Universidade de Aveiro
2024

JOÃO ANTÓNIO
ASSIS REIS

UM SISTEMA CONVERSACIONAL DE PESQUISA SOBRE BASE DE DADOS MÉDICAS

A CONVERSATIONAL QUERY BUILDER ON MEDICAL DATABASES

DOCUMENTO
PROVISÓRIO

Universidade de Aveiro
2024

JOÃO ANTÓNIO
ASSIS REIS

UM SISTEMA CONVERSACIONAL DE PESQUISA SOBRE BASE DE DADOS MÉDICAS

A CONVERSATIONAL QUERY BUILDER ON MEDICAL DATABASES

DOCUMENTO PROVISÓRIO

"*The greatest challenge to any thinker is stating the problem in a way that will allow a solution*"

— Bertrand Russell

Universidade de Aveiro
2024

JOÃO ANTÓNIO
ASSIS REIS

UM SISTEMA CONVERSACIONAL DE PESQUISA SOBRE BASE 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

presidente / president

Prof. Doutor João Antunes da Silva
professor associado da Faculdade de Engenharia da Universidade do Porto

vogais / examiners committee

Prof. Doutor João Antunes da Silva
professor associado da Faculdade de Engenharia da Universidade do Porto

Prof. Doutor João Antunes da Silva
professor associado da Faculdade de Engenharia da Universidade do Porto

Prof. Doutor João Antunes da Silva
professor associado da Faculdade de Engenharia da Universidade do Porto

Prof. Doutor João Antunes da Silva
professor associado da Faculdade de Engenharia da Universidade do Porto

Prof. Doutor João Antunes da Silva
professor associado da Faculdade de Engenharia da Universidade do Porto

agradecimentos / acknowledgements

Agradeço toda a ajuda a todos os meus orientadores, amigos e família.

Palavras Chave

texto livro, arquitetura, história, construção, materiais de construção, saber tradicional.

Resumo

Um resumo é um pequeno apanhado de um trabalho mais longo (como uma tese, dissertação ou trabalho de pesquisa). O resumo relata de forma concisa os objetivos e resultados da sua pesquisa, para que os leitores saibam exatamente o que se aborda no seu documento.
Embora a estrutura possa variar um pouco dependendo da sua área de estudo, o seu resumo deve descrever o propósito do seu trabalho, os métodos que você usou e as conclusões a que chegou.
Uma maneira comum de estruturar um resumo é usar a estrutura IMRaD. Isso significa:

- Introdução
- Métodos
- Resultados
- Discussão

Veja mais pormenores aqui:
`https://www.scribbr.com/dissertation/abstract/`

Keywords textbook, architecture, history, construction, construction materials, traditional knowledge.

Abstract

An abstract is a short summary of a longer work (such as a thesis, dissertation or research paper).
The abstract concisely reports the aims and outcomes of your research, so that readers know exactly what your paper is about.
Although the structure may vary slightly depending on your discipline, your abstract should describe the purpose of your work, the methods you've used, and the conclusions you've drawn.
One common way to structure your abstract is to use the IMRaD structure. This stands for:

- Introduction
- Methods
- Results
- Discussion

Check for more details here:
`https://www.scribbr.com/dissertation/abstract/`

Contents

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

[2]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*

List of Figures

List of Tables

List of Listings

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

CHAPTER 1

Introduction

The continuous quest for medical answers and advancements in clinical research, combined with the diversity of medical databases, has sparked complex challenges for researchers. A recurring issue is the scarcity of specific medical data that are the focus of a study, such as cases of patients with rare diseases. In this regard, a promising strategy has emerged, which consists of integrating multiple and diverse medical databases.

However, the implementation of this strategy is not free of obstacles, with the issue of heterogeneity between databases being a prominent challenge. 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.

It is in this context that the Observational Medical Outcomes Partnership Common Data Model (OMOP CDM) and the Observational Health Data Sciences and Informatics (OHDSI) initiative have emerged as good solutions to the problem of heterogeneity and interoperability among clinical medical data. Generally speaking, OMOP CDM is a common data model that establishes a universal standard for representing patient clinical information, allowing for interoperability among disparate databases. The OHDSI initiative is, in turn, an international collaboration composed of researchers and scientists committed to the mission of developing analytical, open-source solutions for an extensive network of medical databases, following systematic analysis of this heterogeneous data.

With the assistance of OMOP CDM and OHDSI, the challenges of data interoperability are overcome, enabling the discovery of crucial insights for advancing and improving medical studies. Sharing this data presents numerous advantages for researchers, including promoting new fields of study and a significant increase in the impact and recognition of research results.

1.1 Motivation

The search for data sources of interest for a researcher's study can be complex due to the large number of databases in the community. Some of these databases are grouped into catalogs to face this challenge. This strategy consists of characterizing the data by aggregating data and metadata.

The European Health Data Evidence Network (EHDEN) portal is an excellent example of a platform that provides a catalog of medical databases from across Europe. It is a centralized repository that facilitates the discovery of relevant data sources for researchers.

Despite the assistance provided by the catalog offered by EHDEN, identifying the most suitable databases for a specific study remains a challenge. Thus, to facilitate search in the catalog, Network Dashboards have emerged, offering statistical and aggregated information about the databases available on the EHDEN network. With this tool, researchers can filter the most suitable databases for their research needs and make more informed decisions.

Even with all this help, choosing the most appropriate databases is difficult and time-consuming, making it difficult to achieve the ideal search desired for the study. The challenge to be addressed is to assist a medical researcher in reaching the ideal search based on the protocol and parameters of their study.

1.2 Objectives

The main objective of this work is to develop a conversational query builder to help medical researchers define their study objectives. 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 a study protocol?*

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 a study protocol;
2. Developed a chat-like search engine to help discover the best databases for a study;
3. Enhance the engine to collect additional information to provide a query as an outcome.

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 user assistant will return a query to help discover the best databases for a specific research. Studying state-of-the-art Information Retrieval systems to retrieve the most appropriate databases for a researcher's query is essential in this context. In addition, it is vital to explore Large Language Models, a recent advance of algorithms that are known for their language generation ability, and the actual state of conversational user assistants or chatbots. In the end, research about interactive query builder. And so these topics will be discussed below.

2.1 Information Retrieval

In computing and information science, Information Retrieval (IR) involves retrieving information from collections of unstructured data, such as documents. According to Kumar *et al.* [3], 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, thereby preventing 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 [4], 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. Nevertheless, these systems based on terms have limitations, including issues like polysemy, synonymy, and linguistic gaps, which may restrict their effectiveness.

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 [4].

In turn, deep learning methods have evolved. Neural Network Architectures, transfer learning, and pre-training techniques emerged. These approaches have advanced the representation of textual data and bolstered the IR system's comprehension of natural language queries.

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 (masked) language models like Bidirectional Encoder Representations from Transformers (BERT) [5] and Generative Pre-trained Transformer (GPT)-2 has proven to enhance the performance of IR systems, offering 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.* [3], IR generally functions across three main scales: searching the web, retrieving personal information, and conducting searches for enterprises, institutions, and domain-specific contexts.

This section explores 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), briefly explained next.

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). Manning *et al.* [6] 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. So, 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, and this is the problem with the raw method explained above. Manning *et al.* [6] clarified this with the following example: the automotive industry is expected to include the term "auto" in nearly every document. The IDF calculates the rarity of a term across a set of documents. This measure is calculated as the logarithm of the inverse fraction of documents containing the term. The goal is to help prioritize some terms that are sporadic and possibly more informative.

The traditional IR method, TF-IDF, combines TF and IDF definitions, as the name suggests, and then produces a weight for each term in each document, as Manning *et al.* [6] and [7] mention. Equation 2.1 shows that the weight is calculated as the product of TF and IDF values, highlighting terms that are both important within a specific document and relatively uncommon in the document collection.

$$\text{tf-idf}_{t,d} = \text{tf}_{t,d} \times \text{idf}_t. \tag{2.1}$$

Manning *et al.* [6] noted the TF-IDF weight assigned to a term in a document is highest when the term frequently appears in a few documents, providing discriminating solid power. The weight is lower when the term occurs less frequently in a document or is widespread across many documents, indicating a less pronounced relevance signal. The weight is at its lowest when the term is present in nearly all documents. However, TF-IDF does not consider the semantic information of words, which limits its ability to accurately reflect the similarity between texts [7]. To address this limitation, researchers have proposed hybrid methods that combine TF-IDF with semantic understanding to calculate text similarity.

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 [4] noted 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. Recapping from the TF-IDF section, TF is the number of occurrences that a specific term is in a document, and IDF is a measure that indicates the importance of a term in the whole document. Equation 2.2 gives the formula to calculate the BM25 score.

$$score(D,Q) = \sum_{i}^{n} IDF(q_i) * \frac{f(q_i,D) * (k1+1)}{f(q_i,D) + k1 * (1 - b + b * \frac{fieldLen}{avgFieldLen})}. \tag{2.2}$$

As Gomede [8] explained, the BM25 equation is composed of the ith query term (q) and the respective IDF and TF values. Also, include a division between the DL, represented in the formula as fieldLen, and the average document length, avgFieldLen. This ratio evaluates how much the length of the document field deviates from the average length. So, the Connelly [9] 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 is 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, so getting the context and, in turn, getting better results is challenging.

While traditional information retrieval methods like TF-IDF and BM25 have been effective in matching queries to documents based on keyword frequencies and document lengths, the evolution of Artificial Intelligence (AI), data complexity and user needs has led to the development of neural IR systems.

2.1.2 Neural Information Retrieval Systems

Neural IR systems represent a significant advancement in the field of IR. Conforming to Mitra and Craswell [10], these systems utilize neural network models and deep learning techniques to understand and interpret the semantic content of queries and documents, offering a more nuanced approach. Neural IR systems can be broadly categorized into two types: Representation-based and Interaction-based models.

Conforming to Chen *et al.* [11], the representation-based model focuses on creating representations of both queries and documents. They employ techniques to convert text into high-dimensional vector spaces. In these 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 utilizes architectures like Recurrent Neural Networks (RNN) and allows the system to understand the content of documents and queries on a deeper level.

The interaction-based model examines how words in a query relate to words in a document, capturing complex patterns and dependencies between them [11]. 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 some approaches to adopting an interaction-based model. One of them is the Retrieval and Ranker. According to Hambarde and Proença [4], this method can be separated into two stages, as the name suggests: Retrieval and Ranker. The following figure, adapted from Hambarde and Proença [4], shows us an overview of interaction-based IR systems, highlighting the two main stages.

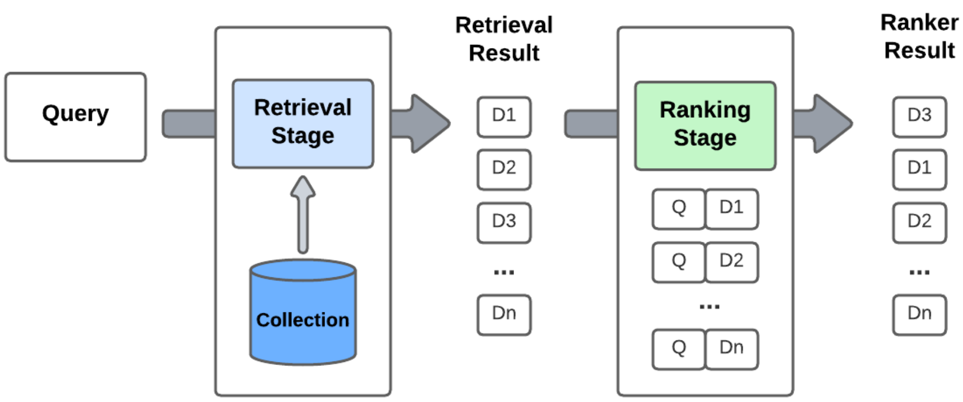

Figure 2.1: Overview of an interaction-based Information Retrieval model: Retrieval and Ranker.

After analyzing the query, the retrieval stage will select 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 [11]. 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, so it is important to give an overview of this technique. It is a field of AI whose primary goal is to understand, interpret, and generate human language. The NLP can be divided into two major components: Natural Language Understanding (NLU) and Natural Language Generation (NLG), according to Ayanouz *et al.* [12]

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.* [13], the user queries can be processed by semantic analysis, pragmatic analysis, and syntactic analysis. Ayanouz *et al.* [12] explained these steps and added two more necessary steps to make it easier to understand: a lexical analysis and discourse integration.

- **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 [14] 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. Simply put, it grapples with the challenge of navigating the intricacies of natural human language [13].

Backtracking, QA is a subfield of IR and NLP. According to Zhong *et al.* [15], 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 utilized by existing online question-answering search engines, this template-based approach faces limitations in handling multiple questions [15].

So, with the advancement of technology, another approach emerged: deep learning-based question-answering. 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.* [15] highlights BERT that has emerged as a powerful model, utilizing contextualized representations for transfer learning. BERT-based models showcase performance in question-answering tasks, even in domains like medicine.

2.2 Large Language Models

It is crucial to trace briefly the development history to understand the concept of Large Language Models (LLM). Liu *et al.* [16] explained this simply and intuitively.

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 [17].

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 [16].

Most recently, the launch of the Transformer Block Architecture by Vaswani *et al.* [1] 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.

Following this, the researchers have involved the scale of model parameters, and the paradigm of “Pre-train, Prompt, Predict" like Liu *et al.* [16] call, gained widespread acceptance. So, 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.

This section discusses mainly LLM, exploring briefly their architecture and comparison between foundation LLM. Finally, it addresses some of its limitations.

2.2.1 Definition

LLM is an advanced LM and belongs to generative AI. It 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 intelligence. These models have a notable ability in natural language tasks, such as text generation and translation, QA, decision-making, summarization, and sentiment analysis.

These models can process and predict patterns with accuracy. Hadi *et al.* [18] 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.* [19], when provided with task descriptions and examples through prompts, LLM can produce textual responses to task queries.

Liu *et al.* [16] say that the release of ChatGPT 1 garnered significant social attention, and research into LLM triggered more interest. This has led to the development of noteworthy products like PaLM, GPT-2, GPT-3, and, most recently, GPT-4, and LLaMA and LLaMa-2.

LLM are revolutionizing NLP and AI research.

2.2.2 Architecture Overview

This subsection discusses the architecture of a LLM, which is supported by the Transformer Block architecture. Also, explains the pre-training process of this advanced LM.

Transformer Architecture

The development and advancement of LLM is thankful for the introduction of Transformers by Vaswani *et al.* [1] 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.* [18], and led to the development of models that have achieved enormous results, such as GPT by OpenAI and BERT by Google. The architecture of this revolutionized model is shown in figure 2.2.

The innovation of this model is due to the multihead self-attention mechanism, one of the key components [1] [18]. 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 [1]. 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 doesn't 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.

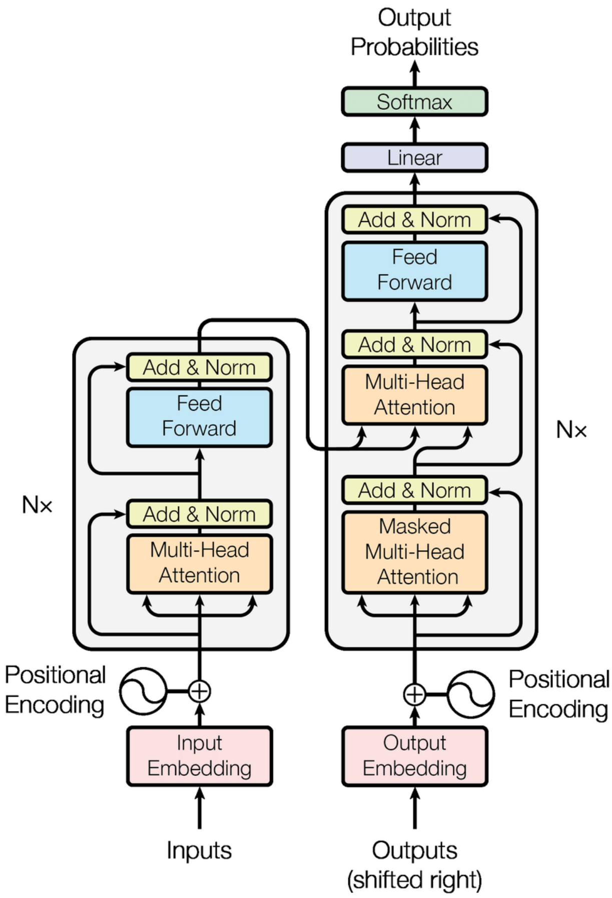

Figure 2.2: The Transformer Block Architecture, from Vaswani *et al.* [1].

Pre-training Process

Learning the patterns and relationships among the data starts with the pre-training process. In compliance with Min *et al.* [20], first, 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.* [18], preparing and preprocessing the data before the training stage is necessary to achieve this. First, 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 [18]. 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 [21] carried out and explained by OpenAI.

After the pre-training process, the LLM goes through an optimization phase.

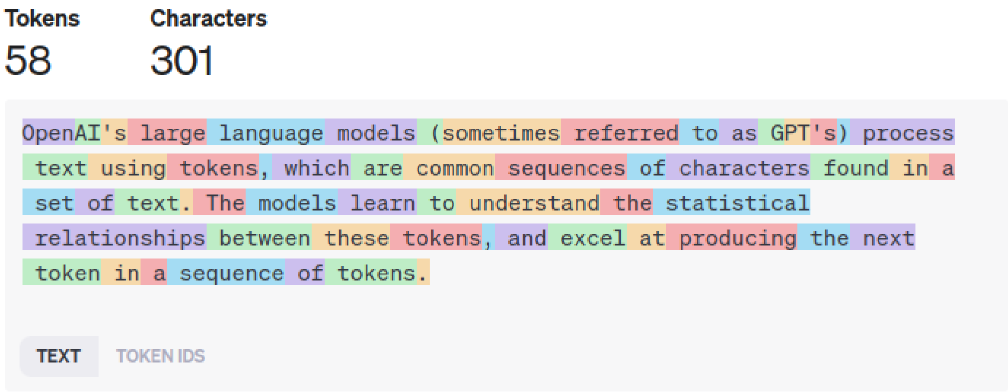

Figure 2.3: Tokenization process visually explained by OpenAI [21].

2.2.3 Optimization Techniques

There are some techniques to optimize the tasks and the accuracy of the LLM. Two of these techniques, Fine-tuning and Prompt Engineering, are explained next.

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 [22] and Hadi *et al.* [18], 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.* [19] 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.* [19] 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. LoRA (Low-Rank Adaptation) is one technique that implements Adapter Tuning, introduced by Hu *et al.* [23]. 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 [24] and Ma *et al.* [25], 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. So, Wei *et al.* [2] 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. Figure 2.4 explains the effect of this technique.

ReAct is a prompt technique introduced by Yao *et al.* [26]. 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.

Meskó [24] raise a series of recommendations for more effective LLM prompts: it must be as precise as possible; providing the setting and the context of the question is essential; describe the goal of the prompt first; give a role to the LLM to get more context (for example, "You are a math teacher and explain the natural numbers"); continuous LLM prompt refinement; prefer open questions over close-questions. Regularly testing prompts in real-world situations is crucial, as their effectiveness is most accurately assessed through practical application.

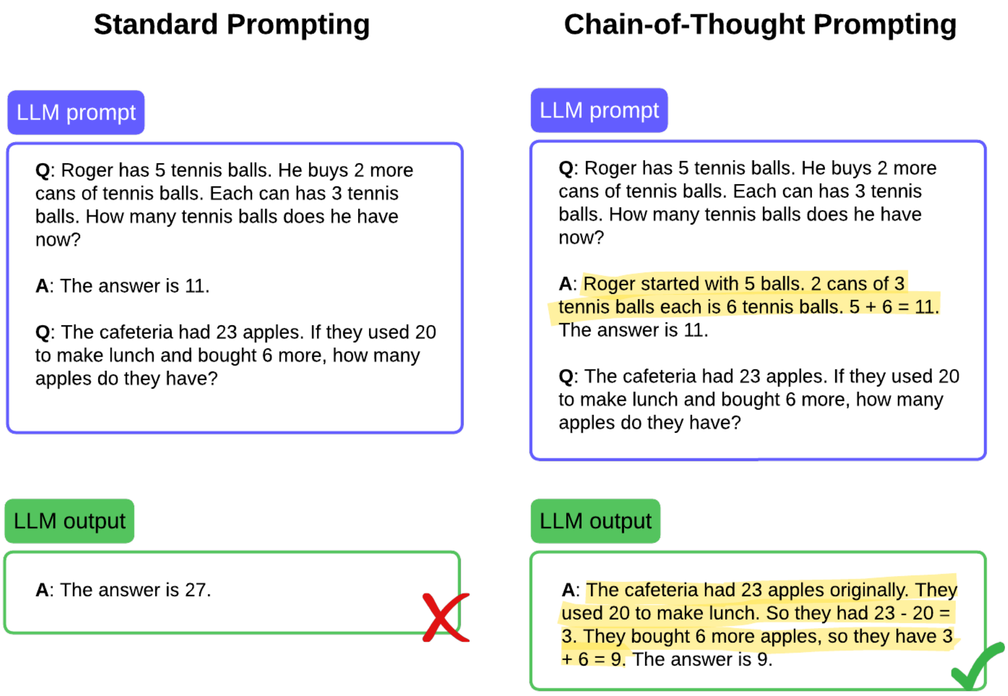

Figure 2.4: Example of Chain-of-thought prompting. The reasoning processes are highlighted in yellow. Adapted from Wei *et al.* [2].

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.* [18] 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 is an open source LLM by Meta (Touvron *et al.* [27]). LLaMa 2 was trained on 40% more data than LLaMa 1, the model from which it came, and has double the context length. So, 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, is a closed source LLM (OpenAI *et al.* [28]). 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 developed by Google (Anil *et al.* [29]). It emerged from PaLM with 540 billion parameters. PaLM 2 is a closed source LLM, and, following the OpenAI approach, has opted to disclose limited technical specifics, including the number of parameters.

The Falcon LLM is an open-source model with impressive performance and scalability (Almazrouei *et al.* [30]). There are three variations of the model size: 7 billion, 40 billion, and the most recent, 180 billion of parameters. This 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 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

Table 2.1: Comparison of foundation Large Language Models.

2.2.5 Limitations

It is safe to say that LLM are significantly impacting the world. According to Liu *et al.* [16], this is justified by their abilities, mainly in-context learning, reasoning for complex content, instruction following, and creative capacity.

However, LLM has some limitations. Hadi *et al.* [18] 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 analyses 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 [31] 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, like LLM, is challenging. So, the explainability of these models is a big limitation [18]. 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 [32]. This is called the Jail Break attack [18]. Another attack is Data Poisoning Attacks, which consist of data poisoning strategies to manipulate the model's output.

Furthermore, Liu *et al.* [16] highlighted another limitation: the temporal lag of the training corpus. LLM cannot retrieve information in real time, and the answer generated may not be the most current.

It is important to be aware of these limitations.

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.

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 [33]. 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.

This section provides a brief overview of chatbots, followed by an in-depth focus on Generative-Based Chatbots. It encompasses important concepts from their definition to some techniques employed in their development and optimization.

2.3.1 Overview of Conversational User Assistants

To provide a comprehensive understanding of conversational user assistants, it's crucial to first explore some of their characteristics, such as domains and method of response generation.

Nuruzzaman and Hussain [34] 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.

It is important to differentiate chatbots in their way of giving or generating a response based on an input query. Peng and Ma [35] 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 [36], was the first chatbot that applied this primitive technique.

A **retrieval-based chatbot** picks 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 [37].

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 [35] [37]. However, this could be complex and costly to implement.

The only type that will be covered will be generative chatbots, due to their capabilities.

2.3.2 Generative-Based Chatbot

Generative-based conversational user assistants are chatbots that use generative models to generate natural language responses. These chatbots utilize sophisticated deep-learning techniques, such as LLM. LLM, as described in section 2.2, has the ability to understand and generate human-like text in context. This advanced LM has been widely used in the modern chatbots. 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).

The RLHF and RAG are explained in more depth below.

Reinforcement Learning from Human Feedback (RLHF)

In the context of AI, according to Li *et al.* [38], RLHF is a popular approach in which an agent learns how to perform a task based on evaluative feedback provided by a human observer.

RLHF in generative-based chatbots 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.* [39] 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.* [40]. 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.

After that, 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 [40].

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)

RAG is a subfield of NLP and AI. This approach was introduced by Lewis *et al.* [41] 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.* [42] 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.

So, according to Lewis *et al.* [41], RAG involves retrieving pertinent information from external knowledge bases, giving more context to the LLM. This enables the LLM to access and utilize up-to-date and/or domain-specific information to enhance response accuracy and relevance. This process leads to reducing hallucinations.

The figure 2.5 shows the workflow of a RAG. This model aims to retrieve relevant information from an extensive corpus of documents when answering questions, subsequently adding this information in the prompt as context to enhance the quality of predictions in compliance with Lewis *et al.* [41].

Gao *et al.* [42] explain simply the workflow. The first step is 1) Retrieve information from an external data source, such as a vector database, as in the example in figure 2.5. This step utilizes IR models, such as BM25, to retrieve relevant information based on the query. The second step is 2) Augment, improving the LLM prompt with the context retrieved in the previous stage. 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.

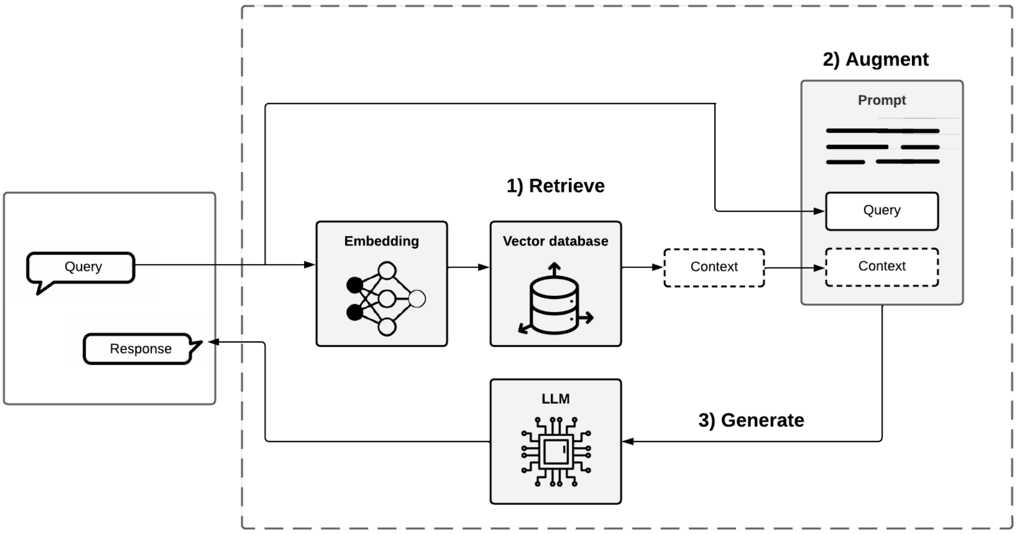

Figure 2.5: Retrieval-Augmented Generation Workflow.

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.* [43] 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 [44].

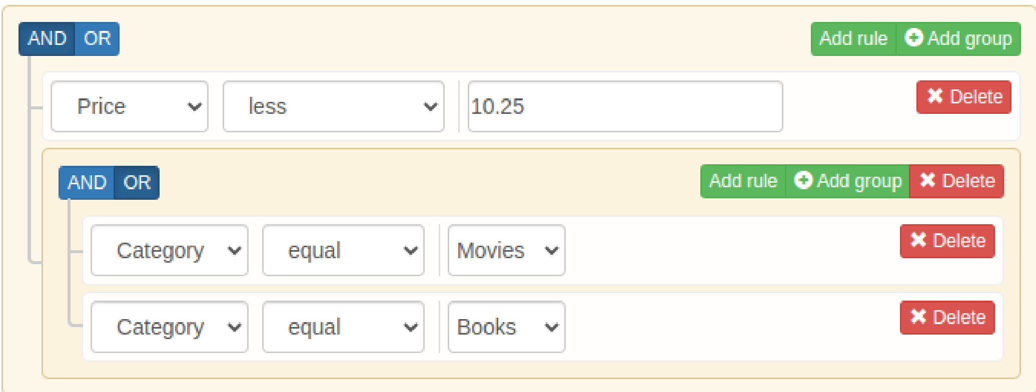

Figure 2.6: Query builder from jQuery QueryBuilder [44].

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 [44].

In summary, a query builder simplifies creating complex queries by providing a visual and interactive interface, making it more accessible [45].

There are several advantages of its use [45]: offers a user-friendly interface with menus, operators, and suggestions to facilitate the creation of accurate queries; users, often without direct permissions to modify the data source, can leverage the query builder to transform datasets without making changes to the underlying database; and, the generated queries are easily modifiable, allowing for flexibility in adjustments or repetitions.

2.4.2 ATLAS

ATLAS, an open-source and web-based software application, is freely accessible and was created by the OHDSI community. It aids in helping researchers conduct scientific analyses 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, according to OHDSI [46], a cohort is a query that defines a set of persons who meet certain inclusion criteria over a specified duration.

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

There are two approaches to building a cohort: rule-based and probabilistic. The most popular one is the rule-based cohort definition, which uses explicit rules to describe when a patient is in the cohort.

In ATLAS, the process of defining a cohort is composed of 3 stages [46]: 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. For instance, if the study is about diabetes, the concept set will include various codes representing diabetes in different medical terminologies. These sets ensure that the cohort captures all relevant instances of the condition or exposure across different healthcare data sources.

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 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.

In compliance with OHDSI [46], a well-defined cohort specifies how a patient enters a cohort and how a patient exits a cohort.

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.

2.5 Summary

To sum up, the proposed query builder is 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 utilization 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 and Falcon 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. However, for this specific case, I don't think much external knowledge will be needed, so the use of fine-tuning should be enough.

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 sophisticated yet relatively 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.

ATLAS by OHDSI provides a cohort definition, which is a query builder to define groups of people based on the research question using healthcare claims data. 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 improves the user experience by making it more intuitive and autonomous.

CHAPTER 3

Conversational Search Assistant

The EHDEN Portal is a web-based platform that facilitates access to information, resources, and tools for data partners, researchers, and other stakeholders involved in the EHDEN project. The portal fosters collaboration, knowledge exchange, and innovation among diverse partners, providing a user-friendly interface to access standardized healthcare data and related resources. EHDEN delivers a catalogue of medical databases across Europe, offering researchers a centralized platform to explore available medical data sources. However, with the increasing number of databases in the catalogue, EHDEN built a tool named Network Dashboards[1] to help researchers to choose the best databases across the catalogue. EHDEN Network Dashboards is a complementary tool of this ecosystem that provides statistical and aggregated information about the databases available on the network.

However, the catalogue's growth is inevitable, making it difficult and time-consuming for a researcher to find his databases of interest. The proposed solution is to build a conversational search tool for the EHDEN catalogue.

This section describes how the EHDEN Catalogue search assistant and its components were implemented and the decisions and steps made over time.

3.1 Data

This section details the structure and contents of the data used in the project, explaining their content, and illustrating how they interconnect to provide a comprehensive overview of the databases in the EHDEN network. The data used for this project is the real EHDEN data from the catalogue and the Network Dashboards. IEETA, a partner of EHDEN, provides this data.

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. 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/{\gls{ehden}}/NetworkDashboards

To comprehend the purpose of the four files, each is detailed with fields, description and an example of entries. 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 analyses, 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:**

[2]https://athena.ohdsi.org/

- 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. The Figure 3.1 shows how these files content are joined.

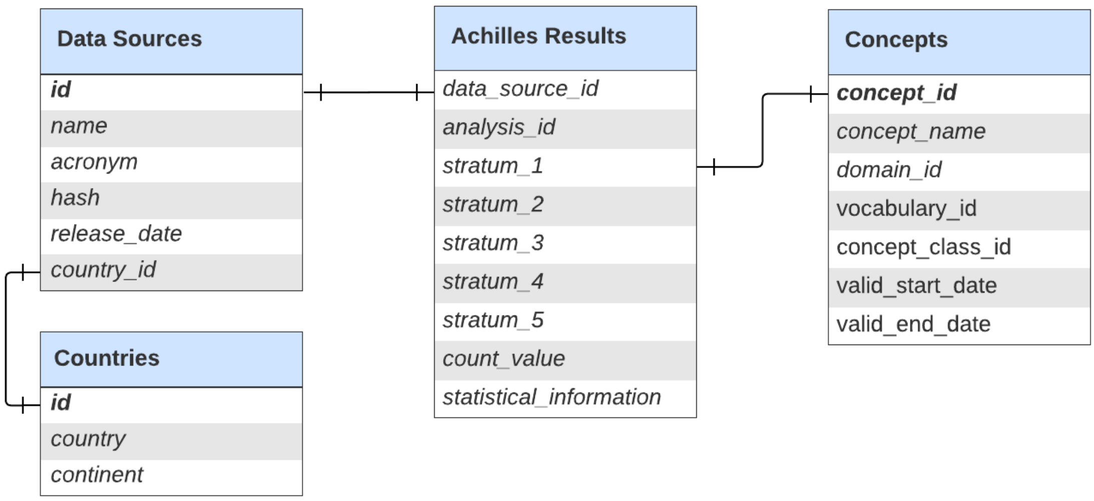

Figure 3.1: The diagram of the connection between the data files.

3.2 Information Retrieval

After understanding the data and its connections, an IR component is crucial to this project to find the most suitable databases. This section discusses mainly the IR methods tested, the process of indexing and seaching in the collection and an API-driven approach of the IR component.

3.2.1 BioASQ Challenge 2024

BioASQ[3] 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, information retrieval, QA from texts and structured data, and multi-document summarization.

The BioASQ includes several challenges. The IEETA team was involved in 'BioASQ Task 12b', focusing on biomedical semantic QA, including IR, QA, and summarization. This

[3] http://bioasq.org/

challenge aims to advance the development of systems capable of understanding and answering biomedical questions. Participants must 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.

To choose and validate the IR method for this project, I joined the IEETA team and was involved in the 2024 edition of the BioASQ challenge. 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, and BGE-M3. BM25 has already been explained, so here's 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 https://arxiv.org/pdf/2107.05720 . This technique belongs to Learned Sparse Retrieval 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. So, it is advantageous because it combines the benefits of dense and sparse models **https://arxiv.org/pdf/2107.05720**. 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` (https://arxiv.org/pdf/2205.04733)

BGE-M3

BGE-M3[5] is a multifunctional technique because it can simultaneously perform the three common retrieval functionalities of embedding model: 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

[4] https://github.com/naver/splade
[5] https://github.com/FlagOpen/FlagEmbedding/tree/master/FlagEmbedding/BGE_M3

project. The technique was implemented using the PyTerrier PISA, a python interface to PISA[6].

This section explains the implementation of BM25, the process of creating documents based on the EHDEN data, and the indexing and searching of these documents.

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. This 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.

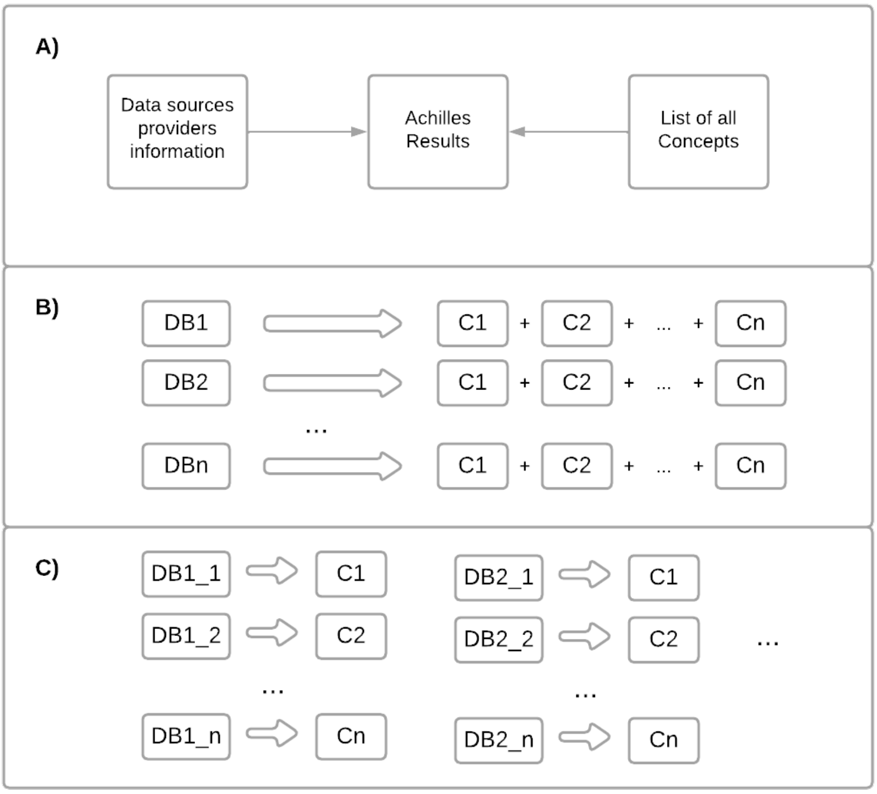

Figure 3.2: Index data structure: A) All information about the databases is composed of three files, B) First approach to the structure of indexed documents, C) Current approach to the structure of indexed documents.

[6] https://github.com/terrierteam/pyterrier_pisa

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.

SO, 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). Standardizing 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 the following information about the databases: a unique numeric identifier, the database name, the hash that identifies the database to be added to the link to access in the EHDEN Catalogue, the total relevance score assigned by the BM25 algorithm, indicating how well the database matches the query, and the concepts list that has been identified.

3.2.3 API-driven Approach

This project's IR component is accessible through an API, allowing seamless integration with external systems. By exposing the endpoints of the IR component, the API facilitates the consumption of search functionalities, ensuring that various external applications can interact with the data effectively.

The API offers endpoints for submitting queries and retrieving search results. When a query is sent to the API, it undergoes the same searching steps described in the previous section 3.2.2.

This API-driven approach enhances the IR component's interoperability and ensures that external systems can seamlessly query and retrieve data. This enables efficient and precise identification of the most suitable databases for any given query, facilitating the integration of the IR component in the following project implementation steps.

3.3 Frameworks to streamline biomedical data discovery[7]

To achieve the proposed goal of making a conversational search assistant, a LLM has a crucial role in generating coherent, contextually relevant text and engaging in human language interactions. The LLM used in this project is `Nous Hermes 2 Mixtral 8x7B` open model. This model belongs to the llama family and has a 47B of parameters. The reason for using this LLM model is that it is a promising model, and the IEETA has installed it for use in its projects.

The LLM is installed a local version of the Ollama framework and deployed in a Virtual Machine to access the LLM using a URL. So, there are some options to integrate the model in the 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.

3.3.1 FlowiseAI

FlowiseAI[8], 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.* [47] 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 **??**.

The conversational agent employs a comprehensive approach to enable dynamic interactions on a healthcare IR platform. It orchestrates the dialogue flow through the Conversational

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

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

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[9] 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 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.

3.3.2 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.

3.4 EHDEN Chatbot[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 **almeida2024federated**. Therefore, the system was implemented to use the MONTRA2-

[9] https://ollama.com/

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

[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*

SDK **almeida2024montra2**. 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 3.3.1, tried to adopt an open-source automation tool to build Chatbot applications, FlowiseAI [47]. 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.

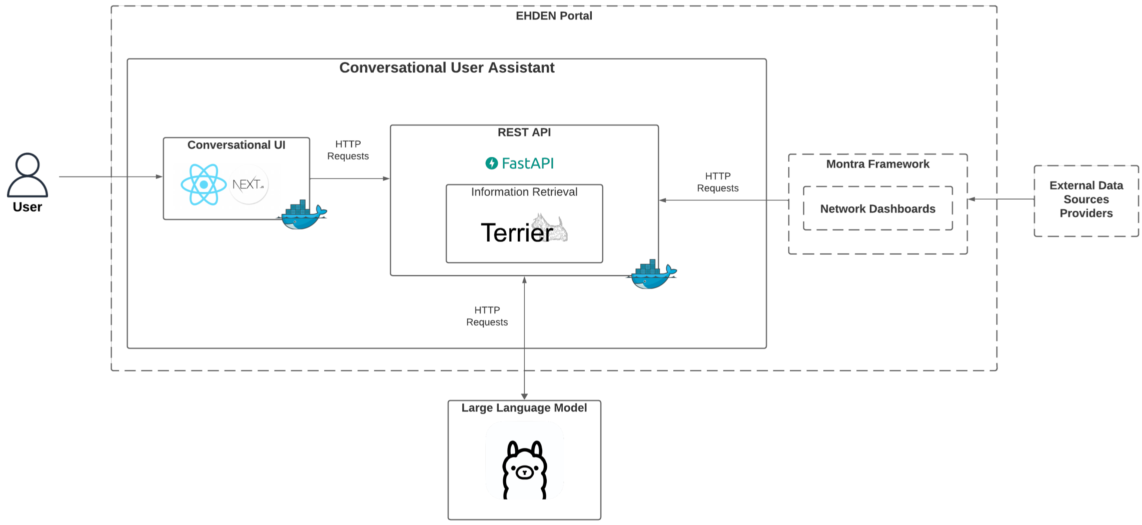

Figure 3.4: Overview of the EHDEN chatbot architecture.

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 recieves a query from the User Interface, the backend communicates with the LLM. The LLM is the same open model used in the previous implementation, `Nous Hermes 2 Mixtral 8x7B`, and it has two tasks that run consecutively: to identify if the query is related to health and to generate responses. If the first task returns false, the LLM generates a response reminding the user of the chatbot's purpose. However, if the first task is true, then the RAG architecture is applied. This means that the BM25 retrieves the best databases, and then the LLM generates a response with that valid information.

CHAPTER 4

Query Builder

4.1 Strategy

4.2 Concept Set Definiton

4.3 Cohort Definiton

4.4 ATLAS ...

```json
{
  "ConceptSets": [
    {
      "expression": {
          "items": "Do you have any other concepts to add to the concept set? (The question should be a yes with the new concepts or no response.)"
      }
    }
  ],
  "PrimaryCriteria": {
    "CriteriaList": [
      {
        "DrugExposure": {
          "CodesetId": 0,
          "First": true
        }
      }
    ],
    "ObservationWindow": {
      "PriorDays": "In terms of the observation window, what is the number of previous days? you can choose from 0, 1, 7, 14, 21, 30, 60, 90, 120, 180, 365, 548, 730 or 1095.",
      "PostDays": "In terms of the observation window, what is the number of days after? you can choose from 0, 1, 7, 14, 21, 30, 60, 90, 120, 180, 365, 548, 730 or 1095."
    },
    "PrimaryCriteriaLimit": {
      "Type": "First"
    }
  },
  "QualifiedLimit": {
    "Type": "First"
  },
  "ExpressionLimit": {
    "Type": "First"
  },
  "InclusionRules": [
    {
      "name": "has hypertension diagnosis in 1 year prior to treatment",
      "expression": {
        "Type": "ALL",
        "CriteriaList": [
          {
            "Criteria": {
              "ConditionOccurrence": {
                "CodesetId": 1
              }
            },
            "StartWindow": {
              "Start": {
                "Days": 365,
                "Coeff": -1
              },
              "End": {
                "Days": 0
```

CHAPTER 5

Results and Discussion

CHAPTER 6

Conclusions

References

[1] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, *Attention Is All You Need*, arXiv:1706.03762 [cs], Aug. 2023. (visited on 12/04/2023).

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

[3] Kumar, T. Rao, A. R. Lakshminarayanan, and E. Pugazhendi, «An Efficient Text-Based Image Retrieval Using Natural Language Processing (NLP) Techniques», in Jan. 2021, pp. 505–519, ISBN: 9789811553998. DOI: 10.1007/978-981-15-5400-1_52.

[4] K. A. Hambarde and H. Proença, «Information Retrieval: Recent Advances and Beyond», en, *IEEE Access*, vol. 11, pp. 76 581–76 604, 2023, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2023.3295776. (visited on 12/06/2023).

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

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

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

[8] E. Gomede, *Understanding the BM25 Ranking Algorithm*, en, Sep. 2023. [Online]. Available: https://medium.com/@evertongomede/understanding-the-bm25-ranking-algorithm-19f6d45c6ce (visited on 12/06/2023).

[9] 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.

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

[11] 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.

[12] 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*, ser. NISS2020, New York, NY, USA: Association for Computing Machinery, 2020, pp. 1–6, ISBN: 978-1-4503-7634-1. DOI: 10.1145/3386723.3387897. (visited on 11/13/2023).

[13] 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*, vol. 50, p. 101 098, Nov. 2021, ISSN: 1567-4223. DOI: 10.1016/j.elerap.2021.101098. (visited on 11/13/2023).

[14] A. Chizhik and Y. Zherebtsova, «Challenges of Building an Intelligent Chatbot», 2020. (visited on 11/13/2023).

[15] B. Zhong, W. He, Z. Huang, P. E. Love, J. Tang, and H. Luo, «A building regulation question answering system: A deep learning methodology», en, *Advanced Engineering Informatics*, vol. 46, p. 101 195, Oct. 2020, ISSN: 14740346. DOI: 10.1016/j.aei.2020.101195. (visited on 12/06/2023).

[16] X. Liu, J. Wang, J. Sun, *et al.*, «Prompting Frameworks for Large Language Models: A Survey», en, *ACM Comput. Surv.*, vol. 1, no. 1,

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

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

[19] H. Naveed, A. U. Khan, S. Qiu, *et al.*, *A Comprehensive Overview of Large Language Models*, arXiv:2307.06435 [cs], Nov. 2023. (visited on 12/15/2023).

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

[21] *OpenAI Platform*, en. [Online]. Available: https://platform.openai.com (visited on 12/15/2023).

[22] S. Kamnis, «Generative pre-trained transformers (GPT) for surface engineering», *Surface and Coatings Technology*, vol. 466, p. 129 680, Aug. 2023, ISSN: 0257-8972. DOI: 10.1016/j.surfcoat.2023.129680. (visited on 01/09/2024).

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

[24] B. Meskó, «Prompt Engineering as an Important Emerging Skill for Medical Professionals: Tutorial», en, *Journal of Medical Internet Research*, vol. 25, e50638, Oct. 2023, ISSN: 1438-8871. DOI: 10.2196/50638. (visited on 12/21/2023).

[25] X. Ma, S. Mishra, A. Liu, *et al.*, *Beyond ChatBots: ExploreLLM for Structured Thoughts and Personalized Model Responses*, arXiv:2312.00763 [cs], Dec. 2023. (visited on 12/21/2023).

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

[27] H. Touvron, L. Martin, K. Stone, *et al.*, *Llama 2: Open Foundation and Fine-Tuned Chat Models*, Jul. 2023. DOI: 10.48550/arXiv.2307.09288. (visited on 01/06/2024).

[28] OpenAI, J. Achiam, S. Adler, *et al.*, *GPT-4 Technical Report*, Dec. 2023. DOI: 10.48550/arXiv.2303.08774. (visited on 01/06/2024).

[29] R. Anil, A. M. Dai, O. Firat, *et al.*, *PaLM 2 Technical Report*, Sep. 2023. DOI: 10.48550/arXiv.2305.10403. (visited on 01/06/2024).

[30] E. Almazrouei, H. Alobeidli, A. Alshamsi, *et al.*, *The Falcon Series of Open Language Models*, Nov. 2023. DOI: 10.48550/arXiv.2311.16867. (visited on 01/06/2024).

[31] K. Church and R. Yue, «Emerging trends: Smooth-talking machines», *Natural Language Engineering*, vol. 29, 2023, ISSN: 1351-3249. DOI: 10.1017/S1351324923000463.

[32] N. Kshetri, «Cybercrime and Privacy Threats of Large Language Models», *IT Professional*, vol. 25, 2023, ISSN: 1520-9202. DOI: 10.1109/MITP.2023.3275489.

[33] C. V. Misischia, F. Poecze, and C. Strauss, «Chatbots in customer service: Their relevance and impact on service quality», vol. 201, Jan. 2022, ISSN: 1877-0509. DOI: 10.1016/j.procs.2022.03.055. (visited on 12/15/2024).

[34] 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, pp. 54–61. DOI: 10.1109/ICEBE.2018.00019. (visited on 11/13/2023).

[35] 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.

[36] 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.

[37] 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.

[38] G. Li, R. Gomez, K. Nakamura, and B. He, «Human-Centered Reinforcement Learning: A Survey», *IEEE Transactions on Human-Machine Systems*, no. 4, Aug. 2019, ISSN: 2168-2305. DOI: 10.1109/THMS.2019.2912447. (visited on 01/17/2024).

[39] 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.

[40] 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.

[41] P. Lewis, E. Perez, A. Piktus, *et al.*, «Retrieval-augmented generation for knowledge-intensive NLP tasks», in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, ser. NIPS'20, Red Hook, NY, USA: Curran Associates Inc., Dec. 2020, pp. 9459–9474, ISBN: 978-1-71382-954-6. (visited on 12/24/2023).

[42] Y. Gao, Y. Xiong, X. Gao, *et al.*, *Retrieval-Augmented Generation for Large Language Models: A Survey*, arXiv:2312.10997 [cs], Dec. 2023. DOI: 10.48550/arXiv.2312.10997. (visited on 12/24/2023).

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

[44] *jQuery QueryBuilder*. [Online]. Available: https://querybuilder.js.org/ (visited on 01/04/2024).

[45] *Introducing Query Builder: Write complex queries without code*, Aug. 2021. [Online]. Available: https://www.dronahq.com/introducing-query-builder/ (visited on 01/05/2024).

[46] OHDSI, *Chapter 10 Defining Cohorts | The Book of OHDSI*. [Online]. Available: https://ohdsi.github.io/TheBookOfOhdsi/ (visited on 01/03/2024).

[47] 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.