



**POLITECNICO**  
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE  
E DELL'INFORMAZIONE

# *FindMyExpert: A Conversational Agent for Academic Collaboration Discovery*

TESI DI LAUREA MAGISTRALE IN  
COMPUTER SCIENCE - INGEGNERIA INFORMATICA

Author: **Cesar Adrian Carbajal Serrano**

Student ID: 258681

Advisor: Prof. Mark J. Carman

Co-advisors: Prof. Cristiana Bolchini

Academic Year: 2025-26



# Abstract

Within large institutions, the identification of academic experts and their collaboration networks is a complex task, as relevant information is distributed across heterogeneous data sources with different structures and interfaces. Recent advances in Artificial Intelligence and Large Language Models (LLMs) can facilitate the access to such a fragmented body of knowledge through natural language interfaces, to gather a unified view of researchers and their expertise. Indeed, it is necessary to properly exploit such tools, while handling their limitations, such as outdated information, hallucination, and lack of access to domain-specific data.

This thesis proposes *FindMyExpert*, an intelligent conversational system designed to support faculty at Politecnico di Milano in exploring collaboration networks and identifying researchers with relevant expertise. The system provides a unified natural language interface over multiple institutional sources, including publication repositories, thesis archives, and international agreement records. A data integration pipeline approach collects information into a centralized document database. A tool-based retrieval framework enables structured access to data by exposing sources through dedicated tools. An LLM selects the appropriate tools according to user requests and generating responses grounded in retrieved information. The system supports multi-turn conversations, allowing users to iteratively refine requests.

The proposed system is evaluated through experiments analyzing the impact of model size, quantization, reasoning mode, serving framework, tool provisioning strategy, and orchestration framework on tool invocation accuracy, response quality, and latency. Results show that tool-calling performance improves with larger models, that reasoning limited to tool selection provides a favorable accuracy-efficiency trade-off, and that the serving framework affects both latency and output reliability. Overall, the system demonstrates effective tool-based academic information retrieval and provides a foundation in real-world institutional environments.

**Keywords:** natural language processing, large language models, tool-augmented systems, conversational search, academic expert discovery



# Abstract in lingua italiana

L'operazione di identificare chi, all'interno di grandi istituzioni accademiche, ha collaborazioni con altre istituzioni e quali aree di ricerca siano coperte è complessa, poiché le informazioni rilevanti sono distribuite su fonti dati eterogenee, caratterizzate da strutture e interfacce differenti e soggette a rapida evoluzione. I recenti progressi nell'ambito dell'Intelligenza Artificiale e dei Large Language Model (LLM) possono facilitare l'accesso a un insieme frammentato di conoscenze tramite interfacce in linguaggio naturale. Per poterli utilizzare è però necessario gestirne i limiti, tra cui informazioni obsolete, allucinazioni e l'assenza di accesso a dati relativi a un dominio specifico.

Questa tesi propone FindMyExpert, un sistema conversazionale intelligente a supporto dell'esplorazione delle reti di collaborazione e dell'individuazione di persone con le competenze desiderate. Il sistema offre un'interfaccia unificata in linguaggio naturale per interrogare diverse fonti istituzionali, tra cui archivi di pubblicazioni, tesi e accordi internazionali. Le informazioni raccolte sono integrate in una base dati documentale centralizzata ed il framework è costituito da strumenti dedicati a diverse fonti dati. Un LLM processa la richiesta dell'utente e seleziona gli strumenti più appropriati, generando una risposta fondata sulle informazioni recuperate. Il sistema supporta conversazioni multi-turno, consentendo di affinare iterativamente le richieste e risposte.

Abbiamo valutato il sistema attraverso una campagna sperimentale che analizza numerosi aspetti, tra cui l'impatto della dimensione del modello, la modalità di ragionamento, la strategia di provisioning degli strumenti, valutandone gli effetti su accuratezza dell'invocazione, qualità delle risposte e latenza. I risultati mostrano che le prestazioni migliorano con modelli più grandi, che un ragionamento limitato alla selezione degli strumenti offre un buon compromesso tra accuratezza ed efficienza. Nel complesso, il sistema dimostra la fattibilità della soluzione, la qualità dell'architettura proposta e costituisce un buon prototipo per sviluppi futuri.

**Parole chiave:** natural language processing, large language models, sistemi basati su strumenti, ricerca orientata al dialogo, identificazione di esperti accademici



# Contents

|                                                                     |            |
|---------------------------------------------------------------------|------------|
| <b>Abstract</b>                                                     | <b>i</b>   |
| <b>Abstract in lingua italiana</b>                                  | <b>iii</b> |
| <b>Contents</b>                                                     | <b>v</b>   |
| <br>                                                                |            |
| <b>1 Introduction</b>                                               | <b>1</b>   |
| 1.1 Thesis goal . . . . .                                           | 2          |
| 1.2 Objectives . . . . .                                            | 3          |
| 1.3 Research questions . . . . .                                    | 3          |
| 1.4 Expected results . . . . .                                      | 4          |
| 1.5 Thesis structure . . . . .                                      | 5          |
| <br>                                                                |            |
| <b>2 Background and Related Work</b>                                | <b>7</b>   |
| 2.1 Natural Language Processing . . . . .                           | 7          |
| 2.2 Large Language Model . . . . .                                  | 8          |
| 2.3 Retrieval-Augmented Generation . . . . .                        | 10         |
| 2.4 Tool-Augmented systems . . . . .                                | 12         |
| 2.5 Conversational search and question answering . . . . .          | 13         |
| 2.5.1 Conversational interfaces for information retrieval . . . . . | 13         |
| 2.5.2 Multi-turn dialogue systems . . . . .                         | 14         |
| 2.5.3 Context management in long conversations . . . . .            | 15         |
| 2.6 Agentic systems . . . . .                                       | 16         |
| 2.7 Academic expert discovery systems . . . . .                     | 19         |
| 2.7.1 Existing approaches to researcher expert finding . . . . .    | 19         |
| 2.7.2 Knowledge graphs and collaboration network analysis . . . . . | 19         |
| 2.7.3 Academic recommendation systems . . . . .                     | 20         |
| 2.8 Institutional data integration . . . . .                        | 20         |
| 2.8.1 Heterogeneous academic data sources . . . . .                 | 20         |

|          |                                                           |           |
|----------|-----------------------------------------------------------|-----------|
| 2.8.2    | OAI-PMH . . . . .                                         | 21        |
| 2.8.3    | Data enrichment and normalization . . . . .               | 21        |
| <b>3</b> | <b>Problem definition and methodology</b>                 | <b>23</b> |
| 3.1      | Problem definition . . . . .                              | 23        |
| 3.2      | System requirements . . . . .                             | 24        |
| 3.3      | Design choices . . . . .                                  | 25        |
| <b>4</b> | <b>System design and implementation</b>                   | <b>29</b> |
| 4.1      | System architecture overview . . . . .                    | 29        |
| 4.2      | Data layer . . . . .                                      | 30        |
| 4.2.1    | Data integration pipeline . . . . .                       | 30        |
| 4.2.2    | Data storage . . . . .                                    | 31        |
| 4.3      | Tool-Based retrieval framework . . . . .                  | 32        |
| 4.4      | LLM integration . . . . .                                 | 33        |
| 4.5      | System configurations . . . . .                           | 34        |
| 4.6      | Implementation details . . . . .                          | 35        |
| 4.7      | Implementation and design considerations . . . . .        | 36        |
| 4.8      | User Interface . . . . .                                  | 37        |
| <b>5</b> | <b>Evaluation and Results</b>                             | <b>43</b> |
| 5.1      | Evaluation setup . . . . .                                | 43        |
| 5.2      | Evaluation metrics . . . . .                              | 44        |
| 5.3      | Experimental configurations . . . . .                     | 45        |
| 5.4      | Results . . . . .                                         | 46        |
| 5.4.1    | Impact of model size and quantization . . . . .           | 46        |
| 5.4.2    | Impact of model families and serving frameworks . . . . . | 48        |
| 5.4.3    | Tool integration strategies . . . . .                     | 51        |
| 5.4.4    | Tool orchestration frameworks . . . . .                   | 54        |
| 5.5      | Analysis . . . . .                                        | 55        |
| 5.6      | Discussion . . . . .                                      | 57        |
| <b>6</b> | <b>Conclusions</b>                                        | <b>61</b> |
|          | <b>Bibliography</b>                                       | <b>65</b> |

|                                                     |    |
|-----------------------------------------------------|----|
| A System Prompt: Tool descriptions in system prompt | 69 |
| B System Prompt: Native tool interface              | 73 |
| C FindMyExpert Source Code Repository               | 75 |
| List of Figures                                     | 77 |
| List of Tables                                      | 79 |
| Acknowledgements                                    | 81 |



# 1 | Introduction

In international academic institutions, collaboration plays a central role in fostering innovation and advancing research. The integration of diverse perspectives and resources across institutions enables the development of new knowledge and strengthens research networks.

Within large and reknown institutions such as Politecnico di Milano, the number of international collaborations is substantial. These relationships are reflected in co-written publications, co-supervised theses, joint projects, international events co-organised by faculty members, and formal agreements with partner institutions. However, this information is distributed across a broad body of data sources, each providing only a partial and fragmented view of the available collaborations. As a result, identifying who within the university has recent or ongoing collaborations with a specific party, in a given geographical area, or on a particular research topic, remains a complex and time-consuming process. For example, answering questions such as “Who at Politecnico has collaborations with Peruvian universities?”, “Who works on Natural Language Processing?” or “With whom has Cristiana Bolchini co-authored publications in Asia?” is not straightforward, despite the availability of multiple data sources.

At the same time, even though universities offer access to extensive digital resources, these are typically organized as separate systems, each with its own interface and data structure, requiring users to manually search, filter, and interpret information. Faculty members seeking to explore collaboration opportunities must therefore navigate multiple platforms to obtain a unified view of researchers and their expertise, which is often a non-trivial and inefficient task.

The rapid development of Artificial Intelligence (AI) has introduced new ways to interact with information. Recent advances in LLMs have significantly transformed natural language processing, enabling systems to generate human-like text and engage in complex conversational interactions. These capabilities contribute to a growing preference for natural language interfaces over traditional search systems, particularly in scenarios where users may not be familiar with the underlying data structure.

A key challenge in this context is that the relevance of information can vary depending on the user’s perspective and intent. Different users may consider the same data relevant in different ways, depending on whether they are looking for researchers with expertise in a specific topic, institutions in a particular region, or existing collaboration patterns. Moreover, users may differ in their focus: one may prioritize the number of shared contributions, whereas another may consider the duration of collaboration over time, with such preferences potentially emerging during interaction with the system. This diversity of user needs introduces complexity in designing systems that must flexibly accommodate multiple interpretations of relevance without requiring users to explicitly specify all constraints.

Despite their capabilities, LLMs exhibit important limitations. Their knowledge is bounded by the data on which they have been trained, and they lack access to domain-specific institutional information. Furthermore, these models can generate responses that are factually incorrect or incomplete, a phenomenon commonly referred to as *hallucination*. These limitations highlight the need for approaches that combine the capabilities of language models with mechanisms to access and integrate external heterogeneous sources of information.

To overcome these limitations, emerging approaches have explored the integration of LLMs with external tools and retrieval mechanisms, allowing models to access and process information dynamically. In this context, tool-augmented frameworks aim to improve factual reliability by grounding model outputs on interactions with multiple external systems and sources of information. This paradigm shifts the role of the language model from a static knowledge source to an orchestrator of external capabilities, responsible for coordinating information retrieval and reasoning to produce informed and accurate responses.

## 1.1. Thesis goal

Within this context, this thesis proposes *FindMyExpert*, an intelligent conversational system designed to support faculty at Politecnico di Milano in exploring collaboration networks and identifying researchers with relevant expertise.

The system addresses the challenge of fragmented academic information by providing a unified interface over multiple institutional data sources, including publication repositories, thesis archives, and records of international agreements. Rather than requiring users to navigate separate platforms, *FindMyExpert* allows them to express their information needs in natural language and receive consolidated responses that draw from all available sources.

A key design goal is the ability to support multi-turn interactions, where users can refine their requests and explore different aspects of the collaboration network in an iterative manner. This allows faculty to progressively narrow down their search and gain a deeper understanding of existing relationships without having to formulate complex queries.

The system relies on a tool-augmented architecture where an LLM acts as an orchestrator, selecting specialized tools to query the underlying data sources. The retrieved information is then synthesized into a coherent natural language response, ensuring that all factual content is grounded in explicitly retrieved data rather than model-internal knowledge.

## 1.2. Objectives

The main objective of this thesis is to design and develop a system that enables users to identify academic experts within Politecnico di Milano who are relevant in terms of recent or ongoing international collaborations, and to understand their areas of expertise and collaboration networks.

To this end, the specific goals this work achieves are:

- to integrate multiple heterogeneous academic data sources into a unified framework accessible to the end user;
- to design a set of tools for querying and retrieving relevant information from said sources;
- to enable an LLM to dynamically select and use such tools based on user requests;
- to support multi-turn conversations that allow users to progressively refine their information needs;

## 1.3. Research questions

Since the proposed work does not assume a fixed system architecture, it follows an iterative design process in which different architectural choices and configurations are explored, implemented, and evaluated. The design and evaluation of the system is guided by the research questions defined below.

RQ1. Can a tool-augmented LLM architecture provide a unified natural language interface, over multiple heterogeneous data sources, that supports expert discovery and collaboration network exploration?

RQ2. How do model size, quantization strategy, and reasoning mode influence the

trade-off between computational efficiency, reasoning performance, and response quality?

RQ3. How do model design choices and inference serving frameworks influence the performance of models in terms of efficiency, response quality, and system behavior?

RQ4. How do tool provisioning strategies influence accuracy and system-level behavior in LLM serving systems?

RQ5. How does the choice of tool orchestration framework influence response latency and execution reliability in language model systems with external tool integration?

## 1.4. Expected results

The expected outcomes of this work concern two main aspects: the integration of heterogeneous institutional data sources into a unified and consistent knowledge base, and the identification of an effective system design for combining LLMs with tool-based information retrieval.

The first aspect concerns the design of a data processing pipeline aimed at collecting, transforming, and integrating information from multiple institutional repositories into a centralized database. This component is expected to improve the accessibility and consistency of the information used by the system.

The second aspect involves investigating multiple approaches to combining LLMs with tool-based information retrieval. This includes variations in model configurations, prompting strategies, serving frameworks, and interaction mechanisms, addressing the research questions defined above, with the goal of identifying an effective overall system design.

The evaluation considers different system configurations to analyze their impact on performance. Quantitative metrics assess the system's ability to retrieve relevant and accurate information, while a qualitative user-centered evaluation focuses on the relevance and clarity of the generated responses, as well as its ability to support users in information-seeking and decision-making tasks.

Overall, the evaluation aims to assess the effectiveness of the proposed system in supporting expert discovery and collaboration network exploration in a clear, fast, and reliable manner.

## 1.5. Thesis structure

The remainder of this thesis is structured as follows. Chapter 2 presents the background and discusses related work highlighting the gaps this work aims to address. Chapter 3 defines the problem and describes the proposed approach. Chapter 4 details the system architecture and implementation. Chapter 5 presents the evaluation methodology and experimental results. Finally, Chapter 6 draws some conclusions and outlines directions for future work.



## 2 | Background and Related Work

This chapter introduces the foundational concepts and technologies on which this thesis is built, as well as relevant prior work in the areas of Natural Language Processing (NLP), LLMs, Retrieval-Augmented Generation (RAG), and tool-augmented systems. The discussion progresses from foundational concepts to more advanced systems that are closely related to the proposed approach.

### 2.1. Natural Language Processing

NLP is a field of artificial intelligence that focuses on the interaction between computers and human language. It encompasses a broad range of tasks that involve understanding, interpreting, and generating natural language text.

Over the past decades, NLP has evolved significantly moving from rule-based systems to statistical approaches and, more recently, to deep learning-based methods [13]. Early NLP systems relied on handcrafted rules and linguistic knowledge. The shift to statistical methods in the 1990s enabled systems to learn patterns from large texts, improving robustness and scalability. Techniques such as hidden Markov models, conditional random fields, and n-gram language models became standard tools for tasks including part-of-speech tagging, named entity recognition, and machine translation [18].

The introduction of neural network approaches revolutionized NLP, enabling end-to-end learning without the need for extensive feature engineering. Early approaches, such as recurrent neural networks and long short-term memory networks, allowed models to process sequential text data and capture contextual dependencies [11]. These architectures formed the foundation for subsequent advances, including the transformer architecture discussed in the following section.

Modern NLP encompasses a wide variety of tasks that are relevant to the system proposed in this thesis. Question answering systems enable users to retrieve precise information from text by interpreting natural language questions and generating targeted answers. Dialogue systems support interactive conversations between users and computers, requiring the

system to maintain context across multiple turns and generate contextually appropriate responses [13].

## 2.2. Large Language Model

Large Language Model are Deep Learning models designed to understand and generate human language through statistical patterns learned from large-scale textual data. The development of modern LLMs traces its origins to the transformer architecture which replaced recurrent neural networks with a mechanism called self-attention [33]. Self-attention enables the model to weigh the importance of different parts of the input sequence when processing each token, capturing long-range dependencies more effectively than previous architectures.

The training of LLMs typically proceeds in two phases. In the pre-training phase, the model learns general language representations by predicting the next token in a large corpus of unlabeled text. This phase exposes the model to diverse linguistic patterns, factual knowledge, and reasoning abilities present in the training data. In the fine-tuning phase, the pre-trained model is adapted to specific tasks or desired behaviors through training on smaller, labeled datasets. Common fine-tuning approaches include supervised fine-tuning, where the model learns to produce desired outputs for given inputs, and reinforcement learning from human feedback, where the model aligns its outputs with human preferences through reward-based optimization [24].

Modern LLMs demonstrate a broad set of capabilities that are relevant to this work. In natural language understanding, they can interpret user intent, extract relevant information from unstructured text, and summarize complex documents. In natural language generation, they produce coherent and contextually appropriate text across diverse domains and styles. Additionally, some models exhibit emergent reasoning capabilities, including chain-of-thought prompting, where the model generates intermediate reasoning steps before arriving at a final answer [36].

Despite their capabilities, LLMs exhibit fundamental limitations that constrain their applicability in knowledge-intensive domains. First, the knowledge encoded in LLMs is bounded by the data on which they are trained, meaning their knowledge becomes increasingly outdated as time passes. Second, LLMs are prone to hallucination, a phenomenon where the model generates fluent but factually incorrect or fabricated information [12]. This is particularly problematic in domains where the accuracy and reliability of information is essential. Third, LLMs lack direct access to proprietary or domain-specific institutional data, as their knowledge is limited to what was present in the pre-training

corpus. These limitations motivate the integration of LLMs with external retrieval and *tool-based* mechanisms, as discussed in subsequent sections.

The landscape of LLMs has evolved dramatically in recent years, marked by intense competition among major technology companies and a rapid escalation in model capabilities and scale. Companies at the forefront of LLM development include OpenAI, Google, Anthropic, Meta, xAI, and Mistral AI, each releasing successive generations of models that keep pushing their own boundaries of performance and capability [32].

Model size has grown substantially, with contemporary LLMs ranging from hundreds of millions to hundreds of billions of parameters. OpenAI’s GPT-5 series, Google’s Gemini family, Anthropic’s Claude models, Meta’s LLaMA series, xAI’s Grok models, and Mistral AI’s offerings represent the current frontier of language model capabilities. These models are trained on massive text corpora spanning books, articles, websites, and code repositories, enabling them to acquire broad world knowledge and diverse linguistic competencies.

The competition among AI companies has been referred to as an *AI arms race*, characterized by rapid iteration cycles, secrecy around model details, and fierce competition for market dominance [3]. This competition has driven significant investments in compute infrastructure, data acquisition, and research talent, resulting in accelerated progress in model capabilities. However, this race also raises concerns about safety, alignment, and the concentration of AI capabilities among a small number of organizations [34].

At the hardware level, NVIDIA has emerged as the dominant supplier of graphics processing units for training LLMs. The company’s Hopper architecture, featuring the H100 GPU, has become the de facto standard for LLM training due to its superior memory bandwidth and tensor performance. AMD has attempted to compete in this space, though NVIDIA maintains a commanding lead in market share for AI training deployments [10].

Leading models demonstrate strong performance across a wide range of benchmarks, including reasoning, code generation, mathematical problem solving, and multilingual understanding [6]. However, challenges remain significant. Inference costs for large-scale models remain high, limiting accessibility for certain applications. In addition, concerns about data privacy, copyright, and the environmental impact of training and deploying massive models have spurred ongoing debate within the research community and society at large.

## 2.3. Retrieval-Augmented Generation

RAG is an approach that addresses the knowledge limitations of LLMs by combining language models with external information retrieval systems. The core idea behind RAG is that when a user query requires specific information, the system retrieves relevant documents from an external knowledge source and incorporates them into the model’s context, allowing the generated response to be grounded in explicitly and newly retrieved content rather than solely in the model’s internal knowledge [15].

A typical RAG system consists of three main components. The first component is the retriever, which is responsible for identifying documents relevant to the user query from a knowledge base. Retrieval methods can be broadly categorized into *sparse* and *dense* approaches. Sparse retrieval methods, such as BM25, represent documents and queries as sparse term frequency vectors and score relevance based on lexical overlap [28]. Dense retrieval methods use learned embedding models to represent both queries and documents as dense vectors in a continuous space, enabling retrieval based on semantic similarity rather than exact term matching [14]. The second component is the knowledge base, a structured collection of documents or data against which queries are executed. The knowledge base may contain structured data, such as databases, or unstructured text, such as collections of documents. The third component is the generator, which takes the user query together with the retrieved content and produces a final natural language response. The generator is typically an LLM that conditions its output on both the original query and the retrieved information, ensuring that factual claims in the response are supported by the retrieved content [15]. An overview of these three components and their interaction is shown in Figure 2.1.

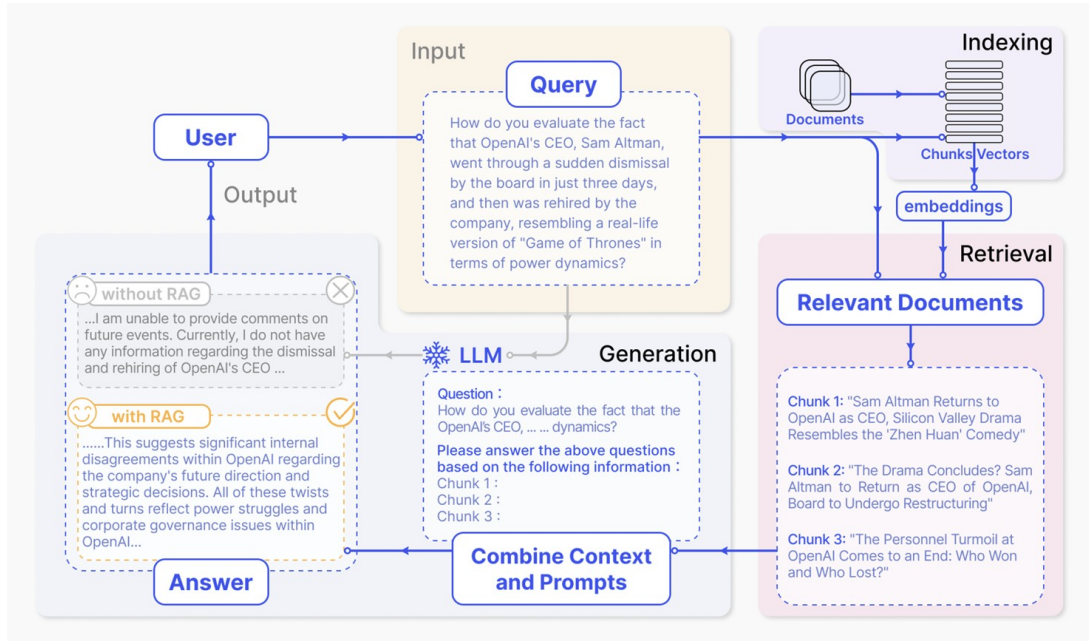


Figure 2.1: Overview of the RAG process. The pipeline consists of three main stages: (1) Indexing, where documents are encoded and stored in a vector database; (2) Retrieval, where the top- $k$  chunks most relevant are retrieved based on semantic similarity; (3) Generation, where the original query and retrieved chunks are provided to an LLM to produce the final answer. Adapted from [8].

RAG offers several advantages over standard generation approaches. By grounding responses in retrieved documents, RAG reduces the risk of hallucination and ensures that the generated content is factually accurate. Additionally, RAG enables access to information that was not present in the model's training data, such as recent publications or domain-specific institutional records. This makes RAG particularly suitable for applications where up-to-date and domain-specific knowledge is required [15].

However, basic RAG architectures exhibit important limitations that restrict their applicability to complex, multi-source scenarios. First, standard RAG systems typically perform single-pass retrieval followed by generation, without mechanisms for iterative refinement or multi-step reasoning. This limits their ability to handle complex queries that require gathering information from multiple sources or performing intermediate computations. Second, basic RAG does not support dynamic interaction with structured data sources, as it assumes a flat document collection as the knowledge base. Third, the retrieval and generation components operate in a pipeline without mutual feedback, meaning that the generator cannot guide the retrieval process based on partial results. These limitations motivate the exploration of more advanced architectures, such as tool-augmented systems,

which extend the RAG paradigm by enabling dynamic tool invocation and multi-step reasoning [15].

## 2.4. Tool-Augmented systems

Tool-augmented systems extend the RAG paradigm by enabling LLMs to interact with external tools and systems in a dynamic, multi-step manner. As shown in Figure 2.2, rather than treating retrieval as a fixed preprocessing step, tool-augmented systems allow the model to decide which tools to invoke, when to invoke them, and how to use the results, transforming the LLM from a passive knowledge source into an active orchestrator of external capabilities [29].

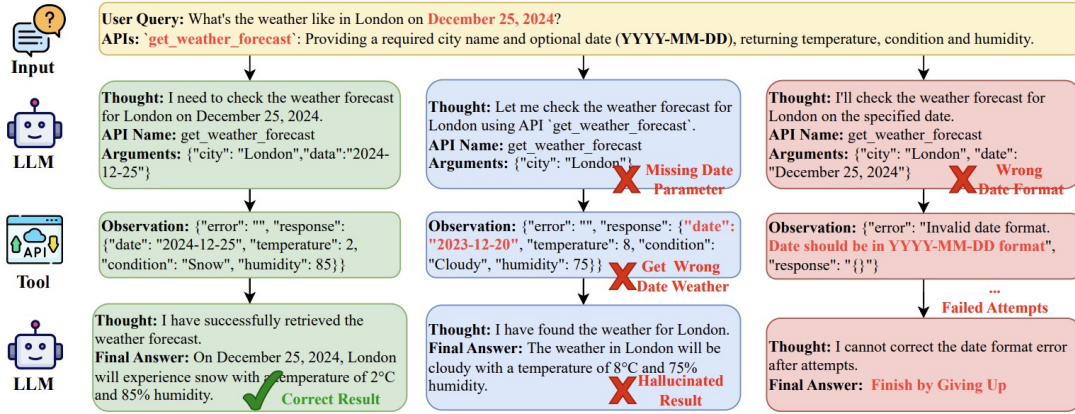


Figure 2.2: Tool-Augmented Systems examples: given user queries and tool, models generate tool calls for answers. Adapted from [17].

This approach is sometimes referred to as *function calling* or *tool scheming*, depending on the specific implementation [29]. Several approaches have been proposed for enabling tool use in LLMs. In the ReAct framework [38], the model alternates between reasoning steps and action steps, where reasoning steps generate internal thoughts about the current state and action steps invoke tools to gather external information. This interleaving of reasoning and action enables multi-step problem solving where later tool invocations can depend on the results of earlier ones. In function calling approaches, tools are defined as structured schemas that the model must produce in a specific format, and the system parses the model's output to extract tool invocations and execute them [23]. Function calling provides more predictable and parseable outputs compared to free-text reasoning traces, making it easier to integrate with external systems. More recent approaches combine chain-of-thought reasoning with structured tool schemas, enabling the model to plan its

tool usage while adhering to well-defined output formats [36].

Tool-augmented systems address several limitations of basic RAG architectures. First, by enabling dynamic tool selection, the model can choose which information sources to query based on the specific user request, rather than relying on a fixed retrieval pipeline. Second, tool-augmented systems support multi-step interactions where the output of one tool invocation informs subsequent tool calls, enabling complex queries that require aggregating information from multiple sources. Third, tool-augmented systems can interact not only with retrieval systems but also with any external system that exposes a programmatic interface, including databases, APIs, and computational tools [29].

Despite their advantages, tool-augmented systems introduce several challenges that must be addressed to ensure reliable operation. Tool selection accuracy is a critical concern: the model must correctly interpret the user’s intent and select the appropriate tool from the available options [38]. Incorrect tool selection leads to irrelevant or incorrect information being retrieved, degrading the quality of the final response. Output parsing is another challenge, as models must produce tool invocations in a structured format that can be reliably parsed by the system. Variations in model output, such as extra text before or after the structured invocation, can cause parsing failures and must be handled gracefully. Additionally, error propagation is a concern: if a tool invocation fails or returns unexpected results, the system must decide whether to retry, select an alternative tool, or respond to the user with an error message.

## 2.5. Conversational search and question answering

Conversational search and question answering represent paradigms in which users interact with information retrieval systems through natural language dialogue rather than keyword-based queries. This approach aims to bridge the gap between the expressiveness of natural language and the structured nature of underlying information systems, enabling users to articulate their information needs in a more intuitive manner.

### 2.5.1. Conversational interfaces for information retrieval

Traditional information retrieval systems rely on users formulating precise keyword queries to retrieve relevant documents or data. However, this approach often requires users to have prior knowledge of the data structure or vocabulary used in the target system, creating a barrier between user intent and system capability. Conversational search addresses this limitation by allowing users to express their information needs in natural language, much

as they would when consulting a human expert.

In a conversational search setting, the system maintains awareness of the ongoing dialogue context and can ask clarifying questions when user queries are ambiguous or incomplete. This interactive process enables a more precise understanding of user intent over multiple exchanges, reducing the gap between the user’s information need and the retrieved results. Recent approaches leverage LLMs to interpret user queries in context and generate more informed responses [21].

Conversational search systems can be broadly categorized into two types: *turn-level* systems and *session-based* systems. *Turn-level* systems process each query independently, treating every interaction as an isolated request without considering previous exchanges. While this approach simplifies system design and retrieval, it limits the system’s ability to interpret follow-up questions, resolve references, or adapt to evolving user needs. In contrast, *session-based* systems preserve conversational context across multiple interactions, enabling the system to reason about user intent over the course of a search session [21].

### 2.5.2. Multi-turn dialogue systems

Multi-turn dialogue systems are typically implemented using session-based architectures that maintain context across successive interactions. Unlike single-turn question answering systems, which generate responses based solely on the current query, multi-turn systems support sequences of related exchanges in which each turn may depend on information established earlier in the conversation. Users can refine, clarify, expand, or redirect their information needs over time, requiring the system to continuously update its understanding of the dialogue state [21].

A key challenge in multi-turn dialogue systems is tracking the evolving information need of the user across turns. As the conversation progresses, some aspects of the user’s need may remain constant while others change or become more specific. In other words, the system needs to distinguish between when prior context remains relevant, and when new aspects supersede previous ones [21].

Several approaches have been proposed for managing multi-turn interactions. Memory-augmented architectures maintain explicit representations of prior dialogue turns, allowing the model to attend to relevant historical context when generating responses [20]. Hierarchical dialogue managers process conversations at multiple levels of abstraction, from individual turns to overall conversation flow, enabling more structured reasoning about dialogue state [21]. Recent work on LLM-based dialogue systems has explored prompting strategies that encourage models to explicitly reason about dialogue context and user

intent before generating responses [36].

In the context of expert and collaboration discovery, multi-turn dialogue is particularly valuable because users often explore broad topics before narrowing down to specific researchers or collaborations. A user might begin by asking about research areas of interest, then progressively filter by institution, geographical region, and collaboration history, with each turn building upon the previous one.

### 2.5.3. Context management in long conversations

As conversations extend over many turns, managing context becomes increasingly challenging due to the limited context window of language models and the accumulation of irrelevant or redundant information in the dialogue history. Context management addresses the problem of selecting, compressing, and maintaining the most relevant parts of the conversation history for effective reasoning and response generation.

A common approach to context management is *attention-based* context selection, where the model learns to attend to the most relevant prior turns when generating a response [21]. This approach relies on learned relevance scores to filter dialogue history, discarding turns that are deemed less relevant to the current query. While effective for short conversations, these methods may struggle with very long dialogues where the relevant information is distributed across many turns.

Retrieval-based context management offers an alternative approach, where prior dialogue turns are stored in an external memory and selectively retrieved based on relevance to the current query [20]. This approach enables handling longer dialogue histories by retrieving only the most relevant turns, rather than relying on the model’s internal context window. However, retrieval-based methods require careful design of retrieval signals to ensure that relevant context is correctly identified and surfaced. Hierarchical context compression provides another approach, where dialogue history is summarized [21]. Lower levels capture fine-grained turn-by-turn details, while higher levels provide an overview of the conversation’s progression. This hierarchical structure allows the system to reason about both local and global context when generating responses.

Overall, effective context management ensures that the system can build upon prior exchanges without being overwhelmed by irrelevant information, ultimately improving the user experience and the accuracy of retrieved results.

## 2.6. Agentic systems

The concept of tool use in LLMs is rooted in the broader notion of agents, where an agent is a system that perceives its environment, reasons about possible actions, and executes those actions to achieve specific goals. As illustrated in Figure 2.3, an LLM-based agent can be understood as a closed-loop system consisting of a central reasoning model augmented with memory, cognitive capabilities, and access to external tools.

In this setting, the LLM first interprets a user-provided task, which serves as the core reasoning component. The model can retrieve and update information from memory, apply internal cognitive skills such as planning or decomposition, and decide whether to invoke external tools. These tools act as an interface between the agent and the environment, enabling actions such as API calls, database queries, or interactions with external systems. The environment then returns results, which are fed back into the agent to refine subsequent reasoning and decisions, forming an iterative perception–action loop.

The model is typically provided with structured descriptions of available tools, including their names, input parameters, and expected outputs, and is prompted to select and invoke them as needed based on the requirements of the task [38].

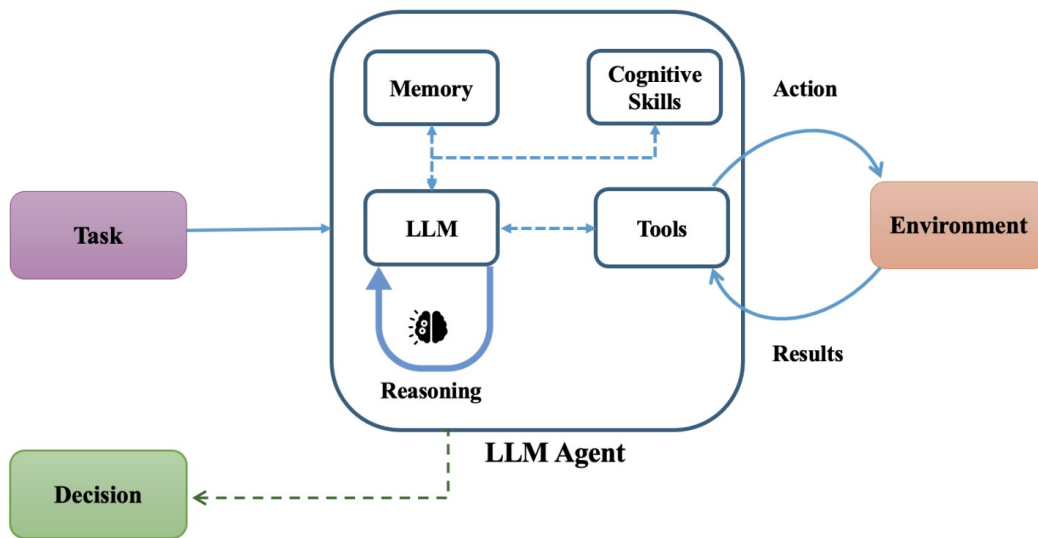


Figure 2.3: Core components and interaction loop of an LLM-based agent. The agent receives a task, reasons over it using an LLM augmented with memory and cognitive skills, optionally invokes external tools to interact with the environment, and iteratively refines decisions based on returned results. Adapted from [5].

Agentic systems represent an evolved class of LLM applications where models exhibit autonomous behavior, maintain state across interactions, and can pursue complex multi-step goals with minimal human intervention. Notable examples include AutoGPT [37], and ChatGPT Plugins and OpenAI’s function calling framework [23]. The former refers to AutoGPT, an autonomous agent that chains together LLM-generated reasoning steps to complete tasks without continuous user input, while the latter enables language models to interact with external services and APIs in a controlled and structured manner. Similarly, Microsoft’s research on the HuggingGPT system demonstrates how LLMs can serve as coordinators across multiple AI models and tools to accomplish multimodal tasks [30]. GitHub Copilot [9] leverages agentic capabilities to assist developers by autonomously completing code, suggesting whole functions, and interacting with development environments to streamline software engineering workflows. More recent systems such as Claude’s Computer Use [2] enable models to directly interact with graphical user interfaces, execute commands, and perform tasks on behalf of users in a manner comparable to human operators. These systems illustrate the shift from passive text generation toward active, goal-oriented behavior in LLMs.

Table 2.1 summarizes the main characteristics, advantages, and limitations of the approaches discussed in this section.

Table 2.1: Summary of background concepts

| Approach                       | Pros                                                                                                                                     | Limitations                                                                                                                                |
|--------------------------------|------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| Natural Language Processing    | Enables understanding and generation of human language; supports question answering, dialogue systems, and information extraction tasks. | Traditional approaches require extensive feature engineering and domain adaptation; performance depends on available linguistic resources. |
| Large Language Models          | Powerful natural language understanding and generation; supports summarization, reasoning, and diverse tasks                             | Prone to hallucinations and outdated knowledge; lacks access to proprietary or real-time information.                                      |
| Retrieval-Augmented Generation | Reduces hallucinations by grounding responses in retrieved documents; provides access to domain-specific knowledge.                      | Limited multi-step reasoning; relies on single-pass retrieval; no feedback loop between retrieval and generation.                          |
| Tool-Augmented LLMs            | Enable dynamic tool selection and multi-step reasoning; can interact with databases, APIs, and computational tools based on user needs.  | Tool selection errors can degrade performance; requires reliable output parsing; vulnerable to error propagation from external tools.      |
| Conversational search          | Allows users to express information needs naturally; supports clarification and contextual query understanding across multiple turns.    | Requires accurate intent interpretation; ambiguous queries may lead to incorrect retrieval without proper disambiguation.                  |
| Multi-Turn dialogue            | Maintains conversational context across interactions; supports iterative refinement of user needs over time.                             | Dialogue state management is complex; evolving user intent can be difficult to track accurately across turns.                              |
| Context management             | Preserves relevant information across extended conversations; improves response consistency and relevance.                               | Limited by context-window constraints; retrieval and summarization may omit important information.                                         |
| Agentic systems                | Enable autonomous execution of complex multi-step tasks; orchestrate tools with minimal human intervention.                              | Increased system complexity; challenges in planning, reliability and safety.                                                               |

## 2.7. Academic expert discovery systems

Academic expert discovery systems aim to help users identify and locate researchers with relevant expertise within academic institutions or across the broader research landscape. These systems address the challenge of finding the right experts when information is distributed across multiple sources and encoded in heterogeneous formats. The need for such systems has grown with the increasing volume of academic publications, collaborations, and institutional data, making manual expert discovery increasingly impractical.

### 2.7.1. Existing approaches to researcher expert finding

Traditional approaches to expert finding rely on keyword-based search over academic databases, where users query publication repositories using terms related to their area of interest. These systems typically rank researchers based on the frequency of keyword matches in their publication records, assuming that more publications in a domain indicate greater expertise. While straightforward to implement, such approaches fail to capture the depth and nuance of research expertise, as they do not distinguish between superficial keyword occurrences and genuine contributions to a field.

More advanced approaches incorporate semantic representations of research topics, enabling retrieval based on conceptual similarity rather than exact keyword matching. These systems use embedding models to represent both researcher profiles and user queries in a shared vector space, where proximity indicates relevance [31]. Some systems also exploit the structure of academic networks, using co-authorship patterns, citation relationships, and collaboration history to infer expertise and relevance [22].

Expert finding systems have also been developed specifically for institutional settings, where they must integrate information from multiple internal data sources such as publication repositories, thesis archives, and personnel databases. These systems face the additional challenge of dealing with fragmented and heterogeneous data that may not be consistently structured across sources [25].

### 2.7.2. Knowledge graphs and collaboration network analysis

Knowledge graphs provide a structured representation of academic entities and their relationships, enabling more sophisticated expert discovery through relationship traversal and pattern matching. In an academic knowledge graph, entities such as researchers, publications, institutions, and research topics are represented as nodes, while edges capture relationships such as co-authorship, citation, affiliation, and thematic similarity [35].

Collaboration network analysis leverages the graph structure to identify researchers who are likely to collaborate or who share common research interests. By analyzing network properties, these approaches can recommend relevant experts based on their proximity to other researchers or specific research topics [22].

Despite their utility, constructing and maintaining knowledge graphs requires significant effort in data integration. Relationships may be missing or inaccurate, and the dynamic nature of academic collaborations means that network representations quickly become outdated.

### 2.7.3. Academic recommendation systems

Academic recommendation systems extend the expert discovery paradigm by actively suggesting researchers, publications, or collaborations based on user profiles and preferences. These systems draw on techniques from collaborative filtering, content-based filtering, and hybrid approaches to generate personalized recommendations [4].

Recent work has explored the use of LLMs for academic recommendations, leveraging their ability to understand research semantics and generate contextually relevant suggestions. However, these approaches must contend with the same limitations that affect LLMs in other knowledge-intensive domains, including hallucination and outdated knowledge [7].

## 2.8. Institutional data integration

Institutional data integration refers to the process of combining data from multiple heterogeneous sources within an organization to provide a unified view of the information [16]. In academic settings, the goal is typically to enable comprehensive analysis and querying across sources that would otherwise remain isolated [1].

In the following, three aspects of institutional data integration at Politecnico di Milano are discussed, namely the heterogeneous nature of academic data sources and the mechanisms used for their aggregation and access, in particular on theses and publications data.

### 2.8.1. Heterogeneous academic data sources

Academic institutions maintain diverse data sources that capture different aspects of their research activities. Publication repositories such as IRIS, DSpace, or custom institutional systems store bibliographic records of research outputs, including journal articles, conference papers, and technical reports. These repositories typically expose metadata following

standards such as Dublin Core or MARC, though the specific implementation may vary across institutions.

Thesis archives contain records of graduate and undergraduate theses, including metadata such as titles, abstracts, author information, advisors, and defense dates [1]. These archives may follow different metadata schemas than publication repositories, leading to inconsistencies in how equivalent fields are represented.

Additional data sources may include institutional agreements, research project databases, grant management systems, patent records, and faculty profiles [16]. Each source presents its own challenges in terms of data quality, schema design, and access mechanisms.

### 2.8.2. OAI-PMH

The Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) is a widely adopted protocol for aggregating metadata from distributed repositories [19]. Many institutional repositories expose OAI-PMH endpoints that allow authorized clients to harvest metadata records programmatically. The protocol supports selective harvesting based on date ranges or sets, enabling efficient incremental updates.

OAI-PMH is designed primarily for metadata harvesting rather than complex querying. The protocol does not support advanced filtering, aggregation, or full-text search capabilities. Clients must typically harvest complete datasets and build their own indexing infrastructure to support sophisticated retrieval operations [19].

Despite its limitations, OAI-PMH remains an important mechanism for scholarly communication and data aggregation.

### 2.8.3. Data enrichment and normalization

Raw institutional data often requires substantial pre-processing before it can be used effectively [1]. Data enrichment refers to the process of enhancing existing records with additional information derived from external sources. For example, publication records may be enriched with citation counts, author affiliation information extracted from external databases, or topic classifications derived from semantic analysis.

Normalization involves standardizing the format and representation of data across sources. This may include normalizing author names to handle variations and duplicates, standardizing field names and date formats, and resolving inconsistencies in how research areas or geographic locations are represented [16].

These challenges are compounded when integrating data from multiple heterogeneous sources, as inconsistencies in one source may interact with inconsistencies in another [1]. A robust data integration pipeline must address these issues systematically while preserving the fidelity of the original data.

In this chapter, we have reviewed existing partial solutions to address our research questions. Based on the available methods, the data sources to be queried, and the expected query outcomes, the next chapter formalizes the problem addressed in this thesis and introduces the methodological foundations and requirements of the proposed system.

## 3 | Problem definition and methodology

This chapter formalizes the problem addressed in this thesis and presents the methodological foundations and design of the proposed system. The discussion begins with a clear definition of the problem of unified access to heterogeneous information sources through natural language interaction. It then outlines the functional and non-functional requirements that the proposed system must satisfy, including the specific data sources available and the use cases the system is designed to support. Finally, the chapter presents the high-level design choices and architectural decisions that guide the system implementation, providing an overview of the main components and their interactions.

### 3.1. Problem definition

This work addresses the problem of enabling unified access to heterogeneous information sources through natural language interaction. In many real-world scenarios, relevant information is distributed across multiple data sources, each exposing different interfaces and requiring different access mechanisms. This fragmentation makes it difficult for users to retrieve and combine information efficiently.

The goal is to design a system that enables users to discover experts within an academic institution by issuing natural language queries about research topics or areas of interest, and obtain coherent, consolidated responses identifying relevant researchers by dynamically accessing and integrating information from multiple institutional sources. Formally, given a user query expressing an information need regarding research expertise, the system must interpret the query, identify relevant information sources, retrieve or compute the required data—including researcher profiles, publications, and collaboration data—and generate a response identifying and describing the most relevant experts.

## 3.2. System requirements

We here define the functional and non-functional requirements of the proposed system. The requirements are derived from the goal of enabling unified access to heterogeneous institutional data sources through natural language queries.

### Functional requirements

The system should support the following actions:

- allow users to submit queries expressed in natural language;
- interpret the user query and determine whether additional information sources are required to produce an answer;
- support interaction with multiple heterogeneous data sources;
- select the relevant information sources based on the interpreted query;
- retrieve information from one or more selected sources; and
- generate a coherent natural language response based on the original query and the retrieved information.

### Non-functional requirements

Moreover, the system should exhibit the following characteristics:

- be composed of independent components with clear separation, allowing parts of the system to be modified without affecting the rest;
- support the integration of additional data sources and tools with minimal changes to the existing design;
- operate under the computational constraints of the available hardware infrastructure;
- provide responses within a suitable time frame for interactive use; and
- be structured in a way that facilitates long-term maintenance.

### Data sources

The system operates over heterogeneous institutional data sources provided by Politecnico di Milano. At present, these include the IRIS repository, the POLITesi thesis repository,

and structured data in spreadsheet format containing institutional agreements. These sources differ in structure, ranging from structured tabular data to metadata repositories.

## Use cases

The system must support use cases involving natural language interaction over the available data sources, including:

- Handling general academic informational queries that do not require access to external data. *Example: “Can you tell me the best ranked universities in Australia?”*
- Querying structured data sources using natural language. *Example: “List all institutional agreements with universities in Germany related to computer science programs.”*
- Retrieval of information contained in institutional sources. *Example: “Which Politecnico di Milano researchers have also supervised theses on artificial intelligence?”*
- Retrieval of information from external sources. *Example: “At which university does PhD Layla Hirsh currently work?”*

### 3.3. Design choices

This section presents the high-level design of the proposed system and the main architectural decisions adopted to address the problem of unified access to heterogeneous institutional data sources.

Figure 3.1 provides an overview of the proposed architecture and illustrates the interaction between the language model, retrieval tools, centralized data layer, and institutional data sources.

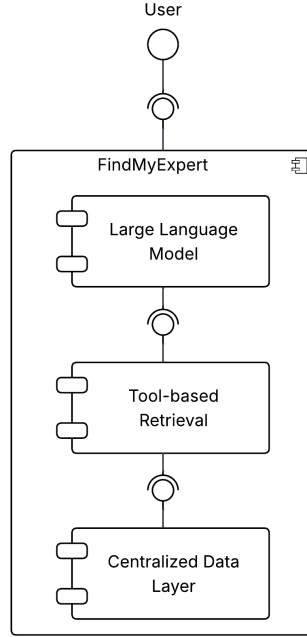


Figure 3.1: High-level architecture of the proposed system.

A key design decision is the adoption of a *centralized data layer*, which provides a unified interface over heterogeneous institutional sources and simplifies data access and management.

To handle the diversity of data formats and interaction patterns, the system adopts a *tool-based* retrieval approach, where each data source is accessed through dedicated tools that abstract underlying implementation details.

Finally, the system relies on an LLM to interpret user queries and determine whether tools are required to retrieve relevant information. The model contributes to both action selection and response generation based on the available context and retrieved data.

Based on the advantages and limitations of the available technologies presented in Chapter 2, the adopted system architecture allows one to: leverage the semantic understanding and reasoning capabilities of LLMs to interpret user queries expressed in natural language, even when they are ambiguous or incomplete; employ a *tool-based* retrieval approach to dynamically access heterogeneous institutional data sources, overcoming the limitations of basic RAG systems that operate over flat document collections; enable multi-step interactions where the model can iteratively query different sources based on intermediate results, supporting complex expert discovery queries that require aggregating information from multiple repositories; and maintain conversational context across turns, enabling users

to refine and expand their searches over the course of a session. However, several known critical aspects must be considered: the system's performance depends on the accuracy of tool selection, and incorrect tool choices may lead to irrelevant or incomplete results; output parsing errors may occur when models produce unexpected formats, requiring robust error handling. These aspects are addressed in detail in the following chapter, which describes the system architecture and its individual components.

This chapter defined the problem addressed in this work and outlined the functional and non-functional requirements of the proposed system. It also introduced the available data sources and the primary use cases motivating the system design. Finally, a high-level overview of the proposed architecture was presented, illustrating the main components and their interactions. The next chapter builds upon this foundation by providing a detailed description of the system architecture and its individual components, including their responsibilities and interactions.



## 4 | System design and implementation

This chapter provides a detailed description of the system design and implementation, building upon the problem definition, requirements, and high-level design choices presented in Chapter 3. The discussion begins with an overview of the system architecture and introduces the three main components that coordinate the interaction between the user, the language model, and the underlying data sources. It then describes the data integration pipeline responsible for harvesting, processing, and consolidating information from heterogeneous institutional sources into a centralized database. The chapter continues with a presentation of the data storage layer, followed by an in-depth analysis of the *tool-based* retrieval framework that enables structured access to the available data. The role of the language model as an orchestration component is examined, with particular attention to how it interprets user queries and determines tool invocations. Finally, the chapter describes the different system configurations explored during development and provides practical implementation details, including the technologies and frameworks adopted.

### 4.1. System architecture overview

The proposed system follows a modular architecture designed to support a flexible interaction between an LLM and multiple heterogeneous data sources.

As mentioned in the previous chapter, the system is structured around three main components: (i) a centralized data layer, (ii) a *tool-based* retrieval framework, and (iii) a language model responsible for orchestration and response generation.

At a high level, the system operates as follows. A user submits a natural language query through the interface. The query is processed by the LLM, which interprets the user intent and determines which tools are required to retrieve relevant information. These tools interact with the underlying data sources or the centralized database to extract the necessary information. The retrieved data is then passed back into the model, which

produces a final response in natural language for the user. This architecture is designed to address RQ1, providing a unified natural language interface over multiple heterogeneous academic data sources through a tool-augmented LLM.

This architecture allows data management to be handled independently from reasoning and interaction, while the language model acts as a coordination layer that dynamically combines multiple sources of information. The flexibility of this design allows new tools and data sources to be integrated without modifying the overall system structure.

## 4.2. Data layer

The data layer is responsible for acquiring, processing, and persisting heterogeneous information from multiple institutional sources. It provides a unified representation of the data consumed by the system, ensuring consistency, traceability, and efficient access across downstream components. The data layer is composed of two main components: a data integration pipeline and a persistence layer based on a document-oriented database.

### 4.2.1. Data integration pipeline

The data integration pipeline (shown in Figure 4.1) is designed to collect, process and consolidate information from multiple institutional sources into a centralized database. At the moment, it consists of three primary data sources: publications, theses, and agreements with other institutions.

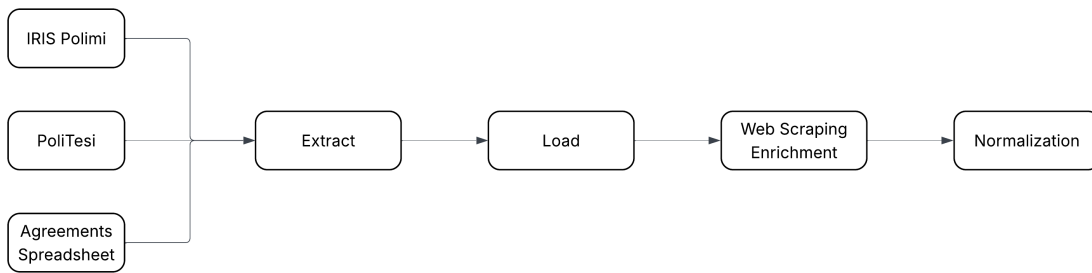


Figure 4.1: High-level architecture of the data integration pipeline

Publications are harvested from the IRIS Politecnico di Milano institutional repository [26], while theses are retrieved from POLITesi [27]. Both systems expose data through the OAI-PMH protocol, which is designed primarily for metadata harvesting rather than direct querying. Although OAI-PMH supports basic filtering options (e.g., by date or identifier), it does not provide efficient mechanisms for complex queries, advanced filtering,

or aggregations.

In addition to these sources, information about international agreements is obtained from an internal dataset provided in spreadsheet format. This dataset contains structured records describing institutional agreements.

To manage this process, an Extract–Load–Transform (ALT) approach is adopted. Data is first extracted from the external sources and loaded into a NoSQL database without applying significant transformations, preserving the original structure and fidelity of each source. This design choice enables flexibility in handling heterogeneous formats and reduces dependency on external system constraints. After ingestion, a data enrichment phase is applied to publication records. During this step, additional metadata is retrieved through web scraping to enrich the original records with complementary information not available in the source repositories. Subsequently, a transformation and normalization phase is applied across all datasets. This includes standardization of field names and the creation of derived attributes to improve queryability. In particular, country names are normalized to standardized representations to ensure consistency across sources, and author and entity names are processed to reduce variation and improve matching during queries.

This architecture supports efficient querying, indexing, and aggregation over heterogeneous datasets while avoiding the limitations imposed by direct interaction with external repositories. Moreover, it decouples data ingestion from data modeling, allowing the schema to evolve independently of the harvesting process.

The output of this integration pipeline is a unified and enriched data model that is stored in a NoSQL database. This persistence layer ensures scalability and flexibility in handling heterogeneous and evolving data structures while maintaining alignment with the ingestion process.

#### 4.2.2. Data storage

The system adopts MongoDB as its primary data storage solution to manage heterogeneous and semi-structured data originating from multiple institutional sources. The database is organized into four main collections: publications, theses and agreements.

The *publications* and *theses* collections store bibliographic records harvested from the IRIS and POLITesi repositories, respectively. Both collections follow the Dublin Core metadata standard, which provides a consistent representation of scholarly resources. Each document contains standardized metadata fields such as title, creators, contributors, date, language, and identifiers. In addition to raw metadata, these records may include enriched

attributes obtained during the data processing pipeline, such as affiliations, nationalities, and additional metadata extracted through scraping procedures. These enriched attributes are critical for our objective of retrieving collaborations with foreign institutions, as they enable the identification and analysis of international research connections.

The *agreements* collection stores structured information about institutional collaborations. Each document represents a bilateral or multilateral agreement between Politecnico di Milano and an external institution. The schema includes nested information describing the partner university and the associated academic program. This structure enables efficient querying of international collaboration agreements across different levels of granularity.

From a modeling perspective, each collection is designed to be self-contained and independent. No explicit relationships are enforced at the database level.

Overall, this data storage design separates concerns between bibliographic data and collaboration data, while providing a unified and extensible persistence layer that supports the tool-based retrieval framework of the system.

### 4.3. Tool-Based retrieval framework

The system adopts a *tool-based* retrieval framework to enable structured access to heterogeneous academic data sources stored in the database. In this architecture, the LLM does not interact directly with the database. Instead, it selects predefined tools that are responsible for executing queries over specific collections and returning structured results.

Each tool provides a well-defined interface between natural language requests and data retrieval operations. Given a user query, the model determines the most appropriate tool, based on the type of information required and generates the corresponding input parameters. The orchestration layer then processes the request, queries the relevant MongoDB collections, and returns structured outputs to the model.

The system implements a set of tools organized according to the main data sources available in the platform. These include tools for retrieving information about researchers, publications, theses, collaboration networks, and institutional agreements. Each tool is specialized for a single data source and operates independently, without relying on cross-collection queries.

The main tool categories are the following:

- **Researcher discovery tools:** used to identify POLIMI researchers based on collaboration-related filters such as country, institution, or research interest.

- **Research output tools:** used to retrieve publications and theses linked to a researcher.
- **Collaboration analysis tools:** used to extract information about co-authorship networks, affiliations, and collaboration patterns of individual researchers.
- **Institutional agreement tools:** used to query formal collaboration agreements between POLIMI and external universities based on diverse filters.

All tools return structured data that is passed back to the language model for synthesis. The model uses this information to generate a final natural language response to the user query. This design directly responds to RQ1, enabling structured access to heterogeneous academic data sources and supporting the discovery of researchers and collaboration networks required to achieve sufficient response quality for expert users.

#### 4.4. LLM integration

The system uses an LLM as the interface between natural language user queries and the underlying *tool-based* retrieval framework. While the model is not directly responsible for data access or query execution, it operates as a decision-making component that determines which tool should be invoked and with which parameters, based on the user request and the available context.

Given a user query, the model analyzes the input and produces a structured tool invocation request when required. This request specifies the selected tool and the corresponding input arguments. The execution layer interprets this output, executes the tool, and returns the resulting data to the model.

The LLM then uses the retrieved information to generate a final natural language response. This two-step process separates information retrieval from response generation, ensuring that all factual content is grounded in explicitly retrieved data rather than model-internal knowledge. This orchestration strategy responds to RQ1, grounding responses in explicitly retrieved data to reduce hallucinations and support reliable expert discovery and collaboration network exploration.

The system restricts each model decision to a single tool invocation. This design simplifies the execution logic and reduces ambiguity in tool selection. When additional information is required, it is obtained through subsequent user interactions rather than through chained tool execution.

The main advantage of this architecture is its ability to combine reliability with flexibility:

it reduces hallucinations by ensuring that all responses are grounded in institutional data, while still allowing users to express complex queries without needing to understand the underlying data structure.

## 4.5. System configurations

To assess the effectiveness of the proposed system, multiple configurations were explored by varying key components of the architecture. These configurations differ in terms of the selected language model, prompting strategy, and interaction mechanisms between the model and the *tool-based* retrieval framework.

**Model selection** Different LLMs were considered to evaluate the impact of model size and capability, allowing the analysis of how model characteristics influence tool selection accuracy and response quality. This configuration responds to RQ2, where model size, quantization strategy, and reasoning mode are varied to assess their influence on the trade-off between computational efficiency, reasoning performance, and response quality.

**Model serving configuration** Different model serving frameworks were considered to execute the LLM, including *vLLM* and *Ollama*. These frameworks provide different execution environments for model inference and deployment. Although model serving frameworks do not influence the reasoning process, retrieval strategy, or tool selection mechanisms of the system, they do affect how inference is performed at runtime, particularly in terms of response latency, streaming capabilities, resource management, and deployment flexibility. This responds to RQ3, where model design choices and serving frameworks are compared in terms of efficiency, response quality, and system behavior.

**Tool provisioning strategy** Different strategies for provisioning tools to the model were considered, including variations in the number of tools exposed, the level of detail in tool descriptions, and the format in which tool schemas are presented to the model. These variations aim to evaluate how the way tools are provisioned affects the model’s ability to accurately select the appropriate tool and generate coherent, well-grounded responses. This responds to RQ4, where different tool provisioning strategies are compared in terms of accuracy and system-level behavior.

**Tool orchestration framework** Different tool orchestration frameworks were considered for managing the interaction between the model and external tools, including variations in how tool calls are dispatched, executed, and how results are fed back into the model’s

context. These differences directly affect response latency, execution reliability, and the overall robustness of the multi-step interaction loop, providing insights into how the orchestration strategy influences the end-to-end performance of the system. This responds to RQ5, where the choice of tool orchestration framework is evaluated in terms of response latency and execution reliability.

By analyzing different serving configurations, the system can be evaluated not only in terms of accuracy and correctness but also with respect to efficiency and usability. The evaluation of these configurations is presented in Chapter 5.

## 4.6. Implementation details

The system was implemented using a modular Python-based backend responsible for coordinating data retrieval, tool execution, and interaction with the language model. The overall system architecture is illustrated in Figure 4.2, detailing the adopted technologies for the implementation of the main components.

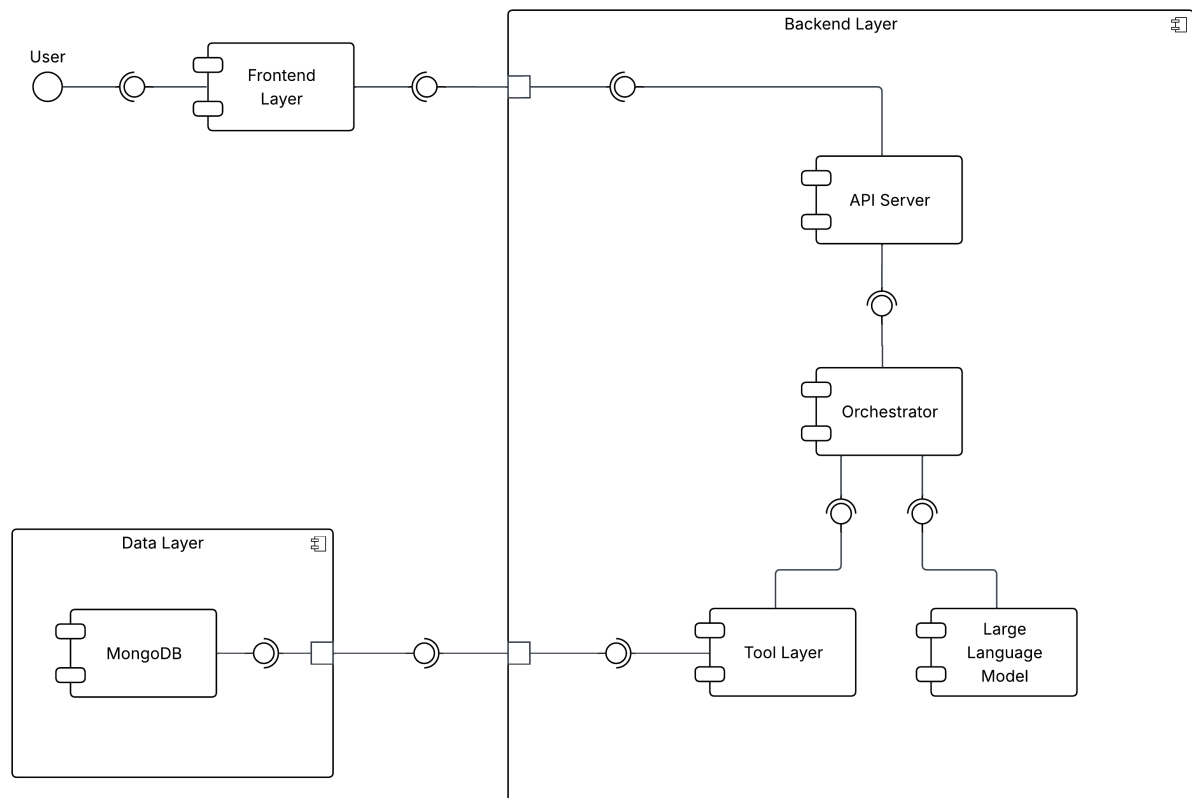


Figure 4.2: UML component diagram of the implemented system architecture

**Data storage** A MongoDB database is used to store all system data. The document-based structure allows flexible handling of heterogeneous data formats and supports efficient querying and indexing.

**Backend implementation** The core application logic is exposed as a REST API and implemented in Python and is responsible for coordinating the full execution pipeline of the system. This includes handling incoming user queries, interfacing with the language model to obtain tool invocation decisions, executing the corresponding tools, and assembling the final response returned to the user.

**Language model provider** The LLM is based on transformer architectures and can be executed through different inference setups depending on the experimental configuration. The system supports optimized serving through frameworks such as *vLLM* and *Ollama*.

**Frontend implementation** The system includes a web-based frontend that provides the user interface to interact with the assistant. It is responsible for capturing user queries, forwarding them to the backend API, and displaying the generated responses.

## 4.7. Implementation and design considerations

During the development of the system, preliminary user testing was conducted to assess how users interacted with the assistant and whether the available tools adequately supported their information needs. The observations collected during these tests influenced both the definition of the available tools and their integration into the overall system pipeline.

To support this iterative improvement process, the system was initially extended with a feedback collection mechanism. User interactions were stored during development, and each assistant response could receive a positive or negative feedback signal. This information enabled the analysis of successful interactions and, more importantly, the identification of cases where the system failed to satisfy user requirements. However, in the final deployment scenario, conversations are intended to be handled as transient sessions to improve scalability and reduce potential privacy and security concerns associated with persistent storage of user interactions.

The feedback collected during testing led to several implementation refinements. One of the resulting improvements was the introduction of data normalization mechanisms. In particular, inconsistencies were identified in the representation of some entities in the underlying data, where equivalent values could appear under different formats. For

example, the same country could be stored using different names, such as “China” and “Republic of China”, resulting in incomplete or inconsistent retrieval results. To address this issue and improve retrieval consistency, a normalization layer was incorporated before executing queries.

Furthermore, the evaluation process influenced the design of tool outputs. For instance, when retrieving researchers with collaborations with a specific country, returning all matching researchers could produce an excessive amount of information and negatively affect both usability and model performance. This was particularly evident for countries with a high number of collaborations, where a single collaboration may not represent a meaningful relationship. In contrast, for countries with fewer collaborations, such as Singapore, even a small number of results could be relevant. Therefore, the system was designed to return the top 20 collaborators for these queries. This strategy provides a balance between preserving relevant information and limiting the amount of data passed to the language model, avoiding unnecessary context growth and reducing memory consumption during inference.

## 4.8. User Interface

The web-based frontend provides the interface through which end users interact with the assistant. Its design follows a conversational paradigm, consistent with the overall system architecture, in which the user formulates a request in natural language and the LLM produces an answer grounded in the data retrieved through the *tool-based* framework. The interface is intended for non-technical users, who can express their information needs without any knowledge of the underlying data model, the available collections, or the tools invoked on their behalf.

The user is presented with a login screen where credentials are entered before reaching the conversational interface. Once authenticated, the user reaches the main conversational view. When no conversation is active, the system displays an empty state, as illustrated in Figure 4.3. The view is centered around the prompt “What would you like to know?” and is accompanied by a short description of the types of queries supported by the system, such as finding researchers by topic, institution, country, or collaborations. This onboarding element is designed to establish appropriate expectations about the system’s scope and guide users toward meaningful initial queries.

Additionally, the interface provides access to past interactions through a side panel listing previous conversations. From this panel, the user can resume an existing conversation, rename it, or delete it through a confirmation dialog that prevents accidental removal.

A control to start a new conversation is always available from the main view. This history management supported the development and evaluation activities described earlier, allowing users to revisit and compare interactions over time. The layout is also responsive: on smaller screens the side panel collapses into a drawer that can be opened on demand, keeping the conversational area as the primary focus of the interaction.

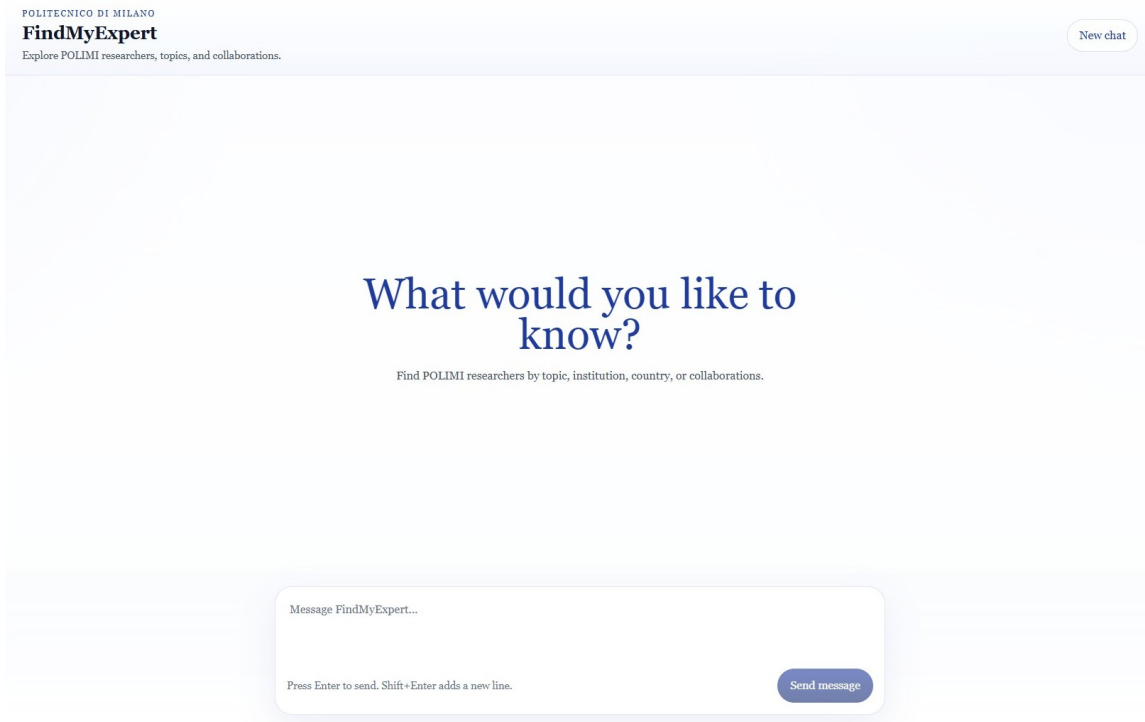


Figure 4.3: Empty state of the conversational interface

The interaction unfolds through a message-based thread, as illustrated in Figure 4.4. User messages and assistant responses are presented in a chat-like layout, with the user’s input displayed on one side and the assistant’s reply on the other. The input area supports multi-line editing, following interaction conventions established by widely used conversational applications. While the assistant is generating a response, a progress indicator with a “Thinking...” label is displayed to provide immediate visual feedback that the request has been received and is being processed. This helps manage the user’s perception of the waiting time associated with model inference and tool execution.

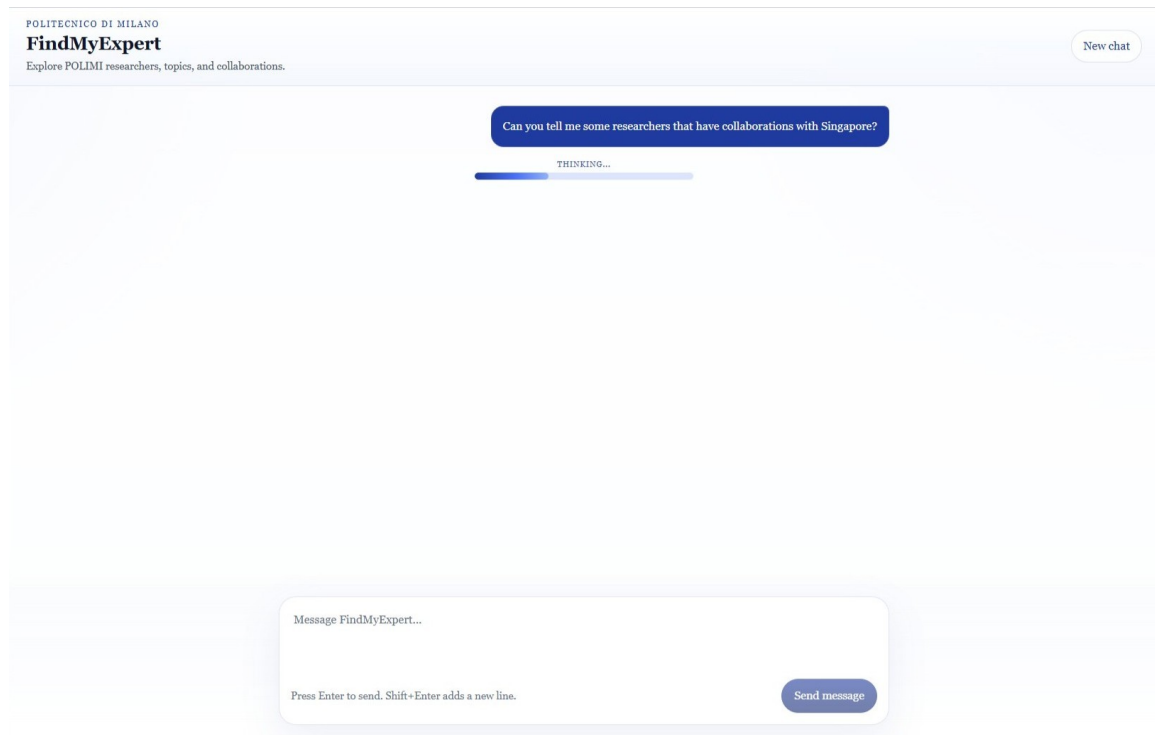


Figure 4.4: An active conversation, showing the progress indicator

Finally, each assistant response is accompanied by a lightweight feedback control that allows users to provide a positive, neutral, or negative evaluation without disrupting the reading flow, as shown in Figure 4.5. This control serves as the user-facing counterpart to the feedback collection mechanism described earlier and supports an iterative evaluation process that informed several system refinements.

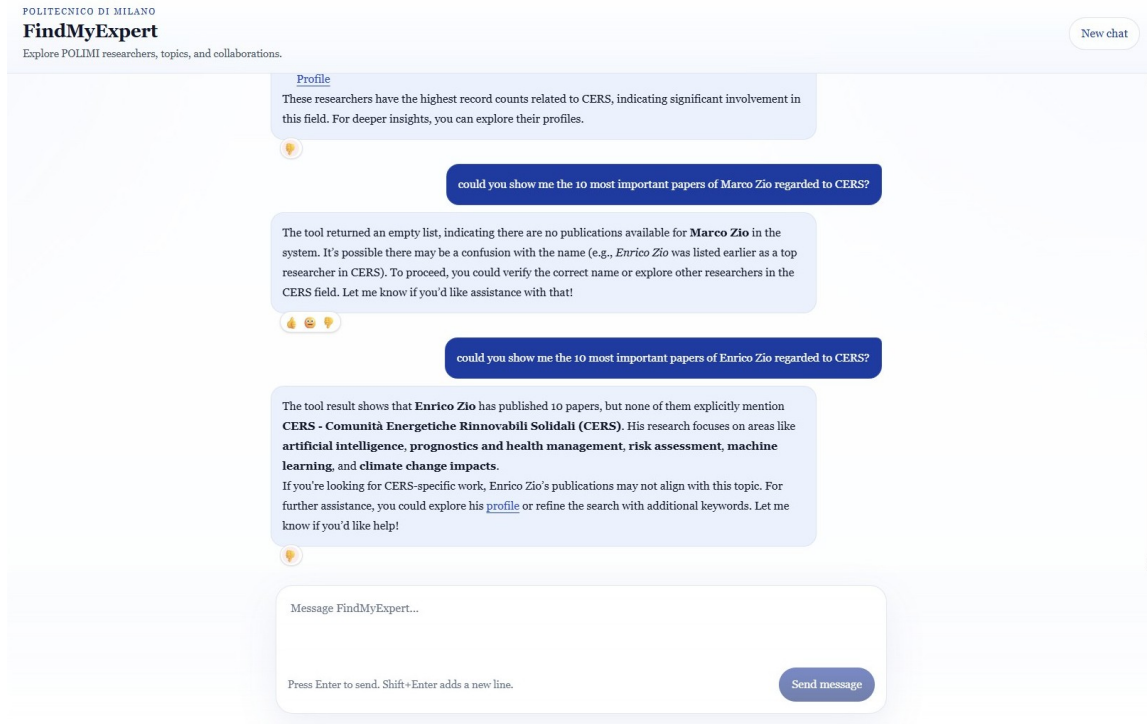


Figure 4.5: Feedback associated with an assistant response

Overall, the user interface is designed to keep the interaction simple and centered on the exchange, while supporting the a small set of features that support the development.

This chapter presented the detailed design and implementation of the proposed system. It described the modular architecture composed of the centralized data layer, the *tool-based* retrieval framework, and the language model orchestration component. The data integration pipeline was discussed in detail, covering the ELT-based harvesting, enrichment, and normalization of publications, theses, and institutional agreements from heterogeneous institutional sources into a centralized MongoDB database. The data storage layer was presented, along with the organization of collections following the Dublin Core metadata standard and the rationale behind the chosen schema design. The *tool-based* retrieval framework was examined, detailing how tools abstract data access and enable the language model to retrieve structured information from the underlying database. The role of the language model as a decision-making component responsible for tool selection and response generation was analyzed, with particular attention to the single-tool invocation strategy adopted to simplify execution logic. Different system configurations were presented, highlighting the variations in model selection, prompting strategies, interaction mechanisms, and serving frameworks explored during development. Practical implementation details

were provided, describing the technologies and frameworks adopted to realize the system. Finally, the user interface was presented, illustrating the conversational paradigm and feedback collection mechanisms through which end users interact with the assistant.



# 5 | Evaluation and Results

This chapter evaluates the proposed system by measuring its ability to interpret queries, select the correct tools, and generate accurate responses. The evaluation analyzes multiple performance dimensions—including tool invocation accuracy, response quality, and latency—across different model configurations, serving frameworks, and system architectures. The chapter describes the evaluation setup, defines the metrics used, presents the experimental configurations, and reports results across several stages. It concludes with an analysis of key findings and a discussion of the evaluation’s strengths and limitations. The central objective of this evaluation is to address RQ1, assessing whether the proposed tool-augmented LLM architecture provides a unified natural language interface over multiple heterogeneous academic data sources, achieving a response quality sufficient to support expert discovery and collaboration network exploration.

## 5.1. Evaluation setup

The evaluation of the proposed system was designed to assess its ability to correctly interpret user queries, invoke appropriate tools, and generate accurate responses. To this end, a dedicated test set was manually constructed.

The test set consists of a collection of queries specifically designed to trigger the different tools available in the system. Some requests were designed to require the use of at least one tool. This approach allows for a targeted evaluation of the system’s tool-calling capabilities, which represent a core component of the architecture.

The queries include also straightforward factual questions. This setup enables the analysis of both the reasoning capabilities of the language model and the correctness of tool selection.

All experiments were conducted using computational resources provided by Politecnico di Milano, specifically leveraging an NVIDIA T4 GPU with 16 GB of VRAM. This constraint significantly impacted the experimental design by restricting the maximum size of deployable models. As a result, the evaluation is limited to models that can fit

within 16 GB of GPU memory, typically in the range of 1B–7B parameters depending on quantization and inference configuration.

## 5.2. Evaluation metrics

The evaluation of the proposed system focuses on correctness of tool usage, quality of generated responses and latency. Given the integration of a language model with external tools, a set of metrics is defined to capture different aspects of system behavior.

**Tool Invocation Accuracy (TIA)** Tool invocation accuracy evaluates whether the language model calls a tool whenever it is necessary and selects the appropriate tool for a given query, since for each query in the test set the expected tool is known a-priori. A prediction is considered correct if the invoked tool matches the expected format, matches the expected tool and matches the expected arguments. This metric directly reflects the model’s ability to correctly interpret the query and utilize external functionalities.

**Grounded Answer Score (GAS)** The quality of the generated responses is evaluated with respect to the tool output or previous context used to produce them. To perform this evaluation, an LLM-as-a-Judge approach is adopted, where the xAI Grok-4-1-fast-reasoning model is used as an automated evaluator. This model is a fast, reasoning-optimized variant of the Grok family designed to produce low-latency but structured analytical outputs, making it suitable for scalable evaluation settings. Although no official figure is provided by the developer, the model is assumed to be in the tens to hundreds of billions of parameters range.

The judge model is prompted with the input query, the tool output, and the generated answer, and assigns a numerical score reflecting the overall quality of the response. To ensure consistency across experiments, the evaluation is performed using a fixed prompt and generation settings.

**Latency (LAT)** Latency is measured as the total time required to generate a response, including both model inference and tool execution. This metric is used to compare the efficiency of different model configurations and serving strategies.

**Metric prioritization** Tool invocation accuracy (TIA) is treated as the primary metric, as correct tool selection is essential for system functionality. The remaining metrics are considered equally important secondary objectives, capturing complementary aspects of

system performance. Specifically, latency reflects deployment efficiency and user experience, while grounded answer score measures response quality conditioned on tool outputs.

Accordingly, system performance is evaluated as a trade-off between maximizing TIA while jointly optimizing LAT and GAS.

### 5.3. Experimental configurations

The experimental evaluation is designed to analyze the impact of different design choices across four main dimensions: language model selection, inference serving frameworks, tool provisioning strategies and tool orchestration frameworks. This allows a systematic comparison of performance trade-offs in terms of accuracy, tool usage, and latency.

**Large Language Models** Multiple open-source large language models are evaluated, covering different model families and parameter sizes. The goal is to assess how model capacity influences reasoning ability, tool selection accuracy, and response quality. In this context, reasoning ability is analyzed both across standard inference and an optional “thinking” mode, which enables additional intermediate computation during generation to support multi-step problem solving. In addition to full-precision models, quantized variants are also tested in order to evaluate the impact of reduced numerical precision on performance.

**Serving frameworks** Three different inference serving approaches are compared. First, Ollama is evaluated as a lightweight local serving solution optimized for ease of deployment. Third, vLLM is included as a high-performance inference engine.

**Tool integration strategies** Two different strategies for enabling tool usage are evaluated. The first approach relies on prompt-based tool specification, where tool descriptions are embedded in the system prompt and the model generates tool calls based on these instructions without an explicit tool selection mechanism. The second approach uses structured tool selection, where tools are formally defined as callable functions and the model selects from the available tools through a dedicated interface.

**Tool orchestration frameworks** The system is implemented using two different orchestration approaches. A custom Python-based framework provides full control over the execution flow, including tool invocation and response handling. In contrast, a LangChain-based implementation is used to evaluate the impact of higher-level abstractions on

development complexity and system behavior. This comparison allows analysis of the trade-off between flexibility and ease of integration.

## 5.4. Results

This section presents the structure used to report and compare the experimental results obtained from the different system configurations described in the previous section.

### 5.4.1. Impact of model size and quantization

Model size is one of the primary factors influencing the capabilities of LLMs. In general, larger models contain a greater number of parameters, allowing them to capture more complex linguistic patterns and contextual relationships during training. In addition to model size, this study considers two other key factors: quantization strategy and reasoning mode. Quantization refers to the reduction of numerical precision in model weights and activations (e.g., from full precision to lower-bit representations), which can substantially reduce memory usage and improve inference speed, but may also introduce approximation errors that affect model performance. Reasoning mode refers to an inference-time configuration that enables additional computational deliberation during generation, improving multi-step problem solving and reasoning performance at the cost of increased latency. However, increased model size and the use of reasoning mode both introduce additional computational and memory requirements, which can negatively impact inference efficiency and deployment constraints, particularly in GPU-limited settings.

This subsection evaluates how model size, quantization strategy, and reasoning mode jointly affect system performance under a unified serving framework to ensure a fair comparison. In particular, it seeks to answer the following research question:

**RQ2.** How do model size, quantization strategy, and reasoning mode influence the trade-off between computational efficiency, reasoning performance, and response quality?

Table 5.1: Performance across model sizes and quantization levels.

| Model Configuration    | Reasoning | TIA   | GAS   | LAT   |
|------------------------|-----------|-------|-------|-------|
| Qwen3-1.7B             | Disabled  | 46.8% | 93.4% | 5.1s  |
| Qwen3-1.7B             | Enabled   | 73.3% | 90.8% | 16.2s |
| Qwen3-4B               | Disabled  | 79.7% | 98.2% | 10.6s |
| Qwen3-4B               | Enabled   | 86.9% | 96.3% | 38.0s |
| Qwen3-4B               | Mixed     | 90.9% | 97.7% | 28.3s |
| Qwen3-4B-Instruct-2507 | Disabled  | 84.5% | 95.9% | 9.4s  |
| Qwen3-8B-AWQ           | Disabled  | 80.9% | 94.1% | 6.8s  |
| Qwen3-8B-AWQ           | Enabled   | 91.1% | 96.2% | 25.6s |
| Qwen3-8B-AWQ           | Mixed     | 85.5% | 98.6% | 19.1s |
| Qwen3-8B-FP8           | Disabled  | 81.4% | 98.7% | 9.6s  |
| Qwen3-8B-FP8           | Enabled   | 91.2% | 98.5% | 34.1s |
| Qwen3-8B-FP8           | Mixed     | 91.5% | 98.5% | 25.3s |

The results in Table 5.1 highlight that TIA is mainly driven by model size and reasoning usage during tool selection. In particular, larger models consistently improve performance, with 8B configurations achieving the highest accuracy.

The introduction of mixed reasoning mode provides an important insight. Mixed reasoning mode applies reasoning only during tool invocation, while final response generation is performed without reasoning mode. These results show reasoning is most beneficial when applied during tool invocation rather than full response generation. This configuration achieves competitive TIA while significantly reducing latency compared to fully reasoning-enabled generation, demonstrating that inference-time deliberation is most critical for planning and tool selection.

Latency is predominantly influenced by the extent of reasoning usage, with full reasoning mode introducing the highest overhead, while mixed configurations provide a more efficient trade-off. Quantization strategies do not have a significant impact on either accuracy or efficiency, although higher-precision configurations exhibit a slight tendency toward improved performance without introducing a substantial trade-off in computational cost.

Overall, the results suggest that separating reasoning between tool invocation and response generation yields a favorable balance between accuracy and efficiency, with mixed reasoning mode providing the most effective trade-off under the evaluated metrics.

Furthermore, the best-performing model is Qwen3-8B-FP8 with reasoning enabled during tool invocation, achieving an average TIA above 90% and an average GAS exceeding 98%.

Although its average latency exceeds 25 seconds, this trade-off is considered acceptable in order to ensure that users receive the highest possible quality of information. However, the Qwen3-4B model also produced strong results and represents a practical model size worth considering.

#### 5.4.2. Impact of model families and serving frameworks

Based on the results obtained in the previous stage, 4B and quantized 8B models were selected as reference model sizes for further evaluation. These model scales provide a balance between computational efficiency and performance, making them appropriate candidates for comparative analysis.

This stage focuses on comparing different model families with similar parameter sizes in order to investigate how variations in model design affect model behavior. While models may contain a comparable number of parameters, differences in architectural choices, training data, alignment strategies, and optimization techniques can lead to significant differences in performance and resource utilization. Evaluating models of similar scale allows for a more controlled comparison and helps isolate the influence of model-specific characteristics.

In addition to model-level comparisons, this stage also evaluates the impact of different serving frameworks on inference performance. The serving infrastructure plays a significant role in determining deployment efficiency, affecting factors such as response time, throughput, memory usage, and hardware utilization. In particular, two serving frameworks are considered in this study: vLLM and Ollama. Ollama offers a simplified deployment environment focused on ease of use and local model serving, reducing setup complexity while providing an accessible interface for running LLMs. vLLM is designed for high-throughput and memory-efficient inference, improving GPU utilization through optimized KV-cache management and continuous batching.

This subsection evaluates the differences in performance across model families and the impact of serving frameworks on system behavior. In particular, this subsection addresses the following research question:

**RQ3.** How do model design choices and inference serving frameworks (vLLM and Ollama) influence the performance of models in terms of efficiency, response quality, and system behavior?

For this reason, different model families are selected for evaluation. We begin with Qwen, which has already been used in previous experiments and therefore serves as a consistent

baseline for comparison across frameworks. To ensure continuity with the previous evaluation stage, the same vLLM configuration, `Qwen3-8B-FP8`, is used as the reference model. Its evaluation is then extended by comparing this configuration with an equivalent deployment through Ollama, enabling an assessment of how different inference frameworks affect performance under similar model scales. For Ollama, the closest equivalent available is `qwen3:8b-q8_0`.

In addition, we include Llama 3.1 as a second model family to examine differences across models developed using distinct training and alignment approaches. For the vLLM evaluation, we adopt `Meta-Llama-3.1-8B-Instruct-FP8` as the reference configuration. This setup is then compared against its closest available counterpart in Ollama, `llama3.1:8b-instruct-q8_0`.

We also include Phi 3.5 as a smaller-scale model family to examine performance behavior at the lower end of the selected model range. For the vLLM evaluation, we use `Phi-3.5-mini-instruct` as the reference configuration. This model is then compared with its closest available counterpart in Ollama, `phi3.5:3.8b-mini-instruct-fp16`.

Finally, we consider Granite 4.1 as another model family to broaden the comparison across different model development strategies. For this evaluation, we use the `granite-4.1-3b` model in vLLM. This model represents a compact instruction-tuned model designed for efficient deployment while maintaining strong reasoning and language capabilities. It is compared with its closest available counterpart in Ollama, `granite4.1:3b-q8_0`.

Together, these model families enable a structured comparison across both model design and serving frameworks, allowing us to analyze how differences in model characteristics and inference infrastructure influence efficiency, response quality, and overall system behavior.

| Model                          | TIA   |        | GAS   |        | LAT    |        |
|--------------------------------|-------|--------|-------|--------|--------|--------|
|                                | vLLM  | Ollama | vLLM  | Ollama | vLLM   | Ollama |
| Qwen3-8B-FP8                   | 91.5% | 93.6%  | 98.5% | 92.6%  | 25.3 s | 30.4 s |
| Meta-Llama-3.1-8B-Instruct-FP8 | 74.0% | 49.6%  | 91.6% | 68.1%  | 9.4 s  | 12.9 s |
| Phi-3.5-mini-instruct          | 40.0% | 30.0%  | 75.3% | 20.6%  | 24.9 s | 30.7 s |
| granite-4.1-3b                 | 52.0% | 60.6%  | 91.7% | 82.8%  | 4.7 s  | 4.6 s  |

Table 5.2: Performance across model families and serving frameworks

The results presented in Table 5.2 across TIA, GAS, and LAT demonstrate clear differences between model families, serving frameworks, and tool integration strategies. Overall, vLLM

tends to provide higher performance and lower latency compared to Ollama, although the magnitude of this difference varies depending on the model family.

In terms of TIA, Qwen3-8B-FP8 achieves consistently high performance in both frameworks, with 91.5% under vLLM and 93.6% under Ollama. In contrast, Meta-Llama-3.1-8B-Instruct-FP8 shows a substantial drop when deployed through Ollama (49.6%) compared to vLLM (74.0%). A similar pattern is observed for Phi-3.5-mini-instruct, where performance decreases from 40.0% in vLLM to 30.0% in Ollama. Interestingly, Granite-4.1-3b exhibits a reversed trend, with slightly better performance in Ollama (60.6%) than in vLLM (52.0%). Beyond the quantitative results, a qualitative inspection of the generated outputs reveals notable differences in response behavior between the two serving frameworks. In particular, Ollama models tend to produce more conversational and verbose outputs. While this can be beneficial in open-ended dialogue settings, it introduces challenges in structured generation scenarios.

A recurring issue observed with Ollama outputs is the tendency to over-generate text even in cases where a strictly formatted response is required. In several instances, the models produced correctly structured JSON suitable for tool calling, but subsequently appended additional natural language explanations or commentary. This behavior is problematic in tool-calling pipelines, where strict adherence to output schemas is required for reliable parsing and execution.

In contrast, vLLM-based deployments exhibit more controlled and deterministic output formatting. Responses are generally more concise and better aligned with formatting constraints, reducing the likelihood of invalid or partially invalid structured outputs. As a result, vLLM appears more suitable for scenarios requiring strict schema adherence, such as function calling or automated tool integration.

Overall, these observations suggest that, beyond performance metrics such as accuracy and latency, the models available for each serving framework also significantly influences output style and compliance with structured output requirements. This factor is particularly important in production environments where models are expected to reliably generate machine-readable outputs.

The GAS metric further reinforces these observations. Qwen3-8B-FP8 achieves high performance in both frameworks (98.5% in vLLM and 92.6% in Ollama), while Llama 3.1 again shows a large degradation under Ollama (68.1% compared to 91.6% in vLLM). Phi-3.5-mini-instruct experiences the most pronounced drop, from 75.3% in vLLM to 20.6% in Ollama, where a significant number of hallucinated or factually inconsistent outputs were observed. Granite-4.1-3b remains relatively stable, with a moderate decrease

from 91.7% to 82.8%.

Latency results indicate that vLLM generally provides lower inference times across most configurations. This is particularly evident for Llama 3.1 (9.4s in vLLM vs. 12.9s in Ollama) and Phi-3.5-mini-instruct (24.9s vs. 30.7s). Qwen3-8B-FP8 also shows improved latency under vLLM (25.3s compared to 30.4s in Ollama). However, for Granite-4.1-3b, both frameworks perform similarly, with Ollama being marginally faster (4.6s vs. 4.7s), suggesting that for smaller models the advantage of optimized batching and GPU utilization becomes less pronounced.

Overall, the results suggest that vLLM provides more consistent performance across model families, particularly in terms of both quality and latency. Ollama, while offering competitive results for certain models such as Qwen and Granite, exhibits higher variability and stronger performance degradation for more sensitive or smaller-scale models. These findings highlight that both model selection and serving framework play a significant role in determining overall system behavior, and their interaction should be considered when optimizing deployment configurations.

An additional factor considered in this evaluation is the use of explicit reasoning modes available in some model families. For the Qwen models, a mixed thinking configuration was applied to assess whether enabling structured reasoning traces influences performance and output behavior under both vLLM and Ollama. This setting was not extended to the other evaluated model families, as equivalent controllable thinking or reasoning modes are not available across Llama 3.1, Phi 3.5, and Granite 4.1 in the selected deployment configurations. As a result, Qwen serves as the only case in which the effect of reasoning-enabled generation can be isolated within this experimental setup.

Although several modern model families, such as OpenAI’s GPT-4-class models, Google’s Gemini series and DeepSeek-R1 variants, provide explicit or implicit reasoning capabilities, these were not included in the present evaluation. This is primarily due to deployment constraints because many of these models either exceed the memory and compute limitations of a single T4 GPU. Consequently, they were excluded from the current study in order to preserve consistency in hardware and evaluation methodology.

### 5.4.3. Tool integration strategies

Modern inference frameworks support structured tool calling, where models can interact with external functions through explicitly defined schemas. In such systems, tool usage is determined by how tool specifications are provided to the model, and how tool invocation is expressed in the model output. These two design choices jointly define the reliability of

tool-integrated inference pipelines.

In this work, we compare two tool provisioning strategies. In the first setting, consistent with all the other experiments previously done, tool descriptions are injected via the system prompt, relying on the model to interpret and follow informal instructions. In the second setting, tool specifications are passed through the native framework interface (i.e., the dedicated tool definition field exposed by the serving stack). These approaches differ in how explicitly tool structure is grounded in the model’s context, which can affect both tool adherence and selection behavior. The different system prompts corresponding to the two settings are provided in Appendix A and Appendix B, respectively.

Another difference lies in the output representation of tool intent: implicit tool representation, where tool intent is embedded within a general-purpose response field and must be parsed by the serving layer, and explicit tool representation, where tool invocations are emitted in a dedicated structured field reserved for tool calls.

Overall, these design dimensions introduce a trade-off between flexibility and robustness: prompt-based tool specification combined with content-based representation of tool intent offers greater flexibility, whereas native tool schemas combined with structured output fields improve interpretability, reliability, and seamless integration with the serving layer.

This subsection evaluates how different tool provisioning strategies affect LLM serving systems performance. In particular, this subsection addresses the following research question:

**RQ4.** How do tool provisioning strategies influence accuracy and system-level behavior in LLM serving systems?

Table 5.3: Performance comparison across different tool provisioning strategies

| Model        | Reasoning | Tool Provisioning | TIA   | GAS   | LAT   |
|--------------|-----------|-------------------|-------|-------|-------|
| Qwen3-8B-FP8 | Disabled  | System prompt     | 81.4% | 98.7% | 9.6s  |
| Qwen3-8B-FP8 | Mixed     | System prompt     | 91.5% | 98.5% | 25.3s |
| Qwen3-8B-FP8 | Disabled  | Tool schema       | 87.4% | 90.7% | 10.7s |
| Qwen3-8B-FP8 | Mixed     | Tool schema       | 88.8% | 91.8% | 27.5s |

During the experiments reported in Table 5.3, we observed that although explicit tool output fields are available when using a native tool schema configuration, the model occasionally emits tool calls in the default response field instead of the designated tool output field. To handle this, we apply a post-processing step that parses and recovers tool

invocations from both output fields. The results reported above were computed using this unified extraction procedure.

Taking this into account, we analyze how reasoning mode and tool provisioning strategy interact with system-level performance. The results highlight clear interactions between reasoning mode, tool provisioning strategy, and system-level performance, revealing trade-offs across tool identification accuracy (TIA), general answer stability (GAS), and latency (LAT).

First, in previous experiments, we observed that when tools are provided via the system prompt, enabling mixed reasoning consistently improves TIA and increases LAT (9.6s vs. 25.3s), while GAS remains largely stable (98.7% vs. 98.5%). This suggests that reasoning primarily improves tool relevance inference under prompt-based tool provisioning, at the cost of higher decoding overhead.

When tools are instead provided via a dedicated tool schema field, the effect of reasoning becomes less beneficial. Under disabled reasoning, schema-based tool provisioning already improves tool identification accuracy compared to prompt-based provisioning (81.4% vs. 87.4%), while maintaining similar latency (9.6s vs. 10.7s). This indicates that explicitly structured tool definitions reduce ambiguity in tool invocation even without additional reasoning capacity.

However, combining mixed reasoning with schema-based tool provisioning does not yield consistent additive gains. Compared to schema-based provisioning with disabled reasoning, mixed reasoning provides only a marginal improvement in TIA (87.4% vs. 88.8%) and GAS (90.7% vs. 91.8%) and substantially increasing LAT (10.7s vs. 27.5s). At the same time, compared to prompt-based provisioning with mixed reasoning, schema-based provisioning under mixed reasoning achieves lower TIA (91.5% vs. 88.8%), worse GAS (98.5% vs. 91.8%) and higher LAT (25.3s vs. 27.5s).

This indicates a strong interaction effect between reasoning mode and provisioning strategy, where schema-based tool definitions improve baseline robustness under low reasoning but become more sensitive under increased reasoning complexity.

Overall, the results suggest that prompt-based provisioning benefits more from reasoning augmentation, at the cost of increased latency, whereas schema-based provisioning improves baseline accuracy but is less robust under increased reasoning complexity. This degradation may be partly attributed to the additional context overhead introduced by explicit tool schemas, which increases input length and may interfere with reasoning-intensive generation.

#### 5.4.4. Tool orchestration frameworks

This subsection evaluates the impact of different tool orchestration frameworks on system performance. The objective is to analyze how the choice of execution layer influences response latency and execution reliability in language model systems with external tool integration. In particular, we compare two implementations: a custom Python-based framework and LangChain, a widely used open-source framework for developing applications powered by LLMs. LangChain has become a common abstraction layer in the LLM ecosystem, providing standardized components for chaining model calls, managing tool integrations, and constructing agent-like workflows.

These two approaches differ in how tool execution is structured and managed, ranging from a lightweight custom Python pipeline with explicit control over execution flow to LangChain, which provides a higher-level abstraction for composing LLM applications, including built-in support for chaining, tool routing, and agent-like execution patterns.

This subsection evaluates how different tool orchestration frameworks affects language model systems performance. In particular, this subsection addresses the following research question:

**RQ5.** How does the choice of tool orchestration framework influence response latency and execution reliability in language model systems with external tool integration?

Table 5.4: Performance across orchestration frameworks

| Model        | Reasoning | Framework   | Tool Provisioning | TIA   | GAS   | LAT   |
|--------------|-----------|-------------|-------------------|-------|-------|-------|
| Qwen3-8B-FP8 | Mixed     | Custom Made | System prompt     | 91.5% | 98.5% | 25.3s |
| Qwen3-8B-FP8 | Disabled  | Custom Made | Tool schema       | 87.4% | 90.7% | 10.7s |
| Qwen3-8B-FP8 | Disabled  | LangChain   | Tool schema       | 84.5% | 88.6% | 12.3s |

Overall, the results reported in Table 5.4 indicate that exhibits lower performance across all three metrics, which can be attributed to two main factors: limitations in reasoning configuration flexibility and differences in tool output parsing robustness.

First, the reasoning configuration in LangChain is constrained to a global binary setting, where reasoning must be enabled or disabled across calls. This prevents the use of mixed reasoning strategies that proved beneficial in earlier experiments under prompt-based provisioning. While enabling reasoning globally would likely improve tool identification accuracy, it also leads to extremely high latency (approximately 55s on average in preliminary tests), making it impractical under the evaluated setting. As a result, LangChain is evaluated in a disabled-reasoning configuration, which directly limits its tool identification performance compared to the custom framework with mixed reasoning (84.5% vs. 91.5%).

Second, we observe differences in tool output handling between the two frameworks. The custom agent pipeline includes a layer that checks both structured tool output fields and fallback JSON patterns in the general response field. This is necessary because, in practice, tool calls were not always emitted in the expected structured field. In contrast, the LangChain-based integration relies primarily on native structured tool outputs and does not consistently recover tool calls when they appear in unstructured response content. This leads to missing or partially parsed tool invocations, which negatively impacts both TIA and GAS.

Finally, LangChain introduces a slight increase in latency (12.3s vs. 10.7s) due to additional abstraction layers in the execution pipeline, though this effect is secondary compared to the constraints imposed by reasoning configuration and parsing behavior.

Overall, these results suggest that while higher-level orchestration frameworks such as LangChain provide usability and abstraction benefits, they may introduce constraints in reasoning control and output robustness that negatively affect performance in tool-intensive settings. In contrast, the custom agent framework allows finer-grained control over reasoning configuration and implements more resilient tool output parsing, resulting in better overall performance under the evaluated workload.

## 5.5. Analysis

The experimental results reveal several important patterns regarding the interplay between model capacity, reasoning strategies, serving infrastructure, and system architecture in tool-augmented language model systems.

### The role of reasoning in tool-calling tasks

The observation that mixed reasoning mode (applying reasoning only during tool invocation) yields superior performance compared to full reasoning mode is particularly useful. Tool invocation primarily demands structured planning, identifying the appropriate function, mapping user intent to available interfaces, and constructing valid arguments. These operations benefit from step-by-step reasoning. In contrast, response generation after tool execution is largely a formatting and synthesis task that does not require the same depth of deliberation. This finding has practical implications: systems can achieve better accuracy-efficiency trade-offs by decoupling reasoning allocation across different pipeline stages rather than applying it uniformly.

## Model size influence

The progressive improvement in TIA from 1.7B to 8B models further confirms that tool calling is a capability that scales with model size. Smaller models, despite achieving reasonable GAS scores, struggle with accurate tool selection. This indicates a capacity bottleneck in compact models, where parameter efficiency comes at the cost of tool-use proficiency.

## Quantization as a deployment strategy

The marginal performance difference between FP8 quantization (81.4% TIA) and AWQ quantization (80.9% TIA) suggests that modern quantization techniques have reached a level of maturity where they can be applied with minimal impact on downstream accuracy. This observation is particularly relevant under the VRAM constraints of the T4 GPU, where quantization becomes a key enabler for deploying models that would otherwise exceed memory limits. In this setting, FP8 demonstrates a slight but consistent performance advantage over AWQ in the 8B parameter model, while maintaining comparable latency characteristics. Overall, these results indicate that FP8 provides a favorable trade-off between efficiency and model quality for the considered deployment environment.

## Serving framework effects on output behavior

The divergence between vLLM and Ollama on GAS and TIA metrics cannot be attributed solely to inference efficiency. The qualitative observation that Ollama models exhibit verbose, conversational output styles that violate strict schema requirements reveals a fundamental difference in how these frameworks shape model behavior. This likely stems from differences in fine-tuning alignments applied during model packaging for Ollama, and serving stack configurations that influence decoding behavior. This finding underscores that serving framework selection is not merely an infrastructure decision but a behavioral influence that affects output reliability in production tool-calling pipelines.

## Tool integration

The finding that schema-based tool provisioning improves baseline TIA (87.4% vs. 81.4%) but becomes less synergistic with reasoning is interesting. The explicit structuring of tool schemas appears to reduce ambiguity. However, when reasoning is introduced, this advantage diminishes. The increased context overhead from detailed schema definitions may also interfere with extended reasoning traces, causing the model to pay more attention

to schema details at the expense of deeper planning.

This result implies that the optimal tool provisioning strategy is context-dependent: prompt-based provisioning with reasoning augmentation is preferable when accuracy is priority and latency is tolerable, while schema-based provisioning without reasoning is more suitable for latency-sensitive deployments where baseline accuracy is acceptable.

The custom pipeline’s superior performance relative to LangChain highlights the value of fine-grained control in production tool-calling systems. The implementation of fallback JSON parsing across multiple output fields demonstrates a useful approach to handling the imperfect adherence to schema expectations that characterizes current language models.

## 5.6. Discussion

The evaluation methodology demonstrates several notable strengths. The use of TIA, GAS, and LAT as complementary metrics provides a multidimensional performance assessment rather than reducing system quality to a single dimension. The primary emphasis on TIA as a functional prerequisite for system utility correctly captures the non-negotiable requirement that tools must be invoked correctly for the system to operate as designed.

In addition, testing different factors one by one (model size, quantization, reasoning mode, serving framework, tool provisioning strategy, and orchestration framework) makes it possible to understand the effect of each factor separately. This helps make better, evidence-based design choices.

## Limitations

Despite the comprehensive evaluation, several limitations constrain the generalizability of the findings. The reliance on a single GPU type (NVIDIA T4) with limited VRAM restricted the range of deployable models, preventing the evaluation of larger models that may exhibit qualitatively different behavior.

Although the fixed test set was designed to target different tools and scenarios, they may not fully capture the diversity of real-world user queries, potentially overestimating performance on in-distribution inputs while underestimating brittleness to out-of-distribution queries.

The LLM-as-a-Judge evaluation using Grok-4-1-fast-reasoning introduces a potential circularity, as the judge’s own reasoning capabilities and alignment may influence GAS scores in ways that do not perfectly correlate with human preferences. While a fixed prompt and generation settings mitigate variability, the absence of human evaluation as a ground-

truth reference point limits the ability to fully validate the judge’s reliability. Additionally, the evaluation did not cover robustness to intentionally misleading or malicious inputs, or complex tool chaining scenarios, which may represent important real-world use cases.

## Potential improvements

Several opportunities for improvement emerge from this analysis. Regarding model selection, when GPU resources permit, future deployments should consider larger and reasoning capable models, as the improvement in TIA with model size suggests continued gains at higher capacity.

Regarding tool integration, the finding that prompt-based provisioning benefits more from reasoning suggests a hybrid approach: using explicit tool schemas for baseline robustness while retaining prompt-based descriptions as supplementary context that reasoning can leverage. Dynamic switching between provisioning strategies based on query complexity or on user request could also be explored, though this would introduce additional system complexity.

Regarding the evaluation methodology, incorporating human evaluations alongside LLM-as-a-Judge assessments would improve the reliability of GAS measurements. Expanding the test set to include intentionally misleading inputs would provide a more complete view of system robustness.

Overall, the system demonstrates strong tool-calling capabilities when appropriately configured, with Qwen3-8B-FP8 under vLLM using mixed reasoning representing the best balance of accuracy, output quality, and latency for the evaluated setting. The findings regarding reasoning configuration, serving framework behavior, and orchestration trade-offs provide a reference line for deploying similar systems in resource-constrained environments.

This chapter presented a comprehensive evaluation across multiple dimensions of system performance, demonstrating that the proposed approach achieves strong tool-calling capabilities under the evaluated setting. The analysis revealed key trade-offs between reasoning strategies, serving frameworks, and orchestration approaches, providing concrete guidance for deploying similar systems in resource-constrained environments. Taken together, these findings provide a positive answer to RQ1: when appropriately configured, the proposed tool-augmented LLM architecture delivers a unified natural language interface over the consolidated heterogeneous academic data sources, supporting expert discovery and collaboration network exploration under the evaluated setting. The following chapter

concludes this thesis by summarizing the contributions, discussing broader implications, and outlining directions for future work.



## 6 | Conclusions

This thesis proposes *FindMyExpert*, an intelligent conversational system designed to assist faculty at Politecnico di Milano in exploring collaboration networks and identifying relevant experts within the institution. The system addresses the challenge of fragmented academic information by providing a unified natural language interface over multiple institutional data sources, including publication repositories, thesis archives, and records of international agreements.

We designed and implemented a data integration pipeline that collects, transforms, and normalizes information from heterogeneous institutional sources into a centralized MongoDB database, improving the accessibility and consistency of academic data. We further developed a *tool-based* retrieval framework that enables structured access to academic data, where each data source is accessed through dedicated tools that abstract underlying implementation details. On top of this, we built a system architecture where an LLM acts as an orchestrator, dynamically selecting tools based on user queries, and synthesizing the retrieved information into coherent natural language responses. Throughout the development process, we conducted a comprehensive evaluation of different system configurations, analyzing the impact of model size, quantization strategy, reasoning mode, serving framework, tool integration strategies, and orchestration approaches on system performance. This allowed us to identify the most relevant and effective solutions, while also providing alternative configurations that can be explored by other researchers.

### Key findings

The evaluation yielded five main insights. First, *tool calling* is a capability that scales with model size. TIA increases steadily from 46.8% for Qwen3-1.7B (reasoning disabled) to 79.7% for Qwen3-4B and up to 81.4% for Qwen3-8B-FP8 under the same configuration, with the best-performing setup, Qwen3-8B-FP8 in mixed reasoning mode, reaching 91.5% TIA and 98.5% GAS. Interestingly, applying reasoning only during tool selection (mixed reasoning), rather than throughout the entire response generation process, proved more efficient without sacrificing accuracy: for Qwen3-8B-FP8, mixed reasoning achieves 91.5%

TIA at 25.3s latency, whereas full reasoning yields 91.2% TIA at 34.1s, and disabled reasoning drops to 81.4% at 9.6s. This suggests that tool invocation benefits from deliberate planning, whereas response synthesis can be performed with less deliberation, reducing computational overhead and response time while maintaining comparable performance.

Second, *quantization* strategies have marginal impact on downstream accuracy. At the 8B scale, FP8 and AWQ quantization yield nearly identical TIA (81.4% and 80.9% respectively with reasoning disabled), while FP8 reaches a GAS of 98.7% compared to 94.1% for AWQ, with comparable latency (9.6s vs. 6.8s). Modern quantization techniques such as FP8 and AWQ can therefore be applied under GPU memory constraints without significant performance trade-offs, making them viable deployment strategies for resource-limited environments.

Third, the *serving framework* influences not only latency but also output behavior. vLLM produces outputs that are better aligned with structured schema requirements, making it more suitable for tool-calling pipelines. In contrast, Ollama models tend toward more verbose and conversational outputs that can violate strict formatting constraints, introducing reliability issues in production environments. For Qwen3-8B-FP8, vLLM achieves 98.5% GAS and 25.3s latency, while Ollama reaches 92.6% GAS at 30.4s. The gap is more pronounced for other model families: Llama 3.1 drops from 91.6% to 68.1% GAS and from 74.0% to 49.6% TIA when moving from vLLM to Ollama, and Phi-3.5-mini-instruct falls from 75.3% to 20.6% GAS.

Fourth, *tool provisioning* strategies interact with reasoning in non-linear ways. Under disabled reasoning, schema-based tool provisioning already improves TIA from 81.4% to 87.4% with comparable latency (9.6s vs. 10.7s), but reduces GAS from 98.7% to 90.7%. Prompt-based tool specification benefits more from reasoning: mixed reasoning raises TIA from 81.4% to 91.5% while keeping GAS stable at 98.5%, whereas schema-based provisioning under mixed reasoning reaches only 88.8% TIA and 91.8% GAS at higher latency (27.5s). This suggests that the optimal provisioning strategy depends on the target deployment configuration and the trade-off between flexibility and robustness.

Fifth, our custom orchestration framework outperforms established orchestration libraries such as LangChain in tool-intensive settings. Under comparable conditions (Qwen3-8B-FP8, reasoning disabled, tool schema), the custom pipeline achieves 87.4% TIA, 90.7% GAS, and 10.7s latency, while LangChain reaches 84.5% TIA, 88.6% GAS, and 12.3s latency. When mixed reasoning is enabled in the custom pipeline, the gap widens further to 91.5% TIA and 98.5% GAS. The custom implementation provides finer-grained control over reasoning configuration and more resilient tool output parsing, accommodating the

variability in model outputs more effectively than structured abstractions.

## Future work

The evaluation was constrained by limited GPU resources, which prevented the testing of larger models that may exhibit qualitatively different behavior. Multiple major LLM families were attempted, including Gemma and Mistral, but could not be successfully deployed or evaluated with the available computational resources. The observed scaling trends suggest that higher-capacity variants could improve accuracy, and access to more powerful hardware would enable exploration of larger model families as well as reasoning-capable models. These models often require significant computational resources and may involve different deployment strategies compared to the configurations tested in this work, but the potential gains in tool selection accuracy and response quality make them a promising direction for future investigation. To this end, we developed a reproducible evaluation pipeline that facilitates repeating the experiments across different hardware architectures, enabling fair comparisons with the configurations tested in this work.

An additional direction for future work concerns alternative data retrieval paradigms. While the current system relies on explicit *tool-based* access to structured data sources, another possible approach is to reduce reliance on predefined tools and expose the underlying database schema directly to the language model. In this setting, the model could generate customized queries which would then be executed in a controlled sandbox environment. This would allow for more flexible and expressive data access patterns, potentially improving coverage and reducing the need for manually defined tool interfaces, while still preserving execution safety through isolation and validation mechanisms.

Additionally, the LLM-as-a-Judge approach, though consistent and scalable, may not perfectly reflect human preferences in all cases. Since the system runs on Politecnico di Milano infrastructure, there is an opportunity to leverage user feedback collected during deployment in a fast and simple way for model fine-tuning. Over time, the interactions and corrections provided by faculty members could be used to create a specialized model variant better suited for the specific task of academic expert and collaboration network discovery. This continuous improvement loop could enhance both tool selection accuracy and response quality, adapting the system to the actual usage patterns observed in production.

As data coverage is concerned, the data integration pipeline is designed to accommodate additional sources with minimal changes. New institutional repositories can be ingested into the centralized database and corresponding tools can be created following the existing framework, extending the system coverage to include more data sources without modifying

the overall architecture. This extensibility ensures that the system can grow alongside the institution's data assets. In particular, incorporating information about joint research projects or co-organized events could further enhance the system ability by providing additional context and enabling richer connections between institutional data sources.

Overall, *FindMyExpert* demonstrates strong tool-calling capabilities when appropriately configured, offering a solid foundation for further development and deployment in real-world academic settings.

# Bibliography

- [1] W. Aboucaya and D. Jasim. A Linked Dataset of PhDs, Academic Networks, and Institutions, 2026. URL <https://arxiv.org/abs/2604.08619>.
- [2] Anthropic. Computer use with Claude. <https://platform.claude.com/docs/en/agents-and-tools/tool-use/computer-use-tool>, 2026.
- [3] P. Bakhtiarifard, P. Tözün, C. Igel, and R. Selvan. Position: Neglecting the Sustainability of AI is Fuelling a Global AI Arms Race, 2026. URL <https://arxiv.org/abs/2502.20016>.
- [4] J. Beel, B. Gipp, S. Langer, and C. Breiteringer. Research paper recommender systems: A literature survey. *International Journal on Digital Libraries*, 2015.
- [5] F. Bousetouane. Agentic Systems: A Guide to Transforming Industries with Vertical AI Agents, 2025. URL <https://arxiv.org/abs/2501.00881>.
- [6] J. Chen, Y. Wu, Q. Wang, Q. Chen, Y. Li, Y. Zhang, X. Zhang, W. Zhou, and C. Ma. SEAL: Can Saturated Benchmarks Be Revived by LLM-as-a-Meta-Judge?, 2026. URL <https://arxiv.org/abs/2605.30104>.
- [7] Y. Du, T. Jin, J. Kang, X. Pang, J. Chai, T. Miao, F. Liu, W. Wang, S. Yao, Y. Zhang, and S. Chen. Towards self-evolving agentic literature retrieval, 2026. URL <https://arxiv.org/abs/2605.14306>.
- [8] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, and H. Wang. Retrieval-augmented generation for large language models: A survey, 2024. URL <https://arxiv.org/abs/2312.10997>.
- [9] GitHub. GitHub Copilot: Your AI pair programmer. <https://github.com/features/copilot>, 2026.
- [10] S. Go, J. Park, S. More, H. Wu, I. Wang, A. Jezghani, T. Krishna, and D. Mahajan. Characterizing the efficiency of distributed training: A power, performance, and thermal perspective, 2025. URL <https://arxiv.org/abs/2509.10371>.

- [11] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 1997.
- [12] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. Bang, D. Chen, W. Dai, H. S. Chan, A. Madotto, and P. Fung. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 2022.
- [13] D. Jurafsky and J. H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, with Language Models*. 3rd edition, 2026. Online manuscript released January 6, 2026.
- [14] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W.-t. Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2020.
- [15] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33. Curran Associates, Inc., 2020.
- [16] K. Ling, Z. Qin, Y. Zhu, H. Zhang, H. Yu, and G. Fan. Seda: A unified system for dataset discovery and multi-entity augmented semantic exploration, 2026. URL <https://arxiv.org/abs/2603.07502>.
- [17] Z. Ma, J. Liu, X. Luo, Z. Huang, Q. Zhu, and W. Che. Advancing Tool-Augmented Large Language Models via Meta-Verification and Reflection Learning. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*. ACM, 2025. doi: 10.1145/3711896.3736835. URL <http://dx.doi.org/10.1145/3711896.3736835>.
- [18] C. Manning and H. Schütze. Foundations of statistical natural language processing. *MIT Press*, 1999.
- [19] J. F. Medrano. Calidad en repositorios digitales en argentina, estudio comparativo y cualitativo, 2017. URL <https://arxiv.org/abs/1705.08275>.
- [20] A. Miller, A. Fisch, J. Dodge, A.-H. Karimi, A. Bordes, and J. Weston. Key-Value Memory Networks for Directly Reading Documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2016.
- [21] F. Mo, K. Mao, Z. Zhao, H. Qian, H. Chen, Y. Cheng, X. Li, Y. Zhu, Z. Dou, and

- J.-Y. Nie. A survey of conversational search. *ACM Trans. Inf. Syst.*, 43(6), 2025. URL <https://doi.org/10.1145/3759453>.
- [22] M. E. J. Newman. Scientific collaboration networks. i. network construction and fundamental results. *Phys. Rev. E*, 64, 2001.
- [23] OpenAI. Function calling. <https://developers.openai.com/api/docs/guides/function-calling>, 2026.
- [24] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. F. Christiano, J. Leike, and R. Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35. Curran Associates, Inc., 2022.
- [25] S. Paul. Building faculty expertise ontology using protege: Enhancing academic library research services, 2026. URL <https://arxiv.org/abs/2601.07451>.
- [26] Politecnico di Milano. IRIS Re.Public@polimi.it. URL <https://re.public.polimi.it>.
- [27] Politecnico di Milano. POLITesi - Digital archive of PhD and post graduate theses. URL <https://www.politesi.polimi.it>.
- [28] S. Robertson and H. Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*, 3, 2009.
- [29] T. Schick, J. Dwivedi-Yu, R. Dessi, R. Raileanu, M. Lomeli, E. Hambro, L. Zettlemoyer, N. Cancedda, and T. Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36. Curran Associates, Inc., 2023.
- [30] Y. Shen, K. Song, X. Tan, D. Li, W. Lu, and Y. Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. In *Advances in Neural Information Processing Systems*, volume 36. Curran Associates, Inc., 2023.
- [31] V. Stoyanov, J. Mayfield, T. Xu, D. Oard, D. Lawrie, T. Oates, and T. Finin. A context-aware approach to entity linking. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction (AKBC-WEKEX)*. Association for Computational Linguistics, 2012.
- [32] M. Suzgun, E. Shen, F. Bianchi, A. Spangher, T. Icard, D. E. Ho, D. Jurafsky, and

- J. Zou. Evaluating Commercial AI Chatbots as News Intermediaries, 2026. URL <https://arxiv.org/abs/2605.22785>.
- [33] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
- [34] J. Vipra and A. Korinek. Market concentration implications of foundation models, 2023. URL <https://arxiv.org/abs/2311.01550>.
- [35] X. Wang, L. Fu, X. Gan, Y. Wen, G. Zheng, J. Ding, L. Xiang, N. Ye, M. Jin, S. Liang, B. Lu, H. Wang, Y. Xu, C. Deng, S. Zhang, H. Kang, X. Wang, Q. Li, Z. Guo, J. Qi, P. Liu, Y. Ren, L. Wu, J. Yang, J. Zhou, and C. Zhou. AceMap: Knowledge Discovery through Academic Graph, 2024. URL <https://arxiv.org/abs/2403.02576>.
- [36] J. Wei, X. Wang, D. Schuurmans, M. Bosma, b. ichter, F. Xia, E. Chi, Q. V. Le, and D. Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35. Curran Associates, Inc., 2022.
- [37] H. Yang, S. Yue, and Y. He. Auto-GPT for Online Decision Making: Benchmarks and Additional Opinions, 2023. URL <https://arxiv.org/abs/2306.02224>.
- [38] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao. ReAct: Synergizing Reasoning and Acting in Language Models, 2023. URL <https://arxiv.org/abs/2210.03629>.

# A | System Prompt: Tool descriptions in system prompt

You are an academic research assistant that helps students find POLIMI researchers for  
 ↪ international thesis collaborations.

PRIMARY TOOL (Use this for most researcher queries):

1. `find_polimi_researchers(countries=[], institutions=[], research_interest="",  
 ↪ from_date="", to_date="", limit=20)`
  - Use this tool to find internal POLIMI researchers that have collaborations  
 ↪ filtered by country, institution, or research interest.
  - Returns basic researcher information only (name, number of matching records,  
 ↪ profile link).

SECONDARY TOOLS (Use these for specific follow-ups):

2. `get_publications(author_name="", from_date="", to_date="", limit=20)`
  - Use this tool to retrieve publications of a single researcher.
  - Primarily for inferring research topics of that author.
  - Returns publication information only.
3. `get_theses(creator="", contributor="", research_interest="", from_date="",  
 ↪ to_date="", limit=20)`
  - Use this tool to retrieve theses records by creator, contributor, or topic.
  - At least one of creator, contributor, or research\_interest must be provided.
  - Returns thesis metadata fields such as title, creator, contributor, date, and  
 ↪ description.
4. `get_collaboration_network(author_name="", fields = [], from_date="", to_date="",  
 ↪ limit=20)`
  - Use this tool to retrieve collaboration information for a specific single  
 ↪ researcher.
  - DO NOT call this tool if the query is about general collaborations with a country,  
 ↪ institution, or topic.

- The optional ``fields`` parameter allows you to receive only a subset of data.
  - ↪ Allowed fields: "publication\_count", "countries", "institutions",
  - ↪ "collaborators"
- Returns collaboration-focused data only, such as publication count, co-authors,
  - ↪ and related affiliations or countries, ordered by relevance.

#### 5. `get_authors_affiliations(author_names=[])`

- Use this tool to retrieve author affiliation for one or more authors.
- Returns basic affiliation information: ``name`` and ``institution``.
- If no affiliation is found, returns that author name with ``institution = None``.

#### 6. `works_with(author_list=[], author_country="", limit=50)`

- Use this tool when the user asks who one or more specific researchers collaborate
  - ↪ with in a determinate country.
- ``author_list`` must contain one or more explicit researcher names.
- Use ``author_country`` only when the user asks to filter collaborators by country.
- Returns each requested author with a list of collaborators.

#### 7. `find_polimi_agreements(cities=[], countries=[], continents=[], universities=[], ↪ ccs=[], promoter="", program_name="", limit=20)`

- Use this tool when the user asks about POLIMI international agreements, partner
  - ↪ universities, related programs, promoters, countries, or continents.
- This is the only tool that queries the agreements dataset.
- Use ``cities``, ``countries``, ``continents``, ``universities``, or ``ccs`` when those
  - ↪ filters are explicitly provided.
- For ``cities``, ``countries`` and ``continents``, input the names translated to Italian
  - ↪ (e.g. Zurich -> Zurigo, France -> France, South America -> America del Sud,
  - ↪ etc.).
- Use ``promoter`` or ``program_name`` only when the user explicitly mentions them.
- Returns agreements grouped by university, including promoter and program
  - ↪ information.

#### CRITICAL RULES:

- You may answer directly in natural language if the user query can be answered using
  - ↪ general knowledge or previous tool outputs.
- If you decide that a tool must be called, respond with JSON only. Do NOT include
  - ↪ explanations, comments, or natural language.
- Only call ONE tool per query.
- Do NOT invent argument values.
- Include an argument ONLY if explicitly provided by the user.
- Never call the same tool twice with the same arguments in a single conversation.
- When calling a tool, follow this exact JSON structure:

```
{
  "tool": "tool_name",
  "arguments": {
```

```
"param1": value1,  
"param2": value2  
}  
}
```

CONVERSATION MEMORY:

- Inspect previous assistant messages and previous tool outputs before deciding which  
↪ tool to call
- If a previous tool output already contains the information needed, DO NOT call a new  
↪ tool
- Treat previous tool outputs as part of the working context
- The user does not see tool outputs, only your natural language responses
- Do not refer explicitly to tools or tool calls in your natural language responses
- If no available tool applies, do NOT call a tool and answer directly in natural  
↪ language. Use ONLY general knowledge.



# B | System Prompt: Native tool interface

You are an academic research assistant that helps students find POLIMI researchers for  
 ↪ international thesis collaborations.

When you need external data, use the provided tools via the model's function-calling  
 ↪ mechanism. Do NOT emit raw JSON or inline tool-call text in your assistant  
 ↪ message.

PRIMARY TOOL (Use this for most researcher queries):

1. `find_polimi_researchers(countries=[], institutions=[], research_interest="",  
 ↪ from_date="", to_date="", limit=20)`
  - Use this tool to find internal POLIMI researchers that have collaborations  
 ↪ filtered by country, institution, or research interest.
  - Returns basic researcher information only (name, number of matching records,  
 ↪ profile link).

SECONDARY TOOLS (Use these for specific follow-ups):

2. `get_publications(author_name="", from_date="", to_date="", limit=20)`
  - Returns publication information only.
3. `get_theses(creator="", contributor="", research_interest="", from_date="",  
 ↪ to_date="", limit=20)`
  - At least one of creator, contributor, or research\_interest must be provided.
  - Returns thesis metadata fields such as title, creator, contributor, date, and  
 ↪ description.
4. `get_collaboration_network(author_name="", fields = [], from_date="", to_date="",  
 ↪ limit=20)`
  - DO NOT call this tool if the query is about general collaborations with a country,  
 ↪ institution, or topic.
  - Returns collaboration-focused data only, such as publication count, co-authors,  
 ↪ and related affiliations or countries, ordered by relevance.

5. `get_authors_affiliations(author_names=[])`
  - Returns basic affiliation information: ``name`` and ``institution``.
6. `works_with(author_list=[], author_country="", limit=50)`
  - Returns each requested author with a list of collaborators.
7. `find_polimi_agreements(cities=[], countries=[], continents=[], universities=[],  
↪ ccs=[], promoter="", program_name="", limit=20)`
  - Use ``cities``, ``countries``, ``continents``, ``universities``, or ``ccs`` when those  
↪ filters are explicitly provided.
  - For ``cities``, ``countries`` and ``continents``, input the names translated to Italian  
↪ (e.g. Zurich -> Zurigo, France -> France, South America -> America del Sud,  
↪ etc.).
  - Use ``promoter`` or ``program_name`` only when the user explicitly mentions them.
  - Returns agreements grouped by university, including promoter and program  
↪ information.

#### CRITICAL RULES:

- Only call ONE tool per query.
- Do NOT invent argument values.
- Include an argument ONLY if explicitly provided by the user.
- Never call the same tool twice with the same arguments in a single conversation.

#### CONVERSATION MEMORY:

- Inspect previous assistant messages and previous tool outputs before deciding which  
↪ tool to call
- If a previous tool output already contains the information needed, DO NOT call a new  
↪ tool
- Treat previous tool outputs as part of the working context
- The user does not see tool outputs, only your natural language responses
- Do not refer explicitly to tools or tool calls in your natural language responses
- If no available tool applies, do NOT call a tool and answer directly in natural  
↪ language. Use ONLY general knowledge.

# C | FindMyExpert Source Code Repository

The software developed for this thesis, *FindMyExpert*, is available as a private GitHub repository at <https://github.com/YoSoyElLobo/FindMyExpert>. Table C.1 lists the main modules of the repository.

Table C.1: Main components of the FindMyExpert repository.

| Component            | Description                                                        |
|----------------------|--------------------------------------------------------------------|
| Flask API backend    | REST backend that orchestrates the LLM and the tool-calling loop.  |
| Next.js web frontend | React chat interface with conversation history and authentication. |
| Gradio front-end     | Lightweight temporary UI used for early prototyping.               |



## List of Figures

|     |                                                                        |    |
|-----|------------------------------------------------------------------------|----|
| 2.1 | Overview of the RAG process . . . . .                                  | 11 |
| 2.2 | Tool-Augmented Systems . . . . .                                       | 12 |
| 2.3 | Core components and interaction loop of an LLM-based agent . . . . .   | 16 |
| 3.1 | High-level architecture of the proposed system. . . . .                | 26 |
| 4.1 | High-level architecture of the data integration pipeline . . . . .     | 30 |
| 4.2 | UML component diagram of the implemented system architecture . . . . . | 35 |
| 4.3 | Empty state of the conversational interface . . . . .                  | 38 |
| 4.4 | An active conversation, showing the progress indicator . . . . .       | 39 |
| 4.5 | Feedback associated with an assistant response . . . . .               | 40 |



# List of Tables

|     |                                                                                |    |
|-----|--------------------------------------------------------------------------------|----|
| 2.1 | Summary of background concepts . . . . .                                       | 18 |
| 5.1 | Performance across model sizes and quantization levels. . . . .                | 47 |
| 5.2 | Performance across model families and serving frameworks . . . . .             | 49 |
| 5.3 | Performance comparison across different tool provisioning strategies . . . . . | 52 |
| 5.4 | Performance across orchestration frameworks . . . . .                          | 54 |
| C.1 | Main components of the FindMyExpert repository. . . . .                        | 75 |



## Acknowledgements

Quiero comenzar agradeciendo profundamente a mis padres, Elvia y César, porque gracias a su esfuerzo y sacrificio tuve la oportunidad de realizar esta maestría y porque me enseñaron una forma de ver la vida que me ha acompañado en cada paso de esta aventura. A mi mami Nérida, por acompañarme siempre. A Dios, por la gente buena puesta en mi camino. A mi tía Esther, por abrirme las puertas de su casa y hacerme sentir siempre bienvenido. A mi tía Gloria, por estar siempre dispuesta a escucharme y acompañarme con cariño. A mi familia en Perú, gracias por recibirme siempre con el mismo cariño y por recordarme que aquello que une la sangre, la distancia no lo puede separar. A mi familia en Italia, gracias por brindarme su apoyo, sin el cual este camino no habría sido posible y por hacer que, en una ciudad fría, encontrara el calor familiar. A mi familia en Alemania, gracias por convertirse en ese pequeño refugio que me acercaba a Perú. A mis amigos en Perú, gracias por estar presentes, cerca o lejos, y hacer que esta travesía sea más llevadera. A mis amigos en Italia, en particular a Abril, Luigi, Giulia, Alberto y Tatan, gracias porque hicieron del día a día un verdadero disfrute y porque celebro esta meta con las personas con las que me divertí en el recorrido.

I would also like to sincerely thank my advisors, Professor Mark Carman and Professor Cristiana Bolchini, for their invaluable support throughout the development of this thesis. I am especially grateful to Professor Mark Carman for his precise and insightful feedback, which were fundamental in shaping the direction and quality of this work. I would also like to thank Professor Cristiana Bolchini for continuously following my progress, ensuring that I stayed on track, and provided support that was essential for moving the thesis forward in a consistent way. This thesis project would not have been as smooth without their invaluable support.

Esta sección se volvería demasiado larga si mencionara a todas las personas que han tenido una influencia en mi formación y que, de alguna manera, han dejado parte de sí en este camino. Solo me queda decir gracias, porque mi único mérito es aprender y recoger lo mejor de las personas que me rodean. Lo que soy no es únicamente mío, sino también un reflejo de todos aquellos que me han acompañado.

