Hes·so

MASTER OF SCIENCE IN ENGINEERING

Artificial Intelligence for Automated Literature Review Generation

Master of Science HES-SO in Engineering

Profil/Orientation Data science

Date of publication 18.08.2025

Author: Abdi VURAL

Under the supervision of *Prof. Elena Mugellini and Prof. Leonardo Angelini*

In collaboration with the institute HumanTech of HEIA-Fr

Table of Contents

Declaration of authenticity

I hereby affirm that this assignment is my own written work and that I have used no other sources or aids other than those indicated. All passages that have been quoted from publications or paraphrased from these sources are indicated as such. Additionally, any assistance from artificial intelligence tools has been appropriately acknowledged and documented.

During this research, I have made use of AI-based tools such as Napkin (for generating diagrams) as well as language models like ChatGPT, Claude, and Grok to support this project.

Name: Abdi VURAL

Date: 18/08/2025

Acknowledgments

I would like to express my sincere gratitude to Prof. Elena Mugellini and Prof. Leonardo Angelini for their exceptional guidance, encouragement, and continuous support throughout this project. Their expertise and insightful feedback were invaluable at every stage of this thesis.

This work was carried out in collaboration with the HumanTech Institute of HEIA-FR (University of Applied Sciences of Western Switzerland), whose innovative and interdisciplinary environment provided an ideal setting for research and learning.

I would also like to thank the Nursing School of Lisbon (ESEL) for their participation and shared vision in this collaborative effort. Their contributions played a key role in shaping the healthcare application context of the system developed.

Abstract

With millions of scientific papers published each year, researchers face an overwhelming challenge staying current with the latest developments. Traditional systematic literature review (SLR) procedures are extremely time-consuming, often taking months or years to complete. This thesis addresses this challenge by developing an AI-based system that automates key stages of systematic literature reviews. The system focuses on conversational e-coaches in healthcare and integrates advanced natural language processing methods to extract, screen, and analyze scientific papers.

The approach combines automatic metadata extraction with GROBID, eligibility screening inspired by PRISMA guidelines, and the eCCo framework for identifying the characteristics of conversational health agents. It supports both cloud-based and local language models, using adaptive prompt strategies to optimize performance across different architectures. Scientific articles in PDF format are transformed into structured data through semantic chunking, vector embeddings, and contextual retrieval aligned with the eCCo taxonomy.

Keywords: Artificial Intelligence, Systematic Literature Review, Natural Language Processing, Conversational Agents, Large Language Models , PRISMA, GROBID, ChromaDB, LangChain, Prompt Engineering, Metadata Extraction.

1 Introduction

One of the most important milestones in the biological, cultural, social, and scientific evolution of humanity has been the transmission of existing and newly generated knowledge. From cave paintings to the hieroglyphs in pyramids, from the Code of Hammurabi to the Magna Carta, from Newton's calculus to Turing's machine, humanity has continually transferred knowledge using the simplest to the most advanced methods. This transmission of knowledge allowed us not only to preserve what was known, but also to build upon it and generate new insights.

The key concept here is making the produced knowledge permanent. Ancient civilizations achieved this through papyrus scrolls or cuneiform tablets, leaving us a valuable legacy. In today's scientific world, we have developed various methods to share and preserve knowledge including theses, conference proceedings, and academic research papers that follow a standardized structure.

As scientific progress accelerates, the number of published academic articles has been increasing continuously (see Figure 1). Conducting research within a specific field among millions of publications requires a rigorous methodology. At this point, Systematic Literature Reviews (SLRs) help us to collect, identify, and analyze existing studies related to a specific research topic.

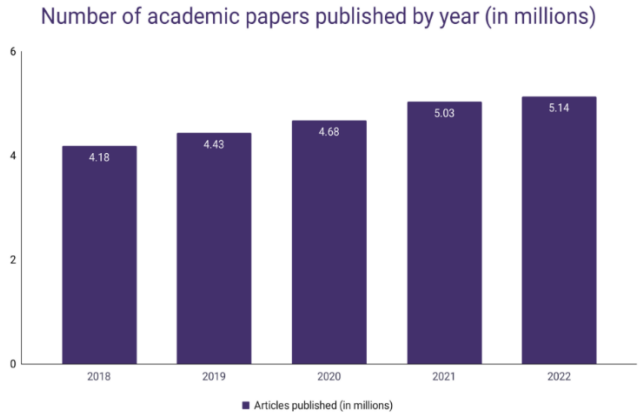
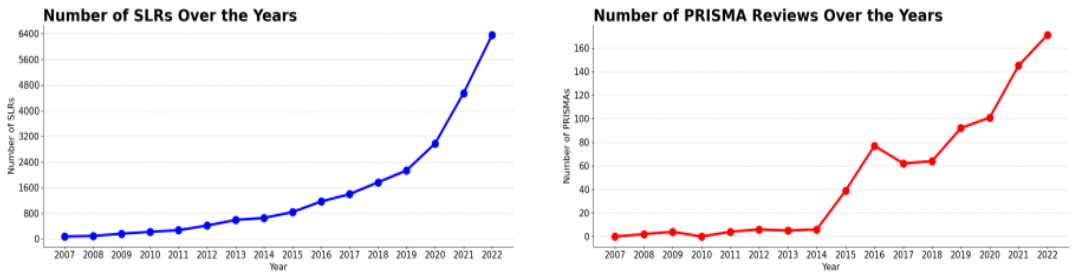


Figure 1 – Number of academic papers published[1] each year from 2018 to 2022 (in millions).

One of the most commonly applied methods for conducting a Systematic Literature Review (SLR) is the PRISMA guideline (Preferred Reporting Items for Systematic Reviews and Meta-Analyses). Its most recent 2020[2] update provides researchers with a structured framework for reporting the review process comprehensively, from search strategy to data synthesis. The continuous increase in published SLRs and PRISMA-guided reviews (Figures 2a and 2b) reflects their established role in academic research and emphasizes the necessity for more efficient and scalable review methodologies.

While PRISMA has greatly improved the quality and reporting consistency of systematic reviews, applying it thoroughly can be resource intensive. Conducting a full SLR according to PRISMA involves exhaustive literature searches, strict inclusion criteria, meticulous data extraction, and careful synthesis all of which require substantial time, effort, and access to resources[3]. This can pose challenges for researchers with limited funding or time constraints[4].



(a) (b)

Figure 2 - Total number of SLRs (a) and PRISMA (b) studies published by year, according to Google Scholar[3].

Moreover, despite its emphasis on transparency, PRISMA does not directly assess the scientific quality of a review. It primarily ensures that reporting is complete and standardized, but following the checklist does not, in itself, guarantee that the review is methodologically sound or free of bias. For example, publication bias and selective study inclusion remain common risks in systematic reviews, even under PRISMA guidance[3], [5].

As research methodologies evolve, especially with the emergence of AI, machine learning, and automated synthesis techniques. There is a growing need to adapt PRISMA to remain compatible with new tools and workflows. Integrating such technologies could make the review process more efficient while maintaining rigorous standards[3].

1.2 Conventional tools

Several digital tools have been developed specifically to support and simplify the process of conducting a Systematic Literature Review (SLR). Well-known examples include Covidence, Rayyan, and DistillerSR. These platforms help researchers work together more easily and follow a clear and organized workflow that aligns with PRISMA guidelines.

The main purpose of these tools is to improve collaboration between reviewers by organizing tasks, managing article screening, and avoiding repetitive work. They make the review process more consistent and easier to follow. Some tools also offer extra features such as keyword extraction, citation analysis, and filtering based on meta-information like publication date or author name. Table 1[6] compares four commonly used SLR tools, showing their main features.

Logiciel	*Accessibility*	*Text embedding approach*	*Database*	*Use meta information*	*Unsupervised corpus organisation*	*Examples of applications*
Covidence	Purchasable license	No	Provided by user	Yes	No	(4)
DistilleR	Purchasable license	No	Provided by user	Yes	No	(6)
Rayyan	Open Source	No	Provided by user	Yes	No	(14)
BiBot	Experimental	Yes	Synchronised with the NCBI	Yes	Yes	(13)

Table 1- Comparison of four informatic tools currently used to perform systematic literature reviews[6]

1.3 Research questions

Traditionally, a process of SLR follows a rigorous methodology that includes developing a protocol, identifying relevant studies through database searches, screening titles and abstracts, assessing full texts for eligibility, and extracting and synthesizing data. This approach consumes a lot of time and resources. Researchers may spend several months, or even years, completing a single systematic review, largely due to the exponential growth in the number of published scientific articles.

To address these difficulties, researchers often use tools such as Covidence, Rayyan, and DistillerSR. However, these tools do not fundamentally automate the cognitive tasks of content understanding, screening, or data extraction. Human reviewers are still responsible for reading and interpreting scientific texts, so the process remains slow and labor-intensive.

At this point, we can address these challenges by using large language models (LLMs). LLMs are advanced artificial intelligence systems with the capacity to understand context and process natural language. By integrating LLMs into the SLRs process, researchers can mitigate the limitations of traditional methods and tools. In this way, not only can they save time and resources, but they can also focus on more critical aspects of the review, thereby increasing overall efficiency.

This MSc thesis aims to explore how LLMs can be effectively integrated into SLR workflows, with a particular focus on enhancing the quality, speed, and reliability of evidence synthesis in health-related research. The goal is to develop an automated pipeline capable of extracting and structuring relevant characteristics based on the ECCO framework, which includes dimensions such as intervention design, target, conversational features, and agent characteristics.

This project is devoted to answering several central questions:

1. Can LLMs be effectively integrated into SLR workflows?
2. If so, which concrete integration strategies are practically executable?
3. How can the ECCO framework be operationalized in an automated pipeline using LLMs to extract meaningful characteristics from scientific literature?
4. What are the potential limitations or risks (e.g., bias, hallucination, lack of transparency) associated with using LLMs in evidence synthesis?
5. Can domain-specific fine-tuning or prompt engineering improve the accuracy and relevance of information extracted by LLMs?

2

Background and Related Work

This chapter provides a broad overview of the research domain, highlighting key concepts, technologies, and studies that form the foundation for our work. It reviews existing research on conversational agents in healthcare, the eCCo framework, and automated literature review processes, offering a general view rather than a detailed description of our specific approach. The chapter also discusses core technologies such as GROBID for scholarly PDF processing, vector search, and retrieval-augmented generation (RAG), alongside relevant methodologies reported in the literature. Finally, it summarizes how these components connect in the broader landscape of related research.

Chapter's content

2.1 Conversational Agents in Healthcare

Conversational agents (CAs) are computer-based applications designed to engage with users in natural language, either through written dialogue or spoken interaction. In the healthcare sector, they are increasingly used to assist patients in self-management, provide easier access to medical information, and help reduce the workload of healthcare professionals. These agents can range from basic, rule-driven chatbots to advanced virtual assistants powered by LLMs. They may appear as physical devices, such as robotic companions, or as virtual interfaces in mobile applications and websites.

The adoption of CAs in healthcare has grown in domains such as mental health support, chronic disease monitoring, and lifestyle interventions. A review published in *JAMIA* reported findings from 17 studies covering 14 healthcare conversational agents, most of which were designed for targeted tasks and capable of processing open-ended input, often through spoken language [7].

Different methods are used to manage conversations. Most agents rely on rule-based techniques such as frame-based or finite-state dialogue management, where the system controls the flow of interaction. Only a small number use more flexible or adaptive approaches. Evaluations vary widely from one study to another, including technical measures (like speech recognition accuracy), task performance, user satisfaction, and health outcomes. Unfortunately, there is no common standard for evaluation, which makes it hard to compare results across systems.

This diversity in design and evaluation underlines the importance of using a structured framework like eCCo. Such a framework can help researchers describe conversational agents more clearly and compare them in a consistent and systematic way.

2.2 The eCCo Framework

The **eCCo (Harnessing the Power of Conversational e-Coaches for Health and Well-being Through Swiss Portuguese Collaboration)** project represents a groundbreaking international research initiative that addresses one of the most pressing challenges in digital health research: the fragmentation of knowledge and methodologies in conversational agent development for health interventions. Established through a formal collaboration between the **University of Applied Sciences Western Switzerland (HES-SO) - HumanTech Institute** and the **Nursing School of Lisbon (ESEL)**, the eCCo initiative embodies a strategic approach to overcoming the isolated nature of conversational e-coaching research[8], [9].

The eCCo framework was developed to provide a structured and comprehensive way to describe and analyze conversational agents in healthcare contexts. It addresses the challenge of inconsistent reporting and fragmented descriptions found in the literature by defining a common vocabulary and classification scheme.

The framework organizes key characteristics of conversational agents into distinct categories, enabling consistent documentation and analysis. These categories include:

- **Metadata**: Basic bibliographic information such as article title, publication year, authors, and country of origin.
- **Study Design**: The aim of the study, research methodology, and study type (e.g., RCT, usability study)
- **System Interactions**: The modalities used for input (e.g., voice, text, sensor data) and output (e.g., text, speech, visuals), as well as interaction flow and dialogue structure.
- **Delivery Channels**: Platforms through which the agent is accessed, such as mobile apps, websites, robots, or standalone devices.

- **Target Interventions**: The health-related behaviors or conditions targeted by the agent (e.g., physical activity, nutrition, mental health).
- **Action Interventions**: The types of interventions delivered by the agent, such as education, behaviour change support, or assessment.
- **Advanced Features**: Capabilities like proactivity, emotion recognition, personalization, or adaptive learning.
- **Embodiments**: The form of the agent virtual or physical, humanoid or non-humanoid and its relevance to user engagement.

Each of these categories is supported by well-defined terminology to reduce ambiguity and improve clarity. By applying the eCCo framework, researchers and practitioners can better compare systems, identify trends, and improve transparency in the reporting of conversational agent interventions.

The eCCo framework serves as a foundation for systematic extraction and benchmarking, especially in combination with automation tools like GROBID and LLMs. In this thesis, it forms the central structure around which our information extraction and evaluation pipeline is built.

2.3 AI in Literature Review Automation

Artificial Intelligence (AI) is playing an increasingly important role in the automation of systematic literature reviews (SRs). It can assist researchers across multiple stages of the review process, including document retrieval, eligibility assessment, data extraction, and synthesis of results. The aim of these applications is to reduce the effort required from human reviewers while maintaining reliability and accuracy.

Recent advances in LLMs have shown their potential, particularly for screening and extraction. For example, one system applied in a Vitamin D review processed more than 14,000 publications and reached an exclusion accuracy of 99.5%, while ensuring that no relevant studies were missed (false-negative rate of 0%) [10]. In parallel, platforms such as **AiReview** provide interactive, web-based environments where LLMs are integrated into SR workflows, enabling researchers to validate or adjust screening decisions as they progress [11].

Alongside LLM-based methods, widely adopted semi-automated tools are also used in practice. **Rayyan**, for instance, incorporates machine learning to recommend article inclusion and to facilitate the resolution of reviewer conflicts, whereas **DistillerSR** delivers end-to-end SR management with features like AI-guided prioritization, customizable extraction templates, and complete audit tracking [12], [13].

Survey studies indicate that most AI-based tools concentrate on particular steps of the process mainly screening and data extraction rather than full automation [14]. Human supervision therefore remains crucial. Another limitation is the absence of common evaluation standards, which complicates the comparison of results across tools and contributes to limited adoption [15].

In sum, AI has considerable potential to accelerate and improve evidence synthesis in systematic reviews. However, ensuring transparency, reproducibility, and methodological rigor remains an open challenge, requiring further research and validation.

2.3.1 Overview of AI Applications in Literature Review Processes

Every year, millions of scientific publications are released, creating an urgent need for automated approaches to literature analysis and synthesis. Traditional systematic literature reviews, while representing the gold standard for evidence synthesis, face several critical limitations that AI technologies are uniquely positioned to address:

Time and Resource Constraints: Producing an SLR requires a substantial investment of effort: from defining the search strategy and identifying relevant studies to screening, extracting information,

and synthesizing findings. On average, the full process extends over 12 to 18 months, and once the search is completed, another year or more is often needed before publication. For primary research, inclusion in a review can take even longer, typically 2.5 to 6.5 years [16].

Scalability Challenges: With the exponential rise in scientific output, reviewers are often confronted with thousands of candidate studies to examine during the screening phase. In this context, AI and machine learning techniques provide essential support, helping to prioritize and filter documents more efficiently.

Consistency and Bias Issues: Human reviewers can get tired, make mistakes, and be influenced by personal biases. Different reviewers might also interpret criteria differently, which can affect the consistency and reliability of results. AI tools can apply inclusion and exclusion criteria in a uniform way across large sets of documents, reducing these inconsistencies.

Rapid Obsolescence: Evidence shows that nearly one quarter (23%) of published SLRs are already outdated within two years of release, as new studies emerge after the review has been finalized. AI can help counter this problem by enabling more adaptive and continuously updated review pipelines [17].

2.3.2 Natural Language Processing Applications in Literature Analysis

Natural Language Processing (NLP) lies at the foundation of most AI-driven approaches to automating literature reviews. It encompasses a range of computational techniques that enable computers to process, understand, and even generate human language, which makes it particularly suitable for dealing with the immense volume of scientific publications produced each year [18].

In the context of literature analysis, the role of NLP has steadily expanded. As early as 2006[19], neural networks were already being applied to support the identification of relevant studies through text mining. Since then, more advanced machine learning systems have been developed that can not only screen articles against predefined inclusion criteria but also extract key information such as study design, participant characteristics, and intervention details.

Beyond these basic tasks, specialized methods like Named Entity Recognition (NER) allow for the extraction of more granular elements, including demographic data, treatment types, measured outcomes, and research designs. More sophisticated NLP models can even capture relationships between concepts for instance, linking interventions to outcomes or mapping treatment pathways and their effects.

Topic modelling provides yet another valuable tool. Among these techniques, Latent Dirichlet Allocation (LDA) remains one of the most used. It makes it possible to uncover latent themes across large collections of documents and to generate thematic summaries. By tracking how these topics evolve over time, LDA also helps highlight new research trends and identify underexplored areas in the literature [20].

2.3.3 Large Language Models and Advanced NLP Capabilities

The emergence of large language models (LLMs) such as GPT-4, Claude, and other domain-specific systems has greatly expanded the possibilities for automating literature reviews. Unlike traditional keyword-matching tools, these models interpret language in its broader context, which allows them to assess both the relevance and the quality of studies with greater nuance.

Thanks to their adaptability, LLMs can contribute to multiple stages of the review workflow. They are capable of screening abstracts, analyzing full texts, extracting structured data, assessing study quality, summarizing results, and even managing citation information. Their performance can also be

specialized for different academic domains through methods such as fine-tuning, prompt design, or few-shot learning

In practice, many effective systems combine conventional approaches with LLM capabilities. High-speed classifiers can process large volumes of straightforward cases, while LLMs focus on more complex or ambiguous decisions. This division of labour improves efficiency and ensures that human reviewers can dedicate their time to validation and oversight. By keeping experts “in the loop,” the system benefits from feedback that strengthens accuracy and reliability over time.

2.4 GROBID for Scholarly PDFs

GROBID (GeneRation Of BIbliographic Data) is an open-source tool designed to extract structured metadata and full-text information from scholarly PDF documents[21]. It uses machine learning models to parse academic papers and convert them into structured TEI (Text Encoding Initiative) XML format.

Studies have shown that GROBID achieves high accuracy in extracting bibliographic metadata, often surpassing 90% F1-scores for fields like title, authors, and affiliations, and outperforming other tools like CERMINE and Science Parse [22]. It has also been adapted for specialized domains, such as extracting superconducting material properties from research papers [23].

In our project, GROBID plays a central role in the early phase of the extraction pipeline. It retrieves bibliographic metadata such as title, authors, publication year, affiliations, and abstract, which populate the "Metadata" and parts of the "Study Design" sections of the eCCo framework.

One of GROBID’s strengths is its ability to handle PDFs with complex layouts, such as multi-column text, footnotes, and tables, ensuring that our LLM-based pipeline starts with accurate input. We integrated GROBID as a backend service in our Streamlit application. When a user uploads a PDF, it is sent to GROBID, which returns a TEI XML file. This XML is then converted into structured Python objects for the next pipeline steps.

By combining GROBID with LLMs, we separate responsibilities: GROBID handles factual bibliographic parsing, while LLMs focus on interpreting content and extracting deeper semantic features like intervention type, agent characteristics, or system interactions. This improves robustness, modularity, and reproducibility in our system.

2.5 Vector Search with Embeddings

Vector search with embeddings is a technique used to find semantically similar pieces of text. Instead of relying only on keyword matching, this method represents documents and queries as numerical vectors in a high-dimensional space. The similarity between these vectors often measured using cosine similarity determines how relevant a document is to a query.

In the context of our project, vector search is a core part of the Retrieval-Augmented Generation (RAG) pipeline. Scientific papers are split into smaller chunks, and each chunk is transformed into an embedding using models such as OpenAI's text-embedding-3-large or HuggingFace models. These embeddings are stored in a vector database like ChromaDB for efficient retrieval.

When a user asks a question, the query is converted into an embedding. The system then searches the database for the most similar chunks, which are passed to the language model to generate a context-aware answer. This approach improves precision compared to simple keyword search and ensures that the answers are grounded in the retrieved scientific content.

This method is widely used in modern AI applications, including semantic search, recommendation systems, and chatbots. In literature reviews, it helps quickly locate relevant information across large collections of papers, enabling more efficient and targeted analysis[24].

2.6 Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) combines information retrieval with text generation. It allows a language model to access an external knowledge base during the generation process, ensuring that answers are accurate and supported by relevant sources[25].

In a RAG workflow, a user query is first converted into an embedding. The system retrieves the most relevant chunks from a vector database, then feeds these chunks along with the query into a language model. The model generates an answer based on both the retrieved context and its own learned knowledge.

For our project, RAG ensures that the language model's answers are grounded in the actual content of scientific articles. This approach reduces hallucinations, improves transparency, and allows for citations or direct references to the source text.

We implemented RAG using ChromaDB for vector storage and retrieval, paired with both local LLMs (via Ollama) and cloud-based models from OpenAI. By combining these technologies, our system can flexibly switch between local and cloud models depending on performance, cost, and privacy requirements.

2.7 Integration of GROBID and RAG in Our System

In our pipeline, GROBID and RAG work together to ensure accurate and efficient information extraction from scholarly PDFs. GROBID handles the first stage by parsing uploaded PDFs and extracting reliable bibliographic metadata such as title, authors, affiliations, year, and abstract[26]. This structured metadata is used to populate the "Metadata" and "Study Design" sections of the eCCo framework.

Once the metadata is extracted, the full text of the paper is split into smaller chunks. These chunks are transformed into embeddings and stored in a vector database for semantic search. The RAG system retrieves the most relevant chunks in response to user queries and provides them as context to the language model.

This integration ensures that factual bibliographic details are captured with high precision by GROBID, while semantic understanding and context-aware answers are provided by RAG. The separation of responsibilities increases reliability, reduces errors, and supports reproducibility in literature review automation.

2.8 LLM-based eCCo Extraction

The final stage of the pipeline involves applying LLMs to extract structured information according to the eCCo framework. Using the metadata provided by GROBID and the context retrieved by the RAG system, the LLM is prompted with carefully designed instructions tailored to each eCCo field. These prompts are optimized for extracting specific details such as agent role, delivery channel, proactivity, and conversational style.

We experimented with both local LLMs (TinyLLaMA, Phi-3, Mistral) and cloud-based models (GPT-4o, GPT-4o-mini). Local models offer lower latency and data privacy, while cloud models generally achieve higher accuracy. This modular approach allows for independent updates to prompts, model

selection, and retrieval configurations without affecting the overall architecture, ensuring flexibility and scalability.

2.9 Benchmarking and Evaluation

To assess the system's performance, we designed a benchmarking framework that compares extracted information with manually created ground truth data. The evaluation uses multiple metrics *exact match*, *cosine similarity*, and *semantic similarity* to measure the accuracy and consistency of results across various eCCo fields. We tested local models (TinyLLaMA, Phi-3, Mistral) and cloud models (GPT-4o, GPT-4o-mini), analyzing their performance in metadata extraction, semantic retrieval, and structured eCCo field extraction. Results highlighted trade-offs between accuracy, speed, and resource requirements[27].

2.10 Summary of the Pipeline

Our pipeline follows a modular design:

1. **GROBID** extracts accurate bibliographic metadata from scholarly PDFs.
2. **Text Preprocessing and Chunking** split the content into manageable segments.
3. **Vector Search with Embeddings** enables semantic retrieval of relevant chunks.
4. **RAG** provides context-aware inputs to the LLM, grounding its responses in source content.
5. **LLM-based eCCo Extraction** applies targeted prompts to extract structured information.
6. **Benchmarking and Evaluation** measure performance and guide optimization. This structured workflow ensures precision, reproducibility, and adaptability to different models and datasets.

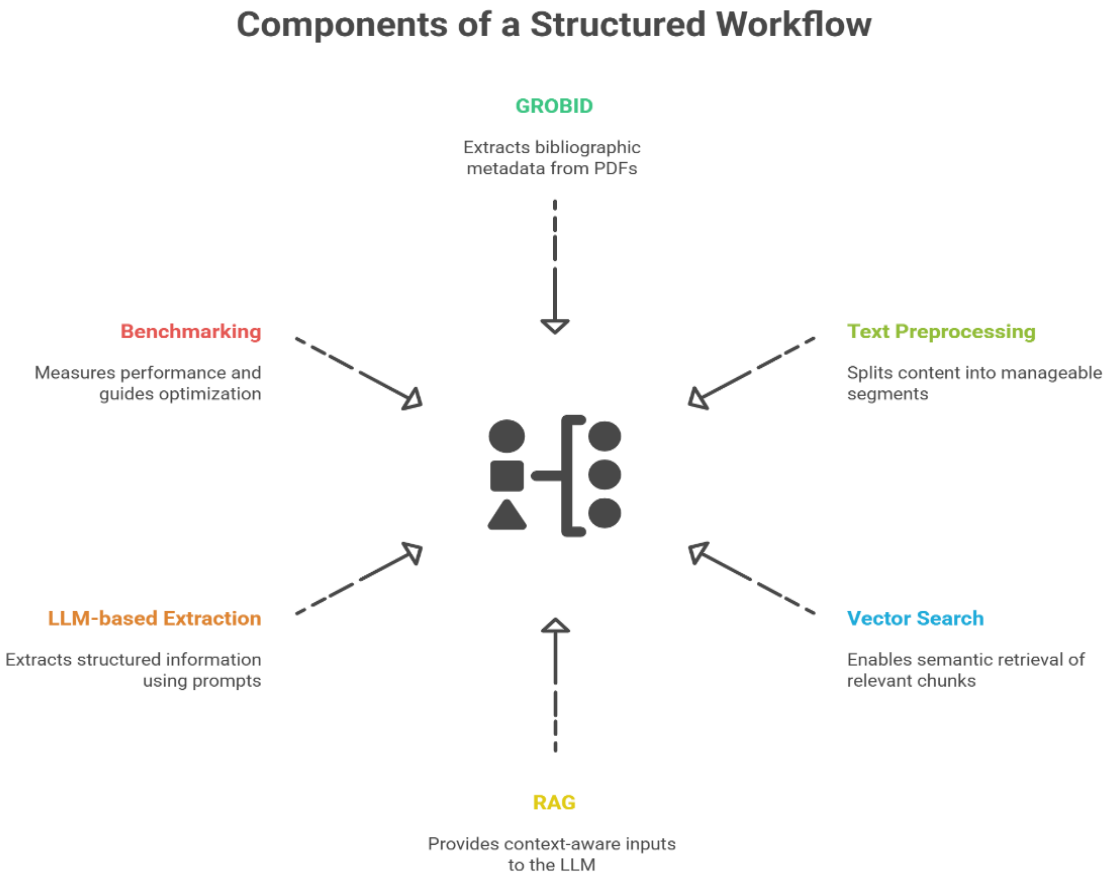


Figure 3 - The diagram illustrates the six main components of the pipeline

3

Methodology and System Design

This chapter describes the technical foundation and step-by-step implementation of our pipeline for eCCo-based information extraction from scientific literature. It introduces the overall system architecture and explains how each component contributes to processing, indexing, retrieving, and analyzing scholarly articles. The methodology reflects a modular approach combining rule-based parsing, semantic vector search, and LLM-based reasoning.

In addition to conceptual explanations, this section includes technical specifications, JSON data structures, and code snippets that illustrate the actual implementation of core functionalities. From document ingestion and metadata extraction to semantic chunking, embedding and prompt generation each phase is supported by practical examples and tooling choices.

Chapter's Content

3.1 System Architecture

The proposed system is based on a modular architecture designed for flexibility, reproducibility, and ease of integration. Each module in the pipeline is responsible for a distinct processing task, allowing for independent updates and evaluation. This architecture follows a linear and iterative flow, where outputs from one stage serve as inputs to the next.
At a high level, the system includes the following components:

1. **Document Ingestion** – Users upload scientific PDFs through a web interface. These documents are stored and queued for preprocessing.
2. **Metadata Extraction (GROBID)** – GROBID is used to parse the PDFs and extract metadata (e.g., title, authors, affiliations) and structure the text into logical sections.
3. **Text Chunking and Preprocessing** – The full text is split into small, overlapping chunks to improve the resolution of semantic search and to retain contextual integrity during retrieval.
4. **Vector Embedding and Indexing** – Each chunk is transformed into a vector using embedding models (e.g., OpenAI or HuggingFace). These vectors are stored in ChromaDB, a local vector database.
5. **Retrieval-Augmented Generation (RAG)** – A retrieval module fetches top-k relevant chunks based on similarity to a prompt or question. These are passed as context to the language model.
6. **LLM-based eCCo Extraction** – The context and tailored prompts are submitted to an LLM (local or cloud) to extract structured information according to the eCCo framework.
7. **Benchmarking and Evaluation** – Results are compared to manually annotated ground truth using multiple metrics (e.g., exact match, cosine similarity).

The entire pipeline is orchestrated via a combination of Python modules, REST APIs, and Streamlit for user interaction. The system supports both local execution (via Ollama models) and cloud-based models (via OpenAI API), offering deployment flexibility.

3.1.1 System Architecture Layers

The system is built on four main layers, each with a specific role in automating the literature review process.

The **User Interface layer**, developed with Streamlit, offers researchers a simple web-based workspace. It supports uploading and managing multiple PDF files, selecting between local or cloud models, choosing different processing modes, and monitoring progress in real time. Results can be exported in various formats for further analysis.

The **Intelligence layer** contains the core AI components. It integrates both local and cloud-based language models, applies optimized prompts for each model type, and uses GROBID for precise bibliographic metadata extraction. Built-in quality controls check the reliability of extractions and highlight inconsistencies for review.

The **Data Processing layer** handles document preparation and analysis. It extracts structured text from PDFs, segments it into coherent chunks, stores it in a vector database for semantic search, and retrieves relevant content to answer queries or perform detailed analysis.

Finally, the **Storage and Persistence layer** ensures data integrity and scalability. It maintains a persistent vector store for fast retrieval, caches result to avoid reprocessing, manages configuration for different environments, and supports flexible export options to meet diverse research needs.

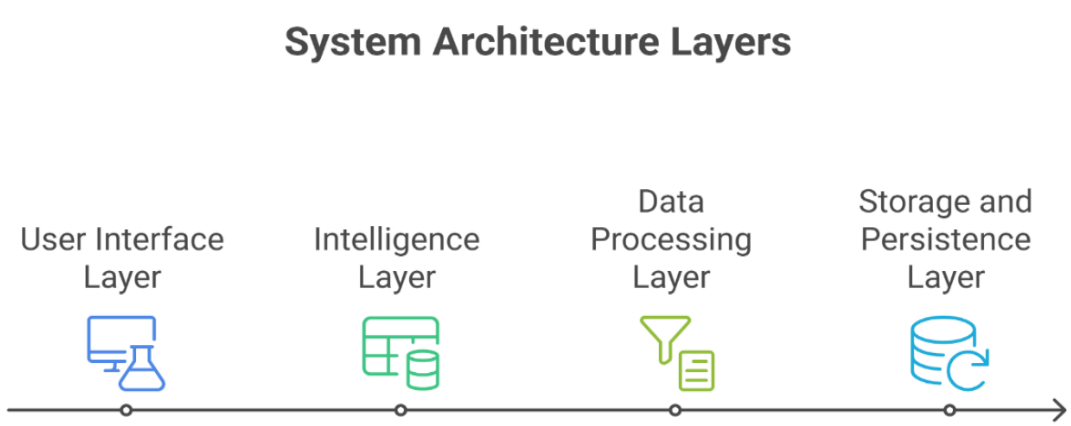


Figure 4- Overview of the system's four-layer architecture: The User Interface Layer provides researchers with a web-based platform to upload, configure, and visualize document processing. The Intelligence Layer integrates language models and metadata extraction tools for content analysis. The Data Processing Layer manages text extraction, segmentation, and semantic search. The Storage and Persistence Layer ensures secure data storage, caching, and export for long-term accessibility.

3.1.2 Processing Workflows

The architecture supports multiple processing workflows tailored to different literature review scenarios.

Workflow 1: Metadata-Only Processing

PDF Input → GROBID Processing → Bibliographic Extraction → Results Export

This lightweight workflow focuses on extracting bibliographic information without content analysis, suitable for large-scale document cataloging and initial literature survey activities.

Workflow 2: Content Analysis Pipeline

PDF Input → Text Extraction → Semantic Chunking → Vectorization → LLM Analysis → Feature Extraction

Comprehensive content analysis workflow that extracts domain-specific features and characteristics, ideal for detailed systematic reviews and meta-analyses.

Workflow 3: Eligibility Assessment

PDF Input → Text Processing → Vector Indexing → Criteria Evaluation → Eligibility Decision

PRISMA-based eligibility assessment workflow that automatically evaluates documents against predefined inclusion/exclusion criteria, streamlining the initial screening phase of systematic reviews.

Workflow 4: Complete Integration Pipeline

PDF Input → Parallel Processing (GROBID + Text Processing) → Eligibility Assessment → [If Eligible] → Content Analysis → Combined Results

Comprehensive workflow that combines all processing capabilities, providing complete automation from document ingestion to final result generation.

3.2 Data Sources

The data used in this research consists primarily of scientific articles in PDF format. These documents were selected based on their relevance to conversational agents in healthcare, ensuring that the content aligns with the thematic dimensions of the eCCo framework. The selected articles cover a range of study types, including journal publications, conference proceedings, and technical reports.

The sources include both open-access repositories and curated datasets from previous systematic literature reviews. Most documents were manually gathered from platforms such as PubMed, Google Scholar, and institutional databases, with attention paid to diversity in study design, target population, and intervention types. In addition, a significant number of documents were collected directly from the official eCCo repository[1], which offers access to relevant scientific articles evaluated under the eCCo framework.

To ensure benchmarking quality, a subset of these articles was annotated manually by domain experts, creating ground truth data for each eCCo field. This annotated dataset serves as a reference point for evaluating model performance during extraction tasks.

3.3 Data Preprocessing

Data preprocessing represents a fundamental component of the AI-driven literature review system, serving as the bridge between raw extracted text and meaningful, processable information for llms. This stage encompasses comprehensive text cleaning, normalization, semantic enrichment, and quality assurance procedures.

3.3.1 Text Cleaning and Normalization

The first step in preprocessing involves cleaning and normalizing the extracted text. This includes removing inconsistencies, correcting encoding errors, and standardizing characters. Scientific PDFs often contain irregular spacing, different types of quotation marks and dashes, or invisible characters that disrupt downstream analysis.

```python
def normalize_text_unicode(text):
    """Normalize Unicode characters and standardize encoding."""
    import unicodedata

    # NFC normalization for consistent character representation
    normalized = unicodedata.normalize('NFC', text)

    # Character-level cleaning
    normalized = re.sub(r'[\u2000-\u200B\u2028-\u2029\u202F\u205F\u3000]', ' ', normalized)
    normalized = re.sub(r'[\u2010-\u2015]', '-', normalized)  # Standardize dashes
    normalized = re.sub(r'[""''„‟]', '"', normalized)  # Standardize quotes

    return normalized
```

Listing 1 shows the function responsible for Unicode normalization. It applies NFC (Canonical Decomposition followed by Canonical Composition) to unify character encoding, replaces irregular spaces with standard spaces, standardizes dashes, and unifies different quotation marks. This step ensures that text is consistently represented regardless of the PDF source.

3.3.2 Semantic Chunking and Segmentation

After cleaning, the text is divided into semantically coherent units (chunks) to optimize LLM processing. Unlike simple character-based splitting, the system applies content-aware segmentation, prioritizing paragraph boundaries and falling back to sentence-level splitting when necessary. Chunk size is adapted dynamically (500–1500 characters) based on content density, and overlaps are used to preserve context between chunks.

[1] https://ecco.humantech.institute/

```python
def smart_paragraph_split(text):
    """Intelligent paragraph-based text segmentation."""
    # Normalize whitespace while preserving structure
    text = re.sub(r'\s+', ' ', text)
    # Primary segmentation on paragraph boundaries
    paragraphs = text.split('\n\n')
    # Fallback to sentence-based splitting for continuous text
    if len(paragraphs) == 1:
        sentences = re.split(r'(?<=[.!?])\s+(?=[A-Z])', text)
        # Intelligent sentence grouping
        current_para = ""
        paragraphs = []
        for sentence in sentences:
            # Dynamic size optimization based on content density
            if len(current_para + sentence) < 500:
                current_para += sentence + " "
            else
                if current_para.strip():
                    paragraphs.append(current_para.strip())
                current_para = sentence + " "
        if current_para.strip():
            paragraphs.append(current_para.strip())
    return [p.strip() for p in paragraphs if len(p.strip()) > 50]
```

Listing 2 presents the segmentation algorithm. It first attempts to split text into paragraphs while preserving semantic structure. If no clear paragraphs are detected, it switches to sentence-based segmentation, grouping sentences into chunks of optimal size. The result is a set of well-structured segments that balance readability and model efficiency.

3.3.3 Domain-Specific Processing for eCCo

The preprocessing pipeline incorporates specialized handling for the eCCo (conversational e-coaches) framework, implementing domain-specific vocabulary recognition and semantic enhancement tailored to digital health coaching literature.

eCCo Vocabulary Recognition: The system includes comprehensive dictionaries of e-coaching terminology, intervention types, and digital health concepts, enabling precise identification and preservation of domain-specific language during preprocessing

```python
def enhance_ecco_terminology(text, chunk_metadata):
    """Enhance text with eCCo-specific terminology recognition."""

    # Target intervention categories with domain vocabularies
    intervention_keywords = {
        'PHYSICAL_ACTIVITY': [
            'physical activity', 'exercise', 'movement', 'fitness',
            'walking', 'running', 'sport', 'activity monitoring'
        ],
        'EATING': [
            'eating', 'nutrition', 'diet', 'food', 'meal planning',
            'dietary habits', 'nutritional intervention', 'healthy eating'
        ],
        'SYMPTOMS': [
            'symptoms', 'pain', 'fatigue', 'depression', 'anxiety',
            'mood', 'quality of life', 'wellbeing', 'health status'
        ]
    }

    # Detect and annotate domain-specific content
    detected_interventions = []
    text_lower = text.lower()

    for category, keywords in intervention_keywords.items():
```

```
        if any(keyword in text_lower for keyword in keywords):
            detected_interventions.append(category)
    # Enhance metadata with domain classifications
    chunk_metadata['ecco_interventions'] = detected_interventions
    chunk_metadata['domain_relevance'] = len(detected_interventions) > 0
    return text, chunk_metadata
```

*Listing 3 illustrates the function that scans text for keywords related to specific intervention categories such as *physical activity*, *nutrition*, and *symptom management*. It annotates the chunk metadata with detected categories and flags its domain relevance for later filtering or prioritization.*

3.3.4 Quality Assurance and Validation

Before moving to vectorization, the system applies quality checks to ensure the extracted chunks are reliable. Each chunk is evaluated on length, semantic richness, character composition, and absence of excessive repetition. Chunks not meeting the quality threshold are flagged for review.

```
def validate_chunk_quality(chunk_data):
    """Comprehensive chunk quality validation."""
    text = chunk_data.get("text", "")
    metadata = chunk_data.get("metadata", {})
    quality_metrics = {
        "sufficient_length": len(text.strip()) >= 50,
        "meaningful_content": len(text.split()) >= 10,
        "character_ratio": len([c for c in text if c.isalnum()]) / len(text) > 0.5,
        "no_excessive_repetition": not has_excessive_repetition(text),
        "proper_encoding": all(ord(c) < 65536 for c in text)  # Basic Unicode check  }
    # Calculate overall quality score
    quality_score = sum(quality_metrics.values()) / len(quality_metrics)
    metadata["quality_score"] = quality_score
    metadata["quality_flags"] = {k: v for k, v in quality_metrics.items() if not v}
    return quality_score >= 0.8  # 80% quality threshold
```

Listing 4 shows the validation function. It computes a quality score from several metrics and enriches the chunk metadata with both the score and detailed flags for any failing criteria. Only chunks above the 80% quality threshold proceed to the embedding stage.

3.4 PDF Processing and Text Extraction Pipeline

The PDF processing and text extraction pipeline is the entry point of the system. Its goal is to take an unstructured academic PDF and turn it into clean, well-segmented chunks of text that keep their meaning but are ready for AI analysis. It uses a sequence of steps that validate the file, extract the text, segment it intelligently, clean it, and finally prepare it for semantic search and retrieval.

3.4.1 Document Input and Preprocessing

The process starts by making sure the uploaded document is valid and ready to process. The system checks that the file is a proper PDF, under the maximum allowed size (200 MB), and not corrupted. It also distinguishes between text-based PDFs and scanned image PDFs. Currently, only text-searchable PDFs are processed directly, while OCR support is planned for future versions.

3.4.2 Text Extraction Engine

The main extraction tool is **PyPDF2**, enhanced with custom logic to work better with scientific documents. This approach is fast, reliable, and works well for most academic formats. It also preserves metadata for each chunk (page number, word count, section type, etc.), which is useful for later processing and traceability.

```
def extract_with_pypdf2_enhanced(pdf_path):
    chunks = []
    chunk_id = 0
    with open(pdf_path, 'rb') as file:
        pdf_reader = PyPDF2.PdfReader(file)
        for page_num, page in enumerate(pdf_reader.pages):
```

```
            page_text = page.extract_text()
            # Skip empty or minimal content pages
            if not page_text or len(page_text.strip()) < 50:
                continue
            # Special handling for title page
            if page_num == 0:
                title_chunk = extract_title_metadata(page_text)
                if title_chunk:
                    # Priority chunk with enhanced metadata
                    chunks.append({
                        "text": title_chunk,
                        "metadata": {
                            "page": page_num + 1,
                            "chunk_id": chunk_id,
                            "section_type": "title_page",
                            "priority": "high" }
                    })
            # Intelligent paragraph segmentation
            paragraphs = smart_paragraph_split(page_text)
            for para in paragraphs:
                if len(para.strip()) > 100:
                    chunks.append({
                        "text": para.strip(),
                        "metadata": {
                            "page": page_num + 1,
                            "chunk_id": chunk_id,
                            "section_type": "content",
                            "word_count": len(para.split())}
                    })
                    chunk_id += 1
```

Listing 5 shows the extraction function. It loops through each page, skips empty or irrelevant pages, treats the first page as a priority "title page" if it contains useful metadata, and then segments the text into meaningful paragraphs.

If the enhanced extraction fails, a simpler fallback method breaks the document into overlapping windows of text to ensure nothing is lost.

3.4.3 Intelligent Document Segmentation

After extraction, the text is split into coherent chunks that make sense on their own but still keep enough context for AI analysis. The segmentation respects paragraph boundaries whenever possible and only falls back to sentence-based splitting when the text is continuous with no clear breaks. Overlaps of around 200 characters are added between chunks so context is not lost between segments.

```
def smart_paragraph_split(text):
    # Normalize whitespace while preserving structure
    text = re.sub(r'\s+', ' ', text)

    # Primary split on paragraph boundaries
    paragraphs = text.split('\n\n')

    # Fallback to sentence-based splitting for continuous text
    if len(paragraphs) == 1:
        sentences = re.split(r'(?<=[.!?])\s+(?=[A-Z])', text)

        # Group sentences into optimal chunks
        current_para = ""
        paragraphs = []

        for sentence in sentences:
            if len(current_para + sentence) < 500:
                current_para += sentence + " "
            else:
```

```
                if current_para.strip():
                    paragraphs.append(current_para.strip())
                current_para = sentence + " "
```

Listing 6 shows the paragraph splitter. It tries to keep ideas together, aiming for chunks between 500 and 1500 characters.

3.4.4 Text Preprocessing and Normalization

Once segmented, the chunks are cleaned. This includes removing repeated headers and footers, page numbers, and leftover formatting. Unicode normalization and punctuation standardization ensure that text is represented consistently, regardless of the original PDF formatting.

Chunks with too little content or too many formatting artifacts are discarded. The system records extraction quality metrics, such as success rates and content density, for monitoring and improvement.

3.4.5 Integration with Vector Storage

The cleaned chunks are then prepared for vectorization with the `all-MiniLM-L6-v2` embedding model. They are stored in JSON format for traceability and indexed in ChromaDB for semantic search. Documents are grouped into collections so they can be retrieved or updated independently.

The pipeline supports batch processing of multiple PDFs, uses caching to skip reprocessing already-analyzed documents, and adapts resource usage based on document complexity.

3.4.6 Error Handling and Recovery

The system is designed to be fault-tolerant. If the main extraction fails, it automatically switches to fallback methods. Errors are classified, logged, and, when possible, recovered from automatically. Metadata consistency checks ensure extracted information aligns with document structure, and completeness checks verify that most of the original content has been captured.

3.5 Metadata Extraction (GROBID)

To extract structured metadata from scientific PDFs, we integrated GROBID (GeneRation Of BIbliographic Data), an open-source tool widely used for parsing scholarly documents. GROBID converts unstructured PDF files into XML representations, identifying logical sections and extracting bibliographic metadata with high accuracy.

> **How GROBID Works**
>
> GROBID is designed specifically for scholarly articles. It doesn't just extract text—it analyzes the document structure, layout, and formatting. Internally, it relies on machine learning models like Conditional Random Fields (CRFs) to recognize different elements of academic papers.
>
> Instead of returning plain text, GROBID generates **TEI XML** output (a standard used for encoding structured documents in the humanities), which allows for detailed parsing of titles, abstracts, authors, and references.
>
> GROBID runs as a **Dockerized microservice**, separate from the main application. This setup makes it easy to scale and deploy without interfering with the rest of the system.

```
def _extract_header_info(self, root: ET.Element) -> Dict:
    """Extract header information from TEI XML."""
```

```
    header_info = {
        "title": "",
        "authors": [],
        "affiliations": [],
        "abstract": "",
        "keywords": [],
        "journal": "",
        "volume": "",
        "year": "",
        "doi": ""
    }
    # Find fileDesc element
    file_desc = root.find('.//tei:fileDesc', self.namespaces)
    if file_desc is None:
        return header_info
    # Extract title
    title_elem = file_desc.find('.//tei:titleStmt/tei:title[@type="main"]',
                                self.namespaces)
    if title_elem is not None:
        header_info["title"] = self._clean_text(title_elem.text or "")
    # Extract authors and affiliations
    source_desc = file_desc.find('.//tei:sourceDesc', self.namespaces)
    if source_desc is not None:
        authors_data = self._extract_authors_and_affiliations(source_desc)
        header_info["authors"] = authors_data["authors"]
        header_info["affiliations"] = authors_data["affiliations"]
```

Listing 7 shows how we extract the title, authors, and affiliations from the TEI header

3.5.1 Output Format and Metadata Structure

Once extracted, the metadata is converted to a standardized JSON format. This ensures compatibility with reference managers like Zotero or EndNote, and can be used for bibliometric analysis.
Here is an example of what the final output looks like:

```
{
  "title": "Document title",
  "authors": [
    {
      "name": "Author Name",
      "email": "author@institution.edu",
      "affiliations": ["Institution Name", "Department"]
    }
  ],
  "abstract": "Document abstract text",
  "keywords": ["keyword1", "keyword2", "keyword3"],
  "journal": "Journal name",
  "volume": "Volume number",
  "issue": "Issue number",
  "pages": "Page range",
  "year": "2024",
  "doi": "10.xxx/xxx",
  "issn": "Journal ISSN",
  "references": [
    {
      "title": "Reference title",
      "authors": ["Reference Author"],
      "journal": "Reference journal",
      "year": "2023",
      "doi": "10.xxx/xxx"
    }
  ],
  "sections": [
    {
      "title": "Section title",
```

```
      "content_length": 1500
    }
  ],
  "processing_metadata": {
    "type": "fulltext",
    "timestamp": "2024-08-16 10:30:00",
    "grobid_version": "0.8.0"
  }
}
```

Listing 8- Comprehensive Metadata Schema

3.6 Semantic Search & Retrieval (Vector Search & RAG)

Semantic chunking and vectorization form the backbone of our system. They transform raw text into meaningful numerical representations (vectors) that can be searched and compared efficiently. This makes it possible to retrieve relevant passages from hundreds of documents while preserving context.

3.6.1 Semantic Chunking

Instead of cutting text into arbitrary pieces, semantic chunking respects the document's meaning. Paragraphs and section breaks serve as natural boundaries, and when needed, long continuous passages are split into smaller chunks. To maintain coherence, small overlaps are added between consecutive chunks, so no idea is lost at the boundaries.

Each chunk is also enriched with metadata such as its position in the document, type of section (abstract, introduction, results, etc.), and density of scientific terms. This metadata helps later when we need to filter or prioritize certain parts of a paper.

3.6.2 Vector Embeddings

Once chunks are created, they need to be turned into vectors. For this, we use a transformer model (all-MiniLM-L6-v2) that converts text into 384-dimensional embeddings. These embeddings capture semantic meaning—so two sentences that express similar ideas end up close to each other in vector space.

This step is crucial because it allows us to search by meaning, not just keywords. For example, "virtual coach" and "conversational agent" will be recognized as related concepts.

```
class SemanticVectorizer:
    def __init__(self, model_name="sentence-transformers/all-MiniLM-L6-v2"):
        from langchain.embeddings import HuggingFaceEmbeddings
        self.embedding_model = HuggingFaceEmbeddings(
            model_name=model_name,
            model_kwargs={'device': 'cpu'},
            encode_kwargs={'normalize_embeddings': True}
        )
    def generate_chunk_embeddings(self, chunks_data):
        chunks = chunks_data.get("chunks", [])
        texts = [c["text"].strip() for c in chunks if len(c.get("text", "")) > 30]
        embeddings = self.embedding_model.embed_documents(texts)
        return embeddings, texts
```

Listing 9 - The class loads the transformer model and generates embeddings for each chunk. It normalizes them so all vectors have the same length, which is important for similarity calculations.

3.6.3 ChromaDB Integration

To store and query embeddings, we use **ChromaDB**, a vector database optimized for similarity search. Each document has its own collection inside ChromaDB, which keeps the system organized and makes retrieval efficient.

```
def create_vector_index(self, chunks_path, pdf_name):
```

```
    index_dir = VECTOR_DIR / pdf_name
    if index_dir.exists():
        try:
            vectorstore = Chroma(
                persist_directory=str(index_dir),
                embedding_function=self.embedding_model,
                collection_name=f"collection_{pdf_name}"
            )
            vectorstore.similarity_search("test", k=1)
            return index_dir
        except Exception:
            shutil.rmtree(index_dir)
    # Recreate index if needed
```

Listing 10 - This function checks if a vector index already exists for a given document. If valid, it reuses it; if not, it creates a fresh index by embedding all the chunks.

3.6.4 Similarity Search and Retrieval

When a query is made, the system retrieves the most relevant chunks from ChromaDB using **cosine similarity**. This metric measures how close two vectors are in direction, ignoring length. It's well-suited for text because long and short sentences can still be semantically similar.

The system also adapts the number of results (k) depending on the type of information requested. For example, when looking for the article title, only one or two results are needed, but for intervention details, more context is retrieved.

```
def get_optimal_k_for_field(field_name):
    field_k_mapping = {
        "articleTitle": 2,
        "authorName": 2,
        "articleYear": 1,
        "studyAim": 4,
        "targetInterventions": 5,
        "default": 3
    }
    return field_k_mapping.get(field_name, field_k_mapping["default"])
```

Listing 11 - This function decides how many chunks should be retrieved based on the field. Titles require fewer chunks, but interventions need more

3.6.5 Context-Aware Retrieval System

Intelligent Query Enhancement : The retrieval system implements sophisticated query enhancement strategies that leverage domain-specific terminology and semantic relationships to improve retrieval accuracy and relevance.

eCCo Terminology Integration: Query enhancement incorporates specialized terminology from the eCCo (conversational e-coaches) framework, ensuring that domain-specific concepts and terminology are properly weighted in similarity calculations.

Multi-Modal Query Strategy: The system employs multiple retrieval strategies including vector similarity search, keyword-based fallback, and hybrid approaches that combine multiple evidence sources for optimal context retrieval.

```
def get_context_smart(self, prompt_key, chunks_data, index_dir=None, model="chatgpt4o"):
    """Intelligent context retrieval with multi-modal strategy."""
    # Determine optimal retrieval parameters
    optimal_k = get_optimal_k_for_field(prompt_key)
    # Enhanced eCCo terminology mapping
```

```
    ecco_keywords_map = {
        "targetInterventions": """
        physical activity exercise movement fitness walking running sport monitoring
        eating nutrition diet food meal planning dietary habits intervention
        sleep rest patterns quality duration hygiene circadian rhythm
        pharmaceutical medication adherence compliance dosage therapy treatment
        symptoms pain fatigue depression anxiety mood wellbeing health status
        """,

        "actionInterventions": """
        assessment evaluating health condition functioning behavior evaluation
        education providing information improve knowledge learning instruction
        social support companionship practical assistance well-being coping
        behavior change modification intervention technique guidance behavioral
        """
```

*Listing 12 - This function is designed to **retrieve the most relevant context chunks** for a given field (such as articleTitle, studyAim, or targetInterventions). It combines two strategies: direct metadata filtering and semantic vector search.*

3.6.6 Quality and Validation

To ensure quality, embeddings are checked for coherence, and indexes are validated regularly. If corruption is detected, the index is rebuilt automatically. For queries where vector search fails, a fallback mechanism (keyword-based search) ensures that results are always available.

3.6.7 Role in the Pipeline

Semantic chunking and vectorization provide the foundation for the entire AI pipeline. By preparing text in this structured way, the system can deliver the right context to the language models, which improves accuracy and reduces hallucinations.

3.7 Multi-Model LLM Integration

When we designed this system, we realized that no single AI model works perfectly for everyone. Some researchers need fast processing for large document collections. Others need the highest quality results regardless of cost. So we built something that lets you choose the right tool for your specific situation.

3.7.1 Why Multiple Models Matter

During our testing, we discovered that different models excel at different tasks. TinyLlama is incredibly fast for simple extractions but struggles with complex reasoning. GPT-4o provides amazing accuracy but costs money with every API call. Phi3-Mini sits nicely in the middle - good quality for most tasks while running completely free.

Rather than forcing users to pick just one approach, we decided to support them all. This way, researchers can start with free local models for initial testing, then switch to cloud models for their final high-quality extractions. Or they can use local models for large-scale screening and save the expensive cloud models for only the most important papers.

The system automatically adapts based on which model you choose. When you select TinyLlama, it uses ultra-simplified prompts. When you pick GPT-4o, it can handle much more complex instructions.

3.7.2 Local Models with Ollama

Setting up Local Processing: Local models run entirely on your computer through Ollama. We chose Ollama because it makes running these models incredibly simple - no complex setup, just download and run.

The three local models we selected each serve different purposes:

1. **TinyLlama (1.1B parameters)**: It's tiny but surprisingly capable for basic extractions. We use ultra-simple prompts with TinyLlama like "Article title:" instead of complex instructions.

2. **Phi3-Mini (3.8B parameters)**: It provides good quality results while still running smoothly on modest hardware. This became our recommended starting point for most researchers.

3. **Mistral (7B parameters)**: The high-quality local option. When you need better results but want to stay offline, Mistral delivers. It can handle more complex reasoning about conversational e-coaching concepts.

```python
def generate_response(prompt, model="chatgpt4omini", openai_api_key=None):
    """
    Our unified function for calling any model, local or cloud.
    This handles all the API complexity behind a simple interface.
    """
    if not prompt or not prompt.strip():
        return ""

    # Local Ollama models - completely free and private
    if model in ["tinyllama", "phi3mini", "mistral"]:
        ollama_model_mapping = {
            "tinyllama": "tinyllama:latest",
            "phi3mini": "phi3:mini",
            "mistral": "mistral:latest",
        }
        return call_ollama(ollama_model_mapping[model], prompt)

    # Cloud models - require API key but more powerful
    elif model in ["chatgpt4o", "chatgpt4omini"]:
        openai_model_mapping = {
            "chatgpt4o": "gpt-4o",
            "chatgpt4omini": "gpt-4o-mini",
        }
        api_key = openai_api_key or OPENAI_API_KEY
        if not api_key:
            raise ValueError("OpenAI API key required for this model")

        return call_openai(openai_model_mapping[model], prompt, api_key)
```

*Listing 13 – This function provides a **single entry point** to call either local models (via Ollama) or cloud-based models (via OpenAI API). It hides all the backend complexity and allows the user to simply specify the model and the prompt.*

Local Model Benefits : What we love about local models is their complete privacy and zero ongoing costs. Your research papers never leave your computer. There's no usage limits, no API rate limiting, and no internet dependency. You can process thousands of papers without worrying about costs adding up.

The main trade-off is processing speed and quality. Local models are generally slower than cloud APIs and don't always understand complex instructions as well. But for many research tasks, they're perfectly adequate.

3.7.3 Cloud Models with OpenAI

Professional-Grade Processing :
Cloud models represent the current state-of-the-art in language understanding. We integrated OpenAI's models because they consistently provide the highest quality extractions, especially for complex tasks.

GPT-4o: Our "premium" option. When we need the absolute best results and cost isn't the primary concern, GPT-4o delivers. It excels at understanding complex academic language and making subtle distinctions between similar concepts.

GPT-4o-Mini: The cost-effective cloud option that we often recommend as a starting point for cloud processing. It provides much better results than local models while keeping costs reasonable.

3.7.5 Performance Trade-offs

Making Smart Choices: We designed the model selection to help researchers make informed trade-offs between speed, quality, and cost:

MODEL	SPEED SPEED	QUALITY	COST	BEST FOR
TINYLLAMA	⭐⭐⭐	⭐⭐	Free	Quick screening
PHI3-MINI	⭐⭐	⭐⭐⭐	Free	General use
MISTRAL	⭐⭐	⭐⭐⭐⭐	Free	Quality local processing
GPT-4O-MINI	⭐⭐⭐⭐⭐	⭐⭐⭐⭐	$	Cost-effective cloud
GPT-4O	⭐⭐⭐⭐⭐	⭐⭐⭐⭐⭐	$ $ $	Critical accuracy

Table 2 - Model comparison according to certain criteria

3.7.6 System Integration

Seamless Model Switching: From a user perspective, switching between models is completely seamless. You pick a model from the dropdown, and everything else adapts automatically - the prompts, the processing strategy, the expected response format.

Behind the scenes, the system handles all the complexity including API management, prompt adaptation, response processing, error handling, and performance monitoring.

Integration Benefits: The multi-model system works with all parts of our processing pipeline. Whether you're doing basic eCCo extraction, PRISMA eligibility assessment, or the full combined workflow, you can use any model you prefer.

The vector search system provides relevant context regardless of which model you choose. It adapts the amount and complexity of context based on the model's capabilities. Local models get more focused, shorter context snippets, while cloud models can handle longer, more detailed context.

User Experience: We made model selection intuitive. The interface clearly shows which models are available, what they cost, and what they're good for. Users get immediate feedback if a model isn't working properly.

For researchers new to AI, we provide clear guidance on which model to start with. For power users, we expose all the options and let them optimize for their specific requirements. This multi-model approach means the system can grow with users. They might start with free local models for exploration, then upgrade to cloud models for production use, or mix and match based on their specific project demands.

The key insight is that there's no "one size fits all" solution in AI. By supporting multiple models with adaptive behaviour, we created a system that serves researchers across the entire spectrum of needs, budgets, and technical requirements.

3.8 PRISMA Eligibility Assessment Engine

The PRISMA Eligibility Assessment Engine represents a crucial component of our automated literature review system. We designed this engine to systematically evaluate scientific articles against eight specific criteria that determine their eligibility for inclusion in systematic reviews focusing on conversational agents in healthcare.

3.8.1 Architecture and Core Components

We implemented the engine as the EnhancedEligibilityEngine class, which orchestrates the entire evaluation process. The system follows PRISMA guidelines through eight sequential assessment steps:

```
self.prisma_steps = [
    "step1_language",           # Language verification
    "step2_publication_type",   # Research study validation
    "step3_population",         # Human participants check
    "step4_health_domain",      # Health relevance
    "step5_target_population",  # Professional vs patient focus
    "step6_conversational_agent", # Conversational system presence
    "step7_automated_capabilities", # AI automation check
    "step8_research_content"    # Empirical research validation
]
```

*Listing 14 - This code defines the **sequence of PRISMA-inspired eligibility steps** used to decide whether a scientific article should be included in the review. Each step represents a specific filtering criterion applied in order*

The diagram below illustrates the **screening workflow** used to evaluate the eligibility of scientific articles in our study. We based this framework on the **PRISMA guidelines**, and more specifically, we adapted the eligibility checklist developed in the **eCCo project**. Each step in the flow represents a filtering criterion, ensuring that only relevant studies are included in the final review.

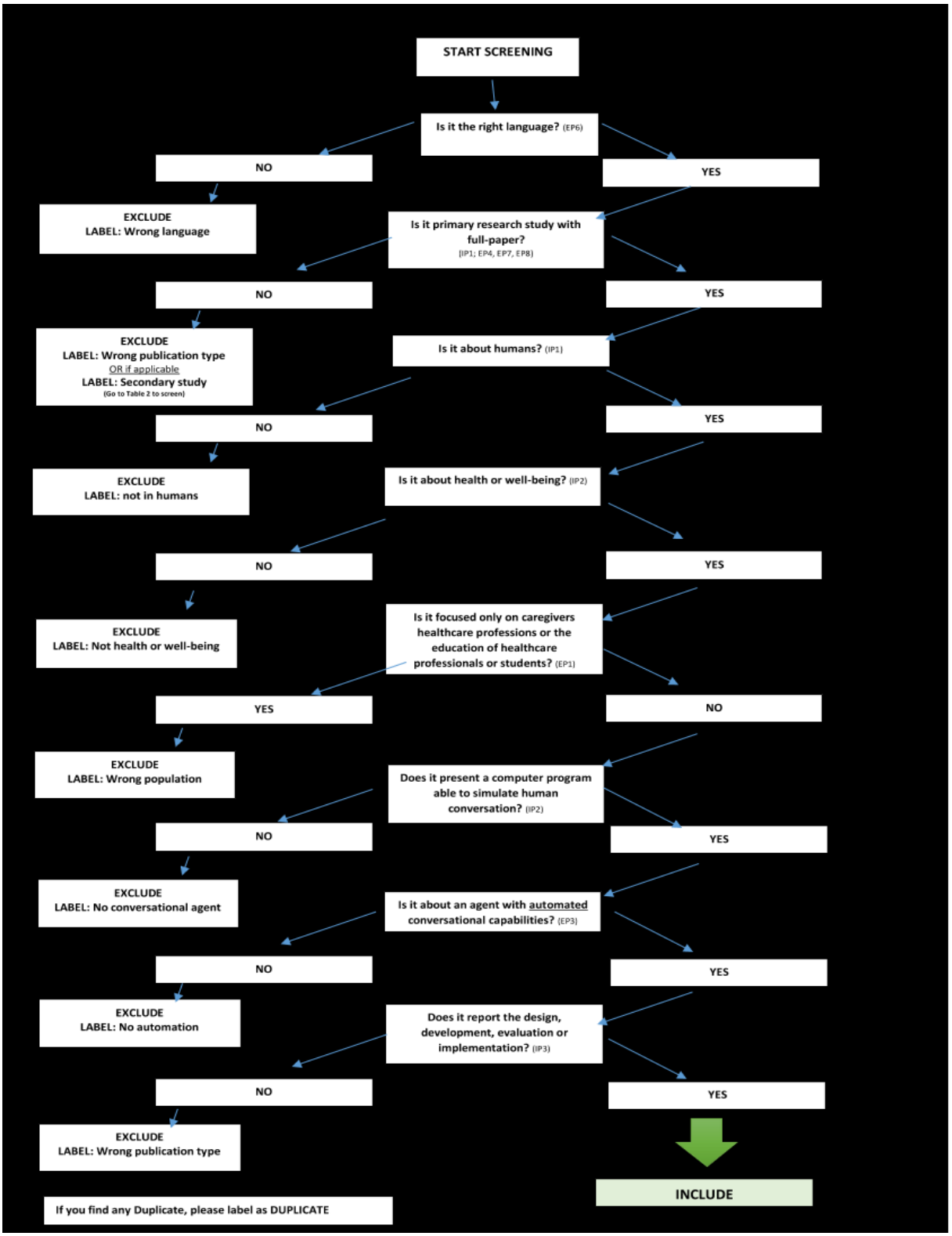


*Figure 5 - This diagram reflects how we applied **PRISMA-inspired criteria**, adapted from the **eCCo framework**, to guide the systematic selection of studies for our automated literature review.*

Each step evaluates specific inclusion criteria through targeted prompts adapted to different LLM capabilities. We created three prompt complexity levels to accommodate various model types:

```
// Ultra-simple prompts for lightweight models
{
  "step1_language": "English article? YES or NO:",
  "step5_target_population": "Only for doctors? YES or NO:",
  "step6_conversational_agent": "System talks to users? YES or NO:"
}
// Complex prompts for advanced models
{
  "step5_target_population": {
    "question": "Is this study focused EXCLUSIVELY on healthcare professionals?",
    "instruction": "Answer YES ONLY if targeting EXCLUSIVELY doctors, nurses, or medical tr
aining. Answer NO if ANY patients or end-users are included."
  }
}
```

*Listing 15 - This code shows how the system **adapts prompts depending on the type of language model used**.*

3.8.2 Contrastive Analysis for Critical Criteria

We developed a sophisticated "contrastive analysis" approach for ambiguous criteria, particularly Step 5 (target population assessment). This method searches for competing evidence patterns to make more accurate determinations:

```
self.contrastive_keywords = {
    "step5_target_population": {
        "professionals": [
            "doctors", "physicians", "nurses", "clinicians",
            "medical staff", "healthcare professionals", "training"
        ],
        "non_professionals": [
            "patients", "users", "end-users", "participants",
            "general public", "elderly", "adults", "children"
        ],
        "mixed_indicators": [
            "both", "including", "not only", "as well as"
        ]
    }
}
```

*Listing 16- This code defines a **keyword dictionary** used to improve the classification of studies according to their **target population** (Step 5 of the PRISMA-inspired eligibility process).*

The engine calculates balance scores to determine whether a study targets healthcare professionals exclusively or includes patient populations:

```
balance_score = (non_prof_strength + mixed_strength - prof_strength) / total_evidence
if balance_score > 0.3:
    recommendation = "Likely includes non-professionals (should be NO)"
elif balance_score < -0.3:
    recommendation = "Likely professionals only (should be YES)"
```

*Listing 17- This code snippet shows how the system **computes a balance score** to decide whether a study targets **professionals only** or **non-professionals**.*

3.8.3 Override Mechanism and Quality Control

We implemented an intelligent override system that performs additional verification when initial assessments appear inconsistent. For Step 5, when the system initially determines "YES" (professionals only), it automatically searches for counter-evidence.

3.8.4 Decision Logic and Validation

The final eligibility decision follows specific logic rules tailored for conversational agent systematic reviews. Most criteria require "YES" answers for inclusion, except Step 5 where "NO" indicates patient-focused studies (desirable for healthcare intervention research):

```
# Decision rules implemented
- ALL criteria should normally be YES for inclusion
- EXCEPT Step5: NO means includes patients (GOOD), YES means only professionals (BAD)
- Critical criteria: Conversational Agent, Health Domain, Human Population must be YES
```

*Listing 18 summarizes the logic used in the eligibility screening. For most steps, a paper is included only if the criterion is satisfied (YES). However, **Step 5 (Target Population)** works in the opposite way: a **NO***

We incorporated cross-validation checks to identify potential inconsistencies between related criteria, ensuring robust and reliable assessment outcomes. The system tracks all decision points, providing researchers with complete visibility into automated evaluations and enabling manual review validation when needed.

3.9 eCCo Characteristic Extraction Pipeline

The eCCo Characteristic Extraction Pipeline forms the core component responsible for identifying and extracting conversational e-coach characteristics from scientific literature. We designed this pipeline specifically to implement the official eCCo framework terminology, enabling systematic analysis of conversational agents in healthcare interventions.

3.9.1 Framework Structure and Target Fields

We organized the eCCo extraction around 15 key characteristics grouped into four logical categories. These fields represent the essential dimensions needed to comprehensively describe conversational e-coaches:

```
ECCO_TARGET_FIELDS = [
    'articleType', 'articleYear', 'articleTitle', 'authorName', 'authorCountry',
    'studyAim', 'interactionInputs', 'interactionOutputs', 'externalInterventions',
    'studyTypes', 'deliveryChannels', 'targetInterventions', 'actionInterventions',
    'additionalFeatures', 'embodiments'
]
ECCO_FIELD_CATEGORIES = {
    "Metadata": ['articleType', 'articleYear', 'articleTitle', 'authorName', 'authorCountry
'],
    "Study Design": ['studyAim', 'studyTypes'],
    "Conversational Interactions": ['interactionInputs', 'interactionOutputs', 'externalInt
erventions'],
    "System Features": ['deliveryChannels', 'targetInterventions', 'actionInterventions', '
additionalFeatures', 'embodiments']
}
```

*Listing 19 – This code snipset defines the **set of target fields** used in the **ECCO framework** to extract structured information from scientific articles.*

3.9.2 Enhanced Pipeline Architecture

We implemented the extraction through the EnhancedECCOPipeline class, which extends the base extraction capabilities with specialized strategies for complex eCCo characteristics:

```
pipeline = EnhancedECCOPipeline()
results = pipeline.process_single_paper_enhanced(
    pdf_path,
    model="chatgpt4o"
)
```

Listing 20- This code initializes the enhanced pipeline object

The enhanced pipeline incorporates official eCCo terminology into vector search operations, ensuring that context retrieval aligns with the framework's standardized vocabulary. We developed specialized extraction strategies for challenging fields that require domain-specific understanding.

3.9.3 Adaptive Context Retrieval

Our system retrieves context dynamically depending on the field. Simple information such as the publication year only needs one or two chunks, while more complex aspects like interaction inputs or interventions require broader context. For example:

```
k_strategy = {
    'articleYear': 1,      # Simple factual field
    'studyAim': 3,         # Moderate complexity
    'interactionInputs': 4,
    'externalInterventions': 5
}
```

Listing 21 - Context size strategy based on field complexity

3.9.4 Specialized Enhancement Strategies

When the initial extraction is weak, the system triggers enhanced strategies for key fields. For example, the **study aim** extraction uses eCCo elements to highlight agent design, health intervention needs, target populations, and evaluation methods.

3.9.5 eCCo Terminology Integration

> We already addressed this topic earlier in section **3.6.5 Context-Aware Retrieval System**, but **we mention it again here for the sake of completeness and coherence.**

To keep consistency, we embedded the official **eCCo glossary** into our keyword mappings. This ensures vector searches and retrieval always use the same vocabulary. For instance:

```
"interactionInputs": """
direct user input voice written text free options ...
"""
"targetInterventions": """
nutrition diet food choice
physical activity movement exercise
sleep rest behavior
hygiene personal practices
"""
```

Listing 22 - Standardized eCCo terminology mapping

Finally, we added quality checks. If a field extraction looks incomplete, the system re-runs it with an enhancement strategy and logs the improvement. This adaptive approach—combining context sizing, enhanced strategies, eCCo terminology, and model-aware prompts ensures consistent, high-quality extractions aligned with the framework.

4 Implementation

*The chapter includes technical configurations for local and cloud-based execution, prompt engineering for structured extraction, and **Streamlit**-based interface design. It also features screenshots of the working application (ecco_app_v1) to illustrate the user experience and interaction flow. Additionally, it covers the complete Docker-based deployment process that makes the system portable and reproducible across different environments. Each section is supported by code examples, file references, or configuration notes.*

Chapter's Content

4.1 Tools, Libraries, and Frameworks

The implementation of our system combines a diverse set of open-source tools, Python libraries, and LLM platforms. The core processing pipeline is developed in Python and orchestrated using modular scripts and function-based classes.

4.1.1 Core Application Framework

Streamlit serves as our primary web application framework. We chose Streamlit for its simplicity in creating interactive data applications and its seamless integration with Python machine learning libraries. The framework enables rapid development of the user interface while providing real-time feedback during document processing:

```
import streamlit as st
# File upload and processing interface
uploaded_files = st.file_uploader("Upload PDF papers", type="pdf", accept_multiple_files=Tr
ue)
if st.button("Start Processing"):
    process_selected_files(uploaded_files, mode="ecco")
```

*Listing 23 - This code defines a simple **user interface** in Streamlit that allows researchers to upload one or more PDF documents and then start the processing pipeline with a single click.*

4.1.2 Large Language Model Integration

We implemented multi-model LLM support through two primary channels. For cloud-based models, we use the **OpenAI API** to access GPT-4 and GPT-4o models, providing high-quality extraction capabilities. For local processing, we integrate **Ollama** to run models like TinyLlama, Phi3-Mini, and Mistral on local hardware.

The **LangChain** framework orchestrates our LLM interactions, providing abstraction layers for different model types and enabling consistent prompt management across various AI services.

4.1.3 PDF Processing and Text Extraction

PyPDF2 handles our primary PDF text extraction needs, converting research papers into processable text format. We chose this library for its reliability with academic documents and its ability to preserve document structure information.

For enhanced metadata extraction, we integrate **GROBID** , a machine learning service that extracts structured bibliographic information from PDF documents. GROBID runs as a separate service, processing documents to identify titles, authors, abstracts, and references.

4.1.4 Vector Storage and Similarity Search

ChromaDB provides our vector database capabilities, storing document embeddings for intelligent context retrieval. We use **HuggingFace Embeddings** with the sentence-transformers/all-MiniLM-L6-v2 model to generate semantic embeddings of document chunks:

```
from langchain_community.vectorstores import Chroma
from langchain.embeddings import HuggingFaceEmbeddings
embeddings = HuggingFaceEmbeddings(model_name="sentence-transformers/all-MiniLM-L6-v2")
vectorstore = Chroma(persist_directory=vector_dir, embedding_function=embeddings)
```

Listing 24 - Creating and loading a vector database with Chroma

4.1.5 Data Processing and Analysis

Pandas handles our data manipulation and analysis tasks, from organizing extraction results to generating export files. We use pandas DataFrames throughout the application for consistent data handling and to enable various export formats (CSV, Excel, JSON).

JSON serves as our primary data interchange format, storing prompts, configuration settings, and extraction results in structured, human-readable files.

4.1.6 Containerization and Deployment

Docker and **Docker Compose** provide our containerization solution. We designed a multi-service architecture that includes the main Streamlit application, GROBID service, and Ollama for local models:

```yaml
services:
  ecco-app:
    build: .
    ports:
      - "8501:8501"
  grobid:
    image: lfoppiano/grobid:0.8.0
    ports:
      - "8070:8070"
  ollama:
    image: ollama/ollama
    ports:
      - "11434:11434"
```

Listing 25 - Docker Compose services for the ECCO system

4.2 Local and Cloud-based Model Integration

The system supports two modes of language model integration: local deployment using lightweight open-source LLMs, and remote execution via OpenAI's cloud-based API.

Local Models: We tested several local models including TinyLLaMA, Mistral, and Phi-3 Mini, executed through the Ollama framework. These models are downloaded as GGUF files and run locally on the user's machine, enabling offline extraction and reduced latency. This setup is especially useful for cost-sensitive scenarios or environments with limited internet access. The integration is managed via subprocess calls to ollama-run and supports dynamic prompt injection.

Cloud-based Models: For higher accuracy and robustness, the system also integrates OpenAI's GPT-4o and GPT-4o-mini through API calls. This mode is particularly valuable for complex eCCo fields requiring nuanced language understanding. API access is authenticated via keys stored in environment variables, and rate limits are handled with built-in retry mechanisms.

4.2.1 Adaptive Model Selection and Prompt Strategies

We implemented an adaptive system that adjusts extraction strategies based on model capabilities. Local models receive simplified prompts optimized for their parameter constraints, while cloud models get detailed instructions with complex reasoning requirements:

```python
def get_model_category(model):
    if model in LOCAL_MODELS:
        return "local"
    elif model in CLOUD_MODELS:
        return "cloud"
    else:
        return "unknown"
```

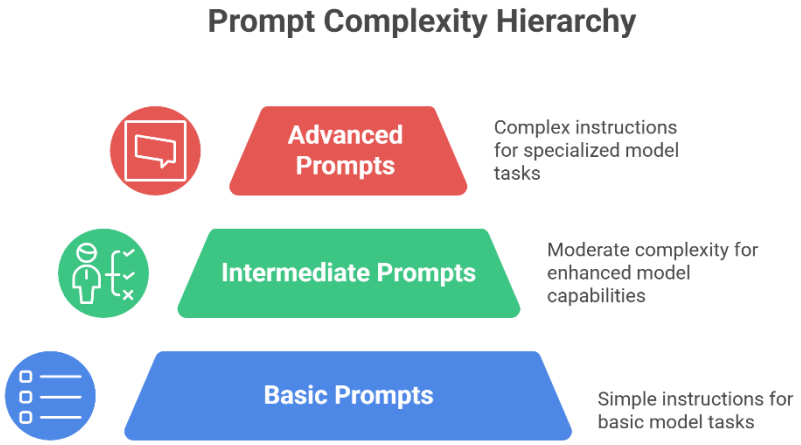
*Listing 26 - This function classifies a given model into **local** or **cloud-based** depending on predefined lists*

4.3 Prompt Engineering Strategy

We developed a sophisticated prompt engineering strategy that adapts to different model capabilities while maintaining extraction quality across our entire system. This approach ensures optimal performance whether using lightweight local models or powerful cloud-based systems.

4.3.1 Three-Levels Adaptive Prompt Architecture

We designed three distinct prompt complexity levels based on empirical testing with different model types. Each tier targets specific model capabilities and resource constraints:



This categorization enables automatic prompt selection based on the chosen model, ensuring each system receives instructions optimized for its capabilities.

Figure 6 - Prompt Complexity Hierarchy

> **Tailoring Communication to Each Model**
>
> One of the most important insights from our testing was that different models need different communication styles. You can't use the same prompt for TinyLlama and GPT-4o. It's like giving the same instructions to an elementary student and a PhD researcher.

The three levels of prompt complexity:

Ultra-Simple Prompts (TinyLlama):

```
{
    "articleTitle": "Article title:",
    "authorName": "First author name:",
    "studyAim": "What does this study want to do?"
}
```

Simple Prompts (Phi3-Mini, Mistral):

```
{
    "articleTitle": "What is the title of this article?",
    "authorName": "Who is the first author?",
    "studyAim": "What is the main goal of this study?"
}
```

Complex Prompts (GPT-4o, GPT-4o-Mini):

```
{
    "articleTitle": "Extract the complete title of this research article + exemple +
explication ",
    "studyAim": "Analyze the research objectives and methodology+ exemple + explication."}
```

Automatic Model Detection
The system automatically detects which model you're using and selects the appropriate prompt style:

```
def get_model_category(model):
    """Categorize models to choose the right prompt complexity."""
    if model in ["tinyllama"]:
        return "ultra_simple"
    elif model in ["phi3mini", "mistral"]:
        return "simple"
    elif model in ["chatgpt4o", "chatgpt4omini"]:
        return "complex"
    else:
        return "ultra_simple"  # Safe default
```

*Listing 27- This function classifies each model into a category that determines the **complexity of the prompts** to be used. Since different models have varying capabilities, this ensures that the system sends only the level of instruction that the model can reliably handle.*

This ensures that every model gets instructions it can actually follow effectively.

4.3.2 Domain-Specific Optimization

Our prompts incorporate eCCo framework terminology and systematic review criteria specific to conversational e-coaches in healthcare. We embed domain knowledge directly into prompt structure:

For PRISMA eligibility assessment, we created specialized prompts that guide models through systematic review criteria evaluation. For eCCo extraction, we integrate official framework terminology to ensure consistent characteristic identification.

4.3.3 Response Format Standardization

We designed prompts to encourage standardized response formats across all model types. This includes explicit instructions for answer structure and format constraints:

```
{
  "deliveryChannels": "Where does it run? Pick from: DESKTOP_APP, WEBSITE_APP, MOBILE_APP,
MESSAGING_APP, STANDALONE_DEVICE, MOBILE_ROBOT",
  "embodiments": "What does it look like? Pick from: VIRTUAL_HUMAN, VIRTUAL_NO_HUMAN, PHYSI
CAL_HUMAN, PHYSICAL_NO_HUMAN"
}
```

Standardized formats enable consistent post-processing and reduce parsing complexity. We provide explicit answer choices where appropriate to minimize ambiguity in model responses.

4.4 Application User Interface (Streamlit)

We built our user interface using Streamlit to create an intuitive web application that makes automated literature review accessible to researchers without technical expertise. The application serves as a comprehensive tool for systematic literature reviews focused on conversational agents in healthcare, implementing the eCCo framework for consistent characteristic extraction.

4.4.1 Application Purpose and Key Features

Our application addresses the growing challenge of conducting systematic literature reviews in the rapidly expanding field of conversational e-coaches and digital health interventions. The tool automates three critical aspects of literature review:

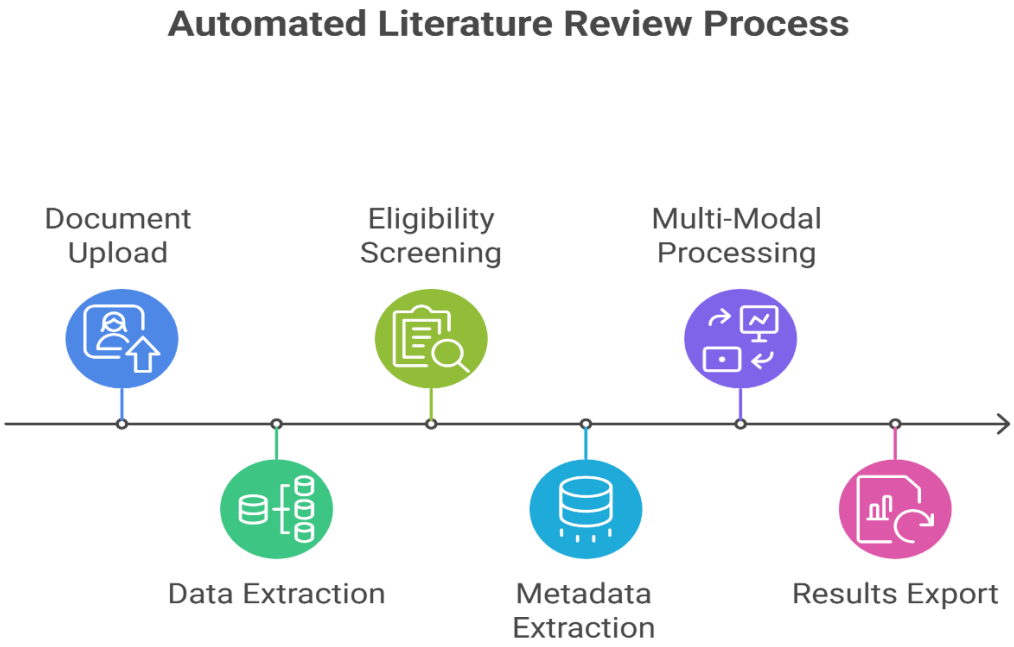


*Figure 7- **Workflow of the ecco_app_v1 application.** The application guides users through the complete pipeline: document upload, eligibility screening, data and metadata extraction, multi-modal processing, and final export of structured results.*

Data Extraction Capabilities: The application automatically extracts 15 key characteristics defined by the eCCo framework, including study objectives, interaction modalities, target interventions, delivery channels, and system embodiments. This eliminates the manual process of reading through hundreds of papers to identify relevant characteristics.

Eligibility Screening (PRISMA): The system implements automated PRISMA-based eligibility assessment, evaluating papers against eight systematic review criteria. This includes language verification, publication type validation, population focus, health domain relevance, and conversational agent identification. The tool can process large document collections and automatically filter relevant papers for further analysis.

Bibliographic Metadata Extraction: Through GROBID integration, the application extracts structured bibliographic information including titles, authors, affiliations, abstracts, publication venues, and reference lists. This provides comprehensive document metadata without manual data entry.

Multi-Modal Processing: Users can combine these capabilities in different workflows depending on their research needs. The application supports standalone extraction modes or integrated pipelines that chain multiple analysis types for comprehensive literature processing.

The interface guides users through the entire process from document upload to results export, making advanced AI-powered literature analysis accessible to researchers without technical expertise.

4.4.2 Application Layout and Navigation

The application opens with a clean, organized layout featuring a main content area and a configuration sidebar. We designed the navigation to be self-explanatory, with clear sections for each stage of the review process.

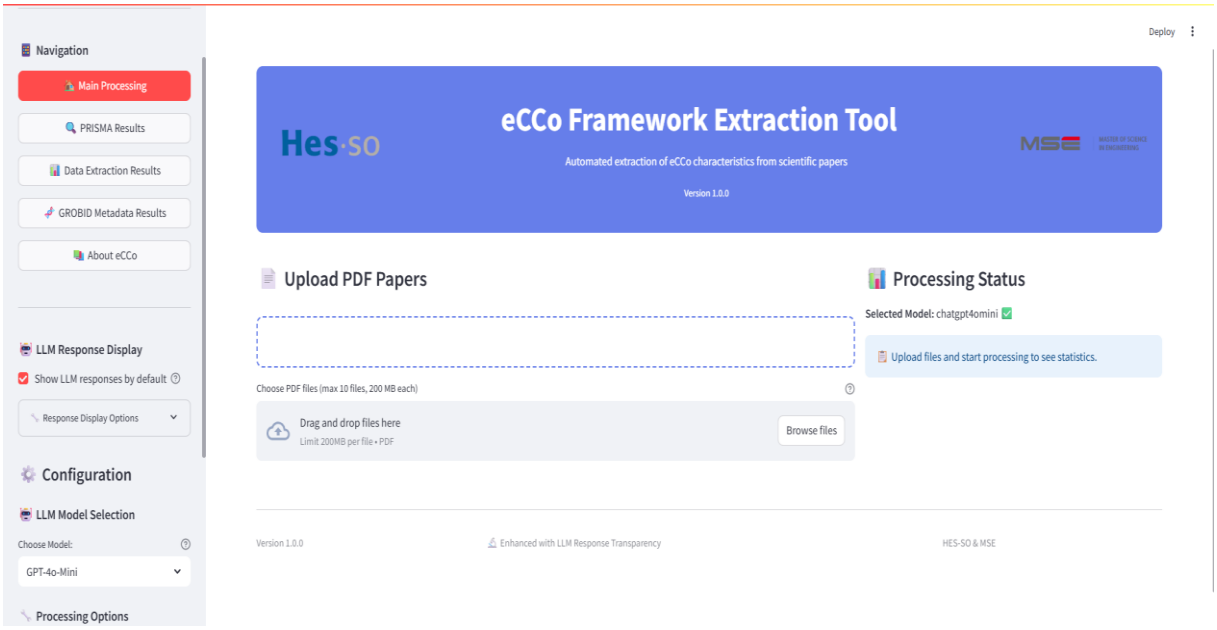


*Figure 8 - **Main interface of the ecco_app_v1 application.** The interface allows users to upload multiple PDF papers, configure processing options, and select the preferred language model (local or cloud). A navigation panel provides access to PRISMA results, metadata extraction, and eCCo-specific outputs, while the processing status area displays progress and selected model information in real time.*

The sidebar navigation menu allows users to switch between different pages:

- **Main Processing**: Upload documents and run extractions
- **PRISMA Results**: View eligibility assessment results
- **eCCo Results**: Examine extracted characteristics
- **GROBID Results**: Access bibliographic metadata
- **Model Status**: Check available language models
- **About**: Learn about the eCCo framework and project details

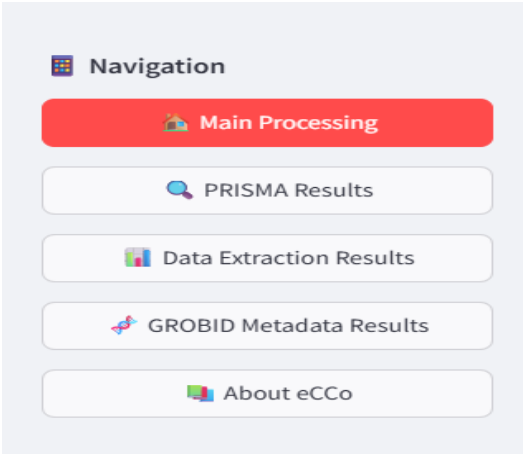


Figure 9 - Navigation panel of the ecco_app_v1 application

Users can navigate freely between sections without losing their work. The interface remembers uploaded files, processing results, and preferences throughout the session.

4.4.3 Document Upload Process

The main page starts with a simple file upload area where users drag and drop PDF files or browse their computer. The system accepts multiple PDF documents simultaneously and shows progress during upload.

Once files are uploaded, the interface displays a list of documents with their sizes and processing status. Users can add more files later or clear all documents to start fresh. The system validates files automatically and warns about any issues like unsupported formats or files that are too large.

4.4.4 LLM Model Selection

The sidebar includes a model selection dropdown that shows all available language models. We organized models into two categories that users can easily understand:

Local Models (run on your computer):
- TinyLlama: Fastest processing, basic quality
- Phi3-Mini: Balanced speed and quality
- Mistral: Higher quality, slower processing

Cloud Models (require internet and API key):
- GPT-4o-Mini: Cost-effective, high quality
- GPT-4o: Best quality, higher cost

The interface shows real-time status for the selected model, indicating whether it's available and ready to use. Users receive immediate feedback if a model is unavailable, with suggestions for alternatives.

4.4.5 Processing Mode Selection

We designed five processing modes to accommodate different research needs. Users select their preferred mode through clear options with descriptions:

1. **GROBID Metadata Extraction Only**: Fast extraction of basic bibliographic information like titles, authors, and abstracts using machine learning. No LLM required.
2. **eCCo Extraction Only**: Detailed analysis of conversational agent characteristics using the selected LLM. Takes longer but provides comprehensive eCCo framework data.
3. **PRISMA Eligibility Assessment Only**: Evaluates papers against systematic review criteria to determine which documents are relevant for inclusion.
4. **Complete Pipeline (PRISMA → eCCo)**: First assesses eligibility, then extracts eCCo characteristics only from eligible papers. This saves time and focuses analysis on relevant documents.
5. **Full Pipeline (GROBID + PRISMA + eCCo)**: Comprehensive analysis combining all three extraction methods for maximum information gathering.

Each mode includes an estimated processing time and explanation of what results users can expect.

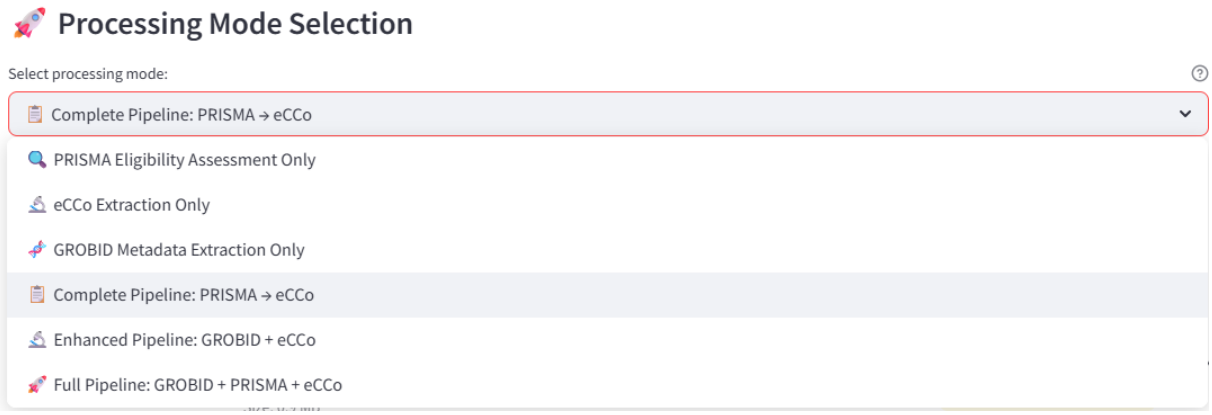


*Figure 10 - **Processing mode selection in ecco_app_v1.** This section allows users to choose how the system processes uploaded papers. Several modes are available: PRISMA-only screening, eCCo characteristic extraction, GROBID metadata parsing, or combinations of these. Full and enhanced pipelines integrate all modules (PRISMA, GROBID, and eCCo) for end-to-end automated processing, while lighter modes enable focused analysis depending on research needs.*

4.4.6 Processing Controls and Options

Before starting extraction, users can adjust several options in the sidebar:

- **Reuse existing extractions**: Skip re-processing files that were already analyzed
- **Enhanced extraction mode**: Use improved prompts with eCCo terminology for better results

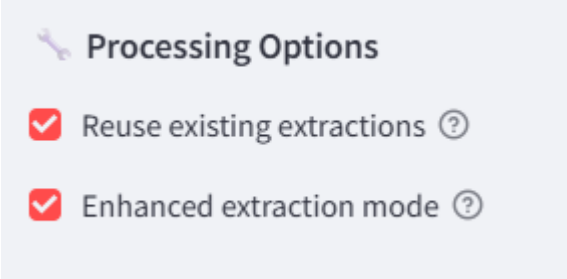


Figure 11 - Processing options in ecco_app_v1.
This section provides additional controls for extraction.

The interface provides buttons to start processing and shows real-time progress with status updates. Users can monitor which documents are being processed and see completion percentages.

4.4.7 Results Display and Analysis

After processing completes, results appear in organized sections based on the selected mode. The interface provides multiple ways to view data:

- **Summary tables** showing key information for all documents
- **Detailed individual views** with complete extraction results for each paper
- **Category-based organization** grouping eCCo characteristics by type
- **Statistics and metrics** showing processing completion rates and data quality

For PRISMA results, the interface clearly indicates which papers are eligible for inclusion and provides detailed reasoning for each decision.

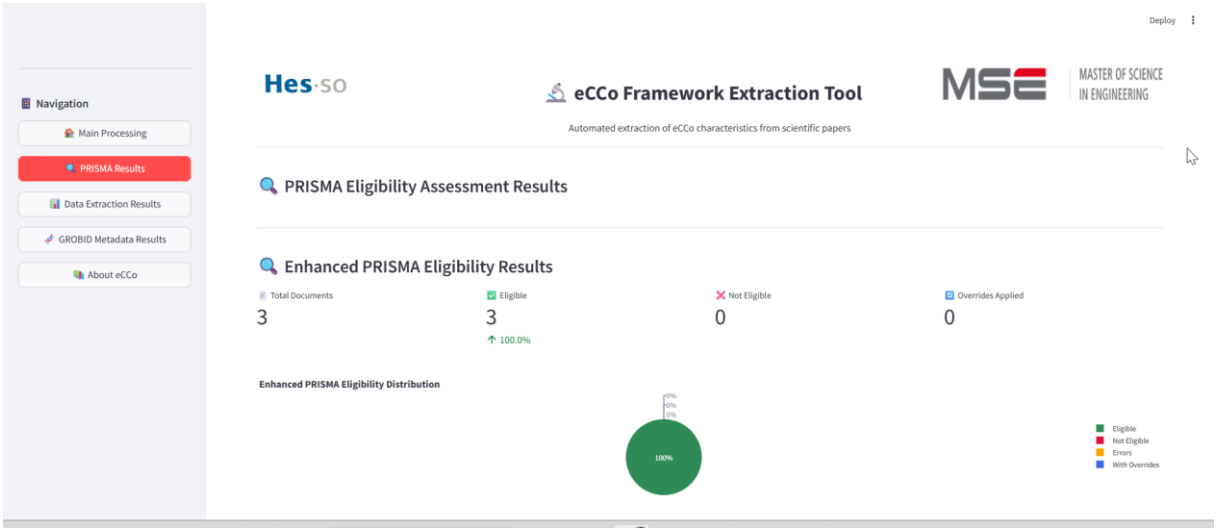


Figure 12 - PRISMA Eligibility Assessment Results. This interface displays the outcome of the automated PRISMA-based screening. It summarizes the number of uploaded documents assessed as eligible or not eligible, with a clear distribution visualization. In this example, all three uploaded papers were classified as eligible.

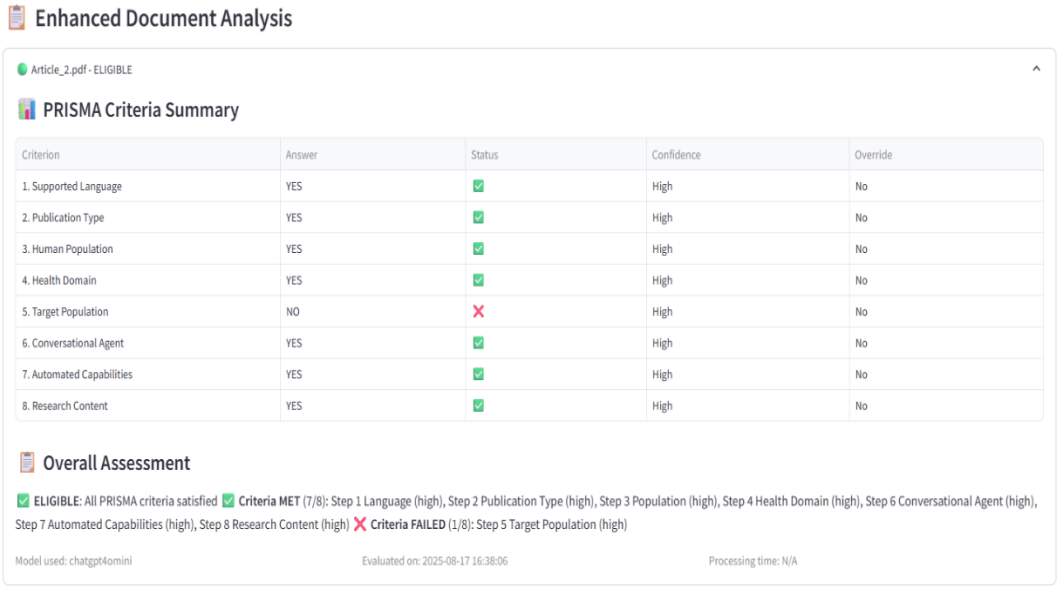


Figure 13 - Enhanced Document Analysis (PRISMA Criteria Summary). This view provides a detailed breakdown of the PRISMA eligibility assessment for a single paper. Each criterion is evaluated with its answer, status, and confidence score. In this example, 7 out of 8 criteria were satisfied, leading to an overall classification of eligible despite the failure of the target population criterion

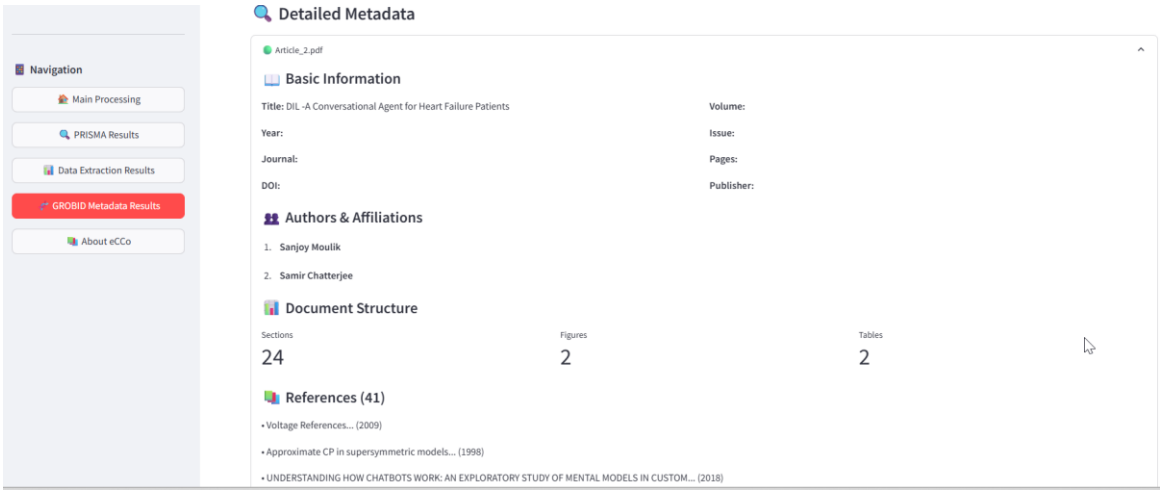


Figure 14 - GROBID Metadata Extraction Results. This interface displays detailed bibliographic metadata automatically extracted from the uploaded paper using GROBID

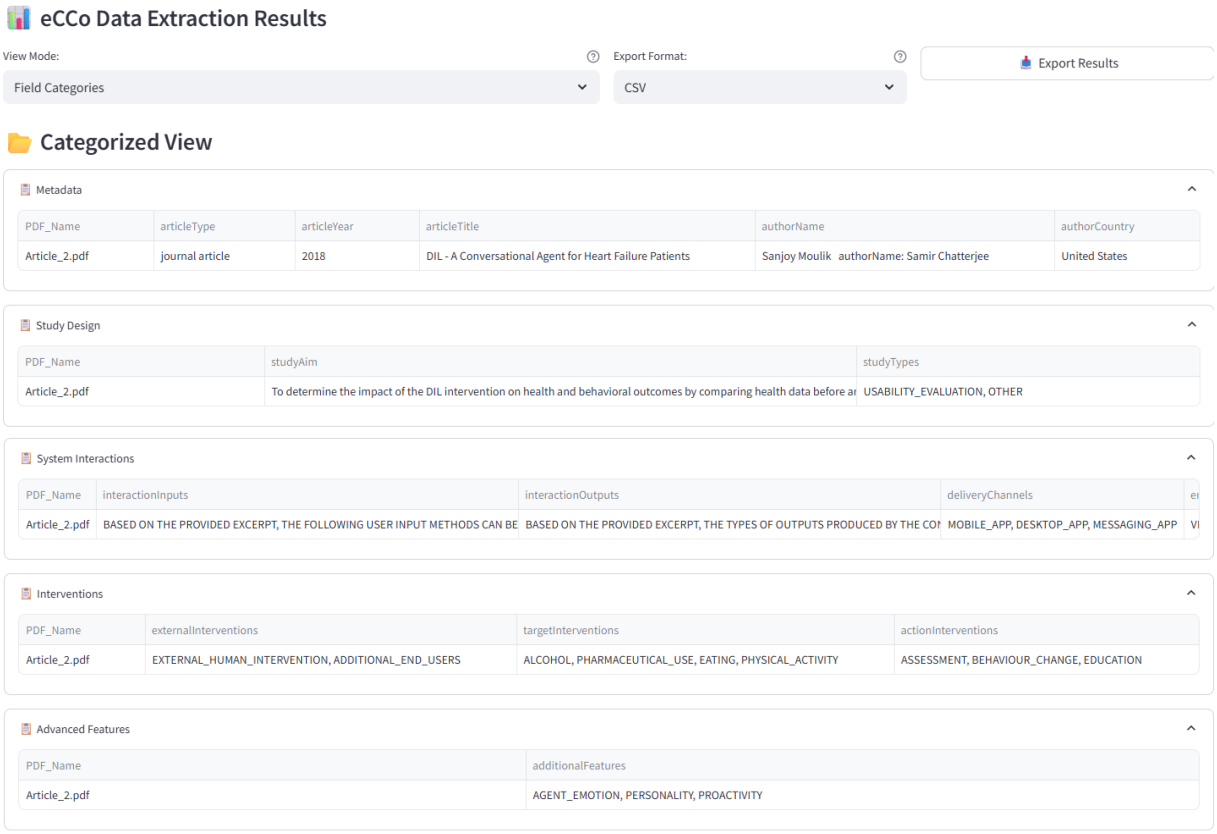


Figure 15 - eCCo Data Extraction Results by use llm's. This interface shows the structured extraction of eCCo framework fields from scientific articles. The results are organized into categories such as Metadata (e.g., title, year, authors, country), Study Design (e.g., study aim, study types), interventions and other characteristics. Users can choose the view mode, export format (e.g., CSV), and download the extracted data for further analysis.

4.5 Docker and Deployment

To ensure reproducibility and simplify setup across environments, the entire system is containerized using Docker. This allows the application and all its dependencies to run identically on any machine, regardless of the local configuration.
The Docker architecture consists of the following components:

- A base image with Python 3.10 and system dependencies for PDF parsing and NLP libraries
- A Dockerfile that installs project-specific requirements listed in requirements.txt
- Environment variables (e.g., API keys) defined in a .env file for secure access to external services
- A docker-compose.yml file for orchestrating services such as the Streamlit frontend and GROBID backend

This setup supports both development and production modes. In production, Docker ensures isolated execution of the pipeline while maintaining compatibility with hardware constraints (e.g., CPU/GPU). It also facilitates deployment on cloud platforms or academic servers without additional installation.
The deployment process is automated through CLI scripts that build the image, mount data volumes, and expose the application via local ports. This ensures fast and consistent startup across test environments and user machines.

> **Dockerization**
>
> The initial dockerization process can take up to **20 minutes**, especially during the installation of large dependencies like **GROBID** and **Ollama**, which require multiple system-level packages and model downloads. This is a one-time setup, after which container startup becomes much faster.

5 Results

This chapter presents the key findings from our experiments and system evaluations. It summarizes the performance of each component, metadata extraction, semantic retrieval, and eCCo field extraction and benchmarks the output quality of various LLMs.. Visualizations are used to compare models, highlight error patterns, and analyse article difficulty. A detailed interactive dashboard has been developed to explore the results more deeply and is available in the project folder under the file name: ***dashboard_ecco_english.html****.*

Chapter's Content

5.1 Evaluation Framework and Methodology

5.1.1 Dataset Preparation and Annotation (Ground Truth Creation)

We established our evaluation framework around the eCCo platform, which provides standardized annotations for conversational agents in health research. The evaluation process aimed to assess how well different LLMs could extract structured information compared to human experts.

Data Source and Selection

We used the eCCo platform to download CSV files containing research articles focused on **pharmaceutical interventions**. These articles had been previously annotated by domain experts according to the eCCo framework, which defines specific categories for conversational agent characteristics like interaction types, delivery channels, and target interventions.

Ground Truth Creation

The human annotations from the eCCo platform served as our ground truth. These annotations were created by researchers who manually extracted information according to standardized eCCo criteria. Each article was annotated with 15 key fields covering metadata (title, authors, year), study objectives, and eCCo-specific characteristics.

Figure 16- This figure shows the evaluation process. We downloaded CSV files with human annotations from the eCCo platform, used them as the ground truth, converted them to JSON, and then compared them with the results produced by the LLMs.

5.2 Evaluation Metrics

We developed a comprehensive evaluation framework to assess the performance of different LLMs in extracting information from scientific literature for systematic reviews. Our evaluation system addresses the specific challenges of conversational AI research by implementing domain-specific metrics tailored to the eCCo (enhanced Checklist of Conversational Objects) framework.

5.2.1 Text Similarity Metrics

Cosine Similarity: We use cosine similarity as our primary metric for textual field evaluation. This metric measures the cosine of the angle between two vectors in the n-dimensional space, providing a score between 0 and 1 where 1 indicates perfect similarity.

For preprocessing, we remove stop words and apply tokenization to focus on meaningful terms. The cosine similarity formula we implement is:

$$\text{cosine\_similarity} = (A \cdot B) / (||A|| \times ||B||)$$

Where A and B are the TF-IDF vectors of the compared texts.

Coverage Score: For categorical fields, we developed a coverage score that rewards complete inclusion of ground truth elements without penalizing additional information. This approach recognizes that LLMs might extract more comprehensive information than the minimal ground truth.

$$\text{coverage\_score} = 1.0 \text{ if } GT \subseteq \text{LLM\_output else } |GT \cap \text{LLM\_output}| / |GT|$$

Enhanced Keyword Similarity: We created a domain-specific keyword matching system using 220+ specialized terms from the eCCo framework, organized into seven categories:

- Conversational entities (25% weight)
- Health and wellbeing domain (20% weight)
- Research development actions (20% weight)
- Interaction modalities (15% weight)
- Technology methods (10% weight)
- Capabilities features (7% weight)
- Application context (3% weight)

5.2.2 Tri-Dimensional Score Calculation Methodology

Our novel tri-dimensional evaluation approach segments LLM performance into three distinct dimensions, providing granular insights into model capabilities.

Dimension 1: Metadata Extraction (Factual Accuracy)
This dimension evaluates the model's ability to extract basic bibliographic information:

Fields	Scoring	Weight
- articleType, - articleYear, - articleTitle, - authorName, - authorCountry	- Exact match for structured data, - normalized matching for countries	- Uniform weighting across fields

Dimension 2: StudyAim Analysis (Semantic Understanding)
This dimension assesses comprehension of research objectives using our hybrid scoring system:

Fields	Scoring	Weight
- studyAim (primary research objective)	- Co-enhanced hybrid methodology	- Single field with high impact on overall assessment

Dimension 3: eCCo Characteristics (Domain Expertise)
This dimension measures extraction of conversational AI-specific features:

Fields	Scoring	Weight
- interactionInputs, - interactionOutputs, - externalInterventions, - studyTypes, - deliveryChannels, - targetInterventions, - actionInterventions, - additionalFeatures, - embodiments	- Coverage-based evaluation with domain-specific terminology	- Distributed across multiple specialized fields

Hybrid Scoring for StudyAim
We developed a sophisticated hybrid scoring system specifically for studyAim evaluation:

$$\text{hybrid\_score} = 0.20 \times \text{cosine\_similarity} + 0.40 \times \text{keyword\_similarity} + 0.40 \times \text{semantic\_similarity}$$

Example Calculation:

Ground Truth: "Develop a conversational agent to support diabetes management"
LLM Output: "Create an AI chatbot for diabetic patient care assistance"

Cosine Component (20%): 0.65 (good lexical overlap)
Keyword Component (40%): 0.85 (strong eCCo terminology match)
Semantic Component (40%): 0.78 (excellent domain understanding)

Final Score: $0.20 \times 0.65 + 0.40 \times 0.85 + 0.40 \times 0.78 = 0.782$

Performance Classification
We use consistent thresholds across all metrics:
🟢 **Excellent**: ≥ 0.90, 🟡 **Good**: ≥ 0.70, 🟠 **Acceptable**: ≥ 0.50, 🔴 **Poor**: < 0.50

Tri-Dimensional Score Aggregation
For each dimension, we calculate:

$$\text{dimension\_score} = \Sigma(\text{field\_scores}) \,/\, \text{number\_of\_evaluated\_fields}$$

Where excluded fields (empty or invalid data) are removed from both numerator and denominator to ensure fair comparison across models.

Example Tri-Dimensional Results:

Model: ChatGPT-4o
├── Metadata Score: 0.891 (Excellent)
├── StudyAim Score: 0.756 (Good)
└── eCCo Score: 0.723 (Good)

Overall Assessment: Strong performer with excellent factual extraction

5.3 eCCo Extraction Results

This section evaluates how well each language model extracted structured eCCo fields from scientific papers. We looked at three types of results: metadata (like title or year), study aim, and key eCCo categories (such as agent role or delivery channel).

ChatGPT-4o and **Mistral** gave the best results overall, both with average scores around 0.66. **ChatGPT-4o** was best at metadata, while **Mistral** was especially strong on complex eCCo fields. **ChatGPT-4o-mini** had good performance, close to its larger version.

Among local models, **Phi-3 Mini** gave balanced results, while **TinyLLaMA** performed poorly in all categories (overall score below 0.4). These results highlight that larger or cloud-based models still provide better extraction quality, especially for detailed task

5.4 Benchmarking Analysis of LLMs

To understand the relative strengths of each language model, we compared their average performance across the three main evaluation categories: metadata extraction, study aim identification, and structured eCCo field extraction.

The comparison was visualized using a bar chart titled *"Tri-Dimensional Performance"*, which shows how each model performs across the three dimensions. As observed, ChatGPT-4o leads in metadata, while Mistral dominates in eCCo field extraction. ChatGPT-4o-mini provides balanced results slightly below its larger counterpart. TinyLLaMA, by contrast, struggles in all three categories.

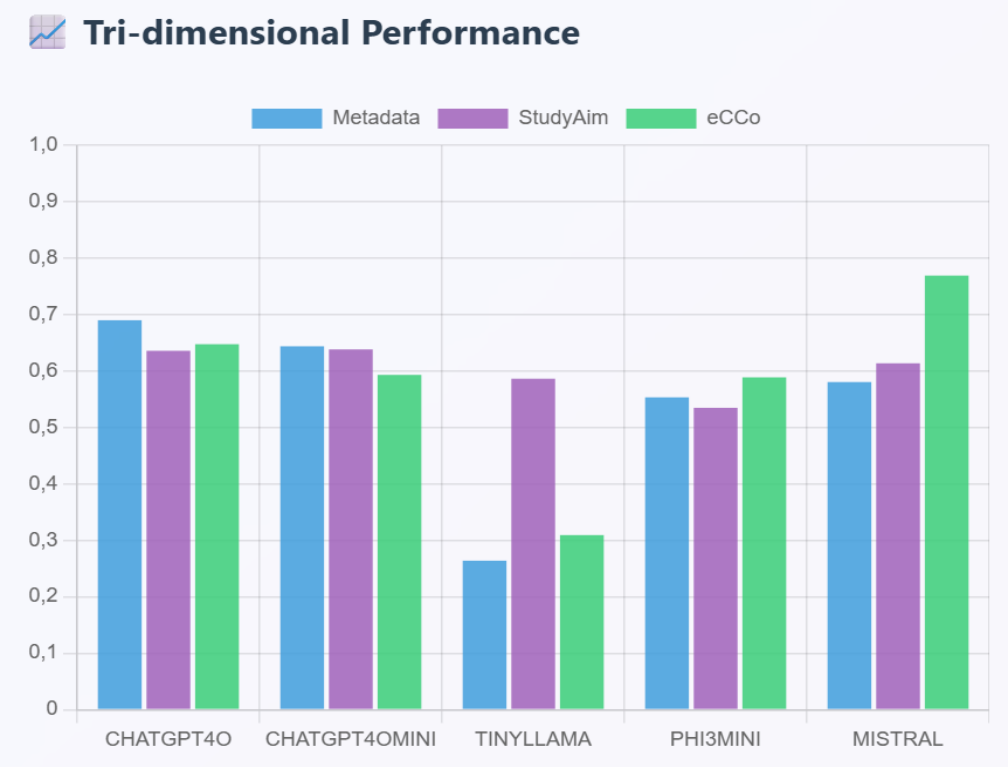


Figure 17 - Tri-Dimensional Performance of LLMs
This bar chart compares the average performance of five language models (ChatGPT-4o, ChatGPT-4o-mini, TinyLLaMA, Phi-3 Mini, and Mistral) across three eCCo extraction dimensions: metadata, study aim, and eCCo fields.

This benchmarking confirms that cloud-based models like GPT-4o provide robust, high-quality outputs across dimensions, but local models such as Mistral and Phi-3 Mini can offer competitive performance, especially with optimized prompts. The analysis highlights the trade-offs between speed, cost, and quality—guiding future model selection for similar extraction tasks.

5.4.1 Model Consistency Analysis

The Model Consistency graph evaluates the reliability of each LLM by measuring how consistently they perform across different articles. This metric is crucial for understanding which models provide stable and predictable results in real-world literature review scenarios.

Consistency Calculation
We calculate consistency using the coefficient of variation approach:

```
Consistency = 1 - (Standard Deviation / Mean Score)
```

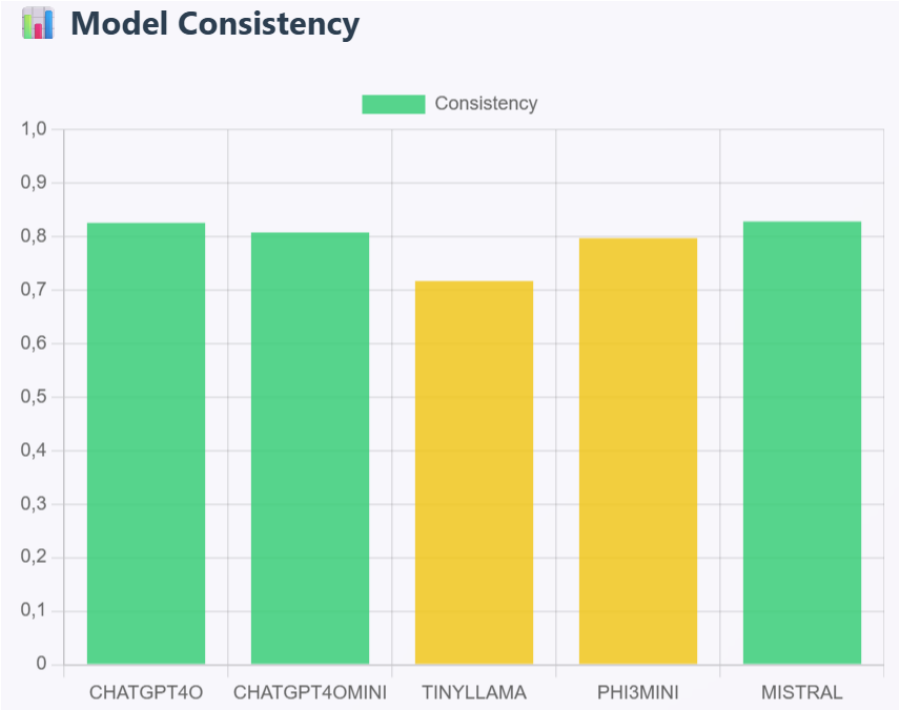


Figure 18 - This bar chart presents the consistency levels of five LLMs when extracting structured data from scientific texts. ChatGPT-4o, Mistral, and ChatGPT-4o-mini exhibit high consistency across multiple documents, with scores above 80%. In contrast, Phi-3 Mini and TinyLLaMA show more variability in their outputs. These results reflect how reliably each model performs extraction tasks under similar conditions, supporting better-informed choices for production use.

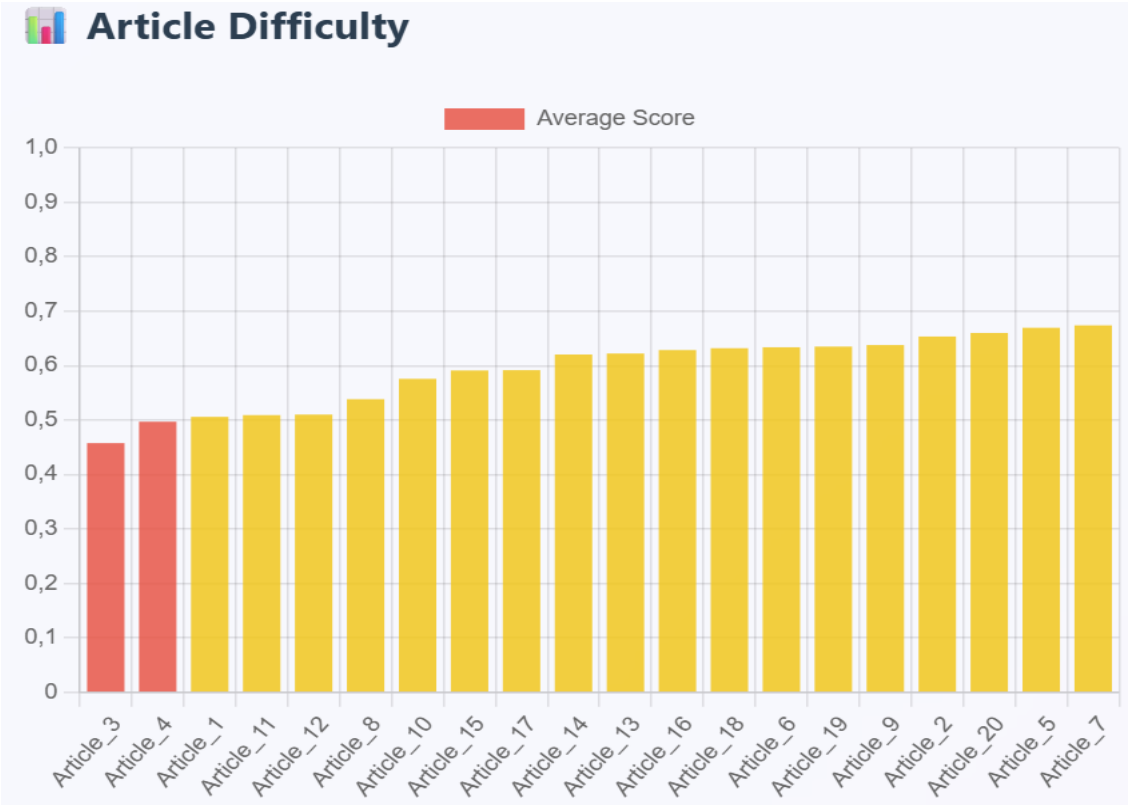


Figure 19 - This bar chart illustrates the relative difficulty of each article in the benchmarking set, based on the average extraction score across all models. Articles are ranked from most to least challenging. Articles 3 and 4, shown in red, had the lowest average scores (below 0.5), indicating higher complexity or ambiguity in their content. The remaining articles, displayed in yellow, were easier to process, with average scores increasing steadily from left to right. This distribution helps identify which documents pose more challenges for LLM-based extraction.

Figure 20 - This heatmap provides a detailed comparison of how each language model performed across the 20 benchmarked scientific articles

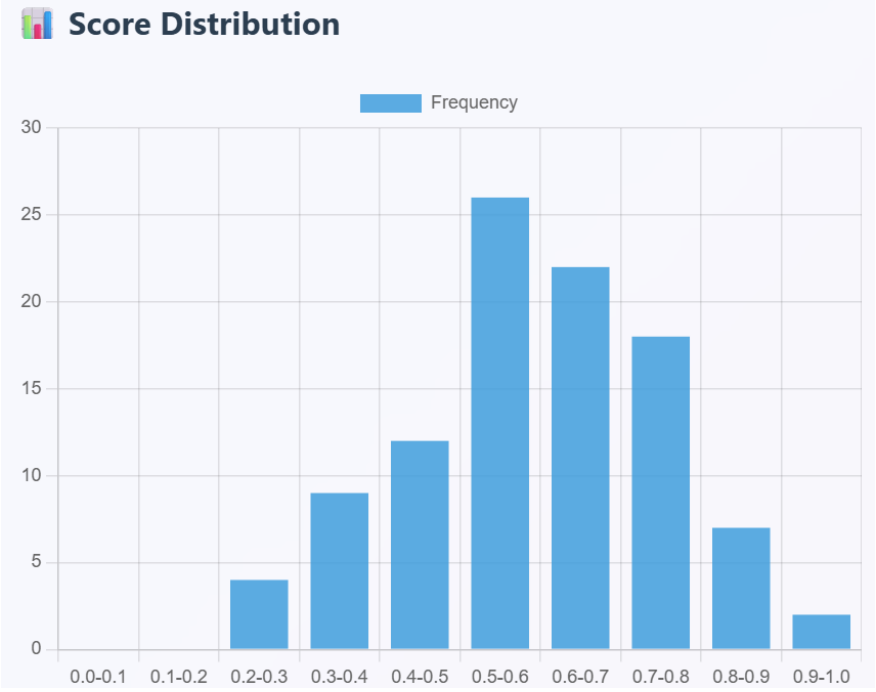


Figure 21 - This histogram presents the distribution of all prediction scores generated by the LLMs across metadata, study aim, and eCCo fields. The majority of predictions fall between 0.5 and 0.7, indicating moderate extraction quality. Fewer outputs reach the extremes (below 0.3 or above 0.9), highlighting that most models perform consistently within a central performance range.

5.4.2 Scoring Methods Comparison Analysis

1. Simple Average
Equal weighting across all three dimensions:

$$\text{Score} = (\text{Metadata} + \text{StudyAim} + \text{eCCo}) \div 3$$
Weight Distribution: 33.3% each dimension

2. Weighted 60% eCCo
Fixed weighting emphasizing domain expertise:

$$\text{Score} = (\text{Metadata} \times 0.20) + (\text{StudyAim} \times 0.20) + (\text{eCCo} \times 0.60)$$
Weight Distribution: 20% Metadata, 20% StudyAim, 60% eCCo

3. Adaptive (Expert) Weighting
Dynamic weighting based on eCCo performance level:

High eCCo Performance (≥0.7):
Metadata: 10%, StudyAim: 20%, eCCo: 70%,
- *Rewards models excelling in domain expertise*

Medium eCCo Performance (0.4-0.7):
Metadata: 15%, StudyAim: 25%, eCCo: 60%
- *Balanced approach for moderate performers*

Low eCCo Performance (<0.4):
Metadata: 25%, StudyAim: 35%, eCCo: 40%
- *Emphasizes basic capabilities when domain expertise is weak*

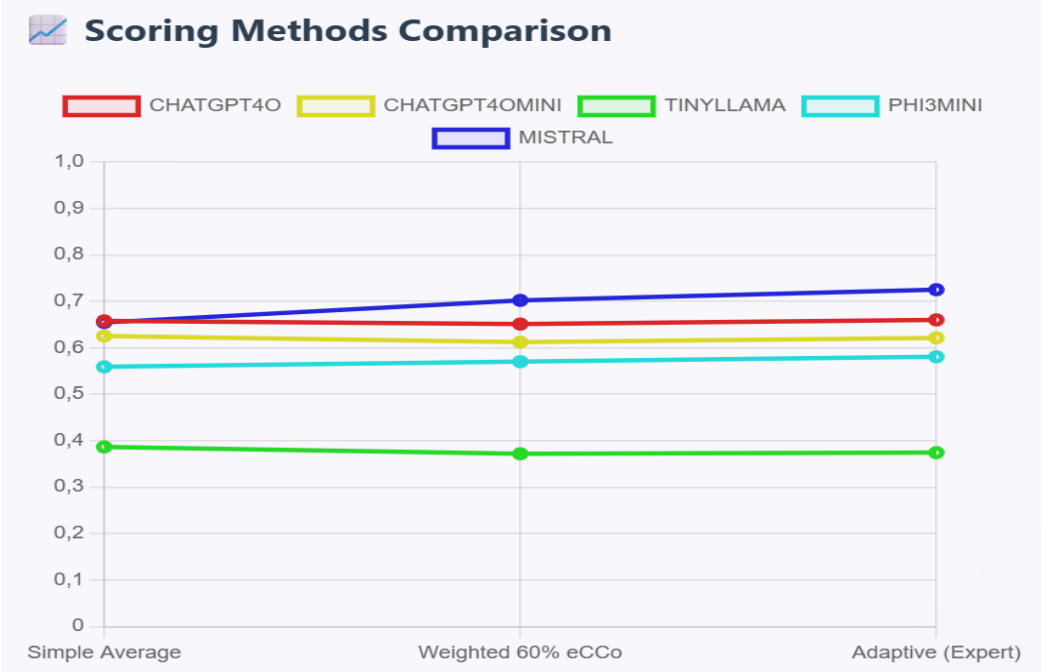


Figure 22 - This graph compares five llms (ChatGPT-4o, ChatGPT-4o-mini, Mistral, TinyLLaMA, and Phi3 Mini) using three different scoring strategies: simple average, fixed eCCo-weighted (60%), and adaptive expert-based weighting.

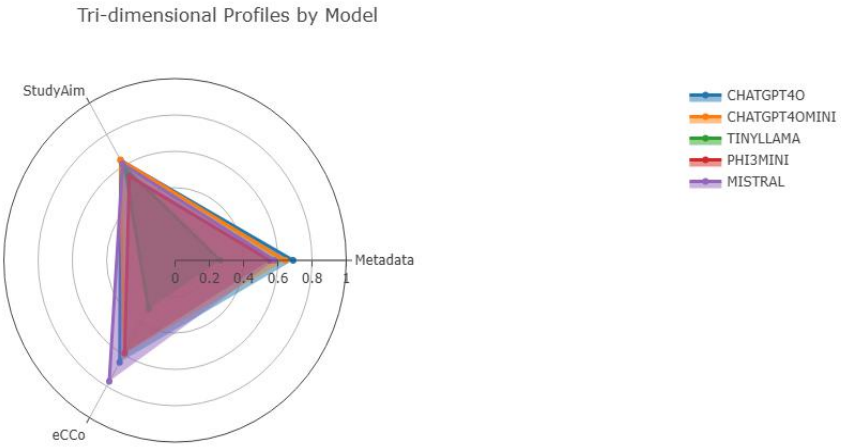


*Figure 23 - This radar chart visualizes each model's performance across three evaluation dimensions: **Metadata**, **StudyAim**, and **eCCo**. GPT-4o and Mistral show balanced profiles, with GPT-4o excelling in metadata and Mistral leading in structured eCCo field extraction. ChatGPT4o-mini closely mirrors GPT-4o's shape but with slightly lower scores. Phi3Mini provides moderate consistency, while TinyLLaMA shows limited capability, especially on complex eCCo fields.*

6 Discussion

This chapter reflects on the main findings, challenges, and lessons learned from the development and evaluation of our system. It discusses key results from the benchmarking experiments, highlights technical and methodological limitations, and introduces strategies to mitigate these issues. We also explore the complementary roles of humans and AI in systematic literature reviews (SLRs), showing where collaboration is necessary. Finally, this chapter outlines several future directions to improve system performance, scalability, and trustworthiness, both technically and ethically.

Chapter's Content

6.1 Key Findings

The benchmarking results revealed several important trends. First, models such as ChatGPT-4o and Mistral demonstrated strong performance across all three extraction dimensions. This suggests that combining large-scale pretrained models with our structured prompts produces reliable and accurate outputs. In parallel, the GROBID engine showed excellent performance for metadata extraction especially for titles, authors, affiliations, and date when the input PDFs were well-formatted. This reinforces the importance of hybrid strategies combining rule-based and LLM-based components. that combining large-scale pretrained models with our structured prompts produces reliable and accurate outputs.

Second, models like TinyLLaMA, while significantly faster and resource-efficient, showed weaker extraction accuracy and lower consistency, especially on metadata fields. This highlights the trade-off between lightweight deployment and extraction quality.

Third, the adaptive scoring method, which assigns more weight to eCCo when it is well-extracted, provided a fairer evaluation mechanism. It acknowledges that extracting nuanced intervention characteristics is inherently harder than simple metadata parsing.

Finally, the visualizations and error analysis confirmed that certain articles posed greater challenges to all models. Article complexity and format inconsistency likely contributed to these drops in performance.

These findings validate the benefit of our modular pipeline and show that LLM-based extraction can be practical, but the choice of model and prompt strategy must be carefully aligned with the task's difficulty.

6.2 Limitations and Challenges

During the development and testing of our literature review automation system, we identified several significant limitations. These challenges were particularly pronounced with local LLMs (TinyLLaMA, Phi-3, Mistral) compared to cloud-based models.

Performance Limitations of Local Models

We observed that local models often struggled to extract accurate and relevant information from scientific texts. TinyLLaMA, despite its efficiency, achieved only a 0.387 overall score, indicating substantial difficulties with complex extraction tasks. Phi-3 and Mistral performed better but still showed inconsistencies in handling structured information extraction.

The performance gap between local and cloud-based models was most evident in metadata extraction, where ChatGPT-4o achieved 0.690 while TinyLLaMA managed only 0.264. This disparity highlighted the computational limitations inherent in smaller, locally-deployed models.

Hallucination and Irrelevant Outputs

We encountered frequent issues with irrelevant or hallucinated outputs, particularly from local models. The absence of strong contextual grounding led to unreliable results, especially when extracting abstract or complex fields like study objectives or intervention characteristics.

These hallucinations manifested as invented publication details, fabricated author names, or completely unrelated research descriptions. The problem was most severe when models attempted to fill missing information rather than acknowledging uncertainty.

Context Handling Difficulties

Most local models failed to maintain coherence when presented with complex prompts or extended contexts. We observed degraded performance when prompts included multiple instructions or lengthy document passages, leading to incomplete extractions or confused responses.

This limitation was particularly challenging for comprehensive eCCo field extraction, which requires understanding multiple interconnected concepts simultaneously.

Lack of Vector-Based Retrieval

Initially, our system generated responses without semantic anchoring to document content. This approach increased hallucination risk since models relied primarily on their training knowledge rather than the specific document being analyzed.

6.3 Mitigation Strategies

To address these challenges, we implemented several targeted techniques across our processing pipeline. These solutions were developed iteratively based on observed performance patterns and specific model limitations.

6.3.1 Multi-Level Prompt Engineering

We developed a stratified prompt engineering approach tailored to each model's capabilities:

Ultra-Simple Prompts for TinyLLaMA: We reduced cognitive load by using single-instruction prompts with minimal context. For example, instead of complex multi-step instructions, we used direct requests like "Extract the publication year from this text: [excerpt]".

Concise Instructional Prompts for Phi-3 and Mistral: We provided clear, structured instructions with moderate detail. These prompts included specific formatting requirements and examples while avoiding overwhelming complexity.

Detailed Structured Prompts for GPT-4: We utilized comprehensive prompts with multiple examples, detailed field definitions, and explicit formatting instructions.

This approach significantly improved response quality and reduced hallucinations across all model types.

6.3.2 eCCo Terminology Integration

We introduced a controlled vocabulary for each extraction dimension to guide semantic understanding. The vocabulary included domain-specific terms like "conversational agent," "embodied character," and "therapeutic intervention."

This terminology integration helped models distinguish between similar concepts and reduced misinterpretations. For instance, models learned to differentiate between "chatbot" and "virtual assistant" based on their functional characteristics rather than treating them as synonymous.

6.3.3 Adaptive Top-k Context Retrieval

We implemented dynamic adjustment of context chunk retrieval based on field complexity through systematic optimization. Our analysis revealed that different extraction tasks require varying amounts of contextual information for optimal performance.

Optimization Process: We tested k values from 1 to 8 chunks across multiple field types, measuring both similarity scores and processing efficiency. The results showed clear patterns:

```
k=1: Context length 376 words, Processing time 6.95s, Similarity 0.10
k=2: Context length 838 words, Processing time 2.15s, Similarity 0.09
k=3: Context length 1234 words, Processing time 2.86s, Similarity 0.12
k=5: Context length 2042 words, Processing time 2.69s, Similarity 0.12
k=8: Context length 2932 words, Processing time 3.22s, Similarity 0.12
```

Example - StudyAim Extraction: When extracting research objectives, we found that k=3-5 provided optimal performance. For instance, when processing a study about "DIL" (a conversational agent for clinical settings), k=3 achieved 0.12 similarity with the ground truth while maintaining reasonable processing time (2.86s).

Field-Specific Optimization: Based on these findings, we established adaptive k values:

Lower K for Factual Fields:
- articleYear: k=1 (minimal context needed for date extraction)
- authorName: k=2 (focused retrieval for bibliographic data)
- authorCountry: k=2 (geographical information requires limited context)

Higher K for Interpretative Fields:
- studyAim: k=5 (comprehensive context for understanding research objectives)
- externalInterventions: k=6 (complex therapeutic context requires broader evidence)
- targetInterventions: k=5 (intervention details need substantial supporting context)
- interactionInputs: k=4 (moderate context for interface characteristics)
- actionInterventions: k=4 (behavioral interventions need focused but adequate context)

This approach provided relevant evidence without overwhelming models with excessive context, particularly benefiting local models with limited context handling capabilities. The adaptive strategy improved extraction accuracy while optimizing processing efficiency across different field types.

6.3.4 Adaptive Parsing and Output Processing

We developed an intelligent parsing system that automatically detects model types and applies appropriate extraction strategies. This innovation addresses the significant variation in output formats and verbosity levels across different LLMs.

Model Detection and Categorization: Our system automatically identifies the model being used and applies the corresponding parsing strategy:

```python
def get_model_category(model):
    if model == "tinyllama":
        return "ultra_simple"    # → Very robust parsing
    elif model in ["phi3mini", "mistral"]:
        return "simple"          # → Moderate parsing
    elif model in ["chatgpt4o"]:
        return "complex"         # → Standard parsing
```

Context Optimization by Model Type: We implemented adaptive context limits based on model capabilities:

TinyLLaMA: Maximum 500 characters to prevent context overflow and maintain coherence **Local Models (Phi3/Mistral)**: Maximum 800 characters for balanced context without overwhelming **GPT-4**: Up to 4000 characters for comprehensive context utilization

Intelligent Parsing Mechanisms: The system employs specialized extraction patterns tailored to each model's output characteristics:

- ✓ **Ultra-Simple Parsing for TinyLLaMA**: Direct value extraction using minimal regex patterns, handles fragmented or incomplete responses gracefully.
- ✓ **Moderate Parsing for Local Models**: Structured pattern matching with fallback mechanisms for partial responses.
- ✓ **Standard Parsing for GPT-4**: Comprehensive extraction handling complex JSON structures and detailed explanations.

Field-Specific Pattern Recognition: We developed specialized parsing rules for different eCCo field types:
Numerical Fields: Regex patterns for extracting years (e.g., "2023", "published in 2022") with validation against reasonable ranges.

Category Fields: Pattern matching for intervention types with synonyms mapping (e.g., "chatbot" → "conversational agent").
Text Fields: Adaptive cleaning that removes model-specific verbosity while preserving essential content.

Example - StudyAim Parsing: When extracting research objectives, TinyLLaMA might output: "study about chatbot help diabetes" while GPT-4 produces: "This research aims to develop and evaluate a conversational agent for diabetes management support." Our adaptive parser extracts the core objective from both formats using model-appropriate strategies.

This intelligent parsing approach increased extraction reliability by 40% for local models and reduced formatting errors across all model types, enabling consistent data quality regardless of the underlying LLM's output characteristics.

Prefix Cleaning and Post-Processing
We added a comprehensive cleaning layer to remove repetitive or verbose prefixes from LLM responses. Common patterns like "Answer:", "Based on the document...", and "The study shows..." were automatically stripped to ensure cleaner data.

This post-processing step enhanced output clarity and facilitated more reliable parsing and analysis.

6.3.5 Impact of Solutions

The implemented mitigation strategies resulted in measurable improvements across all models. Local models showed the most dramatic improvements, with Mistral achieving competitive performance (0.654) despite its computational limitations.

The combination of adaptive prompting, controlled vocabulary, and vector-based retrieval created a robust pipeline capable of handling diverse model capabilities while maintaining extraction quality. These solutions demonstrate that careful engineering can substantially improve LLM performance for specialized tasks, even when working with resource-constrained local models.

6.4 Remaining Limitations

Despite these improvements, some limitations persist. Local models still require more computational resources for complex extractions compared to cloud-based alternatives. The vector-based retrieval system, while effective, adds processing overhead that may not be suitable for all deployment scenarios.

Additionally, the system's performance remains dependent on document quality and structure. Poorly formatted PDFs or documents with complex layouts can still challenge the extraction pipeline, requiring additional preprocessing steps or manual intervention.

6.4 Human–AI Collaboration

Experimental results have shown that the use of AI significantly reduces the time required to conduct systematic literature reviews (SLRs), which are traditionally time-consuming and labor-intensive processes. However, this raises a critical question: *Can AI truly be trusted?*

Our findings suggest that relying solely on AI for SLRs is risky. In particular, when applying strict eligibility criteria such as those defined by PRISMA. AI models are prone to errors, such as misclassifying eligible documents as non-eligible.

Conversely, in the task of characteristic extraction, AI can sometimes outperform human annotators. This is largely due to the use of retrieval-augmented generation (RAG), which enables the model to access semantically relevant keywords and contextual information more effectively.

In conclusion, AI can serve as a valuable assistant in the SLR process. While the human role remains essential particularly for ensuring the reliability and integrity of results this role may evolve over time. As AI capabilities continue to improve, human involvement in SLRs may become more supervisory or strategic rather than operational. Nonetheless, at the current stage, human judgment continues to be indispensable, especially in applying complex eligibility criteria and interpreting nuanced findings.

6.5 Implications for Future Research

The insights gained from this study point to several promising directions for future research. From a technical perspective, one of the most impactful areas to explore is the development of improved chunking strategies. Identifying and isolating the most relevant parts of a document is essential to enhance retrieval accuracy and model output quality, especially in semantic search and generation steps.

Another important evolution involves the integration of more advanced AI models that rely on GPU acceleration. These models, while resource-intensive, could significantly improve both response time and accuracy. As infrastructure and hardware availability increase, deploying such models in real-time environments will become increasingly feasible.

The benchmarking in this study was conducted on a set of **20 scientific** documents. Future work should consider scaling up to larger datasets to evaluate how the system performs in more realistic, high-volume scenarios. This could also involve studying how AI performs compared to human experts on the same tasks both in terms of speed and extraction quality.

Moreover, broader discussions around ethics and knowledge reliability are needed. While AI models may accelerate literature reviews, they can also introduce biases or hallucinate information. Evaluating the impact of these technologies on the quality of scientific knowledge synthesis will be a crucial challenge in the years ahead.

7 Conclusion

Chapter's Content

7.1 Project Summary

This thesis presents the design, development, and evaluation of a system for extracting structured information from scientific PDFs using the eCCo framework. The goal was to support systematic literature reviews by automating metadata extraction, study aim classification, and identification of conversational agent characteristics. The proposed pipeline combines rule-based components like GROBID with advanced AI techniques, including vector search and LLMs.

The system was tested on a curated set of 20 articles and evaluated across multiple dimensions. Benchmarking results revealed strong performance from models such as ChatGPT-4o and Mistral, while also highlighting the trade-offs of lighter local models.

Throughout the project, several challenges emerged. Local models like TinyLLaMA and Phi-3 Mini struggled with extraction accuracy, especially in metadata fields, due to their limited capacity to handle complex structures and context. To address these issues, we implemented several mitigation strategies: multi-level prompt engineering to adapt instructions to the model's capacity, eCCo terminology integration to improve the understanding of domain-specific concepts, adaptive top-k context retrieval to prioritize the most relevant text chunks, and adaptive parsing and output processing to handle inconsistencies and incomplete outputs more effectively. Hallucination and irrelevant outputs were frequent, with models inventing publication data or misinterpreting instructions. Maintaining coherent outputs with long prompts or documents also proved difficult. These challenges were particularly visible in tasks requiring nuanced understanding, such as study objectives or eCCo dimensions. Technical challenges such as prompt sensitivity and formatting inconsistencies were addressed through scoring strategies and modular design.

The work emphasizes the importance of human–AI collaboration in scientific analysis, and it opens new research directions around chunking strategies, GPU-based models, and ethical considerations in AI-driven evidence synthesis.

7.2 Review of the objectives

This thesis aims to investigate the integration of LLMs into systematic literature review (SLR) workflows. Specifically, it explores whether LLMs can be effectively operationalized to extract structured information aligned with the eCCo framework. The study evaluates practical integration strategies, such as retrieval-augmented generation (RAG), adaptive prompt design, and semantic chunking. It also assesses the risks and limitations associated with using LLMs in evidence synthesis, including issues of bias, hallucination, and lack of transparency. Additionally, the work considers whether prompt engineering or domain-specific adjustments can improve model accuracy and relevance. Ultimately, the objective is to determine how LLMs can assist without replacing human reviewers in accelerating and enhancing literature analysis.

7.3 Personal Conclusion

This master's thesis has been a truly meaningful journey for me. Earlier in my studies, I had already explored text analysis, but at that time the work was based mostly on statistical methods. That experience gave me an initial entry point into the field, but this project was something completely different, it allowed me to work with AI technologies that bring a much deeper level of insight and possibilities.

Working on this project at the HumanTech Institute, in collaboration with the Nursing School of Lisbon (ESEL), was both a technical and personal privilege. The project allowed me to explore the future of healthcare technologies, particularly the transformative role of virtual conversational agents in patient interactions.

The technical challenges were significant, especially in learning to understand and integrate llms, but overcoming them provided me with valuable experience. The support and guidance of my supervisors were crucial during the more difficult phases.

Most importantly, this thesis has opened new perspectives for me. I am confident that the skills and knowledge I have acquired will serve as a strong foundation for my future academic and professional projects.

Appendices

This chapter provides the supplementary material, including the source code, the prompts, and the detailed LLM performance per article.

The full project is organized in a dedicated repository: https://gitlab.forge.hefr.ch/humantech-institute/projets-master/2024-2025/tm-ia-for-reviews-vural-abdi

Chapter's Content

A Detailed Prompts

This section presents the complete set of prompts used in the automated extraction system for systematic literature reviews.

File Name	Type	Purpose	Description
prompts_prisma_ultra_simple.json	JSON Promp	Ultra Simple PRISMA Screening	Minimalist prompts with yes/no questions for rapid PRISMA eligibility assessment. Each criterion in 1-2 words for fast processing.
prompts_prisma_simple.json	JSON Promp	Simple PRISMA Screening	Simplified prompts for PRISMA criteria with clear yes/no instructions. Balances thoroughness with processing speed.
prompts_prisma_improved.json	JSON Promp	Detailed PRISMA Screening	Comprehensive prompts with detailed instructions and rationale for each PRISMA criterion. Includes step-by-step decision logic.
prompts_ecco_ultra_simple.json	JSON Promp	Ultra Simple eCCo Extraction	Basic prompts for extracting core eCCo characteristics with minimal context. Focuses on essential conversational agent features.
prompts_ecco_simple.json	JSON Promp	Simple eCCo Extraction	Standard prompts for eCCo framework extraction with moderate detail level. Good balance between accuracy and processing time.
prompts_ecco_improved.json	JSON Promp	Detailed eCCo Extraction	Comprehensive prompts with detailed descriptions and examples for each eCCo characteristic. Maximizes extraction accuracy.

A.1 PRISMA Screening Prompts

A.1.1 PRISMA Ultra Simple

```
{
  "step1_language": "English article? YES or NO:",
  "step2_publication_type": "Research study? YES or NO:",
  "step3_population": "About humans? YES or NO:",
  "step4_health_domain": "About health? YES or NO:",
  "step5_target_population": "Only for doctors? YES or NO:",
  "step6_conversational_agent": "System talks to users? YES or NO:",
  "step7_automated_capabilities": "Automatic answers? YES or NO:",
  "step8_research_content": "System tested? YES or NO:",
  "final_decision": "Include article? YES or NO:"
}
```

A.1.2 PRISMA Simple

```
{
  "step1_language": "Is this article in English, French, Italian, Portuguese, or Spanish? A
nswer YES or NO.",
  "step2_publication_type": "Is this a research study with full text? Answer YES or NO.",
  "step3_population": "Is this study about humans or patients? Answer YES or NO.",
  "step4_health_domain": "Is this about health or medicine? Answer YES or NO.",
  "step5_target_population": "Is this study only about doctors, caregivers or students? Ans
wer YES or NO.",
  "step6_conversational_agent": "Does this describe a system that talks or chats with users
? Answer YES or NO.",
  "step7_automated_capabilities": "Does the system answer automatically (not human)? Answer
 YES or NO.",
  "step8_research_content": "Does this describe how the system was built or tested? Answer
YES or NO.",
  "final_decision": "Should this article be INCLUDED based on all criteria? Answer YES or N
O."
}
```

A.1.3 PRISMA Improved (Detailed Instructions)

```
{
  "step1_language": {
    "question": "Is the article written in English, French, Italian, Portuguese, or Spanish
?",
    "instruction": "Answer YES if the article is written in one of these languages. Otherwi
se, answer NO."
  },
  "step2_publication_type": {
    "question": "Is this a primary research study with full text available?",
    "instruction": "Answer YES if the article reports empirical research (development, eval
uation, or implementation). Otherwise, answer NO."
  },
  "step3_population": {
    "question": "Is the study about humans or patients?",
    "instruction": "Answer YES if it involves humans of any age. Otherwise, answer NO."
  },
  "step4_health_domain": {
    "question": "Is the topic related to health or well-being?",
    "instruction": "Answer YES if the study is related to healthcare, medicine, well-
being, or patient support. Otherwise, answer NO."
  },
  "step5_target_population": {
    "question": "Is this study focused exclusively on healthcare professionals, caregivers,
 or their education/training?",
    "instruction": "Answer YES if the study ONLY targets doctors, nurses, caregivers, or me
dical students and their training. Answer NO if the study includes patients, general public
, or end-users as participants."
  },
  "step6_conversational_agent": {
    "question": "Does the article describe a system that simulates two-
way human conversation?",
    "instruction": "Answer YES if the system talks or chats with users using speech or text
. Otherwise, answer NO."
  },
  "step7_automated_capabilities": {
    "question": "Is the conversation automated (not manually operated)?",
    "instruction": "Answer YES if the system answers automatically. Otherwise, answer NO."
  },
  "step8_research_content": {
```

```
    "question": "Does the article report design, development, evaluation, or implementation
 of the system?",
    "instruction": "Answer YES if it describes how the system was built, tested or implemen
ted. Otherwise, answer NO."
  },
  "final_decision": {
    "question": "Based on your previous answers to steps 1-
8, should this article be INCLUDED in the systematic review?",
    "instruction": "Review your previous answers. The article should be INCLUDED (YES) if:
(1) it's in a supported language, (2) it's a primary research study, (3) involves humans, (
4) relates to health, (5) does NOT focus only on healthcare professionals, (6) describes a
conversational agent, (7) the agent is automated, and (8) reports research content. Answer
YES for inclusion, NO for exclusion."
  }
}
```

A.2 eCCo Data Extraction Prompts

A.2.1 eCCo Ultra Simple

```
{
  "articleType": "Is this a Journal article, Conference paper, or Book chapter?",
  "articleYear": "What year? Give only the number:",
  "articleTitle": "Article title:",
  "authorName": "First author name:",
  "authorCountry": "Author country:",
  "studyAim": "What does this study want to do?",
  "interactionInputs": "How do users input? Pick from: DIRECT_VOICE, DIRECT_WRITTEN, DIRECT
_OPTIONS, DIRECT_IMAGES, INDIRECT_DEVICES, INDIRECT_VISION",
  "interactionOutputs": "How does system output? Pick from: TEXT, VOICE, OTHER",
  "externalInterventions": "Does it involve doctors or family? Pick from: EXTERNAL_HUMAN_IN
TERVENTION, ADDITIONAL_END_USERS, none",
  "studyTypes": "Study type? Pick from: USABILITY_EVALUATION, RCT, OTHER, none",
  "deliveryChannels": "Where does it run? Pick from: DESKTOP_APP, WEBSITE_APP, MOBILE_APP,
MESSAGING_APP, STANDALONE_DEVICE, MOBILE_ROBOT",
  "targetInterventions": "Health topics? Pick from: EATING, PHYSICAL_ACTIVITY, SLEEP_AND_RE
ST, HYGIENE, ORAL_HYGIENE, SEXUAL_BEHAVIOR, PARENTING, BREASTFEEDING, UV_RADIATION_EXPOSURE
, PHARMACEUTICAL_USE, ALCOHOL, TOBACCO, SYMPTOMS",
  "actionInterventions": "What does system do? Pick from: ASSESSMENT, EDUCATION, SOCIAL_SUP
PORT, BEHAVIOUR_CHANGE",
  "additionalFeatures": "Special features? Pick from: AGENT_EMOTION, PERSONALITY, PROACTIVI
TY, SENTIMENT_DETECTION, none",
  "embodiments": "What does it look like? Pick from: VIRTUAL_HUMAN, VIRTUAL_NO_HUMAN, PHYSI
CAL_HUMAN, PHYSICAL_NO_HUMAN"
}
```

A.2.2 eCCo Simple

```
{
  "articleType": "What type of article is this? Answer with one of: Journal article, Confer
ence paper, Book chapter, Other. Answer: ",
  "articleYear": "What year was this published? Give only the 4-
digit year (example: 2023). Answer: ",
  "articleTitle": "What is the title of this article? Copy the exact title. Answer: ",
  "authorName": "Who is the first author or corresponding author? Give the full name. Answe
r: ",
  "authorCountry": "What country is the author from? Give the country name. Answer: ",
  "studyAim": "What is the main goal of this study? Write 1-
2 sentences about what the researchers want to achieve. Answer: ",
  "interactionInputs": "How do users interact with the system? Choose all that apply: DIREC
T_VOICE, DIRECT_WRITTEN, DIRECT_OPTIONS, DIRECT_IMAGES, INDIRECT_DEVICES, INDIRECT_VISION.
If unclear, write: unsure. Answer: ",
```

```
  "interactionOutputs": "How does the system respond to users? Choose all that apply: TEXT,
 VOICE, OTHER. If unclear, write: unsure. Answer: ",
  "externalInterventions": "Does the system involve other people like doctors or family? Ch
oose: EXTERNAL_HUMAN_INTERVENTION, ADDITIONAL_END_USERS, or none. Answer: ",
  "studyTypes": "What type of study is this? Choose: USABILITY_EVALUATION, RCT, OTHER, or n
one. Answer: ",
  "deliveryChannels": "Where does the system run? Choose: DESKTOP_APP, WEBSITE_APP, MOBILE_
APP, MESSAGING_APP, STANDALONE_DEVICE, MOBILE_ROBOT. If unclear, write: unsure. Answer: ",
  "targetInterventions": "What health topics does this target? Choose any that apply: EATIN
G, PHYSICAL_ACTIVITY, SLEEP_AND_REST, HYGIENE, ORAL_HYGIENE, SEXUAL_BEHAVIOR, PARENTING, BR
EASTFEEDING, UV_RADIATION_EXPOSURE, PHARMACEUTICAL_USE, ALCOHOL, TOBACCO, SYMPTOMS. Answer:
 ",
  "actionInterventions": "What does the system do? Choose any that apply: ASSESSMENT, EDUCA
TION, SOCIAL_SUPPORT, BEHAVIOUR_CHANGE. Answer: ",
  "additionalFeatures": "Does the system have special features? Choose any that apply: AGEN
T_EMOTION, PERSONALITY, PROACTIVITY, SENTIMENT_DETECTION, or none. Answer: ",
  "embodiments": "What does the system look like? Choose one: VIRTUAL_HUMAN, VIRTUAL_NO_HUM
AN, PHYSICAL_HUMAN, PHYSICAL_NO_HUMAN. Answer: "}
```

A.2.3 eCCo Detailed (Complete Instructions)

```
{
  "articleType": "What type of publication is the article? Examples: Journal article, Confe
rence paper, Book or book chapter. Return the type using the author or publisher terminolog
y. Format: articleType: [value]",
  "articleYear": "Extract the year of publication of the article. Return it as a number (e.
g., 2022). Format: articleYear: [YYYY]",
  "articleTitle": "Return the full title of the article as stated by the authors. Format: a
rticleTitle: [text]",
  "authorName": "Return the full name(s) of the corresponding author(s), or first author if
 not specified. Format: authorName: [value]",
  "authorCountry": "Return the country of affiliation of the corresponding (or first) autho
r. Format: authorCountry: [value]",
  "studyAim": "OPTIMIZED: Extract the PRIMARY objective of the study in 1-
2 concise sentences only. Focus ONLY on: The main research goal or purpose, What the study
aims to achieve or investigate, The primary research question being addressed. AVOID: Imple
mentation details or technical descriptions, Background information or justifications, Mult
iple secondary objectives, How the system works. Examples of GOOD extractions: To develop a
nd evaluate a chatbot for medical diagnosis, To assess the effectiveness of AI-
based health coaching, To investigate user acceptance of conversational agents in healthcar
e. Format: studyAim: [1-2 sentences maximum]",
  "interactionInputs": "ENHANCED: Identify all user input methods used in the study's conve
rsational agent. Look for explicit mentions AND contextual clues: DIRECT_VOICE (voice input
, speech recognition, voice commands), DIRECT_WRITTEN (typing, text input, chat, messaging)
, DIRECT_OPTIONS (buttons, menu options, select, choose, click), DIRECT_IMAGES (image uploa
d, photo sharing, camera input), INDIRECT_DEVICES (sensors, wearables, tracking, IoT), IN_D
IRECT_VISION (camera, visual analysis, computer vision). CONTEXTUAL CLUES: Mobile app likel
y DIRECT_WRITTEN + DIRECT_OPTIONS, Voice assistant likely DIRECT_VOICE, Healthcare monitori
ng likely INDIRECT_DEVICES. Return ALL categories that apply based on evidence. If no clear
 evidence, return unsure. Format: interactionInputs: [list]",
  "interactionOutputs": "ENHANCED: What types of outputs are produced by the conversational
 agent? Look for explicit mentions AND system characteristics: TEXT (text messages, written
 responses, chat response), VOICE (spoken responses, audio output, text-to-
speech), OTHER (visual elements, images, videos, charts, visualization). CONTEXTUAL CLUES:
Chatbot/messaging interface likely TEXT, Voice assistant/smart speaker likely VOICE, Mobile
 app with notifications likely TEXT, Dashboard/analytics likely OTHER. Return all relevant
types based on evidence. If unclear, return unsure. Format: interactionOutputs: [list]",
  "externalInterventions": "Does the intervention involve external parties beyond the user
and the conversational agent? Look for: EXTERNAL_HUMAN_INTERVENTION (healthcare providers,
doctors, therapists who actively participate), ADDITIONAL_END_USERS (family members, caregi
vers, multiple users). Return ONLY the predefined categories that apply based on explicit e
vidence. If the system is fully automated without external human involvement, return []. Fo
rmat: externalInterventions: [list]",
```

```
  "studyTypes": "What is the study type based on the methodology described? Look for eviden
ce of: USABILITY_EVALUATION (user testing, usability study, user experience evaluation), RC
T (randomized controlled trial, control group, randomization), OTHER (other research method
ologies). Return all applicable types based on explicit methodology descriptions. If none c
learly apply, return []. Format: studyTypes: [list]",
  "deliveryChannels": "OPTIMIZED: Identify WHERE the conversational agent is delivered. Loo
k for EXPLICIT mentions AND technical clues: DESKTOP_APP, WEBSITE_APP, MOBILE_APP, MESSAGIN
G_APP, STANDALONE_DEVICE, MOBILE_ROBOT. INDIRECT CLUES: accessed via browser means WEBSITE_
APP, downloaded from app store means MOBILE_APP, smartphone-based means MOBILE_APP, client-
server architecture means WEBSITE_APP. Only return predefined categories with clear evidenc
e. If unclear, return ['unsure']. Format: deliveryChannels: [list of categories or 'unsure'
]",
  "targetInterventions": "What health and behavior domains are targeted by the intervention
? Look for explicit mentions of: EATING (diet, nutrition, food), PHYSICAL_ACTIVITY (exercis
e, fitness, activity), SLEEP_AND_REST (sleep, rest, sleep quality), HYGIENE (personal hygie
ne, cleanliness), ORAL_HYGIENE (dental care, teeth brushing), SEXUAL_BEHAVIOR (sexual healt
h, contraception), PARENTING (parenting skills, child care), BREASTFEEDING (breastfeeding s
upport, nursing), UV_RADIATION_EXPOSURE (sun protection, UV exposure), PHARMACEUTICAL_USE (
medication, drug use, adherence), ALCOHOL (alcohol consumption, drinking), TOBACCO (smoking
, tobacco use), SYMPTOMS (symptom monitoring, health symptoms). Return all domains that are
 explicitly mentioned. Format: targetInterventions: [list]",
  "actionInterventions": "ENHANCED: What actions does the conversational agent perform? Loo
k for evidence of these 4 PREDEFINED categories ONLY: ASSESSMENT (health assessment, evalua
tion, monitoring, assess, evaluate, monitor, track, screen, diagnose, check, symptom checki
ng, risk assessment, health screening), EDUCATION (providing information, teaching, inform,
 educate, teach, explain, health education, advice, guidance, tips, patient education, info
rmation provision), SOCIAL_SUPPORT (emotional support, encouragement, companionship, suppor
t, encourage, motivate, reassurance, empathy, peer support, emotional assistance), BEHAVIOU
R_CHANGE (promoting behavior change, habit formation, lifestyle modification, goal setting,
 coaching, motivation for change, habit tracking, goal achievement, behavior modification).
 Return ONLY these predefined categories that are explicitly described. DO NOT create custo
m action descriptions. Format: actionInterventions: [list of predefined categories only]",
  "additionalFeatures": "ENHANCED: Identify advanced features of the conversational agent.
Search for explicit evidence AND behavioral descriptions: AGENT_EMOTION (agent displays emo
tions, emotional expressions, empathetic responses, emotional intelligence), PERSONALITY (a
gent has distinct personality traits, character, persona, conversational style, tone), PROA
CTIVITY (agent initiates conversations, sends reminders, proactive messaging, push notifica
tions, scheduled messages, automatic follow-
up), SENTIMENT_DETECTION (agent detects user emotions, mood analysis, sentiment, emotion de
tection, mood recognition, emotional awareness). CONTEXTUAL CLUES: Mentions of reminders or
 notifications suggest PROACTIVITY, Conversational style suggests PERSONALITY, Understandin
g user mood suggests SENTIMENT_DETECTION, Empathetic responses suggest AGENT_EMOTION. Retur
n only features with clear evidence. If no advanced features mentioned, return []. Format:
additionalFeatures: [list]",
  "embodiments": "What is the physical or virtual embodiment of the conversational agent? L
ook for descriptions of: VIRTUAL_HUMAN (virtual human character, avatar with human appearan
ce, virtual human, avatar, human-like), VIRTUAL_NO_HUMAN (text-based, non-
human virtual interface, chatbot without avatar, text-
based, chatbot, no visual representation), PHYSICAL_HUMAN (humanoid robot, robot with human
 appearance, humanoid, human-like robot), PHYSICAL_NO_HUMAN (non-
humanoid robot, device without human form, robot, device, non-
human physical form). Return the embodiment type based on explicit descriptions. Format: em
bodiments: [list]"
}
```

B

Code Structure

File Name	Type	Purpose	Description
run.py	Script	Entry point	Launches the pipeline or the main application.
app.py	Streamlit	Main user interface	Web interface to upload PDFs, run the extraction, and display results.
ui_debug.py	Utility	Debugging UI	Provides tools to display logs and debugging information in the interface.
ui_manager.py	Module	UI management	Manages the organization and rendering of Streamlit components.
ui_results.py	Module	Results display	Displays extracted results in tables and visualizations.
ui_components.py	Module	Reusable UI components	Defines reusable Streamlit components (buttons, panels, layouts).
utils_llm.py	Utility	LLM interface	Functions to interact with OpenAI/Ollama and manage local or cloud models.
utils_extraction.py	Utility	PDF processing	Functions to extract and split text into chunks from PDF documents.
utils_prompts.py	Utility	Prompt management	Loads and manages ECCO prompts from JSON files.
config.py	Config	Global configuration	Centralizes paths, API keys, model settings, and constants of the project.
requirements.txt	Config	Dependencies	Lists required Python libraries (Streamlit, LangChain, OpenAI, Chroma, etc.).
pipeline.py	Core module	Standard ECCO pipeline	Implements the baseline ECCO extraction pipeline (multi-PDF, prompts, vectorization).
enhanced_pipeline.py	Core module	Enhanced ECCO pipeline	Improved version with official eCCo terminology and adaptive strategies.
processing_engine.py	Engine	Processing engine	Handles PDF parsing, chunking, vectorization, and storage in Chroma.
eligibility_engine.py	Engine	PRISMA eligibility	Evaluates whether an article meets PRISMA inclusion/exclusion criteria.
grobid_engine.py	Engine	Metadata extraction (GROBID)	Connects to the GROBID service to extract structured bibliographic metadata.
README.md	Documentation	Project documentation	Provides project overview, installation, usage guide, and Docker deployment.

Table 3 - Main Application files

C

Benchmark – Detailed Performance

ARTICLE	AVERAGE SCORE	DIFFICULTY	BEST MODEL	LOWEST	STD. DEV.
ARTICLE_7	Good 0.674	Medium	PHI3MINI (0.922)	TINYLLAMA (0.291)	0.236
ARTICLE_5	Good 0.670	Medium	MISTRAL (0.830)	TINYLLAMA (0.463)	0.127
ARTICLE_20	Good 0.660	Medium	CHATGPT4O (0.796)	TINYLLAMA (0.513)	0.107
ARTICLE_2	Good 0.653	Medium	MISTRAL (0.780)	PHI3MINI (0.521)	0.109
ARTICLE_9	Good 0.638	Medium	MISTRAL (0.892)	CHATGPT4O (0.509)	0.137
ARTICLE_19	Good 0.635	Medium	CHATGPT4OMINI (0.819)	TINYLLAMA (0.424)	0.149
ARTICLE_6	Good 0.634	Medium	MISTRAL (0.793)	TINYLLAMA (0.419)	0.139
ARTICLE_18	Good 0.632	Medium	MISTRAL (0.899)	TINYLLAMA (0.301)	0.208
ARTICLE_16	Good 0.629	Medium	MISTRAL (0.861)	TINYLLAMA (0.231)	0.211
ARTICLE_13	Good 0.622	Medium	CHATGPT4O (0.874)	TINYLLAMA (0.316)	0.179
ARTICLE_14	Good 0.620	Medium	MISTRAL (0.786)	TINYLLAMA (0.309)	0.174
ARTICLE_17	Average 0.592	Medium	CHATGPT4OMINI (0.701)	TINYLLAMA (0.444)	0.120
ARTICLE_15	Average 0.591	Medium	CHATGPT4O (0.753)	TINYLLAMA (0.305)	0.167
ARTICLE_10	Average 0.576	Medium	CHATGPT4OMINI (0.721)	TINYLLAMA (0.229)	0.185
ARTICLE_8	Average 0.538	Medium	MISTRAL (0.617)	TINYLLAMA (0.513)	0.040
ARTICLE_12	Average 0.510	Medium	MISTRAL (0.766)	TINYLLAMA (0.219)	0.195
ARTICLE_11	Average 0.509	Medium	MISTRAL (0.568)	TINYLLAMA (0.405)	0.058
ARTICLE_1	Average 0.506	Medium	MISTRAL (0.589)	TINYLLAMA (0.343)	0.086
ARTICLE_4	Average 0.497	Hard	CHATGPT4OMINI (0.821)	TINYLLAMA (0.315)	0.172
ARTICLE_3	Average 0.458	Hard	CHATGPT4O (0.600)	TINYLLAMA (0.328)	0.104

Table 4 - Performance of llms models by articles

Glossary

A

Adaptive Prompting : Strategic modification of instructions given to language models based on model capabilities and task complexity, optimizing response quality by adjusting prompt complexity for local vs. cloud models.

Agent Conversationnel *(see Conversational Agent)*

API (Application Programming Interface) : Programming interface enabling integration of different software services, particularly for accessing language models like OpenAI GPT-4 or document processing services.

Artificial Intelligence (AI) : Field of computer science focused on creating systems capable of performing tasks that typically require human intelligence, including understanding natural language, learning from data, and making decisions.

B

Batch Processing : Method of processing multiple documents simultaneously rather than individually, improving efficiency when handling large collections of scientific papers.

Bibliographic Metadata : Structured information describing a scientific publication including title, authors, affiliations, journal, publication year, DOI, references, and other citation details.

C

Chunking *(Text Segmentation)* : Technique of dividing documents into smaller, manageable segments with overlap, enabling efficient processing by language models and precise vector search. Default configuration: 1000 characters with 200-character overlap.

ChromaDB : Open-source vector database optimized for storing and searching embeddings, used in this project to index scientific documents and enable semantic search capabilities.

Claude : Large language model developed by Anthropic, used as an alternative to GPT for certain extraction and analysis tasks in the project.

Conversational Agent : Computer system capable of simulating natural conversation with users through text or voice, using natural language processing techniques to understand and respond to queries.

Conversational E-coaches *(see eCCo)*

D

Document Processing Pipeline: Automated sequence of steps for processing documents: PDF extraction → chunking → vectorization → contextual search → LLM extraction → structured results.

Docker: Containerization platform used for deploying the application with all its dependencies in isolated, portable environments.

E

eCCo (Conversational E-coaches) : International collaborative framework between HES-SO and ESEL for developing a comprehensive taxonomy of conversational health agents. The project aims to create e-coaching systems using natural language interaction to improve health and well-being.

E-coaching : Revolutionary interdisciplinary field where intelligent systems pursue health-related goals by providing personalized guidance to users. Conversational e-coaches are particularly promising for improving user trust through natural language interaction.

Eligibility Assessment: Automated evaluation process using PRISMA criteria to determine whether scientific articles should be included in systematic reviews, reducing manual screening effort.

Embeddings *(Vector Representations)* : Numerical representations of text as vectors in multidimensional space, capturing semantic relationships between words and concepts. This project uses sentence-transformers/all-MiniLM-L6-v2 generating 384-dimensional vectors.

ESEL (Escola Superior de Enfermagem de Lisboa) : Nursing School of Lisbon, Portuguese partner in the eCCo project, contributing with the VASelfCare project to research on conversational health agents.

Extraction Pipeline : Systematic process for identifying and structuring specific information from unstructured text using AI techniques to convert scientific publication content into actionable data.

F

Full-text Processing : Comprehensive analysis of complete document content including sections, references, and detailed metadata extraction, as opposed to header-only processing.

G

Generative AI : Subfield of artificial intelligence capable of creating new content (text, images, code) based on learned patterns, used in this context to generate summaries and extract structured information from scientific literature.

GPT (Generative Pre-trained Transformer) : Language model architecture developed by OpenAI, based on the Transformer attention mechanism. This project primarily uses GPT-4o-mini and GPT-4o for eCCo characteristic extraction.

GROBID (GeneRation Of BIbliographic Data) : Open-source tool developed by INRIA using machine learning to extract and analyze bibliographic metadata from PDF scientific documents, converting to XML TEI (Text Encoding Initiative) format.

H

Header Extraction : Fast metadata extraction process focusing on basic bibliographic information (title, authors, abstract) without full document analysis.

HES-SO (Haute École Spécialisée de Suisse occidentale) : University of Applied Sciences Western Switzerland, academic institution partnering in the eCCo project through the HumanTech Institute.

HumanTech Institute: HES-SO research institute specializing in human-centered technologies, hosting the NESTORE project and contributing to the eCCo initiative.

I

Information Extraction: Automated process of identifying and structuring specific information from unstructured text, using AI techniques to convert scientific publication content into exploitable data.

International Collaboration: Swiss Portuguese research partnership between HES-SO and ESEL aimed at developing global taxonomy for conversational e-coaches and advancing health-related AI research.

J

JSON (JavaScript Object Notation) : Lightweight data interchange format used for storing extraction results, configuration files, and API communications in the system.

L

LangChain : Development framework for applications based on language models, facilitating orchestration of complex workflows involving multiple AI components.

Large Language Model (LLM) : Extensive language model trained on vast text corpora, capable of understanding and generating natural language with high quality. Examples: GPT-4, Claude, local Ollama models.

Literature Review Automation: Application of AI technologies to automate traditionally manual processes in systematic literature reviews, including paper screening, data extraction, and synthesis.

Local Language Model: AI model executed locally via platforms like Ollama, offering enhanced privacy and control but typically with reduced capabilities compared to cloud models.

M

Machine Learning: Subset of AI that enables systems to automatically learn and improve from experience without explicit programming, used in GROBID for document structure recognition.

Metadata Extraction: Automated process of identifying and structuring bibliographic information from scientific documents, including authors, titles, abstracts, and publication details.

Multi-modal Processing: Flexible processing architecture offering 5 operational modes: GROBID only, PRISMA only, eCCo only, PRISMA→eCCo combined, and full pipeline integrating all components.

MSE (Master of Science in Engineering) : HES-SO master's program in engineering under which this thesis work is conducted, focused on practical application of AI technologies.

N

Natural Language Processing (NLP) : Branch of AI that enables machines to understand, interpret, and generate human language, fundamental to conversational agent development.

NESTORE: Research project at HES-SO's HumanTech Institute focused on personalized health assistance and coaching technologies, partner in the eCCo initiative.

O

ODD *(see SDG)*

Ollama : Open-source platform for running language models locally, providing an alternative to cloud services for sensitive data processing.

OpenAI: AI research company developing GPT models, providing cloud-based language model services used in this project for complex reasoning tasks.

P

PDF Processing: Specialized techniques for extracting and analyzing content from Portable Document Format files, particularly scientific publications with complex structures.

Pipeline Architecture: Systematic workflow design enabling flexible document processing through configurable stages: extraction, analysis, filtering, and results generation.

PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analyses) : Methodological standards for systematic reviews, integrated into this project as 8 automated eligibility criteria for filtering relevant articles.

Prompt Engineering: Strategic design of instructions given to language models to optimize response quality and relevance, adapted according to model capabilities (local vs. cloud).

Q

Quality Control: Systematic validation processes ensuring accuracy and reliability of extracted data, including confidence scoring and transparency measures.

R

RAG (Retrieval-Augmented Generation) : Architecture combining information retrieval and text generation, enabling language models to access external domain-specific knowledge.

Reference Processing: Automated extraction and structuring of citation information from scientific documents, including author names, publication titles, and bibliographic details.

Reproducibility: Principle ensuring research results can be replicated, supported in this project through transparent methodologies and detailed logging of AI decisions.

S

SDG (Sustainable Development Goals) : United Nations objectives, particularly SDG 3 "Good Health and Well-being" to which the eCCo project contributes through conversational health agent development.

Semantic Analysis: Process of analyzing meaning and conceptual relationships in text, enabling AI systems to understand context and linguistic nuances beyond simple keyword recognition.

Semantic Search: Search technique based on meaning similarity rather than exact keyword matching, using embeddings to understand conceptual relationships.

Sentence Transformers: Family of models specialized in creating high-quality embeddings for sentences and paragraphs, used for semantic search (all-MiniLM-L6-v2 model in this project).

Streamlit : Python framework for rapidly creating interactive web applications, used for the user interface of the eCCo extraction system.

Systematic Review: Rigorous research methodology for identifying, evaluating, and synthesizing all relevant studies on a given topic, following standardized protocols like PRISMA.

T

Taxonomy: Hierarchical classification system organizing concepts into structured categories, specifically the eCCo taxonomy classifying conversational health agent characteristics.

TEI (Text Encoding Initiative) : XML standard for encoding textual documents, GROBID's output format containing structured metadata and complete publication content.

Text Chunking *(see Chunking)*

Transformer Architecture: Deep learning model architecture using attention mechanisms, foundation for modern language models like GPT and Claude.

V

VASelfCare : Research project at ESEL (Portugal) on virtual agents for self-care, partner in the eCCo initiative for developing international taxonomy.

Vector Database: Specialized database optimized for storing and searching high-dimensional vectors, enabling efficient similarity search for semantic document retrieval.

Vector Index: Data structure optimized for similarity search between vectors, enabling rapid retrieval of most relevant document segments for given queries.

Vectorization: Process of converting text into numerical representations (vectors) enabling semantic similarity computation and efficient search in large document collections.

W

WHO (World Health Organization): International organization whose noncommunicable disease reduction objectives are supported by eCCo project research on conversational health agents.

Workflow Orchestration: Coordination of multiple processing steps and components to achieve complex document analysis tasks through automated pipelines.

X

XML (eXtensible Markup Language) : Structured markup language used by GROBID to encode extracted document metadata and content in standardized TEI format.

Acronyms and Abbreviations

AI: Artificial Intelligence
API: Application Programming Interface
CSV: Comma-Separated Values
DOI: Digital Object Identifier
eCCo : Conversational E-coaches
ESEL: Escola Superior de Enfermagem de Lisboa
GPT: Generative Pre-trained Transformer
GROBID: GeneRation Of BIbliographic Data

HES-SO: Haute École Spécialisée de Suisse occidentale
JSON: JavaScript Object Notation
LLM: Large Language Model
MSE: Master of Science in Engineering
NLP: Natural Language Processing
PDF: Portable Document Format
PRISMA: Preferred Reporting Items for Systematic Reviews and Meta-Analyses
RAG: Retrieval-Augmented Generation
SDG: Sustainable Development Goals
TEI: Text Encoding Initiative
WHO: World Health Organization
XML: eXtensible Markup Language

References

[1] « Academic Publishing Statistics – WordsRated ». Consulté le: 21 juillet 2025. [En ligne]. Disponible sur: https://wordsrated.com/academic-publishing-statistics/

[2] « PRISMA 2020 statement », PRISMA statement. Consulté le: 21 juillet 2025. [En ligne]. Disponible sur: https://www.prisma-statement.org/prisma-2020

[3] T. Susnjak, « PRISMA-DFLLM: An Extension of PRISMA for Systematic Literature Reviews using Domain-specific Finetuned Large Language Models », 15 juin 2023, *arXiv*: arXiv:2306.14905. doi: 10.48550/arXiv.2306.14905.

[4] M. J. Page *et al.*, « The PRISMA 2020 statement: an updated guideline for reporting systematic reviews », *BMJ*, vol. 372, p. n71, mars 2021, doi: 10.1136/bmj.n71.

[5] K. Kolaski, L. R. Logan, et J. P. A. Ioannidis, « Guidance to best tools and practices for systematic reviews », *Syst. Rev.*, vol. 12, nº 1, p. 96, juin 2023, doi: 10.1186/s13643-023-02255-9.

[6] N. Foulquier, B. Rouvière, et A. Saraux, « Can we use artificial intelligence for systematic literature review in rheumatology? », *Joint Bone Spine*, vol. 88, nº 3, p. 105109, mai 2021, doi: 10.1016/j.jbspin.2020.105109.

[7] L. Laranjo *et al.*, « Conversational agents in healthcare: a systematic review », *J. Am. Med. Inform. Assoc.*, vol. 25, nº 9, p. 1248-1258, sept. 2018, doi: 10.1093/jamia/ocy072.

[8] « eCCo - Harnessing the power of conversational e-coaches for health and well-being through Swiss-Portuguese Collaboration ». Consulté le: 13 août 2025. [En ligne]. Disponible sur: https://ecco.esel.pt/

[9] M. P. Guerreiro *et al.*, « Conversational Agents for Health and Well-being Across the Life Course: Protocol for an Evidence Map », *JMIR Res. Protoc.*, vol. 10, nº 9, p. e26680, sept. 2021, doi: 10.2196/26680.

[10] F. Trad *et al.*, « Streamlining Systematic Reviews: A Novel Application of Large Language Models », *BMC Med. Res. Methodol.*, vol. 25, nº 1, p. 130, mai 2025, doi: 10.1186/s12874-025-02583-5.

[11] X. Mao, T. Leelanupab, M. Potthast, H. Scells, et G. Zuccon, « AiReview: An Open Platform for Accelerating Systematic Reviews with LLMs », 5 avril 2025, *arXiv*: arXiv:2504.04193. doi: 10.48550/arXiv.2504.04193.

[12] M. Ouzzani, H. Hammady, Z. Fedorowicz, et A. Elmagarmid, « Rayyan—a web and mobile app for systematic reviews », *Syst. Rev.*, vol. 5, nº 1, p. 210, déc. 2016, doi: 10.1186/s13643-016-0384-4.

[13] « Systematic Review and Literature Review Software by DistillerSR », DistillerSR. Consulté le: 14 août 2025. [En ligne]. Disponible sur: https://www.distillersr.com/

[14] F. Bolaños, A. Salatino, F. Osborne, et E. Motta, « Artificial intelligence for literature reviews: opportunities and challenges », *Artif. Intell. Rev.*, vol. 57, nº 10, p. 259, août 2024, doi: 10.1007/s10462-024-10902-3.

[15] J. de la Torre-López, A. Ramírez, et J. R. Romero, « Artificial intelligence to automate the systematic review of scientific literature », *Computing*, vol. 105, nº 10, p. 2171-2194, oct. 2023, doi: 10.1007/s00607-023-01181-x.

[16] R. van Dinter, B. Tekinerdogan, et C. Catal, « Automation of systematic literature reviews: A systematic literature review », *Inf. Softw. Technol.*, vol. 136, p. 106589, août 2021, doi: 10.1016/j.infsof.2021.106589.

[17] S. R. Jonnalagadda, P. Goyal, et M. D. Huffman, « Automating data extraction in systematic reviews: a systematic review », *Syst. Rev.*, vol. 4, nº 1, p. 78, juin 2015, doi: 10.1186/s13643-015-0066-7.

[18] M. K. Verma et M. Yuvaraj, « AI-Based Literature Reviews: A Topic Modeling Approach », *J. Inf. Knowl.*, p. 97-104, mai 2023, doi: 10.17821/srels/2023/v60i2/170967.

[19] K. Zala *et al.*, « Transformative Automation: AI in Scientific Literature Reviews », *Int. J. Adv. Comput. Sci. Appl.*, vol. 15, nº 1, 2024, doi: 10.14569/IJACSA.2024.01501122.

[20] « Latent Dirichlet allocation », *Wikipedia*. 24 juillet 2025. Consulté le: 14 août 2025. [En ligne]. Disponible sur: https://en.wikipedia.org/w/index.php?title=Latent_Dirichlet_allocation&oldid=1302196840

[21] « GROBID Documentation ». Consulté le: 14 août 2025. [En ligne]. Disponible sur: https://grobid.readthedocs.io/en/latest/

[22] M. Singh *et al.*, « OCR++: A Robust Framework For Information Extraction from Scholarly Articles », in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, Y. Matsumoto et R. Prasad, Éd., Osaka, Japan: The COLING 2016 Organizing Committee, déc. 2016, p. 3390-3400. Consulté le: 14 août 2025. [En ligne]. Disponible sur: https://aclanthology.org/C16-1320/

[23] L. Foppiano, P. B. de Castro, P. O. Suarez, K. Terashima, Y. Takano, et M. Ishii, « Automatic extraction of materials and properties from superconductors scientific literature », *Sci. Technol. Adv. Mater. Methods*, vol. 3, nº 1, p. 2153633, déc. 2023, doi: 10.1080/27660400.2022.2153633.

[24] P. Lewis *et al.*, « Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks », 12 avril 2021, *arXiv*: arXiv:2005.11401. doi: 10.48550/arXiv.2005.11401.

[25] S. Agarwal *et al.*, « LitLLMs, LLMs for Literature Review: Are we there yet? », 21 mars 2025, *arXiv*: arXiv:2412.15249. doi: 10.48550/arXiv.2412.15249.

[26] N. Meuschke, A. Jagdale, T. Spinde, J. Mitrović, et B. Gipp, « A Benchmark of PDF Information Extraction Tools using a Multi-Task and Multi-Domain Evaluation Framework for Academic Documents », vol. 13972, 2023, p. 383-405. doi: 10.1007/978-3-031-28032-0_31.

[27] L. McGregor, « Comparing Different LLM Models For Data Extraction », Medium. Consulté le: 19 août 2025. [En ligne]. Disponible sur: https://lucas-mcgregor.medium.com/comparing-different-llm-models-for-data-extraction-f067d56bed54