproducible logging system for queries and responses** - Using Quarto to facilitate analysis and demonstration of the system's output.
8. **Minimal interface-sources** - Design a minimal interface for exporting sources.
9. **Minimal interface-export to internal systems** – Facilitate integration into existing clinical workflows.
10. **Python-Based Implementation** – Use Python to orchestrate all data handling, retrieval, and LLM interactions.

The authors have come to an agreement with Dr. Andersson that the core requirements 1, 6, and 10 will be given the highest priority. Once these core components are in place, attention can be directed toward the implementation of requirements 5 and 7. Furthermore, a minimal user interface will be developed to meet requirement 8, while requirements 2, 3, 4 and 9 will only be implemented if resources allow.

1.2 Purpose

The purpose of this thesis is to investigate the extent to which literature search and summarisation in clinical pharmacology can be partially automated by utilising a RAG-LLM setup coupled with an API to PubMed.

1.3 Objectives

This thesis aims to develop a prototype system consisting of a locally deployed Large Language Model (LLM) combined with a Retrieval-Augmented Generation (RAG) component, designed as a free-text clinical decision support tool for medical experts. The system will connect to an API that provides access to clinical research databases such as PubMed, enabling retrieval of new information when a free-form query is poorly covered by the RAG (e.g., due to low semantic relevance or outdated content). The system's performance will be evaluated against the same LLM without RAG integration by measuring (i) the frequency of hallucinated or unsupported statements, and (ii) the clinical relevance of the generated answers.

1.4 Problem Statement

This thesis will address the following questions:

1. What is the current workflow for physicians when answering clinical questions?
2. How can a free-text query be converted into a structured search?
3. How can the system's responses be evaluated for quality and relevance?
4. Which LLM is suited for this task based on the hardware and job description?

1.5 Justification for the Thesis

This section briefly explains the motivation behind choosing this thesis topic, as well as its relevance to Region Skåne and society.

1.5.1 Lyséns interest

Martin Lysén a registered biomedical scientist licensed to practice in Sweden. Used his contacts at Region-Skåne to connect with Dr. Peder Andersson; Lysén saw this thesis as a natural connection for his dual academic fields: medical science and computer science.

1.5.2 Anderssons interest

For Therese Andersson, the choice between studying medical tech and computer science was a close call. Even during her time at the Department of Computer Science, Andersson has reflected about how her acquired skills could be applied. With a strong interested in the AI-field and a desire to create something that makes a difference, this thesis seemed like a great place to start.

1.5.3 Interest of Region Skåne and society as a whole

As healthcare systems face increasing patient volumes and growing complexity in pharmacological treatment, the demand for timely evidence-based support in clinical pharmacology continues to increase. Clinicians often rely on expert consultation to assess drug safety interactions and appropriate use in specific patient populations—tasks that require time-intensive literature searches and critical synthesis of complex data. Developing tools that can partially automate literature retrieval and summarisation offers a promising way to reduce this burden. This thesis will develop and evaluate a prototype system to examine whether such an AI-assisted tool can provide sufficiently high-quality output to be of use to trained medical experts.

1.6 Delimitations

This project will be limited to systems running on Ubuntu 18+. Article retrieval will be restricted to the use of the Entrez API (PubMed) and the Semantic Scholar API, as these two platforms are commonly used for literature searches. The RAG setup will be managed using LangChain.

1.7 Division of labour

Table 1.1: Division of labor between Andersson T. and Lysén

Activity	Lysén (%)	Andersson T. (%)
Development	50	50
Design	60	40
Testing/Evaluation	50	50
Report/presentation	50	50
Poster	40	60

Chapter 2
Technical Background

This chapter describes the key concepts and technologies necessary to understand the thesis. It introduces relevant terminology and provides a brief explanation of Large Language Models, Retrieval-Augmented Generation, and the article retrieval process from PubMed. Furthermore, it introduces Morphik, a tool used in this thesis for handling multimodal data e.g. images, video and audio.

2.1 Terminology

API

API (Application Programming Interface) is a set of rules, protocols, and tools that allow different software applications to communicate with each other. It defines how requests and responses should be structured so that systems can exchange data and functionality in a standardized way [1].

Embedding

The process of converting a piece of data—most often a word, phrase, or entire document—into a fixed-length numeric vector. These vectors are produced by embedding models, ranging from static methods such as word2vec or GloVe to contextual models such as BERT or Sentence-Transformers. The key advantage of embeddings is that semantic relationships between inputs are preserved in the vector space, enabling efficient similarity calculations (e.g., cosine similarity) and improving performance on tasks such as classification, clustering, and semantic search [2].

Furthermore, Sarma et al. [3] demonstrated that embedding approaches which incorporate both general-purpose and domain-specific representations achieve higher performance on tasks within specialized domains. Their results indicate that domain-adapted embeddings provide more informative representations than generic embeddings alone.

Graphics cards

GPUs, often referred to as graphics cards, consist of hundreds to thousands of processing cores that work parallel, in contrast to CPUs that consist of fewer cores that work sequentially. Although there are CPUs today that can handle some smaller-scale machine learning applications, GPUs are preferred when working with machine learning as they break down complex problems into smaller tasks and processes these simultaneously [4].

LangChain

An open-source framework that simplifies building applications powered by large language models. It abstracts common LLM tasks (prompt construction, token handling, retries) and lets developers 'chain' multiple components, such as prompt templates, memory, embeddings, and tools, into modular, reusable workflows. This approach makes it easier to create AI applications, from chatbots to data-analysis pipelines [5].

Medical Subject Headings (MeSH)

A controlled vocabulary thesaurus maintained by the U.S. National Library of Medicine—to refine searches and improve relevance [6].

OCR

Optical Character Recognition is a technology developed to extract data from scanned documents, images, and PDFs. These programs convert the data into a machine-readable format for further processing [7].

Ollama

An open-source platform that simplifies running and managing large language models (LLMs) directly on your local computer, providing an API to interact with language models. It allows users to run powerful AI models offline, offering benefits such as enhanced data privacy and security, as data is processed on the user's device rather than in the cloud. Ollama supports customisation through model-files, allowing users to tailor a model to their needs [8].

PMID

Short for PubMed identifier. A unique number assigned to each PubMed entry [9].

PubMed

A website developed and maintained by the National Center for biotechnology Information (NCBI), at the U.S National Library of Medicine. The website is free and contains citations and abstracts of biomedical literature. When available,

links to full text journal articles are provided; however, these are not always free [9].

QUARTO

Is an open-source, Pandoc-based publishing system that lets you write documents, books, reports, etc., from a single source file. In this thesis, pairing QUARTO with a Python logging script allows us to automatically generate readable reports on attempts, including tables and figures that illustrate questions, context, and the resulting outputs from LLM-RAG runs [10].

Relis

Is a national network of four regional medicine information and pharmacovigilance centres in Norway and answers medicine-related questions for individual patients, but also questions of more general character. SVELIC is the Swedish sister site to Relis [11].

Tokenization

The process by which a model converts raw text input into numerical tokens, including the use of masks to handle special characters and whitespace. Although all LLMs perform tokenization internally, it is sometimes necessary to manage this process separately—for example, to chunk text into smaller segments that fit within the model's context window [12, 13].

VRAM

VRAM is the GPU's memory as opposed to RAM which is the CPU's memory. Having too little VRAM can result in an LLM spilling over to the CPU's RAM which can lead to a performance drop. There are frameworks, such as Ollama, that support a hybrid model where CPU and GPU work together, making it possible to load models that would not fit on GPU alone [14].

2.2 Large Language Model (LLM)

An artificial intelligence model trained on vast amounts of text data. By learning statistical patterns in word sequences, LLMs can generate coherent, contextually appropriate text, answer questions, translate languages, summarise content, and perform a wide range of other language tasks. These models typically employ deep neural networks—most commonly transformer architectures—which enable them to capture long-range dependencies and nuanced meanings in language. Although LLMs are highly capable, they are not infallible. They generate responses by predicting the most likely next words based on their training data, rather than by verifying facts against external sources. As a result, an LLM may produce statements that are factually incorrect or misleading (e.g. misattributing quotes, dates, or statistics), fabricate information (such as citing non-existent studies or

experts), provide incomplete or oversimplified answers (omitting important caveats or nuances that a human expert would include), or misinterpret context (misunderstanding ambiguous prompts or ignoring prior context, potentially leading to contradictions) [12].

In this thesis, the LLMs used are already fine-tuned and ready for conversation. However, there are approximately three techniques that are used to regulate a LLM. These techniques are: through training data, through in-context learning, and with different hyperparameter configurations [15]. In addition, Belcic and Stryker [16] note that adjusting hyperparameters along with prompt engineering is one of the primary LLM customization methods. The prompt rules applied in this thesis are described in chapter 3.4.4.

2.2.1 Parameters

Large language models have parameters that can be categorized into three primary categories: weights, biases, and hyperparameters. According to Belcic and Stryker [16] these parameters can be described as settings that control and optimize a LLM's output and behaviour. There are models with sizes ranging from 0.5 to over several hundred billion parameters. The more parameters a model has, the more accurately the model will capture data patterns, which means that these models will be able to detect more complex patterns and relationships compared to models with fewer parameters [17].

Whereas weight and bias are configured during the model's training process, the hyperparameters are set prior to the training [17]. Belcic and Stryker [16] describe hyperparameters as external settings that determine a model's behaviour, shape, size, resource use, and other characteristics. According to Belcic and Stryker [16], although all parameters help the model, it is the hyperparameters that most directly affect the output of the model. Three examples of hyperparameters:

Context Window

LLMs have varying capacities to retain context, a limitation often referred to as the context or token window. When the input approaches or exceeds this window's size, the model may begin to "forget" earlier information, leading to reduced coherence or accuracy in its responses [12, 13].

Top-K

When generating text, a model predicts one token at a time. For each token, it evaluates all possible next tokens based on its training data and the context generated so far. The Top-K parameter limits this evaluation to the K most likely tokens. The model then randomly selects the next token from these K options, with probabilities weighted according to their likelihood [18].

Temperature

Peeperkorn et al. [15] describe temperature as a hyperparameter that controls the randomness of an LLM. LLMs generate responses by predicting the most likely next token based on a probability distribution and it is the temperature, t, that regulates the distribution by redistributing the probability mass. If the temperature is close to or greater than 1, the distribution becomes more flattened, leading to more randomness and uncertainty, while a temperature $t = 0$ essentially means that the LLM will take the token with the highest probability [15, 19].

2.2.2 Multimodal Large Language Model

In contrast to a Large Language Model, which is trained on text data and focuses on processing and generating text, a Multimodal Large Language Model (MLLM), integrates multiple types of data, allowing it to interpret both textual and non-textual information [20].

2.2.3 qwen2.5vl

The principal model used in this thesis is Qwen2.5-VL (qwen2.5vl:7b, 6.0 GB) which is included in the Morphik repository. It was released on February 19, 2025, and is based on the Qwen architecture, functioning as a visual language model [21].

2.2.4 nomic-embed-text

Nomic-embed-text (nomic-embed-text:latest, 274 MB) is the recommended embedding model for Morphik. Released on February 2, 2024, it is primarily used to generate embeddings for user queries. Without ColPali, it would also be used to embed ingested documents; however, when ColPali is enabled, document embedding is handled by ColPali instead [22].

2.2.5 llama3.1:8b

Llama 3.1 8B (llama3.1:8b, 4.9 GB) was released on July 23, 2024, and was included in the thesis to serve as the PubMed query creator. This text-only model was chosen as the largest model compatible with the workstation [23].

2.3 Retrieval Augmented Generation

Although impressive, Large Language Models face some challenges. LLMs are trained on large datasets and rely on this data when providing responses, but if given queries that handle knowledge beyond their training data, they may produce false or misleading information, also called hallucinations [24]. Retrieval-Augmented Generation, RAG, is used to improve LLM generated responses by combining an LLM's internal knowledge base with external sources, for example an organization's specific knowledge base [25]. By doing this, the LLM can retrieve

updated and more specific data and as a result reduce hallucinations [24]. In this thesis, the external knowledge base is PubMed.

RAG with vector similarity search

To be able to retrieve data, it has to be processed first, this step is called *indexing* and is the first step in the RAG process. In this step, raw data are extracted and converted into a uniform plain text format. The data is then divided into chunks and with the help of an embedding model encoded into a vector representation and placed in a vector database [24].

The *retrieval* is triggered when the RAG system retrieves a query from a user. In this step, the query and data from the database are used to create context for the LLM. The query has to first go through the same process as the processed data, in other words, the query has to be encoded into a vector representation with the help of the embedding model used during the indexing step. After the query has been turned into a vector, the system will then compute a similarity score between the query and the data in the database and retrieve the top-K chunks with the highest similarity. The retrieved chunks, together with the query, are then used as context in the prompt forwarded to the LLM [24].

Generation is the last step in the RAG system. In this step, the prompt is created with the context and sent to the LLM tasked with generating a response [24].

Multimodal RAG

Whereas a standard RAG can only handle text, a multimodal RAG builds upon this and also incorporates multiple modalities such as text, images, audio, and video [26].

2.4 PubMed article retrieval

This section describes the article retrieval process from PubMed. It also introduces Metapub a Python library used for accessing biomedical literature and databases.

2.4.1 E-utilities

E-utilities are the public APIs for the NCBI Entrez system, providing access to databases such as PubMed, PMC, and others. They consist of eight server-side programs that accept a fixed URL syntax for search, linking, and retrieval operations.

To ensure legal use, the thesis authors relied on Metapubs error handling, which is described in 2.4.3

2.4.2 Ranking of results

By default, PubMed sorts search results using the *Best Match* algorithm. This method combines traditional term-matching with additional signals, such as publication recency and metadata, and then applies a machine learning model to rerank the top results. The exact weighting of these factors is not publicly disclosed, which means that the internal ranking cannot be fully explained or reproduced.

2.4.3 Metapub

Metapub is an open-source repository authored by `nthmost` and other contributors under the Apache 2.0 license. It was selected for its simplicity and ready-made classes, which allow search queries to be submitted as plain strings. The library also manages rate limiting, ensuring compliance with the three-calls-per-second restriction that applies when using E-utilities without an API key [27].

In addition, *Metapub* provides a `FindIt` class that takes a DOI or PMID and attempts to return a direct URL to a downloadable PDF of the article. If retrieval fails, the class instead returns an error message explaining the cause. Possible messages include:

- `MISSING: ...` – Required data not available (DOI, volume/issue, etc.)
- `PAYWALL: ...` – Requires subscription or payment
- `DENIED: ...` – Access forbidden or login required
- `TXERROR: ...` – Technical, network, or server error
- `NOFORMAT: ...` – Publisher does not provide the expected format
- `NOTFOUND: ...` – Content not found at the expected location

As seen in Figure 2.1, FindIt uses a system based on structured error handling where `MetapubError` and its subclasses inherit from Python's built-in Exception class. The system distinguishes between different types of failure and, as mentioned earlier, will always return an error message describing the error.

AccessDenied will be raised when the publisher explicitly denies access due to a paywall, subscription, or login requirement. This is triggered when the HTTP response from the publisher of the article is 401 Unauthorized or 403 Forbidden [27].

When initializing `FindIt`, one of multiple additional parameters is `verify`, which means that the generated URL will be verified by testing a HTTP response. In this thesis, the `verify` parameter is set to True and the authors rely on this verification to see if the wanted article is open access and legal to download.

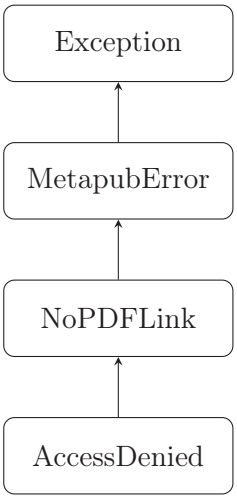


Figure 2.1: Illustrates how `MetapubError` and its subclasses inherit from Python's built-in `Exception` class.

2.5 Backend

The backend is largely handled by Morphik, a database solution tailored for use in AI training, and comes with features usable to the thesis such as PDF mining and vector embedding. It has support for local hosting, making it possible to run without passing telemetry to a third party.

For this thesis Morphik will be used in a research and development setting, which falls under the Additional Use Grant section of their license:

> The Licensed Work is (c) 2023–2025 Morphik, Inc. Additional Use Grant: You may make production use of the Licensed Work provided that the total gross revenue attributable to that use (by you or your organization) is less than US $2000 per month [28].

2.5.1 Morphik UI

Morphik includes a simple browser UI that allows users to upload documents, create knowledge graphs, and design workflows. The document upload feature supports various file types including PDFs and text files, which are processed and stored for analysis in the database. The PDF viewer tab enables users to browse and review uploaded documents within the interface. The knowledge graph creation feature builds structured representations of relationships and entities extracted from documents, visualizing connections between concepts. The workflow design feature allows users to create custom processing pipelines that define how documents are processed, analysed, and transformed for specific use cases [28].

2.5.2 Morphik API

The Morphik REST API serves as the core interface for programmatic interaction with the Morphik system. This API provides endpoints for document management, knowledge graph operations, and workflow execution. The Morphik UI consumes this API for all user-facing operations [28].

Beyond supporting the frontend interface, the REST API is used in backend integrations. For this thesis the API is paired together with Metapub functionality. This enables retrieval of articles from the NCBI database and ingestion into the Morphik database.

2.5.3 Morphik image extraction

Morphik includes support for the ColPali model developed by Faysse et al. at Illuin Technology and CentraleSupélec [29]. ColPali performs document retrieval directly from visual data by ingesting each page as an image and encoding it through a Vision–Language Model. Each page is represented as a set of patch embeddings, projected into a shared multimodal space where both visual and textual cues are captured jointly. During retrieval, a late interaction mechanism compares query token embeddings with page patch embeddings, enabling fine-grained matching between query terms and relevant document regions. This approach removes the need for OCR or manual text segmentation and provides faster indexing while maintaining high retrieval accuracy for visually rich documents. The late interaction (LI) operator is formally defined in [29] according to:

> Late Interaction. Given query q and document d, we denote as $E_q \in \mathbb{R}^{N_q \times D}$ and $E_d \in \mathbb{R}^{N_d \times D}$ their respective multi-vector representation in the common embedding space $\mathbb{R}^D$, where N_q and N_d are respectively the number of vectors in the query and in the document page embeddings. The late interaction operator, $LI(q, d)$, is the sum over all query vectors $E_q(j)$, of its maximum dot product $\langle \cdot \mid \cdot \rangle$ with each of the N_d document embedding vectors $E_d(1\!:\!N_d)$.

$$LI(q, d) \;=\; \sum_{i=1}^{N_q} \max_{j=1}^{N_d} \langle E_q(i),\, E_d(j) \rangle, \tag{2.1}$$

where $E_q \in \mathbb{R}^{N_q \times D}$ and $E_d \in \mathbb{R}^{N_d \times D}$, and D is the embedding dimension.

ColPali training data

According to Faysse et al. [29], ColPali was trained on a mixture of datasets, of which approximately 63% originated from academic sources. An undisclosed proportion of these sources were directly related to the medical sciences. The full composition of the training data, as reported in their paper, is shown in Table 2.1.

Table 2.1: ColPali training data, taken from Faysse et al. [29]

Dataset Split	Split Size	Language	Domain
DocVQA	39,463	English	Scanned documents from UCSF Industry
InfoVQA	10,074	English	Infographics scraped from the web
TATDQA	13,251	English	High-quality financial reports
arXivQA	10,000	English	Scientific figures from arXiv
Scraped PDFs	45,940	English	Varied PDFs from 3,885 distinct URL domains
TOTAL	118,695	English-only	Mixed

2.6 Report creation

After the sources have been queried and retrieved, an html book report is generated according to the rules defined in a *.ccs* file and a set of *.qmd* files detailing the content of each chapter. The report includes the user's original question to the LLM, the transformed queries submitted to the Morphik database and to PubMed, and log entries of the retrieved articles. In addition, the LLM's answer to the posed question is provided accompanied with the source chunks used as context to generate the answer.

Chapter 3
Method and Analysis

This chapter describes the work process of the thesis and the methods used throughout the thesis. It also discusses why these different methods were chosen. The following methods have been used throughout the thesis:

- Literature review
- Interview with clinical staff
- Prototype development
- Evaluation and analysis of results
- Documentation and reproducibility using Quarto

3.1 Overview of the work process

The authors of the thesis have worked according to iterative development principles, with the thesis divided into three main phases: an elicitation and research phase, a prototype development phase, and an evaluation and analysis phase. The full work process can be seen in Figure 3.1. Before the research began, a project plan was developed.

The research phase took place during the beginning of the thesis and included a literature review, meetings with Dr. Andersson, an observation session of the clinical workflow, and a focus group meeting. This is explained more thoroughly in chapter 3.3.

Once the research was completed, the development of the prototype began. The prototype was developed incrementally with continuous feedback provided by Dr. Andersson and his colleagues through weekly meetings and email correspondence. This phase consisted of multiple cycles. Before each development cycle, the authors discussed how to implement the desired functions. Once the implementation was completed, it was tested independently before being integrated into the RAG pipeline. After testing, the process was iterated until a functional prototype was achieved.

After a functional prototype was achieved, Dr. Andersson and a colleague assessed the RAG system as a whole and evaluated the reports produced by the system. Their feedback provided guidance for subsequent development cycles.

Throughout the thesis, the authors worked closely with Dr. Andersson, who was kept informed of the problems encountered during development and provided feedback and guidance whenever possible.

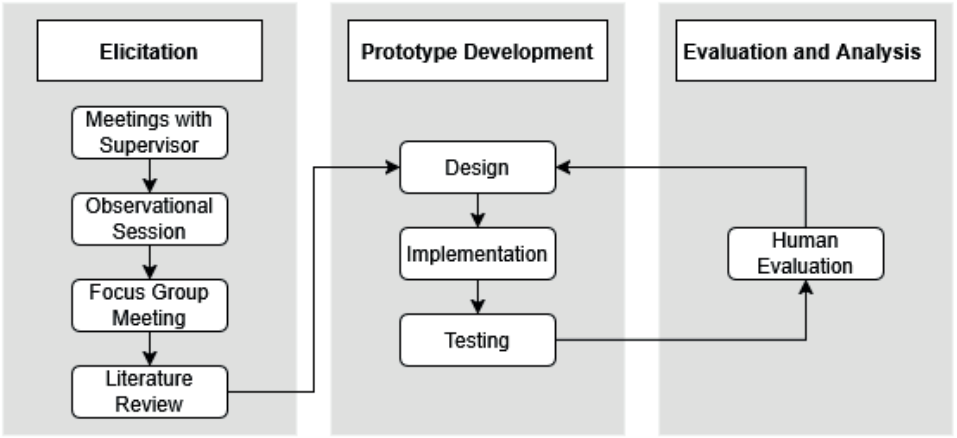


Figure 3.1: The work process

3.2 Use of AI in thesis

We hereby affirm that this Bachelor thesis was composed by ourselves and that the work herein is our own except where explicitly stated otherwise in the text. This work has not been submitted for any other degree or professional qualification, except as specified; nor has it been published. Where other people's work has been used (either from a printed source, internet, or any other source), this has been carefully acknowledged and referenced. During the preparation of this thesis, we have used ChatGPT-5 but also the models hosted locally on Ollama such as llama3.2-vision and qwen2.5vl to assist us in the writing process to improve language, flow, and readability, but also to give feedback on written code. After using this tool, We have reviewed and edited the content as needed, and take full responsibility for the content of the whole thesis.

3.3 Elicitation and research

This section describes the preliminary studies that were carried out before the implementation phase. At the beginning of the research phase, a meeting was held with Dr. Andersson in which a use case was presented to give the authors a clearer understanding of his expectations and requirements for the prototype. The fundamental requirements are listed in chapter 1.1.3. Although a set of baseline

requirements was established early on, the authors remained open to adjustments throughout the implementation phase as new insights emerged.

3.3.1 Observation of clinical workflow

The first step in the process was to gather information about how physicians in the Division of Clinical Chemistry and Pharmacology work today. To obtain this information, an observational session was conducted where one of the authors shadowed a physician performing two recurring tasks, illustrated in Figure 1.1. Throughout the session, the author occasionally asked the physician to explain the reasoning behind certain actions. Although most physicians have their own ways of working, these workflows were confirmed during a meeting with several physicians to represent the basic workflow.

3.3.2 Focus group meeting

During the development of this thesis two focus group meetings were conducted. A larger meeting was held at the beginning of the development phase with the physicians at the department. During this session the purpose of the thesis was presented and the new suggested workflow was shown, illustrated in Figure 1.2. Some questions that were discussed:

- Which platforms do physicians primarily use to search for medical literature?
- Whether there are any alternatives to using PubMed?
- How are the results from PubMed ranked?
- How are the searches structured?

The second focus group meeting consisted of Dr. Andersson, an intern, and a former engineer now doctor, and took place a month into the development phase. The focus of this meeting was on how existing code could be improved and on what new functions should be focused on in upcoming development cycles. Some of the topics that were discussed:

- Whether it would be possible to manually remove or add articles?
- Since, at this stage of development, an LLM generated the queries sent to PubMed, a discussion arose about implementing a verification step allowing the user to confirm whether the generated query was reasonable.
- The possibility to add a filter to include or exclude specific MeSH terms.
- Whether multiple PubMed searches could be conducted using a single query.

3.3.3 Literature review

Initially the authors read up on articles describing how to set up a RAG-LLM setup in order to do it from the ground up using OLLAMA together with LangChain and some database. Multiple types of databases were discussed to find one that efficiently could store and connect the entry based on semantic score. Eventually

the decision was made to use Morphik for the backend since it solved the DB, UI and API handling, although the existing functionality had to be expanded on. Morphik was discovered after searching on online forums for open access solutions on how to extract information about tables and figures from PDF files.

Much of the literature focused on how to implement the components, such as the documentation for Morphik and Metapub repositories.

Metapub was discovered during research on PubMed's API, E-utilities. When reviewing the official documentation for E-Utilities, it was found that the examples were written in Perl, a programming language that neither of the authors was familiar with, and the documentation itself was not very well documented. Therefore, a simpler approach was desired. Since the prototype was intended to be written in Python, a Python library capable of handling the E-utilities APIs was sought, and Metapub was eventually found on GitHub.

3.4 Prototype development

Python libraries such as *pdfplumber*, *Camelot*, and *PyMuPDF*, which are widely used for table extraction and data analysis, were initially reviewed and tested; however, when applied to a range of medical articles obtained from PubMed, they were unable to extract tables reliably. Consequently, Morphik was selected, and once the workflow had been defined (Figure 1.2), development proceeded with configuring Morphik in a local environment to ensure information privacy and implementing classes to query the Morphik database using their Python Software Development Kit (SDK). The main components were:

Ollama_ communication - Used to call Morphik using the Python SDK

pubmed_ API - Used to query PubMed and create lists of available articles for download

queryPM - A class responsible for taking a user question and returning one or more valid PubMed queries in a JSON dictionary. Because of the unpredictable nature of LLMs, a utility class was constructed alongside which parsed the output from the LLM and tried to validate it as JSON. In this class, methods were also included to expand and create fallback variations of the initial query to widen the scope of the search.

A separate class *generate query with variants and log results* was responsible for iterating over the created queries and variations and log the result of every search, and to allow the user to alter the query before sending it to PubMed, this to allow for human validation in the pipeline.

A class named *protomain* was created to call on every component class in order to carry out the intended pipeline.

Frontend adaptation

To integrate the *protomain* with the existing browser UI, modifications were made to the Morphik repository. A new page, *prototype main*, was added and styled as a CLI-like interface. It includes contextual button prompts that depend on the message type forwarded through the *send_ message* utility class.

The utility class exposes two main methods: *send_ message* and *wait_for_ input*. *send_ message* behaves similarly to a print command but includes an internal flush to compensate for latency introduced by the Python server. *wait_for_ input* displays the appropriate prompt type and collects user input.

Three prompt types are implemented:

- **yes/no**: Presents two buttons for binary choices.
- **numerical**: Displays a list of numbered buttons, each representing a selectable object.

- **text**: Provides a free-text input field.

3.4.1 MeSH-dictionary

After feedback from a few reviewed reports, it became clear that the LLM tended to hallucinate MeSH terms. To handle this problem, the entire 2025 MeSH library with around 30,000 terms was ingested into the database in the form of .txt files and then supplemented with metadata consisting of the term, its synonyms, tree-number and definition.

3.4.2 Workstation setup

The machine was provided by Dr. Andersson with the Linux distribution Pop!_OS, while Pop! is a fork from Ubuntu 17.10, this was a slight deviation from the previously agreed upon Ubuntu 18 OS but was deemed inconsequential by the authors.

To get the Morphik backend up and running, the setup guide for Linux in the Morphik documentation was followed [30], except for a few notable points:

- A virtual environment was created for the repo. Since the Linux distribution depends on Python for system-critical functions, this step was necessary to ensure system stability.
- The package **flash-attn** had to be pinned to version $2.7 < 2.8$ to make **ColPali** work. This particular package also required compilation from source, as none of the precompiled packages matched the workstation.
- The package **setuptools** had to be manually added using ‘uv pip install‘.

3.4.3 PubMed API

To search and query PubMed, the Python library Metapub was used. Metapub consists of several data retrieval classes, but in this thesis **PubMedFetcher** was utilized. PubMedFetcher serves as the primary interface for PubMed literature searches and supports article retrieval with PMID, DOI, PMC ID, as well as by citation details. In this thesis, the authors mainly worked with PMID.

A class called pubmed_api was created to handle the queries sent to PubMed. The class consists of three methods:

- `search_query_string(base_query)` - Takes a query and returns a list of valid PMIDs from PubMed.
- `direct_pdf_link_pmid(pmid)` - Looks for a direct link to the pdf file for a given PMID, returns the link if found.
- `fetch_pdf(pmid, out_dir="downloads")` - Downloads the pdf for a given PMID, saves it to the specified directory (default: "downloads"), and returns the path to the file.

search_query_string() is the core of the class. This method takes a query formatted by the LLM according to the ruleset described in 3.4.4 and uses the PubMedFetcher to get a list with PMIDs related to the query. Although the maximum PMIDs that can be returned are 250, the limit has been set to 50 in this thesis. When using the PubMedFetcher, some PMIDs may be returned as "None", thus the list must be checked for invalid values. Before the list is returned by the method, it is looped through to extract data about all the found articles. Data that are extracted: pmid, doi (if available), title, journal, publication date, url to the article in PubMed and MeSH-terms (if available). This data is then written to a CSV file and saved for later use.

3.4.4 Prompt engineering

To enable the generation of formatted JSON objects, two rulesets were defined in two .txt files. These files were referenced by the LLM in queryPM and served as contextual guidance during every invocation. The purpose of this approach was to constrain the model's output format and ensure consistency across queries.

Despite these rules, the LLM frequently produced incorrectly formatted or incomplete JSON objects. The implemented utility class often failed to correct for longer or more complex user queries, primarily due to excessive JSON nesting and syntax errors.

The rulesets were not static but evolved progressively as new types of formatting errors were identified after each round of prompt generation. The overarching goal was to improve reproducibility and enforce a consistent JSON structure that could be reliably parsed and interpreted by downstream components. The smaller of the rulesets was a stripped down version of the ruleset presented in 3.4.4.

Query creation

The creation of the query is a two step process where the first iteration takes the user question and returns two sets of dictionary's containing the identified condition and intervention terms, these can then be referenced against the MeSH dictionary, the found MeSH terms are then passed on to the second iteration and used to construct the final query.

Ruleset

The full ruleset used to instruct the LLM in how to parse the context is presented below. The smaller rule set contains the same rules but is trimmed down to only instruct how to create the condition and intervention terms.

1. **Overall structure**
    - Every query must follow the pattern: (condition terms) AND (intervention terms)

2. **Condition terms**
    - Identify all words related to disease, injury, toxicity, or outcome.
3. **Intervention terms**
    - Identify all drugs, treatments, or exposures.
    - If a combination drug has no exact MeSH entry, list each component separately using OR.
    - Example: “piperacillin/tazobactam” → (Piperacillin OR Tazobactam)
    - If two or more drugs are explicitly co-administered, combine them with AND within the intervention group.
    - Example: (Vancomycin AND (Piperacillin OR Tazobactam))
4. **Parentheses**
    - Always group logically: ((condition terms) AND (intervention terms))
5. **Fallback**
    - Ensure every key concept (condition or intervention) is represented, even if no official MeSH term exists.
    - Primary attempt: use provided MeSH terms.
    - Secondary attempt: create a free-text [Title/Abstract] variant if no MeSH term exists.
    - Example: "Central Fever" → Central Fever[Title/Abstract]
    - Multiple variants: combine MeSH and free-text alternatives using OR.
    - Example: Fever[MeSH] OR Neurogenic Fever[Title/Abstract] OR Central Fever[Title/Abstract]
6. **Output format**
    - Output must always be valid JSON.
    - If multiple separate clinical questions are detected in the same input, output a list of objects.
    - Each object must include:
        - "id" — sequential number or question label (e.g. "1", "2")
        - "condition_terms" — array of condition terms
        - "intervention_terms" — array of intervention terms
        - "final_query" — full PubMed query string for that sub-question
    - Example output:
      [{ "id": "1", "condition_terms": ["Pneumonia[MeSH]", "Community-Acquired Pneumonia[MeSH]"], "intervention_terms": ["Azithromycin[MeSH]"] "final_query": "((Pneumonia[MeSH] OR Community-Acquired Pneumonia[MeSH]) AND (Azithromycin[MeSH]))"}]
    - If no valid condition/intervention pair is found, output a single JSON object:
      {"final_query": "NO VALID QUERY"}

3.5 Evaluation and analysis of results

In this section, the resulting reports created from the system are discussed and how they were evaluated.

3.5.1 Human Evaluation

Given that the authors did not possess the necessary medical expertise to assess the factual accuracy or practical usefulness of the LLM-generated responses, an expert review was required. For this purpose, Dr. Andersson and his colleagues evaluated the generated reports and provided clinical feedback.

To ensure a degree of standardization, the user queries were sourced from completed reports hosted on SVELIC, the Swedish sister site of the Norwegian Relis.

The physicians examined each generated report and compared it with the corresponding existing answer to the same question. Their evaluation focused primarily on the relevance of the cited sources, followed by an assessment of the factual accuracy and clinical appropriateness of the LLM's generated answer.

3.6 Evaluation of sources

- Academic papers
  - Sources [1, 3, 12, 15, 24, 26, 29]
  - Academic papers published by academic publications that require a peer review, these hold a high standard and can be generally trusted.
- Government sources
  - Sources [6, 9, 11]
  - Government agency's generally hold a high standard for the information published on their sites as it is meant to serve the public, although agency's can be susceptible to political pressure.
- IBM, Nvidia and technical reports from other sources
  - Sources [2, 4, 7, 13, 16, 17, 18, 19, 20, 21, 22, 23, 25, 31]
  - IBM and Nvidia are at the forefront of AI and have a vast pool of published papers, while they can be considered a reliable source of information, it is probably skewed towards their own solutions. In the same vein other actors in the industry that release technical reports do so in order to inform the industry and investors about the capability of their new models. However, these tech reports are not peer reviewed by a trusted third party.
- Medium articles and git repositories
  - Sources [5, 8, 10, 14, 27, 28, 30]

– Medium is a publishing platform where anyone can contribute, which means that its content has to be approached critically. In this case, the Medium article referenced together with the referenced GitHub repositories provided practical insights on how to get the prototype working. However, it is likely that the authors' solutions are biased.

Chapter 4
Results

In this chapter, the results from the reports evaluated by Dr. Andersson's colleagues are presented, together with screenshots of the prototype. The limitations identified during the development of this thesis are also described. In addition, the web-based user interface is included in Appendix A.

4.1 Visual representation of the prototype

Figure 4.1 presents a simplified overview of the final prototype.

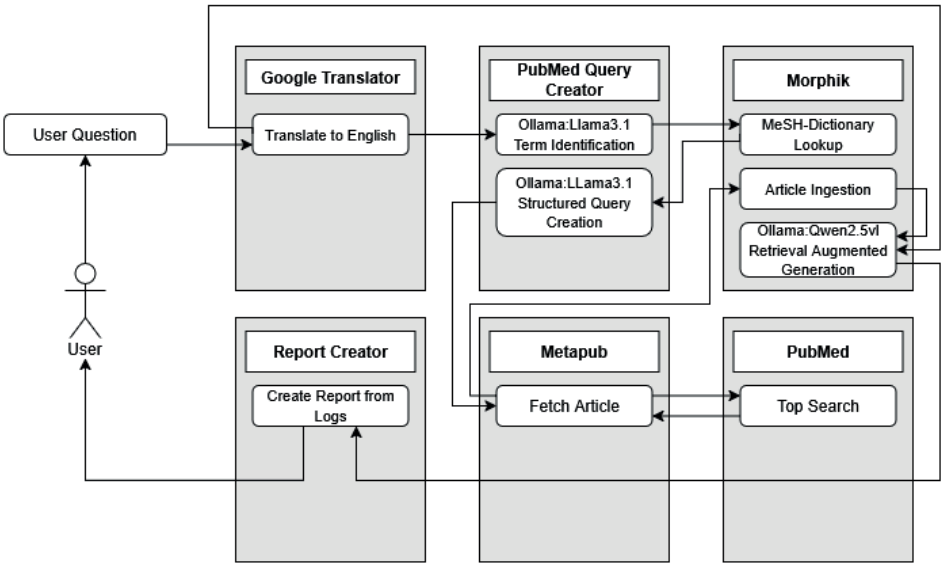


Figure 4.1: Simplified overview of how the prototype works

4.2 Limitations

This chapter describes some challenges encountered during the writing of this thesis.

4.2.1 Score conflict

To filter relevant chunks retrieved by Morphik during a query, the authors originally intended to use the `min_score` argument, which excludes chunks from the search results based on a threshold value between 0 and 1. This argument, however, assumes that the ingested chunks are normalized and that similarity is computed using a cosine similarity dot product.

When ColPali is enabled, this assumption no longer holds. The late interaction mechanism aggregates similarity scores across multiple patch embeddings according to equation: 2.1 which sums the maximum dot products between each query vector and its most similar document vector. Even if individual dot products are normalized to values within $[0, 1]$, their summation across all query vectors naturally produces scores greater than 1. Consequently, the `min_score` parameter becomes ineffective, as all retrieved chunks surpass the threshold. As a result, it was not possible to meaningfully filter between retrieved chunks when using ColPali-based embeddings.

4.2.2 Morphik

Multiple of the utilities in Morphik could not be implemented by the authors, among the features that did not work where the agent workflow where the LLM with reasoning capability uses a toolset to try and answer user questions.

Another feature that did not work was the knowledge graph creation where entries in the database could be visually represented based on entity relationship structure between them.

4.2.3 Hardware limitations

The workstation provided by Dr. Andersson was initially expected to be more than sufficient for the task. However, over time it became clear that the hardware represented roughly the minimum specifications required to operate a multimodal RAG–LLM system. This is largely due to the resource-intensive nature of Morphik, which alone requires approximately 8 GB of VRAM to load the ColPali model. In addition, the LLMs used in the thesis each occupied around 8 GB of VRAM when loaded. As a result, the remaining VRAM available for the context window and query processing was limited to at most 8 GB, constraining system performance.

4.2.4 Limitations of the LLMs

The LLMs employed in this thesis were general-purpose models. Although suitable for prototyping, they are not optimized for clinical or biomedical domains. Future work should therefore consider using specialized, domain-specific models—both text-based and multimodal—that have been primarily trained on medical literature. Such models may improve the relevance, accuracy, and reliability of generated search strategies in clinical contexts.

4.2.5 Metapub

Although Metapub provided a lot of help during the implementation of the prototype, there were some problems that were encountered. MetaPub's FindIt class includes publisher-specific strategies to locate direct links to a downloadable PDF file for different medical articles, while respecting publisher policies and restrictions, albeit being right most of the time, there were some instances where links to open access articles were not found. To illustrate this, the following example is presented:

A simple query: "Brivaracetam [Title/Abstract] english[filter]" was tested. Although a manual search on PubMed gave 562 results, only 30 pmids were retrieved with the PubMedFetcher and used in this test. Of these 30 articles, 10 links to PDF files were found. However, when looking at each returned error manually, it was found that eight other articles provided a free PDF file, but these were not found by the FindIt class.

Three articles returned the error "NO FORMAT: No handler found for journal". Two of these were actually titles that were found in another NCBI database, called *Bookshelf*. Bookshelf is a collection of biomedical books that can be searched directly from the Bookshelf database or from linked data in other NCBI databases. Therefore, while these entries appeared in the PubMed search results, it is likely that Metapub does not handle these cases. The remaining was published by BMJ and while supposed to be supported by Metapub, no PDF link was found.

Four articles returned the error "DENIED". Three of these articles were published by *Elsevier*, while the remaining one was published by *Wolters Kluwer*. According to the Metapub documentation, both publishers are expected to be supported.

One article returned the error "DANCE FUNCTION FAILED". This article was published by *Indian Pediatrics*, which appears to be a publisher not yet supported by Metapub.

To work around this, the article abstracts and metadata had to be used instead when a full-length article could not be found.

4.3 Reports

In this section, examples of a generated report will be presented along with the interview notes from a physician who was asked to review the reports according to factuality and usefulness.

4.3.1 Other Reports and Physician Evaluation

In addition to the primary report presented in this thesis, several other clinical queries were processed and compiled into full reports. Dr. Andersson and his colleagues were asked to review these reports and provide feedback regarding

their factual accuracy, relevance, and overall usefulness. However, the responses received by email were typically brief and focused mainly on suggestions for improving the layout or the presentation of search queries. This made it difficult for the thesis authors to draw meaningful conclusions from the written feedback alone.

Because of this, the thesis authors conducted a follow-up interview in which the physician were asked to revisit each chapter of the generated reports. This interview provided substantially more detailed and actionable feedback, forming the basis of the results presented in 4.4.

One example of an additional report concerned the question:

> "Can repeated infections be a side effect of brivaracetam? Adult patient with epilepsy who, after switching from levetiracetam to brivaracetam, experienced an increased frequency of infections. Since the switch 6 months ago, the patient has had 4–6 infections, including colds with cough and runny nose, and in 2–3 of these cases also fever."

The written feedback received for this report stated, for example, that "the search function has accurately identified the central message", but the remainder of the email focused largely on layout improvements and alternative ways to structure queries. Because such remarks did not address the factual or clinical relevance of the report's content, the subsequent interview became essential for the final evaluation.

4.4 Showcased report

In this section, we present the layout of the report template and provide an example of a query, its corresponding result, and an evaluation gathered from a physician during an interview.

4.4.1 Startpage

The start page seen in Figure 4.2 of each generated report provides the user with an overview of the report's content and structure.

4.4.2 Question

The question page seen in Figure 4.3 displays the user's query, which is machine-translated to English to ensure reproducibility and to improve processing by the LLM.

4.4.3 PubMed Query

The PubMed query page seen in Figure 4.4 displays the queries generated by the system and submitted to PubMed, along with the number of results returned by each search. In addition, the page provides a summary of newly identified

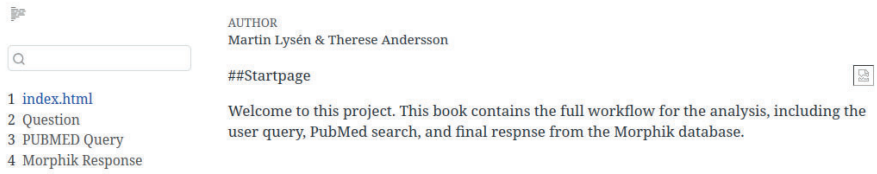


Figure 4.2: The start page of a generated report. Reports were produced as HTML books according to the formatting guidelines described in the Quarto documentation.

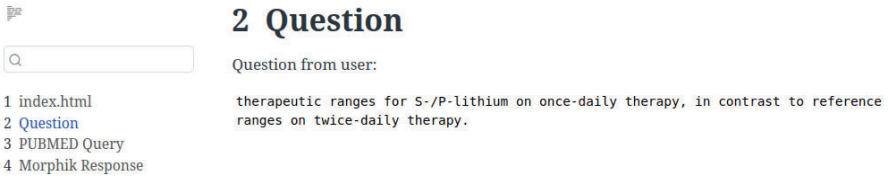


Figure 4.3: The question page shows the prompt provided by the user. All questions are machine-translated from the detected language into English to facilitate processing by the LLM.

articles for each query variation, giving an indication of whether alternative query formulations retrieved additional relevant literature.

3 PUBMED Query

Generated query and its variations, with # of results and # unique results per variant:

```
--- Queries with results ---

Query: ((Piperacillin[MeSH] OR Tazobactam[MeSH] OR Piperacillin-tazobactam))
Articles: 36194218, 38739397, 37078771, 39630396, 33106863, 33735856, 37558318, 391862
39186305, 39186290, 39186287, 30861061, 35106617, 38306577, 38549422, 38869238, 386069
36920195, 36545869, 39773457, 36919866, 38252887, 39826571, 36633812, 30968302, 313191
36745895, 37721612, 36593150, 37721616, 37721617, 37721615, 32500262, 32598886, 341124
29337796, 8277912, 33021531, 30175410, 29912125, 7655135, 39699210, 39319995, 3124068
39212919, 35352374, 11735679, 36257611, 36044050
----

Query: NO VALID QUERY
Articles: 36563828, 39283665, 35793299, 35038963, 39553448, 36375442, 40989749, 387135
36353685, 24327309, 27382148, 32804862, 25517353, 38497320, 40250848, 33813032, 348727
35198503, 32028894, 33080028, 38027579, 17391073, 35122126, 34020375, 37159238, 374541
34590274, 30157520, 18214614, 34548347, 40297352, 37050919, 29099475, 38253751, 113492
37027560, 36069977, 35045582, 34780354, 36347868, 39302778, 39172870, 21390321, 240644
35073845, 38422919, 37167032, 29868062, 20298525
----


--- Variants with zero results ---

- NO VALID QUERY
- NO VALID QUERY
----
```

This list detail the order in which the querys where sent to pubmed and how many results we got from that search we also list how many of those results where new when compared to previous searches

```
--- Diminishing returns summary ---

Diminishing returns summary:
(((Lithium[MeSH] OR Lithium Compounds)) AND ((Once-Daily Schedule[Title/Abstract] OR
Twice-Daily Schedule[Title/Abstract]))) | total:  2 | new unique:  2
NO VALID QUERY | total: 50 | new unique: 50
NO VALID QUERY | total:  0 | new unique:  0
NO VALID QUERY | total:  0 | new unique:  0
(((Therapeutic Range[MeSH] OR Reference Range)) AND ((Lithium[MeSH] OR Lithium
Compounds))) | total: 50 | new unique: 50
NO VALID QUERY | total: 50 | new unique:  0
NO VALID QUERY | total:  0 | new unique:  0
NO VALID QUERY | total:  0 | new unique:  0
NO VALID QUERY | total: 50 | new unique:  0
NO VALID QUERY | total:  0 | new unique:  0
NO VALID QUERY | total:  0 | new unique:  0
NO VALID QUERY | total: 50 | new unique:  0
NO VALID QUERY | total:  0 | new unique:  0
NO VALID QUERY | total:  0 | new unique:  0
(((Lithium Toxicity[MeSH] OR Kidney Diseases[MeSH] OR Nephrotoxicity Syndromes[MeSH]))
((Piperacillin[MeSH] OR Tazobactam[MeSH] OR Piperacillin-tazobactam))) | total: 50 | r
unique: 50
NO VALID QUERY | total: 50 | new unique:  0
NO VALID QUERY | total:  0 | new unique:  0
NO VALID QUERY | total:  0 | new unique:  0
((Lithium Toxicity[MeSH] OR Kidney Diseases[MeSH] OR Nephrotoxicity Syndromes[MeSH]))
total: 50 | new unique: 50
NO VALID QUERY | total: 50 | new unique:  0
NO VALID QUERY | total:  0 | new unique:  0
NO VALID QUERY | total:  0 | new unique:  0
((Piperacillin[MeSH] OR Tazobactam[MeSH] OR Piperacillin-tazobactam)) | total: 50 | ne
unique: 43
NO VALID QUERY | total: 50 | new unique:  0
NO VALID QUERY | total:  0 | new unique:  0
NO VALID QUERY | total:  0 | new unique:  0
```

Figure 4.4: PubMed queries generated by the system, including variations automatically produced by the LLM.

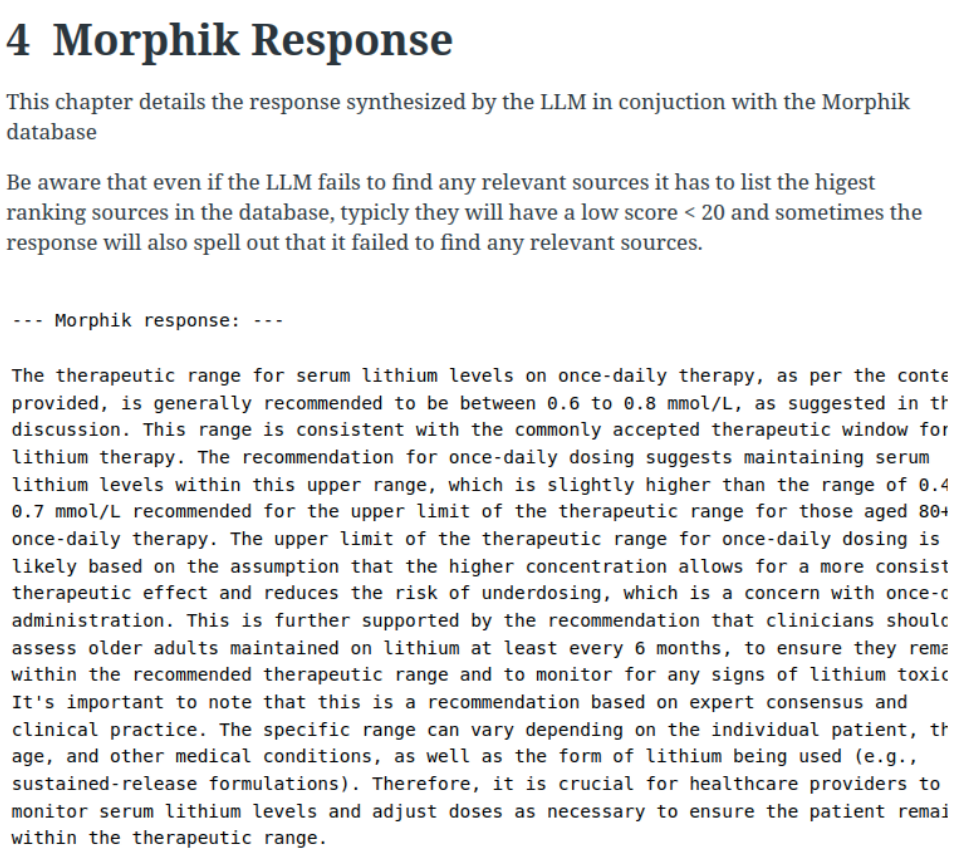
4.4.4 Morphik Response

The Morphik response page consists of the answer (as seen in Figure 4.5 generated by the LLM (the `response.completion`) and the chunks (as seen in Figure 4.6) retrieved from the database by Colpali to serve as context (the `response.sources`).

The sources were formatted to present the PMID of each article, the semantic score, the publication type (indicating the level of confidence a physician may reasonably place in the source), and the document type, where `text/plain` denotes an abstract and `application/pdf` denotes a full-text article.

Each source chunk was additionally accompanied by a Vancouver-style reference generated from its metadata.

Among the sources included in the model's final response, three were abstracts and one was a full-text article.



4 Morphik Response

This chapter details the response synthesized by the LLM in conjuction with the Morphik database

Be aware that even if the LLM fails to find any relevant sources it has to list the higest ranking sources in the database, typicly they will have a low score < 20 and sometimes the response will also spell out that it failed to find any relevant sources.

```
--- Morphik response: ---

The therapeutic range for serum lithium levels on once-daily therapy, as per the conte
provided, is generally recommended to be between 0.6 to 0.8 mmol/L, as suggested in th
discussion. This range is consistent with the commonly accepted therapeutic window for
lithium therapy. The recommendation for once-daily dosing suggests maintaining serum
lithium levels within this upper range, which is slightly higher than the range of 0.4
0.7 mmol/L recommended for the upper limit of the therapeutic range for those aged 80+
once-daily therapy. The upper limit of the therapeutic range for once-daily dosing is
likely based on the assumption that the higher concentration allows for a more consist
therapeutic effect and reduces the risk of underdosing, which is a concern with once-d
administration. This is further supported by the recommendation that clinicians should
assess older adults maintained on lithium at least every 6 months, to ensure they rema
within the recommended therapeutic range and to monitor for any signs of lithium toxic
It's important to note that this is a recommendation based on expert consensus and
clinical practice. The specific range can vary depending on the individual patient, th
age, and other medical conditions, as well as the form of lithium being used (e.g.,
sustained-release formulations). Therefore, it is crucial for healthcare providers to
monitor serum lithium levels and adjust doses as necessary to ensure the patient remai
within the therapeutic range.
```

Figure 4.5: The first half of the Morphik response page. It displays a warning informing the reader that the LLM will generate an answer based on the retrieved sources, and that it may occasionally fail to recognise when the provided context is insufficient.

```
PMID: 19374461
Score: 29.5390625
publication_types: {'D016428': 'Journal Article', 'D016454': 'Review'}
document_type: text/plain

Vancouver Reference:
Grandjean, EM, Aubry, JM. Lithium: updated human knowledge using an evidence-based
approach. Part II: Clinical pharmacology and therapeutic monitoring. CNS Drugs.
2009;23:331-49. doi:10.2165/00023210-200923040-00005.
----
PMID: 32936583
Score: 28.9296875
publication_types: {'D016428': 'Journal Article'}
document_type: text/plain

Vancouver Reference:
Fernandes, V, Al-Sukhni, M, Lawson, A, Chandler, G. Lithium Prescribing and Therapeuti
Drug Monitoring in Bipolar Disorder: A Survey of Current Practices and Perspectives. J
Psychiatr Pract. 2020;26:360-366. doi:10.1097/PRA.0000000000000493.
----
PMID: 30375703
Score: 28.4453125
publication_types: {'D016428': 'Journal Article'}
document_type: application/pdf

Vancouver Reference:
Shulman, KI, Almeida, OP, Herrmann, N, Schaffer, A, Strejilevich, SA, Paternoster, C,
Amodeo, S, Dols, A, Sajatovic, M. Delphi survey of maintenance lithium treatment in ol
adults with bipolar disorder: An ISBD task force report. Bipolar Disord. 2019;21:117-1
doi:10.1111/bdi.12714.
----
PMID: 20592663
Score: 29.390625
publication_types: {'D016428': 'Journal Article', 'D016449': 'Randomized Controlled Tr
document_type: text/plain

Vancouver Reference:
Singh, LK, Nizamie, SH, Akhtar, S, Praharaj, SK. Improving tolerability of lithium wit
once-daily dosing schedule. Am J Ther. 2011;18:288-91. doi:10.1097/MJT.0b013e3181d070c
```

Figure 4.6: This section presents the included source chunks along with their PMID, retrieval score, publication type, document type, and a Vancouver-style reference that can be copied for further use.

4.4.5 Physician evaluation

A physician evaluated the answer generated by the LLM by comparing it with a report written independently by a colleague who had been asked to address the same question. Importantly, this comparison case was not an existing Relis/Svelic report. The physician provided the prompt directly, without informing the authors that a colleague would later prepare an independent report on the same question. This deviated slightly from the intended evaluation protocol, which relied on previously published Relis/Svelic answers.

Some key differences noted were that the LLM emphasized therapeutic ranges for patients aged 80 and older, whereas the colleague focused on the expected effect of a single administration on lithium serum levels, and recommended lowering the dose to ensure that the peak value remained within known acceptable limits.

The physician remarked that this was a difficult question with no clear-cut answer, as there are likely no studies directly addressing the issue; instead, one must infer from studies that only tangentially relate to the core question.

The physician expressed the view that they would not want the generated answer to be part of a clinical version of the system, as they would be unwilling to sign off on an LLM-generated conclusion.

When asked about the relevance of the included sources, the physician stated that all provided sources were relevant.

When asked whether reading these sources was a good use of their time, the physician responded that all of the articles served as a good starting point and would not delay the preparation of their report.

As can be seen in figure 4.4 the generated queries from the question where:

1. (Piperacillin[MeSH] OR Tazobactam[MeSH] OR Piperacillin-Tazobactam)
2. ((Lithium[MeSH] OR Lithium Compounds) AND (Once-Daily Schedule [Title/Abstract] OR Twice-Daily Schedule[Title/Abstract]))
3. ((Therapeutic Range[MeSH] OR Reference Range) AND (Lithium[MeSH] OR Lithium Compounds))
4. ((Lithium Toxicity[MeSH] OR Kidney Diseases[MeSH] OR Nephrotoxicity Syndromes[MeSH]) AND (Piperacillin[MeSH] OR Tazobactam[MeSH] OR Piperacillin-Tazobactam))

The physician noted that Query 1 was irrelevant, as it concerns antibiotics that have no direct connection to lithium serum concentrations. In reality, this query originated from the prompt instructions used by the LLM to construct search queries and was likely included due to an internal malfunction.

Query 2 was considered relevant, although it produced only two results.

Query 3 was also judged relevant. However, the term Therapeutic Range is not a valid MeSH term and was therefore used incorrectly. Despite this error, the query still returned 50 results, likely because "Reference Range" is valid and captured the intended concept.

Query 4 was deemed irrelevant to the core clinical question, which concerned the validity of lithium administration. Although the physician acknowledged a potential indirect connection via toxicity considerations, the query was considered outside the primary scope, particularly as the intervention terms referred to antibiotics rather than lithium compounds. Additionally, the model generated a query that was a subset of query 4; this was disregarded as it did not contribute any new information.

4.5 Mail Follow-up

Based on questions raised during the follow-up interview, an additional email was sent to Dr. Andersson requesting a reevaluation of the report cited in Section 4.3.1.

4.5.1 General Feedback

Dr. Andersson noted several issues of phrasing and interpretation in the original answer. First, general clinical advice was included in a way that took up disproportionate space—for example recommendations regarding infection monitoring when initiating new medications. He also questioned the phrasing “only slightly significant”, asking whether statistical significance or effect size was intended, and highlighted instances of conversational language such as “you described”.

4.5.2 Search Strategy Comments

He suggested that the infection component of the search strategy could be specified further using additional [tiab]/[MeSH] terms (e.g., fever, common cold, respiratory tract infections). The search term “adverse effects” could also be refined to improve relevance.

4.5.3 Medical Assessment of the Sources

Dr. Andersson evaluated the four articles used as evidence in the LLM’s answer

> “Citalopram-associated SIADH” Not relevant. While the article aligns with the general topic of SSRI-associated hyponatraemia, it is not applicable to the brivaracetam question.

“Do antiepileptic drugs increase the risk of infectious diseases? A meta-analysis of placebo-controlled studies” Considered highly relevant, but with critical caveats. In this meta-analysis, brivaracetam and levetiracetam were combined in the statistical analysis due to similar pharmacodynamics (SV2A binding). The observed

significant increase in infection risk only appeared when the two drugs were analysed together, not separately. This limits interpretability, especially since the patient had been treated with levetiracetam prior to switching to brivaracetam. He also noted that the previous interpretation—"brivaracetam, when compared to other AEDs, was associated with a significantly increased risk of infection"—was incorrect. The comparison was made against placebo, not against other antiepileptic drugs, and the observed effect might be driven by levetiracetam rather than brivaracetam.

> "Investigating anti-inflammatory and immunomodulatory properties of brivaracetam and lacosamide in experimental autoimmune encephalomyelitis (EAE)" Not directly relevant. Although the study provides insight into inflammation-related mechanisms in human cells, its primary focus is not clinical outcomes in humans. Such mechanistic studies are generally not cited when addressing this type of clinical question.
>
> "Status epilepticus in the ICU" Not relevant, as it describes ICU management of status epilepticus and does not address infection risk or brivaracetam.

4.5.4 Summary

Only one of the four cited articles was genuinely relevant to the clinical question. Based on this, Dr. Andersson concluded that additional evidence needs to be identified to properly assess the association between brivaracetam and infection risk. He further stated that the previous conclusion regarding an increased infection risk was an over interpretation of the available evidence.

Chapter 5
Conclusion

This chapter summarises the main findings of the thesis and addresses the central question: Is it feasible to implement a partly automated article retrieval system suitable for supporting clinical experts? The chapter also reflects on the degree to which client requirements were met, the lessons learned, and opportunities for future development.

5.1 Evaluation of System Requirements

The thesis set out to implement a system capable of transforming free-text clinical questions into structured search queries, retrieving relevant articles, and providing contextual evidence to support clinical decision-making. The results indicate that the current prototype can automate parts of the workflow, however it still requires an expert in the loop to validate the created queries. The following subsections summarise the extent to which each client requirement was achieved.

5.1.1 Automated Query Transformation

The system successfully converted free-text clinical questions into MeSH-based search strategies reflecting the clinical workflow. While the transformation worked reliably in many cases, LLM outputs exhibited variability, making expert validation necessary. Human oversight remains essential to ensure accuracy and relevance.

5.1.2 Article Retrieval and Semi-Automation

Automated retrieval from PubMed was feasible using the Metapub library. Occasional failures occurred, primarily for articles outside the open-access domain, which was acceptable within the scope of this thesis.

5.1.3 Multimodal RAG Pipeline

The system achieved full multimodal retrieval using Morphik's RAG framework. The ColPali embedding model enabled processing of figures and tables—originally

a major technical challenge—and incorporated them as contextual evidence. Textual and visual content were therefore successfully integrated as source chunks during retrieval.

5.1.4 Reproducible Logging and Reporting

Structured log files captured queries, retrieved sources, and system responses. These logs were then used to generate the reports. The main challenge was identifying which elements were essential to users—a topic suitable for future refinement.

5.1.5 Python-Based Implementation

Core system components were implemented in Python. The exception was Morphik-specific frontend elements, which were developed in React for consistency with the user interface.

5.1.6 Unmet or Partially Achieved Requirements

The following requirements were not fully realised:

- **Minimal interface-sources:** Users desired the ability to export references directly to Zotero. The system presents references in Vancouver format, but no automated export feature was implemented.
- **Minimal interface-export to internal systems:** Integration with Relis was not completed due to late receipt of API documentation.
- **MeSH Tag Editing System:** The interactive system for adding, removing, or weighting MeSH terms during query refinement was not implemented. With the MeSH dictionary now ingested into the database, this functionality could be developed in future iterations.

5.2 Reflection on Research Questions

In this section, the research questions posed in the problem statement 1.4 are addressed, their degree of successful resolution is evaluated, and suggestions for future work are presented.

5.2.1 #1 Current workflow for physicians

The workflow analysis enabled the development of a search strategy in PubMed. Feedback highlighted that the current system may be restrictive for questions not fitting the standard “condition AND intervention” format. Future development should allow general free-text queries and consider usability studies to identify alternative search strategies.

5.2.2 #2 Converting Free-Text Queries into Structured Searches

LLMs can assist in transforming free-text questions into structured search queries. Outputs require expert validation, and improvements are possible through better prompt design and algorithmic checks to reduce errors and unnecessary use of context.

5.2.3 #3 Evaluating System Responses

The authors found that structured interviews with clinicians are more effective than brief written feedback for assessing report quality and relevance. Future evaluations should allocate sufficient time for interviews, ensuring focus on the core questions of factuality and relevance for the posed question.

5.2.4 #4 LLM Selection for the Task

This thesis did not conduct an exhaustive comparison of LLMs mainly due to time constraints. Models such as qwen2.5vl:7b and llama3.1:8b were sufficient for specific subtasks, but a wide range of alternative models exists and new ones presented regularly. Additionally there are models released that state to be developed specifically for the biomedical research domain, consequently no definitive conclusions could be drawn regarding the optimal LLM for the overall system.

5.3 Future development

In this section some future considerations that arose are discussed.

5.3.1 Retrieval from Additional Data Sources

The reliance on PubMed alone highlighted the need to consider alternative sources for future development. Systems could integrate European databases such as Embase or open-access platforms like Semantic Scholar, which provide bulk retrieval APIs similar to E-utilities, albeit without the convenience provided by Metapub.

5.3.2 Reducing Vendor Dependence and Improving RAG Architecture

The system relied heavily on Morphik for database management, API handling, and UI components, introducing technical constraints and potential vendor lock-in. ColPali embeddings improved multimodal processing, enhancing retrieval quality. Future development should phase out Morphik in favor of a standalone, domain-specific RAG architecture to improve flexibility, maintainability, and long-term control.

5.4 Ethical Considerations

Introducing AI-based systems into clinical workflows requires careful ethical consideration. Key issues include patient trust, transparency, and responsibility for

clinical decisions. Clear guidelines must be established regarding when and how it should be disclosed that an AI system contributed to a patient's treatment plan, whether for individual patients or for broader clinical recommendations.

Additionally, based on the physician evaluation, the authors recommend that LLM-generated conclusions should not be included in clinical reports. The physicians explicitly stated they would be unwilling to endorse or "sign off" on AI-generated conclusions. Including such completions could introduce bias, potentially influencing clinical judgment and reducing trust in the system.

Overall, further development of systems of this type should carefully weigh potential clinical benefits against ethical, societal, and patient-safety risks.

Bibliography

[1] Joshua Ofoeda, Richard Boateng, and John Effah. "Application Programming Interface (API) Research: A Review of the Past to Inform the Future". In: Int. J. Enterprise Inf. Syst. (IJEIS) 15.3 (2019), pp. 76–95. DOI: 10.4018/IJEIS.2019070105. URL: https://www.igi-global.com/article/application-programming-interface-api-research/232166.

[2] Hesam Sheikh Hessani. LLM Embeddings Explained: A Visual and Intuitive Guide. [accessed: 2025-11-20]. 2025. URL: https://huggingface.co/spaces/hesamation/primer-llm-embedding%20section=what_makes_a_good_embedding%3F.

[3] Prathusha Kameswara Sarma, Yingyu Liang, and William A. Sethares. "Domain Adapted Word Embeddings for Improved Sentiment Classification". In: Annual Meeting of the Association for Computational Linguistics. 2018. URL: https://api.semanticscholar.org/CorpusID:44143625.

[4] Josh Schneider and Ian Smalley. CPU vs. GPU for machine learning. [accessed: 2025-09-02]. URL: https://www.ibm.com/think/topics/cpu-vs-gpu-machine-learning.

[5] Harrison Chase. LangChain. [cited 2025-09-01]. Oct. 2022. URL: https://github.com/langchain-ai/langchain.

[6] National Library of Medicine. Medical Subject Headings (MeSH)–Home Page. [accessed: 2025-11-21. URL: https://www.nlm.nih.gov/mesh/meshhome.html.

[7] IBM. "What is OCR?" In: (2025). accessed: 2025-11-19. URL: https://www.ibm.com/think/topics/optical-character-recognition.

[8] Ollama. Ollama. [cited 2025-09-02]. Sept. 2025. URL: https://github.com/ollama/ollama.

[9] NCBI. NIH Public Access and PMC – PubMed Central. [accessed: 2025-11-21]. URL: https://pmc.ncbi.nlm.nih.gov/about/public-access-info/.

[10] J.J. Allaire et al. Quarto. Version 1.7. Apr. 2025. DOI: 10.5281/zenodo.5960048. URL: https://github.com/quarto-dev/quarto-cli.

[11] SVELIC. Om RELIS. [accessed: 2025-11-21]. URL: https://svelic.se/om-relis/.

[12] M. Alam et al. "A Review on Large Language Models: Architectures, Applications, Taxonomies, Open Issues and Challenges". In: IEEE Access (2024). [cited 2025-09-02]. URL: https://ieeexplore.ieee.org/document/10433480/.

[13] D. Bergmann. "Context window". In: (2025). [accessed: 2025-10-17]. URL: https://www.ibm.com/think/topics/context-window.

[14] Lyx. Context Kills VRAM: How to Run LLMs on consumer GPUs. [cited 2025-09-02]. May 2025. URL: https://medium.com/@lyx_62906/context-kills-vram-how-to-run-llms-on-consumer-gpus-a785e8035632.

[15] Max Peeperkorn et al. "Is Temperature the Creativity Parameter of Large Language Models?" In: (2024). arXiv: 2405.00492 [cs.CL]. URL: https://arxiv.org/abs/2405.00492.

[16] I Belcic and C Stryker. "LLM parameters, defined". In: (2025). [accessed: 2025-11-13]. URL: https://www.ibm.com/think/topics/llm-parameters.

[17] I Belcic and C Stryker. "What are model parameters?" In: (2025). [accessed: 2025-11-13]. URL: https://www.ibm.com/think/topics/model-parameters.

[18] IBM. "Foundation model parameters: decoding and stopping criteria". In: (2025). accessed: 2025-11-19. URL: https://www.ibm.com/docs/en/watsonx/saas?topic=prompts-model-parameters-prompting.

[19] Joshua Noble. What is LLM temperature. [accessed: 2025-11-21]. URL: https://www.ibm.com/think/topics/llm-temperature.

[20] NVIDIA. What is a multimodal LLM (MLLM)? [accessed: 2025-10-16]. URL: https://www.nvidia.com/en-us/glossary/multimodal-large-language-models/.

[21] Shuai Bai et al. "Qwen2.5-VL Technical Report". In: (2025). arXiv: 2502.13923 [cs.CV]. URL: https://arxiv.org/abs/2502.13923.

[22] Zach Nussbaum et al. "Nomic Embed: Training a Reproducible Long Context Text Embedder". In: (2025). arXiv: 2402.01613 [cs.CL]. URL: https://arxiv.org/abs/2402.01613.

[23] Grattafiori et al. "The Llama 3 Herd of Models". In: (2024). arXiv: 2407.21783 [cs.AI]. URL: https://arxiv.org/abs/2407.21783.

[24] Yunfan Gao et al. "Retrieval-Augmented Generation for Large Language Models: A Survey". In: (2024). arXiv: 2312.10997 [cs.CL]. URL: https://arxiv.org/abs/2312.10997.

[25] NVIDIA. What Is Retrieval-Augmented Generation (RAG)? [accessed: 2025-10-31]. URL: https://www.nvidia.com/en-us/glossary/retrieval-augmented-generation/.

[26] Mohammad Mahdi Abootorabi et al. "Ask in Any Modality: A Comprehensive Survey on Multimodal Retrieval-Augmented Generation". In: (2025). arXiv: 2502.08826 [cs.CL]. URL: https://arxiv.org/abs/2502.08826.

[27] nthmost et al. metapub: Python toolkit for NCBI metadata and PubMed text mining. Version 0.6.4. [cited 2025-09-22]. Aug. 2024. URL: https://github.com/metapub/metapub.

[28] morphik-org. morphik-core. [cited 2025-10-01]. Oct. 2025. URL: https://github.com/morphik-org/morphik-core.

[29] Manuel Faysse et al. "ColPali: Efficient Document Retrieval with Vision Language Models". In: (2025). arXiv: 2407.01449 [cs.IR]. URL: https://arxiv.org/abs/2407.01449.

[30] morphik-org. morphik-docs. [cited 2025-10-31]. Oct. 2025. URL: https://www.morphik.ai/docs/self-hosting.

[31] Jobit Varughese. What is a multimodal LLM (MLLM)? [accessed: 2025-10-16]. URL: https://www.ibm.com/think/topics/multimodal-llm.

Appendix A

Screenshots of Prototype.

The following screenshots present the web-based user interface for the prototype system.

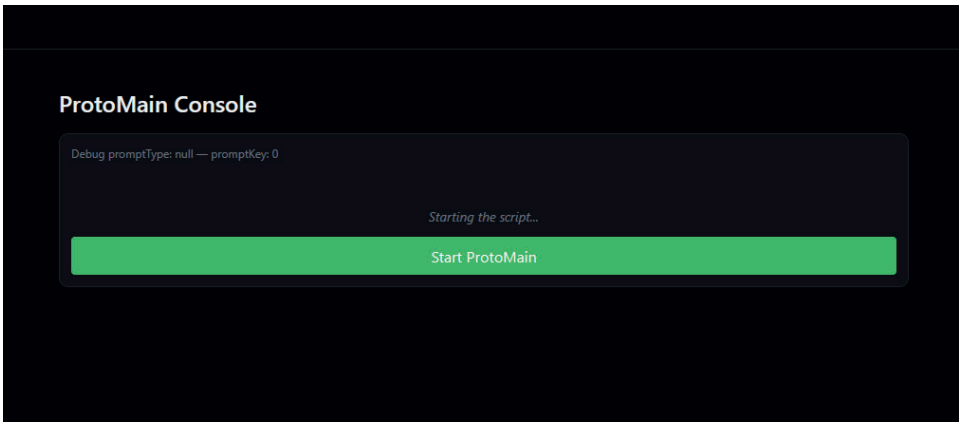


Figure A.1: Start button to launch the protomain script in the web UI.

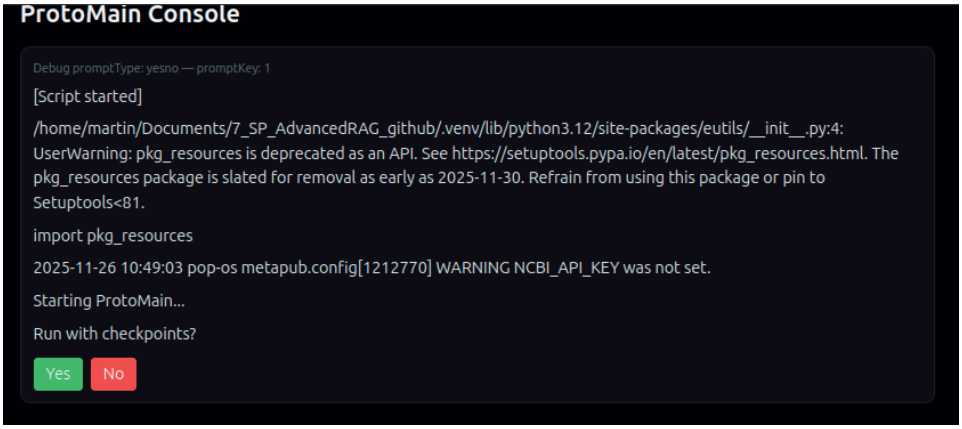


Figure A.2: Choice to run with or without checkpoint facilitated with Yes No prompt to the user to help them navigate trough the program flow.

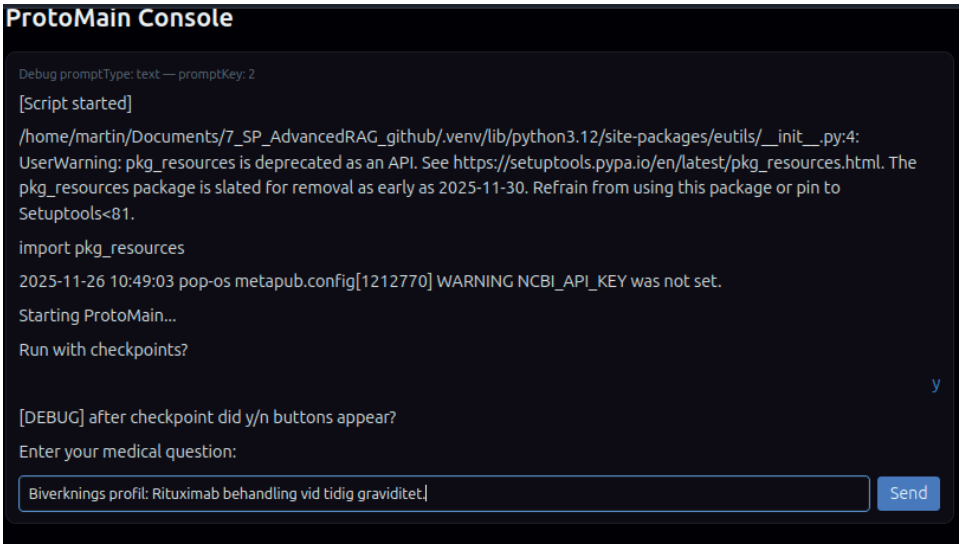


Figure A.3: Text box with prefilled answer to show user that freetext enquires are viable.

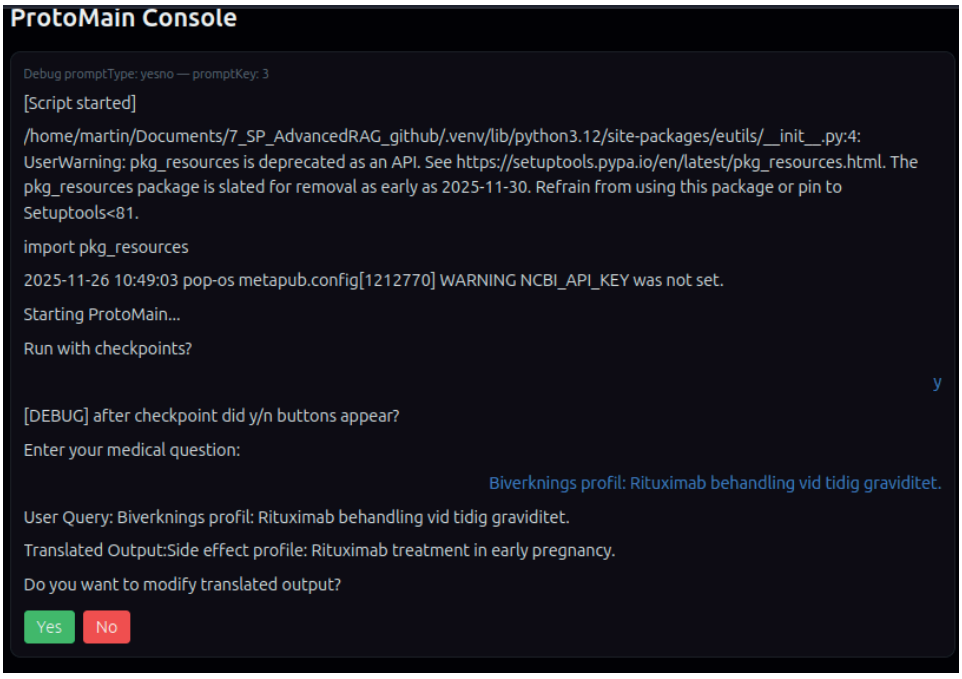


Figure A.4: Output of machine translated query to English and prompt to validate the the meaning was preserved.

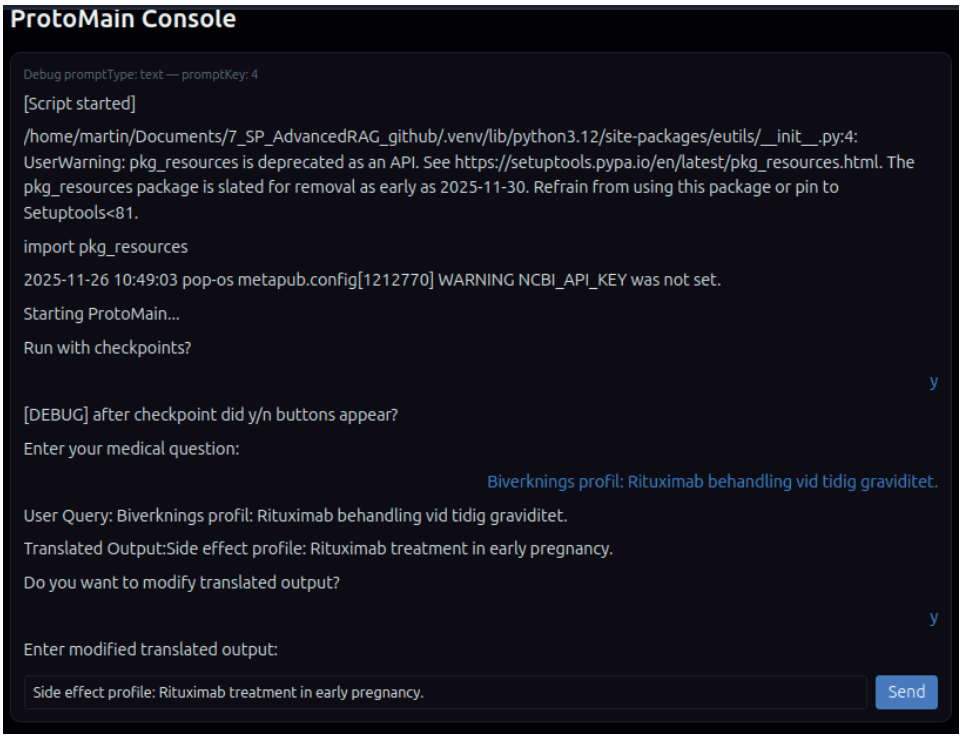


Figure A.5: The translated question is prefilled in the input box for faster user editing.

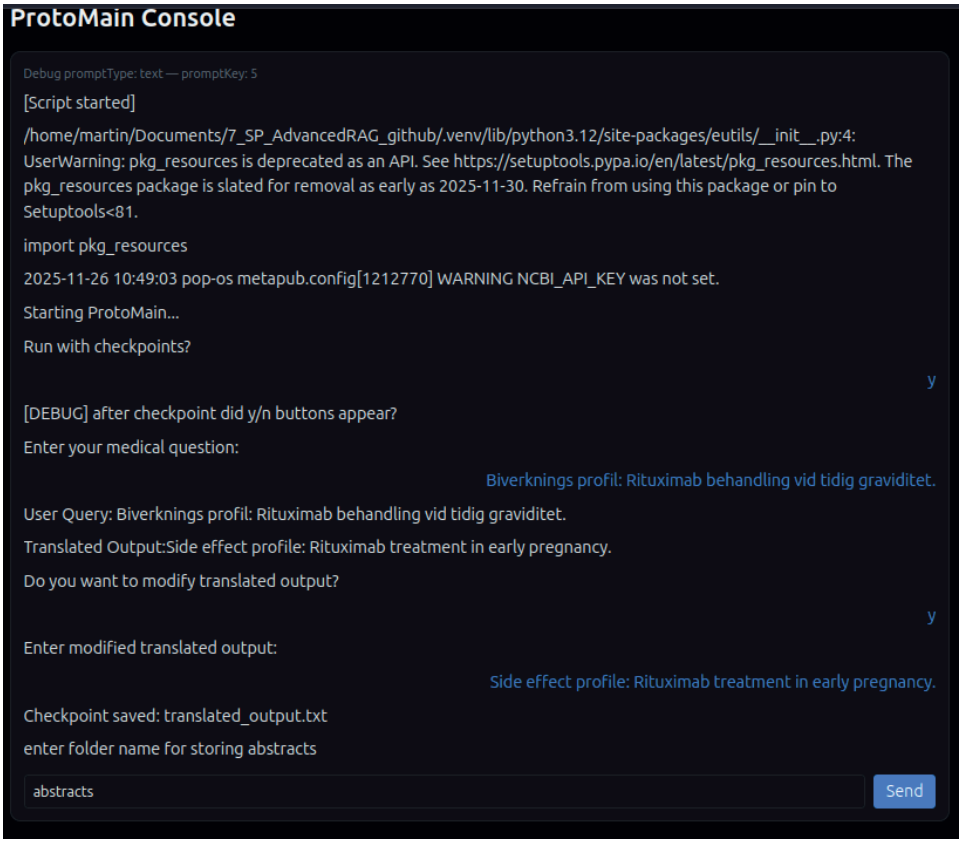


Figure A.6: Prompt to name the project of the query which will be the folder name and the title of the generated report, default option to just store the results in general purpose folder abstracts.

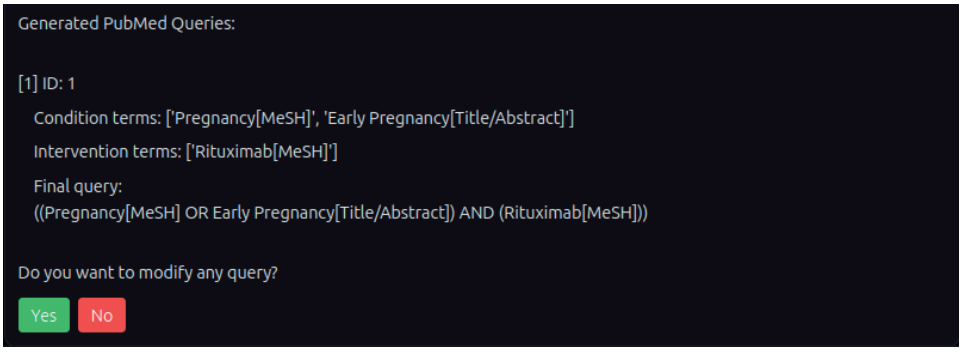


Figure A.7: List of generated queries, presented with the condition and intervention terms the LLM identified in the question and the final query to send to PubMed, the user can opt to modify any of the queries in the list.

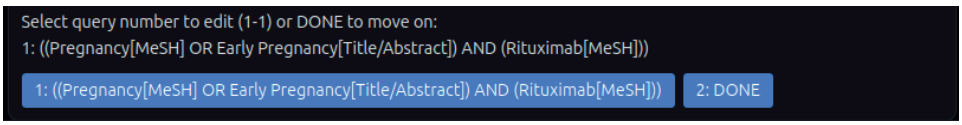


Figure A.8: Button based prompt for each query in the list, DONE button escapes the prompts and continues the pipeline.

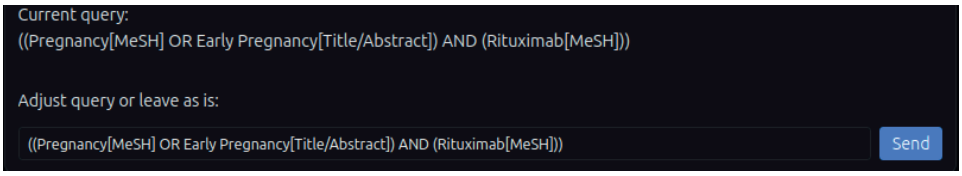


Figure A.9: Prefilled prompt of query chosen to edit.

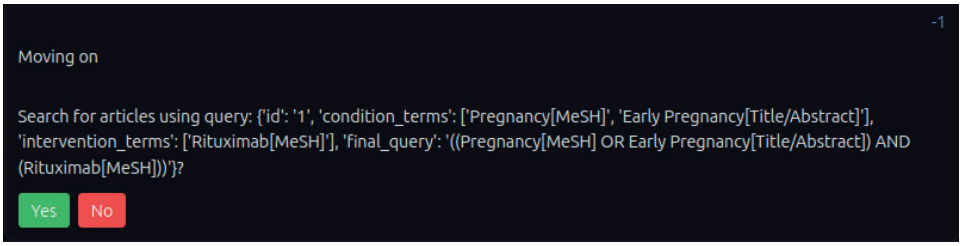


Figure A.10: Prompt to search using displayed query or skip to next in list.

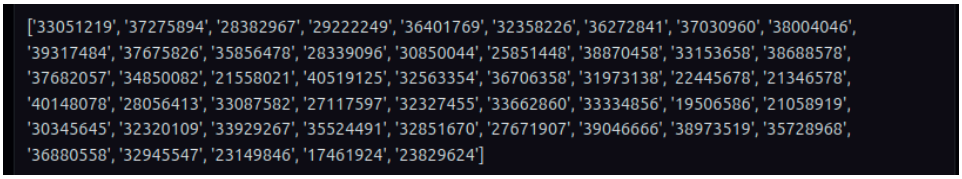


Figure A.11: Decpits PMIDS of every article found using the selected query.

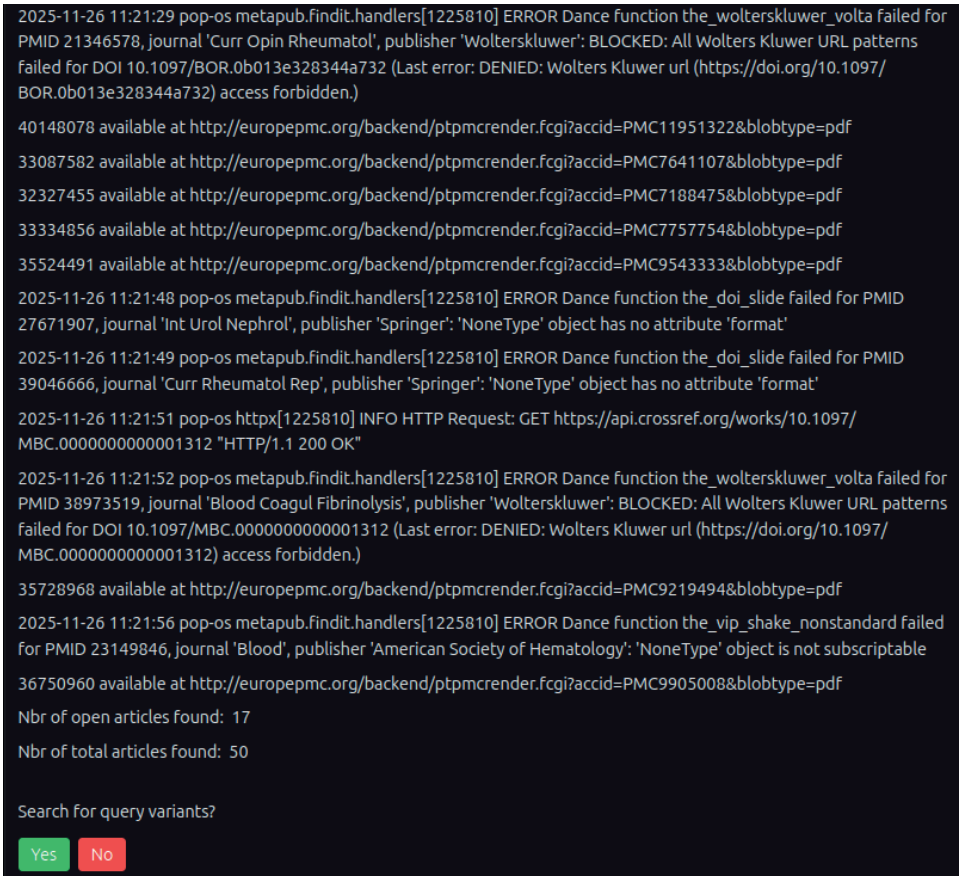


Figure A.12: Streaming list of accessibility check for every found article and prompt on whether to search using expanded query and fallback methods.

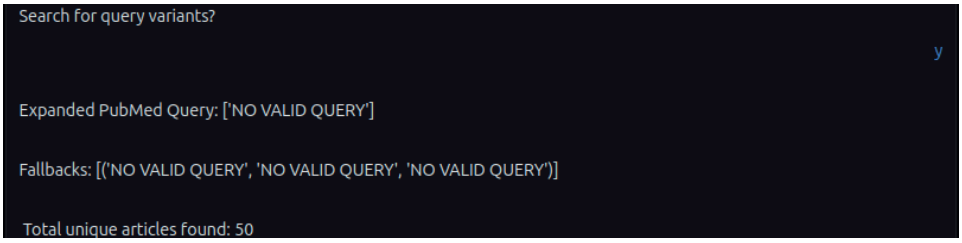


Figure A.13: Result from expanded query and fallback methods, typically fails if query is not of the exact format condition AND intervention.

Figure A.14: Streaming list of ingestion progression for every found article.

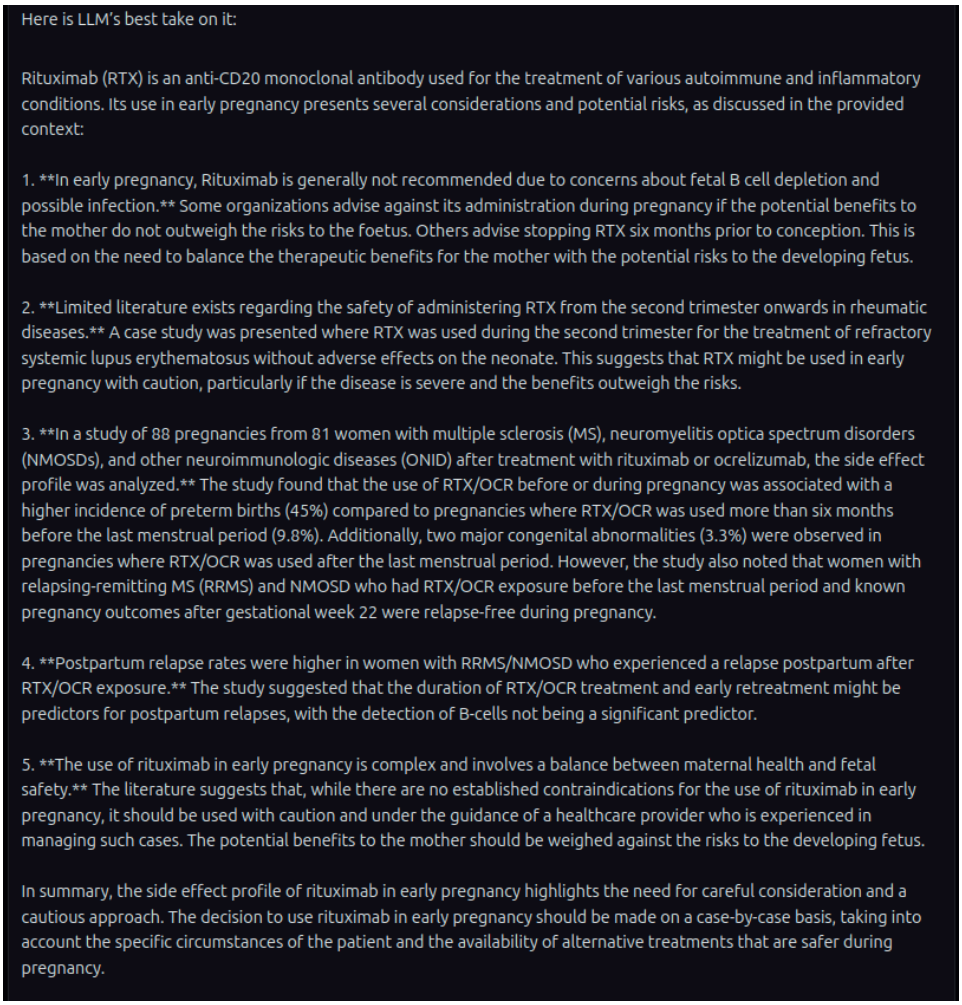


Figure A.15: Answer from LLM after ingestion and choosing chunks with highest semantic score as sources.

Figure A.16: One of four source chunks used in LLM answer, source chunks are formated and presented in easier to read form in the generated report.

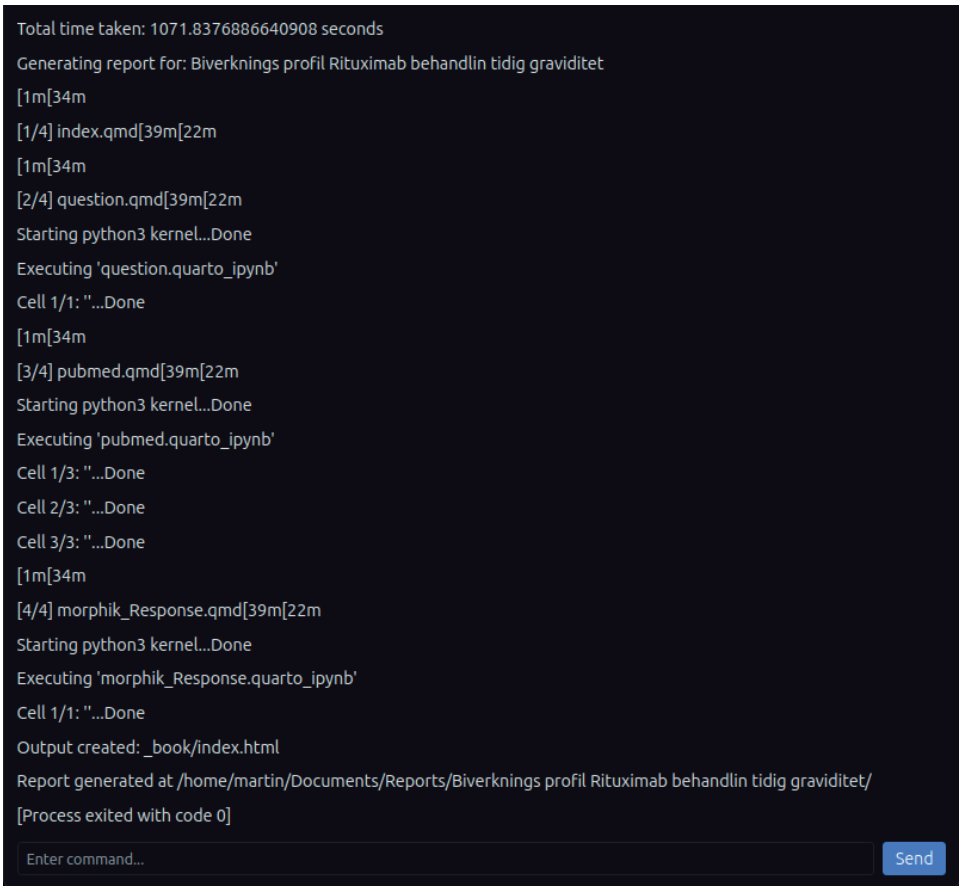


Figure A.17: Final step of protomain pipeline, displays runtime in seconds and progress of report creation as well as location of created report on the Computer.

Printed by Tryckeriet i E-huset, Lund 2025

LUND
UNIVERSITY

Series of Bachelor´s theses
Department of Electrical and Information Technology
LU/LTH-EIT 2026-1107
http://www.eit.lth.se