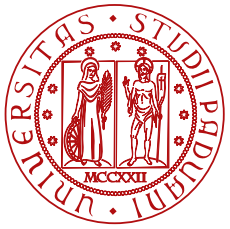

Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Department of information engineering

Master Thesis in Computer Engineering

Development and Optimization of a Retrieval Augmented Generation System for Enhanced Conversational AI Assistance

Master Candidate

Mauro Mickel

Student ID 2062721

Supervisor

Prof. Loris Nanni

University of Padova

Academic Year
2023/2024
Date: 16/10/2024

To my family
and friends

Abstract

For a considerable time, chatbots have acted as helpful means to assist users in retrieving information. However, with the advancement of generative language models, it has become possible to elevate these bots; they are now able to understand user needs and respond to them. This thesis explains the creation of a platform for building a system which integrates individual approaches to the construction of chatbots, developed during the internship in TownHall Reply S.P.A. This research aims to investigate generative language networks incorporated in a chatbot system to elevate user interaction and satisfaction by presenting a more relevant and contextual response. The analysis extensively covers all the steps involved in creating the chatbot, subjecting it to tests specifically designed for this purpose. The central objective of the research is to thoroughly examine the challenges the system must face, with a particular focus on responding in a complete and accurate manner. The proposed solutions employ various methodologies, including the use of different databases, testing multiple embedding models to generate suitable vector spaces, the application of RAG and Reranking to achieve more precise results, and leveraging Large Language Models (LLMs) to formulate appropriate human-like responses, as well as the importance of prompt engineering. Each of these aspects constitutes a key discussion point, where tests are conducted to find the optimal setup, considering time, resources, and accuracy. The results contribute to a deeper understanding of potential issues in the fields of Natural Language Processing and Deep Learning, while also laying the groundwork for future advancements in this increasingly utilized domain.

Sommario

Per un periodo considerevole, i chatbot hanno agito come strumenti utili per assistere gli utenti nel recupero di informazioni. Tuttavia, con l'avanzamento dei modelli linguistici generativi, è diventato possibile migliorare questi bot; ora sono in grado di comprendere le esigenze degli utenti e rispondere ad esse. Questa tesi spiega la creazione di una piattaforma per la costruzione di un sistema che integri approcci individuali alla progettazione di chatbot, sviluppata durante il tirocinio formativo universitario presso l'azienda TownHall Reply S.P.A. Questa ricerca mira a investigare i large language models incorporati in un sistema di chatbot per migliorare l'interazione e la soddisfazione degli utenti, presentando risposte più pertinenti e contestuali. L'analisi approfondisce ampiamente tutti gli step per la creazione del chatbot, sottoponendolo a test ideati ad hoc. L'obiettivo centrale della ricerca è testare in maniera completa le sfide che il sistema deve affrontare, con particolare attenzione al rispondere in maniera completa e corretta. Le soluzioni proposte utilizzano diverse metodologie, tra cui lo sfruttamento di diversi database, il testare diversi modelli di embedding per generare spazi vettoriali adatti al contesto, l'utilizzo di RAG e Reranking per ottenere risultati più precisi e nell'utilizzo di LLM per formulare risposte adeguate nel linguaggio umano, focalizzando l'attenzione sul prompt engineering che risulta un tassello essenziale per il corretto funzionamento. Ognuno di questi aspetti costituisce un elemento di discussione, in cui vengono eseguiti test per cercare il miglior setup possibile tenendo in considerazione tempo, risorse e precisione. I risultati contribuiscono alla comprensione di potenziali problemi nel campo del Natural Language Processing e del Deep Learning e allo stesso tempo creano le basi per evoluzioni future in questo campo sempre più usato.

Contents

List of Figures

List of Tables

List of Algorithms

List of Code Snippets

List of Acronyms

LLM Large Language Model

NLP Natural Language Processing

RAG Retrieval Augmented Generation

IR Information Retrieval

ANN Approximate Nearest Neighbor

DB Database

AI Artificial Intelligence

ML Machine Learning

HNSW Hierarchical Navigable Small World

RDMS Relational Database Management System

DPR Dense Passage Retrieval

1 Introduction

1.1 Motivations

The landscape of contemporary artificial intelligence is advancing and changing with transformative technologies, including the development of new generative technologies, at a rapid pace. Key players in this transformation are Large Language Models, because they have ushered in a new period of text generation and communication. As artificial intelligence is more easily incorporated into our daily lives, we exist in a landscape of profound change in terms of human-machine interaction, shifting to developing a relationship that is more harmonious and symbiotic. Signs of this transformation are apparent in a variety of work activities and daily tasks that convey an increasing reliance on intelligent digital tools. The rapid evolution of AI technologies, particularly in regard to their capabilities for understanding human language and generating relevant and insightful responses can serve as markers of our reliance on human-like digital being altering our technology in transformative ways.

1.2 Idea

Given these considerations, this thesis focuses on the study of the development process of an intelligent chatbot system, which is aimed at responding to users' questions in a more effective and relevant manner. Some of the factors include technical advancements, such as artificial intelligence and deep learning

which have recently created new avenues for the design of conversational interfaces. The main goal of this work is exactly to use such technologies to build a conversational agent, which is smooth and natural in users request interpretation.

A system of this sort will not only assist the users in locating the answers that they are searching for, but it will also serve the purpose of improving peoples experiences by trying to make their interaction with the gadgets less complicated and more satisfying. The chatbot endeavors to build a rapport with the users by offering fitting responses to their requests thus boosting the likelihood of the platforms usage. This work thus aims at the development and practical application of methodologies related to the design and development of such intelligent chatbots, especially focusing on the challenges and opportunities posed by current advances in technology.

1.3 Hystory of Chatbots

The domain of operations of an early interaction with machines began with the 1960s, with the focus on simulation of interventions in channels of ordinary human interactions and repetitive tasks. One of the early efforts was ELIZA, developed in 1966. The program implemented relatively simple keyword-oriented rules in dialogues. While ELIZA was limited, it rather successfully argued the possible existence of conversations where a human could actually have an interaction with a machine through language, which opened avenues for further experimentation. For quite some following decades, however, chatbots were confined to static models with predefined answers, still without understanding true language.

The introduction of the internet commercialized chatbots in the 1990s to facilitate automated interfaces for customer support. These applications were, however, rather basic. While these bots were handy in facilitating alternate means of doing menial tasks, they could hardly become wise. While they were mostly useful in providing automatic answers to cut down on operational costs, they did not greatly help provide quick support in cases needing human intervention.

During the 2010s, a watershed occurred by mingling Artificial Intelligence (AI) and Machine Learning (ML). Usage of Natural Language Processing (NLP) turned things upside down as the AI-based chatbots could better comprehend

human language more effectively, better contextual understanding, and, thus, a sharper response. Another class of tools sprang into existence since that time, namely virtual assistants like Siri, Alexa, and Google Assistant. These provided the real sense of fluid-like conversation with machines through artificial intelligence in their own right; it was no longer just the option of answering a simple question; the tools could now perform complex activities in dialogue with many other device application, if any, while keeping on learning on any new query.

Recently, the emergence of advanced language models like GPT-3 and, more recently, GPT-4 has helped raise standards even higher. These models which are based on deep neural networks can produce complicated written texts, hold logical dialogues, and change in accordance with the context until they reach a point where users can converse with them naturally, without any difficulty. With the access to data and computing power came faster growth resulting in deploying chatbots in healthcare, education, entertainment, and marketing industries, with better interactions between human and machines.

Currently, chatbots can answer complex questions, produce creative writing work, and even predict users behavior due to predictive models. There is additional enhancement to their function by integration with other blends such as conversational AI making them essential in various fields. The capabilities of chatbots will dramatically enhance in the future, such that every person can expect a free-flowing, orthographic as well as context-specific intelligent interaction. This could indeed revolutionize the way we do our work and how we communicate with one another.

1.4 Retrieval Augmented Generation

An innovative approach for enhancing the quality of responses is through the use of Retrieval Augmented Generation (RAG). The model unites the natural language generation characteristic of transformer models like GPT with techniques from Information Retrieval (IR) in such a way that allows the chatbot to access external data from a large corpus in real time. The main idea of RAG is that other than relying utterly on the pre-trained knowledge of the model itself, the system would actively, from some given databases or documents, find information relevant to the discussion and hence make its responses more relevant and accurate. This paper will discuss how the integration of RAG may optimize the efficiency of chatbots for application contexts that involve well-documented

answers in a timely and current manner, leading the way to new applications in customer service. As illustrated in 1.1, the process of RAG is quite straightfor-

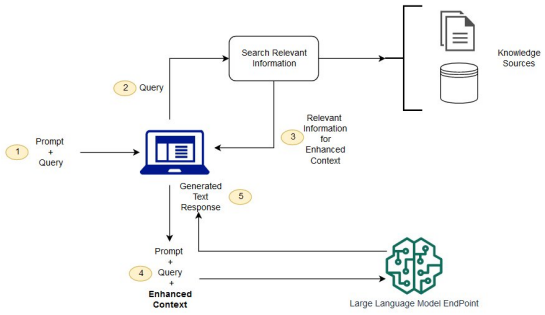

Figure 1.1: Visual example of RAG

ward. The user provides the system with a query. The system uses the query to fetch from the knowledge base the most relevant and important documents. The degree of similarity of these documents to the query will form the basis upon which they are selected. Next, the system feeds the prompt and user query through a LLM together with retrieved documents to come up with a human-like response. And this, consequently, results in returning this information to the user by answering his question accurately and naturally.

2 Background

This chapter details the theory behind the various technologies introduced in this thesis.

Specifically, it will begin with discussing vector databases, where the primary difference with traditional databases is that vector databases allow the storage and querying of data represented as vectors, addressing a solution to the difficulty of finding answers within document chunks.

Then, it analyzes the concept of embeddings, which enable dense numerical representations of text, expressed as vectors, to efficiently perform queries and quickly access relevant information from a knowledge base.

After, it moves to the part of the project when it comes to the use of the LLMs. It is explained how the LLM can deliver a clear and succinct answer, adhering to all the users informational needs.

Lastly, the concept of prompt engineering is explained. It involves strategically designing task-specific instructions, referred to as prompts, to guide model output without altering parameters.

2.1 Vector Database

VECTOR database is a type of database that stores data as high-dimensional vectors, which are mathematical representations of features or attributes. The vectors are usually generated by applying some kind of transformation or embedding function to the raw data [5]. Vector databases are built to manage vector embeddings, and therefore offer a complete solution for the management

of unstructured and semi-structured data[17]. It leverages the power of these vector embeddings to index and search across a massive dataset. This is possible because vector databases typically implement one or more Approximate Nearest Neighbor (ANN) algorithms, that search through hashing, quantization, or graph-based search, so that one can search the database with a query vector to retrieve the closest matching database records [19]. A vector database is different from a vector search library or vector index: it is a data management solution that enables metadata storage and filtering, is scalable, allows for dynamic data changes, performs backups, and offers security features. In figure 2.1 it is possible to see a 2D representation of a vector DB. One of the first significant Database (DB) was Annoy [7], developed by Spotify, which introduced an efficient method for approximate ANN search in large datasets. Subsequently, Faiss [4], created by Facebook AI, improved the efficiency and scalability of these operations, enabling its application on even larger datasets through compression techniques and optimization of similarity metrics. In parallel, the HNSW [8] algorithm provided a graph-based structure that enabled much faster and more accurate approximate nearest neighbor searches in high-dimensional spaces. Finally, among the most recent developments is Chroma , a modern vectordb that stands out for its optimized integration with machine learning models and chatbot applications based on large language models, making it particularly suitable for RAG systems and advanced AI applications. Vector databases are essential in artificial intelligence research and handling huge quantities of unstructured text data. Their capability to store information in vector spaces leads to better performance and accuracy in things like semantic similarity computations or contextual queries. There are several reasons why one should choose a quality vector database including search efficiency, scalability, accurate similarity measures and various distance metric support. All of these factors led us to select and test three different vector databases, listed below.

- Atlas MongoDB
- Postgresql
- Faiss

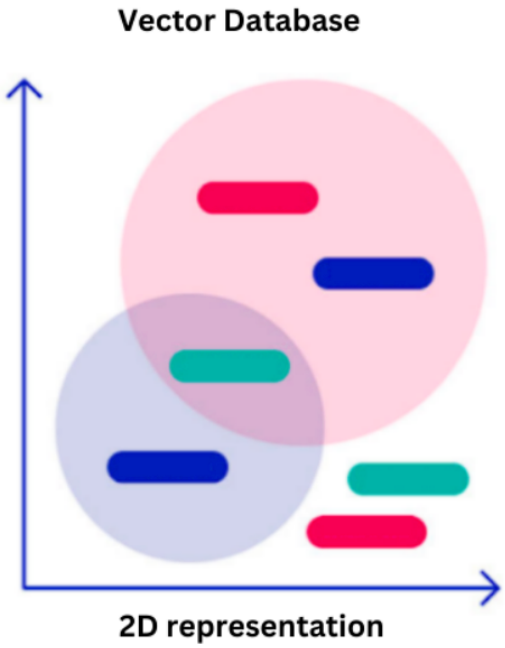

Figure 2.1: 2D example of a vector DB

2.1.1 Atlas MongoDB

Atlas MongoDB is a fully managed cloud database platform built on MongoDB, one of the popular NoSQL databases. One of its major strengths is automatic scalability: Atlas MongoDB can seamlessly grow in capacity and performance, adapting to the needs of the application without requiring manual intervention. Other strengths include advanced geographic replication, global high-availability, which is crucial for distributed applications like RAG or other forms of distributed AI, automated security management, backups, and updates. These improve security and reliability.

The weaknesses, however, also include costs: being a managed service, costs can get extremely high when needs dictate heavier workloads or intensive storage, and it is less flexible compared with a self-managed database, which allows more granular control to be applied to configurations. In addition, since it is built to deal with semi-structured data, it may not be suitable for all kinds of applications; for example, those that require stricter schemas and relational constraints.

MongoDB Atlas is ideal for RAG due to the mass amounts of unstructured and semi-structured data it can support in a flexible and scalable manner. Those very relevant features are to train and deploy AI models for handling complex

data such as images, text, and other types of unstructured inputs whereby high demands for latency and throughput are required.

2.1.2 PostgreSQL

Pgvector is a PostgreSQL extension designed to support the storage and search of vectors, a data format commonly used in NLP applications such as RAG and other AI models. One of Pgvector's main strengths is the ability to natively integrate vector similarity search directly within PostgreSQL, leveraging the powerful relational and transactional capabilities of the database. This allows the combination of traditional structured data with vectors, enabling more complex and sophisticated queries on mixed data.

Among its strengths, Pgvector offers support for vector indexing through algorithms like Annoy and Hierarchical Navigable Small World (HNSW), making vector searches fast and efficient even on large datasets. Additionally, being based on PostgreSQL, it benefits from all the security, reliability, and integration features of this mature and well-established Relational Database Management System (RDMS).

However, there are some weaknesses. The performance of vector search can degrade with extremely large datasets compared to more specialized solutions, which are specifically optimized for handling vectors at scale. Moreover, using Pgvector requires tuning to achieve the best performance.

Pgvector is particularly suited for RAG because it effectively manages vectors that represent images, text, or other features learned. Thanks to its integration with PostgreSQL, it also allows the combination of these vectors with traditional structured data, offering greater flexibility in queries and centralized data management.

2.1.3 Faiss

FAISS, or Facebook AI Similarity Search, is an open-source library developed to perform efficient similarity searching in high-dimensional vector spaces, one of the fundamental operations for many AI applications, including RAG. Among FAISS's major advantages are large-scale vector similarity searches with advanced techniques such as compression, clustering, and indexing for memory use and speed optimization. It's also pretty highly scalable, along with its GPU capabilities, which are quite helpful for a high-scale application.

Other advantages of using FAISS include flexibility: it supports various distance metrics, including Euclidean and Cosine. It allows balancing precision with speed depending on particular needs. It is particularly well-suited for the search vectors representing images, text, or embeddings from ML models and is excellent in AI scenarios, where the system should be able to quickly search for relevant information in huge amounts of data.

However, one of the biggest flaws in FAISS is that it does not really fit the bill for a relational database-one that would typically present very serious data management features like transactionality or handling structured data-and this is usually expected from any RDMS like PostgreSQL. It does not make it very suitable for the requirements of most applications needing data management that is complex in nature and exceeds the vector similarity search. Another weakness with FAISS is that sometimes it requires a great deal of configuration and optimization, especially when the datasets are really large.

FAISS was particularly fitted for RAG, since it allows for fast similarity searches over large datasets-a key ingredient when aiming at improving the accuracy and efficiency of the generation models. In RAG, where the system needs to quickly retrieve relevant information from a vast knowledge base to generate content with high accuracy, FAISS provides the necessary speed and efficiency needed to handle these workloads.

2.2 Embedding

Text embeddings serve as fundamental components in IR systems and retrieval augmented language models [14].In the field of IR, the first-stage retrieval often relies on text embeddings to efficiently recall a small set of candidate documents from a large-scale corpus using ANN search techniques [22]. Text embeddings represent discrete text inputs (e.g.,sentences, documents, and code) as fixed-sized vectors that can be used in many downstream tasks. [20]. Embeddings also represent information-dense representations of textual semantics, with each embedding being a floating-point vector. The distance between two embeddings in the vector space is correlated with the semantic similarity between the two inputs in the original format. In summary, Embedding helps computers understand the "meaning" represented by human information. The first significant approach was Word2Vec [9]. This particular model made it possible to encode words into continuous vectors, by only taking into account the

context of the words in question, thereby changing the course of understanding the language. Furthermore, it revolutionized NLP by introducing BERT [3], where it became possible to produce context representation that is dependent on not only the words before or after a word, but also both. Furthermore, it gave rise to another technology, the SBERT [14], which enabled one, due to his implantation of the BERT technology, to create so called embeddings of sentences which were more advantageous in comparing one text with another. It is also worth mentioning that the concepts above were adapted for multilingual use in XLM-RoBERTa [2]. This let the generation of effective embeddings for languages by abstracting separation of hubs and enhancements of available NLP apps in several communities around the globe.

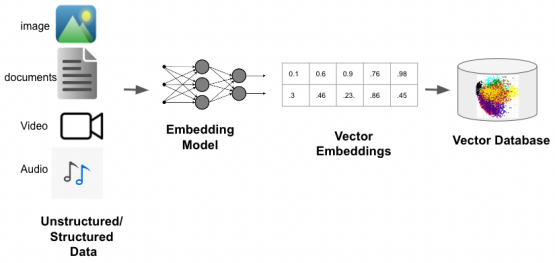

Figure 2.2: Example of an embedding process and storage in the vector database

This thesis focuses on three different embedding models listed below, each of which offers unique characteristics and specific advantages for natural language representation and text processing.

- All-MiniLM-L6-v2
- Text-Embedding-3-Small (OpenAI)
- Multilingual-E5-large

2.2.1 All-MiniLM-L6-v2

All-MiniLM-L6-v2 is an embedding model developed by Microsoft, designed to generate efficient and accurate vector representations of text. One of its main strengths is its lightweight nature and speed, thanks to its MiniLM architecture, which allows it to produce high-quality embeddings with a reduced number of

parameters. This makes it ideal for real-time applications and use on resource-constrained devices without significantly compromising result quality. Additionally, All-MiniLM-L6-v2 has been trained on a wide range of data, giving it good generalization capabilities across various domains and topics.

However, there are some weaknesses as well. While the model offers competitive performance, it may not achieve the same level of accuracy as larger and more complex models like various BERT or RoBERTa versions, especially in highly specialized contexts or when working with very long texts. Moreover, its reduced size means it might struggle to capture certain linguistic nuances or complex contexts.

All-MiniLM-L6-v2 is an excellent embedder for RAG because it provides fast and high-quality embeddings, allowing the system to efficiently retrieve relevant information from large datasets. Its ability to generate vector representations for text makes this model particularly well-suited for powering search and retrieval systems that need to match textual queries with documents or text fragments, thus facilitating the generation of informative and relevant content in response to user questions or requests.

2.2.2 Text-Embedding-3

Text-Embedding-3- is an advanced embedding model designed to generate vector representations of text, offering a balance between quality and performance. One of its main strengths is its ability to capture contextual meanings in depth, thanks to training techniques on large corpora of data and optimized neural network architectures. This enables it to produce embeddings that are highly informative and useful for complex tasks such as semantic search and information retrieval. Additionally, the model is designed to be scalable, making it suitable for production environments and high-demand scenarios.

However, there are also some limitations. While Text-Embedding-3 is powerful, it may require significant computational resources for training and inference, which can pose a challenge for large-scale implementations, especially in contexts with limited budgets or suboptimal hardware. Moreover, the quality of the embeddings can vary depending on the specificity and complexity of the text domain; for highly technical or niche texts, further optimizations may be necessary.

Text-Embedding-3 proves to be an excellent embedder for RAG as it provides

high-quality embeddings that facilitate the retrieval of relevant information from large volumes of data. Its ability to generate detailed semantic representations allows the system to effectively match textual queries with documents or text fragments, thus improving the quality of generated responses and ensuring informative and contextually relevant content for user needs.

2.2.3 Multilingual-E5-large

Multilingual-E5-large consists of a cutting-edge embedding model with the ability to generate high-quality vector representations of texts in multiple languages. One of its main strengths lies in its multilingual capabilities, whereby it can perform uniformly well across several languages-Hence, something that seems an important requirement in the current globalized context where applications are expected to cater to diverse user bases. It has been trained on extensive multilingual data sets for formulating semantic nuances and contextual meanings concerning certain tasks, thus making it an efficient model for tasks such as cross-lingual information retrieval and translation.

However, it also has its weaknesses. Although Multilingual-E5-large performs relatively well across languages, in certain contexts its performance may still fall short when it comes to language-specific models, especially when considering rare languages and particular domains, where denotations regarding domain-specific vocabulary and context are of great importance. Besides, being a large model, it needs substantial computational resources to train and perform inference, posing a barrier to deploying live systems on hardware or budget-constrained environments.

Multilingual-E5-large is a great embedder for RAG because it retrieves useful information from multilingual data sets. Its potential to dedicate high-quality embeddings allows the system to match queries in multi-languages with useful documents or text fragments, into consequence improving the quality and relevance of responses generated. Thus, it is an invaluable application of this work that calls for seamless multilingual support, making it possible for users to extract accurate and context-sensitive information across a diverse range of languages.

2.3 LLM

Large Language Models (LLMs) have drawn a lot of attention due to their strong performance on a wide range of natural language tasks, since the release of ChatGPT in November 2022[10]. LLM are state-of-the-art artificial intelligence systems designed to mimic human language comprehension and generation closely. They rely on deep strategic learningmostly, transformers that allow them to handle extents of text sequences and identify exceedingly complex patterns, relationships, and contextual meanings hidden among words. LLM are pre-trained on large datasets that usually include over billions of words drawn from numerous sources, including books, websites, and articles. Such extensive cumulative training enables LLM to perform complete tasks such as completing text, translating between languages, summarizing, and even mimicking human-like dialogue. Models like GPT-4 [13] and BERT have become an inseparable part of AI applications such as chatbots, virtual assistants, and content generation; understanding and generating coherent contextualized language is now central to them. The history of LLM began with simple models like Word2Vec, which was used in 2013 to generate vector representations for words in a way that preserved their semantic relationship [9]. The wave of transformer architecture developed by GPT and BERT in 2018 was a few steps ahead, tackling more extensive and complex datasets. The bilaterally devoted context of each word allowed BERT to know their contextual nature from the context surrounding it that used both directions and trained other natural language processing tasks [3]. GPT-3 would further revolutionize the field by training at an enormously larger scale than its predecessors, having 175 billion parameters and performing zero-shot and few-shot learning. Today, models such as GPT-4 and LLama[21] stand toe-to-toe being at the forefront of their genre, providing ever more nuanced and contextually accurate language understanding and generation across languages and genres.

- gpt-4o
- LLama 3
- Mistral 7B

2.3.1 GPT-4

GPT-4 is a very recent LLM developed by OpenAI, known for its astonishing capabilities in the field of NLP understanding and generation. It is good at generating coherent and contextually relevant texts on most topics, as it has been broadly trained with diverse datasets. This allows GPT-4 to excel in tasks that require subtle understanding and imagination; therefore, it is extremely useful in applications for content creation, dialogue systems, and information synthesis. Besides, GPT-4 integrates advanced reasoning into its framework, making the model solve complex tasks like summarization and question answering, even code generation.

However, there are a few weaknesses. GPT-4 performance may heavily depend on the quality of the provided prompts, and sometimes it produces wrong or misleading information if the input does not exhibit clarity or specificity. Lastly, being a big hosted model, doing inferences requires a high cost. Also, with GPT-4, extended conversations can sometimes present some consistency challenges since it may lose track of the context in a longer conversation.

GPT-4 works perfectly for RAG because it will be in a position to incorporate information from retrieved sources into its generative processes seamlessly; thus, improving the relevance and accuracy of its outputs. The current version, GPT-4, now extends its powerful language generation capability by also adding a retrieval mechanism to better synthesize information from large databases and ultimately provide well-informed, contextually rich responses. This synergy will make it particularly formidable in applications requiring up-to-date information, as it generates content that is not only coherent but based on the most relevant data given to the system.

2.3.2 LLama

Llama3 is a powerful LLM developed by Meta, aiming to achieve state-of-the-art results in natural language understanding and generation. One of its key features is the generation of high-quality, contextually appropriate text on a wide variety of topics due to training on extensive and diverse datasets. Optimized for performance, Llama3 efficiently generates coherent and relevant outputs faster, and this is highly useful for real-time applications. Also, the architecture is fine-tuning-friendly, allowing developers to adapt the model to specific domains or

tasks, which further increases its versatility.

On the other hand, Llama3 also has some disadvantages. Similar to other LLM, it can be extremely sensitive to phrasing in input and may give responses that are inexact or biased if the input is imprecise or badly constructed. While impressive in capability, the model requires formidable computational resources both for training and inference, respectively, which restricts the accessibility of this for constrained hardware environments. Additionally, Llama3 may at times have a difficult time managing context over longer interactions, thus showing inconsistencies in longer dialogues.

Llama3 is particularly suited to RAG, as it can successfully incorporate externally retrieved information from databases or knowledge bases into its generative processes. That enables the model to further enhance its responses with the most recent information and contextually relevant one, hence returning more and more accurate outputs to the user. Powerful generation combined with precise retrieval made Llama3 an excellent application for any applications requiring access to real-time information and the generation of informative content, thus enhancing a user experience within a represented domain.

2.3.3 Mistral

Mistral is a high-performance LLM, with apparent strengths regarding comprehension and generation in natural language. Its major strength lies in its efficient architecture, whose responses are given in record time while still upholding coherency and consideration of context. Mistral learns subtle features from its training on a wide range of datasets, making it perfect for use in conversational agents, content generation, and summarization, among other applications. Because Mistral is lightweight in terms of size, it can easily be deployed by developers or companies in environments that are computationally poor.

But Mistral does have weaknesses. Like others LLM, it sometimes produces responses that are either not quite right or irrelevant, particularly where the input is ambiguous or out of context. Additionally, while Mistral is optimized for efficiency, this can be at the cost of depth for some complex reasoning tasks compared to larger models, which may have greater insight but at the cost of speed and resource consumption. Moreover, it is hard to maintain context over a long talk, hence, longer conversations may lack coherence.

Mistral is particularly suited for RAG because it can well couple its generative

capabilities with external information obtained from databases or knowledge sources. This synergy will enable the model to enrich its outputs with relevant and timely information to provide users with exact and contextually rich responses. Mistral can generate informative content using retrieval mechanisms in such a way that the generated text is always informed by the best available information, thereby making this model powerful for application areas related to real-time information synthesis or those where high-quality content generation is needed.

3

Experiments

In this Chapter, we shall elaborate on the experiments that were conducted for the purposes of this work. We will take a close look at the storage methods which were put to the test, the different embedding models (embedders) performance, as well as the LLM used in the course of the analysis. The tests were conducted on a custom dataset comprising various documents provided by the company. All technical specifications regarding the dataset can be found in Chapter 8, Section 8.2.

3.1 Storage and Embedding

The storage experiments, which cover both the embedding process and database management due to their close correlation, primarily focus on time efficiency. The first step involves applying an algorithm to extract text from the various file types in the dataset. The RecursiveCharacterTextSplitter [1] method from the LangChain library [2] was chosen for this task. The parameters chunk-size and chunkoverlap were selected based on tests described in 3.2.2, as they are crucial for retrieval performance.

Next, the embedding models were evaluated with a specific focus on their average embedding speed when processing the dataset. This measurement

[1] https://api.python.langchain.com/en/latest/character/langchain_text_splitters.character.RecursiveCharacterTextSplitter.html

[2] https://api.python.langchain.com/en/latest/langchain_api_reference.html

is essential, as it impacts how quickly the system can convert raw data into embeddings, which is crucial for maintaining efficiency in real-time applications. Additionally, the size of the embedding vectors produced by each model was assessed, as larger vectors can increase storage requirements and slow down retrieval processes, while smaller vectors may risk losing important information.

Following this, the average data-saving speed was analyzed across the three databases mentioned earlier. This analysis involved measuring how quickly each database could store the generated embedding data, ensuring that they could handle the volume efficiently without introducing bottlenecks.

Finally, both the embedding models and databases were evaluated in terms of maintenance costs and computational complexity. These factors are critical in a production environment, where cost efficiency and scalability are vital. This thorough evaluation helps ensure that the selected solutions provide optimal performance while remaining cost-effective in the long run.

3.2 Retrieval

The retrieval process is one of the most critical components of the system, and thus received significant attention during the testing phase. Selecting the right embedding model, with carefully tuned parameters, can mean the difference between a system with near-perfect performance and one that fails to meet user expectations. Initially, an in-depth analysis of the Massive Text Embedding Benchmark (MTEB) [3] [12] was conducted, from which the three embedding models mentioned earlier were chosen. These models were selected based on several important factors.

A key consideration was ensuring compatibility with the Italian language, as the entire system is built around documents written in Italian. It was essential that the chosen models could accurately understand and process queries and content in this language. Another crucial factor was the performance metrics reported for each model, which were weighed against the size of the model. Model size directly impacts computational complexity and the associated costs in a production environment, making it a crucial criterion for model selection. Tests were conducted to assess the *retrieval speed*, a critical performance metric

[3] `https://huggingface.co/spaces/mteb/leaderboard`

given that users expect fast responses. The system's ability to retrieve documents quickly and efficiently is essential for delivering a positive user experience.

Additionally, tests were performed on *faithfulness*, a metric that evaluates whether the system's responses are hallucinated or factually grounded. Specifically, it measures if the response generated by the query engine accurately reflects information found in the source documents. Ensuring faithfulness is vital to maintain the integrity and reliability of the systems output, preventing the model from generating incorrect or misleading information.

Finally, a *Relevancy Evaluator* was employed to assess the relevance of the retrieved documents. This metric determines whether the system's responses adequately address the user's query, ensuring that the retrieved information aligns with the question posed. By evaluating both the query and the source documents, the relevancy metric ensures that the responses are not only fast and factually accurate, but also directly answer the user's specific needs.

In conclusion, the combination of these tests, retrieval speed, faithfulness, and relevancy, was critical in determining the optimal embedding models for the system, ensuring both high performance and cost-effectiveness. These carefully selected models not only guarantee rapid responses but also improve the overall reliability and relevance of the retrieval process, which is central to the success of the system.

3.2.1 Distance metrics and top K parameter

Regarding RAG, the probability that a query is related or similar to the documents in the knowledge base is estimated through distance metrics. These metrics are applied to determine the relationship between the query and the embedded representation of the documents concerning their closeness. Commonly, the general distance metrics used here include cosine similarity, Euclidean distance, and dot product. Cosine similarity is widely used because it is a measure of the angle between two vectors, and it has a good estimation of the similarity independently of their magnitude. This is especially useful when dealing with high-dimensional spaces-as is common for embeddings. On the other hand, Euclidean distance quantifies the straight-line distance between two points in space. Because of that, this metric is more sensitive to the absolute position of vectors, while dot product will pay more attention to the alignment and the magnitude of the vectors. For these reasons all these 3 metrics were tested

Besides the different distance metrics, the retrieval process naturally depends on the top-k parameter: the number of top-ranked documents, sorted by their scores with respect to similarity that are returned to the model for response generation. The larger the *k* value, the more context there is, but at an increased computational cost, which may prolong the response times. On the other hand, smaller *k* values speed up the process at the risk of excluding important information. The right balance for top-k is what will ensure that optimization for both the accuracy and efficiency of a RAG system is achieved in a way that the final response includes the most relevant documents. For these reasons, a test was conducted to determine the optimal value for the *k* parameter. Starting with low values, it was progressively increased until reaching a threshold where the system became too slow and its performance decreased due to the inclusion of too much unrelated information.

3.2.2 Chunk Parameters

The chunk size and chunk overlap parameters are critical for the proper functioning of retrieval systems, as they significantly impact both efficiency and accuracy in various ways. These parameters directly influence how data is segmented and retrieved, and thus must be carefully tuned for optimal performance.

- *Relevance and Granularity*: A smaller chunk size results in more granular chunks of text. This increased granularity can be beneficial for capturing fine details, but it also introduces a risk: essential information might be fragmented across multiple chunks and may not appear within the top-retrieved chunks, particularly if the top_k_similar [4] setting is too restrictive. As a result, the retrieval system might fail to retrieve all the necessary information to answer a query accurately. On the other hand, larger chunk sizes are more likely to capture complete and coherent sections of information, increasing the chance that the top-retrieved chunks will contain all the relevant details for the query. This makes the system more reliable in ensuring that the correct answers are included in the retrieval process.
- *Response Generation Time*: As chunk size increases, the amount of information fed into the LLM for generating a response also grows. While providing the model with a broader context can help generate more accurate and complete answers, it comes with a trade-off: longer processing times. If the system is overloaded with too much information, the response generation time can become noticeably slower, which can degrade the overall

[4] Parameter to control how many of the top-retrieved chunks are actually taken into context

user experience. Therefore, finding a balance between the amount of information provided and maintaining quick system responsiveness is crucial for a high-performing retrieval system.

- *System Performance Balance*: The core challenge in selecting the optimal chunk size is achieving a balance between ensuring that all essential information is captured while maintaining fast processing times. An overly small chunk size may result in incomplete or fragmented data retrieval, while a too-large chunk size may slow down response times, making the system inefficient. This balance is highly dependent on the specific dataset and the use case, as different domains may require varying levels of detail or speed.

To address these challenges, comprehensive testing was performed to determine the most effective chunk size and chunk overlap configurations. Each model was tested using a range of values for both parameters, allowing for a detailed analysis of how these settings affected both retrieval accuracy and system responsiveness. By systematically adjusting these parameters, it was possible to identify the optimal configuration that balances speed, accuracy, and relevance for the specific dataset in use.

These tests are crucial because they help ensure that the retrieval system can handle a wide variety of queries efficiently while maintaining high-quality results. Ultimately, fine-tuning chunk size and chunk overlap parameters ensures that the system is not only accurate in retrieving relevant information but also capable of delivering responses in a timely manner, which is essential for any real-world application.

3.3 Answer Generation

Tests on three leading large language models *Llama 3*, *Mistral 7B*, and *GPT-4o* can be conducted to assess their performance across several important dimensions: *response time*, *completeness of response*, and the *human-like quality* of their answers. Response time and completeness performance evaluations were conducted using a set of 100 queries, created with ChatGPT, each provided with a specific context from which the necessary information to answer could be extracted. For the human like quality, 30 queries were created with ChatGPT and then the answers were evaluated by 10 people.

Response time refers to how quickly each model can generate an answer after receiving a query, which is crucial for real-time applications or user-facing

systems where delays can negatively affect the user experience. It is tested just calculating how much time the model requires to answer the query.

Completeness of response evaluates how well the models answer the query in full, ensuring they provide all necessary details without leaving out important information or producing partial answers. This is particularly significant in scenarios where accuracy and thoroughness are essential, such as technical support or information retrieval. This metric was tested with Self-Consistency Testing, that is based on running the same query multiple times, then checking for missing details in one version compared to others. Consistency across outputs can be a proxy for completeness.

In addition, the human-like quality of the responses is another critical factor. This metric looks at how natural, coherent, and conversational the responses are, assessing how closely they resemble the way a human would answer the same question. A more human-like response enhances the user experience by making interactions smoother and more intuitive, especially in customer service or chat-based applications. This was tested using human evaluation. Annotators were asked to rate the responses on a scale of 1 to 5 based on how natural and conversational the answers feel. Evaluators assessed dimensions like fluency, coherence, relevance to the query, and emotional tone. This method gives a direct comparison to human language use. The 3 scoring criteria:

1. Fluency: Does the response flow naturally, without awkward phrasing or grammatical errors?
2. Coherence: Is the response logically structured and consistent?
3. Engagement: Does the response sound engaging or empathetic, similar to human interaction?

By comparing the performance of Llama 3, Mistral 7B, and GPT-4o across these areas, we can better understand each model's strengths and weaknesses, from fast but potentially less detailed responses to slower yet more comprehensive and human-like outputs.

4

Results: Part I

This chapter presents the outcomes of the experiments detailed in 3, along with the rationale behind the selection of specific models and the reasoning for these choices.

4.1 Storage Time

About the databases, everything has gone as expected from the test results. In particular, as it is possible to see from Table 4.1, the performances in terms of timing were pretty similar between all the options. Ultimately, PostgreSQL

Input type	AtlasMongo	Faiss	Postgresql
PDF	6.7	6.5	6.4
TXT	6.2	6.1	5.9
Image	5.8	6	5.4
Web Page	3.2	2.8	2.5

Table 4.1: Mean Time in milliseconds taken to save the data of various db for a single page of different type of data input using the embedder Multilingual-E5-large with chunk size of 1024 and chunk overlap of 256

database with PGvector plugin has been chosen, since it was guaranteed the best compromise: cheaper, moderately fast and data are more easily managed. FAISS was excluded because it's an engine supporting little data with a primary focus on embedding storage. On the other hand, Atlas MongoDB was excluded because it is a paid service, supports only a limited amount of search indexes,

and raises some privacy concerns since it is a hosted database that could be risky for the data managed.

About time performance, also the embedding tests gave expected results: as can be seen in Table 4.2, All-MiniLM was faster than every other model since it is lighter than the others and produces smaller embeddings; on the other side, the slowest was Multilingual-e5, since this model is heavier but also because it produces longer embedding vectors. This test was conducted not only to determine which model was faster but to understand how much the time required to embed degrades as the model increases in complexity.

Parameters	Text-Embedding-3-Small	Multilingual-E5-large	All-MiniLM-L6-v2
CS 256 CO 64	1.5	1.6	1.0
CS 512 CO 128	1.6	1.6	1.1
CS 1024 CO 128	1.8	1.9	1.3
CS 1024 CO 256	1.9	2.1	1.3
CS 2048 CO 256	2.0	2.2	1.5

Table 4.2: Time required by different embedding models to embed a single page of a text file with different chunk size(CS) and chunk overlap(CO)

4.2 Retrieval Performance

In the Retrieval Phase, the first step was to identify which metric would work better for the Search Index of the database. All three most common methods have been tested, whose results are presented in Table 4.3. Cosine similarity function proves to be better compared to the other two methods and hence is preferred. Theoretically, this type of problem matches the cosine function well because it is very efficient when handling sparse vectors: it looks only at non-zero dimensions hence minimizing computational overhead. A weakness of the cosine function is a tendency towards favoring features with high values. Besides, it continues to remain insensitive to the number of features two vectors share as long as the values for those features remain small, a limitation in some contexts.

	Cosine	**Euclidean**	**Dot**
Faithfulness	0.85	0.83	0.80
Relevancy	0.87	0.84	0.85

Table 4.3: Performance of Retrieval using different distance metrics

I	Text-Embedding-3-Small			Multilingual-E5-large			All-MiniLM-L6-v2		
	Time	Faithfulness	Relevancy	Time	Faithfulness	Relevancy	Time	Faithfulness	Relevancy
CS 256 CO 64	0.79	0.94	0.82	0.81	0.94	0.84	0.71	0.92	0.79
CS 512 CO 128	0.84	0.96	0.85	0.85	0.95	0.84	0.76	0.93	0.83
CS 1024 CO 128	0.86	0.97	0.94	0.87	0.96	0.95	0.78	0.95	0.91
CS 1024 CO 256	0.87	0.98	0.96	0.87	0.98	0.97	0.80	0.97	0.93
CS 2048 CO 256	0.90	0.96	0.92	0.92	0.95	0.94	0.83	0.94	0.90
II	Time	Faithfulness	Relevancy	Time	Faithfulness	Relevancy	Time	Faithfulness	Relevancy
CS 256 CO 64	1.57	0.83	0.74	1.64	0.78	0.75	***1.32***	0.78	0.72
CS 512 CO 128	1.66	0.86	0.80	1.72	0.85	0.81	1.41	0.82	0.79
CS 1024 CO 128	1.68	0.89	0.83	1.76	0.89	0.85	1.43	0.84	0.81
CS 1024 CO 256	1.70	0.90	0.85	1.77	***0.91***	***0.91***	1.45	0.85	0.85
CS 2048 CO 256	1.72	0.87	0.83	1.80	0.88	0.87	1.47	0.83	0.82

Table 4.4: Retrieval performance (average time, average faithfulness and average relevancy) using different parameters and embedding models on postgresql; I when only 15 documents saved in the db; II when 500 documents saved in the db

In Table 4.4, the results of extensive performance tests are presented, showing how different chunking parameters affect system efficiency and accuracy. It immediately becomes clear that the optimal balance between chunk size and overlap plays an important role in maintaining high retrieval performance. The consequence of setting the chunk values too low is that logical connections between chunks become unmanageable, and the performance drops because the context then gets fragmented. On the other extreme, if the chunk size is set too large, the system finds it difficult to draw out the relevant information; in most cases, critical details needed for an elaborative response get missed out. These tests showed that the best configuration was to have a chunk size at 1024 and a chunk overlap at 256 since in such a setting, the trade-off was the best between capturing enough context and keeping system efficiency.

The table below also enumerates the performance of the three embedding models tested. All-MiniLM, considering the overall retrieval accuracy, ranks last but is the fastest by a big margin. When the need for speed outweighs the need for precision, this model would be the best one. The other two models, Multilingual-E5 and OpenAI's model, are very close regarding their performance. Careful evaluation led to the selection of Multilingual-E5 because it was cost-effective and easy to deploy. The free model is downloadable and can be run locally, which gives a great deal, more control and flexibility compared to the OpenAI model. While performant, it is costly and relies on hosting elsewhere, raising potential issues in terms of long-term scalability and accessibility.

Finally, the tests show how performance degrades as the number of documents increases-a behavior which was expected but is notable nonetheless. Inter-

estingly, this performance drop is a lot more drastic during the initial stages-say, from 15 to 500 documents-but the decline, indeed, does become shallow with further scaling-say, from 500 to 10,000. That would imply that beyond some starting point, stability in the system's retrieval, in terms of dataset sizes, increases, even though times are lengthening; it becomes more manageable with higher document counts.

4.3 Answer Generation

In the response generation phase, a first test focused on the top-k-similarity parameter that regulates the number of chunks of a document that will be retrieved considering their similarity with respect to the query and then passed to the LLM for composing the answer. As it can be seen in Table 4.5 the best performance was obtained with a top-k value of 6. A further increase in the number of chunks made it difficult for the model to generate a response since the bigger context diluted the important information and it was harder to retain and focus on key information. Regarding the tests on performance conducted

Top-k Value	Time	Completeness
3	1.2	84%
6	1.8	88%
10	1.9	88%
20	2.2	86%%

Table 4.5: Performance of the answers based on the number of chunks in the context given to the model

on various LLM, the results are presented in Table 4.6. It is clear that GPT significantly outperforms the other models in nearly every category. In terms of response completeness, GPT shows remarkable performance: only in 12% of the cases was it possible to obtain additional information by rerunning the query multiple times. This is a strong result, although there is still room for improvement, as will be discussed in the following chapters.

When it comes to producing human-like responses, GPT stands head and shoulders above the competition. Other models often struggle to deliver answers that feel natural or conversational to the user, frequently generating responses that come across as mechanical or artificial. In contrast, GPT excels at crafting responses that are fluent, coherent, and interactive, creating an experience that

feels as though one is engaging with a human rather than a machine. This ability to maintain a conversational tone while still being precise and accurate is one of the key differentiators that sets GPT apart from other models.

In the context of handling empty responses, GPT also leads the field. Many of the other models frequently fail to understand the provided context or formulate an adequate response, often resulting in empty or incomplete answers. These failures are likely due to the models' inability to interpret more complex queries or their difficulty in retrieving relevant information from the context. GPT, on the other hand, almost always manages to find some relevant information, rarely returning blank responses. This robustness in generating a meaningful answer, even in challenging scenarios, highlights GPTs highly capacity for understanding and addressing user queries effectively.

In conclusion, while GPT demonstrates superior performance across the board, particularly in response completeness, human-like interaction, and minimizing empty responses, there are still some areas where further optimization can be achieved. These aspects will be explored in more detail in the upcoming chapters, where methods to enhance performance and reduce the occurrence of incomplete responses will be discussed.

	LLama3	**Minstral 7B**	**GPT 4o**
Time	1.5	0.9	1.8
Completeness	81%	78%	88%
Empty Answer	16%	12%	3%
Human-like score	3.9	3.1	4.8

Table 4.6: Performance of different LLM using context retrieved with Multilingual-E5-large and Postgresql

5 Multi-Stage Retrieval

5.1 What is Multi-Stage Retrieval

Multi-stage retrieval within RAG systems represents one of the most significant innovations in the field of NLP, particularly in the context of question answering and text generation supported by external information. The fundamental idea behind this approach is to overcome the limitations of pure generation models, such as GPT, which, while powerful in generating coherent text, may fail when it comes to providing accurate and up-to-date responses on specific or highly specialized topics [18]. In these contexts, models that integrate retrieval of external information can bridge the gap between fluent generation and information accuracy, ensuring that responses are not only semantically correct but also supported by verified sources[18]. Historically, IR systems were

Figure 5.1: Example of a process of multistage retrieve

based on simple models, such as TF-IDF (Term Frequency-Inverse Document Frequency) and BM25, which use keywords to assess the similarity between a query and a document. While effective in many cases, these approaches were limited in their ability to capture more complex semantic relationships between words or phrases. With the introduction of dense embeddings based on neural

networks, the retrieval systems' ability to understand the underlying meaning of words and sentences improved significantly by leveraging the continuous semantic representation space generated by models such as Word2Vec[9], BERT, or DPR (Dense Passage Retrieval). The evolution toward multi-stage retrieval marked a further enhancement in performance. Rather than relying on a single retrieval phase, this approach breaks the process down into multiple stages, each designed to play a specific role in refining the selection of documents.

5.1.1 2 Stage Retrieval

In the simplest two-stage configuration, the first level (often called broad retrieval) aims to quickly retrieve a wide but heterogeneous set of potentially relevant documents, using fast techniques like dense embeddings or approximate semantic similarity models. Speed is crucial at this stage, as the system must filter through millions of documents in a short time. However, the broadness of this initial retrieval means that many documents may not be strictly relevant[11].

The second stage uses a fine-grained retrieval that applies more complex and computationally expensive models to reorder or refine the documents already retrieved. This second stage may rely on models like BERT-based rerankers or cross-attentive models, which directly compare the query to each document, significantly improving accuracy and relevance. This approach is particularly advantageous because it balances the need for speed with accuracy, enabling more sophisticated models to be used only when necessaryon a reduced subset of documents.

An example of a two-stage retrieval can be seen in figure 5.1 The initial stage performs a broad search, retrieving a large set of candidate documents, typically using fast but less accurate methods like inverted indexing. In the subsequent stage(s), reranking models apply more computationally expensive algorithms, like neural networks or transformers, to a smaller subset of documents. This approach enables both speed and accuracy, balancing the trade-offs between computational resources and search precision.

5.1.2 3 Stage Retrieval

Recently, an additional refinement to multi-stage retrieval has been proposed, known as three-stage retrieval. This approach adds a third selection

level that focuses on final refinement and integrates models for paragraph-level or even sentence-level reranking. After the broad retrieval phase and subsequent reranking, the third stage aims to identify not just the most relevant documents but also the specific segments within these documents that answer the query most precisely. This level of granularity is especially useful in fields like medical literature or legal databases, where a single sentence or paragraph can make the difference in providing a correct and pertinent answer.

The three-stage retrieval approach represents a further improvement in information processing, as it increases overall accuracy while reducing noise in the input dataset. For example, a legal assistance system might first search through millions of legal documents, then select the most relevant sources through advanced reranking, and finally, identify the specific clause or ruling that answers the posed question. This strategy has been successfully implemented in some of the most advanced models, such as ColBERT (Contextualized Late Interaction over BERT), which combines BERT's contextual understanding with optimized techniques for scalable retrieval.

5.1.3 Related Work

Several empirical studies have demonstrated that multi-stage retrieval leads to significant performance improvements compared to single-stage retrieval systems. For example, a key paper [6] on Dense Passage Retrieval (DPR) showed that using dense embeddings for the first retrieval phase, followed by fine-tuning with BERT-based models in the second stage, resulted in an increase in precision in question answering compared to traditional retrieval methods like BM25. Another relevant study [15] on the RocketQA system employed a two-stage reranking strategy to enhance precision in answering questions based on large text corpora. The effectiveness of these approaches comes from the ability of each stage to progressively reduce the set of candidate documents, focusing increasingly on the most relevant ones.

One of the main theoretical advantages of multi-stage retrieval is the combination of high recall with increasing precision. In the first phase, the main objective is to maximize recall, meaning to ensure that a broad and diverse set of relevant documents is retrieved. In subsequent phases, the focus shifts to precision,reducing the set to a few high-quality documents that precisely address the query. This process allows the system to optimally balance com-

putational efficiency and result quality: in the initial phase, simpler but faster algorithms avoid computational overload, while in the later phases, more computationally expensive models are applied only to a much smaller subset of candidates. In conclusion, multi-stage retrieval has proven to be an essential component of modern NLP systems, especially in combination with generative models. Its ability to balance speed and accuracy, handle large amounts of data, and provide detailed, evidence-supported responses makes it an indispensable technique. The advancements in multi-stage retrieval have fundamentally reshaped how machines retrieve and synthesize information, offering significant improvements in precision and efficiency over traditional single-stage methods.

5.2 Implementation

A theoretical study was conducted to see what advantages either a 2-stage or a 3-stage retrieval system could offer, after which the optimal arrangement was decided upon for implementation. Eventually, a 2-stage approach was chosen for a few strong reasons: firstly, the 3-stage system seemed somewhat computationally heavy, a status undesirable in applications like chatbots, where response times are crucial. The performance difference was, in fact, rather minor between the 2-stage and 3-stage systems, too small to justify the extreme increase in latency associated with the latter. Another influential factor in this decision is cost considerations since using only one reranking model instead of two helps minimize the costs while yielding satisfactory results.

For the selection of the reranking model, a comparative analysis was performed by using the MTEB leaderboard [1], only considering the best available free models. The key factors related to the number of parameters directly linked with computational complexity, the performance metrics provided by the developers, and multilingual support-all target documents are in Italian. This choice comes after deep analyses and considerations: bge-reranker-v2-m3 [2] is a lightweight, newly developed reranking model that features high inference speed and was pre-trained on the Italian language as well. This would fit into the needs of efficiency and effectiveness in the system when it comes to user query processing. Different from embedding model, reranker uses question

[1] https://huggingface.co/spaces/mteb/leaderboard
[2] https://huggingface.co/BAAI/bge-reranker-v2-m3

and document as input and directly output similarity instead of embedding. It is possible to get a relevance score by inputting query and passage to the reranker. This allows for the use of two models with different embedding sizes without any issues.

5.3 Results

As illustrated previously, it was decided to adopt a 2-stage retrieval to further enhance the system's overall performance. As the main reranker, BGE-reranker was chosen by considering its excellent performance indicators ranking among the top competitors on the MTEB leaderboard. After several exhaustive tests, the results are recorded in the following Table 5.1. Based on the findings from the experiments, the decision was made to utilize All-MiniLM as the primary embedding model. This choice was motivated by its ability to streamline the process and reduce response times significantly. By opting for All-MiniLM with BGE-reranker, we aim to enhance the overall efficiency of the system while maintaining a high level of performance.

It becomes obvious from the results that both the relevancy and faithfulness metrics are greatly improved by the implementation of the reranking system. In the most favorable cases, faithfulness has increased as high as 5%, while for relevancy it has risen as high as 7%. This improvement is indicative of the reranker being very effective in refining the retrieved results so that they are not just relevant but also more faithfully representative of what is actually in the source documents.

As one would have anticipated, however, the BGE-reranker increased the processing time by quite a lot, with the average response time ballooning an additional 7 seconds. This is a classic trade-off between accuracy and speed when advanced models are deployed. Therefore, to balance enhanced performance against reasonable processing times, top-k was set to 10. This setting provides a good trade-off between significant improvement in retrieval performance and reasonable degradation of response time. It would, for instance, be possible to increase the top-k value to 20 in applications that require maximum precision; however, this would obviously further impact system latency. Such considerations become highly important in view of tuning for an optimum overall system performance, especially when applications call for both speed and accuracy.

	Time	**Faithfulness**	**Relevancy**
Best Without Reranking	1.77	0.91	0.91
Reranking top10	4.32	0.94	0.93
Reranking top15	6.12	0.96	0.96
Reranking top20	8.84	0.97	0.98

Table 5.1: Performance of the system with 2-stage retrieval of different top elements with All-MiniLM and Bge-reranker-v2

6 Prompt-Engineering

6.1 What is prompt engineering

Prompt engineering is the art of designing and optimizing commands or instructions, known as prompts, given to AI models, such as LLM, to obtain desired responses or results effectively. These models process natural language based on the prompts they receive, so the clear and precise formulation of the prompt is essential for obtaining accurate and useful responses. Good prompt engineering requires understanding how the model interprets language, leveraging formats, keywords, and context to guide the model in generating relevant content. [1] Historically, language models required intensive training and the creation of complex datasets to achieve good results on specific tasks. However, with the advent of large pre-trained models, the idea of manipulating the promptthe formulation of the input given to the modelemerged as a way to guide and refine the generated response without needing to modify the underlying model. This technique has produced remarkable results, allowing users to fully leverage the capabilities of language models simply by adjusting the input text. The strengths of prompt engineering lie in its flexibility and its ability to adapt to a wide range of applications without requiring repeated training. It is used because it enables the generation of more relevant, coherent, and specific outputs, improving the model's understanding of context. An ambiguous formulation, in fact, could lead to incorrect or irrelevant answers. This field became crucial with the rise of deep neural network-based artificial intelligence

models starting in 2020. Its development is closely linked to the advancement of machine learning techniques and the evolution of models' ability to understand and generate natural language. [16]

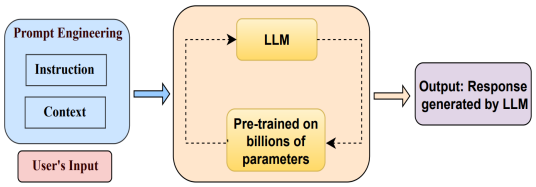

Figure 6.1: Visual breakdown of prompt engineering components: LLM pre-trained, instruction and context as key elements for the prompt, and a user input interface

6.1.1 Empty Answer

One of the major issues associated with LLM is that sometimes, the models could stumble to provide a relevant or complete answer to a query, especially when the context is complex or the question is mixed up. This could be caused because sometimes the model may not have the complete understanding of the context of the question. This issue can be resolved through prompt engineering by furnishing the model with very particular and organized input and guiding it towards a clearer picture of what to do and how to do it. For instance, one way to make sure the model outputs more relevant responses is to add certain information, such as the expected response format or the thematic context. Explicit instructions in the prompt could ensure that the model attempts to provide a response, even in cases where it doesn't have all the needed information. This ensures that the model won't just get stuck or produce generic output.

6.1.2 Out of Knowledge-Base

In the context of RAG, a common issue with LLM is their tendency to generate responses that go beyond the provided knowledge base, drawing on unverified or hallucinated information learned during pre-training. This happens because LLM, while powerful, may blend pre-trained knowledge with retrieved information, producing answers that do not accurately reflect the external sources. Prompt engineering can address this issue by guiding the model to more rigorously rely on the knowledge base. Well-structured prompts that emphasize the

importance of basing responses solely on retrieved information (e.g., instructions like "Answer only using the provided documents" or "Do not generate information not found in the texts") can significantly reduce the likelihood of generating responses outside the required context. In this way, prompt engineering becomes an effective tool to improve the reliability of responses in RAG systems, keeping the focus strictly on the provided external information.

6.2 Find the best prompt

To find the optimal prompt, we focused on addressing several key issues: response completeness, empty responses, out-of-context answers, ensuring responses were always in Italian, providing accurate citations for the information retrieved, and dealing with information spread across multiple chunks of textparticularly because the model often missed details in later chunks.

6.2.1 Language Consistency

For the issue of language, the solution was relatively simple. We included the prompt command "Answer always in Italian", which consistently ensured that the model responded entirely in the desired language. Prior to this, the model sometimes reverted to English or mixed languages in its output, but this adjustment resolved that issue completely.

6.2.2 Out-of-Context Responses

To mitigate out-of-context responses, several tests were conducted with different prompt formulations. Ultimately, the most effective solution was to explicitly state "use ONLY the documents as context" in the prompt. This drastically improved the models ability to focus solely on the provided documents, preventing it from incorporating irrelevant information from its pre-trained knowledge, and reducing hallucinations.

6.2.3 Empty Responses

Addressing the problem of the model providing empty responses was slightly more complex. Initially, the model would return blank responses when it failed to find relevant information in the context, without indicating why this occurred.

To remedy this, we added the prompt "If you don't find the answer in the context, just say it". This clarified the behavior and ensured that, when the answer was unavailable in the context given, the model would explicitly state that instead of remaining silent, enhancing transparency.

6.2.4 Handling Multiple Chunks of Information

For the issue of retrieving information spread across multiple document chunks, we introduced the prompt "Since the context given is split in different documents, answer using all the documents". While this improved the retrieval of information from multiple sources, it was not sufficient to fully solve the problem. The model sometimes included unnecessary details from irrelevant documents. To refine this behavior, we expanded the prompt with "answer using all the documents you find appropriate", which led to better selection of relevant documents while avoiding irrelevant information.

6.2.5 Citing Sources

Providing accurate source citations was another area of concern. Initially, we used the prompt "Specify the document name, the document URL, and in which page you found the data", but the model inconsistently cited sources, sometimes omitting the information altogether. By emphasizing the importance of citations with "Specify ALWAYS the document name, URL, and page" (with "ALWAYS" in capital letters), the model demonstrated a significant improvement. This modification made the model much more consistent in providing source information, as explained in the subsequent results chapter 6.3.

6.2.6 Completeness of Responses

Completeness was the most challenging aspect to address. Multiple prompt variations were tested before finding the most effective approach. Initially, prompts like "Answer with all the information you find appropriate" and "Give all the information related to the query" were tested, but they failed to yield comprehensive answers. We then experimented with defining a role for the model, using prompts like "You are a smart bot that has to fulfill the request of the user". While this improved the models ability to provide more detailed responses, it still wasnt sufficient.

The next step was to include the prompt "Give precise answers to the query", but this led to overly long and verbose responses, which were not ideal. After further refinement, we landed on "Give short but very precise answers with all the information necessary to understand the answer", which struck the right balance between brevity and completeness. This formulation encouraged the model to provide concise yet fully informative responses, ensuring that the user received all relevant information in a clear and direct manner.

Overall, this iterative process of prompt engineering significantly improved the performance of the system, solving key issues and optimizing the quality of the model's responses across various parameters.

6.3 Results

To select the most effective prompts, a series of tests were conducted on the system using 100 specially created queries. These queries were designed to challenge the system's capabilities, with 20 of them intentionally crafted to elicit responses not found within the provided knowledge base. This approach aimed to evaluate whether the system would resort to its pre-existing knowledge to generate answers. As illustrated in Table 6.1, the use of prompts significantly

	Expected	**Empty**	**Out of Context**
Basic prompt	79	3	18
Anti-Empty prompt	84	0	16
Anti-OOC prompt	95	2	3
Final prompt	98	0	2

Table 6.1: Results obtained with different prompt. First row shows the results of the system with a basic prompt, second row shows the results with a prompt focused on avoiding getting a empty answer, third rows shows results with a prompt focused on forcing the LLM to answer only with the context in the knowledge base and last row shows the results with the final prompt

enhances performance, particularly by preventing the system from fabricating answers or providing responses outside the relevant context. The results demonstrate a remarkable improvement, with the system's accuracy increasing from 79% to 97% in terms of expected responses. Furthermore, the prompt "Answer always in Italian" enabled the model to respond to 100% of the queries in Italian.

Additionally, the prompts contributed to the completeness of the answers. While the responses are not yet entirely comprehensive, they exhibit a substantial

increase in the relevance of the information provided, which is articulated more clearly and precisely.

Lastly, the integration of page numbers and links to the sources of information was successfully implemented. Out of the 80 queries considered (excluding those outside the knowledge base for obvious reasons), the system only failed to provide the page number in one instance, did not include the link in two instances, and in five cases, when extracting information from multiple documents, it failed to provide links to all relevant sources.

Overall, these results underscore the importance of providing carefully crafted, specific prompts to generative models, highlighting their crucial role in optimizing performance and enhancing the quality of responses.

7

Results: Part II

Chapter 7 will present the results obtained from the integration of prompts and reranking techniques, offering a comprehensive overview of the system's performance. This chapter aims to delve into the impact that these enhancements have had on the overall functionality and effectiveness of the retrieval process.

7.1 Final System

The final system architecture incorporates PostgreSQL as the data storage system, extended by the PgVector plugin with the chunk size being set to 1024 and chunk overlap set to 256, and cosine similarity function as a search index. The main embedding model is Multilingual-e5-Large; the reranking model is BGE-reranker-v2-m3. In the actual generation, the main large language model GPT-4o is combined with the prompt strategy in Chapter 6 to establish an efficient and robust RAG system. In conclusion, all the final performance metrics of the system are very high as summarized in Table 7.1.

The project successfully attained its goal of developing an intelligent chatbot that is capable of responding with human-like responses, though there is always room for optimization. The system returns high relevancy and faithfulness in the answers, hence making sure that the user gets appropriate contextually correct information. Also, one of the most crucial objectives was ensuring that responses come with references to the sources, which has also been achieved. This will be easier for the user to verify the information provided, filling in the gaps or ambiguities by directly looking up the original documents. The balance

of speed and accuracy has been well considered to improve user experience.

Multilingual-e5-Large embeddings mean multilinguality is supported with good performance without high computational cost. Adding BGE-reranker-v2-m3 enhances the quality of retrieval documents, refining results for more relevance. Meanwhile, responses using GPT-4o are sure to be coherent and naturally phrased in a manner that will make interactions feel fluent and smooth for the user. Putting these together into a holistic approach means users have an advanced retrieval technique coupled with powerful generation, thus giving rise to a highly functional, friendly, and therefore trustworthy chatbot.

Time	**Completeness**	**Human-like score**	**Correctness**
6.26	94%	4.4	97%

Table 7.1: Performance of the full system

8 System

In this chapter we are gonna present the setup, the data and the machines used to conduct the experiments in the study.

8.1 Machines

The complete setup was developed and tested on a computer provided by the company. The technical specifications of the computer can be seen in Table 8.1.

CPU	RAM	GPU	Disk
i7	16 GB	integrated	512 GB

Table 8.1: Specs of the computer

8.2 Dataset

The experiments were conducted using data from various files, with the future intention of expanding the system to accommodate additional data types. As the initial set of tests proceeded on a limited dataset to check for the proper functioning of the system, there was also an increase in the document numbers to examine its performance and stability with heavier loads of data. Gradual increase in data load thus allowed a thorough assessment of the system's ability to maintain efficiency and accuracy. The data primarily consisted of three file

types: textual files, images containing text, and web pages with textual content. Table 8.2 provides an overview of the number of documents used for each file type for the two test. The files contain a wide range of information across vari-

	I	**II**
Text Files	5	320
Images	5	80
Web Pages	5	100
Total	15	500

Table 8.2: Number of documents in dataset for each data type

ous fields, including economic, medical, and social data, as well as instructions for fulfilling requests and completing forms. This diversity allows us to assess whether the system can effectively differentiate between documents of varying categories during the retrieval process. Additionally, some files included complementary information designed to test the system's ability to gather data from multiple documents and provide comprehensive responses. The textual files utilized in this study averaged around 20 pages in length and included not only written content but also images and tables. The images, on the other hand, were screenshots of text pages extracted from PDF files. Web pages were retrieved using a script from a mock website. A preprocessing step was applied to the web pages, during which irrelevant sectionssuch as headers, footers, navigation bars, and formswere excluded to retain only the essential textual content.

9 Conclusions and Future Works

This thesis has undertaken an in-depth exploration of the development process of a RAG system, with a specific knowledge-based optimization aimed at enhancing human-machine interaction through the use of recent LLM models to create a powerful AI assistant. Through careful analysis and experimentation, this work has highlighted the capabilities and limitations of various embedding models and the new LLM.

9.1 Conclusion

The initial experimentation involved the creation and comparison of a standard RAG with the most common models in the literature. Despite differences in training setups, comparable results were achieved. Subsequently, the study considered data storage solutions. Various methods for storing data were tested, and based on specific tests, PostgreSQL with the pgvector plugin was chosen. Different methods were implemented for saving text files, images, and web pages, which would constitute the knowledge base of the chatbot, with particular attention given to the parameters used for data storage. An analysis of various embedding models was conducted, taking into account timing, costs, and accuracy. After extensive testing, the "multilingual-e5-large" embedder was selected, as it not only supports multilingual capabilities (including Italian) but also offers a good tradeoff between execution speed and performance, reducing management costs since it is open source. However, given the need for high

performance, a reranking model was tested to refine the process. This addition significantly improved the results, achieving a very high accuracy, though it also considerably increased execution time. In the concluding portion of the thesis, a meticulous comparative analysis has been conducted among various Large Language models (LLMs) available. After thoroughly analyzing the properties of these models, however, it has been determined that GPT-4 was predominated, given its performance superiority over the other examined models. Because of this capability, the GPT-4 has provided better-structured and human-like-generated responses than any of its counterparts. Several tests were carried out to determine whether the responses were complete, accurate, and truthful. Thereafter followed a human twist on the responses, analyzing their coherence and possibly naturalness in interactions. An important part of the analysis was a test for the information: it was essential to find out whether the model relied strictly on data present in the knowledge base or used pre-existing knowledge already built into that model. In addition, a detailed prompt design analysis was done to derive the maximum relevance of the obtained responses. This required optimal phrasings when interacting with the model to optimize the responses and achieve the responses according to their objectives.

There were also instances where users asked questions unrelated to the context, and through a properly designed prompt, it was possible to guide the LLM to generate a response indicating that such information could not be retrieved from the specific knowledge base. This step was crucial, as without it, the system would not have provided any response, potentially causing confusion or dissatisfaction for the users.

Overall, the system is functional and consistently delivers appropriate responses, which are almost always complete. Additionally, it always includes the sources from which the information is drawn, allowing users to quickly verify any uncertainties directly in the corresponding files. This ensures a more transparent and reliable user experience.

9.2 Future Works

Future work in this domain holds the potential to explore several significant directions. Given the rapid advancements in the field of LLM, it is highly likely that future Molden Architectures will integrate novel approaches and cutting-edge techniques to push the boundaries of optimization systems even

further. This could involve the incorporation of more sophisticated algorithms, enhanced training methodologies, or more refined evaluation metrics to boost overall system performance. Additionally, deeper investigation into key parameters, such as chunk size and chunk overlap across different types of sources, could substantially increase the value and versatility of RAG. By better understanding how these parameters interact across diverse datasets, the precision and relevance of generated outputs could be improved significantly. Furthermore, the fine-tuning of models for highly specific tasks offers the potential to deliver even greater levels of specificity and efficiency, allowing for more focused and context-aware performance in specialized applications. As a result, the overall capacity for handling complex, domain-specific challenges could be elevated, leading to improvements in both scalability and applicability across a wider range of use cases.

References

[1] Banghao Chen et al. *Unleashing the potential of prompt engineering in Large Language Models: a comprehensive review*. 2024. arXiv: 2310.14735 [cs.CL]. URL: https://arxiv.org/abs/2310.14735.

[2] Alexis Conneau et al. *Unsupervised Cross-lingual Representation Learning at Scale*. 2020. arXiv: 1911.02116 [cs.CL]. URL: https://arxiv.org/abs/1911.02116.

[3] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL]. URL: https://arxiv.org/abs/1810.04805.

[4] Matthijs Douze et al. *The Faiss library*. 2024. arXiv: 2401.08281 [cs.LG]. URL: https://arxiv.org/abs/2401.08281.

[5] Yikun Han, Chunjiang Liu, and Pengfei Wang. *A Comprehensive Survey on Vector Database: Storage and Retrieval Technique, Challenge*. 2023. arXiv: 2310.11703 [cs.DB]. URL: https://arxiv.org/abs/2310.11703.

[6] Vladimir Karpukhin et al. "Dense Passage Retrieval for Open-Domain Question Answering". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Ed. by Bonnie Webber et al. Online: Association for Computational Linguistics, Nov. 2020, pp. 6769–6781. DOI: 10.18653/v1/2020.emnlp-main.550. URL: https://aclanthology.org/2020.emnlp-main.550.

[7] Wen Li et al. *Approximate Nearest Neighbor Search on High Dimensional Data — Experiments, Analyses, and Improvement (v1.0)*. 2016. arXiv: 1610.02455 [cs.DB]. URL: https://arxiv.org/abs/1610.02455.

[8] Yu. A. Malkov and D. A. Yashunin. *Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs*. 2018. arXiv: 1603.09320 [cs.DS]. URL: https://arxiv.org/abs/1603.09320.

[9] Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. 2013. arXiv: 1301.3781 [cs.CL]. URL: https://arxiv.org/abs/1301.3781.

[10] Shervin Minaee et al. *Large Language Models: A Survey*. 2024. arXiv: 2402.06196 [cs.CL]. URL: https://arxiv.org/abs/2402.06196.

[11] Niklas Muennighoff. *SGPT: GPT Sentence Embeddings for Semantic Search*. 2022. arXiv: 2202.08904 [cs.CL]. URL: https://arxiv.org/abs/2202.08904.

[12] Niklas Muennighoff et al. *MTEB: Massive Text Embedding Benchmark*. 2023. arXiv: 2210.07316 [cs.CL]. URL: https://arxiv.org/abs/2210.07316.

[13] OpenAI et al. *GPT-4 Technical Report*. 2024. arXiv: 2303.08774 [cs.CL]. URL: https://arxiv.org/abs/2303.08774.

[14] Nils Reimers and Iryna Gurevych. "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Ed. by Kentaro Inui et al. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 3982–3992. DOI: 10.18653/v1/D19-1410. URL: https://aclanthology.org/D19-1410.

[15] Ruiyang Ren et al. *RocketQAv2: A Joint Training Method for Dense Passage Retrieval and Passage Re-ranking*. 2023. arXiv: 2110.07367 [cs.CL]. URL: https://arxiv.org/abs/2110.07367.

[16] Pranab Sahoo et al. *A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications*. 2024. arXiv: 2402.07927 [cs.AI]. URL: https://arxiv.org/abs/2402.07927.

[17] Samrat Sahoo et al. *The Universal NFT Vector Database: A Scaleable Vector Database for NFT Similarity Matching*. 2023. arXiv: 2303.12998 [cs.DB]. URL: https://arxiv.org/abs/2303.12998.

[18] Yuichi Sasazawa et al. *Text Retrieval with Multi-Stage Re-Ranking Models*. 2023. arXiv: 2311.07994 [cs.IR]. URL: https://arxiv.org/abs/2311.07994.

[19] Raymie Stata, Krishna Bharat, and Farzin Maghoul. "The Term Vector Database: fast access to indexing terms for Web pages". In: *Computer Networks* 33.1 (2000), pp. 247–255. ISSN: 1389-1286. DOI: https://doi.org/10.1016/S1389-1286(00)00046-3. URL: https://www.sciencedirect.com/science/article/pii/S1389128600000463.

[20] Hongjin Su et al. *One Embedder, Any Task: Instruction-Finetuned Text Embeddings*. 2023. arXiv: 2212.09741 [cs.CL]. URL: https://arxiv.org/abs/2212.09741.

[21] Hugo Touvron et al. *LLaMA: Open and Efficient Foundation Language Models*. 2023. arXiv: 2302.13971 [cs.CL]. URL: https://arxiv.org/abs/2302.13971.

[22] Liang Wang et al. *Improving Text Embeddings with Large Language Models*. 2024. arXiv: 2401.00368 [cs.CL]. URL: https://arxiv.org/abs/2401.00368.

Acknowledgments

Al termine di questo percorso, sento la necessità di esprimere un profondo riconoscimento verso chi mi ha accompagnato in questa ardua impresa, facendomi crescere come persona e come studente.
Un sentito ringraziamento a Loris Nanni, che mi ha seguito non solo nella tesi magistrale, ma anche in quella triennale. Grazie a lui ho sviluppato la mia passione per lintelligenza artificiale e, in particolare, per il Deep Learning e le Reti Generative.
Un ringraziamento speciale va anche alluniversità e alla città di Padova. In questi sei anni di permanenza qui mi sono trovato sempre benissimo e ho imparato ad apprezzare il fascino di questa magnifica città.
Ringrazio chiunque mi abbia sostenuto e aiutato durante questo viaggio, fatto di alti e bassi, di gioie e di momenti tristi, che, se condivisi, diventano sempre più leggeri.
Infine, ringrazio chiunque leggerà questo messaggio e chiunque leggerà questa tesi. Potrà sembrare insignificante per qualcuno, ma per me è un grande passo verso la maturità.

Grazie davvero con il cuore.
Mauro