Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Excertos de Código

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

BERT . . . . . Bidirectional Encoder Representations from Transformers

GPT . . . . . Generative Pre-trained Transformer

ML . . . . . Machine Learning

RAG . . . . . Retrieval-Augmented Generation

CAPÍTULO 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, Networkdashboards has 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, namely: i) Information retrieval systems, ii) Large Language Models, iii) Generative-based conversational virtual agents, and iv) Interactive query builder;
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.

CAPÍTULO 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 will help researchers redefine their studies effectively. The conversational virtual assistant will return a query to help discover the best databases for 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 the LLM, a recent advance of algorithms to perform Natural Language Processing (NLP) tasks, 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, IR involves retrieving information from a database or multiple databases. According to P M et al. [3], the IR system requires users to input queries, retrieving pertinent information from the database that aligns with the users' needs. Thus, this prevents the user from accessing a massive amount of information.

In conformity with Hambarde e 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.

In turn, deep learning methods have evolved. Neural Network Architectures, transfer learning, and pre-training techniques emerged. These approaches have advanced the repre-

sentation 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 concentration on crucial query segments and documents for improved matching. Moreover, incorporating pre-trained 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 of the semantics and context within natural language queries and documents.

This field has many applications in the real world. P M et al. [3] highlights the following: streamlined and adaptable indexing and retrieval, information extraction, semantic matching, and multimedia retrieval. 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 NLP.

2.1.1 Overview of Information Retrieval Systems

According to Hambarde e Proença [4], an IR system can be separated into two stages: Retrieval and Ranker. The following figure, adapted from Hambarde e Proença [4], shows us an overview of modern IR systems, highlighting the two main stages.

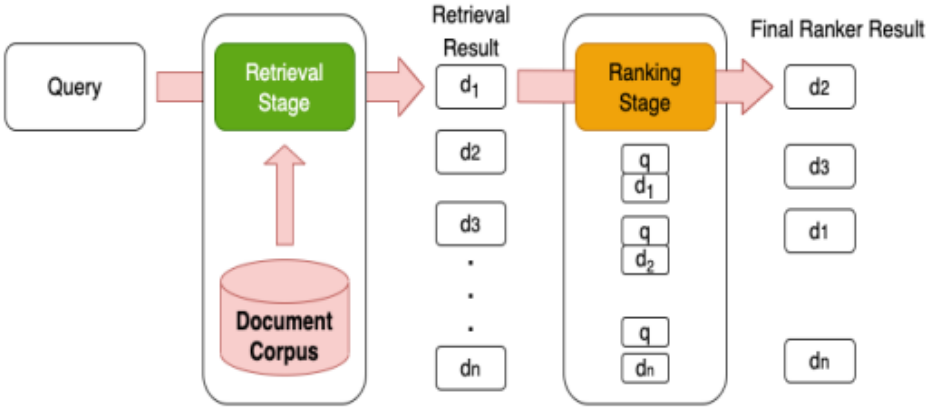

Figura 2.1: Overview of modern IR system [REFAZER IMAGEM]

After analyzing the query, the retrieval stage will select an initial set of documents that are potentially pertinent to the query. Subsequently, the relevance of these documents undergoes reassessment through the similarity scores. The ranking is then refined using diverse algorithms and models, including the vector space model, Latent Semantic Indexing, Latent Dirichlet Allocation, and pre-trained models such as BERT.

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. 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.

2.1.2 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. [5] 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. [5] 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. [5] and Chizhik e Zherebtsova [6] 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.

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

Figura 2.2: Overview of modern IR system [REFAZER IMAGEM]

Manning et al. [5] 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. It builds upon the TF-IDF model to provide more accurate and context-aware document ranking. Hambarde e 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.

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

Figura 2.3: BM25 equation

As Gomede [7] 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 [https://www.elastic.co/blog/practical-bm25-part-2-the-bm25-algorithm-and-its-variables] 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 vast 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. Another limitation is the ignorance of other crucial factors to get a better search beyond the factors relative to the TF and DL.

2.1.3 IR in Question Answering

Question Answering (QA) and IR are closely related fields, together with NLP. According to Zhong et al. [8], QA aims to give users accurate and prompt responses to posed questions.

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 convolutional neural networks (CNN), 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 recogni-

tion. Zhong et al. [8] highlights the tool 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 Artificial Intelligence (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. [9]

The **NLU** component plays a crucial role in processing and transforming unstructured data into a format the system can comprehend seamlessly. Essentially, the function of NLU is to identify topics and entities, identify the intention, and determine the structure and syntax of the sentence.

In agreement with Ngai et al. [10], for easy understanding by the chatbot, the user queries can be processed by semantic analysis, pragmatic analysis, and syntactic analysis. Ayanouz et al. [9] 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 e Zherebtsova [6] 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 LLM. Liu et al. [11] explained this in a simple and intuitive way.

Before LLM, there were only simple Language Models (LM), which, in the initial stage, utilized a fully supervised learning approach, where task-specific models were trained on the target task dataset exclusively. Most of these were predictive models 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, in 2017, the launch of the self-attention mechanism revolutionized this field, and the Transformer architectures have become increasingly popular. 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. [11] 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, explore briefily their architectures

2.2.1 Definition

LLM are revolutionizing NLP and AI research. LLM are an AI created to comprehend, generate, and engage in human language interactions. Essentially, these advanced AI systems can 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 great accuracy due to their significant model parameters, often comprising hundreds of billions of parameters. Hadi et al. [12] 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. [13], when provided with task descriptions and examples through prompts, LLM can produce textual responses to task queries. So, we can put LLM in the generative AI field.

Liu et al. [11] say that the release of ChatGPT 1 garnered significant social attention, and research into LLM has evolved. 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.

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 self-attention mechanism and encoder-decoder structures. This new technology enables parallelization and efficient handling of long-range dependencies, according to Hadi et al. [12], and led to the development of models that have achieved enormous results, such as GPT by OpenAI and BERT by Google.

The innovation of this model is due to the 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.

Another key component is the Encoder and Decoder Stacks. Essentially, the encoder processes the input sequence, and the decoder generates the output sequence. Each stack contains six (6) similar layers, and these layers apply the attention mechanism.

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.

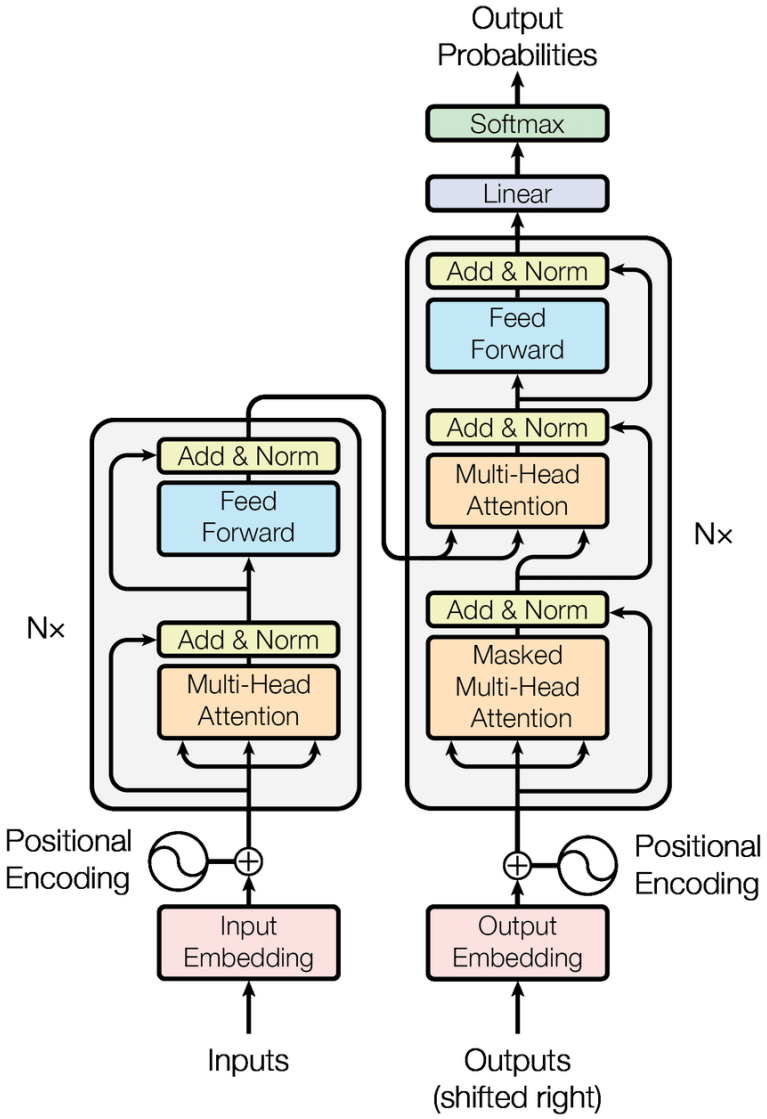

Figura 2.4: The Transformer architecture. From Vaswani et al. [1]

Pre-training

Hadi et al. [12]

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.

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, superfluous, and potentially harmful content from the massive text data. Then, according to Hadi et al. [12], duplicate data in a corpus make less diversity of LLM. So, the duplication of data will make the training process unstable, impacting the overall performance of the model. 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. 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.

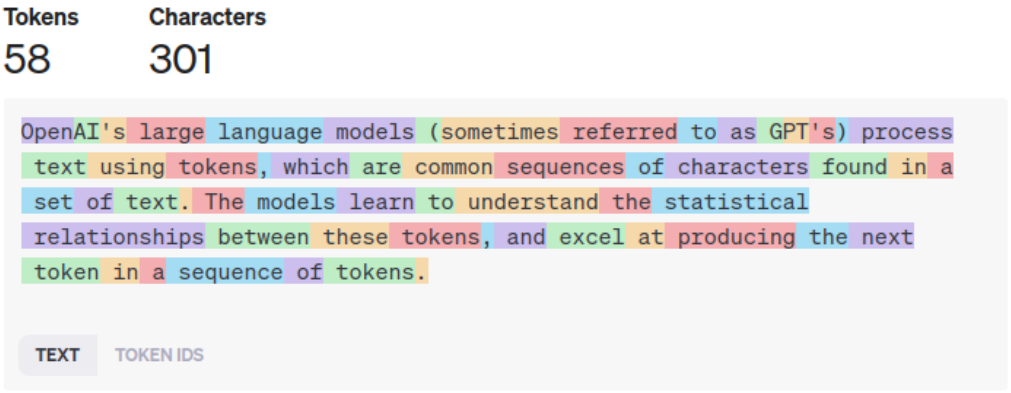

Figura 2.5: Tokenization process visually explained by OpenAI [14]

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 [15] and Hadi et al. [12], 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. [13] distinguishes some of them, such as Parameter-Efficient Tuning. As LLM typically requires a lot of computational resources, like memory and computing, this 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. [13] 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, an encoder-decoder module is incorporated with the attention and feed-forward layers within the transformer block. The fine-tuning process selectively targets these layers, leaving the remainder of the model frozen to preserve existing knowledge.

2.2.3 Comparison between main models

The best way to compare LLM is to evaluate the model's performance, conforming to Hadi et al. [12]. Hadi et al. [12] 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 records 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. [16]). 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 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. [17]). 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. [18]). 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. [19]). There are three variations of the model size: 7 billion, 40 billion, and the most recent, 180 billion of parameters. This Falcon 180B, equipped with an impressive 180 billion parameters and trained on 3.5 trillion tokens, currently leads the Hugging Face Leaderboard for pre-trained Open Large Language Models. It is accessible for both research and commercial purposes.

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

Tabela 2.1: Comparison of the main LLM

2.2.4 Limitations

It is safe to say that LLM are significantly impacting the world. According to Liu et al. [11], 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. [12] 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. [12] 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 in data poisoning strategies to manipulate the model's output.

Furthermore, Liu et al. [11] 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, including NLP, Machine Learning (ML), and AI.

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

2.3.1 Overview of Conversational Virtual Assistants

Nuruzzaman e Hussain [20] separated chatbot applications based on their main features and functionalities. There are four chatbot models: goal-based, knowledge-based, service-based, and response generated-based. Goal-based chatbots are designed for particular tasks and structured to engage in concise conversations to collect user information for task completion.

Borah et al. [21] said 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 e Zherebtsova [6] are in line with Borah et al. [21], and divided chatbots into three categories based on response generation architectures: rule-based chatbots, retrieval-based chatbots, and generative-based chatbots.

According to Chizhik e Zherebtsova [6], 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 e Zherebtsova [6] and Borah et al. [21], 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. [21] 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. [21] 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 Common System Architectures

Most chatbot implementations apply standard components, such as NLP, dialogue, knowledge, and data storage modules. According to Ngai et al. [10] and to Cem [22], the function of each module can be summarized as follows.

The dialogue module is in charge of handling the conversation flow. To effectively communicate with the user, conversation agents must understand the human language using a NLP Engine to decide what to do with the intention found.

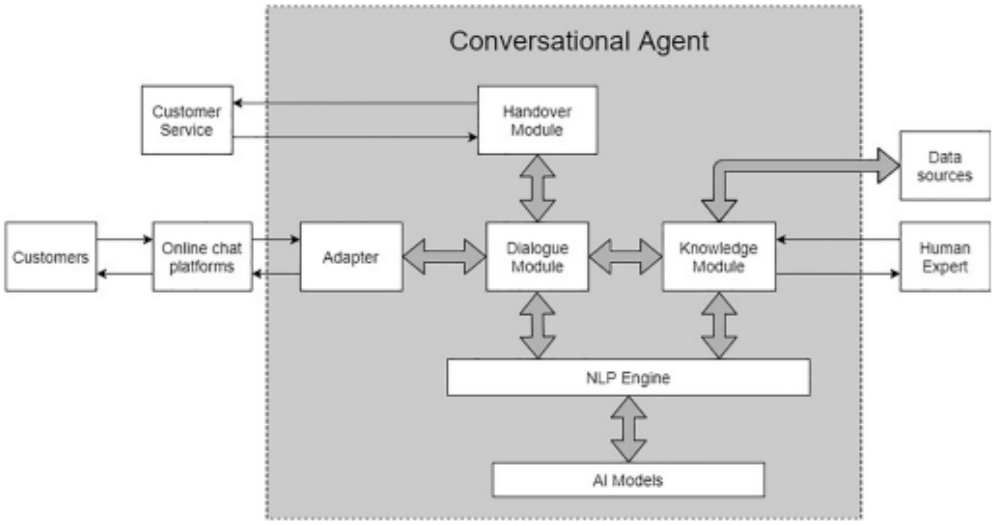

Figura 2.6: [REFAZER IMAGEM] Overview of a conversational system architecture From [An intelligent knowledge-based chatbot for customer service]

The knowledge module is the source of data and knowledge of the chatbot. After knowing the user's intention, this module will retrieve the data to respond to the user.

Many chatbots use a knowledge base for the knowledge module. In compliance with Pereira et al. [23], this choice is justified because it is advantageous to have the data organized semantically. This establishes a coherent structure for structured and unstructured data, simplifying the deduction of new knowledge.

Additionally, some chatbots are integrated with web scrapers to pull data from online resources and display it to users

2.3.3 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, RAG, and Prompt Engineering.

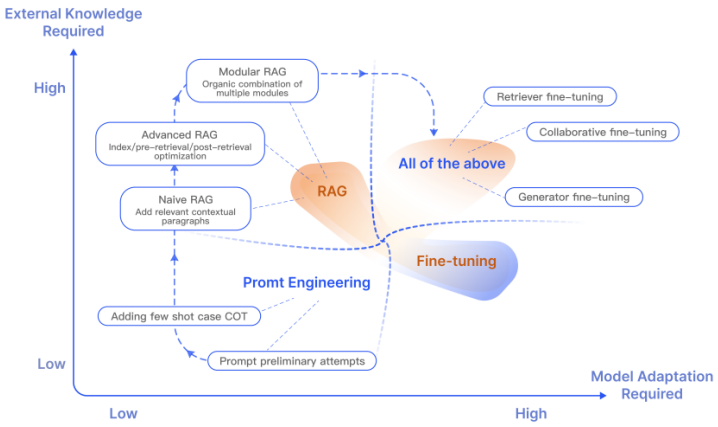

Figura 2.7: [REFAZER IMAGEM] optimization methods of generative-based chatbots. Adapted from [Retrieval-Augmented Generation for Large Language Models: A Survey]

The figure above 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.

2.3.4 Prompt Engineering

With the emergence of new technologies, such as LLM, other research fields were born. Prompt Engineering is one of these cases and has been widely applied. In compliance with Meskó [24], this emerging field involves designing, refining, and implementing prompts or instructions to direct the output of LLM, aiding in diverse tasks. LLM can follow specific directions provided by users in natural language after being tuned with instructions,

Ma et al. [2] noted that research demonstrates that thoughtful prompts enhance LLM performance across reasoning and planning tasks. This improvement is achieved through intermediate representations involving thoughts, decomposition, search-decomposition mix, structure, abstraction, and optimization.

Also, Ma et al. [2] find with their work that prompt engineering has many benefits for the user when it is well implemented because prompt-based methods assist humans in forming mental models for problem-solving. Also, they conclude that task decomposition enables the breakdown of a complex task into specific sub-tasks, each matched with readily available tools suitable for its completion. This method of prompting engineering had very successful results.

```
I want to accomplish the main goal of: {text}
To better assist me, please break down the problem into sub-problems.
Each sub-problem should help me to solve the original problem.
Make it so that each sub-problem is not trivial and can be helpful.
Take my context and personalization cues to personalize the sub-problems.
Make sure each sub-problem is concise and less than 15 words.

Personalization Cue: {selected_options}
My Context: {user_context}

Output format (make sure only output a valid JSON object that can be parsed with javascript function JSON.
    parse).
Do not include any ``` or json'.
{{
  "sub_problems": A list of strings (max 8), each as a valid sub-query
}}

Output:
```

Figura 2.8: [REFAZER IMAGEM] Example of a prompt that applies task decomposition From Ma et al. [2]

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.

2.3.5 Retrieval-Augmented Generation (RAG)

Gao et al. [25] 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, 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. [25], RAG involves retrieving pertinent information from external knowledge bases, giving more context to the LLM. This approach was introduced by Lewis et al. [26] 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 following figure 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. [26].

Gao et al. [25] explain in a simple way the workflow. The first step is 1) Retrieve information from an external data source, such as a vector database, as in the example of the figure. According to Gao et al. [25], this step utilizes Encoding Models, such as BM25, to

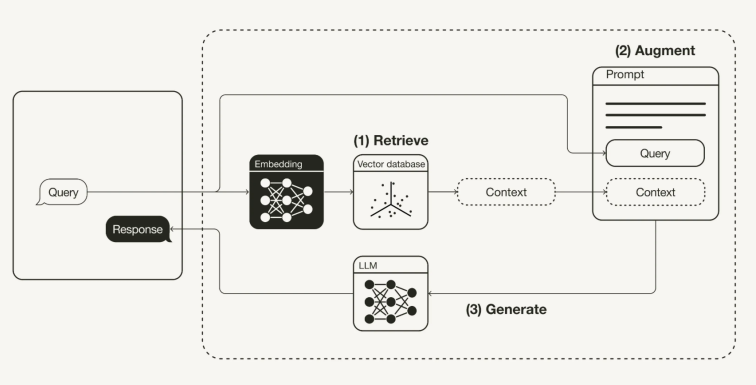

Figura 2.9: [REFAZER IMAGEM] RAG workflow. Adapted from [https://towardsdatascience.com/retrieval-augmented-generation-rag-from-theory-to-langchain-implementation-4e9bd5f6a4f2]

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 [Domain Specific Query Generation from Natural Language Text], JSON, and MongoDB [https://react-querybuilder.js.org/docs/utils/export]. This tool lets users construct queries visually, eliminating manual research or coding.

It is important to note that there aren't many cases of query builder documentation.

2.4.1 General Query Builder

Users interact with the query builder through a user-friendly interface. The following figure shows a query builder interface from jQuery QueryBuilder [https://querybuilder.js.org/].

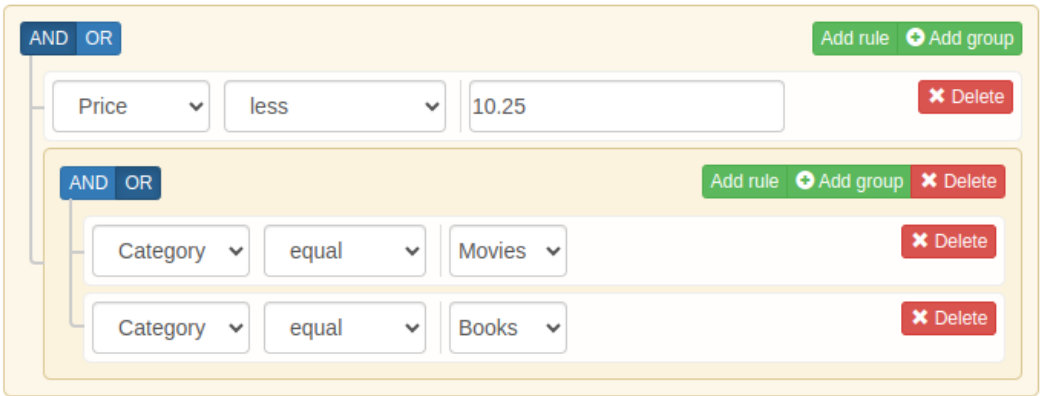

Figura 2.10: Query builder from jQuery QueryBuilder [https://querybuilder.js.org/]

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 the figure as 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 Query

[ainda em desenvolvimento]

2.5 Summary

[ainda em desenvolvimento]

* a junção das áreas NLP com IR parece bastante promissora

* apesar de poucas informação sobre query builders, é possivel reunir e estruturar conceitos-chave para o seu desenvolvimento

* A área de NLP e llm está em constante atualização atualmente

* tornar a criação de queries mais intuitiva do atlas

CAPÍTULO 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 above shows the work proposal for the following months.

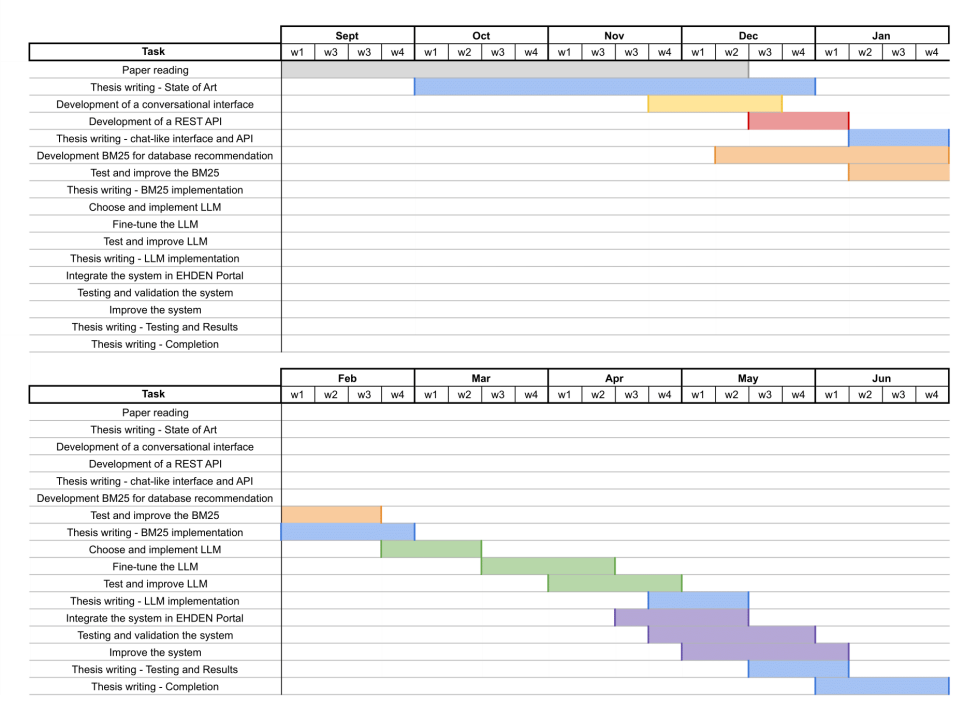

Figura 3.1: Proposal thesis work in Gantt Diagram

This semester's work focused on the State of Art understanding and writing, going through a process of research and reading articles. 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. Then, the BM25 score is calculated between the query and the concepts of which database, making a database ranking.

For the following months, I will continue the development of the database recommendation, based on BM25, improving some aspects, like concept linking. 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 correct 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 Gantt Diagram shows.

Referências

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

[2] X. Ma, S. Mishra, A. Liu et al., *Beyond ChatBots: ExploreLLM for Structured Thoughts and Personalized Model Responses*, arXiv:2312.00763 [cs], dez. de 2023. (acedido em 21/12/2023).

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

[4] K. A. Hambarde e 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. (acedido em 06/12/2023).

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

[6] A. Chizhik e Y. Zherebtsova, «Challenges of Building an Intelligent Chatbot,» 2020. (acedido em 13/11/2023).

[7] E. Gomede, *Understanding the BM25 Ranking Algorithm*, en, set. de 2023. URL: https://medium.com/@evertongomede/understanding-the-bm25-ranking-algorithm-19f6d45c6ce (acedido em 06/12/2023).

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

[9] S. Ayanouz, B. A. Abdelhakim e M. Benhmed, «A Smart Chatbot Architecture based NLP and Machine Learning for Health Care Assistance,» em *Proceedings of the 3rd International Conference on Networking, Information Systems & Security*, sér. NISS2020, New York, NY, USA: Association for Computing Machinery, 2020, pp. 1–6, ISBN: 978-1-4503-7634-1. DOI: 10.1145/3386723.3387897. (acedido em 13/11/2023).

[10] E. W. T. Ngai, M. C. M. Lee, M. Luo, P. S. L. Chan e T. Liang, «An intelligent knowledge-based chatbot for customer service,» *Electronic Commerce Research and Applications*, vol. 50, p. 101 098, nov. de 2021, ISSN: 1567-4223. DOI: 10.1016/j.elerap.2021.101098. (acedido em 13/11/2023).

[11] X. Liu, J. Wang, J. Sun et al., «Prompting Frameworks for Large Language Models: A Survey,» en, *ACM Comput. Surv.*, vol. 1, n.º 1,

[12] 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. de 2023. DOI: 10.36227/techrxiv.23589741.

[13] H. Naveed, A. U. Khan, S. Qiu et al., *A Comprehensive Overview of Large Language Models*, arXiv:2307.06435 [cs], nov. de 2023. (acedido em 15/12/2023).

[14] *OpenAI Platform*, en. URL: https://platform.openai.com (acedido em 15/12/2023).

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

[16] H. Touvron, L. Martin, K. Stone et al., *Llama 2: Open Foundation and Fine-Tuned Chat Models*, jul. de 2023. DOI: 10.48550/arXiv.2307.09288. (acedido em 06/01/2024).

[17] OpenAI, J. Achiam, S. Adler et al., *GPT-4 Technical Report*, dez. de 2023. DOI: 10.48550/arXiv.2303.08774. (acedido em 06/01/2024).

[18] R. Anil, A. M. Dai, O. Firat et al., *PaLM 2 Technical Report*, set. de 2023. DOI: 10.48550/arXiv.2305.10403. (acedido em 06/01/2024).

[19] E. Almazrouei, H. Alobeidli, A. Alshamsi et al., *The Falcon Series of Open Language Models*, nov. de 2023. DOI: 10.48550/arXiv.2311.16867. (acedido em 06/01/2024).

[20] M. Nuruzzaman e O. K. Hussain, «A Survey on Chatbot Implementation in Customer Service Industry through Deep Neural Networks,» em *2018 IEEE 15th International Conference on e-Business Engineering (ICEBE)*, out. de 2018, pp. 54–61. DOI: 10.1109/ICEBE.2018.00019. (acedido em 13/11/2023).

[21] B. Borah, D. Pathak, P. Sarmah, B. Som e 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.

[22] D. Cem, *How to Build a Chatbot: Components & Architecture in 2023*, en-US. (acedido em 13/11/2023).

[23] A. Pereira, J. Almeida, R. Lopes e J. Oliveira, «Querying semantic catalogues of biomedical databases,» English, *Journal of Biomedical Informatics*, vol. 137, 2023, ISSN: 1532-0464. DOI: 10.1016/j.jbi.2022.104272.

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

[25] Y. Gao, Y. Xiong, X. Gao et al., *Retrieval-Augmented Generation for Large Language Models: A Survey*, arXiv:2312.10997 [cs], dez. de 2023. DOI: 10.48550/arXiv.2312.10997. (acedido em 24/12/2023).

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