Universidade de Aveiro
2024

JOÃO ANTÓNIO
ASSIS REIS

UM SISTEMA CONVERSACIONAL DE PESQUISA SOBRE BASES DE DADOS MÉDICAS

A CONVERSATIONAL QUERY BUILDER ON MEDICAL DATABASES

Universidade de Aveiro
2024

JOÃO ANTÓNIO
ASSIS REIS

UM SISTEMA CONVERSACIONAL DE PESQUISA SOBRE BASES DE DADOS MÉDICAS

A CONVERSATIONAL QUERY BUILDER ON MEDICAL DATABASES

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Informática, realizada sob a orientação científica do Doutor João Rafael Almeida, Professor auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e do Doutor José Luís Oliveira, Professor catedrático do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro.

o júri / the jury

agradecimentos / acknowledgements

Em primeiro lugar, expresso minha profunda gratidão aos meus orientadores, especialmente ao Professor João Rafael Almeida. O seu apoio foi além da orientação do projeto; guiou-me ao longo de todo este processo, motivando-me a enfrentar e superar novos desafios. Agradeço também ao colaborador desta dissertação, Tiago Almeida, que sempre esteve disponível para me aconselhar e guiar no melhor caminho.

A todos os meus amigos, estou imensamente grato pelo constante companheirismo e pela partilha da vida académica ao longo dos últimos anos. À Inês, um agradecimento especial pela paciência e pelo suporte incondicional.

Por fim, dedico este trabalho à minha família que sempre me apoiou e tornou possível a minha jornada académica até aqui.

Palavras Chave

Sistema conversacional de pesquisa, Descoberta de dados biomédicos, LLM, RAG, EHDEN, OHDSI.

Resumo

A utilização e análise de dados médica em larga escala, permitiu que fosse possivel realizar estudos com mais impacto. Isto tornou-se possível devido às estratégias atuais para lidar com os problemas de interoperabilidade de bases de dados. À medida que essas estratégias foram adotadas,novos desafios surgiram, nomeadamente relacionados com a descoberta de base de dados adequadas a cada estudo. Inicialmente, isso foi tratado utilizando catálogos de bases de dados e interfaces para comparar características das mesmas. No entanto, essas soluções tornaram-se menos eficazes devido à complexidade dos dados existentes. Com o objetivo de simplificar a identificação das bases de dados adequadas e simplificar a realização de estudos distribuídos, esta dissertação propõe um sistema conversacional de pesquisa para ajudar os investigadores médicos a encontrar de forma eficiente as bases de dados mais adequadas para os seus estudos. é constituída por um modelo BM25 para sugerir as bases de dados mais adequadas, e também utilizada um modelo de linguagem para geração e processamento de texto em linguagem natural. Os resultados demonstram que o sistema conversacional de pesquisa melhora significativamente a eficiência na descoberta de bases de dados e simplifica o processo de definição de estudos observacionais.

Keywords Conversational query builder, Biomedical data discovery, LLM, RAG, EHDEN, OHDSI.

Abstract Real-world evidence in the medical domain refers to clinical evidence derived from real-world data analysis. This become possible due to current strategies to handle the database interoperability issues. As these strategies have been widely adopted, new challenges in data discovery have appeared. One of these challenges is the discovery of databases for specific health studies. Initially, this was handled using database catalogs and interfaces to compare database characteristics. However, these solutions become less effective as database networks expand. Aiming to improve the discovery of databases and simplify the execution of multicenter studies, this dissertation proposes a conversational query builder to assist medical researchers in efficiently finding the most suitable databases for their studies. The solution implements the BM25 technique for optimal database retrieval and uses a large language model (LLM) for text generation and natural language processing (NLP) tasks. The results demonstrate that the conversational query builder significantly improves the efficiency of database discovery, but more importantly, it simplifies the process of defining cohorts over observational databases.

reconhecimento do uso de ferramentas IA / recognizing the use of AI tools

I acknowledge using Grammarly (Grammarly, https://www.grammarly.com/) and ChatGPT-3.5 (Open AI, https://chat.openai.com/) to improve some sentences, namely grammatical structure, punctuation, and vocabulary.

Contents

List of Figures

List of Tables

List of Code Snippets

Glossary

EHDEN . . . . . European Health Data Evidence Network

OMOP CDM . . . . Observational Medical Outcomes Partnership Common Data Model

OHDSI . . . . . Observational Health Data Sciences and Informatics

IR . . . . . Information Retrieval

TF . . . . . Term Frequency

IDF . . . . . Inverse Document Frenquency

TF-IDF . . . . . Term Frequency - Inverse Document Frenquency

DL . . . . . Document Length

BM25 . . . . . Best Matching 25

QA . . . . . Question Answering

AI . . . . . Artificial Intelligence

NLP . . . . . Natural Language Processing

NLU . . . . . Natural Language Understanding

NLG . . . . . Natural Language Generation

LM . . . . . Language Models

LLM . . . . . Large Language Models

SLM . . . . . Statistical Language Models

NLM . . . . . Neural Language Models

PLM . . . . . Pre-trained Language Models

RNN . . . . . Recurrent Neural Networks

BERT . . . . . Bidirectional Encoder Representations from Transformers

GPT . . . . . Generative Pre-trained Transformer

ML . . . . . Machine Learning

RAG . . . . . Retrieval-Augmented Generation

RLHF . . . . . Reinforcement Learning from Human Feedback

LoRA . . . . . Low-Rank Adaptation

CoT . . . . . Chain of Thought

ReAct . . . . . Reason-Action

TII . . . . . Technology Innovation Institute

IEETA . . . . . Institute of Electronics and Telematics Engineering of Aveiro

LSR . . . . . Learned Sparse Retrieval

CHAPTER 1

Introduction

In the field of medical research, the lack of sufficient samples to conduct robust and statistically significant studies is still a challenge [1]. This problem has more impact in studies involving rare diseases, specific patient subgroups, or when analyzing rare outcomes in common diseases [2]. This lack of information can lead to underpowered studies, limiting the reliability and generalizability of the research findings [3]. Furthermore, the limitations imposed by insufficient data are significant since individual healthcare institutions often have limited patient populations, which restricts the diversity and volume of data available for research [1, 3].

Multicentre studies are a possible strategy to overcome the issue of insufficient samples used in a study [4]. By aggregating data from multiple sources, researchers may increase the number of participants and the population diversity [5]. Although this solution seems to solve the problem, it raises more challenges, namely related to the interoperability between databases [6] and privacy issues [7]. In other words, databases contain different types, formats, and/or sources, which are often not compatible with each other. The existence of these diverse data is not very effective, as they cannot be easily shared or integrated with other data.

To address the challenge of data heterogeneity in multi-institutional studies, the **Observational Health Data Sciences and Informatics (OHDSI)**[1] initiative proposed some strategies. OHDSI is a collaborative network that aims to improve health by empowering a community to collaboratively generate evidence that promotes better health decisions and better care [8]. A key contribution of OHDSI is the development of standardized data models and tools that facilitate the harmonization of observational health data [9]. The most notable is the Observational Medical Outcomes Partnership Common Data Model (OMOP CDM). The OMOP CDM standardizes the format and content of clinical data, enabling data from different sources to be integrated and analyzed cohesively [8, 10]. The OHDSI tools enable researchers to effectively overcome the barriers posed by data heterogeneity since the data would be

[1] https://www.ohdsi.org/

mapped into the OMOP CDM format. This procedure also enhances the reproducibility and scalability of research, contributing to more reliable and impactful healthcare insights [11].

1.1 Motivation

Solving the heterogeneity issues simplified the growth of multi-institutional observational databases. This raised new opportunities and challenges, considering multicentre studies [12]. One of the main challenges is the selection of the most appropriate databases for conducting a study. However, there are already some solutions to address that problem [13–15].

The European Health Data Evidence Network (EHDEN) project is one of several initiatives to simplify multicentre studies. In this project, the consortium members helped European data providers migrate their databases to the OMOP CDM data schema and publish their metadata in a web catalogue. This comprehensive catalogue offers researchers a centralized platform to explore the variety of available data sources [16]. With the increasing number of data partners, selecting databases only using metadata becomes complex, which leads to the creation of the EHDEN Network Dashboards tool[2]. It gives statistical and aggregated information about the databases available on the network. Researchers get more context to select, manually, the databases that satisfy their study requirements.

Despite these improvements, the catalogue now hosts the information of more than 200 databases. Researchers often spend considerable time and effort to find the most suitable database for their research criteria. The current methods of database discovery often rely on manual search and evaluation, which can be time-consuming and subject to human error. For these reasons, the challenge of efficiently discovering the most appropriate databases for a specific study.

1.2 Objectives

With the increasing adoption of chatbot-like tools, the main objective of this work is to develop a conversational query builder to help medical researchers define their study objectives. This approach offers a more personalized and efficient method of database discovery. Researchers can interact with the chatbot conversationally, specifying their needs and receiving tailored recommendations. To achieve this objective, the present dissertation seeks to answer the following research question:

> *How can a conversational query builder support medical researchers when defining an observational study?*

To answer this question, the work can be addressed by focusing on different aspects, namely by dividing it into three stages:

1. Study of state-of-the-art: i) methodologies to build a conversational user interface, ii) procedures to retrieve the databases of most interest, and iii) explore the definition of an observational study;

[2] https://github.com/EHDEN/NetworkDashboards

2. Develop a chat-like search engine to help discover the best databases for a study;
3. Enhance the engine to support the cohort definition for an observational study.

1.3 Dissertation outline

This dissertation is organized into several sections, each designed to build upon the previous one to provide a comprehensive understanding of the project and its findings. The document starts with a introduction to provide background information, outlining the motivation, the problem and the proposed solution.

The introduction is followed by Chapter 2, which delves into the state of the art. This chapter reviews the current literature and technologies relevant to the study. It aims to explore areas such as information retrieval procedures to access the most appropriate databases, large language models, methodologies to build a conversational user interface, interactive query builders, and the definition of a study protocol.

The Conversational Search Assistant, described in Chapter 3, aims to develop a chat-like search engine to help discover the best databases for a study. The Query Builder, outlined in Chapter 4, explains how to enhance the engine to collect additional information to provide a query as an outcome.

Chapter 5 presents the findings of the study, analyzing and interpreting the results in the context of the research objectives. To conclude the dissertation, the Chapter 6 summarizes the findings and contributions of the project to this field.

1.4 Contributions

This dissertation led to scientific contributions during its development due to its promising solution for a real problem. The following are the conference papers and posters that originated from this dissertation:

Conferences

- "Using Flowise to Streamline Biomedical Data Discovery and Analysis", 22nd IEEE Mediterranean Electrotechnical Conference (MELECON), 2024 [17].
- "A chatbot-like platform to enhance the discovery of OMOP CDM databases", 34th Medical Informatics Europe (MIE) Conference, 2024 [18].
- "HealthDBFinder: a question-answering task for health database discovery", 37th IEEE International Symposium on Computer-Based Medical Systems (CBMS), 2024 [19].

Posters

- "A Chatbot to help discover OMOP DCM databases within EHDEN Network", OHDSI Europe Symposium, 2024.

CHAPTER 2

State of Art

This dissertation suggests the development of a conversational query builder that functions as a chat-based interface within the EHDEN project ecosystem. This interface aims to help researchers redefine their studies effectively. The conversational assistant should return a ranked database list to help discover the best databases for specific research and also collect additional information to define an observational study. To accomplish these goals, Information Retrieval (IR) is a crucial field to study because it can retrieve the most appropriate databases for a researcher's query. In addition, it is crucial to explore the recent advancement of algorithms, such as Large Language Models (LLM), known for their language generation ability. Understanding the current state of conversational user assistants or chatbots is also essential.

2.1 Information Retrieval

In computing and information science, IR involves retrieving information from collections of unstructured data, such as documents. According to Kumar *et al.* [20], an IR system requires input user queries and uses them to retrieve information from its collections, aligning with the users' needs. This process efficiently filters and narrows down the vast amount of available unstructured data. It prevents users from being overwhelmed by the sheer volume of data and aiding them in quickly accessing relevant and specific content.

In conformity with Hambarde and Proença [21], conventional text retrieval systems were predominant in the initial stages of the IR field. These systems mainly depended on matching terms between queries and documents. However, these conventional IR systems have limitations, including issues like polysemy, synonymy, and linguistic gaps, which may restrict their effectiveness [21].

With the advancement of technology, deep learning techniques emerged, improving conventional text retrieval systems and overcoming the constraints associated with term-based retrieval methods. For this reason, the performance of these systems increased significantly, resulting in a more accurate and streamlined retrieval of information for end-users [22].

Deep learning methods have advanced, with the emergence of neural network architectures, transfer learning, and pre-training techniques. These approaches have advanced the representation of textual data and bolstered the IR system's comprehension of natural language queries [22].

More recently, transformer architectures with attention mechanisms have been implemented in IR systems to enable more effective handling of complex queries and documents. Moreover, incorporating pre-trained language models like Bidirectional Encoder Representations from Transformers (BERT) [23] and Generative Pre-trained Transformer (GPT)-2 has proven to enhance the performance of IR systems. These methods offer an advanced understanding and processing of context, capturing the relationships and nuances within the natural language query and the documents.

This field has many applications in the real world, such as search engines like Google, digital libraries, and e-commerce platforms. In compliance with Kumar *et al.* [20], IR generally functions across three main scales: i) searching the web, ii) retrieving personal information, and iii) conducting searches for enterprises, institutions, and domain-specific contexts. This section describes the IR field, more specifically, traditional methods, neural IR systems and how IR can be joined with natural laguage.

2.1.1 Traditional methods

Some successful classical methods are Term Frequency - Inverse Document Frenquency (TF-IDF) and Best Matching 25 (BM25).

TF-IDF

To better understand the TF-IDF method, first, it is necessary to understand the concepts of Term Frequency (TF) and Inverse Document Frenquency (IDF).

Starting with TF-IDF method, it is composed of the TF and the IDF, two essential concepts to understand this classical IR method. Liang and Niu [24] explained these concepts in their work. It is reasonable to assume that a document containing a query term more frequently is more relevant to that query. Therefore, it should be assigned a higher relevance and/or score. TF is the number of term occurrences in a document. However, to evaluate the relevancy of a query, each term is regarded with equal importance. Manning *et al.* [25] provided the following example to better explain TF: the automotive industry is expected to include the term "auto" in nearly every document. The most frequent terms are not always the most important. Equation 2.1 shows how the TF is calculated.

$$\mathrm{tf}_{t,d} = \frac{\text{Term } t \text{ frequency in document } d}{\text{Total words in document } d}. \tag{2.1}$$

In order to reduce the weight of terms that appear frequently, the IDF helps to prioritize some terms that are sporadic and possibly more informative and relevant [24] . The IDF calculates the rarity of a term across a set of documents. This measure is calculated as the logarithm of the division of the total number of documents by the number of documents in the collection containing the term, as the Equation 2.2 describes. Aligning with the previous

example of the automotive industry, the generic term "auto" will have a low IDF value. However, the term "telematics" will have a high IDF value, underlining the unique and informative significance of the term for distinguishing documents.

$$\text{idf}_t = \log(\frac{\text{Total documents}}{\text{Documents with term } t}). \tag{2.2}$$

The TF-IDF method combines TF and IDF to assign a weight to each term in a document, highlighted by as Manning *et al.* [25] and Lan [26]. Equation 2.3 shows that the weight is calculated by multiplying the TF and IDF values. So, it is possible to highlight terms that are both important within a specific document and relatively uncommon in the document collection.

$$\text{tf-idf}_{t,d} = \text{tf}_{t,d} \times \text{idf}_t. \tag{2.3}$$

TF-IDF does not consider the semantic information of words, which limits its ability to accurately reflect the similarity between texts [26]. In summary, this IR method evaluates the importance of a term within a document relative to its occurrence across a collection of documents.

BM25

BM25, the short form for Best Matching 25, is a ranking algorithm for IR systems, especially in the context of search engines. Hambarde and Proença [21] stated that BM25 and other initial retrievers are employed for their effectiveness in recalling pertinent documents from an extensive pool. The core components of BM25 include TF, IDF, Document Length (DL), and tuning parameters. Equation 2.4 gives the formula to calculate the BM25 score.

$$score(D,Q) = \sum_{i}^{n} IDF(q_i) * \frac{f(q_i,D) * (k1+1)}{f(q_i,D) + k1 * (1 - b + b * \frac{fieldLen}{avgFieldLen})}. \tag{2.4}$$

As Gomede [27] explained, the BM25 equation is composed of the *ith* query term (q) and the respective IDF and TF values. Also, include a division between DL (fieldLen) and the average document length (avgFieldLen). This ratio evaluates how much the length of the document field deviates from the average length. Connelly [28] explained intuitively: *a document tends to receive a lower score when it contains more terms, especially those that do not match the query.* The value b is a fine-tuning parameter, and it is responsible for length normalization. When b is larger, the ratio has a more significant effect on the overall score. Finally, the *k1* value means term frequency saturation. It is a fine-tuning parameter that prevents the term frequency component of BM25 from having an unlimited impact on the document score.

This algorithm can be simple and effective in IR tasks, mainly search tasks. Also, it can handle large collections. For these reasons, it is widely used and called a classic. However, BM25 can not perform a semantic analysis of the query and the documents.

Traditional IR methods like TF-IDF and BM25 have been effective in matching queries to documents based on keyword frequencies and document lengths. However, the evolution of

Artificial Intelligence (AI) has led to the development of neural IR systems to adress some limitations of these classical methods.

2.1.2 Neural information retrieval systems

Neural IR systems represent a significant advancement in the field of IR. Conforming to Mitra and Craswell [22], these systems use neural network models and deep learning techniques. This approach can understand and interpret the semantic content of queries and documents. Neural IR systems can be broadly categorized into two types: representation-based and interaction-based models.

Conforming to Chen *et al.* [29], the representation-based model focuses on creating representations of both queries and documents. They employ techniques to convert text into high-dimensional vector spaces. Semantically similar terms and phrases are represented by vectors that are close to each other, capturing the underlying meaning and relationships of words beyond their surface-level appearances. This approach employs architectures like Recurrent Neural Networks (RNN) and allows the system to understand the content of documents and queries on a deeper level, as Liu *et al.* [30] mentioned.

The interaction-based model examines how words in a query relate to words in a document, capturing complex patterns and dependencies between them [29]. They often use attention mechanisms, as seen in transformer-based models like BERT, to dynamically weigh the importance of different parts of the text based on their relevance to the query. However, only the interaction-based model will be explored.

There are different approaches to implementing an interaction-based model. One of these approaches involves a retrieval stage and a ranker stage [22]. 2.1 shows an overview of interaction-based IR systems, highlighting the two main stages.

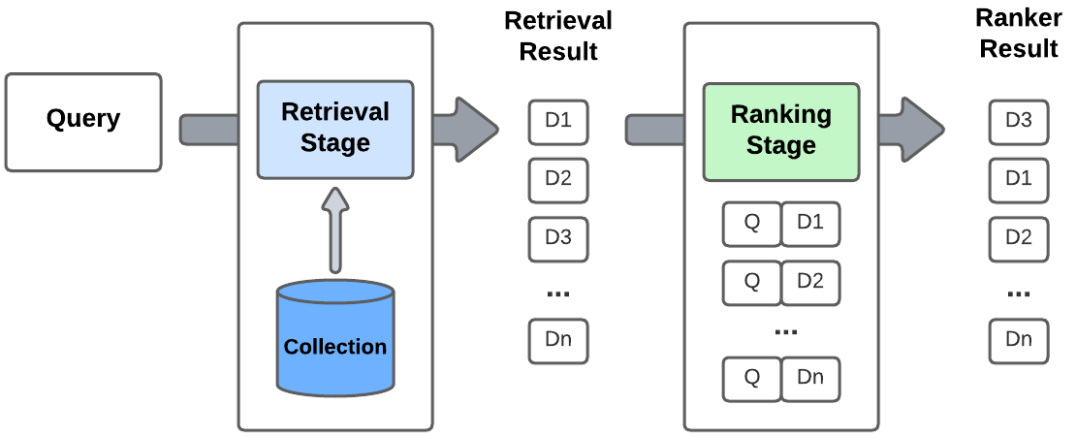

Figure 2.1: Overview of an interaction-based Information Retrieval model: Retrieval and Ranker.

After analyzing the query, the retrieval stage selects an initial set of documents that are potentially pertinent to the query, as shown in Figure 2.1. Subsequently, the relevance of these documents undergoes reassessment through the similarity scores. This is followed by the

ranking stage, in which the primary objective is to adjust the order of the initially retrieved documents based on their relevance scores using a neural reranking, like BERT [29]. This phase prioritizes the enhancement of result effectiveness rather than efficiency. In the end, it returns a rank of documents as close as possible to the user's query criteria.

2.1.3 Question answering

Natural Language Processing (NLP) is the basis for building a Question Answering (QA) system. The NLP can be divided into two major components: Natural Language Understanding (NLU) and Natural Language Generation (NLG), according to Ayanouz *et al.* [31].

The NLU component is focused on enabling machines to understand and interpret human language in a meaningful way. NLU involves processing and analyzing natural language data to comprehend its meaning, context, sentiment, and intent.

In agreement with Ngai *et al.* [32], the user queries can be processed by semantic analysis, pragmatic analysis, and syntactic analysis. Ayanouz *et al.* [31] explained these steps and added two more necessary steps to make it easier to understand: a lexical analysis and discourse integration. The process of query analysis by NLU involves the following steps:

- **Lexical Analysis**: This step involves analyzing and identifying the structure of words. It breaks down the text into chapters, sentences, phrases, and words. Chizhik and Zherebtsova [33] defined lexical analysis as the pre-processing of the text following the steps: tokenization, removal of special characters, links, and punctuation, and removal of stop-words.
- **Syntactic Analysis**: The syntactic analyzer parses the grammar and arrangement of words, making the relationships between different words more explicit. Essentially, it rejects sentences with incorrect structures. This analysis can be seen as the process of normalizing tokens.
- **Semantic Analysis**: This step ensures the text is meaningful and interprets its correct meaning by mapping syntactic constructions. It ensures that only semantically valid content is retained. The recognition of entities is part of this analysis.
- **Pragmatic Analysis and Discourse Integration**: This step analyzes the overall context to derive the conclusive interpretation of the actual message in the text. It considers factors like the true meaning of a phrase or sentence based on the broader context.

The other component is NLG. Language generation is responsible for crafting coherent and linguistically accurate responses. It transforms data into natural language text, enabling effective communication and information dissemination [32].

Backtracking, QA is a subfield of IR and NLP. According to Zhong *et al.* [34], QA focuses on providing a single and specific answer to a question posed in natural language. Unlike IR, which aims to return a broad range of relevant information or documents in response to a query, QA seeks to pinpoint and provide one precise answer.

The traditional approach to question analysis and answering often involves mapping questions into predefined templates, such as "What-type" and "How-type". While widely used by existing online question-answering search engines, this template-based approach faces limitations in handling multiple questions [34].

With the advancement of technology, another approach emerged: deep learning-based question-answering [33]. In contrast with the traditional approach, this approach employs deep learning techniques, like RNN, to offer automatic representation and analysis of questions. These neural models, trained through end-to-end approaches, excel in extracting and understanding complex characteristics in textual documents.

Recently, deep learning approaches with attention mechanisms and transfer learning have enhanced the flexibility of representation in text classification and named entity recognition. Zhong *et al.* [34] highlights BERT that has emerged as a powerful model, using contextualized representations for transfer learning. BERT-based models showcase performance in question-answering tasks, even in domains like medicine.

2.2 Large Language Models

Before LLM, there were only simple Language Models (LM), a subfield of NLP and AI, that have been called foundation models. A LM is a statistical model used to predict the next word in a sequence of words. It calculates the probability of a given word occurring in a sequence, helping to determine which words are likely to appear next in a given context [35].

Most of these predictive models were based on probabilities and Markov assumptions, also known as Statistical Language Models (SLM). This was heavily dependent on feature engineering. Afterward, as deep learning gained prominence, an architecture designed to learn data features automatically. In other words, neural networks for NLP emerged to enhance LM's capabilities. Integrating feature learning and model training, Neural Language Models (NLM) established a comprehensive neural network framework applicable to diverse NLP tasks [36].

Most recently, the launch of the transformer Block Architecture by Vaswani *et al.* [37] revolutionized this field. These deep-learning architectures led to the development of pre-trained models not explicitly designed for a particular task, including BERT and GPT, collectively known as Pre-trained Language Models (PLM). PLM have shown significant performance enhancements across various NLP tasks.

The scale of the parameters of these models has been involved, and the paradigm of "Pre-train, Prompt, Predict" like Liu *et al.* [36] described, gained widespread acceptance. In terms of interaction with LM, the prompts became crucial. Researchers name these PLM with hundreds of billions of parameters as LLM. Prompts effectively allow LLM to deal with a large number of complex and diverse tasks without a lot of effort.

2.2.1 Definition

LLM is designed to comprehend and generate text that is coherent and contextually relevant, engaging in human language interactions. Essentially, these advanced AI systems mimic

human speech. These advanced AI systems have a notable ability in natural language tasks, such as text generation and translation, question-answering, decision-making, summarization, and sentiment analysis [38].

These models can process and predict patterns with accuracy. Hadi *et al.* [39] combine sophisticated SLM and deep learning techniques to train, analyze, and understand huge volumes of data, learning the patterns and relationships among the data. For this reason, according to Naveed *et al.* [40], when provided with task descriptions and examples through prompts, LLM can produce textual responses to task queries.

Liu *et al.* [36] noted that the release of the first version of ChatGPT garnered significant social attention, and research into LLM triggered more interest. This has led to the development of noteworthy products like PaLM [41], GPT-2, GPT-3, and, most recently, GPT-4 [42], and LLaMA and LLaMa-2 [43].

2.2.2 Architecture overview

The development and advancement of LLM is thankful for the introduction of transformers by Vaswani *et al.* [37] in 2017. Most LLM are built on the transformer model, which is based on a multihead self-attention mechanism and feedforward layers. This new technology enables parallelization and efficient handling of long-range dependencies, according to Hadi *et al.* [39], and led to the development of models that have achieved enormous results, such as GPT by OpenAI and BERT by Google.

Transformer architeture

The architecture of this revolutionized model is shown in Figure 2.2. The transformer architecture uses stacked self-attention and point-wise fully connected layers for both the encoder and decoder. These components are illustrated in the left and right halves of Figure 2.2, respectively.

The innovation of this model is due to the multihead self-attention mechanism, which is one of its key components [37, 39]. It allows the model to weigh the importance of different words in a sequence when processing each word. This mechanism enables the model to focus on relevant information, capturing dependencies regardless of word order. However, the key advantage of this multihead self-attention mechanism is its highly parallelization [37]. This characteristic enables the transformer model to be easily distributed and trained on a large scale using GPUs. The ability to parallelize computations means that transformers can handle larger datasets and more complex tasks, unlike previous architectures like RNN, where sequential processing of data was required.

Since the model does not have recurrence and convolution to understand the order of the input sequence, another component, position encoding, provides some information about the position of the tokens in the sequence. This is crucial for capturing sequential information in the data.

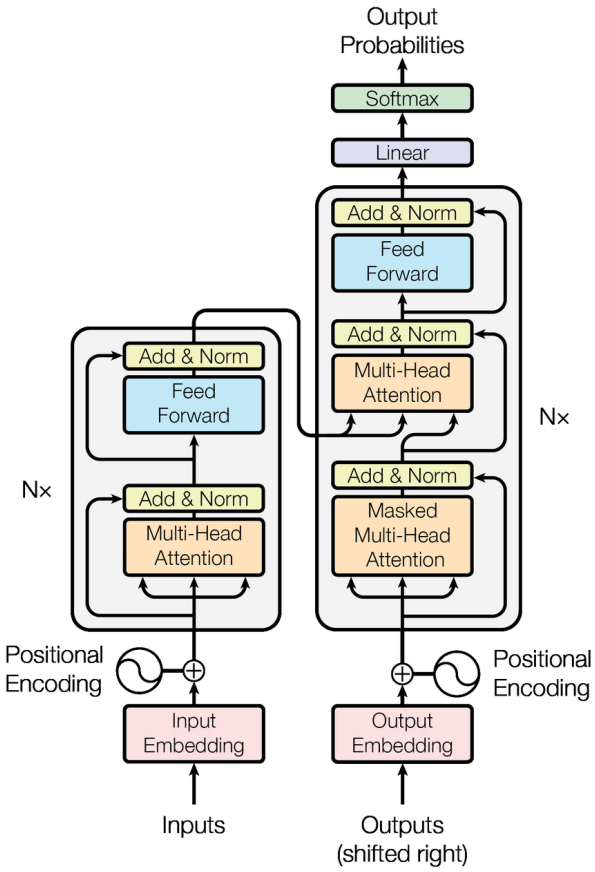

Figure 2.2: The transformer block architecture, from Vaswani *et al.* [37].

Pre-training process

Learning the patterns and relationships among the data starts with the pre-training process. In compliance with Min *et al.* [44], the LLM needs to access a vast volume of textual data from multiple sources. The goal of this phase is to predict the succeeding word in a sentence based on the context given by the previous words through unsupervised learning.

According to Hadi *et al.* [39], preparing and preprocessing the data before the training stage is necessary to achieve this. The fisrt step is demand quality filtering from the training corpus. It is vital to remove unwanted, repetitive, duplicated, superfluous, and potentially harmful content from the massive text data. Next, it is necessary to pay attention to privacy. The data could have sensitive or personal information, so it is vital to address privacy concerns by removing this information from the pre-training corpus.

An important step, the tokenization, follows this [39]. This step aims to divide the unprocessed text into sequences of individual tokens, which are subsequently input into LLM. Moreover, it is vital in mitigating the computational load and enhancing efficiency during the pre-training phase. Figure 2.3 visually presents the tokenization process [45] carried out and explained by OpenAI. In the example of the Figure 2.3, the text inserted has 58 tokens. The colors are meaningless and used to illustrate each token. After the pre-training process, the LLM goes through an optimization phase.

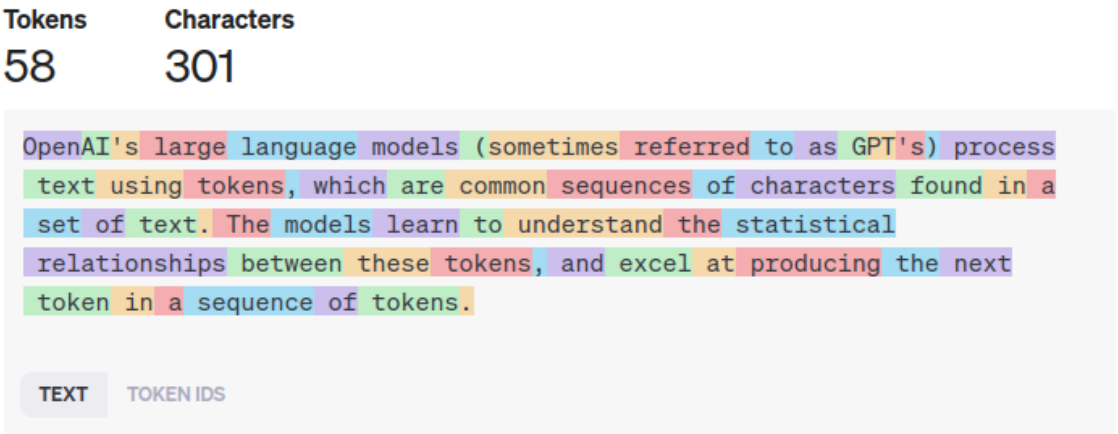

Figure 2.3: Tokenization by GPT-4 example [45]: the colors are used to illustrate each token.

2.2.3 Optimization techniques

There are some techniques to optimize the tasks and the accuracy of the LLM. Fine-tuning and prompt engineering are techniques commonly used to enhance the performance of models for specific tasks. These techniques play a crucial role in customizing models and improving their precision for specific applications.

Fine-tuning

During pre-training, models are generally trained with the objective of next token prediction, learning the nuances of language structure and semantics. According to Kamnis [46] and Hadi *et al.* [39], the fine-tuning phase involves adapting a pre-trained model to specific tasks and aligning it with human preferences, improving the performance on particular domains.

In this stage, the model is presented with labeled data to produce more contextually accurate responses for the specific task. Fine-tuning enables the LLM to specialize in diverse applications, ranging from language translation and question-answering to text generation.

Some approaches could be applied to fine-tune the model. Naveed *et al.* [40] distinguishes some of them, such as parameter-efficient tuning. As LLM typically requires a lot of computational resources, like memory and computing, the parameter-efficient tuning approach is helpful because it allows the model to train by updating fewer parameters, adding new ones, or selecting existing ones. Inside this approach, there are also some different methods. The commonly used indicated by Naveed *et al.* [40] are prompt tuning, prefix tuning, and adapter tuning.

The prompt tuning method integrates trainable tokens, named soft prompts, to the beginning or within the input of a LLM, and only these tokens are adjusted during training to adapt the model for a specific task. This method keeps the rest of the model unchanged, ensuring the core knowledge and capabilities of the model are preserved while it learns to handle new types of requests or information.

In prefix tuning, a sequence of trainable tokens is introduced to transformer layers, with only the prefix parameters undergoing fine-tuning, while the remaining model parameters

remain unchanged. These added prefixes function as virtual tokens, allowing input sequence tokens to attend to them during processing.

Meanwhile, in adapter tuning, small modules called adapters are added inside each layer of the transformer. These adapters can be trained to adapt the model for specific tasks. The fine-tuning process works by slightly altering the model's internal features, allowing it to learn task-specific patterns without changing the entire model. Low-Rank Adaptation (LoRA) is one technique that implements Adapter Tuning, introduced by Hu *et al.* [47]. Instead of adding new layers like traditional adapters, LoRA learns low-rank matrices that are used to update the weights of the existing layers, maintaining the original weights of the model. This approach allows for efficient fine-tuning of the model on specific tasks while maintaining the model's original performance and avoiding significant increases in computational costs.

Prompt engineering

With the emergence of LLM, other research fields were born. Prompt Engineering is one of these cases and has been widely applied. In compliance with Meskó [48] and Ma *et al.* [49], this emerging field involves designing, refining, and implementing prompts or instructions to direct the generated output of LLM, aiding in diverse tasks. LLM can follow specific directions provided by users in natural language after being tuned with instructions.

There are some techniques of prompt engineering, such as Chain of Thought (CoT) and Reason-Action (ReAct). CoT is a popular problem-solving approach for prompt engineering that aims to break complex tasks into multiple and simpler subtasks and solve them. Wei *et al.* [50] explained that this method involves explicitly modeling the reasoning processes that lead to a final answer, rather than directly generating an answer. This explanation of reasoning often leads to more accurate results. The example of the Figure 2.4 contrasts standard prompting with CoT prompting. In the standard prompting, the LLM prompt is a math problem with an example that provides directly the respective answer. This approach leads to an incorrect response. Conversely, the CoT prompting includes the same math problem but with an example that provides a step-by-step breakdown of the calculations. The detailing of reasoning makes the difference and leads to the correct answer.

ReAct is a prompt technique introduced by Yao *et al.* [51]. The idea behind this is to simultaneously include both reasoning and action within a single prompt. To solve a complex task, ReAct consists of three tasks for every subtask: 1) **Reason** involves analyzing the current situation and determining the necessary steps; then, 2) **Action** entails executing a task based on the reasoning. 3) **Observation** then refers to examining the outcomes following the action.

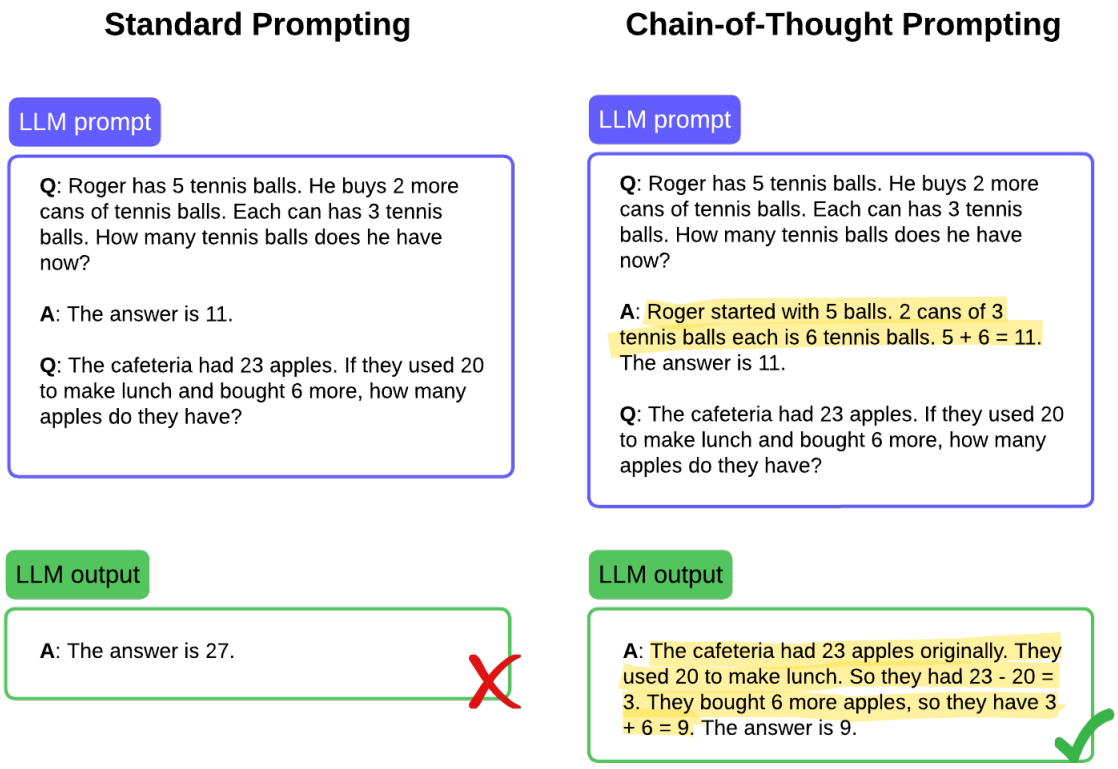

Figure 2.4: Example of Chain-of-Thought prompting. The reasoning processes are highlighted in yellow. Adapted from Wei *et al.* [50].

2.2.4 Comparison between foundation Large Language Models

The best way to compare LLM is to evaluate the model's performance. Hadi *et al.* [39] identified five factors to make this comparison: the size of the training corpus, the quality of the training corpus, the number of parameters, the complexity of the model, and some test tasks.

The primary foundation models of LLM are GPT-4 by OpenAI, LLaMA 2 by Meta, PaLM 2 by Google, and Falcon by Technology Innovation Institute (TII). These LLM are provided by big companies and have outstanding progress in the evolution of this area. These models gave rise to many others.

LLama 2 [43] is an open source LLM by Meta. LLaMa 2 was trained on 40% more data than LLaMa, the model from which it came, and has double the context length. The model size of LLaMa 2 is 7 billion, 13 billion, or 70 billion parameters. With 4096 context length and trained on 2 trillion pretraining tokens, this LLM is commonly fine-tuned for chat use cases. Many other models, like Alpaca, Vicuna, and Llama-2-chat, came from LLaMa and deserve further analysis. It is accessible for both research and commercial purposes

The recent GPT model from OpenAI, GPT-4 [42], is a closed source LLM. Trained on a meticulously curated dataset from various textual sources, including books, articles, and websites, GPT-4 exhibits remarkable performance with text and image inputs. It is the LLM behind ChatGPT. It has 32 000 context length. OpenAI has chosen to provide limited

technical details about the training methodology used for this advanced model, including specific information on parameter counts.

The Google generative chatbot, Bard, uses as LLM the PaLM 2 model [41] developed by Google. It emerged from PaLM with 540 billion parameters. PaLM 2 is a closed source LLM that opted to disclose limited technical specifics, as the GPT-4 by OpenAI.

The Falcon LLM is an open-source model with impressive performance and scalability [52]. There are three variations of the model size: 7 billion, 40 billion, and the most recent, 180 billion of parameters. The Falcon 180B is equipped with an impressive 180 billion parameters and trained on 3.5 trillion tokens. It is accessible for both research and commercial purposes.

The Mistral[53] was introduced by Mistral AI. It is an open source LLM with 7 billion of parameters. This model has 8 192 context length and trainde on 128K tokens. Mistral 7B originated two other models: Mixtral 8x7B [54] and Mixtral 8x22B. The table 2.1 summarizes the important aspects of comparison between this foundation LLM.

Model	**Provider**	**Model size (Parameters)**	**Context Length**	**Tokens**	**Fine-tuneability**	**Open-source**
GPT-4	OpenAI	-	-	-	No	No
LLaMa 2	Meta	7B, 13B, 70B	4096	2T	Yes	Yes
PaLM 2	Google	-	-	-	No	No
Falcon	TII	7B, 40B, 180B	2048	3.5T	Yes	Yes
Mistral	Mistral AI	7B	8192	128K	Yes	Yes

Table 2.1: Comparison of foundation Large Language Models.

The table comprises some important aspects, such as if the model is opened or closed source. The source availability of the model is an important aspect in order to choose the LLM for this project. An open-source LLM is more available for modification. The GPT and PaLM models have this less positive point.

2.2.5 Limitations

According to Liu *et al.* [36], the abilities of LLM, mainly in-context learning, reasoning for complex content, and creative capacity, proved the versatility of these advanced LM. However, LLM has some limitations. Hadi *et al.* [39] address some of them, and the most important ones are biased responses, hallucination, explainability, and cyber-attacks.

We already know that LLM are pre-trained with extensive training data. But suppose that data contains some biased information related to factors such as gender, socioeconomic status, and/or race. In that case, this may result in analyzes and recommendations that are discriminatory or inaccurate across diverse domains. The problem of bias applies not only to training data but also to user interaction bias, algorithmic bias, and contextual bias. The user interaction bias means that, as user prompts shape responses, and if users consistently ask biased or prejudiced questions, the model may acquire and reinforce these biases in its replies.

A severe limitation that is an active area of research is hallucination. Church and Yue [55] characterized LLM hallucinations as when the model attempts to fill gaps in knowledge or

context, relying on learned patterns during training. Such occurrences can result in inaccurate or misleading responses, detrimental to the user and the model's reliability.

The way the LLM makes decisions is unknown. Comprehending the decision-making process of a complex model with billions of parameters is difficult. The explainability of these models is a big limitation [39]. Sometimes, it is necessary to decipher the factors that influenced an LLM's decision and this limitation poses difficulties in offering a clear and concise explanation. In vital sectors like healthcare, where decisions carry substantial consequences, ensuring transparency and the capability to elucidate the model's predictions is essential.

Another limitation is the cyber-attacks. A LLM can suffer some prompt injections from a malicious user to extract sensitive information from the model, according to Kshetri [56]. This is called the Jail Break attack [39]. Another attack is Data Poisoning Attacks, which consist of data poisoning strategies to manipulate the model's output.

Furthermore, Liu *et al.* [36] identified another limitation: the outdated nature of the training data. LLMs cannot access real-time information, meaning the generated responses may not be the most current.

2.3 Conversational user assistants

Conversational user assistants, also known as chatbots, chatterbots, or virtual assistants, have become a vital aspect of the digital landscape. These tools are generally dialogue systems that understand, interpret, and generate human language, enabling them to communicate with users to dissolve their questions [57].

Chatbots are increasingly being used in various contexts due to their many benefits. These aspects that make companies bet on the use of chatbots are the continuous availability to support and assist the customer, ensuring more consistent support; the cost-efficiency by reducing the human customer support; the time-saving both for the organization and for customers due to the immediate responses to the user queries; the ease and intuitiveness of this systems; and, improve service with every interaction [58]. Because of this, the utility of the chatbots as tools is increasing as the technology advances. The rise of conversational user assistants is underpinned by a convergence of technologies, specifically by LLM.

2.3.1 Overview of conversational user assistants

Nuruzzaman and Hussain [59] defined the differences between chatbots with opened or closed domains. In an open-domain environment, conversations can go in any direction without a predefined goal or intention. Conversely, in closed-domain environments, the conversation is centered on a particular topic. A closed-domain chatbot is designed with a clear objective.

Peng and Ma [60] distinguish three main types of chatbots based on their response generation: rule-based, retrieval-based and generative-based chatbots. A **rule-based chatbot** examines fundamental features of the user's input statement and generates a response based on a predefined set of manually crafted templates. This type is more applicable in a closed-domain conversation. ELIZA, introduced by Weizenbaum [61], was the first chatbot that applied this primitive technique.

A **retrieval-based chatbot** chooses a response from an extensive precompiled dataset. It selects the most promising reply from the top-k ranked candidates. Thus, they refrain from producing new text. It has limited flexibility regarding closed-domain and in terms of errors [62].

A **generative-based chatbot** generates a text sequence as a response rather than choosing it from a predefined set of candidates. These chatbots are very flexible and can handle open domains because they are implemented with deep learning techniques. The interactions will be more identical to those of humans, as it implements a self-learning method from a large quantity of interaction data [60, 62]. However, this could be complex and costly to implement.

2.3.2 Generative-based chatbot

Generative-based conversational user assistants are chatbots that use generative models to generate natural language responses. These chatbots use sophisticated deep-learning techniques, such as LLM. These have the ability to understand and generate human-like text in context. ChatGPT is an example of this type of chatbot.

Although an LLM can generate text from a query, it is not prepared to be applied to a chatbot. There are some techniques for improving and optimizing the model to behave like a chatbot, such as Reinforcement Learning from Human Feedback (RLHF). In addition, some techniques aim to combat some of the limitations of chatbots, such as hallucination, by increasing their knowledge, such as Retrieval-Augmented Generation (RAG).

Reinforcement learning from human feedback

According to Li *et al.* [63], RLHF is a popular approach in which an agent learns how to perform a task based on evaluative feedback provided by a human observer. It is a topic that has been explored in the field of conversational AI. This technique is a transformative technique that combines reinforcement learning and supervised learning to refine LLM for chatbot applications. Tran *et al.* [64] stated that RLHF aims to align chatbot responses to human preferences, improving chatbots' performance and making them more human-like.

The process encompasses several crucial stages, in conformity with Axelsson *et al.* [65]. Initially, the LLM is pre-trained on a large dataset of text, which allows it to learn a wide range of language patterns and knowledge. After pre-training, the model undergoes a phase of supervised fine-tuning. In this phase, the LLM is trained on a dataset of conversational examples that are specifically curated to reflect the desired outputs for the chatbot.

Afterwards, humans provide feedback on the model's outputs. This feedback is crucial because it is used to build and train a reward model. The reward model learns to predict the quality of the model's responses based on the human-provided feedback [65].

The LLM is further fine-tuned using reinforcement learning, where it learns to generate responses that maximize the predicted reward, using the reward model created in the previous phase. This stage enables the model to optimize its responses through the reward model, which is based on human feedback.

The process often involves several iterations of feedback and fine-tuning to continually improve the chatbot's performance. This can resolve errors, improving the refining the conversational style.

Retrieval-augmented generation

RAG is a subfield of NLP and AI. This approach was introduced by Lewis *et al.* [66] in 2020 and combines retrieval-based and generative models to enhance content generation and information retrieval processes. For a clearer comprehension of this method, Gao *et al.* [67] made a survey into RAG systems and distinguish the parametric knowledge from non-parametric knowledge.

Traditionally, LLM can adapt their knowledge and responses to a specific domain by fine-tuning models with parameters. This is parametric knowledge because the LLM knowledge is provided through the model's training data. However, entirely parameterized LLM have limitations, including data not currently updated and hallucinations. The non-parametric knowledge, provided by external information sources, emerged to solve these limitations. This non-parametric knowledge approach is known as RAG.

RAG involves retrieving pertinent information from external knowledge bases, providing the LLM with up-to-date and domain-specific context to enhance response accuracy and relevance, thereby reducing hallucinations. According to Lewis *et al.* [66], this process aims to retrieve relevant information from a vast corpus of documents and incorporate it into the prompt to improve the quality of predictions.

Using the Figure 2.5 as an example, Gao *et al.* [67] explain simply the workflow of a RAG. The first step is 1) Retrieve. It aims to retrieve information from an external data source, such as a vector database. This step uses IR models, such as BM25, to retrieve relevant information based on the query. The second step is 2) Augment, that aims to add the information retrieved as context to the LLM prompt. The last step is 3) Generation. Using the prompt with context, the LLM generates a response to the query, based on the external information retrieved.

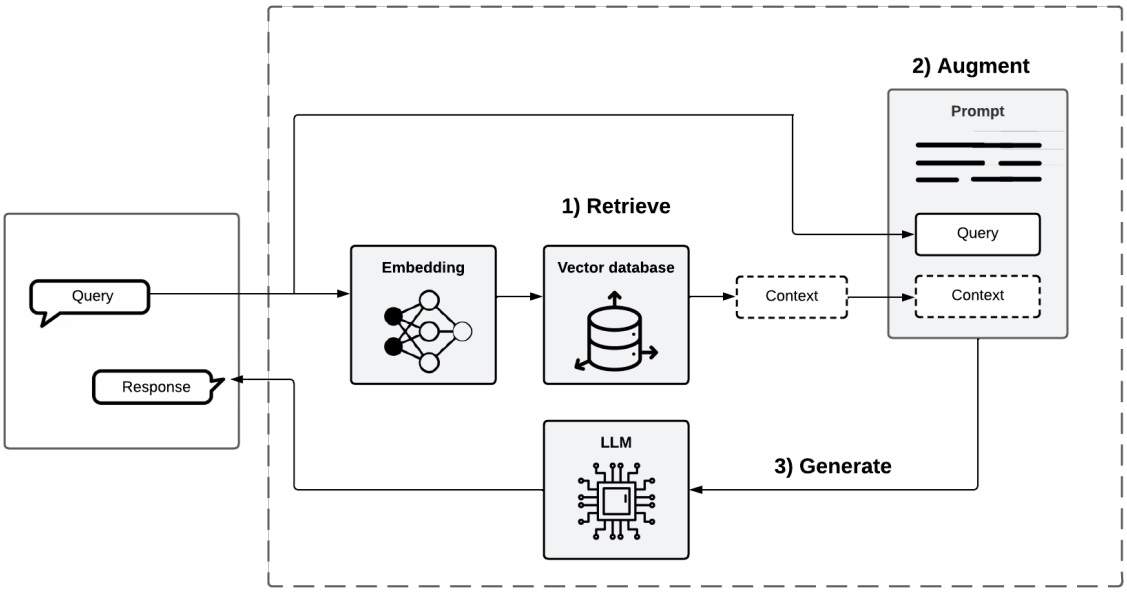

Figure 2.5: Retrieval-Augmented Generation workflow.

2.4 Interactive query builder

A query builder is a user interface tool for dynamically searching and filtering database objects, constructing a query according to user preferences, as the work of Mussa *et al.* [68] shows. This query could be in different formats, such as SQL and JSON. This tool lets users construct queries visually, eliminating manual research or coding.

This section is particularly significant as there is limited documentation on conversational query builders. Therefore, it documents the general workings of query builders and delves into a detailed explanation of the functioning of the ATLAS cohort definition.

2.4.1 General query builders

Users interact with the query builder through a user-friendly interface. Figure 2.6 shows a query builder interface from jQuery QueryBuilder [69].

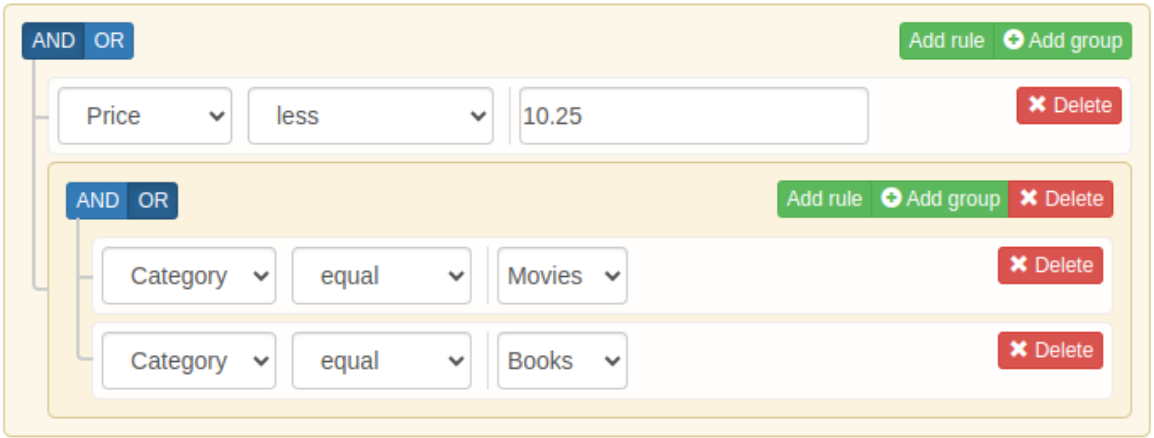

Figure 2.6: Simple query builder from jQuery QueryBuilder [69].

Users can add rules and conditions/groups with some clicks. Each rule typically consists of a field, an operator, and a value. The conditions/groups could be an AND or an OR. Using figure 2.6 as an example, there is a group with two elements joined with a condition AND: a rule and another group. This other group is composed of two rules joined with a condition OR.

As users build their queries, the query builder internally represents the conditions in a structured format, often a structured JSON of rules and groups, that reflects the logical structure of the query [69]. In summary, a query builder simplifies creating complex queries by providing a visual and interactive interface, making it more accessible [70].

2.4.2 OHDSI cohort creator

OHDSI provides diverse open-source tools to support various use cases on observational patient-level data. One of these software tools is ATLAS[1]. ATLAS is a freely accessible, open-source, web-based software developed by the OHDSI community. It aids in helping researchers conduct scientific analysis on standardized observational data converted to the OMOP CDM.

Using healthcare claims data, researchers can define cohorts by categorizing groups of people according to their exposure to a medication or their diagnosis of a certain health condition. ATLAS offers the functionality to search medical concepts, enabling the identification of cases with particular conditions or drug exposures. Moreover, it allows for the examination of patient profiles within a given cohort, providing a way to visualize the healthcare records of specific subjects.

There are some different definitions of cohort, but, in the OHDSI research, a cohort is a query that defines a set of persons who meet certain inclusion criteria over a specified duration [71]. Cohorts serve as fundamental units for addressing research questions. A key characteristic of these cohorts is their independent definition. The distinct structure facilitates their reuse across different research contexts.

Cohort definition

In ATLAS, the process of defining a cohort is composed of 3 stages [71]: cohort entry events, inclusion criteria, and cohort exit.The creation of a cohort starts with cohort entry events, defining the initial event criteria. This involves the primary identification of the population of interest, which might include users of a certain drug, individuals with a specific diagnosis, or a combination of factors. The concept set needs to be specified in the cohort entry events. It is a collection of standardized medical concepts used to define clinical elements like diseases, drugs, or procedures. Additional initial event criteria can also be added to refine the population further, such as the event occurring within a certain time frame.

After defining the initial event, the next step is to establish inclusion criteria. These criteria are based on a combination of domain-specific attributes to further refine and specify the cohort population, ensuring that it aligns closely with the research objectives. The inclusion

[1] https://github.com/OHDSI/Atlas

criteria can be based on a range of factors such as age limits, the presence of certain symptoms, or a specified duration of medication use.

Finally, defining the cohort exit criteria is crucial for determining when individuals no longer belong to the cohort. This stage is important for studies where the duration of membership in the cohort is relevant to the research question.

After defining the cohort, it is possible to generate an SQL code with the query to get the list of individuals who meet the criteria. ATLAS also facilitates the reuse of cohort definitions across different studies by allowing users to export and import cohort definitions in JSON format. This enhances the efficiency and reproducibility of research within the OHDSI network. A cohort definition can be seen as a query builder, a little different when compared to other general query builders. Figure 2.7 shows an example of a cohort definition in the ATLAS tool.

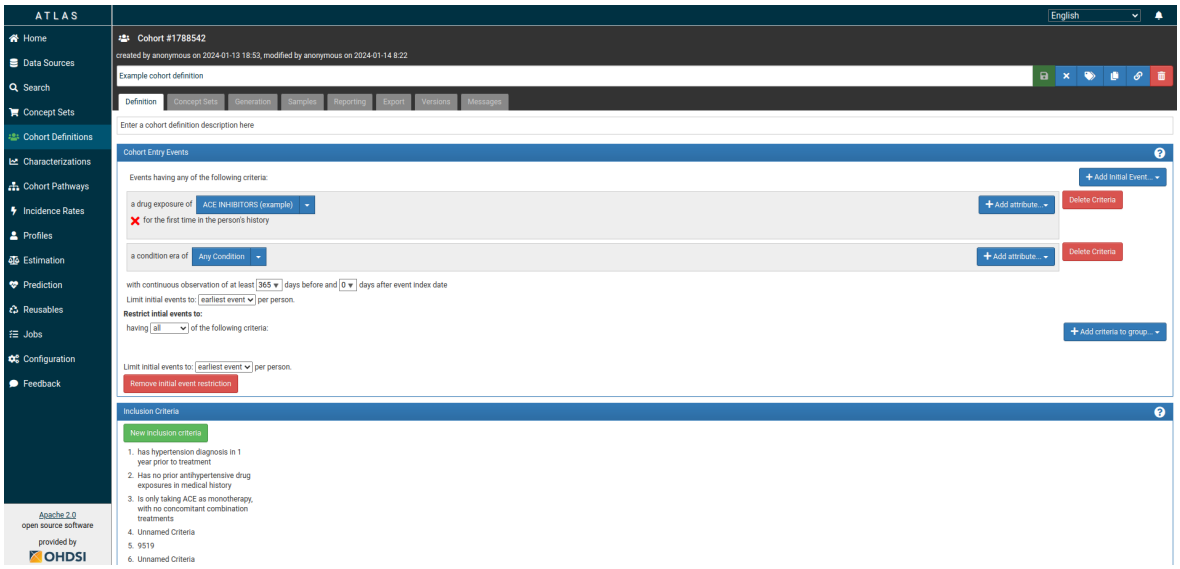

Figure 2.7: Example of a cohort definition in ATLAS.

2.5 Summary

The dissertation involves the development of a conversational query builder. This system should be a generative-based chatbot with a closed domain. Closed-domain chatbots are specialized in specific areas, offering precise responses. Generative chatbots use advanced LM to create dynamic responses closer to human interactions.

The use of LLM allows improvements in the NLG capabilities of chatbots and guides conversations more effectively, especially in the task of defining cohorts in medical research. LLaMa-2, Falcon and Mistral appear to be good options to implement since they are open-source and have the possibility of fine-tuning the model. Fine-tune has the role of optimizing chatbot performance, alongside Prompt engineering and RAG.

In terms of IR, in order to retrieve the most interesting databases according to the user's needs, the BM25 technique proves to be a good balance between effectiveness and efficiency.

BM25 is a straightforward algorithm that improves upon the traditional TF-IDF approach. Neural IR systems, like interaction-based models, show good results in the IR tasks, but are complex and the neural networks require substantial computational power. It is a lot simpler to implement than the Neural IR systems, as well as requires fewer computer resources.

OHDSI provides the software tool, ATLAS, to help researchers conduct observational studies. This tool has a feature to create study cohorts to define groups of people based on the research question. However, the interface revealed not very user-friendly and intuitive because it requires the user to have a good knowledge of its use and important concepts. Therefore, a chatbot that builds a cohort definition for this tool can improve the user experience by making it more intuitive and autonomous.

CHAPTER 3

Conversational Search Assistant

The EHDEN portal provides a catalogue of medical databases across Europe, offering researchers a centralized platform to explore available medical data sources. EHDEN built Network Dashboards tool to offer statistical and aggregated information about the databases available on the network. This tool helps researchers to choose the best databases across the catalogue. However, with the increasing number of databases in the catalogue, the process of find the most suitable databases becames difficult and time-consuming for a researcher.

This chapter focuses on the second goal of this dissertation: develop a chat-like search engine to help discover the best databases for a study. The following sections describes how the EHDEN Catalogue search assistant and its components were implemented and the decisions and steps made over time.

3.1 Data

The data utilized for this project comprises real EHDEN data sourced from the catalogue and the Network Dashboards. There are four main files that contain the data necessary to obtain an overview of the databases available in the EHDEN network: the countries file, the data sources file, the medical concepts file, and the Achilles Results file[1]. It is essential to understand the content of each file and the relationships between their data, as illustrated in Figure 3.1.

[1] https://github.com/EHDEN/CatalogueExport

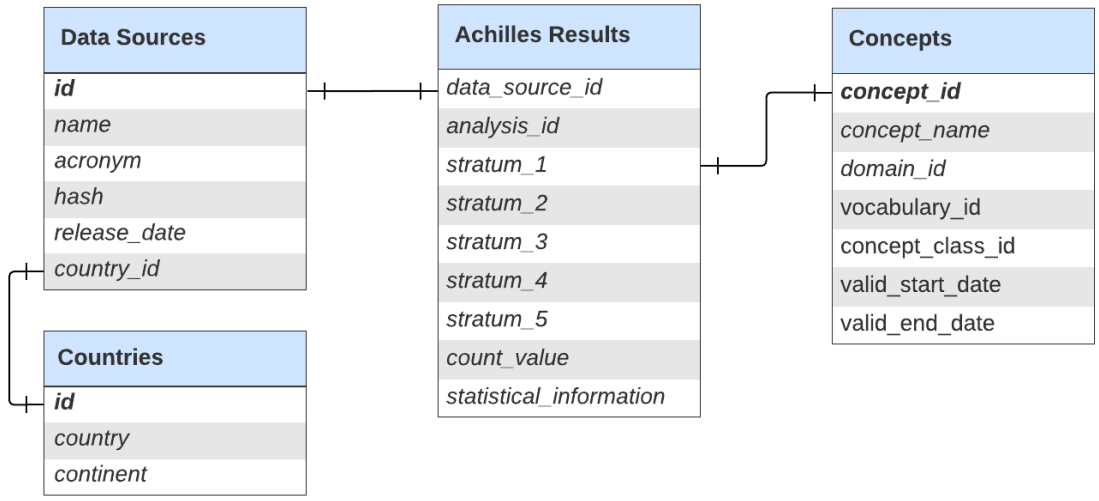

Figure 3.1: The diagram of the connection between the data files.

To comprehend the purpose of the four files, each is detailed with fields, description and an example of entries [19]. When the data is private and sensitive, not-real examples are included to better understand the content and connections between files.

Countries file (`countries.csv`).

- **Fields:** `id`, `country`, `continent`.
- **Description:** This file contains of real-world data, listing countries and their corresponding continents. It is used to provide geographical context for the analyzes, and supporting studies that require demographic segmentation.
- **Example Entries:**
  - `233, Ukraine, Europe`
  - `149, Montenegro, Europe`

Data sources file (`data_sources.csv`).

- **Fields:** `id`, `name`, `acronym`, `hash`, `release_date`, `country_id`.
- **Description:** This file includes essential details of the databases such as the source identification, the database name and the hash code of the EHDEN catalogue.
- **Example Entries:**
  - `1, MEDIBASE, Medical Database for Health Information Exchange, <hash>, NA,107`
  - `2, GRAVITAS, Global Repository of Advanced Vaccine Innovations Technologies and Strategies, <hash>, NA, 228`

Medical concepts file (`concepts.csv`).

- **Relevant fields:** `concept_id`, `concept_name`, `domain_id`, `vocabulary_id`.

- **Description:** The concepts file contains metadata on medical concepts, extracted from OHDSI Athena[2].
- **Example Entries:**
  - `2966436, Latanoprost (Apotex) 0.005% Eye Drops 2.5 Ml Bottle, Drug, AMT, Containered Pack, NA, 1009551000168106, 2016-11-01, 2099-12-31, NA`
  - `2966944, Letrozole (Apotex) 2.5 Mg Tablet, Drug, AMT, Trade Product Unit, NA, 1009571000168102, 2016-11-01, 2099-12-31, NA`

Database summaries (`achilles_results.csv`).

- **Relevant fields:** `data_source_id`, `analysis_id`, `stratum_n (from 1 to 5)`, `count_value`, `statistical information`.
- **Description:** Metadata summary of metadata present in the databases providing aggregated analytics. The "stratum_1" field points to the concept ID.
- **Example Entries:**
  - `143, 1, NA, 8507.0, NA, NA, NA,988296, NA, NA, NA, NA, NA, NA, NA, NA, NA`
  - `138, 2, 8532.0, 8532.0, NA, NA, NA, 1354216, 526.0, 52600.0, 26300.0, 52.6, 25774.0, 52.6, 157.79, 368.2, 499.7`

The combination of these four files gives the necessary information about the databases. Each row of the Achilles Results file contains statistical information about each concept available in the database, and a set of characteristics that are commonly used to filter databases in a cohort study. For instance, the number of samples segregated by range of age.

3.2 Information Retrieval

To to gain more knowledge about implementing IR methods and, at the same time, validate the IR results of this dissertation, I was involved in a challenge, BioASQ[3]. BioASQ addresses the information access problem for biomedical experts by organizing challenges in biomedical semantic indexing and QA. These challenges cover tasks such as hierarchical text classification, machine learning, IR, QA from texts and structured data, and multi-document summarization.

This section outlines the IR component, covering the BioASQ challenge and its relevance, and the implementation of BM25 for indexing and searching databases.

3.2.1 BioASQ challenge 2024

The BioASQ includes several challenges. My team was involved in "BioASQ Task 12b", focusing on biomedical semantic QA. This challenge aims to advance the development of systems capable of understanding and answering biomedical questions. Participants must

[2] https://athena.ohdsi.org/
[3] http://bioasq.org/

respond to test questions using various types of information, including relevant concepts, articles, and snippets. The challenge involves 5,000 training questions with gold-standard answers and introduces 500 new test questions, all constructed by European biomedical experts.

The challenge consists of two phases, A and B:

- **Phase A**: Participants respond to released questions with relevant articles and snippets.
- **Phase B**: Participants provide exact and ideal answers based on questions and provide relevant articles and snippets.

My task was implementing and testing some IR methods to determine which performs better. This task allowed me to have more concrete results. The techniques used and tested were BM25, SPLADE [72], and BGE-M3 [73]. As BM25 has already been explained in the section 2.1.1, here is a brief overview of the other methods tested.

SPLADE

SPLADE[4] is a neural retrieval model that learns query/document sparse expansion through the BERT Masked Language Model head and sparse regularization, according to Formal *et al.* [72] . This technique belongs to Learned Sparse Retrieval (LSR) because it combines elements of traditional sparse retrieval techniques with machine learning, particularly deep learning.

SPLADE is designed to balance the effectiveness of dense retrieval models with the interpretability and efficiency of sparse representations. It is advantageous because it combines the benefits of dense and sparse models [72]. This method can improve the relevance and accuracy of the search results, particularly in complex queries where understanding the context and the semantic relationships between terms is essential. The model used was `naver/splade-cocondenser-ensembledistil`, proposed by Formal *et al.* [74].

BGE-M3

Chen *et al.* [73] proposed the technique BGE-M3[5]. It is a versatile technique because it can perform three common retrieval functions: dense retrieval, multi-vector retrieval, and sparse retrieval. Also, this technique is multilingual because it supports over 100 languages and is multi-granular because it can process inputs ranging from short sentences to long documents, accommodating up to 8192 tokens. The model used was `BAAI/bge-m3` with 1024 of dimension and 8192 tokens of sequence length.

3.2.2 BM25 implementation

Although we have tested various methods, the implementation of BM25 has shown promising results. BM25 is the selected method for implementing the IR component in this project. The library implemented was PyTerrier[6], a Python framework for performing IR experiments.

[4] https://github.com/naver/splade
[5] https://github.com/FlagOpen/FlagEmbedding/tree/master/FlagEmbedding/BGE_M3
[6] https://github.com/terrier-org/pyterrier

Database indexing process

The IR component must index a collection of documents that thoroughly represent all the concepts within the databases, ensuring a clear understanding of what each database encompasses. To achive this, the data mentioned in the Section 3.1 is used to create the documents.

Figure 3.2 represents the different stages of the data. The first part of the figure, A), represents the three data files detailed in the Section 3.1. These are received from the EHDEN Network Dashboard. One contains information about the data sources, another contains the metadata summary of the databases (Achilles Results), and the last contains all medical concepts used in this community. These three files data was combined and readjusted to be indexed as documents. The remaining parts of this figure (B and C) represent the strategies adopted to create the documents of each database.

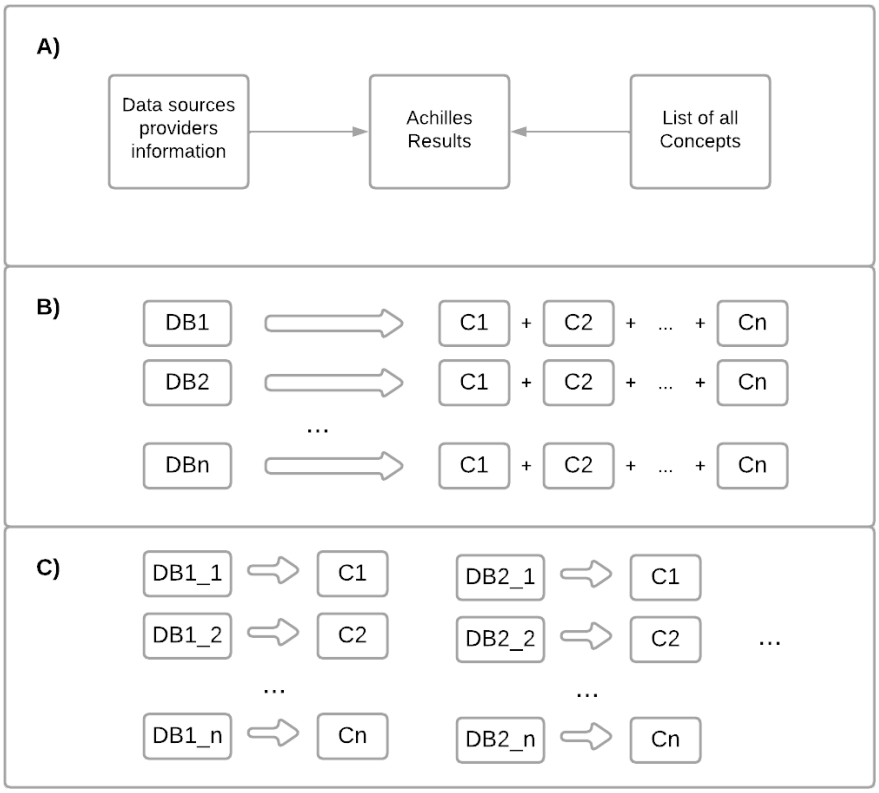

Figure 3.2: Index data structure: A) Metadata about the databases, B) First approach to the structure of indexed documents, C) Current approach to the structure of indexed documents.

The first approach to structure of the documents to be indexed is shown in Figure 3.2, part B). The document structure consists of a pairing of database IDs with their corresponding content. The content is represented as a single string, where each concept from each database is concatenated and separated by spaces. However, this way of indexing the database contents has several drawbacks:

- **Lack of Granularity**: By concatenating all concepts into a single string, the individual

relevance of each concept is lost. When a search query is made, the search engine treats the entire string as a single document, reducing search results' precision and recall.

- **Poor Search Results**: The search engine returns unsatisfactory results because the indexing does not accurately reflect the importance of each concept. This leads to low search scores, resulting in commonly retrieving irrelevant documents.

The approach to creating the document structure has been refined to enhance the granularity and searchability of the database content. Each concept in the database is individually indexed, as shown in Figure 3.2, part C). Each concept within a database is treated as a separate document for indexing purposes. This is achieved by assigning a document number that uniquely combines the database ID and the concept's position within its list. The document number serves as a unique identifier for each concept, ensuring that each piece of data can be individually retrieved and queried. The content of each document, represented by the concept itself, allows for reflecting the importance of concepts within the database, facilitating more precise and efficient retrieval of information.

Database searching process

When the IR system receives a query in free text, the system processes it using NLP techniques applied to the query to enhance the search performance. These processes can include tokenization (splitting the text into individual terms or words), stemming or lemmatization (reducing words to their base form), and stop-word removal (eliminating common words that do not contribute to the search). Standarding the text and transforming the query into important terms improves the match between the queries and indexed concepts as documents.

After processing the query, the BM25 algorithm evaluates the relevance of each concept within the database to the query, assigning a score that reflects its relevance. The method returns the 100 most relevant results and scores each concept. Then, these concepts are grouped by databases. The total relevance score for each database is calculated by summing the scores of its concepts. This comprehensive compilation process highlights the most relevant concepts. To ensure that only the most pertinent databases are presented to the user, the engine applies a predefined threshold to filter out databases with lower cumulative scores. The remaining results are then sorted in descending order of their total scores, prioritizing those with the highest relevance to the query.

Each element in the ranked list of databases contains some information about the databases. The ranked databases have a unique numeric identifier, the respective database name, and the hash that identifies a database in the EHDEN Catalogue. Also, it includes the total relevance score assigned to a database by the BM25 algorithm, indicating how well the database matches the query. At last, the concepts that has been identified are also included.

3.3 Large Language Model

The LLM used in this project is `Nous Hermes 2 Mixtral 8x7B` [75] open model, finetuned from Mixtral [76]. This model belongs to the llama family and has 47B of parameters. The

reason for using this LLM model is that it is a promising model. The LLM is installed on a local version of the Ollama[7] framework and deployed in a Virtual Machine to access the LLM using a URL. The model GPT cannot be used for this project, due to the use of sensitive data. Since GPT is not open-source, the data falls out of control. It raises privacy concerns for private data in public services.

3.3.1 Tasks

The LLM has two types of tasks: text generation (NLG) and text analysis (NLU) . Table 3.1 shows the different tasks given to the LLM and the respective descriptions. Also, it presents the output format and the temperature used in each task. The output format is the generated response format expected from LLM. The temperature of a LLM refers to a hyperparameter that controls the randomness of predictions made by the model. A higher temperature results in more creative and random outputs. Otherwise, a lower temperature results in more deterministic and focused outputs.

Type of Task	Task	Description
NLG	Generate response	Generate a response to a given message.
	Generate response with databases	Generate a response to retrieve the best databases to the user.
	Generate question	Generate a question related to a given question.
NLU	analyze topic	Identify if the user message is related to health. If so, identify the research topic in the field of health.
	analyze yes or no answer	Indentify if the answer is positive or negative for a respective question.
	analyze answer to question	indentify if the answer is a response for a respective question.

Table 3.1: Overview of the LLM tasks.

3.3.2 LLM configuration and implementation

After defining the tasks, Table 3.2 clarifies the parameters and the output format of each task.

Type of Task	Task	Output format	Temperature (0 - 1)
NLG	Generate response	Plain text	0.4
	Generate response with databases		0.4
	Generate question		0.4
NLU	analyze topic	JSON object	0.1
	analyze yes or no answer		0.1
	analyze answer to question		0.1

Table 3.2: LLM parameters and output information.

Inside the generation main task, the LLM can generate a response that may include the most suitable databases and formulate a question to ask the user. The temperature in this type of task is medium to provide a balance between randomness and determinism. The

[7] https://ollama.com/

output format is plain text because it is intended to send messages to the user. These tasks can be viewed as NLG tasks.

In the main analysis task, the language model should respond in a JSON object with a specific structure as specified in the prompt. To return a JSON response, the temperature is lowered in order to make the LLM more deterministic and avoid creativity. The response in JSON format is crucial for the system to comprehend the user's response. These tasks is related to the NLU tasks.

The model require specific prompts to generate appropriate responses. Each task assigned to the LLM has its own specific prompt and a corresponding temperature setting, which controls the randomness of the output, as we can see in the Code Snippet 3.1. These parameters — prompt and temperature — are defined in a JSON file.

```
{
  "url": "http://<LLM_URL>:<port>/api/generate",
  "model": "nous-hermes2-mixtral:latest",
  "system": "...",
  "generate_question": {
    "prompt": "Your goal is generate a question. You already chat with the user; you do not
↪  need to welcome him. Just ask the following question (you should rewrite the question):
↪  <question>. This question aims to extract more user research requirements. You should
↪  write a message with plain text, not in JSON.",
    "temperature": 0.4
  },
  "topic": {
    "prompt": "...",
    "temperature": 0.1
  },
  "yesno": {
      "prompt": "...",
      "temperature": 0.1
  },
  "is_answer_to_question": {
      "prompt": "...",
      "temperature": 0.1
  },
  "retriever": {
      "prompt": "...",
      "temperature": 0.4
  },
  "generate_answer": {
      "prompt": "...",
      "temperature": 0.4
  }
}
```

Code Snippet 3.1: The configuration JSON file of the LLM

The file serves as a configuration file, ensuring that the LLM receives the correct instructions and settings for each task. Thereby optimizing its performance and ensuring the generated responses meet the desired criteria.

The URL field contains the Ollama LLM endpoint to generate text. The model field indicates the LLM model used, and the system field refers to the system prompt. The system

prompt guides how these advanced AI models interpret and respond to user queries. It directs the LLM behavior and ensures that the generated outputs align with the intended goals.

The following fields are the tasks, with the respective prompt and temperature associated. The prompt of the question generation task has a "<question>" string, as the example shows, where the cohort question replaces it later. This prompt refinement is done in every task prompt.

3.4 Frameworks to streamline biomedical data discovery[8]

There are some options to integrate the LLM in this system. FlowiseAI and Langflow are frameworks that enable build an LLM system without worrying with the orchestration flow between components. An overview of these frameworks and, in the case of FlowiseAI, an implementation are presented below.

3.4.1 FlowiseAI

FlowiseAI[9], an open-source automation tool, plays a pivotal role by facilitating the integration of different AI components, combined with IR techniques, as Reis *et al.* [17] stated. FlowiseAI enables the creation of customized orchestration flows for LLM with AI agents and other tools. The workflows within FlowiseAI consist of interconnected nodes or blocks that represent various actions or operations. The specific workflow implemented is illustrated in Figure 3.3.

The conversational agent employs a comprehensive approach to enable dynamic interactions on a healthcare IR platform. It orchestrates the dialogue flow through the Conversational Agent Node, utilizing an LLM to provide a coherent and context-aware user experience. This node is configured with parameters that control tool access, chat model specifications, and memory capabilities, allowing it to maintain context or state information across interactions for structured conversations.

The core of conversational dynamics is powered by the ChatOllama Node, a chatbot engine designed for processing user queries and the generation of relevant responses. The implementation rely on Ollama since other solutions like ChatGPT API possess privacy issues. The Ollama operates based on a set of parameters including a base URL, alongside model specifications and a temperature setting that modulates probabilistic distribution over the predicted tokens. Furthermore, the Conversational Agent incorporates a Buffer Memory component, essential for the retention of interaction histories and stateful data. This component ensures the persistence of conversational context, a critical feature for enhancing user engagement and response relevance.

To provide the most relevant medical databases, a RAG architecture was adopted, as detailed in the section 2.3.2. This approach applied to this scenario involves retrieving the best databases from the IR component and adding them to the LLM prompt. RAG enables

[8]This section is mainly based in the publication *Using Flowise to Streamline Biomedical Data Discovery and Analysis, IEEE 22nd Mediterranean Electrotechnical Conference (MELECON), 2024*

[9]https://github.com/FlowiseAI/Flowise

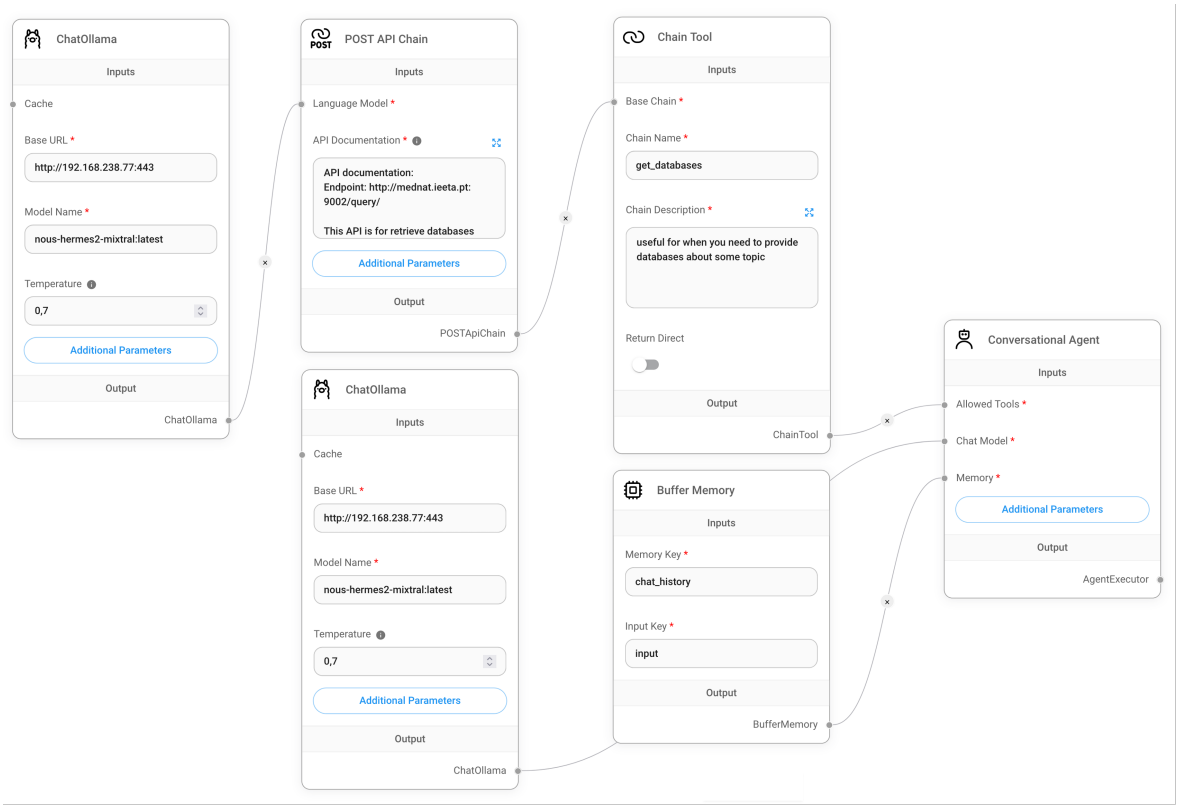

Figure 3.3: Worflow implemented in FlowiseAI tool.

the LLM to have up-to-date, valid, and domain-specific information to enhance response accuracy and relevance.

The integration with the Chain Tool facilitates the application of prompt engineering techniques, enabling the agent to access the list of the recommended best databases. The tool consumes the endpoint of the IR component, which is better described in the section 3.2.2. The conversational agent is equipped with real-time, accurate database recommendations, enhancing the quality of information provided to the user.

3.4.2 Langflow

Langflow[10] is another open-source tool to build AI applications. It is also a low-code tool that allows the integration of LLM and AI components. This tool simplifies the process of creating flows, such as chatflows. Users can drag components from the sidebar onto the canvas and connect them to begin building their applications. The platform allows for exploration by editing prompt parameters, grouping components into high-level components, and creating custom components. This intuitive interface makes Langflow a powerful tool for developing LLM-based applications.

An implementation was tested, but without much success due to some of the tool's limitations, such as the lack of basic functionalities like a conversational agent.

[10] https://github.com/langflow-ai/langflow

3.5 EHDEN CHATBOT ARCHITECTURE[11]

The system was designed to be integrated as a tool in the EHDEN Portal. Therefore, authentication issues were solved by the current mechanisms available on this platform [77]. Therefore, the system was implemented to use the MONTRA2-SDK [13]. This also provided the Network Dashboards tool, an interface to show the metadata, when researchers want to get more details about the suggested databases.

The previous implementation, detailed in the section 3.4.1, tried to adopt an open-source automation tool to build Chatbot applications, FlowiseAI [17]. However, these become limited to address the new requirements. Therefore, FlowiseAI was replaced by a Python-based backend, developed for this system. In addition to the IR function, the backend also orchestrates chat flow between the various components. Figure 3.4 represents the key components of the system and their interconnections.

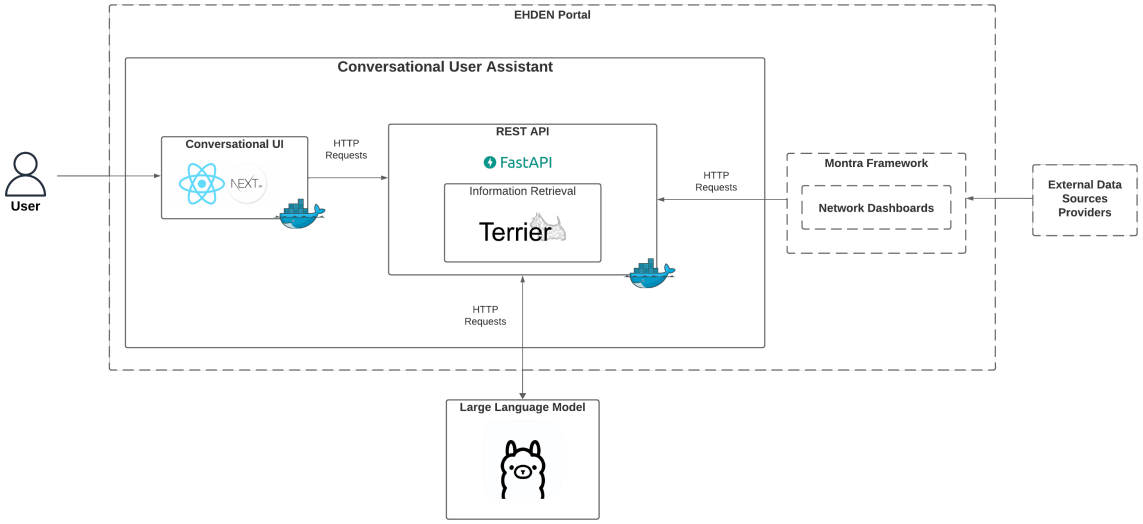

Figure 3.4: Overview of the EHDEN chatbot architecture.

The Conversational User Interface is the primary interface for user interaction, built on the React framework. The User Interface records the user questions and conveys them to the backend to be processed in the component dedicated to IR.

The backend API, built on the FastAPI framework, handles the HTTP requests from the other components. When it receives a query from the User Interface, the backend communicates with the LLM. The LLM is the same open model detailed in the Section 3.3, `Nous Hermes 2 Mixtral 8x7B`. In this implementation, the LLM has two tasks: analyze the topic of health and generation of a response with databases or generate a simple response. If the first task returns false, *i.e.*, if the user message is not related to health, the LLM generates a simple response reminding the user of the chatbot's purpose. Otherwise, if the first task is true, then the RAG architecture is applied. This means that the generation of a response

[11]This section is mainly based in the publication *A chatbot-like platform to enhance the discovery of OMOP CDM databases, 34th Medical Informatics Europe Conference (MIE), 2024*

with databases task is applied. The BM25 retrieves the best databases to the LLM, so it could generate a response with that valid information.

For research purposes, researchers may save their conversations to facilitate data analysis and enhance their research outcomes. When a user decides to save a conversation, he can do so by pressing a save button, which triggers an API endpoint designed to store the conversation. The conversations are subsequently added to a JSON file, creating a structured and accessible format for future analysis. This method ensures that valuable data is systematically archived and easily retrievable for subsequent research endeavors.

Storing these dialogues can be particularly valuable for research purposes, for example, identify frequently asked questions by medical researchers. By analyzing the saved conversations, researchers can gain insights into common concerns, informational gaps, and the types of support often sought by medical researchers. By taking advantage of this stored information, the system can be improved, or even develop other platforms to improve the overall quality and impact of medical research.

CHAPTER 4

Query Builder

OHDSI provides diverse open-source tools to support various use cases on observational patient-level data. Among these tools is ATLAS, a software tool designed to help researchers conduct observational studies. ATLAS includes a feature for creating study cohorts, allowing researchers to define groups of people based on their specific research questions.

This chapter is dedicated to the third goal of the system: enhancing the conversation search assistant to support the cohort definition for an observational study. The conversational query builder should assist researchers in defining their cohorts within ATLAS. The following section discusses the implementation of the query builder, as well as the decisions and strategies involved.

4.1 Implementation strategy

The definition of a cohort requires the definition of a concept set. The strategy of the cohort construction is focused first on creating the concept set and then on the remaining of the cohort's parameters.

4.1.1 Interaction between components

The conversational query builder involves a sequence of steps and interactions to achieve the desired outcome. Figure 4.1 describes all the interactions between components and the user. The system components are the IR component, which is accessible via an API endpoint, and the chatbot user interface component, which also represent the LLM component. The EHDEN Portal and ATLAS are external tools.

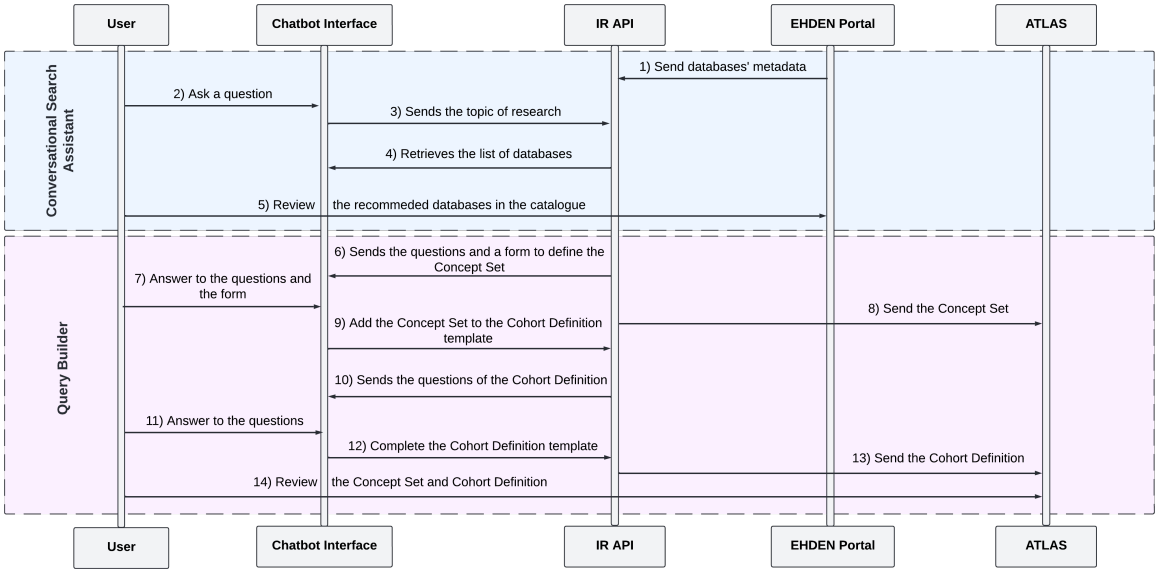

Figure 4.1: Interaction diagram between the user, the system components and external tools.

This diagram illustrates the interactions outlined in the previous Chapter 3 from steps 1 to 5. Additionally, it depicts all the interactions of the conversational query builder, steps 6 to 14, as explained in this chapter.

Reminding the interactions explained in Chapter 3, the IR component receives the databases' metadata from the EHDEN Portal (step 1), creating and indexing the documents. In step 2, the user asks a question to the Chatbot Interface. The LLM applies his task of identifying the research topic and sends it to the IR API (step 3). The IR API retrieves the most suitable databases for the research concept and sends the list of databases to the Chatbot Interface (step 4). The user can review the recommended databases in the catalogue in the EHDEN Portal, as the last step of this stage.

The query builder phase starts. As mentioned before, the cohort's construction requires a concept set. So, the IR API, when sending the databases to the chatbot interface, also sends questions and then a form to help define the concept set (step 6). The user answers the question and fills out the form in order to create the concept set that could be sent to the ATLAS instance. This describes the steps 7 and 8 of the Figure 4.1.

The rest of the cohort is determined now that the concept set is defined. In step 9, the concept set is added to the cohort definition template. The IR API sends a question to complete a field needed for the cohort definition. After the user answers to it, the LLM, in the Chatbot Interface, analyzes if the user message is an answer to the question. If so, the answer is saved in the cohort template. These are steps 10, 11, and 12, which are repeated until the cohort template is complete. Finally, the API sends the cohort definition to the ATLAS instance if the user requests it, where they can review it in ATLAS. This describes the final steps, 13 and 14.

4.1.2 Template and questions

The ATLAS platform accepts a cohort definition in a JSON format. Two crucial JSON files are needed to generate this JSON object, personalized with the user's requirements information. One is the template for users to fill out during the conversation, shown in Code Snippet 4.1. This template is an empty cohort definition JSON structure. The other file, Code Snippet 4.2, contains the questions associated with each key field of the template. Each question should be simple and efficient so the medical researcher can respond to it, and its response is the value of that key. Also, each question in the template is manually inserted in the JSON file.

```
{
  "ConceptSets": null,
  "PrimaryCriteria": {
      "CriteriaList": [
          {
              "ConditionOccurrence": {
                  "CodesetId": 0,
                  "ConditionTypeExclude": null,
                  "ConditionSourceConcept": null,
                  "First": null
              }
          }
      ],
      "ObservationWindow": {
          "PriorDays": 0,
          "PostDays": 0
      },
      "PrimaryCriteriaLimit": {
          "Type": "First"
      }
  },
  "QualifiedLimit": {
      "Type": "First"
  },
  "ExpressionLimit": {
      "Type": "First"
  },
  "InclusionRules": [],
  "CensoringCriteria": [],
  "CollapseSettings": {
      "CollapseType": "ERA",
      "EraPad": 0
  },
  "CensorWindow": {}
}
```

Code Snippet 4.1: The cohort template JSON file.

```
    {
      "ConceptSets": "Are there any other concepts you'd like to add? If yes, please add them.
↪  If no, simply respond with 'no'.",

      "PrimaryCriteria": {
        "ObservationWindow": {
          "PriorDays": "Regarding the observation window, what is the minimum number of days
↪  needed before the continuous observation? You must choose from 0, 1, 7, 14, 21, 30, 60,
↪  90, 120, 180, 365, 548, 730 or 1095.",
          "PostDays": "How many days after event index date? You must choose from 0, 1, 7,
↪  14, 21, 30, 60, 90, 120, 180, 365, 548, 730 or 1095."
        }
      }
    }
```

Code Snippet 4.2: The cohort questions file.

To a better understanding of these files utility, it is given an example using the Code Snippets 4.1 and 4.2. Assuming that the concept set is already defined, the next question in the file is the one associated with the "*PriorDays*" key. The question is asked to the user. When the user answers, the LLM applies a NLU task to analyze if the user's answer is a response to the question posed. If the user's answer is related to the question, the LLM extracts the information to fill the cohort template on the key "*PriorDays*". Otherwise, the same question is repeatedly asked until the user responds to that question.

4.1.3 Template pointer

The cohort questions file has the questions to complete the cohort template file. There is a pointer that tracks which questions have been answered and which ones still need a response. The template pointer sequentially follows the keys in the questions file. The pointer retrieves a JSON key from the questions file (Code Snippet 4.2). This key points to a question that needs to be asked to the user. When the user answers this question, it is easy to fill the template file (Code Snippet 4.1), because the pointer indicates which key of the template corresponds to the answer. When this occurs, the pointer moves to the next key in the questions file. If the user has not answered the associated question yet, the key stays in the same position.

During the conversation, the pointer helps:

- To determine the appropriate question to ask the user.
- To identify which key of the cohort template to use to save the user's answer.
- To move on to the next question of the cohort template.

This solution keeps track of the user's responses and dynamically updates the template with the relevant information throughout the conversation.

4.2 Concepts sets

In Code Snippet 4.1, a cohort is defined by multiple JSON fields. One of these fields is the concepts set (the cohort template key is "*ConceptSets*"), which is a list of concepts required

to meet the study requirements of the researcher. In order to properly define a cohort, the concept set must be established first. Therefore, our strategy is to define the concept set in the conversation before moving on to the remaining fields of the cohort template.

4.2.1 Expression

A concept set expression is comprised of a list of concepts with the following attributes:

- **concept**: Definition of a concept, using data contained in the concepts file (`concepts.csv` as described on Section 3.1).
- **isExcluded**: Exclude the concept from the concept set.
- **includeDescendants**: Add all of descendants of a concept, with it also included.
- **includeMapped**: Allow the search for non-standard concepts.

The JSON Code Snippet 4.3 represents a concept definition within a concept set expression. This example is not real information; it is just to define a concept set visually.

```
{
  "id": 0,
  "name": "<CONCEPT SET NAME>",
  "expression": {
      "items": [
          {
              "concept": {
                  "CONCEPT_ID": 231256,
                  "CONCEPT_NAME": "Covid-19",
                  "DOMAIN_ID": "Condition",
                  "VOCABULARY_ID": "ASG67HR",
                  "CONCEPT_CLASS_ID": "ASG67 code",
                  "CONCEPT_CODE": "953635",
                  "VALID_START_DATE": "2000-01-01",
                  "VALID_END_DATE": "2099-12-31"
              },
              "isExcluded": false,
              "includeDescendants": true,
              "includeMapped": false
          },

          (...)
      ]
  }
}
```

Code Snippet 4.3: Concept set expression example.

4.2.2 Concepts sets creation

The goal is to create a JSON object identical to the one in the Code Snippet 4.3 with the concepts selected by the medical researcher. When a user enters a query, the conversational search assistant retrieves the best databases related to the health topic of that query. It should also inquire whether the user has additional concepts for their study requirements. In an implementation vision, the first key to complete in the cohort template is "*ConceptsSets*", so the question regarding that key is rewritten by the LLM and then asked to the user.

The user should respond to whether there are additional concepts to add. If there are, he should indicate them. If not, he should provide a negative answer. The chatbot continues to ask if there are more concepts until the user confirms that there are no more to add.

In the IR component, the BM25 technique searches in the collection for the concepts the user indicates in his responses. After identifying each concept in the collection using the IR technique search, additional information from the concept file is added. This includes the concept ID, concept code, and other attributes that define the concept as represented in Code Snippet 4.3.

The list of concepts identified is sent to the chatbot interface. The chatbot interface shows the concepts in a form. For example, Figure 4.2 shows a form to create the concept set after the user mentions his interest in COVID-19. The user gives a name to the concept set and selects the concepts and the other attributes.

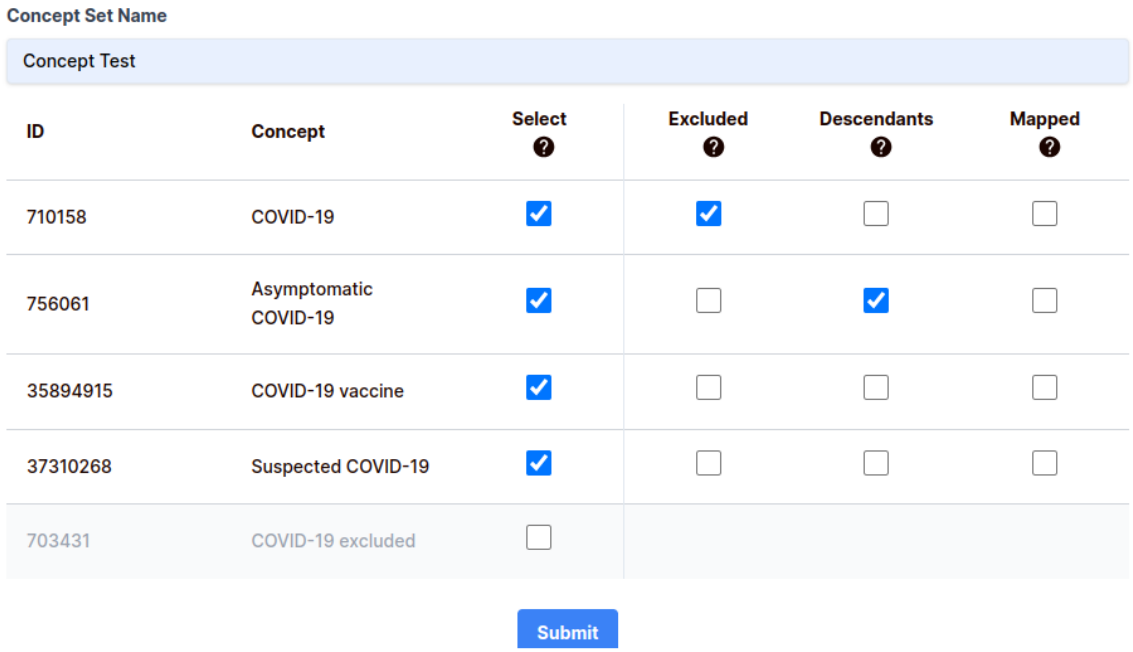

Figure 4.2: The form to create the Concept Set.

Finally, the user submits the forms, creating the concept set. The chatbot sends a message informing the user that they can download the concept set JSON file, which is accepted in the ATLAS instance, to create a new concept set. Alternatively, the chatbot can send the file directly to the ATLAS instance. The pointer of the cohort template moves to the next key.

4.3 Cohort definition

Once the Concept Set is defined, the list of concepts and their attributes are saved on the template, where the template key points to. Then, the template pointer moves to the next template key. The LLM generates a specific question related to the question that the current key points to. This generated question is then presented to the user through the interface.

The user provides a response to the question presented. The user message could be a possible response to the question or, for example, can be a random message. The LLM checks if the user's message is indeed an answer to the question. This task is mentioned in the Section 3.3. If the response is valid, the LLM extracts the value and the template is updated with that value. The template pointer advances to the next key, where the next question is generated by the LLM. Otherwise, if the response is invalid or unrelated, the pointer remains on the same key because the question is not answered yet.

This cycle of getting the question of the pointer, generating the question, presenting it, receiving a response, and verifying the response repeats until the entire cohort is complete, and all keys in the template are filled with the appropriate user responses.

At the end, users can download the cohort definition by clicking on a button sent in a message. The cohort definition is in the right structure for import into the ATLAS platform. Alternatively, the chatbot can send the Cohort Defintion directly to the ATLAS instance.

4.4 Integration with OHDSI ATLAS

The OHDSI community provides a WebAPI[1] that contains all OHDSI RESTful services that can be called from OHDSI applications. This API has many features, such as providing a centralized API for working with databases converted to the OMOP CDM, searching the OMOP CDM standardized vocabularies for medical concepts and constructing concept sets. Also, it allows defining cohort definitions for use in identifying patient populations.

4.4.1 Communication with ATLAS webAPI

When the user interacts with the tool and finalizes the definition of a concept set or cohort, the IR API facilitates seamless integration with the OHDSI WebAPI by consuming specific endpoints designed for these tasks.

Creating a concept set

Initially, for the creation of a concept set, the necessary structure for the body of the OHDSI WebAPI endpoint is prepared, as shown in the Code Snippet 4.4. This process begins by making a POST request[2] to the OHDSI WebAPI. This POST request is responsible for creating the concept set, although at this stage it is devoid of any concept data. The response from this request includes the ID of the newly created concept set within ATLAS.

```
{
  "id": 0,
  "name": "<Concept Set Name>",
  "description": "Hippocrates costum concept set"
}
```

Code Snippet 4.4: The body to create the concept set in ATLAS.

[1] https://github.com/OHDSI/WebAPI
[2] https://api.ohdsi.org/WebAPI/conceptset/

Subsequently, the concepts that were defined throughout the user's interaction are sent to the PUT endpoint[3]. This PUT request populates the previously created concept set with the specified concepts, completing the definition process.

Defining a cohort

Following the concept set creation, the process of defining a cohort involves preparing the necessary structure for the body of the POST endpoint[4]. The body structure for this request is as shown in the Code Snippet 4.5.

```
{
  "id": 0,
  "name": "[Hippocrates] Cohort {current_time}",
  "expressionType": "SIMPLE_EXPRESSION",
  "description": "Hippocrates custom cohort definition",
  "expression": "<Cohort Definition>"
}
```

Code Snippet 4.5: The body to create the cohort definition in ATLAS.

The "*expression*" field contains the cohort definition that was constructed during the user's interaction. This POST request sends the cohort definition to the OHDSI WebAPI, where it is created and stored in the ATLAS platform.

User interaction and submission

Upon completion of either the concept set or cohort definition, the user is presented with an option to either download the JSON object, which can be imported into ATLAS manually, or to send it directly to ATLAS through the chatbot interface. The chatbot provides a message with two buttons, each corresponding to the creation of a new Concept Set or Cohort Definition on the OHDSI platform. By clicking the appropriate button, the defined concept set or cohort is automatically submitted to ATLAS, ensuring a streamlined and efficient workflow for the researcher.

4.4.2 Network interactions

The interaction between various components and institutions is crucial for the effective functioning of the query builder and the communication with the ATLAS platform. Figure 4.3 illustrates these interactions in detail.

[3] https://api.ohdsi.org/WebAPI/conceptset/<ConceptSetID>/items
[4] https://api.ohdsi.org/WebAPI/cohortdefinition/

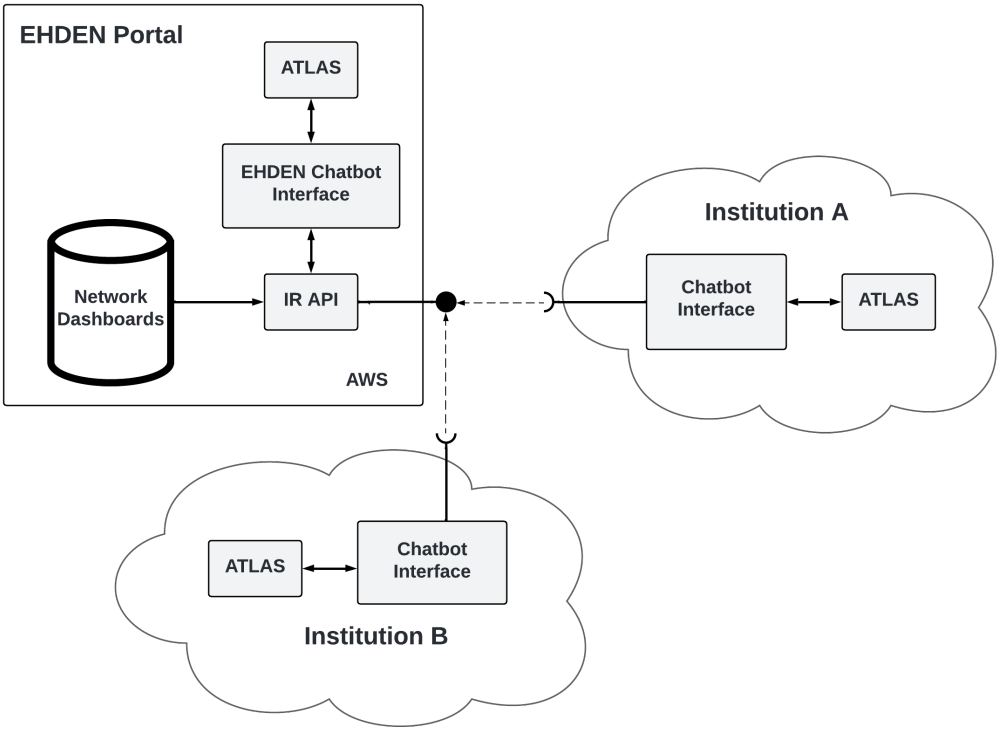

Figure 4.3: Network diagram illustrating the connections between the system and other institutions.

The EHDEN Portal serves as the central hub, hosting the EHDEN Chatbot Interface. Within this portal, the Network Dashboards provide metadata about various databases, which are crucial for the initial indexing and subsequent query building.

The EHDEN Chatbot Interface interacts directly with the user, guiding them through the process of defining concept sets and cohorts. It communicates with the IR API to retrieve relevant questions and forms, which are then presented to the user for completion.

The IR API acts as a mediator between the user inputs received via the Chatbot Interface and the ATLAS platform. It processes user queries, retrieves relevant database metadata, and prepares the necessary JSON structures for Cohort and Concept Set definitions.

Multiple institutions, with its own Chatbot Interface, can consume the IR API of the EHDEN Portal. Institution A and Institution B represent different institutions that use the EHDEN infrastructure for their research needs. The chatbot interfaces of the institutions communicate with their respective ATLAS instances and the EHDEN Portal, ensuring data consistency and integration.

Institutions A and B, through their Chatbot Interfaces, interact with the IR API and ATLAS to perform similar tasks. Each institution can define its cohorts and concept sets, leveraging the centralized capabilities of the EHDEN infrastructure while maintaining individual ATLAS instances for specific research activities.

CHAPTER 5

Results and Discussion

The current methods applied to help medical researchers discover databases of interest, specifically on the EHDEN Portal, are based on graphical and tabular views. These methods also have shown good results when dealing with a low number of databases, or concepts. However, with the increase of the databases in the network, such techniques may not be the best option.

The increasing number of databases in the EHDEN Portal leads to a comprehensive database catalogue with information from almost 200 databases in Europe. It spans across 28 countries and offers data concerning 256 million patients, derived from 34 distinct data sources [18].

The databases in the EHDEN Catalogue are supported by 38 unique attributes focused on exposing the core characteristics regarding data access. Additionally, the EHDEN Network Dashboards, it is hosted a staggering 8,504,324 unique concepts to accurately represent the scope of each database. These concepts are organized into 25 domains, including but not limited to Measurement, Condition, Procedure, Observation, and Drug. This amount of data raised unseen challenges that this project tried to address with this new approach.

The presented system aims to optimize the process of identifying and selecting relevant observational databases for medical research. The tool simplifies the discovery of medical observational databases in the EHDEN network, addressing the challenges of navigating through the vast and varied catalogues of medical databases. It also tackles the challenge of defining a cohort study in the ATLAS platform by improving the tool to allow for a more conversational approach to defining and providing a cohort query as an outcome. This section details the results about the IR component, the LLM implementation and the conversational query builder.

5.1 Information Retrieval solution

My participation in the BioASQ challenge was an opportunity to address the absence of an annotated dataset to validate the results of this study. Nevertheless, it is crucial to note

that BioASQ results are not just about IR but the whole task, detailed in the Section 3.2.1. Although the results of the whole task were not enough to validate the IR results, BioAsq was important for gaining knowledge and experience in implementing the method.

With no annotated datasets available, the BM25 validation is done manually, using a question set inspired by OHDSI's Book [78]. This manual process consists in select a disease, drug, or procedure, and do the same manual process of a researcher who is part of EHDEN. In simple terms, this process involves accessing the Network Dashboards tool of a specific database and going to the Concept Browser tab, where it is possible to search the concepts contained in that database. For each suggested database and its corresponding concepts identified by the search, this explained process is carried out to compare the concepts found on the Concept Browser with the concepts identified for each suggested database. The Figure 5.1 illustrates this process, where a search in a specific database is conducted using the term "covid".

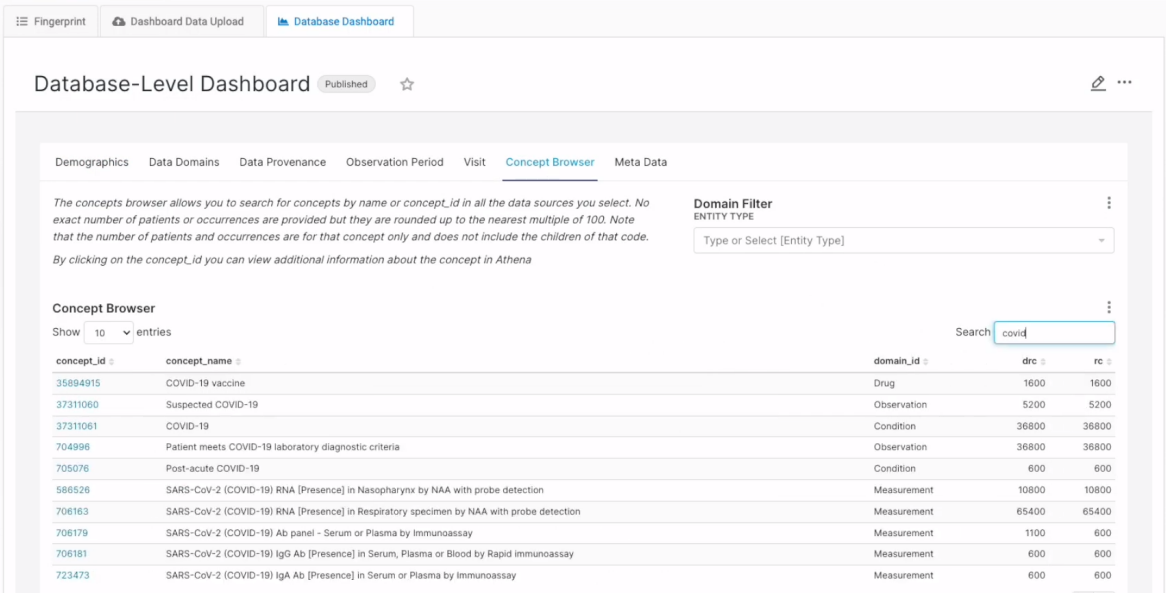

Figure 5.1: Manual process to validate the IR results.

As this comparison showed promising results, the BM25 method became the method of choice. It is recognized that this process is not the most valuable to validate the results. Since no metrics are available for validation, alignment with workflow in EHDEN formed the basis of our validation approach. However, annotating the dataset may enable the identification of more powerful IR models in the future.

5.2 LLM implementation: FlowiseAI vs Langflow

Langflow and FlowiseAI are similar tools that provide multiple tools that allow the building of different workflows, architectures, and integrations with external tools. The table 5.1 shows a comparison between these two frameworks, identifying the strengths and weaknesses of each one. For this implementation, integration with Ollama is necessary as the chosen LLM is installed in a local version of the Ollama framework.

FlowiseAI can run in air-gapped environments with local LLM, embeddings and vector databases. Also, it creates autonomous agents that can use tools to execute different tasks, such as Custom Tools, OpenAI Assistant, and Function Agent. FlowiseAI has proved to be a great solution to build an LLM-based chatbot. However, the FlowiseAI implementation become limited to address the requirements of the query builder. It became difficult to control the logic and the flow required by the query builder requirements.

Langflow is a dynamic graph where each node is an executable unit, so the way of development of a LLM application is very identical to FlowiseAI. But also, Langflow shown to be limited for this project because of missing basic features.

	FlowiseAI (version 1.6.0)	**Langflow** (version 0.6.10)
Strengths	Customization	Flexible customization
	Ollama integration	Better development environment
	Visually intuitive	Visually intuitive
Weaknesses	Limited development environment	Missing basic features
	Limited documentation	Outdated documentation
	Limited features	Ollama integration missing

Table 5.1: Strengths and weaknesses comparison between FlowiseAI and Langflow.

Starting with FlowiseAI, it is a great tool to build simple LLM applications, but have some limitations:

- **Limited features** - There are tasks that require more costumization, and the costum tool that FlowiseAI provides is not enough for some of these tasks.
- **Limited documentation** - There are some tools that not have documentation, and it became difficult sometimes to explore a tool without information about it.
- **Limited development environment** - It is difficult to debug. When something in your system goes wrong, it is difficult to find what is causing that error.

Otherwise, at the time of writing, Langflow offers features that either improve upon or address certain limitations of FlowiseAI, such as better features for development environment and more customization; for example, creating a complete custom component is possible. However, it also addresses some limitations:

- **Missing basic features** - Lacked basic functionalities such as a conversational agent.
- **Outdated documentation** - There was documentation of tools that did not exist, and there was not documentation of some that did exist.
- **Ollama integration missing** - For this project, it is important to have an integration with Ollama, because the LLM model is installed a local version of the Ollama framework.

Although these frameworks prove to be good options, they have limitations that do not satisfy this project's complexity. So, the LLM orchestration flow was implemented using a Python-based backend, developed for this project system.

5.3 Conversational search assistant

The integration of IR techniques within a conversational user interface represents a significant advancement in the field of data discovery [79]. Its ability to return a list of databases that are pertinent to the user's query not only saves time but also introduces a level of precision in the selection process that was previously unattainable through manual methods. The integration of generative AI helps the conversation be more human-like.

Researcher can establish questions like "We want to conduct a characterization study with main research topic about Covid-19.", and the tool responds with the most relevant databases that may have the information to answer such questions. Then, the research can analyze this information in more detail, or refine the question. Figure 5.2 shows the evolution of the data discovery techniques, starting with a database catalog, which evolved to more graphical representations until the proposed chatbot-like search engine. Also, Figure 5.2 shows how this engine can streamline the process of identifying the most suitable observational databases for medical research, with a simple interaction.

Therefore, the method of simplifying the discovery of observational databases has an impact on the speed and quality of research efforts. Also, the tool has the potential to level the playing field by providing less experienced individuals with access to complex databases that they might otherwise overlook or find too challenging to navigate. However, the interaction with the proposed tool is somewhat restricted due to the data content. Users may expect to have deep insights into the databases, but that may expose sensitive data. Future enhancements could include the integration of such information available only to users with access to them, which will include the use of Rule-Based Access Control (RBAC) mechanisms over this tool.

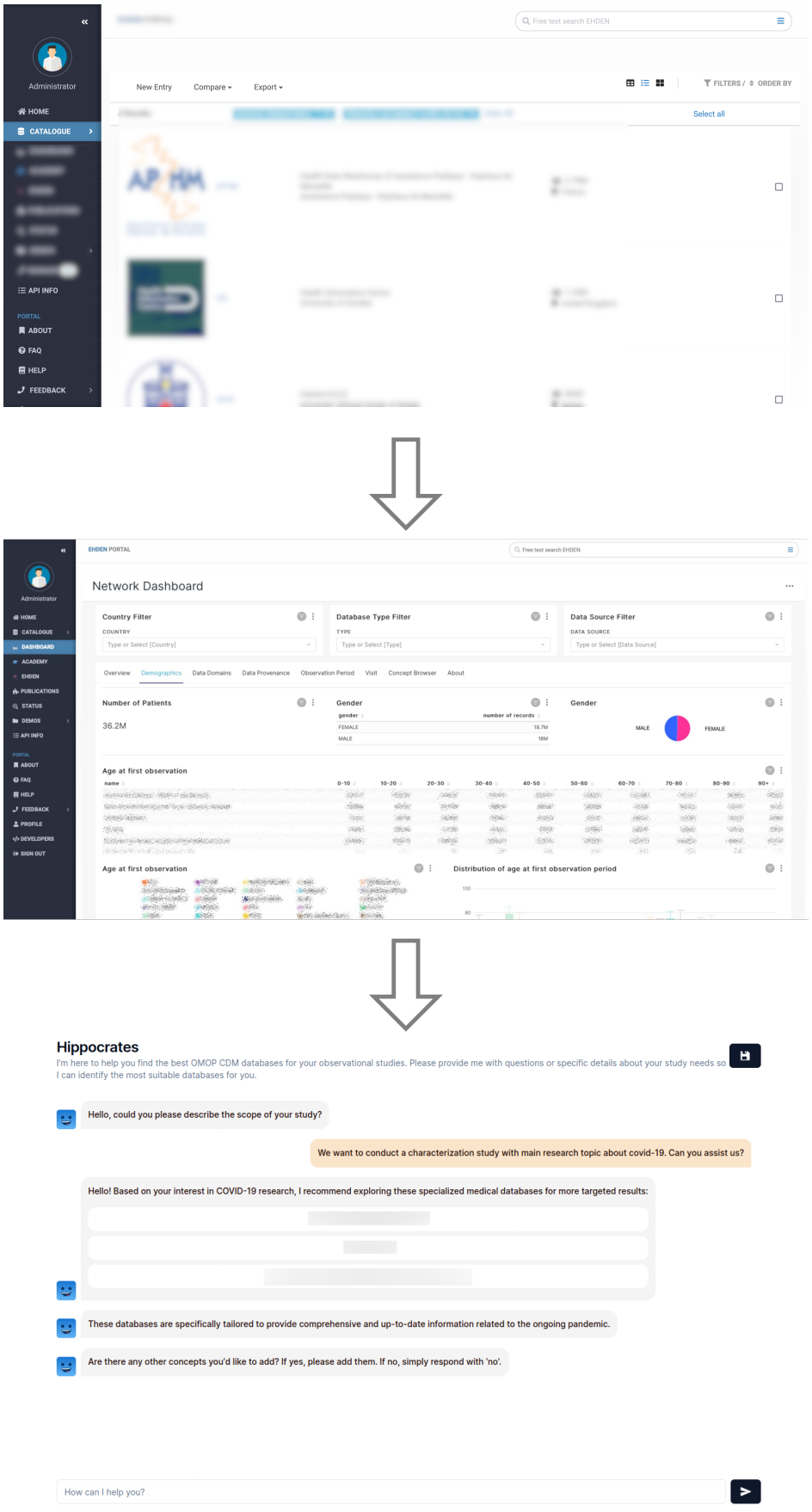

Figure 5.2: Evolution of the data discovery techniques, starting with a database catalogue, evolved to more graphical representations until the proposed chatbot-like search engine.

5.4 Query builder

The query builder component of the system is an essential part developed to simplify the process of making cohort definitions on ATLAS platform. The present version allows users to specify concept sets and observation windows for a cohort. Concept set, on the other hand, is what defines study group's overall health outcomes such as relevant medical conditions, procedures and medications while observation window means time frame in which patients' data will be observed. The following figures, Figure 5.3 and Figure 5.4, shows the continuation of the conversation in Figure 5.2, where the concept set and cohort are defined, respectively.

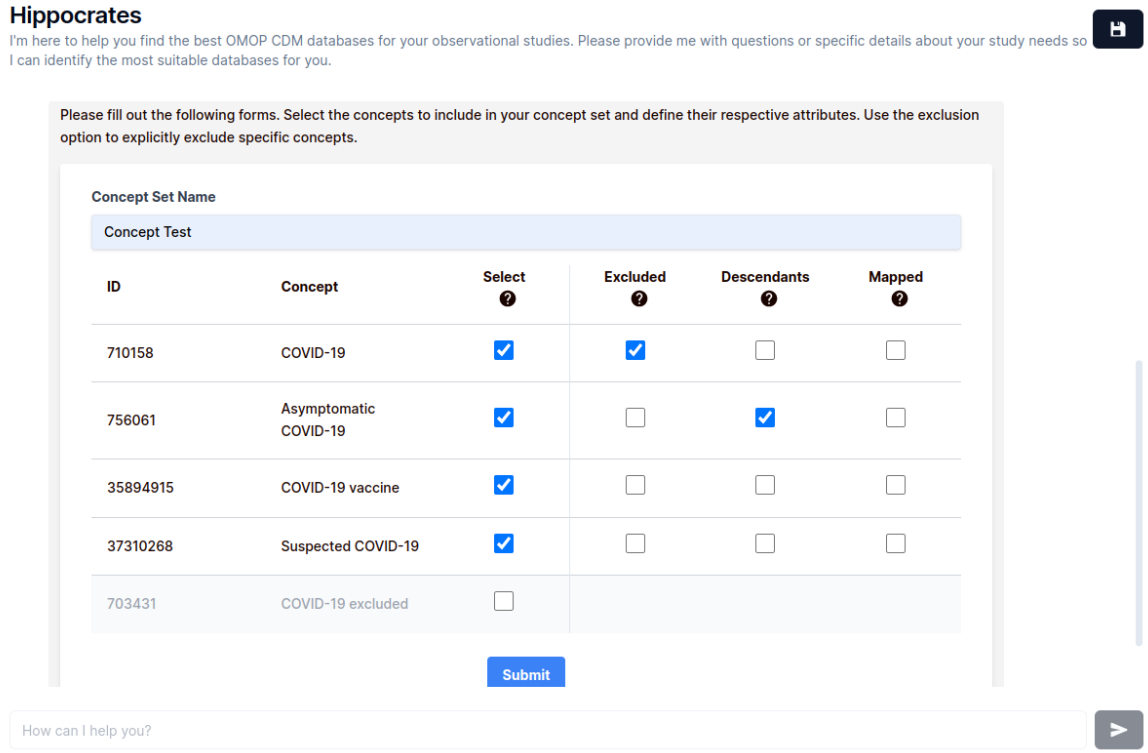

Figure 5.3: Continuing the conversation to define the concept set.

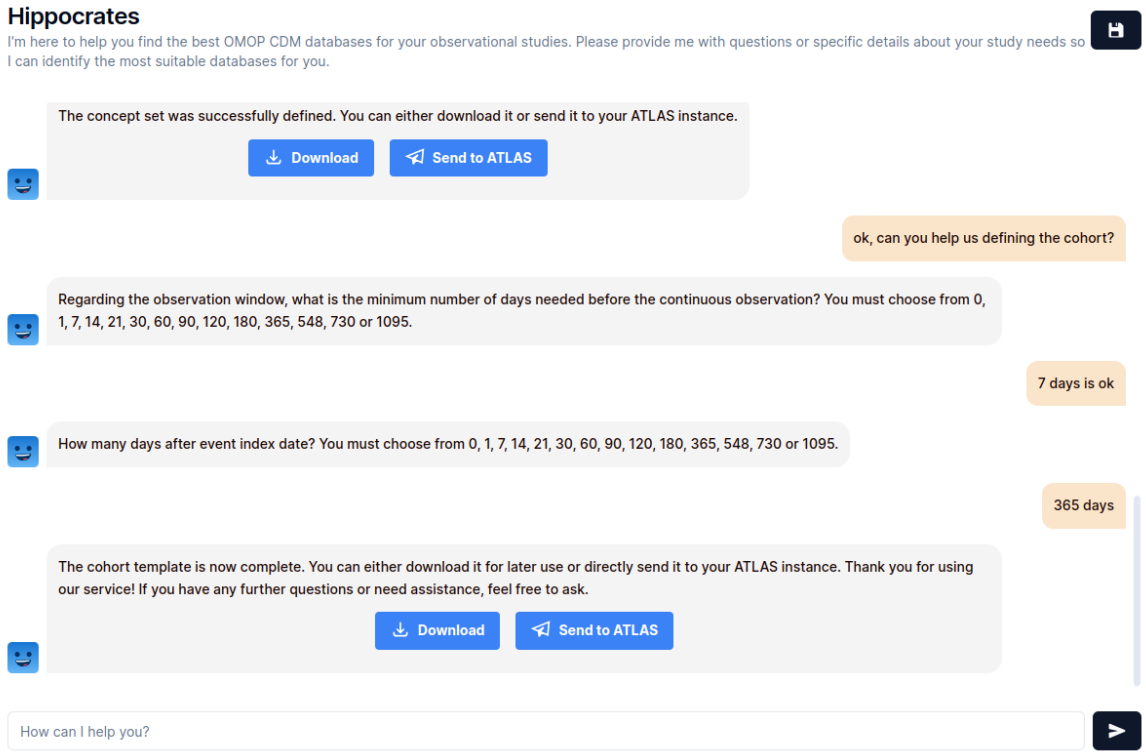

Figure 5.4: Continuing the conversation to define the cohort.

One of the significant developments made on the query builder is its ability to send defined cohort directly to an instance of ATLAS platform. Being able to send these definitions directly to the ATLAS instance points out considerable improvements towards efficiency as well as time-saving for medical researchers. This shortcut process enhances research quality, reduces errors, and accelerates its pace, facilitating data manipulation for medical researchers dealing with complex observational data.

Full cohort definition procedure includes additional parameters like inclusion/exclusion criteria, cohort exit criteria, and cohort attributes that were not fully covered so as not to surpass project's timeline. Some improvements would center on integrating these attributes, enabling the system to better assist researchers in defining their cohorts in their entirety.

CHAPTER 6

Conclusions

This dissertation proposes developing a conversational system to assist medical researchers in defining their study objectives. Answering the research question, this dissertation provides a personalized and efficient method for biomedical databases discovery and definition of a study protocol. Researchers can interact with the system conversationally, outlining their requirements and receiving customized recommendations.

The research question guided to accomplish three objectives. The literature review on conversational interfaces, NLP, and IR was crucial to develop a chat-like search engine to help discover the best databases for a study and enhance this engine to collect additional information to provide a query as an outcome. The successful implementation of these objectives not only simplifies the process of identifying the most relevant databases, but also helps in the definition of study protocols process, empowering researchers, regardless of their prior experience with database search tools and query builders.

Regarding implementation, it is possible to extract some conclusions about the choices, strategy, and areas. About BM25, it shows promising results, but creating an annotated dataset may enable us to identify how to improve these results or other powerful IR techniques. To add the LLM into the system, FlowiseAI shows to be an excellent tool for the development of LLM-based applications. However, it is limited due to this project's complexity, specifically the query builder phase. So, the option to integrate the LLM is a Python-based backend developed for this project.

In conclusion, the introduction of this conversational search tool addresses the significant challenge of navigating a growing repository of medical databases. This system enables users to engage in natural language dialogue, significantly reducing the time and complexity traditionally associated with identifying suitable databases for research studies. Moreover, the system reflects a deep understanding of the needs within the medical research community for reliable, up-to-date, and accessible data.

Also, introducing a conversational query builder is innovative due to the limited existence of a query build that works in a conversational manner. This feature of the system not only

saves time for medical researchers but also saves the researcher from the complexity of defining a cohort or a concept set in the current tool, ATLAS.

However, some work remains to be done in the future, namely to enhance the query builder feature in order to provide the full cohort definition. Due to the complexity of a cohort definition, the current system only attempts the concept set definition and the observation window, which are parts of the cohort. More work is needed to enhance this system, such as improving the user interface, enhancing the IR methods to produce better results, and creating synthetic labeled dataset to test and validate the IR methods.

The proposed system addresses some real and meaningful challenges in the medical research field, essentially the complexity of finding the best databases in a vast catalogue and defining a study protocol in the ATLAS platform. It led to the creation of three scientific papers that contributed to the advancement of the field. This system has the potential to be very useful for medical researchers and play a significant role in driving technological progress.

References

[1] J. R. Rogers, G. Hripcsak, Y. K. Cheung, and C. Weng, «Clinical comparison between trial participants and potentially eligible patients using electronic health record data: A generalizability assessment method», *Journal of biomedical informatics*, 2021. DOI: 10.1016/j.jbi.2021.103822.

[2] U. Topaloglu and M. B. Topaloglu, «Using a federated network of real-world data to optimize clinical trials operations», *JCO clinical cancer informatics*, 2018. DOI: 10.1200/CCI.17.00067.

[3] G. Lombardo *et al.*, «Electronic health records (EHRs) in clinical research and platform trials: Application of the innovative EHR-based methods developed by EU-PEARL», *Journal of Biomedical Informatics*, 2023. DOI: 10.1016/j.jbi.2023.104553.

[4] J. R. Almeida, L. B. Silva, I. Bos, P. J. Visser, and J. L. Oliveira, «A methodology for cohort harmonisation in multicentre clinical research», *Informatics in Medicine Unlocked*, 2021. DOI: 10.1016/j.imu.2021.100760.

[5] D. C. Kaelber, A. K. Jha, D. Johnston, B. Middleton, and D. W. Bates, «A research agenda for Personal Health Records (PHRs)», *Journal of the American Medical Informatics Association*, 2008. DOI: 10.1197/jamia.M2547.

[6] A. Pereira, J. R. Almeida, R. P. Lopes, and J. L. Oliveira, «Querying semantic catalogues of biomedical databases», *Journal of Biomedical Informatics*, 2023. DOI: 10.1016/j.jbi.2022.104272.

[7] J. R. Almeida, J. P. Barraca, and J. L. Oliveira, «A secure architecture for exploring patient-level databases from distributed institutions», in *2022 IEEE 35th International Symposium on Computer-Based Medical Systems (CBMS)*, IEEE, 2022. DOI: 10.1109/CBMS55023.2022.00086.

[8] G. Hripcsak *et al.*, «Characterizing treatment pathways at scale using the OHDSI network», *Proceedings of the National Academy of Sciences*, 2016. DOI: 10.1073/pnas.1510502113.

[9] K. Park *et al.*, «Exploring the potential of omop common data model for process mining in healthcare», *Plos one*, 2023. DOI: 10.1371/journal.pone.0279641.

[10] J. R. Almeida, J. F. Silva, S. Matos, and J. L. Oliveira, «A two-stage workflow to extract and harmonize drug mentions from clinical notes into observational databases», *Journal of Biomedical Informatics*, DOI: 10.1016/j.jbi.2021.103849.

[11] C. Reich *et al.*, «OHDSI Standardized Vocabularies—a large-scale centralized reference ontology for international data harmonization», *Journal of the American Medical Informatics Association*, 2024. DOI: 10.1093/jamia/ocad247.

[12] J. R. Almeida, A. Pazos, and J. L. Oliveira, «Clinical Data Integration Strategies for Multicenter Studies», in *Doctoral Conference on Computing, Electrical and Industrial Systems*, Springer, 2023. DOI: 10.1007/978-3-031-36007-7_13.

[13] J. R. Almeida and J. L. Oliveira, «MONTRA2: A web platform for profiling distributed databases in the health domain», *Informatics in Medicine Unlocked*, 2024. DOI: 10.1016/j.imu.2024.101447.

[14] L. B. Silva, A. Trifan, and J. L. Oliveira, «MONTRA: An agile architecture for data publishing and discovery», *Computer methods and programs in biomedicine*, 2018. DOI: 10.1016/j.cmpb.2018.03.024.

[15] J. L. Oliveira, A. Trifan, and L. A. B. Silva, «EMIF Catalogue: a collaborative platform for sharing and reusing biomedical data», *International journal of medical informatics*, 2019. DOI: 10.1016/j.ijmedinf.2019.02.006.

[16] J. R. Almeida, J. M. Silva, and J. L. Oliveira, «A FAIR approach to real-world health data management and analysis», in *2023 IEEE 36th International Symposium on Computer-Based Medical Systems (CBMS)*, IEEE, 2023. DOI: 10.1109/CBMS58004.2023.00338.

[17] J. A. Reis, J. R. Almeida, T. M. Almeida, and J. L. Oliveira, «Using Flowise to Streamline Biomedical Data Discovery and Analysis», in *2024 IEEE 22nd Mediterranean Electrotechnical Conference (MELECON)*, IEEE, 2024.

[18] J. A. Reis, J. R. Almeida, T. M. Almeida, and J. L. Oliveira, «A chatbot-like platform to enhance the discovery of OMOP CDM databases», in *Digital Personalized Health and Medicine*, IOS Press, 2024.

[19] J. R. Almeida *et al.*, «HealthDBFinder: a question-answering task for health database discovery», in *2024 IEEE 37th International Symposium on Computer-Based Medical Systems (CBMS)*, IEEE, 2024.

[20] Kumar, T. Rao, A. R. Lakshminarayanan, and E. Pugazhendi, «An Efficient Text-Based Image Retrieval Using Natural Language Processing (NLP) Techniques», in Jan. 2021, ISBN: 9789811553998. DOI: 10.1007/978-981-15-5400-1_52.

[21] K. A. Hambarde and H. Proença, «Information Retrieval: Recent Advances and Beyond», *IEEE Access*, 2023, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2023.3295776.

[22] B. Mitra and N. Craswell, «An Introduction to Neural Information Retrieval»,

[23] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, Oct. 2018.

[24] M. Liang and T. Niu, «Research on Text Classification Techniques Based on Improved TF-IDF Algorithm and LSTM Inputs», *Procedia Computer Science*, 7th International Conference on Intelligent, Interactive Systems and Applications, Jan. 2022, ISSN: 1877-0509. DOI: 10.1016/j.procs.2022.10.064.

[25] C. Manning, P. Raghavan, and H. Schuetze, «Introduction to Information Retrieval», 2009.

[26] F. Lan, «Research on Text Similarity Measurement Hybrid Algorithm with Term Semantic Information and TF-IDF Method», *Advances in Multimedia*, 2022, ISSN: 1687-5680. DOI: 10.1155/2022/7923262.

[27] E. Gomede, *Understanding the BM25 Ranking Algorithm*, Sep. 2023. [Online]. Available: https://medium.com/@evertongomede/understanding-the-bm25-ranking-algorithm-19f6d45c6ce.

[28] S. Connelly, *Practical BM25 - Part 2: The BM25 Algorithm and its Variables*, Apr. 2018. [Online]. Available: https://www.elastic.co/blog/practical-bm25-part-2-the-bm25-algorithm-and-its-variables.

[29] H. Chen, Z. Dou, Q. Zhu, X. Zuo, and J.-R. Wen, «Integrating Representation and Interaction for Context-Aware Document Ranking», *ACM Transactions on Information Systems*, 2023. DOI: 10.1145/3529955.

[30] J. Liu, X. Chu, Y. Wang, and M. Wang, «Deep Text Retrieval Models based on DNN, CNN, RNN and Transformer: A review», in *2022 IEEE 8th International Conference on Cloud Computing and Intelligent Systems (CCIS)*, Nov. 2022. DOI: 10.1109/CCIS57298.2022.10016379.

[31] S. Ayanouz, B. A. Abdelhakim, and M. Benhmed, «A Smart Chatbot Architecture based NLP and Machine Learning for Health Care Assistance», in *Proceedings of the 3rd International Conference on Networking, Information Systems & Security*, 2020, ISBN: 978-1-4503-7634-1. DOI: 10.1145/3386723.3387897.

[32] E. W. T. Ngai, M. C. M. Lee, M. Luo, P. S. L. Chan, and T. Liang, «An intelligent knowledge-based chatbot for customer service», *Electronic Commerce Research and Applications*, Nov. 2021, ISSN: 1567-4223. DOI: 10.1016/j.elerap.2021.101098.

[33] A. Chizhik and Y. Zherebtsova, «Challenges of Building an Intelligent Chatbot», 2020.

[34] B. Zhong, W. He, Z. Huang, P. E. Love, J. Tang, and H. Luo, «A building regulation question answering system: A deep learning methodology», *Advanced Engineering Informatics*, Oct. 2020, ISSN: 14740346. DOI: 10.1016/j.aei.2020.101195.

[35] T. A. Chang and B. K. Bergen, *Language Model Behavior: A Comprehensive Survey*, Aug. 2023.

[36] X. Liu *et al.*, «Prompting Frameworks for Large Language Models: A Survey», *ACM Comput. Surv.*,

[37] A. Vaswani *et al.*, *Attention Is All You Need*, Aug. 2023.

[38] W. X. Zhao *et al.*, *A Survey of Large Language Models*, Nov. 2023. DOI: 10.48550/arXiv.2303.18223.

[39] M. U. Hadi *et al.*, «Large language models: A comprehensive survey of its applications, challenges, limitations, and future prospects», Jul. 2023. DOI: 10.36227/techrxiv.23589741.

[40] H. Naveed *et al.*, *A Comprehensive Overview of Large Language Models*, Nov. 2023.

[41] R. Anil *et al.*, *PaLM 2 Technical Report*, Sep. 2023. DOI: 10.48550/arXiv.2305.10403.

[42] OpenAI *et al.*, *GPT-4 Technical Report*, Dec. 2023. DOI: 10.48550/arXiv.2303.08774.

[43] H. Touvron *et al.*, *Llama 2: Open Foundation and Fine-Tuned Chat Models*, Jul. 2023. DOI: 10.48550/arXiv.2307.09288.

[44] B. Min *et al.*, *Recent Advances in Natural Language Processing via Large Pre-trained Language Models: A Survey | ACM Computing Surveys*.

[45] *OpenAI Platform*. [Online]. Available: https://platform.openai.com.

[46] S. Kamnis, «Generative pre-trained transformers (GPT) for surface engineering», *Surface and Coatings Technology*, vol. 466, Aug. 2023, ISSN: 0257-8972. DOI: 10.1016/j.surfcoat.2023.129680.

[47] E. J. Hu *et al.*, *LoRA: Low-Rank Adaptation of Large Language Models*, Oct. 2021. DOI: 10.48550/arXiv.2106.09685.

[48] B. Meskó, «Prompt Engineering as an Important Emerging Skill for Medical Professionals: Tutorial», *Journal of Medical Internet Research*, Oct. 2023, ISSN: 1438-8871. DOI: 10.2196/50638.

[49] X. Ma *et al.*, *Beyond ChatBots: ExploreLLM for Structured Thoughts and Personalized Model Responses*, Dec. 2023.

[50] J. Wei *et al.*, *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*, Jan. 2023. DOI: 10.48550/arXiv.2201.11903.

[51] S. Yao *et al.*, *ReAct: Synergizing Reasoning and Acting in Language Models*, Mar. 2023. DOI: 10.48550/arXiv.2210.03629.

[52] E. Almazrouei *et al.*, *The Falcon Series of Open Language Models*, Nov. 2023. DOI: 10.48550/arXiv.2311.16867.

[53] A. Q. Jiang *et al.*, *Mistral 7B*, Oct. 2023.

[54] A. Q. Jiang *et al.*, *Mixtral of Experts*, Jan. 2024. DOI: 10.48550/arXiv.2401.04088.

[55] K. Church and R. Yue, «Emerging trends: Smooth-talking machines», *Natural Language Engineering*, 2023, ISSN: 1351-3249. DOI: 10.1017/S1351324923000463.

[56] N. Kshetri, «Cybercrime and Privacy Threats of Large Language Models», *IT Professional*, 2023, ISSN: 1520-9202. DOI: 10.1109/MITP.2023.3275489.

[57] B. Borah, D. Pathak, P. Sarmah, B. Som, and S. Nandi, «Survey of Textbased Chatbot in Perspective of Recent Technologies», *Communications in Computer and Information Science*, 2019, ISSN: 1865-0929. DOI: 10.1007/978-981-13-8581-0_7.

[58] C. V. Misischia, F. Poecze, and C. Strauss, «Chatbots in customer service: Their relevance and impact on service quality», Jan. 2022, ISSN: 1877-0509. DOI: 10.1016/j.procs.2022.03.055.

[59] M. Nuruzzaman and O. K. Hussain, «A Survey on Chatbot Implementation in Customer Service Industry through Deep Neural Networks», in *2018 IEEE 15th International Conference on e-Business Engineering (ICEBE)*, Oct. 2018. DOI: 10.1109/ICEBE.2018.00019.

[60] Z. Peng and X. Ma, «A survey on construction and enhancement methods in service chatbots design», *CCF Transactions on Pervasive Computing and Interaction*, 2019, ISSN: 2524-521X. DOI: 10.1007/s42486-019-00012-3.

[61] J. Weizenbaum, «ELIZA—a computer program for the study of natural language communication between man and machine», *Communications of the ACM*, Jan. 1966. DOI: 10.1145/365153.365168.

[62] R. Agarwal and M. Wadhwa, «Review of State-of-the-Art Design Techniques for Chatbots», 2020, ISSN: 2662-995X. DOI: 10.1007/s42979-020-00255-3.

[63] G. Li, R. Gomez, K. Nakamura, and B. He, «Human-Centered Reinforcement Learning: A Survey», *IEEE Transactions on Human-Machine Systems*, Aug. 2019, ISSN: 2168-2305. DOI: 10.1109/THMS.2019.2912447.

[64] Q.-D. Tran, A.-C. Le, and V.-N. Huynh, «Enhancing Conversational Model With Deep Reinforcement Learning and Adversarial Learning», *IEEE Access*, 2023, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2023.3297652.

[65] A. Axelsson, H. Buschmeier, and G. Skantze, «Modeling Feedback in Interaction With Conversational Agents—A Review», *Frontiers in Computer Science*, 2022, ISSN: 2624-9898. DOI: 10.3389/fcomp.2022.744574.

[66] P. Lewis *et al.*, «Retrieval-augmented generation for knowledge-intensive NLP tasks», in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, Curran Associates Inc., Dec. 2020, ISBN: 978-1-71382-954-6.

[67] Y. Gao *et al.*, *Retrieval-Augmented Generation for Large Language Models: A Survey*, Dec. 2023. DOI: 10.48550/arXiv.2312.10997.

[68] O. Mussa, O. Rana, B. Goossens, P. Orozco-TerWengel, and C. Perera, «ForestQB: An Adaptive Query Builder to Support Wildlife Research», 2022.

[69] *jQuery QueryBuilder*. [Online]. Available: https://querybuilder.js.org/.

[70] *Introducing Query Builder: Write complex queries without code*, Aug. 2021. [Online]. Available: https://www.dronahq.com/introducing-query-builder/.

[71] OHDSI, *Chapter 10 Defining Cohorts | The Book of OHDSI*. [Online]. Available: https://ohdsi.github.io/TheBookOfOhdsi/.

[72] T. Formal, B. Piwowarski, and S. Clinchant, *SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking*, Jul. 2021. DOI: 10.48550/arXiv.2107.05720.

[73] J. Chen, S. Xiao, P. Zhang, K. Luo, D. Lian, and Z. Liu, *BGE M3-Embedding: Multi-Lingual, Multi-Functionality, Multi-Granularity Text Embeddings Through Self-Knowledge Distillation*, Feb. 2024. DOI: 10.48550/arXiv.2402.03216.

[74] T. Formal, C. Lassance, B. Piwowarski, and S. Clinchant, *From Distillation to Hard Negative Sampling: Making Sparse Neural IR Models More Effective*, May 2022. DOI: 10.48550/arXiv.2205.04733.

[75] Teknium, theemozilla, karan4d, and huemin_art, *Nous Hermes 2 Mixtral 8x7B DPO*. [Online]. Available: https://huggingface.co/NousResearch/Nous-Hermes-2-Mixtral-8x7B-DPO.

[76] A. Q. Jiang *et al.*, «Mixtral of experts», *arXiv:2401.04088*, Jan. 2024.

[77] J. R. Almeida, A. Zúquete, A. Pazos, and J. L. Oliveira, «A Federated Authentication Schema among Multiple Identity Providers», *Heliyon*, 2024. DOI: 10.1016/j.heliyon.2024.e28560.

[78] O. H. D. S. a. Informatics, *The Book of OHDSI*. [Online]. Available: https://ohdsi.github.io/TheBookOfOhdsi/.

[79] W. Ritzel Paixão-Côrtes, V. Stangherlin Machado Paixão-Côrtes, C. Ellwanger, and O. Norberto de Souza, «Development and usability evaluation of a prototype conversational interface for biological information retrieval via bioinformatics», in *Human Interface and the Management of Information. Visual Information and Knowledge Management: Thematic Area, HIMI 2019, Held as Part of the 21st HCI International Conference, HCII 2019, Orlando, FL, USA, July 26–31, 2019, Proceedings, Part I 21*, Springer, 2019. DOI: 10.1007/978-3-030-22660-2_43.