Contents

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

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 virtual 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 virtual 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.* [2], 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 [3], 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.

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) 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.* [2], IR generally functions across three main scales: searching the web, retrieving personal information, and conducting searches for enterprises, institutions, and domain-specific contexts.

In this section, I explored the IR field, more specifically, how a IR system works, what are the traditional methods and how IR can be applied with Natural Language Processing (NLP).

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 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.* [4] explained these concepts as follows.

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.* [4] 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.* [4] and Chizhik and Zherebtsova [5] mention. 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. The formula is given by the following mathematical expression:

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

Manning *et al.* [4] 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.

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 [3] 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. The formula is given by the following mathematical expression:

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

As Gomede [6] 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 *Practical BM25 - Part 2: The BM25 Algorithm and its Variables* [7] 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 [8], 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.

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. 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.

Interaction-based model

There are some approaches to adopting an interaction-based model. One of them is the Retrieval and Ranker. According to Hambarde and Proença [3], this method can be separated into two stages, as the name suggests: Retrieval and Ranker. The following figure, adapted from Hambarde and Proença [3], shows us an overview of interaction-based IR systems, highlighting the two main stages.

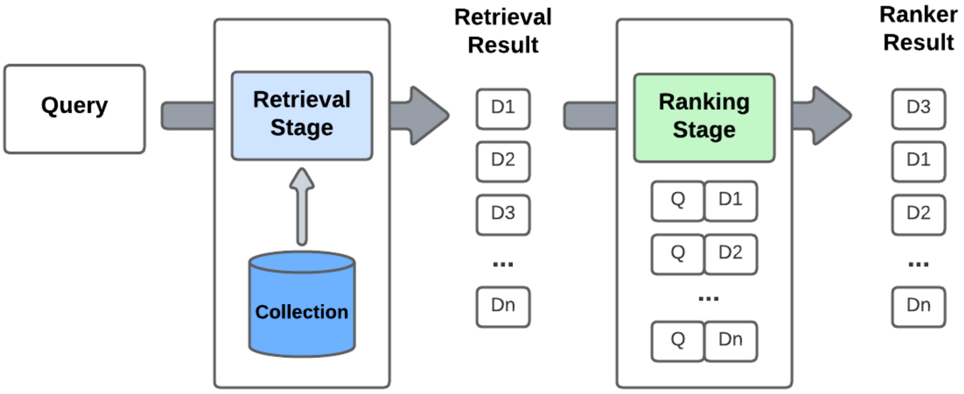


Figure 2.1: Overview of an interaction-based Information Retrieval model: Retrieval and Ranker.

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. 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

Question Answering (QA) is a subfield of IR and NLP. According to Zhong *et al.* [9], 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.

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.* [9] highlights BERT. BERT 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.

Natural Language Processing (NLP)

NLP is the basis for building a QA system. 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.* [10]

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.* [11], the user queries can be processed by semantic analysis, pragmatic analysis, and syntactic analysis. Ayanouz *et al.* [10] 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 [5] 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.

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.* [12] 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 [13].

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.

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.* [12] 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.

In this section, I defined a LLM, explored briefly their architecture, compared between foundation LLM, and finally, addressed 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.* [14] 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.* [15], when provided with task descriptions and examples through prompts, LLM can produce textual responses to task queries.

Liu *et al.* [12] 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

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.* [14], 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. 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. 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.

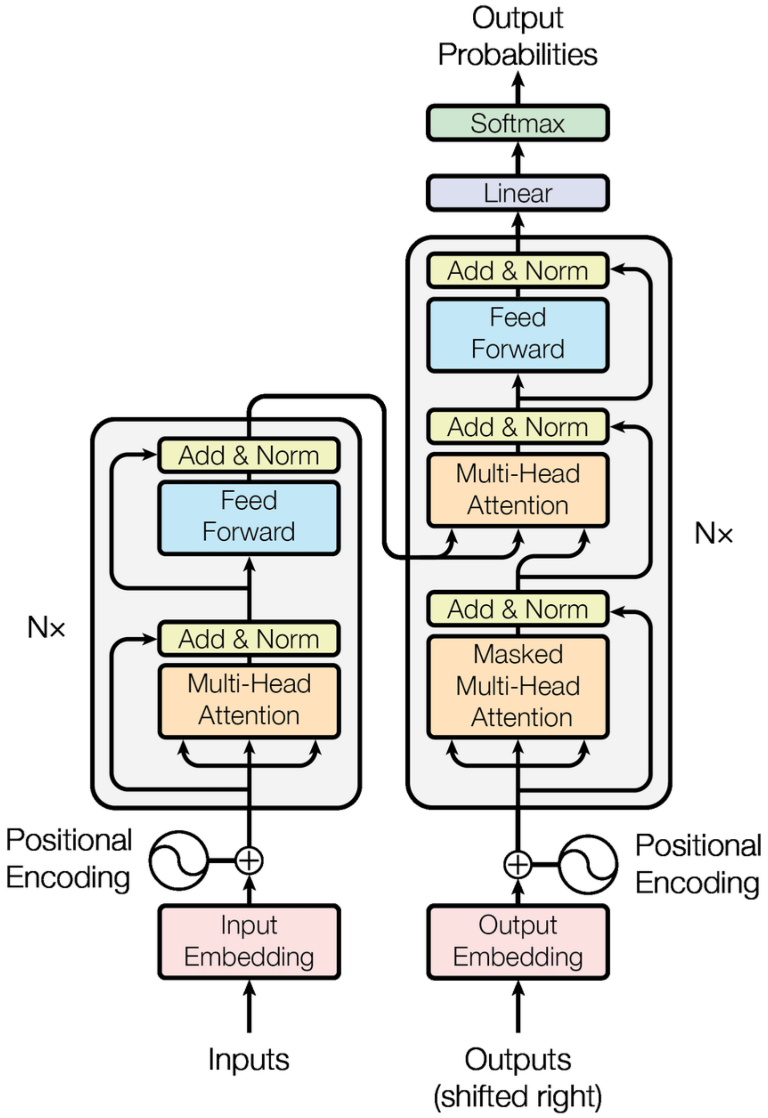


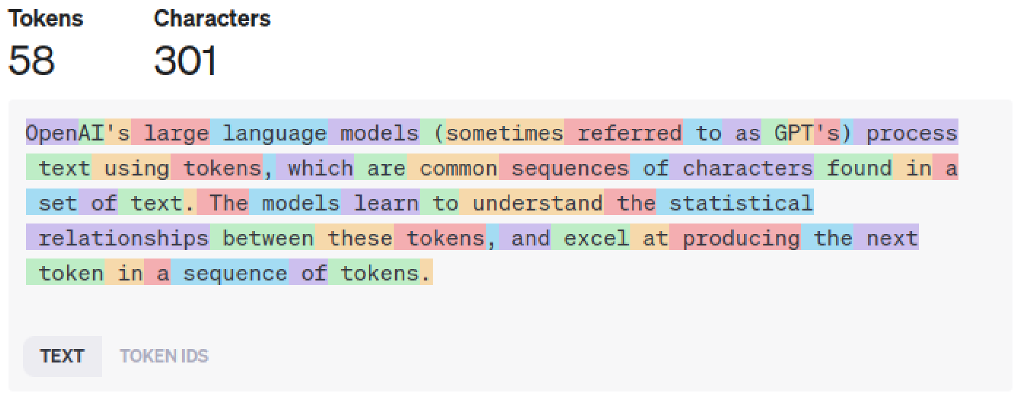
Figure 2.2: The Transformer Block Architecture, from Vaswani *et al.* [1]. [colocar a figura mais pequena para não haver uma quebra de página tão grande]

Pre-training

Learning the patterns and relationships among the data starts with the pre-training process. 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.* [14], 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 [14]. 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 carried out and explained by OpenAI.



Tokens 58

Characters 301

OpenAI's large language models (sometimes referred to as GPT's) process text using tokens, which are common sequences of characters found in a set of text. The models learn to understand the statistical relationships between these tokens, and excel at producing the next token in a sequence of tokens.

Figure 2.3: Tokenization process visually explained by OpenAI [16].

After the pre-training process, the LLM goes through a fine-tuning phase.

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 [17] and Hadi *et al.* [14], 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.* [15] 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.* [15] are Prompt Tuning, Prefix Tuning, and Adapter Tuning.

The Prompt Tuning method integrates trainable prompt token embeddings as prefixes or freely within input token embeddings. During fine-tuning, only the parameters associated with these embeddings are updated for the specific task. At the same time, the remaining model weights remain frozen, facilitating task adaptation without compromising pre-existing knowledge.

In Prefix Tuning, task-specific trainable prefix vectors are 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 [https://arxiv.org/abs/2106.09685]. 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.

2.2.3 Comparison between foundation Large Language Models

The best way to compare LLM is to evaluate the model's performance. Hadi *et al.* [14] 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.* [18]). 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.* [19]). 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.* [20]). 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.* [21]). 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.

2.2.4 Limitations

It is safe to say that LLM are significantly impacting the world. According to Liu *et al.* [12], 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.* [14] 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. Hadi *et al.* [14] 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. 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. This is called the Jail Break attack. Another attack is Data Poisoning Attacks, which consist of data poisoning strategies to manipulate the model's output.

Furthermore, Liu *et al.* [12] 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 Virtual Assistant

Conversational Agents, 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, improve service with every interaction. Because of this, the utility of the chatbots as tools is increasing as the technology advances.

The rise of conversational virtual assistants is underpinned by a convergence of technologies, specifically by LLM.

In this section, I explored the implementation of chatbots, their components, and existing tools.

2.3.1 Overview of Conversational Virtual Assistants

Borah *et al.* [22] stated that the process of response generation is not the same for all chatbots and distinguished them into four models: Retrieval-Based, Generative-Based, Long and Short Conversation, and Open or Close Domain. Chizhik and Zherebtsova [5] are in line with Borah *et al.* [22], and divided chatbots into three categories based on response generation architectures: rule-based chatbots, retrieval-based chatbots, and generative-based chatbots.

According to Chizhik and Zherebtsova [5], a rule-based chatbot examines fundamental features of the user's input statement and generates a response based on a predetermined set of manually crafted templates.

Conforming to Chizhik and Zherebtsova [5] and Borah *et al.* [22], 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 domain since they are usually applied to one domain, and in terms of errors, because, for example, the user cannot make grammatical mistakes.

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 Machine Translation 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. However, this could be complex and costly to implement.

Beyond that, Borah *et al.* [22] compare long and short conversations. He concluded that a more extended conversation requires saving what has been said, which makes it challenging to automate, unlike a short conversation.

Also, Borah *et al.* [22] 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. Crafting meaningful responses in such contexts demands an extensive breadth of world knowledge spanning countless topics. Conversely, closed-domain systems have more constrained inputs and outputs, focusing on specific goals. Consequently, many chatbots are inherently closed-domain, designed with a clear objective.

2.3.2 Generative-Based Chatbot

Generative-based conversational virtual assistants are chatbots that use generative models to generate natural language responses. These chatbots utilize sophisticated deep-learning techniques, such as LLM as a NLP engine to understand the user query and formulate a response in context. The LLM has been widely used in the new chatbots.

There are some methods to improve the results of a LLM used on chatbot: fine-tuning the LLM, Retrieval-Augmented Generation (RAG), and Prompt Engineering.

		Model Adaptation Required	
		Low	High
External Knowledge Required	Low	Prompt Engineering	Fine-tuning
	High	Retrieval-Augmented Generation (RAG)	All of the methods

Table 2.2: Optimization methods of generative-based chatbots.

The table 2.2 shows how these methods influence model adaptation and external knowledge. Mixing all methods requires a high model adaptation and external knowledge, but provides better results.

The following sections demonstrate how Prompt Engineering and RAG can optimize LLM.

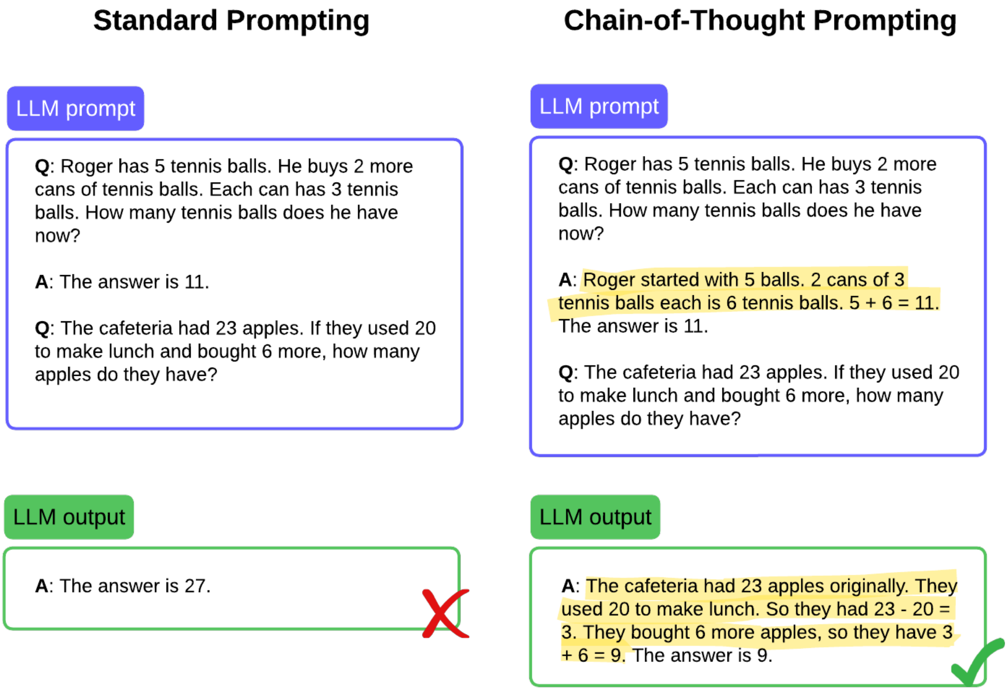
2.3.3 Prompt Engineering

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 [23], 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.* [24] 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.* [25]. 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.



Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

A: The answer is 27.

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had 23 - 20 = 3. They bought 6 more apples, so they have 3 + 6 = 9. The answer is 9.

Figure 2.4: Example of Chain-of-thought prompting. The reasoning processes are highlighted in yellow.

Meskó [23] 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.

2.3.4 Retrieval-Augmented Generation (RAG)

Gao *et al.* [26] made a survey into RAG systems and distiguish 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 challenges in retaining all the knowledge during training, and the inability to dynamically update model parameters, leading to outdatedness and hallucination. 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 Gao *et al.* [26], RAG involves retrieving pertinent information from external knowledge bases, giving more context to the LLM. This approach was introduced by Lewis *et al.* [27] in 2020 and enabled LLM to access and utilize up-to-date or domain-specific information to enhance response accuracy and relevance while reducing hallucinations.

Workflow of a RAG system

The figure 2.5 shows the workflow of a RAG. This model aims to retrieve pertinent information from an extensive corpus of documents when answering questions or generating text, subsequently utilizing this information to enhance the quality of predictions in compliance with Lewis *et al.* [27].

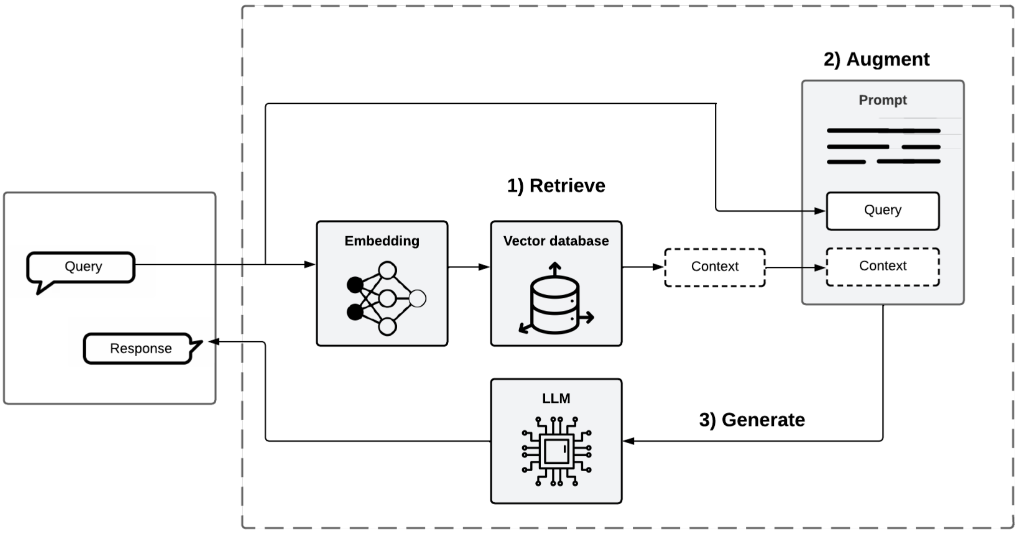


Figure 2.5: Retrieval-Augmented Generation Workflow.

Gao *et al.* [26] 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. According to Gao *et al.* [26], this step utilizes Encoding 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 from external data sources. The last step is 3) Generation. Using the prompt with the retrieved context, the LLM generates a response to the query.

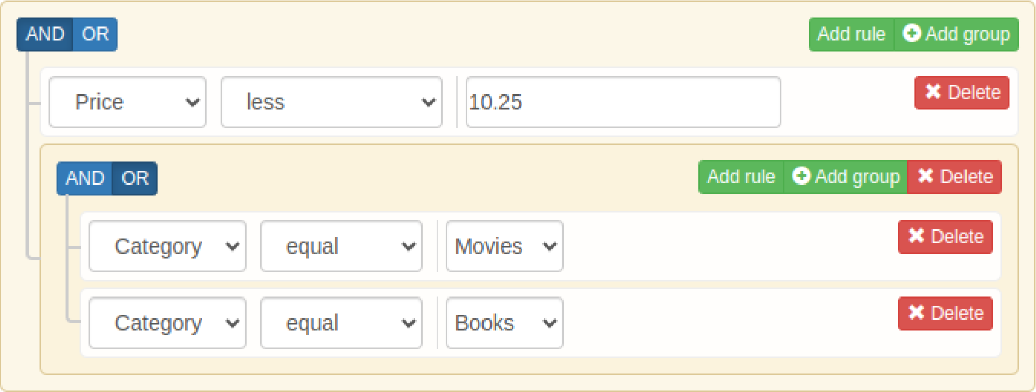
2.4 Interactive Query Builder

A query builder is a user interface tool for dynamically searching and filtering database objects, constructing a query according to user preferences. This query could be in different formats, such as SQL, JSON, and MongoDB. This tool lets users construct queries visually, eliminating manual research or coding.

It is important to note that there aren't many cases of conversational query builder documentation, so I will document how general query builders work and explain more deeply how the ATLAS cohort definition works.

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 [28].



Price less 10.25

Category equal Movies

Category equal Books

Figure 2.6: Query builder from jQuery QueryBuilder [28].

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.

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

There are several advantages of its use: 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 [29], 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 [29]: 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 [29], 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 a 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.

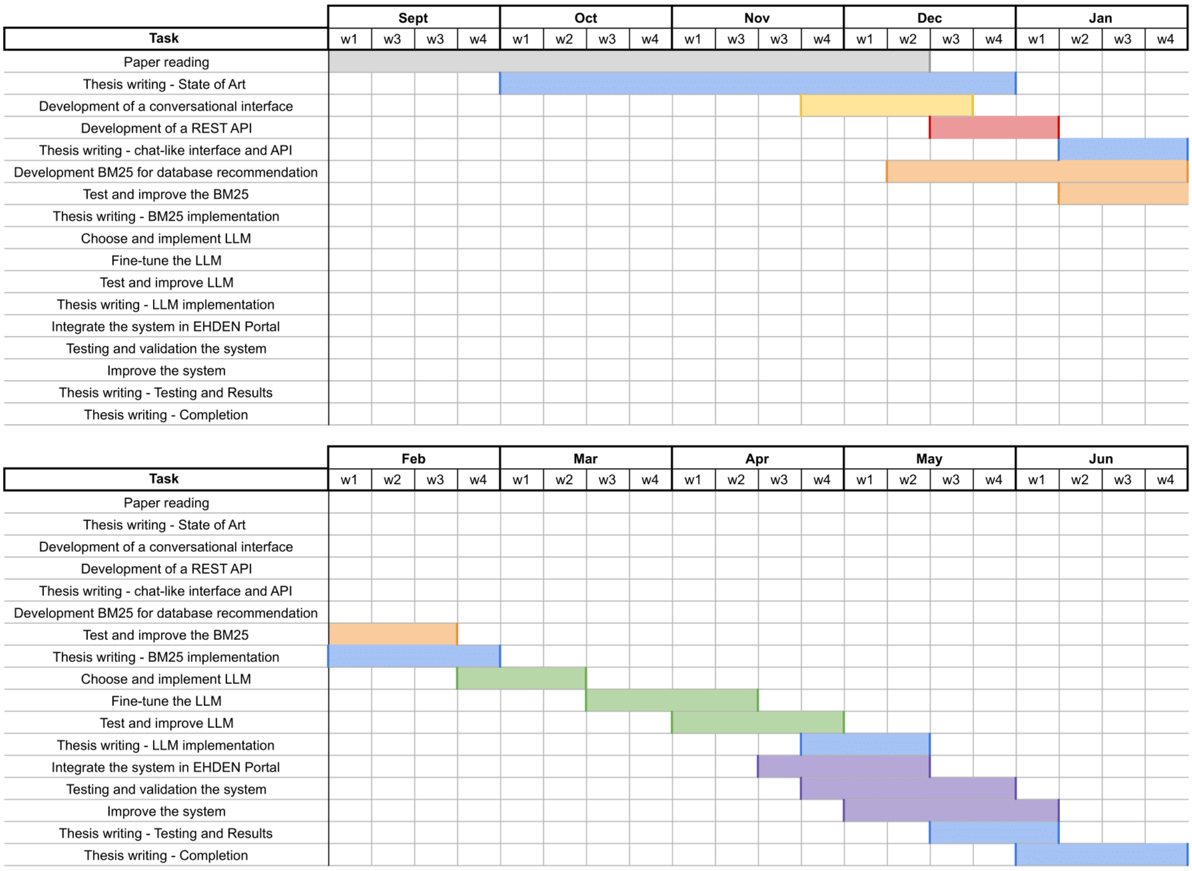
ATLAS by OHDSI provides a cohort definition, which is a query builder to define groups of people based on the research question using healthcare claims data. However, the interface revealed not very user-friendly and intuitive because it requires the user to have a good knowledge of its use and important concepts. Therefore, a chatbot that builds a cohort definition improves the user experience by making it more intuitive and autonomous.

CHAPTER 3

Methodology

For this thesis work, after an extensive literature review on crucial concepts, such as IR and LLM, we are ready to achieve the primary goal of this work.

The Gantt diagram, figure 3.1, shows the work proposal for the following months. This diagram is divided into 2 parts: the first refers to the work of the first semester, and the second to the work proposed for the second semester.



	Sept				Oct				Nov				Dec				Jan			
Task	w1	w3	w3	w4	w1	w2	w3	w4	w1	w3	w3	w4	w1	w2	w3	w4	w1	w2	w3	w4
Paper reading	■	■	■	■	■	■	■	■	■	■	■	■	■	■						
Thesis writing - State of Art					■	■	■	■	■	■	■	■	■	■	■	■				
Development of a conversational interface												■	■	■	■					
Development of a REST API															■	■	■			
Thesis writing - chat-like interface and API																		■	■	■
Development BM25 for database recommendation														■	■	■	■	■	■	■
Test and improve the BM25																		■	■	■
Thesis writing - BM25 implementation																				
Choose and implement LLM																				
Fine-tune the LLM																				
Test and improve LLM																				
Thesis writing - LLM implementation																				
Integrate the system in EHDEN Portal																				
Testing and validation the system																				
Improve the system																				
Thesis writing - Testing and Results																				
Thesis writing - Completion																				

	Feb				Mar				Apr				May				Jun			
Task	w1	w2	w3	w4	w1	w2	w3	w4	w1	w2	w3	w4	w1	w2	w3	w4	w1	w2	w3	w4
Paper reading																				
Thesis writing - State of Art																				
Development of a conversational interface																				
Development of a REST API																				
Thesis writing - chat-like interface and API																				
Development BM25 for database recommendation																				
Test and improve the BM25	■	■	■																	
Thesis writing - BM25 implementation	■	■	■	■																
Choose and implement LLM				■	■	■														
Fine-tune the LLM							■	■	■	■										
Test and improve LLM									■	■	■	■								
Thesis writing - LLM implementation												■	■	■						
Integrate the system in EHDEN Portal											■	■	■	■						
Testing and validation the system												■	■	■	■	■				
Improve the system													■	■	■	■	■			
Thesis writing - Testing and Results															■	■	■			
Thesis writing - Completion																	■	■	■	■

Figure 3.1: Proposal thesis work in a Gantt Diagram.

3.1 Work Done

As the figure 3.1 shows, this semester's work was focused on state-of-the-art understanding and writing, going through a process of research and reading articles. The state-of-the-art objectives established in chapter 1 have been completed. These objectives are to study methodologies to build a conversational user interface, procedures to retrieve the databases of most interest and explore the definition of a study protocol. It was essential not only to learn more about the actual state of some technologies and concepts but also to get more into the project goal and requirements.

In terms of development, I have developed a prototype of a chat-like interface using NextJS, a well-known ReactJS framework. Also, I have developed a REST API in FastAPI, a modern web framework based on Python. The interface and the API communicate between them through HTTP requests.

Beyond this, inside the backend module, I have started implementing and testing a BM25 method to retrieve the best databases for a given user query. First, NLP methods are applied to understand the plain-text query. For now, the NLP method is very simple, because only tokenize and. Then, the BM25 score is calculated between the query and the concepts of which database, making a database ranking. To implement this IR technique, I am using the pyterrier library, and it could be changed to another library later if I discover another one more effective.

So, for now, the application developed can receive a user query through the frontend module and then, the backend module retrieves a rank of databases based on the query understanding.

It is important to note that the work done this semester will probably change. For example, in the figure 3.1, the frontend task is given as closed, but it will be constantly changing. This application has room for improvement and optimization in all aspects, which will be addressed in future work.

3.2 Future Work

For the following months, I will continue the development of the database recommendation, based on BM25, improving some aspects, like concept linking. Also, test and improve this IR implementation based on the testing results.

After this, I will choose, implement, and fine-tune a LLM in order to guide the conversation to build the final query with the cohort definition structure. By doing this, the conversation flows naturally. Finally, test and improve the model based on the testing results.

The next step is integrating this system in the EHDEN Portal and testing and validating the system.

The documentation of the implementation and the respective steps taken will be carried out throughout its development, as the 3.1 shows.

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] 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`.

[3] 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).

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

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

[6] 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).

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

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

[9] 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).

[10] 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).

[11] 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).

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

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

[14] 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`.

[15] 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).

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

[17] 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).

[18] 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).

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

[20] 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).

[21] 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).

[22] B. Borah, D. Pathak, P. Sarmah, B. Som, and S. Nandi, «Survey of Textbased Chatbot in Perspective of Recent Technologies», English, *Communications in Computer and Information Science*, vol. 1031, pp. 84–96, 2019, ISBN: 9789811385803, ISSN: 1865-0929. DOI: 10.1007/978-981-13-8581-0_7.

[23] 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).

[24] 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.

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

[26] 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).

[27] 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).

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

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