

**University of Salerno**  
**Department of Information and Electrical  
Engineering and Applied Mathematics**

Master's Degree in Computer Engineering  
Enterprise Software Architectures Curriculum



**GRADUATION THESIS**

**KNOWLEDGE GRAPHS FOR TIME ASSISTIVE  
MANAGEMENT IN COGNITIVE IMPAIRMENT**

**Supervisor**

**Prof. Sabrina Senatore**

**Candidate**

**Alexandra Tarantino**

**0622701904**

Academic Year 2023-2024

---

# **Abstract**

## **DESCRIPTION OF THE PROBLEM**

Mild Cognitive Impairment (MCI) represents an intermediate stage between normal cognitive aging and dementia, characterized by a decline in memory and executive functions that, although they do not completely impair daily autonomy, significantly affect quality of life. Among the most critical difficulties are deficits in time management, manifested through an altered perception of the duration of activities, difficulties in planning and organizing daily routines, and erroneous monitoring of the passage of time, resulting in forgetfulness of commitments, failure to follow therapies, and a persistent state of disorientation.

Assistive technologies (ATs), particularly Time Assistive Technologies (TATs) as electronic calendars, voice reminders, and virtual assistants, although useful for remembering commitments and deadlines, often have unintuitive interfaces and limited customization, forcing the caregiver to frequent manual interventions to update content and adapt it to the user's needs.

There emerges a need for technological solutions that favor natural interaction, accessible even to those with MCI: simple conversational systems that facilitate the entry, consultation, and modification of commitments. Such solutions should make time management more intuitive and autonomous, reducing caregiver burden and improving the daily well-being of people with MCI.

## **FRAMING OF THE THESIS IN THE CONTEMPORARY TECHNICAL SCENARIO**

Large Language Models (LLMs) have revolutionized natural language processing in recent years, but they suffer from "hallucinations" and lack of transparency about sources, compromising the reliability of answers. The Retrieval-Augmented

---

Generation (RAG) paradigm has overcome these limitations by integrating a mechanism for retrieving information from unstructured external documents by semantic search, however, it ignores logical relationships among information, providing fragmented and unrelated contexts.

GraphRAG, an evolution of RAG proposed in 2024, combines vector search with traversal of a Knowledge Graph, overcoming the constraints of textual documents through a semantic map in which nodes represent entities and relationships capture conceptual dependencies. This approach improves both accuracy and completeness while enabling source traceability and explainability of the generation process.

The present work is among solutions that leverage LLMs for natural language processing, but aims to overcome their limitations in terms of accuracy and transparency. Central to the proposal is the adoption of the GraphRAG paradigm, based on the use of a Knowledge Graph built with the support of a dedicated ontology. The user with MCI is offered a conversational Time Assistive Technology, accessible via natural language, which allows personal information to be stored and guarantees complete, reliable and personalized answers based on the individual profile.

## **PERSONAL CONTRIBUTION OF THE CANDIDATE**

The main contribution of this thesis is the design and implementation of an original GraphRAG-based solution for temporal assistance of patients with Mild Cognitive Impairment, a recent approach not yet adopted in the literature.

The developed system consists of two components: the Knowledge Graph Construction Component, which transforms natural language input into a personal knowledge graph, and the GraphRAG Component, which implements the GraphRAG approach to retrieve information and answer user questions.

A distinctive element is the integration of semantic support in the populating phase: a domain ontology (TAMOntology) has been defined and implemented to guide the extraction, classification, and insertion of data into the graph, ensuring semantic consistency and facilitating inference. Through the use of the ontology,

---

the information retrieval phase is also optimized, improving the accuracy, completeness, and relevance of the answers provided to the user.

This approach proves to be more reliable than LLM-only based solutions, which are prone to "hallucinations," and is more efficient and comprehensive than traditional RAGs that do not exploit relational links between data, providing people with MCI with natural language interaction, correct, personalized answers, and with source transparency.

## **DESCRIPTION OF THE APPLICATION/EXPERIMENTAL CON- TENTS OF THE THESIS WORK**

The developed system consists of a set of integrated components built in the Python environment and based on the use of the Neo4j database for knowledge graph management. A complete pipeline for graph construction and querying has been implemented. The process is based on a specially defined domain ontology, the TAMOntology, which provides the semantic schema underlying the knowledge representation. A prototype web user interface, built with Streamlit, has also been constructed, which allows interactive visualization of the implemented features.

To evaluate the benefits of using a knowledge graph in a RAG approach, an experimental protocol was implemented that, through the RAGAS framework, computes specific metrics for RAG-type systems: Context Precision, Context Recall, Answer Relevancy, Faithfulness, and Semantic Similarity.

The experimentation also involved the creation of ten synthetic profiles, each consisting of thirty interactions, employed to generate the knowledge graphs in Neo4j. Ten test datasets, each consisting of ten questions, were constructed with which the systems could be compared and metrics calculated.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Problem definition . . . . .	1
1.2	Relevance of the problem in the context of computer engineering . . . . .	2
<b>2</b>	<b>State of the art</b>	<b>4</b>
2.1	Detailed analysis of the state of the art . . . . .	5
2.1.1	Assistive Technologies (AT) . . . . .	5
2.1.2	Time Assistive Technologies (TAT) . . . . .	8
2.2	Identification of possible advancements from the state of the art . . . . .	9
2.2.1	Natural interaction and personalization in TATs . . . . .	10
2.3	Ontologies . . . . .	11
2.3.1	Ontology components . . . . .	13
2.3.2	Types and Classification of Ontologies . . . . .	14
2.3.3	Languages and standards for ontology modeling . . . . .	16
2.3.4	Semantic Inference . . . . .	19
2.4	Knowledge Graphs . . . . .	20
2.4.1	Historical Evolution . . . . .	22
2.4.2	Graph data models for knowledge representation . . . . .	23
2.4.3	Internal structure of a Knowledge Graph . . . . .	26
2.4.4	Knowledge Graph construction . . . . .	29
2.4.5	Knowledge Graph querying . . . . .	34
2.4.6	Application Scenarios . . . . .	38
2.5	GraphRAG . . . . .	45
2.5.1	Comparison with related techniques . . . . .	46
2.5.2	GraphRAG Architecture . . . . .	48
2.5.3	Application domains and case studies . . . . .	51
<b>3</b>	<b>Original contribution to the solution of the problem</b>	<b>53</b>
3.1	Definition of the proposed methodology . . . . .	54

3.2	Design of the proposed system . . . . .	57
3.2.1	User Interface . . . . .	59
3.2.2	TAM Ontology . . . . .	61
3.2.3	Ontology Parser . . . . .	67
3.2.4	KG Construction . . . . .	68
3.2.5	GraphRAG . . . . .	74
3.2.6	Neo4j Graph . . . . .	78
3.2.7	Embedding and GenerativeAI Models . . . . .	80
3.3	Technological and/or application innovations . . . . .	81
3.4	Tools, technologies and models used in the implementation . . . . .	83
3.4.1	Protégé . . . . .	83
3.4.2	Neo4j . . . . .	83
3.4.3	Python Framework and Package . . . . .	84
3.4.4	OpenAI Models . . . . .	85
<b>4</b>	<b>Experimental validation and application aspects</b>	<b>86</b>
4.1	Description of evaluation metrics . . . . .	86
4.1.1	Context Precision . . . . .	88
4.1.2	Context Recall . . . . .	90
4.1.3	Faithfulness . . . . .	92
4.1.4	Answer Relevancy . . . . .	93
4.1.5	Answer Similarity . . . . .	96
4.2	Definition of the Experimental Protocol . . . . .	97
4.3	Description of the data used for experimentation . . . . .	99
4.3.1	User Profiles Generated . . . . .	100
4.3.2	Knowledge Graph construction from User Interactions . .	104
4.3.3	Test Dataset . . . . .	107
4.4	Presentation and Analysis of results . . . . .	108
4.4.1	Average comparison of metrics . . . . .	108
4.4.2	Analysis on individual profiles . . . . .	110
4.4.3	Qualitative analysis on question types . . . . .	112
4.5	Assessment of the significance of the results obtained . . . . .	118
<b>5</b>	<b>Conclusions</b>	<b>120</b>
5.1	Final considerations . . . . .	120
5.2	Future improvements . . . . .	122

# Chapter 1

## Introduction

### 1.1 Problem definition

*Mild Cognitive Impairment* (MCI) is a clinical condition defined as an intermediate stage between normal cognitive aging and dementia. According to the National Institute of Neurological Disorders and Stroke (NINDS) [1], it manifests as a decline in cognitive function - particularly memory, attention, and planning - that, while measurable, does not markedly impair a person's functional autonomy. However, this condition represents a stage of cognitive vulnerability, which can evolve into more severe neurodegenerative diseases, such as Alzheimer's disease.

One of the critical issues concerns time management and daily activities. People with MCI may have difficulty remembering commitments, maintaining the correct sequence of actions, meeting deadlines, and orienting themselves over time. These difficulties lead to disorganization of routines, with significant effects on personal autonomy, quality of life, and psychological well-being, as well as increased caregiving burden for family members and caregivers [2].

In recent years there has been an increasing diffusion of technological tools for cognitive support [3]. So-called *Time Assistive Technologies* (TATs) include mobile device applications, voice reminders, and digital assistants designed to facilitate time and task management. However, such solutions still have substantial limitations: interaction is often unnatural, adaptation to user preferences is reduced, and information is treated as mere data, without semantic structuring to enable understanding or reasoning.

Many systems adopt a logic in which content is stored and displayed on demand, but not related to each other or analyzed in a contextual way. This approach is insufficient to meet the needs of users with MCI, who require not only a digital agenda, but an intelligent system that can support memory, simplify understanding of their agenda, and return reliable relevant information.

In this context, this thesis addresses the problem of designing a system that enables the user to communicate experiences, activities, appointments, and goals in natural language, and that is able to return answers that are consistent, verifiable, and tailored to their profile. The proposed approach combines the use of knowledge graphs, dynamically constructed from natural language interactions, with an ontological domain schema that guides their semantic structuring and enables their querying. The goal is to overcome the limitations of traditional TAT, realizing a system that is more accessible, transparent, and tailored to individual needs.

## 1.2 Relevance of the problem in the context of computer engineering

The design of digital systems for cognitive support of people with MCI poses complex challenges, involving several areas of computer engineering: from knowledge representation to human-computer interaction to semantic and dynamic data management.

In particular, *Time Assistive Technologies* (TATs) constitute an application field that requires solutions that can facilitate the temporal organization of daily life, the planning of activities, and the retrieval of contextual information. Although there are widespread tools—such as digital calendars, reminder systems or mobile applications—they are often inflexible, difficult to customize and poorly adaptable to the evolution of the user profile. Interaction, predominantly based on menus and icons, may be unsuitable for people with reduced cognitive abilities, and does not allow for continuous, natural or contextualized communication.

The challenge is to design a system that can offer concrete, personalized sup-

## *1. Introduction*

---

port. Specifically, the system must be capable of:

- **Structure knowledge in a formal and coherent way**, transforming textual inputs into semantic representations, so as to ensure consistency, interrogability and possibility of inference.
- **Interact in natural language**, breaking down technological barriers and adapting to the user's communicative abilities.
- **Manage a personal and dynamic knowledge base** that is progressively enriched through interactions, shaping the user's habits, activities and goals over time.
- **Retrieve and connect semantically relevant information**, building structured information contexts even from distributed and implicitly connected data.
- **Provide traceable and traceable answers to data**, ensuring that any information returned is supported by verifiable content present in the knowledge base.

The adoption of a knowledge graph, dynamically generated from textual input and queryable in natural language, introduces a mode of interaction that enhances the semantic coherence of information and ensures traceability of the answers provided by the system. From an engineering perspective, this implies the design of pipelines dedicated to semantic processing, the management of entities and relationships according to a consistent ontological schema, and the integration of components for information extraction and response generation.

This architecture, although focused on the assistive domain, has features that are also applicable to other contexts in which structured knowledge representation, natural language interaction, and response tracking are essential requirements. These include decision support in clinical settings, personalized training, or knowledge management in complex, distributed environments such as enterprise environments.

# Chapter 2

## State of the art

This chapter provides a detailed analysis of the technologies and models currently available to support time management in individuals affected by *Mild Cognitive Impairment* (MCI) or other forms of cognitive decline.

The chapter presents a review of *Assistive Technologies for Cognition* (ATC), with a specific focus on *Time Assistive Technologies* (TAT). Then explores the most recent approaches aimed at improving natural and personalized interaction in assistive systems, highlighting the role of *Knowledge Graphs* (KGs) as semantic infrastructures for representing structured knowledge. In this context, the emerging paradigm of *Graph-based Retrieval-Augmented Generation* (GraphRAG) is introduced, which leverages the semantic structure of KGs to guide information retrieval and generation in a more accurate and traceable manner.

An additional section is dedicated to *ontologies*, which provide the essential semantic support to ensure consistency, interoperability, and inferential capabilities within KGs.

The objective is to outline the theoretical and technological framework that underpins the design choices made in this thesis work and to highlight its originality in comparison to the current state of the art.

## 2.1 Detailed analysis of the state of the art

### 2.1.1 Assistive Technologies (AT)

Assistive technologies (ATs), particularly Assistive Technology for Cognition (ATC), are tools that aid cognitive function while performing tasks. They compensate for cognitive deficits by enabling or enhancing daily activities for people with dementia, Alzheimer's, or brain injury. These technologies include both low-tech (calendars, diaries) and high-tech solutions (smartphone apps, electronic reminders). They are designed to support different cognitive domains such as memory, attention, planning, and time management. ATs include various devices, software and intelligent systems that promote autonomy, quality of life and active participation of people with cognitive difficulties [4].

#### Types of Technologies

ATs include numerous devices, applications, and intelligent systems created to support the autonomy, quality of life, and participation of people with cognitive difficulties. According to the research of Yousaf et al. [5], who reviewed more than 600 mobile applications for assisting people with dementia, these technologies fall into six main categories:

##### 1. Activities of daily living (ADL) based cognitive training.

This category includes technologies that help people with dementia perform daily activities independently. It is divided into six areas:

- *Memory aids* help people manage daily activities by remembering appointments, medications or tasks. Examples include smartwatches with notifications, voice assistants such as Amazon Alexa for personalized reminders, and push notifications on smartphones.
- *Language skills*: some technologies help maintain or improve communication through apps with language games, word-image association exercises, and pronunciation programs.
- *Problem-solving technologies* offer exercises to stimulate logical reasoning, such as visual puzzles, logic sequences, or simple decision-making scenarios.

## 2. State of the art

---

- *Attention improvement* apps reinforce the ability to focus on specific tasks, with interactive exercises to identify details, remember sequences, or perform quick tasks.
- *Daily scheduling and organization* tools allow users to plan activities, such as digital calendars or specialized apps. Wearable devices with reminders also help with personal organization.
- *Communication tools* such as chatbots and virtual assistants facilitate social interaction with caregivers and family members, offering intuitive interfaces to encourage communication.

### 2. Monitoring

These tools monitor the health and safety of people with dementia in real time. For example, *Safety You* [6] shows real-time footage to detect falls, *PainChek™* [7] monitors pain with a digital scale, and devices such as *GENEActiv* [8] track physical activity and behavior.

### 3. Dementia screening

These are apps designed for early identification and assessment of a person's cognitive status. These apps offer quick and reliable tests to detect early signs of cognitive impairment. For example, the *MOBI-COG* [9] app runs clock drawing and mnemonic tests, while the *CAIDE Risk Score* [10] estimates long-term dementia risk.

### 4. Reminiscence and socialization

These applications facilitate reminiscence therapy and social interaction through shared memories and personal multimedia materials. For instance, *InspireD* [11] allows storage of photos, videos, and audio to recall memories and foster emotional connection with caregivers.

### 5. Tracking

These tools use GPS tracking to monitor and prevent wandering. Applications such as *Prototype Locating System* [12] provide real-time location data to caregivers. Smartwatches and GPS-enabled bracelets also belong to this category.

### 6. Caregiver support

These apps provide information, tips, and tools for caregivers. For example,

## 2. State of the art

---

*Dementia Caregiver Solutions* offers guides for daily management, behavioral strategies, and self-care tips, reducing caregiver burden.

### Benefits of ATC

The use of ATs for people with MCI offers significant benefits in many ways, as evidenced in several recent studies. ATs are proving to be promising tools for combating cognitive decline, improving quality of life, and promoting users' autonomy.

One of the main benefits of ATs is their positive impact on global cognitive function, as demonstrated by the meta-analysis conducted by Park and Ha [13], which found statistically significant effects on attention, memory, and visuospatial perception in subjects with MCI and dementia. In particular, digital tools were found to be effective in providing regular and adaptive cognitive stimulation, reducing the speed of cognitive impairment.

An additional relevant benefit is the enhancement of social participation. Technologies, especially when combined with behavioral change techniques (BCTs), are able to boost social interactions and reduce isolation, which is crucial for emotional and cognitive well-being. The systematic review protocol by Zhu, Al Mahmud, and Liu [3] aims precisely to identify the most effective strategies in this area, emphasizing the importance of ATs as tools to promote active involvement in social life.

ATs also improve self-management and perception of time, facilitating the organization of the day and adherence to routines, aspects that are often compromised in individuals with MCI. Studies such as those by Nishiura et al. [14] and Dsouza et al. [15] highlight how time orientation devices—such as talking digital calendars or visual reminders—are useful in supporting daily time management and reducing dependence on caregivers.

Finally, the adoption of AT can ease caregivers' caregiving burden by improving communication and reducing stress, as also found in the qualitative study by Baudin et al. [16], which explored the direct experiences of family members involved in the use of digital time-management devices in individuals with MCI.

## Challenges and limitations

Although ATs offer real possibilities for improving the lives of people with MCI, there are still several challenges that limit their widespread adoption and long-term effectiveness. These difficulties are design, technical, and cultural in nature.

One of the biggest obstacles is the great diversity in users' needs. People with MCI have very different profiles: residual abilities, personal preferences, social environment, and family support vary significantly from person to person. Unfortunately, many of the technologies available today follow a "one size fits all" approach that is inflexible and difficult to customize [17].

When technologies do not meet people's real needs, it is natural that few will choose to adopt them. To solve this problem, experts suggest developing technologies together with end users, actively involving both people with MCI and their caregivers at all stages of development. Working methods are needed to continuously gather feedback and transform it into more accessible, intuitive and truly useful features.

Another major critical issue is the lack of systematic usability testing. Assessing how people interact with technology is essential to find out what blocks them, what frustrates them, and where improvements can be made.

### 2.1.2 Time Assistive Technologies (TAT)

Among assistive technologies for daily activities, Time Assistive Technologies (TAT) play a key role. These tools improve autonomous time management in people with MCI or dementia. TATs respond to an early and widespread cognitive difficulty: the loss of the ability to navigate time, organize tasks, and meet deadlines.

These technologies include electronic calendars, visual and voice reminders, simplified graphical interfaces, fixed or mobile devices, and intelligent virtual assistants. All are designed to facilitate daily organization and provide greater confidence in time management. Their purpose is not to replace human assistance, but to increase patient autonomy and reduce caregiver burden.

Among the most established solutions is the MEMOplanner (MMP), one of the most popular TAT tools in Sweden in recent years. This physical device with a touch screen clearly displays the daily and weekly schedule, using symbols, colors, and voice prompts to support time orientation and task performance. According to Baudin et al. [16], regular use of the MMP promotes greater regularity in routine, improves understanding of time, and reduces disorientation, while still requiring caregiver intervention for personalization and maintenance.

Another significant example is AiZi Assist, an Android app that offers comprehensive support for people with Alzheimer's disease. In addition to traditional voice reminders, AiZi includes advanced features such as facial recognition, GPS tracking, cognitive games, and a simplified and adaptable user interface. As highlighted by Joshi et al. [18], AiZi represents an effective person-centered approach in which technology is integrated into daily life without replacing the caregiver's role, but supporting it.

## **2.2 Identification of possible advancements from the state of the art**

In recent years, digital solutions for people with dementia or MCI have evolved toward more natural, empathetic, and personalized interactions. It is no longer just about providing functional tools, but about creating intuitive and reassuring experiences that are close to the human way of communicating. This approach addresses the need to develop interfaces that are accessible to those with cognitive decline, fostering familiarity, emotional engagement, and relational continuity.

To achieve greater adaptability and personalization, many solutions integrate conversational modules based on large language models (LLMs), knowledge management systems, and multimodal interfaces. The goal is to ensure understanding of natural language and create consistent, personalized and reassuring interactions that can adapt to each user's needs and characteristics.

A significant example is MemoryCompanion [19], a system that offers cognitive and emotional support to people with Alzheimer's disease. The platform combines a GPT family language model with technological components that enhance

natural interaction and user engagement. MemoryCompanion is based on three core modules:

- An **automatic voice transcription module**, which enables the user to communicate orally with the system, facilitating interaction even in the presence of motor or literacy difficulties;
- An **audiovisual module**, based on voice cloning technologies and realistic facial animation (talking-face), which generates a virtual face and reassuring voice, often modeled on familiar subjects, with the aim of strengthening the emotional connection;
- A **dynamic profiling system**, which continuously collects and updates relevant personal information (preferences, habits, family relationships, emotions), building a structured semantic profile that guides the personalization of future conversations.

Using these elements, the system progressively adapts interactions, creating a consistent relationship centered on the user's identity. Evaluations show that MemoryCompanion outperforms models such as ChatGPT-4 in terms of perceived empathy, contextual relevance, and dialogue continuity, highlighting the importance of the relational aspect in dementia support technologies.

### 2.2.1 Natural interaction and personalization in TATs

Interest in natural and personalized interactions has led to significant improvements in applications for emotional well-being and socialization. However, this approach is still little explored in Time Assistive Technologies (TATs), where the management of daily activities is mainly through electronic calendars or button applications, which require manual actions to manage commitments.

This type of interaction, although functional, has several limitations for people with MCI or mild dementia: it is inflexible, context-insensitive, and often difficult to use without help. The need for more accessible tools paves the way for new natural language-based modes of interaction that allow users to manage their time as they would with a personal assistant.

One possible advance is to develop task management systems with conversational

interaction, overcoming the limitations of static interfaces. The user could ask, "What do I have to do today?", "What time is my doctor's appointment?", or "Remind me to take my medicine after lunch," receiving consistent, personalized and contextual responses.

This mode is not limited to event entry and consultation, but becomes a true digital diary that keeps track of past and future activities, providing a fluid experience. Users can interact by recounting their days, receive reminders based on their routines, and get empathetic and contextual responses that consider their personal profile.

An important reference is the work of Nasiri [20], who presents a chatbot based on language models combined with a dynamic knowledge graph. In this architecture, the user's sentences are segmented into triples (subject, predicate, object) using Stanford's OpenIE module, and stored as nodes and relations in a knowledge graph. To retrieve information, the system uses Sentence-BERT, a neural network that computes the semantic similarity between requests and stored information, generating accurate and relevant responses.

Another relevant contribution is CareCompanion [21], a virtual assistant to improve the quality of life for people with Alzheimer's and related dementias. The system uses NLP, personalized knowledge graphs, and speech recognition to provide intelligent reminders, contextual prompts, and time management assistance. CareCompanion offers continuous and natural interaction, allowing the user to converse to plan activities or request support, representing an evolution of TAT toward a more human and accessible conversational approach.

### 2.3 Ontologies

In computer science, an **ontology** is defined as a set of formal terms used to represent knowledge. Gruber [22] defines an ontology as *an explicit specification of a conceptualization*. A conceptualization is an abstract view of the world that is intended to be represented for a specific purpose. Knowledge representation is based on such a conceptualization, including objects, concepts and the relationships between them [23].

From a more technical perspective, Sowa [24] defines an ontology as a *specification of the types of entities that exist or can exist* in a given domain or domain. In more formal terms, an ontology can be represented as a collection of names designating concepts and relation types, organized according to a partial ordering established by the type/subtype relation.

According to Guarino [25], an ontology can be defined as an engineering artifact, consisting of a specific vocabulary used to describe a given reality, accompanied by a set of explicit assumptions regarding the intended meaning of the terms used in the vocabulary.

Neches et al. [26] view ontologies as *high-level abstract declarative hierarchies*, rich enough to establish the fundamental rules for modeling a domain. From this perspective, an ontology provides the basic terms, key relationships, and syntactic rules for combining these elements, thus constituting the conceptual infrastructure of a subject area.

Instead, Wielinga and Schreiber [27] define an ontology as a *theory of entities that can exist in the mind of an agent endowed with knowledge*. It can be viewed as a meta-model that describes the structure of a knowledge base.

### Role of ontology in the Semantic Web

The Semantic Web, as conceived by Tim Berners-Lee [28], is a vision of the Web in which information is endowed with machine-understandable meaning, enabling intelligent systems to automatically extract, integrate and reason about data published online [29]. In such a scenario, ontologies are critical because:

1. **They provide formal semantics to data:** thanks to standardized languages such as **OWL (Web Ontology Language)**, ontologies describe explicitly and computationally interpretable classes of objects, the relationships between them, and the logical constraints that govern their interaction.
2. **They support automatic inference:** by integrating ontological reasoners, data annotated according to an ontology can be processed to generate new implicit knowledge, improving the completeness and intelligence of answers to user queries.

3. **They enable semantic interoperability between heterogeneous sources:** ontologies act as a semantic bridge between datasets that, while describing the same domain, use different terminologies and structures. Through ontological alignment and semantic mapping, knowledge from distributed sources can be correlated and integrated.
4. **They enable data linking:** in the context of Linked Open Data, ontologies enable a shared and formalized definition to be associated with each resource, facilitating linking between datasets published on different servers in a uniform and cohesive manner.

### 2.3.1 Ontology components

The formalization of knowledge in ontologies is based on a well-defined set of components, generally recognized in the literature as [22]:

#### Classes or concepts

The classes, also known as *types* or *universals*, represent sets of individuals that share common characteristics. In a broad sense, a concept can refer to any entity about which statements can be made. Most ontology modeling languages (such as OWL or DAML) allow concepts to be defined on the basis of shared attributes. For example, all mammals share certain attributes, except for the ability to speak.

#### Relations

The relations describe how individuals (instances or particulars) are connected to each other. In other words, they represent forms of interaction between concepts belonging to the same domain. Formally, a relation is a subset of the Cartesian product of  $n$  sets, i.e.,  $R : C_1 \times C_2 \dots \times C_n$ . Ontologies can include different types of relations: *subclass-of* and *connected-to* are common examples of binary relations.

#### Functions

The functions are a special case of relations, in which the last element of the tuple is uniquely determined by the preceding elements. The *author-of* or *price-of-a-computer* relationships represent examples of functions. The latter, for example,

returns the price based on parameters such as CPU type and speed, memory capacity, and disk space.

### Axioms

The axioms represent statements formulated in formal logic that form the semantic core of the ontology within the application domain. They model propositions that are always considered true and are used to constrain the semantics of the model. Axioms can be classified into several categories based on their semantic meaning [30].

### Instances

The instances are individual entities that represent concrete objects, such as people, proteins or machines. They constitute the database of the ontology. Once the main components (concepts, relations, etc.) have been modeled, it is possible to implement the ontology in languages with different levels of formalization: from informal to semi-formal languages to strict formalisms [31].

## 2.3.2 Types and Classification of Ontologies

The classification of ontologies is an essential step in understanding their use, level of abstraction, and suitability for specific application scenarios. A first perspective places ontologies within the **Knowledge Organization Systems (KOS)** ecosystem, along with thesauri, taxonomies and semantic networks. In this context, ontologies are considered highly structured and functional relational models, distinguished by their ability to represent knowledge in a formal, explicit and computationally interpretable way, going far beyond the simple hierarchical structures of controlled vocabularies or traditional classifications [32].

From a technical perspective, ontologies can be classified according to various criteria, reflecting their , purpose, level of formalization, and expressiveness.

### Classification by level of complexity

This distinction is based on the conceptual and logical richness of the ontology structure, and the degree of formalization employed:

- **Light-weight Ontologies:** these are lightweight ontologies, characterized by a simple structure, often based on taxonomies or conceptual hierarchies. They are generally limited to the definition of classes, subclasses and "is-a" relationships, without complex logical constraints. They are mainly used for specific tasks and to facilitate interoperability between systems. A typical example is the use of SKOS for publishing thesauri in the Semantic Web.
- **Heavy-weight Ontologies:** these are highly expressive ontologies that include logical axioms, cardinality constraints, inverse properties, semantic restrictions and other complex components. They support automatic inferences and semantic reasoning. Such ontologies are suitable for critical contexts such as healthcare, engineering or intelligent systems, where modeling semantics with precision and consistency is essential.

### Classification by Purpose or Scope (Ontological Commitment)

This criterion is based on the concept of ontological commitment, i.e., the type of knowledge that the ontology intends to represent. Within this scope we distinguish:

- **Top-level ontologies:** describe general, abstract categories such as *entity*, *event*, *time*, *space* or *causal relationship*, independent of specific domains. They serve as a common conceptual reference and are useful for ensuring interoperability between domain ontologies. Known examples include DOLCE, SUMO and GFO.
- **Domain Ontologies:** represent specific concepts related to a particular application domain, such as medicine, biology, finance or law. For example, FIBO (Financial Industry Business Ontology) is an ontology for the financial industry, while SNOMED CT is widely used in healthcare.
- **Task Ontologies:** describe the concepts and relationships needed to perform recurring tasks such as diagnosis, classification, planning or maintenance. They are often reusable across domains, as they formalize the *what is to be done* rather than the *where*.
- **Method Ontologies:** represent solution strategies for specific tasks, such as *propose-and-revise*, *generate-and-test* or *design-by-analogy*. They focus on *how to* perform a task rather than on the specific content.

### Classification by formality and expressiveness

A further classification is based on the **level of formalization** and **expressiveness** of the language used to represent the ontology. In this area, distinctions are made:

- **SKOS (Simple Knowledge Organization System)**: while not an ontology language in the full sense, SKOS allows thesauri, taxonomies and classifications to be represented in RDF format. It is ideal for migrating traditional KOS to the Semantic Web, but it has limited expressive capabilities and does not support complex inferences.
- **RDFS (RDF Schema)**: allows the definition of classes, subclasses, properties, domain and ranges, offering formal but relatively simple semantics. It is often used for shared vocabularies, but does not support advanced logical axioms.
- **OWL (Web Ontology Language)**: is the language of choice for defining formal ontologies in the Semantic Web. OWL allows the definition of complex classes, logical restrictions, equivalences, disjunctions, inverse properties and cardinality. With the support of logical reasoners such as Pellet or Hermit, it allows automatic reasoning based on represented knowledge.

These languages reflect different levels of trade-offs between expressiveness, computability, and interoperability, and their choice depends on the specific goals of the ontology and the application context.

#### 2.3.3 Languages and standards for ontology modeling

Languages for ontology modeling are a fundamental building block for the formal representation of knowledge. They provide the syntactic and semantic tools for building ontologies that can be processed by machines in an automatic, interoperable, and reasoned manner.

A good ontology language must meet a number of requirements:

- it must be sufficiently expressive to represent the concepts and relationships of a complex domain;
- it must have a well-defined formal semantics that allows for automatic interpretation;

- it must be readable and manageable by both human experts and automated systems;
- it must support inference mechanisms, allowing new knowledge to be derived from known facts.

Over the years, several languages and standards have been developed to meet these needs, each with different levels of complexity and expressive power. The main ones are presented below:

### RDF (Resource Description Framework)

RDF [33] is the standard proposed by W3C for the representation of structured information on the Web. It is based on a labeled directed graph model, in which the fundamental units of information are RDF triples, each consisting of a subject, a predicate, and an object. This schema allows statements about entities and relationships to be described in a simple but flexible way, through a structure that can be easily extended and integrated.

RDF does not impose a rigid semantics on relationships, but provides a generic framework for describing resources. However, to introduce hierarchical structures, class definitions, and properties, reference is made to RDFS (RDF Schema), an extension of RDF.

One of the strengths of RDF is its interoperability: RDF data can be shared and understood by different systems, thanks to the use of global identifiers (URIs) and a standardized structure. In addition, RDF underlies many Semantic Web technologies, including Linked Open Data, SPARQL, OWL, and SKOS.

### RDFS (RDF Schema)

RDFS [34] extends RDF by introducing concepts typical of semantic modeling, such as classes, subclasses, properties, and domains. It thus allows information described using RDF to be structured and typed, while maintaining a lightweight and flexible approach.

With RDFS it is possible, for example, to define that a given property (e.g., hasAuthor) has the class Book as its domain and the class Person as its range.

This type of assumption improves the ability of systems to interpret and validate statements, enabling basic forms of semantic inference.

However, RDFS has significant limitations in terms of expressiveness: it does not allow the expression of complex logical constraints, such as cardinality, disjunction between classes, inverse properties, or equivalence between concepts. For these reasons, in cases where greater semantic rigor is needed, OWL is used.

### **OWL (Web Ontology Language)**

OWL [35] represents the most expressive language among those recommended by W3C for ontology modeling. Based on description logics (Description Logics), OWL allows the formal definition of concepts, relationships and constraints, supporting automatic reasoning via reasoners.

An OWL ontology can include:

- definitions of complex classes via intersections, unions, complements;
- properties with features such as symmetry, transitivity, functionality;
- cardinality constraints, restrictions on the value of properties;
- logical axioms (equivalence, disjunction, inclusion, inconsistency).

OWL is available in three main variants:

- **OWL Lite**: suitable for simple applications, it allows hierarchical modeling with elementary restrictions.
- **OWL DL**: ensures decidability and completeness of inferences, offering extensive support for descriptive logic (compatible with SROIQ logic).
- **OWL Full**: maximally expressive, allows RDF and OWL to be freely combined, but loses the guarantees of efficient automatic reasoning.

### **SKOS (Simple Knowledge Organization System)**

SKOS [36] is an RDF-based language designed to represent controlled vocabularies, thesauri, taxonomies, and classification schemes in a manner compatible with the Semantic Web. It is designed to provide a simplified model suitable for

representing conceptual structures without requiring the formal rigor of OWL.

With SKOS it is possible to:

- represent concepts (`skos:Concept`) and relationships between them (`skos:broader`, `skos:narrower`, `skos:related`);
- manage multilingual labels (`skos:prefLabel`, `skos:altLabel`);
- associate descriptive notes, definitions and references with each concept.

#### 2.3.4 Semantic Inference

Semantic inference represents one of the most significant capabilities of ontology-based systems, and is what distinguishes them from simple data schemas. It allows new implicit knowledge to be derived from explicit statements through the application of logical rules defined within the ontology model.

Semantic inference is based on the use of *reasoners* (inference engines) that operate on ontologies built using formal languages, such as OWL (Web Ontology Language), based on descriptive logics. Such reasoners are able to:

- **Automatically classify classes**, inferring implicit hierarchies between concepts;
- **Check the logical consistency** of the ontology, detecting contradictions between axioms or statements;
- **Deduce membership of individuals in classes**, based on the properties they possess;
- **Inferring implicit relations**, such as by applying transitivity or symmetry of properties.

From a theoretical point of view, inference is possible because of the formal semantics provided by SROIQ(D) descriptive logic, which forms the basis of OWL DL, ensuring decidability and computational tractability. This makes it possible to effectively integrate complex ontologies into intelligent systems.

Popular inference engines include Hermit, which uses advanced techniques for

inference on OWL 2 ontologies; Pellet, which fully supports OWL DL and enables consistency checking; and FaCT++, designed for high performance on large ontology bases.

However, inference also comes with challenges. It can be computationally expensive, especially when applied to large datasets or highly expressive ontologies. In addition, it can produce redundant or unwanted inferences if the ontology is not modeled rigorously. Therefore, it is critical that ontologies are carefully designed, with clear definitions of classes, properties, and semantic constraints.

## 2.4 Knowledge Graphs

In recent years, Knowledge Graphs (KGs) have emerged as a powerful paradigm for the organization, representation and processing of knowledge, especially in contexts characterized by complex, interconnected and dynamic information. These graphs provide a structured semantic representation of entities, their attributes and the relationships between them, enabling not only effective knowledge management but also the activation of automatic inference and reasoning processes.

However, the exact definition of what constitutes a Knowledge Graph is not unique: it varies depending on the application contexts and theoretical perspectives adopted.

According to one of the most widely cited definitions, proposed by Ehrlinger and Wöß [37], a Knowledge Graph can be formally understood as:

”a knowledge base that integrates information within an ontology and applies an inference engine to derive new knowledge.”

This formulation emphasizes three key aspects: the semantic formalization of information through the use of ontologies; the integration of data from heterogeneous sources; and the possibility of generating new implicit knowledge through automatic inference mechanisms. From this perspective, a Knowledge Graph is not a static repository of data, but rather a dynamic system that can evolve over time and be continuously enriched.

## 2. State of the art

---

Several other authors have contributed to the definitional debate, offering complementary points of view with their definitions, summarized in Table 2.1.

Definition	Source
”A knowledge graph (i) mainly describes real world entities and their interrelations, organized in a graph, (ii) defines possible classes and relations of entities in a schema, (iii) allows for potentially interrelating arbitrary entities with each other and (iv) covers various topical domains.”	Paulheim [38]
”Knowledge graphs are large networks of entities, their semantic types, properties, and relationships between entities.”	Journal of Web Semantics [39]
”Knowledge graphs could be envisaged as a network of all kinds of things which are relevant to a specific domain or to an organization. They are not limited to abstract concepts and relations but can also contain instances of things like documents and datasets.”	Semantic Web Company [40]
”We define a Knowledge Graph as an RDF graph. An RDF graph consists of a set of RDF triples where each RDF triple ( $s, p, o$ ) is an ordered set of the following RDF terms: a subject $s \in U \cup B$ , a predicate $p \in U$ , and an object $\in U \cup B \cup L$ .”	Färber et al. [41]
”[...] systems exist, [...], which use a variety of techniques to extract new knowledge, in the form of facts, from the web. These facts are interrelated, and hence, recently this extracted knowledge has been referred to as a knowledge graph.”	Pujara et al. [42]

Table 2.1: Selected definitions of Knowledge Graphs from the literature

Paulheim [38], for example, describes a KG as a graph representation of real entities and their relationships, supported by a conceptual schema that defines their properties and constraints.

The Journal of Web Semantics [39] expands the definition, describing them as ”large networks of entities, their semantic types, properties and relationships,” emphasizing the scalable and multidimensional aspect.

The Semantic Web Company [40] shifts the focus to the practical value of the graph, defining it as ”a network of all things relevant to a specific domain or

organization, encompassing both abstract concepts and concrete resources such as documents or datasets.”

Other authors, such as Färber et al. [41], take a more technological view, identifying the Knowledge Graph with an RDF graph composed of triples (subject, predicate, object), while Pujara et al. [42] see it as a system that can automatically extract facts from the web and integrate them to extend a knowledge base.

Despite the variety of definitions, a consensus emerges on some key features that distinguish Knowledge Graphs from other data structures: the semantic organization of information, the oriented graph structure, the ability to integrate heterogeneous data, and the activation of automatic inferential processes.

#### **2.4.1 Historical Evolution**

The concept of Knowledge Graph (KG) has evolved significantly over the past two decades, rooted in the early developments of the Semantic Web and the emergence of graph databases. The Semantic Web, promoted by Tim Berners-Lee in the early 2000s, aimed to transform the World Wide Web into an environment in which data could be shared, understood and processed automatically by both humans and machines. In that vision, fundamental tools such as RDF (Resource Description Framework) for representing data, OWL (Web Ontology Language) for defining formal ontologies, and SPARQL, a standardized query language for accessing RDF graphs, were introduced[43].

In parallel, interest in more expressive and flexible data structures has fostered the emergence of graph databases, which are capable of modeling complex connections between entities in a more natural way than traditional relational databases. This need emerged particularly in dynamic areas such as information retrieval, natural language processing (NLP) and enterprise knowledge management, where the rigidity of relational models showed obvious limitations in representing interconnected and unstructured knowledge.

A crucial turning point occurred in 2012 with the launch of the Google Knowledge Graph [44], a technology that has profoundly affected the way search engines present results. For the first time, Google introduced a system that could provide

semantic and contextual answers based on real entities and their relationships, rather than just textual matches. For example, a search on "Leonardo da Vinci" returned not just documents containing the name, but a structured information box with biographical details, major works, and historical and artistic connections. This innovation concretely demonstrated the potential of KGs in improving the user experience and catalyzed strong interest in academia and industry as well.

In subsequent years, the adoption of Knowledge Graphs intensified, supported by the development and standardization of technologies such as RDF and SPARQL, which facilitated interoperability and semantic enrichment among heterogeneous data sources. Projects such as DBpedia, Wikidata, and the Linked Open Data Cloud have promoted the creation of large-scale open graphs, providing solid foundations for knowledge sharing and integration on a global scale [45].

Simultaneously, there has been the spread of enterprise Knowledge Graphs, a "second wave" of development in which private and public organizations have begun to build internal graphs to support activities such as heterogeneous data integration, decision support, business intelligence, and user experience personalization. These graphs are often built on domain-specific ontologies and include both structured and semi-structured data from CRM, ERP, social media, system logs, and other sources [45].

Today, Knowledge Graphs emerge as key infrastructures in the artificial intelligence ecosystem. Their use is particularly relevant in applications that require semantic reasoning, contextual understanding, and explainability, such as recommendation engines, question answering systems, personalized medicine, Industry 4.0, and cognitive robotics. Because of their semantic structure and ability to represent and infer new knowledge, KGs today represent one of the key technologies for making intelligent systems truly inclusive, adaptive, and transparent.

#### 2.4.2 Graph data models for knowledge representation

In the context of *Knowledge Graphs (KGs)*, data models play a key role in defining the structure and representation of knowledge. The choice of model directly affects how the graph is constructed, stored, queried and, more generally, used. Popular models for KG representation include the *Resource Description Frame-*

*work (RDF)* and the *Property Graph Model (PGM)*, also known as *Labeled Property Graph (LPG)* [45].

### Resource Description Framework (RDF)

The *Resource Description Framework (RDF)* is one of the most established models in the context of the Semantic Web. Created to describe metadata on the Web, RDF is now considered the leading standard for formal knowledge representation, and is supported by a robust ecosystem of technologies standardized by the W3C.

The structure of RDF is based on the concept of a triple, a statement of the type <subject, predicate, object>. The subject represents an entity, the predicate the relation, and the object can be another entity or a literal value. Resources (subjects and objects) are uniquely identified using IRI (Internationalized Resource Identifier).

RDF is accompanied by schema languages such as RDFS and OWL, which allow the definition of classes, properties, hierarchies, and logical constraints. In addition, the SHACL standard allows validation of RDF data according to specific form constraints, as is done, for example, in the semantic extension schema.org.

Queries on KG RDF are performed using *SPARQL*, the official query language of the W3C. RDF is the technological basis for well-known projects such as DBpedia, Wikidata (with RDF mappings), and many other linked datasets in the *Linked Open Data* cloud.

### Property Graph Model (PGM)

As an alternative to RDF, the *Property Graph Model (PGM)*, particularly used in native graph databases, has emerged over time.

This model is based on nodes and directed arcs. Its distinguishing feature is the ability to assign labels that define type and, more importantly, properties (in the form of key-value pairs) to both nodes and arcs. This facilitates the immediate association of specific metadata not only to entities but also to the relationships themselves [46].

Resource identification in the PGM is generally local to the system, as there is no standardized mechanism such as IRI. In addition, the PGM does not have native support for formal ontologies such as RDFS/OWL, but can be extended with external tools.

The Property Graph is supported by many databases, including Neo4j, JanusGraph and TigerGraph, and by dedicated query languages such as *Cypher* [47], *Gremlin* [48], *PGQL* [49], and *G-Core* [50].

### RDF vs Property Graph

While RDF represents a largely standardized solution with strong support for formal semantics and interoperability through open standards, PGM offers greater flexibility in data modeling and more direct and intuitive handling of metadata associated with relationships.

RDF's triple model ensures a uniform representation of entities and relationships, while also allowing powerful semantic inferences to be constructed through the use of ontologies such as RDFS and OWL. However, this structure can be difficult to interpret without additional processing or inference processes, as information about an entity is often distributed across multiple distinct triples. Although the introduction of RDF-Star has improved expressiveness in metadata management, practical limitations persist: for example, it is not easy to correctly model homonymous relationships with different properties without resorting to supporting constructs.

In contrast, in PGM, each relationship between two entities can have its own distinct properties, even if it shares the same name. This makes the Property Graph particularly suitable for complex scenarios such as *social network analysis*, deep graph traversing, and advanced network analysis applications, where efficient and differentiated relationship management is crucial. In addition, the closeness between the conceptual model and the physical model in native graph databases greatly simplifies knowledge graph design and evolution [45].

That said, neither model is perfect for every type of application. The adoption of one or the other depends strongly on the specific needs of the project:

- If the goal is to ensure maximum interoperability, leverage shared ontologies, and support formal semantic inferences, *RDF* is the most appropriate choice.
- If, on the other hand, the focus is on high performance, flexibility in handling complex data, rapid schema evolution, and network analysis, the *Property Graph Model* offers concrete advantages and greater development agility.

### 2.4.3 Internal structure of a Knowledge Graph

Regardless of the representation model adopted (RDF, Property Graph or hybrid models), a *Knowledge Graph (KG)* is based on a common conceptual structure that determines its internal organization. This structure is what enables the graph to represent knowledge in a way that is queryable, interpretable and navigable, both by machines and humans [51].

#### Entities

Entities represent the fundamental elements of a knowledge graph. They are distinct real-world objects or concepts that have their own identity and are relevant to the knowledge domain that the graph is intended to model.

Each entity is usually represented as a node within the graph. Each entity is associated with unique information that distinguishes it from other entities, such as an identifier, a textual label, and a set of attributes that describe its characteristics.

A practical example of entities in a knowledge graph is illustrated in Figure 2.1, where different types of objects are represented with their properties: a `Person`(`Name:Daniel`), an `Order`(`ID:100`), and a `Product`(`Name:Salmon`).

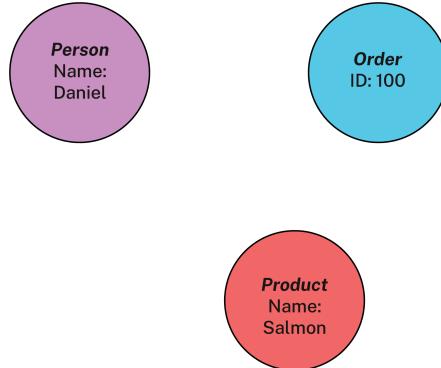


Figure 2.1: Example: nodes in an e-commerce graph

## Relationships

Relationships constitute the semantic links between entities. Each relationship expresses a meaningful connection, defining how two entities are related to each other. In a knowledge graph, relationships are represented as oriented arcs that join nodes corresponding to the entities involved.

Relationships are not simply meaningless links: each relationship has its own label or type, which specifies the kind of connection that exists. A concrete example of a relationship in a knowledge graph is shown in Figure 2.2, where the entity `Person(Name: Daniel)` is connected to the entity `Order(ID: 100)` through a `PLACED_ORDER` relationship.



Figure 2.2: Example: relationships in an e-commerce graph.

The variety of relationships that can be represented allows the knowledge graph to model a rich and articulated network of knowledge, where the context and meaning of information is not only stored, but also understandable by machines. This is one of the key elements that distinguishes a knowledge graph from a simple database.

## Attributes

Attributes enrich the representation of entities (and, in some cases, relationships as well) by providing additional descriptive details. While entities and relation-

ships define the structure of the graph, attributes serve to better characterize each element, offering additional quantitative or qualitative information.

Attributes in a knowledge graph are represented as key-value pairs associated with nodes or arcs. In the example shown in Figure 2.1, there are attributes for each node: a **Person** named Daniel, an **Order** with ID 100, and a **Product** named Salmon.

### Organizational Principles

Organizational principles define how information within the graph is structured and interpreted, giving it coherence, meaning and the ability to support more complex reasoning.

In practice, organizational principles are translated into structures such as taxonomies, conceptual schemas or true formal ontologies. Through them, the knowledge graph not only collects entities and relationships, but also clearly specifies what types of entities can exist, how they can be related to each other, what properties they may possess, and what rules must be followed to ensure the consistency of the knowledge represented.

A concrete example of organizing principles within a knowledge graph is illustrated in Figure 2.3, where different types of foods, such as **Snacks**, **Fruits**, **Fresh Foods**, and **Fish**, are linked to the higher category **Foodstuffs** through the **TYPE\_OF** relationship. In this scheme, the hierarchical relationships define that each specific type is a specialization of the more general concept of foodstuffs.

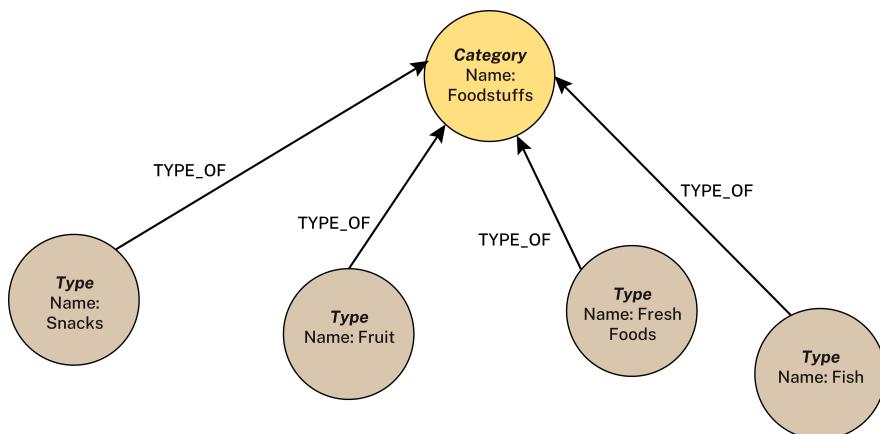


Figure 2.3: Example: classification schema in an e-commerce graph.

This example shows how ontologies or conceptual schemas guide not only the classification of entities, but also the structure of the graph itself, making it possible to automatically infer memberships, implicit relationships and semantic connections between data.

The principles make it possible to:

- **Correctly interpret the meaning of information**, making the graph understandable to both users and machines;
- **Validate the semantic coherence** of the data, ensuring that the information entered complies with the rules established by the schema or ontology;
- **Enable automatic inference**, allowing systems to infer new knowledge from existing knowledge, thus dynamically extending the information network.

#### **2.4.4 Knowledge Graph construction**

Knowledge Graph (KG) construction is a complex and multifaceted process aimed at extracting, organizing, and structuring knowledge from heterogeneous sources—structured, semi-structured, and unstructured—such as websites, corporate databases, scholarly publications, or user-generated content on social media. This process is essential for transforming large volumes of raw data into semantic representations that can be queried, navigated, and interpreted by both humans and intelligent systems.

As noted in [43], this process is typically decomposed into three basic dimensions, also illustrated in the taxonomy shown in Figure 2.4.

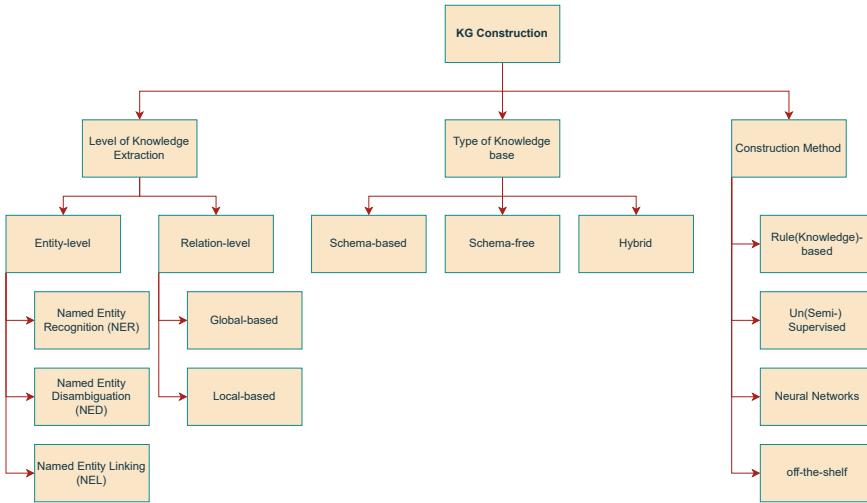


Figure 2.4: Taxonomy of Knowledge Graph Construction dimensions.

### Level of knowledge extraction

The first dimension concerns the extraction of entities and relationships from the source data [52]. As far as entities are concerned, three main subphases are distinguished:

- **Named Entity Recognition (NER)**: consists of automatic identification of mentions of relevant entities (e.g., people, organizations, places, events) within natural or semi-structured texts. This step enables the recognition of information units that will later be represented as nodes in the graph.
- **Named Entity Disambiguation (NED)**: once recognized, entities must be disambiguated, that is, traced back to unambiguous concepts of reality. This step is critical to avoid ambiguity (e.g., "Apple" as a company or as a fruit) and to ensure the consistency of the knowledge graph.
- **Named Entity Linking (NEL)**: disambiguated entities are finally linked to unique identifiers (IRIs - Internationalized Resource Identifiers), enabling the integration of data from different sources and the creation of robust semantic connections.

In parallel, relationships are extracted:

- *Local relations* refer to connections explicitly expressed within individual documents or sentences (e.g., "Marie Curie discovered radium").

- *Global relations* infer implicit connections between entities that can only emerge by aggregating information from multiple different sources or documents.

### Type of Knowledge Base

The second dimension concerns the conceptual model adopted to organize entities and relationships. Three main strategies are distinguished in the literature:

- **Schema-based Knowledge Graph:** the construction of the graph is guided by a predefined ontology or schema. Entities, properties and relationships must strictly adhere to an existing formal structure, ensuring high semantic consistency and interoperability.
- **Schema-free Knowledge Graph:** extraction occurs without the support of a predefined conceptual model. Techniques such as *Open Information Extraction (OpenIE)* are used to extract entities and relationships directly from textual data, without a formal constraint of schema membership.
- **Hybrid Approach:** combines the two paradigms, using reference ontologies for the main structure and leaving room for dynamic data expansion through schema-free techniques.

### Construction method

Finally, the third aspect concerns the techniques **and methods** adopted to construct the Knowledge Graph. These can be classified into four main categories:

- **Knowledge-based approaches:**

Knowledge-based approaches represent the most traditional methodology for KG construction. In this case, knowledge is formalized through rules created by domain experts, using controlled vocabularies, thesauri and ontologies. The extraction process is then guided by a predefined semantic schema, which allows high levels of consistency, interpretability and accuracy to be maintained.

A representative example is DBpedia, in which structured data in Wikipedia infoboxes are mapped to a pre-existing ontology. In highly regulated domains, such as health care or finance, these approaches are particularly effective because they ensure strict control of semantics.

However, manual construction of rules and schemas requires a significant investment in time and expertise, and is difficult to scale in large or rapidly changing domains.

- **Learning-based methods:**

With the increasing availability of annotated data, there has been widespread use of machine learning algorithms to automate some of the key steps in KG construction, particularly the extraction of entities and relationships. These methods are often supervised or semi-supervised, and make use of classical statistical models.

These models learn to recognize semantic structures in textual data from manually annotated corpora. Compared to knowledge-based approaches, learning methods offer greater flexibility and good adaptability to new domains, but they tend to be sensitive to the quality of training data and require a sufficient number of annotated examples to be effective.

- **Neural network models:**

The emergence of deep learning has opened up new possibilities for the automatic construction of KGs, reducing the need for feature engineering and improving the generalization ability of systems. Neural models, such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs) or BiLSTM-based models, have been successfully employed for end-to-end extraction of entities and relationships from natural texts.

More recently, the use of Graph Neural Networks (GNNs) has made even more sophisticated modeling possible by exploiting the topological structure of the graph to predict new connections (knowledge graph completion). For example, models such as ConvE and RotatE have been used to represent RDF triples (subject, predicate, object) in latent vector spaces, facilitating the discovery of implicit knowledge.

These approaches are particularly suitable in open-domain contexts, where relationships between entities are not always explicit and data can be highly variable. However, their application requires significant computational resources and large datasets for training.

- **Off-the-shelf NLP tools:**

As an alternative to custom models, there are now a number of NLP tools and libraries that offer integrated pipelines for information extraction, enabling more immediate and accessible construction of KGs, even in less specialized contexts.

Libraries such as [spaCy](#), [Stanford CoreNLP](#), [AllenNLP](#) and commercial solutions such as [IBM Watson NLU](#) provide ready-made capabilities for entity recognition, syntactic analysis, semantic classification and relation extraction.

These tools are very useful when working with few resources or tight schedules, and are often used to create prototypes or early versions of the system. However, they can have limitations if they need to be adapted to very specific or customized contexts.

### Current challenges in Knowledge Graph construction

Despite methodological and technological advances, the construction of large-scale, high-quality Knowledge Graphs still presents significant challenges:

- **Integration of heterogeneous and large-scale data:** One of the main problems in KG construction is the integration of data from different sources, with very different structures and formats. When working on a very large scale, with billions of data, the problem becomes even more complex and requires advanced solutions to manage and harmonize the information.
- **Automation vs. manual labor:** Creating a KG entirely by hand is expensive and unsustainable. Automated techniques, such as machine learning and deep learning, are increasingly used, but they are still unable to handle very complex data independently. For this reason, the most realistic approach today is a semi-automated one, in which automated tools are complemented by human intervention, for example, to validate results or curate ontologies.
- **Representing complex knowledge:** KGs still struggle to model advanced concepts such as temporal dimension, data versions, degree of uncertainty, context, or complex relationships. These aspects require more expressive models that are able to go beyond simple true/false representation.

- **Ensure quality and curation:** A Knowledge Graph is only useful if the data are accurate and reliable. However, finding and correcting errors on very large graphs is not easy, partly because there is a lack of standardized tools and benchmarks to assess quality in a systematic way.
- **Limitations of data models:** The RDF model, although widely used, has some limitations. For example, it does not handle properties that change depending on the type of entity well, and representing metadata on relationships can be complex.
- **Update management:** KGs need to be updated frequently to remain relevant. However, integrating new data efficiently, especially if they arrive at high rates, requires infrastructure and strategies that are not yet fully mature.
- **Metadata management:** In addition to "core" data, a KG must also manage metadata such as provenance, reliability, or temporal validity. Integrating all this information in a consistent way is not trivial.
- **Standardization:** While RDF has an established ecosystem of standards, other models—such as Property Graph—have not yet reached the same level of formalization. This can create interoperability difficulties between different tools and platforms.

#### 2.4.5 Knowledge Graph querying

Querying a Knowledge Graph (KG) represents the process by which the knowledge contained in the graph is made accessible and usable by intelligent applications, such as semantic search engines, *Question Answering* (QA) systems and virtual assistants. This field of research is constantly expanding, fueled by advances in machine learning and Natural Language Processing (NLP).

Compared to relational databases, KG queries have distinctive characteristics: they are highly dependent on join operations, frequently involve many-to-many relationships, and may require multiple traversals or recursive operations. These complexities make query optimization and management of intermediate results particularly onerous.

Key challenges also include schema flexibility (allowing the same information to be represented in different ways), the need for advanced semantic matching techniques, and automatic reformulation of queries to ensure completeness and accuracy of results.

In addition, because many KGs adopt the open world assumption, it is often necessary to infer missing knowledge to provide comprehensive answers. Finally, the design of intuitive query interfaces, accessible even to nonexpert users, is a crucial element for the widespread adoption of KGs [53].

### **Formal query languages**

To query KGs accurately, several formal languages have been developed, each optimized for certain graph models and application requirements.

The best known in the RDF context is SPARQL, standardized by the W3C, which allows structured patterns to be expressed in the form of subject-predicate-object triples. SPARQL is widely used in Linked Data datasets, such as DBpedia and Wikidata.

In the relational world, extensions of SQL have emerged to support graph structures, including Microsoft SQL Graph and OrientDB, which combine relational and graph features.

For Property Graph, we find native languages such as Cypher (Neo4j), PGQL (Oracle) and GSQL (TigerGraph), designed to express patterns of nodes and arcs with properties. These are joined by Gremlin, a graph traversal language that integrates imperative and declarative modes, and is supported by many graph platforms.

Finally, some approaches are based on extensions of Datalog, a declarative logic language well suited for formulating recursive and complex queries.

### **Keyword Search**

Keyword search is a simple and accessible query mode in which the user provides only a list of terms. The system typically returns sub-graphs or trees containing

those terms, using ranking strategies based on measures such as weight of arcs or distance between relevant nodes [54].

This mode is useful for non-technical users, but can introduce semantic ambiguities. To improve its accuracy, some solutions automatically translate the query into SPARQL by exploiting semantic similarity and co-occurrence techniques between terms and concepts in the graph.

### Query-by-Example (QBE)

Query-by-Example (QBE) allows users to provide concrete examples—such as one or more tuples linking entities together—as a basis for finding similar structures in the graph. The system implicitly builds a query from these examples and executes it to return similar results [55].

This approach is particularly suitable for users who are familiar with the domain but not with formal languages, and typically consists of two steps: extraction of the example pattern and subsequent matching on the KG [56].

### Faceted Search

Faceted search is an interactive, incremental exploration method that guides the user in progressively selecting categories, properties, and values (the “facets”). With each interaction, the system dynamically updates the visible results, allowing progressive refinement of the search [57].

It is very effective in complex information environments, such as cultural portals, digital museums, or biomedical knowledge bases, where the user may have needs for exploration rather than point queries.

### Visual Query Interfaces

Visual query interfaces allow users to construct queries by directly drawing a graphical pattern (nodes, arcs, labels), which the system translates into a structured query (e.g., SPARQL). This is useful for users with knowledge of graph structure but little familiarity with query languages. These interfaces implement sub-graph matching and enumeration techniques to return results [58].

A well-known example is the GORELATIONS [59] interface for DBpedia, which allows users to explore and query semantic relationships graphically.

### Natural Language Querying (NLQ)

Natural language querying represents a very promising area for making KGs accessible to those without technical skills. The goal is to allow the user to ask questions such as “Who wrote The Name of the Rose?” and get a direct answer from the graph [60].

NLQ systems use neural models (sequence-to-sequence, ranking, classification) to interpret the question, identify entities and relationships, and generate an equivalent query.

### Conversational KG Question Answering (CQA)

Conversational QA extends the capabilities of NLQ to conversational scenarios. In this context, users can ask a sequence of interrelated questions with implicit reference to the previous context. The system maintains the memory of the dialogue, dynamically updating the intent and references [61].

This approach is the basis of intelligent chatbots and virtual assistants, capable of supporting smooth user interactions.

### Multimodal Question Answering (MMQA)

Multimodal QA extends KG questioning beyond written text to include images, audio, video, or tables. In this scenario, multimodal content is associated with graph entities through shared *entity linking* and embedding techniques, which map the different modalities to a unified semantic space [62].

### Embedding-based Query Processing

In recent years, the use of embeddings in Knowledge Graphs (KGs) has gained increasing attention as an effective strategy to support the querying of complex graphs, especially in contexts where the data is incomplete, the schema is flexible or uncertain, or a semantic inference component is needed.

In this approach, each entity and relation in the graph is mapped into a low-dimensional vector space, where the structural and semantic properties of the original graph are approximately preserved. In other words, the information encoded in the graph is translated into geometric shapes that allow mathematical operations to be performed on them. This makes it possible to transform complex queries into simple distance or similarity calculations between vectors.

Querying a KG by embedding thus results in a vector matching problem: for example, if one knows the vector of an entity (head) and that of a relation, it is possible to predict the missing tail vector (tail) in a triple. This is the principle behind applications such as link prediction, query completion, and complex query answering.

#### 2.4.6 Application Scenarios

Knowledge Graphs have profoundly innovated the way knowledge is represented and managed. Thanks to their graph structure, they facilitate a deeper understanding of information, facilitating automatic reasoning and contextualization of data, both for humans and intelligent systems. For these reasons, KGs are now used in numerous application domains, contributing decisively to solving concrete problems.

##### Generic Knowledge Graphs

Generic Knowledge Graphs—also known as *open-world*, *cross-domain* or *domain-independent*—have existed since before the term “knowledge graph” was formally coined. Since the beginning of the Semantic Web era, these graphs have been associated with the concept of *linked data*, configuring themselves as ideal tools for representing and interconnecting entities from different domains [37].

Their goal is to offer a broad, cross-domain view of real-world knowledge by integrating heterogeneous data from a multiplicity of sources. Not surprisingly, many open KGs now form the basis for artificial intelligence systems, recommendation engines, decision support systems (DSS), and intelligent Question Answering (QA) systems [63, 64, 65, 66].

The following are the main examples of Open Knowledge Graphs:

- [DBpedia](#), one of the earliest and most relevant Knowledge Graphs, published in 2007. It was extracted mainly from the structured data on Wikipedia pages, making encyclopedic information accessible in RDF format and linking it to other open datasets, thus becoming a central node of the Semantic Web. DBpedia is widely used for applications in natural language processing, knowledge expansion, recommender systems, and semantic search.
- [Wikidata](#), launched in 2012 as a sister project of Wikipedia, aims to be a collaborative knowledge base, accessible and editable directly by users. Like DBpedia, it collects encyclopedic information, but is distinguished by a focus on the provenance and context of the information. Each piece of data, such as the population of a city, is enriched with qualifiers (e.g., statistical source) that make the knowledge more reliable and verifiable. Wikidata uses a custom data model that supports n-ary relationships and explicit context. It is connected to the Semantic Web via RDF and OWL mappings and provides SPARQL endpoints for querying. It is now one of the most widely used open knowledge graphs in question answering, artificial intelligence, and semantic Web content enrichment projects.
- [YAGO](#) (Yet Another Great Ontology), published in 2008, is built by combining data extracted from Wikipedia with the lexical structure of WordNet in order to enrich the hierarchy of conceptual types. Unlike DBpedia, YAGO focuses on semantic accuracy, extracting a limited number of relations to keep the graph compact but highly accurate. It is particularly popular in academia for applications in logical reasoning, semantic geography, and multilingualism.
- [Freebase](#), launched in 2007, was a collaborative knowledge base that aimed to collect real-world facts in a structured way. The company that ran Freebase was acquired by Google in 2010, and the knowledge base served as the initial foundation for the Google Knowledge Graph.
- [NELL](#) (Never-Ending Language Learning) is a system started in 2010 and designed to autonomously learn knowledge from the web in a continuous manner. It analyzes about 500 million web pages, using a predefined ontology to guide the extraction of entities and relationships. NELL "agents" operate continuously, adding new facts and deleting obsolete or erroneous ones.

- [KBpedia](#) is a knowledge base designed to support artificial intelligence and machine learning applications. Launched in 2016 and made open source in 2018, it integrates knowledge from multiple authoritative sources, including Wikipedia, Wikidata, schema.org, DBpedia, GeoNames, OpenCyc and UMBEL. It is particularly useful in building training sets for machine learning algorithms and knowledge-based inference systems.
- [Datacommons.org](#) is an open Knowledge Graph project launched by Google in 2018, created to integrate public sources and make socio-demographic, economic, environmental and geographic data available in a unified form. It uses a schema inspired by schema.org, slightly extended, and represents data as RDF triplets with associated source values.

	Year	Domain	Model	Entities	Relations	Types	Update
Freebase	2007	Cross	RDF	22M	3.2B	53K	2016
DBpedia	2007	Cross, MLang	RDF	50M	21B	1.3K	2023
YAGO	2007	Cross	RDF(-Star)	67M	2B	10K	2020
NELL	2010	Cross	Custom, RDF	2M	2.8M	1.2K	2018
Wikidata	2012	Cross, MLang	Custom, RDF	100M	14B	300K	2023
Artist-KG	2016	Artists	Custom	161K	15M	1	1016
AI-KG	2020	AI Science	RDF	820K	1.2M	5	2020
CovidGraph	2020	COVID-19	PGM	36M	59M	128	2020
VisualSem	2020	Cross, MLang	Custom	90K	1.5M	49K	2020
WorldKG	2021	Geographic	RDF	113M	829M	1176	2021

Table 2.2: Comparison of representative open knowledge graphs.

### Domain-specific Knowledge Graph (DSKG)

Alongside large open, multi-domain knowledge graphs, research has developed a growing focus on Domain-specific Knowledge Graphs (DSKGs), designed to represent knowledge related to highly specialized contexts such as medicine, finance, engineering, or education.

A Domain-specific Knowledge Graph is defined by [43] as:

”an explicit conceptualization to a high-level subject-matter domain and its specific subdomains represented in terms of semantically interrelated entities and relations”

## 2. State of the art

---

These graphs are distinguished by three basic features: they are based on rigorous ontologies or schemas that logically describe entities, properties and relations relevant to the domain; their content is focused on a specific area, such as cardiology, law, or robotics; and the connections between entities follow precise patterns designed to support automatic inference, efficient queries, and reliable analysis.

Unlike open KGs, which tend to emphasize breadth and variety, DSKGs focus on semantic depth, precision of patterns, and relevance of data to a specific context.

Domain-specific Knowledge Graphs find application in a number of areas, major areas of use include:

- **Healthcare:** biomedical knowledge graphs integrate clinical data, medical ontologies and electronic health records to support activities such as drug discovery, diagnostic assistance and treatment personalization.
- **Education:** educational knowledge graphs map curricula, instructional concepts, learning materials and student profiles, enabling personalized recommendation systems and adaptive learning environments.
- **Finance:** in finance, KGs model entities such as companies, financial instruments, transactions and market relationships, facilitating predictive analytics, risk assessments and fraud detection.
- **ICT and Cybersecurity:** technology domain-specific knowledge graphs represent vulnerabilities, cyber threats, attack patterns, and countermeasures, improving the threat response capability and resilience of information systems.
- **Society and politics:** in these areas, KGs are used to analyze relationships between political actors, social movements, historical events and public opinion dynamics, supporting trend analysis and fact-checking activities.
- **Transportation and tourism:** knowledge graphs in these areas shape transportation networks, tourist attractions, itineraries and user preferences, improving the quality of services and personalizing the travel experience.

Examples of DSKGs built for real-world applications in different domains are listed in Table 2.3. One or more significant examples are given for each domain,

## 2. State of the art

---

accompanied by a brief description of their main features and a link to the reference document.

Domain	Knowledge Graph	Description
Healthcare	Hetionet [67]	Integrates data from 29 public databases to support drug discovery by linking genes, diseases, drugs, and proteins.
	RDBridge [68]	Built through text mining, links genes, drugs, and pathways for rare diseases.
	HeCiX-KG [69]	Integrates ClinicalTrials.gov and Hetionet to support clinical research and drug repurposing.
Education	KnowEdu [70]	System for automatic construction of educational KGs from learning materials.
	EDUKG [71]	Interdisciplinary knowledge graph for K-12 education, integrating various school concepts.
Finance	FinDKG [72]	Models financial events and time relationships between economic entities.
	FinCaKG-Onto [73]	Captures causal relationships between financial events by improving semantic representation.
	FinKG-JP [74]	Knowledge graph focused on Japanese companies with manually constructed ontologies.
ICT and Cybersecurity	Cyber-All-Intel [75]	Cyber threat extraction and representation system using KG.
	Unified Cyber Ontology (UCO) [76]	Standard ontology for representing cybersecurity data.
	UCKG [77]	Integrates several ontologies to represent cybersecurity information.
Society and Politics	PoliGraph [78]	Knowledge graph that maps political events and connections between institutional actors.
	Opinion-aware Knowledge Graph [79]	Integrates opinions and political entities extracted from texts to detect ideologies.
	PEG [80]	Framework that integrates multiple knowledge graphs to improve expert models in politics.
Transportation and Tourism	TravelRAG [81]	Personalized tourism information retrieval using multi-layer KG.
	Mobility Knowledge Graph (MKG) [82]	Models urban travel and trips using transportation smart card data.
	Tourism Knowledge Graph for Hainan [83]	Framework for building tourism KGs from regional web data.

Table 2.3: Examples of Domain-specific Knowledge Graphs (DSKGs) across different application domains.

## Advantages of Knowledge Graphs

Knowledge Graphs (KGs) are playing an increasingly prominent role in the representation, management, and enhancement of knowledge within complex information systems. Their adoption is growing rapidly in multiple contexts—from scientific research to healthcare, from recommender systems to semantic search engines—thanks to a number of distinctive advantages that make them often preferable to other data representation technologies.

### Natural modeling of interconnected knowledge

One of the main strengths of KGs is their ability to represent complex, interrelated information in an intuitive and scalable way. Unlike relational databases, which organize data in rigid tables, KGs adopt a graph structure that allows entities and relationships to be modeled flexibly, reflecting reality as a dynamic system of concepts and connections.

This structure facilitates semantic navigation of data, allowing users and systems to interpret not only the data themselves, but also the context in which they are located. This is particularly relevant in fields such as medicine or scientific research, where the meaning of a concept depends closely on its relationships to other concepts.

### Support for inference and automatic reasoning

Through integration with formal ontologies and the ability to define logical constraints, KGs enable automatic inference processes. This means that, from a set of explicit facts, the system is able to infer new implicit information, enriching the content of the graph.

For example, in a health KG, if it is known that *drug X treats disease Y* and that *patient Z has disease Y*, it is possible to infer that *drug X might be indicated for the treatment of patient Z*. Such semantic reasoning is difficult to achieve in systems without an explicit knowledge base.

### Semantic interoperability and integration of heterogeneous data

KGs, especially when built on standards such as RDF, OWL, and SPARQL, are designed to facilitate interoperability among heterogeneous systems. This feature enables consistent aggregation of information from different sources, structured according to different formats and models.

This capability is critical for data integration processes, in contexts such as smart cities, digital health or Industry 4.0, where information is derived from disparate sources: IoT sensors, legacy databases, documents, web or social media. KGs offer a unified framework that can effectively structure, connect and query this data.

### **Adaptability and scalability**

Knowledge Graphs are highly modular and adaptable. New entities, relationships or attributes can be introduced without altering the overall structure of the graph, unlike in relational databases, where structural changes may require extensive intervention. This makes them particularly suitable for evolving domains, such as biomedicine, emerging technologies or enterprise knowledge management.

Moreover, with advances in native graph databases (such as Neo4j or Amazon Neptune), KGs are able to scale efficiently, even when they contain millions or billions of nodes and relationships, while maintaining high performance in complex queries and traversing operations.

### **Accessibility through natural language and end-user support**

KG's integration with Natural Language Processing (NLP) and Large Language Models (LLM) technologies enables intelligent interfaces that support natural language queries. Users can thus ask open-ended questions without the need to know the graph structure or query language, receiving consistent and contextualized answers.

This feature makes KGs ideal for conversational systems, virtual assistants, decision support tools, and semantic search engines, enhancing the user experience and facilitating access even by non-technical users.

### **Traceability, transparency and explainability**

Unlike many *black-box* model-based technologies such as deep neural networks, Knowledge Graphs offer transparency and traceability. Each statement is represented explicitly (typically as a subject-predicate-object triple) and can be traced back to a source, allowing verification and explanation of the data.

This property is particularly important in highly regulated contexts, such as health care, finance, justice, or public administration, where it is essential to ensure auditability and justifiability of decisions and information provided by information systems.

## 2.5 GraphRAG

A particularly relevant use of *Knowledge Graphs* is seen in

**Retrieval-Augmented Generation (RAG)** [84, 85, 86] systems, which represent an important development in the field of Artificial Intelligence and natural language processing. *Large language models* (LLMs), such as GPT-4 [87], LLaMA [88] or Qwen2 [89], have demonstrated a remarkable capability in understanding and generating text, but are often limited by the lack of up-to-date, domain-specific or proprietary knowledge that is not included in the training data. This can lead to phenomena known as *hallucination* [90], or the generation of incorrect or invented content.

The RAG paradigm was created precisely to mitigate this problem by combining the effectiveness of LLMs with an information retrieval component from external sources. In this approach, the model dynamically queries a textual corpus to integrate factual and relevant knowledge into the generated response. However, traditional RAG methods operate on unstructured textual documents and often disregard the relationships between the data, limiting the quality of retrieval and the consistency of responses.

**Graph Retrieval-Augmented Generation (GraphRAG)** [91, 92, 93, 94] is an evolution of RAG that was created to overcome these limitations. Unlike traditional RAG, it does not rely on retrieval of simple documents, but exploits structured elements of a graph—such as nodes, relationships, or subgraphs—that represent knowledge in relational form. This allows the system to more accurately capture connections between concepts, improving the quality and completeness of the responses generated.

Figure 2.5 shows a comparison between the execution of a standalone LLM, a traditional RAG system and a GraphRAG system, in relation to the process of generating answers from a user query.

## 2. State of the art

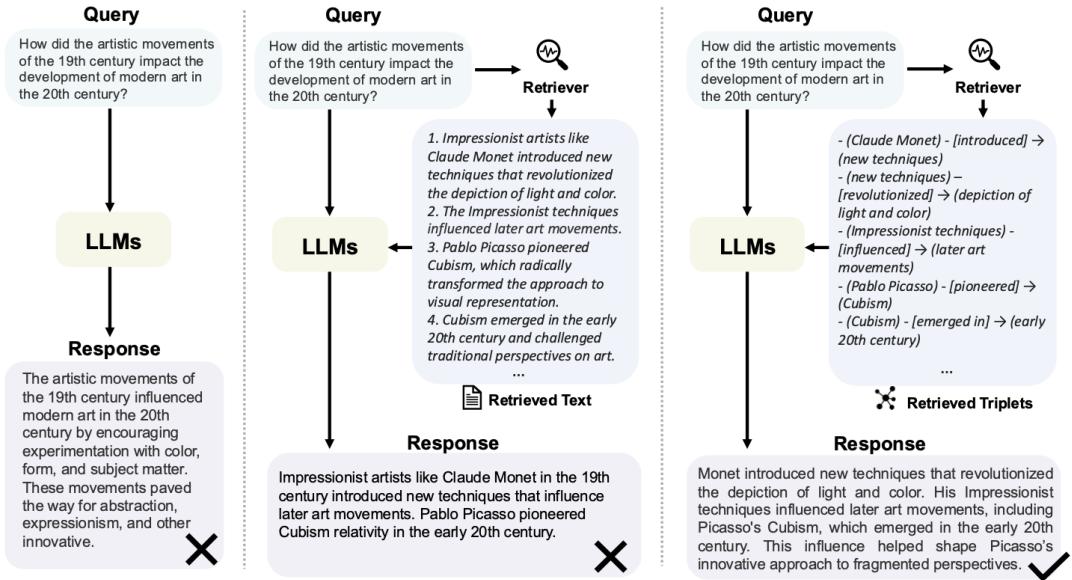


Figure 2.5: Comparison between Direct LLM, RAG, and GraphRAG

### 2.5.1 Comparison with related techniques

The GraphRAG approach represents a structural and conceptual extension of the Retrieval-Augmented Generation (RAG) paradigm, but it differs from other related techniques involving Large Language Models (LLMs) and structured knowledge bases.

#### RAG

Retrieval-Augmented Generation (RAG) is an established method for enriching language models with external knowledge, usually taken from indexed text corpora. However, its focus remains limited to semantic similarity between queries and segments of unstructured text, without explicitly considering the relationships between the retrieved information. The adoption of GraphRAG over the traditional RAG paradigm offers several significant benefits:

- **Increased accuracy of responses:** the study conducted by Data.world (2023) reported an average improvement of 54.2% in the accuracy of LLM-generated responses, with performance up to three times better than classic RAG [95].
- **Reduced tokens and increased efficiency:** Microsoft observed a reduction in token usage between 26% and 97%, thanks to the increased relevance of the retrieved information.

vance of the provided context, with direct impact on computational costs and scalability [91].

- **More complete and useful responses:** LinkedIn reported a 28.6% reduction in average ticket resolution time in its customer support system, due to the greater consistency and richness of information in responses generated with GraphRAG [96].
- **Transparency and traceability:** each element used in generation can be traced back to the entities and relationships in the graph, improving transparency and trust in the results.
- **Inference and semantic understanding:** knowledge graphs allow modeling and representation of the meaning of data, not just its textual form. This allows for inference of unstated relationships, integrating scattered information through semantic paths.
- **Control, governance, and security:** the use of knowledge graphs enables granular control of access permissions, monitoring of data provenance, and verification of semantic consistency – all crucial aspects in regulated domains such as healthcare and finance.

### LLMs on Graphs

A second line of research involves using LLMs to operate directly on graph data, often in combination with *Graph Neural Networks (GNN)* techniques. In this context, the goal is to improve the performance of LLM on typical graph learning tasks, such as node classification, arc prediction, or graph structure embedding [97].

GraphRAG, on the other hand, does not focus on graph analysis per se, but uses the graph as a structured source of knowledge from which to retrieve information elements to be integrated into the text generation process.

While they may share some representation techniques (e.g., the use of GNN to extract embedding from the graph), the two approaches clearly differ in their purposes and workflows.

### Knowledge Base Question Answering (KBQA)

Knowledge Base Question Answering (KBQA) [98] is a category of systems designed to answer questions by querying structured knowledge bases, such as RDF graphs or ontology databases. Specifically, retrieval-based KBQA (IR-based KBQA) employs query and matching strategies to identify and aggregate relevant information from the knowledge base.

GraphRAG shares with KBQA the retrieval from a knowledge base, but goes beyond this approach in several respects:

- KBQA is considered a subset of GraphRAG methods in that it focuses on a single task (answering queries) and is typically tied to an RDF-type knowledge base.
- GraphRAG is a broader paradigm, including applications in question answering, summarization, reasoning, recommendation, and beyond.
- KBQA often works with formal query languages (such as SPARQL) and requires parsing and transforming queries into structured queries. GraphRAG, on the other hand, retrieves more or less relevant portions of the graph and transforms them into textual context for language generation, with a more flexible approach that is less constrained by rigid formalisms.

#### 2.5.2 GraphRAG Architecture

The architecture of a GraphRAG system typically consists of three main phases, which define its operational pipeline: **Graph-Based Indexing**, **Graph-Guided Retrieval**, and **Graph-Enhanced Generation** [93]. An example of an architecture composed of these phases is shown in Figure 2.6.

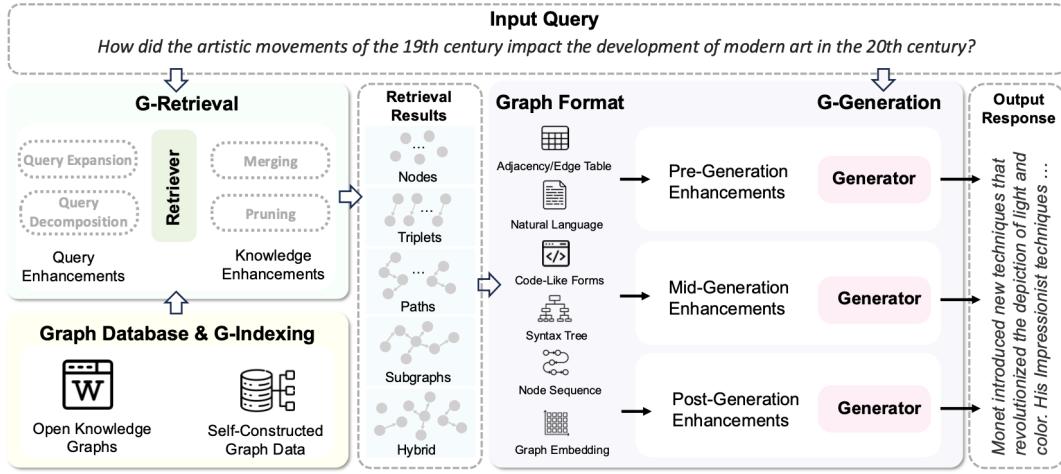


Figure 2.6: GraphRAG system architecture: indexing, retrieval, and generation.

### Graph-Based Indexing

The *Graph-Based Indexing* stage is the starting point of the GraphRAG architecture and is crucial to ensure effective and targeted information retrieval in the next stage.

At this stage, the system identifies and organizes a knowledge graph (KG) that forms the structured knowledge base. Such a graph can be imported from existing and established knowledge bases, such as DBpedia, Freebase or Wikidata, or built from scratch from textual or semi-structured data from documents, business logs, support tickets, conversations or proprietary sources. In the latter case, automatic knowledge extraction techniques (KGC - Knowledge Graph Construction), sometimes driven by large language models (LLM) or based on specific rules and patterns, must be adopted.

Indexing is not limited to the simple creation of nodes and arcs, but includes semantic mapping of entities, defining properties and structuring connections according to functional criteria for efficient graph navigation. In practice, indexes are constructed to speed up traversal operations, minimize latency during retrieval, and define information granularity, i.e., the size and detail of the elements that will be selected in the retrieval phase.

Within the indexing phase, an increasingly popular and effective strategy is vector indexing. This approach consists of transforming graph data into vector rep-

resentations within a continuous semantic space, allowing much faster retrieval and comparison operations to be performed than those based solely on discrete structures or symbolic patterns.

### **Graph-Guided Retrieval**

The *Graph-Guided Retrieval* phase represents the core of the GraphRAG approach. Starting with a textual query formulated by the user, the system must be able to identify in the graph those portions that contain knowledge that is relevant and useful for generating the answer.

The main difference from traditional Retrieval-Augmented Generation lies in the type of data involved and the logic of selection: whereas classical RAG retrieves text passages on the basis of semantic similarity between embedding vectors, GraphRAG actively uses the structure of the graph, the relationships between entities, and the semantic properties of connections.

Depending on the system configuration and query complexity, the retrieval process can return different types of elements: from single nodes or entities to RDF triplets (subject-relation-object), from semantic paths connecting related entities, to entire subgraphs representing a larger conceptual context. Retrieval can take place in a *one-shot* mode or be implemented in iterative or multi-stage schemes, in which the context is progressively refined.

In some advanced systems, the graph is segmented into semantic communities (graph communities), and selection occurs at both local scale (retrieval of specific details) and global scale (extraction of structured summaries), allowing the system to balance granularity and information coverage. In addition, query expansion techniques or filters based on ontologies, metadata, or semantic similarities between nodes can be applied to improve the relevance and accuracy of retrieval.

### **Graph-Enhanced Generation**

The last stage, called *Graph-Enhanced Generation*, consists of integrating the information retrieved from the graph within the language generation process performed by an LLM model.

Since these models are optimized to operate on sequential text, it is necessary to transform the graph data into a suitable representation. This conversion can take many forms: from naturalistic textual descriptions that verbalize nodes and relationships, to arc tables, tree structures, formal graph languages (e.g., GML, GraphML), or even vector embeddings learned through Graph Neural Network (GNN) type models.

Once a compatible format is obtained, this information is concatenated to the initial query in the form of context and passed to the text generator. Depending on the strategy adopted, semantic enrichment can take place before, during, or after generation, with targeted interventions on the prompt (pre-generation), step-by-step processing (mid-generation), or post-processing of the response produced (post-generation).

### 2.5.3 Application domains and case studies

The GraphRAG framework is rapidly emerging as a versatile solution for improving the performance of LLMs in a variety of application scenarios.

#### Application domains

GraphRAG has found application in several vertical domains where the presence of structured knowledge is particularly relevant:

- **E-commerce:** used to improve interactions in customer service systems, answering product questions and offering personalized suggestions. A case study demonstrates the use of a support ticket-generated graph to power an intelligent QA system [96].
- **Biomedicine:** plays a central role in assisted diagnosis, drug discovery, and querying biomedical knowledge graphs such as CMeKG and CPubMed-KG. GraphRAG enables more transparent and accurate integration between clinical queries and scientific knowledge [99].
- **Academic and bibliographic:** used to explore networks of publications, authors, and institutions, enabling intelligent retrieval of related articles, prediction of collaborations, and analysis of scientific trends.

- **Legal:** facilitates case law research, contract analysis, and regulatory browsing through graphs representing rules, cases, and citations.
- **Literature and digital libraries:** GraphRAG enables querying knowledge graphs built from publishing metadata to find similar books, track literary series, and connect authors to genres or themes.
- **Finance and Intelligence:** used to detect fraud, analyze transaction networks, or generate strategic reports based on semantic connections between events, individuals, and entities.

### Case Studies

An example implementation is **KRAGEN** [100], a GraphRAG-based system that implements the *Graph-of-Thoughts (GoT)* technique to support multistep reasoning on knowledge graphs.

In the Alzheimer’s case study, KRAGEN uses a knowledge graph called AlzKB, containing structured data on genes, drugs, and diseases. The graph, hosted on Neo4j and extended with vector embeddings via Weaviate, is queried by KRAGEN to answer questions such as, ”*What drugs bind to APOE and PLA2U genes?*”

KRAGEN breaks down the question into semantic subcomponents (e.g., *drugs that bind to APOE*), performs retrieval on each subquery, and synthesizes the results into a coherent answer, displaying each step of reasoning in the user interface. This ensures explainability, transparency, and controllability, addressing one of the historical limitations of LLMs.

Experimental results show that KRAGEN outperforms standard QA approaches and generic LLM models such as BioGPT, especially in multi-hop tasks. The framework is already seen as a reference model for clinical and biomedical application of LLM+KG systems, with potential extension toward the use of real patient data.

# Chapter 3

## Original contribution to the solution of the problem

This chapter outlines the design and implementation contribution developed as part of the thesis, which focused on the creation of an intelligent system that can support people with cognitive fragility in managing time and daily activities.

The system is designed to favor natural and accessible interaction based on everyday language. The user can communicate his or her to-do list, current projects and upcoming events, as well as ask questions about what has happened or has yet to happen.

The proposed solution consists of **three main functionalities**, each implemented as a stand-alone module but seamlessly integrated into the overall system:

1. **Ontology Parsing:** this component interprets the semantics of the reference domain from an ontology in OWL/Turtle format. Parsing transforms the ontology into a structured representation (SchemaEntity) that provides the system with the foundation for understanding and organizing knowledge, ensuring consistency in both processing input and identifying meaningful connections.
2. **KG Construction:** this pipeline processes user-expressed input in natural language, transforming it into structured knowledge. Through a series of specialized modules, the system identifies entities, relationships, and contextual information from the texts, organizing them into a knowledge graph stored in Neo4j. In this way, a coherent representation of the activities and

### *3. Original contribution to the solution of the problem*

---

events that characterize the user experience is progressively constructed.

3. **GraphRAG:** this component allows the knowledge base to be queried using questions formulated in natural language. From the user's query, the system triggers a procedure that retrieves the relevant context using hybrid search techniques and graph navigation. The retrieved information is then processed by a language model to generate appropriate textual responses based on the stored knowledge and calibrated to the specific content of the question.

These three components work in synergy to support knowledge creation, organization, and retrieval, giving the system the ability to dynamically structure and interpret the information shared by the user.

## **3.1 Definition of the proposed methodology**

To meet the needs of people with Mild Cognitive Impairment (MCI), who frequently experience difficulties in managing daily activities and planning their routines, the implementation of a digital activity diary, i.e., an intelligent time management support system, is proposed. The system is designed to assist users in recording, organizing and retrieving information related to their daily routines in a simple, natural and accessible way.

Through natural language-based interaction, the system allows the user to document activities already performed, plan future ones, indicate upcoming events, describe ongoing projects, and provide other relevant personal information. Unlike classic calendars or reminder systems, which rely on rigid interfaces and explicit commands, the proposed digital diary allows for more intuitive and natural interaction, eliminating the use of pre-filled buttons, menus, or forms, and making the experience closer to a conversation.

The user can query the system by asking questions in natural language in order to retrieve previously stored information. Questions can be about what they have done, what they have yet to do, or specific aspects related to their personal organization. The system is able to reason about the stored data, combine information, identify relationships and provide consistent, contextual and personalized answers.

### *3. Original contribution to the solution of the problem*

---

The proposed methodology is based on three key principles, each of which contributes to the definition of a system that is coherent, adaptable, and oriented toward concrete user support:

1. **Knowledge structuring through semantic support:** knowledge provided by the user is not recorded as simple text, but is organized according to a coherent conceptual schema built from a domain ontology. This makes it possible to attribute formal meaning to data, ensure internal consistency, enable integration with other systems, and enable automatic inference processes.
2. **Natural and intuitive interaction:** the entire system is designed to break down technological barriers, facilitating accessibility by people who are frail or have limited digital skills. The use of natural language as an input and output mode allows for smooth, personal, low-cognitive-load interaction.
3. **A personal and dynamic knowledge base:** each user interacts with his or her own knowledge base, which grows and evolves over time. With each new piece of information or question, the system updates its internal representation of the user's reality, building an increasingly complete information profile capable of providing relevant and tailored responses.

As part of this thesis, a prototype user interface is planned, not aimed at the end user, but designed to demonstrate the operation of the system and verify its behavior. This interface allows for the insertion of new textual content and the display of responses generated by the system as a result of requests or queries.

The general operation consists of three main phases, as illustrated in Figure 3.1:

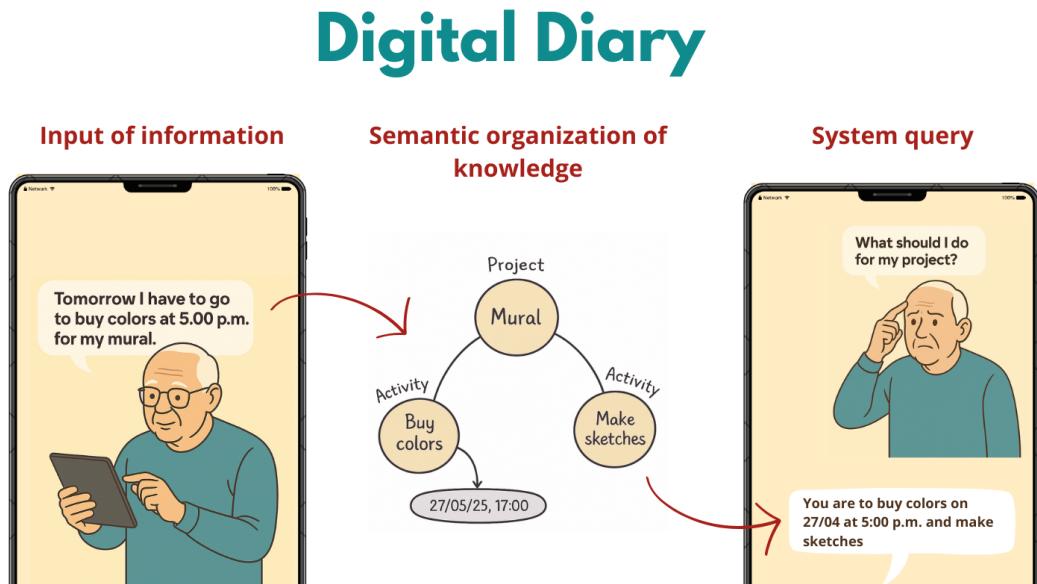


Figure 3.1: Digital Diary: input, semantic processing, and querying

- Information input:** the user, via the interface, provides a set of data expressed in natural language regarding his or her daily routine. Inputs may include scheduled activities, projects to follow, appointments, reminders or personal reflections.
- Semantic organization of knowledge:** the system analyzes the input received, extracts meaningful entities and relationships, and inserts them into a structured knowledge base. This process is guided by a semantic reference model, the design and implementation of which are an integral part of this thesis work. The model makes it possible to formally represent the reality described by the user, providing a logical structure capable of supporting automatic interpretation and reasoning about the data.
- System querying:** the user can, at any time, enter new information or ask questions of the system. The latter is able to query the knowledge base, retrieve relevant data, compose a coherent context, and generate a comprehensible textual response, which is finally displayed via the interface.

The functionality outlined in this section will be discussed in depth in the next section, where the design aspects of the system and the individual components that make up its overall architecture will be examined.

## 3.2 Design of the proposed system

This section discusses the design of the system and its constituent components, analyzing the main implementation choices made and the considerations that motivated them. The objective is to illustrate how the functionalities presented in the previous section were implemented, providing a comprehensive view of the underlying architecture.

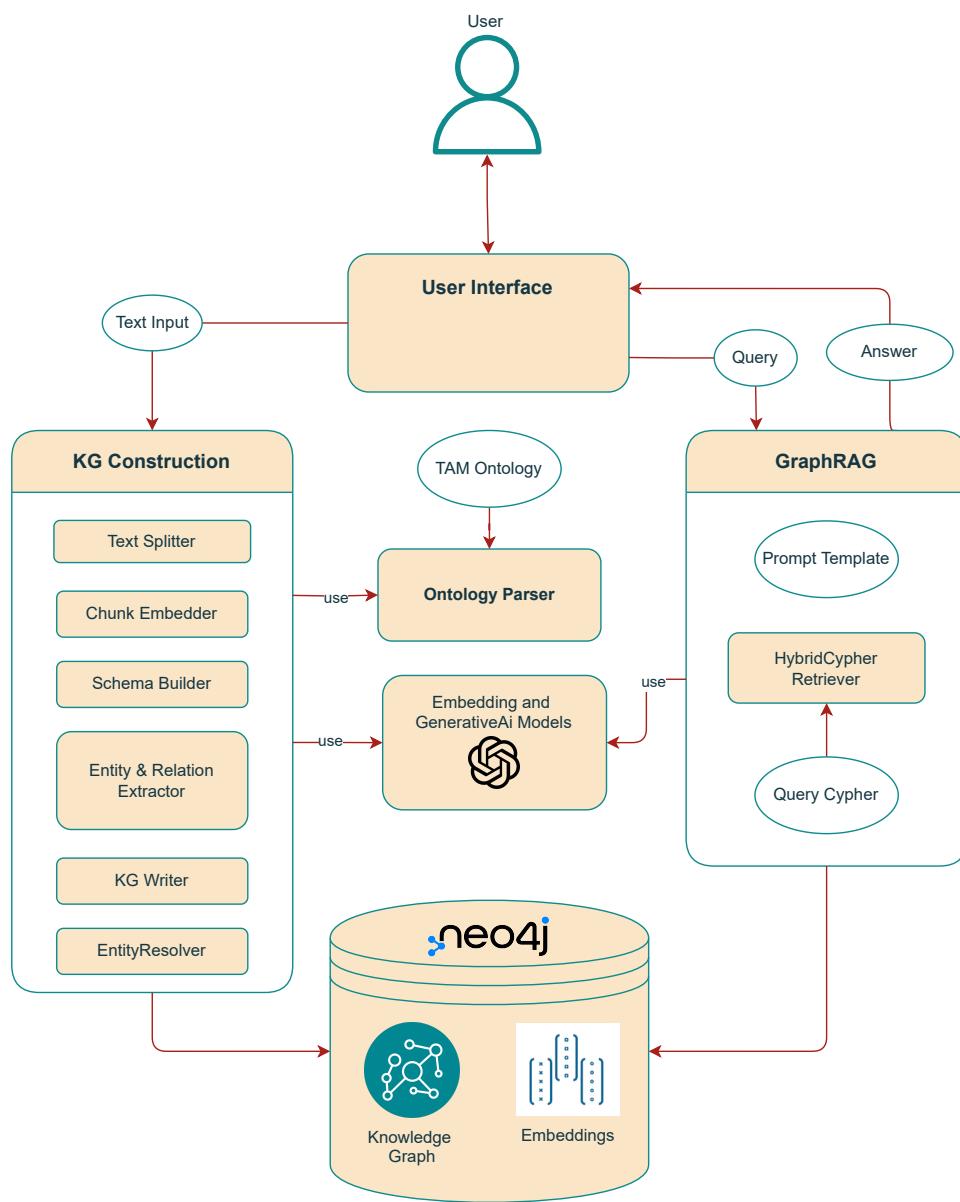


Figure 3.2: Logical architecture of the proposed system

### *3. Original contribution to the solution of the problem*

---

To implement the system, a **modular architecture** represented in the diagram in Figure 3.2 was developed. This architecture consists of the following basic elements:

#### **1. User interface**

An essential interface, developed as a demonstration prototype, that allows interaction with the system. Although it is not designed for the end user with MCI, it plays a crucial role in the scope of this thesis, allowing the implemented functionality to be verified and presented. The interface supports the input of natural language content and the display of responses processed by the system.

#### **2. TAM Ontology**

Underlying the semantic organization of the data is the *TAM Ontology* (Time Assistive Management Ontology), a conceptual model designed to represent entities, properties and relationships relevant to the domain of time management and daily activities. This ontology guides the knowledge base construction process by providing a consistent and formally defined logical structure.

#### **3. Ontology Parser**

The *Ontology Parser* module is responsible for translating the TAM ontology, defined in OWL/Turtle format, into a structured representation in Python. This schema is used by the *KG Construction* component to guide the consistent insertion of information into the knowledge graph, ensuring compliance with the semantic structure defined by the ontology.

#### **4. KG Construction**

The component dedicated to processing textual input provided by the user. This module analyzes the linguistic content, identifies meaningful entities and relationships, and inserts them into the knowledge base. The data is structured according to the semantic schema derived from the ontology, ensuring consistent and meaningful organization.

#### **5. GraphRAG**

This module implements a Graph Retrieval-Augmented Generation approach, allowing the user to query the system through queries formulated in natural language. Starting with a text query, the module queries the

### *3. Original contribution to the solution of the problem*

---

knowledge base, retrieves relevant information, and generates a contextualized answer that is presented to the user.

#### **6. Knowledge Graph**

The user's knowledge is represented within a *knowledge graph* (*Knowledge Graph*), implemented using the *Neo4j* graph database. This structure stores entities, relationships, and associated metadata, maintaining a consistent and queryable representation of the user's profile.

#### **7. Embedding and GenerativeAI Models**

For linguistic and semantic processing of content, the system makes use of models provided by OpenAI. The models are used to compute vector representations (embedding), extract entities and relationships from texts, and generate responses from retrieved context. The choice of specific models and associated motivations will be detailed in a dedicated section.

#### **3.2.1 User Interface**

To enable experimentation and visualization of the system's functionality, a prototype web interface was developed using the Streamlit framework, as shown in Figure 3.3.

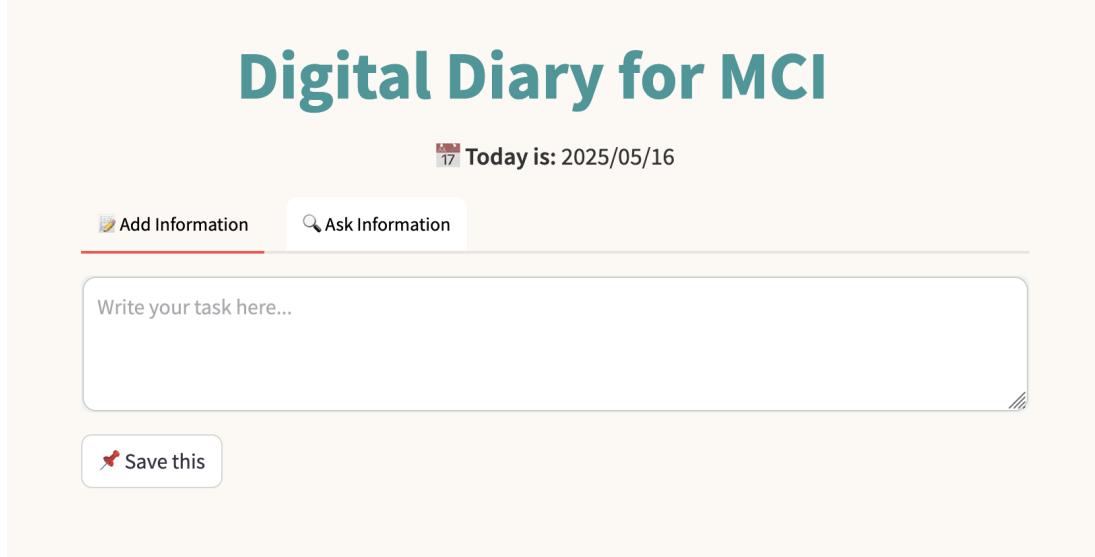


Figure 3.3: User Interface Prototype

The application is divided into two main sections, accessible through interface tabs: **Add Information** and **Ask Information**.

### *3. Original contribution to the solution of the problem*

---

#### Add Information

This section allows the user to enter textual input describing activities, projects, events, or routines to be stored in the knowledge graph. The interaction takes place through a text area and a confirmation button.

When the user submits the input, the text is passed to the *KG Construction* component, if the process is successful, the interface displays a confirmation message informing the user that the information has been successfully registered.



Figure 3.4: User Interface: Add Information

#### Ask Information

This section allows the user to query the system to retrieve stored knowledge. After typing the question and clicking the button, the input is forwarded to the *GraphRAG* component. The resulting response is then displayed directly in the interface.

### 3. Original contribution to the solution of the problem

---

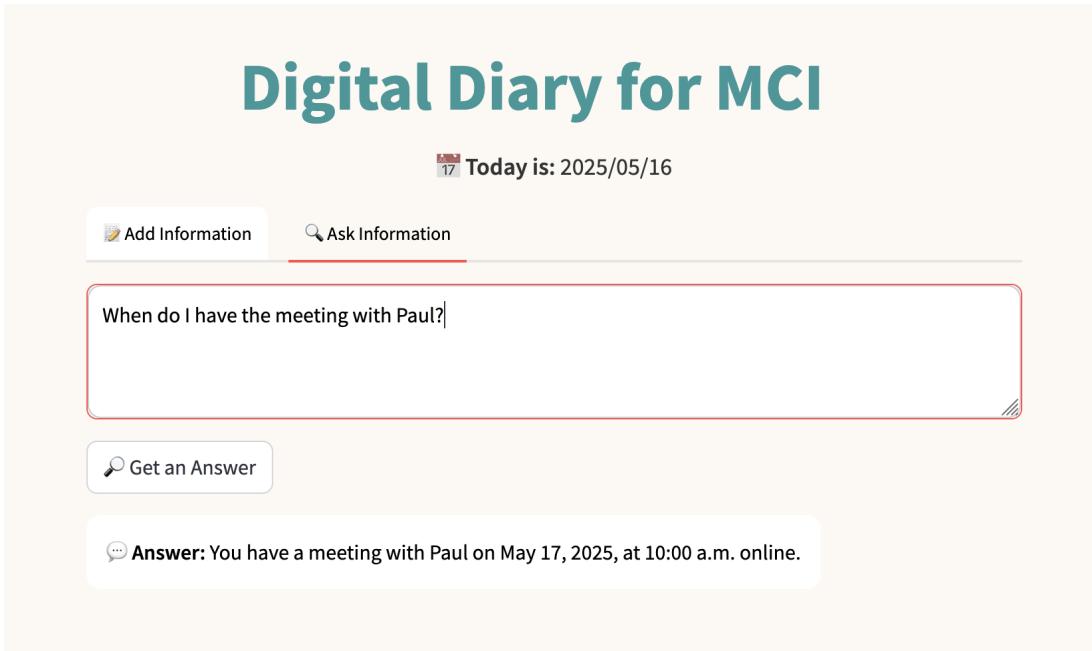


Figure 3.5: User Interface: Ask Information

#### 3.2.2 TAM Ontology

In the proposed system, the **TAMOntology** (Time Assistive Management Ontology) ontology provides the semantic foundation on which the structured representation of information related to time management, activities, and routines of people with *Mild Cognitive Impairment* (MCI) is based. The ontology is responsible for formally defining the key concepts of the application domain, including activities, events, projects, routines, places and people, specifying for each the descriptive properties and semantic relationships that determine their behavior and connections.

In this context, ontology plays a crucial role not only in the representation of knowledge, but especially in its automatic organization, validation and interpretation. In fact, the semantic support offered by TAMOntology constitutes an original and distinctive contribution, as it allows information within the knowledge graph to be structured according to an explicit and shared logic, enabling mechanisms of automatic inference and reasoning.

Ontology is actively used during the construction and updating phase of the knowledge graph, as a reference for the extraction and typing of entities and rela-

### *3. Original contribution to the solution of the problem*

---

tionships obtained through linguistic models. Its adoption, although it does not guarantee absolute accuracy, helps to improve the quality of the data and reduce the introduction of erroneous or redundant elements into the knowledge base.

Another advantage is the ability to perform semantic inference on the data. In the system described, inference is limited to a simple form based solely on hierarchies between classes. Specifically, subclasses automatically inherit all properties and relationships defined in the parent class. This allows the system to enrich the knowledge graph with new information, even if not directly specified.

TAMOntology features a modular and dynamic architecture that allows for modifications and extensions without requiring recompilation of the system. Memory loading occurs with each insertion operation, ensuring that changes to the ontology structure are immediately operational. This design feature offers advantages in terms of adaptability to new functional requirements, emerging use cases, or assistive scenarios not initially anticipated, thus maximizing the flexibility of the system.

## **Ontology Design**

The design of the *TAMOntology* was conducted following the principles and methodological steps suggested in a Protegè guide [101]. This approach involves a series of steps to guide the construction of a solid, coherent, and practice-oriented ontology.

In a first step, a domain analysis was conducted, asking three basic questions:

- **What concepts should be represented?** The ontology was to model elements such as activities, events, appointments, routines, projects, places, people, medicine, and visits.
- **For what purpose will it be used?** The ontology was supposed to support the recording and organization of the user's routine, allowing the user to tell what he or she has done and plan what he or she needs to do.
- **What queries should the system be able to answer?** Some of the most significant queries include "What do I need to do today?", "What did

### *3. Original contribution to the solution of the problem*

---

I do yesterday?", "What do I have left to complete for project X?", "What medication do I need to take today?"

Based on these needs, an exploration of existing ontologies was undertaken. Several semantic models were analyzed, including Basic Formal Ontology, Event Ontology, Time Ontology in OWL, LODE, Activity Ontology and others. However, none of these covered the domain in its entirety: some focused only on time, others on events or activities, but none offered an integrated view of personal routines composed of events, activities, and projects.

Given also the contained and well-bounded nature of the domain, it was not advantageous to reuse portions of multiple heterogeneous ontologies. It was therefore preferred to design an ad hoc ontology, tailored to the system's functionality and designed to effectively address the specific needs of users with MCI. This does not exclude, in any case, the possibility of integrating other ontologies in the future, should the domain expand.

Having finished the analysis phase of existing ontologies, we moved on to the enumeration of the main concepts to be represented. Six core classes were defined: **Activity**, **Event**, **RoutineActivity**, **Person**, **Place** and **Project**. These classes were organized hierarchically around a common superclass, **TAMThing**, designed to facilitate extensibility. Next, descriptive and relational properties were associated with each class:

- **DatatypeProperty**, used to represent instance-specific attributes (such as name, description, date, time, priority, status);
- **ObjectProperty**, used to express semantic relationships between entities (such as "an activity takes place at a location" or "a person works on a project").

All properties have been given domain and range, accompanied by descriptive comments, and designed to be easily recognizable even in NLP model-driven automatic extraction processes.

## **Conceptual Structure**

The conceptual structure of the TAM ontology was designed to represent information relevant to routine management and task planning in a simple but

### 3. Original contribution to the solution of the problem

---

comprehensive way. The conceptual schema has been formalized according to the OWL (Web Ontology Language) paradigm and visualized in UML form, as illustrated in Figure 3.6.

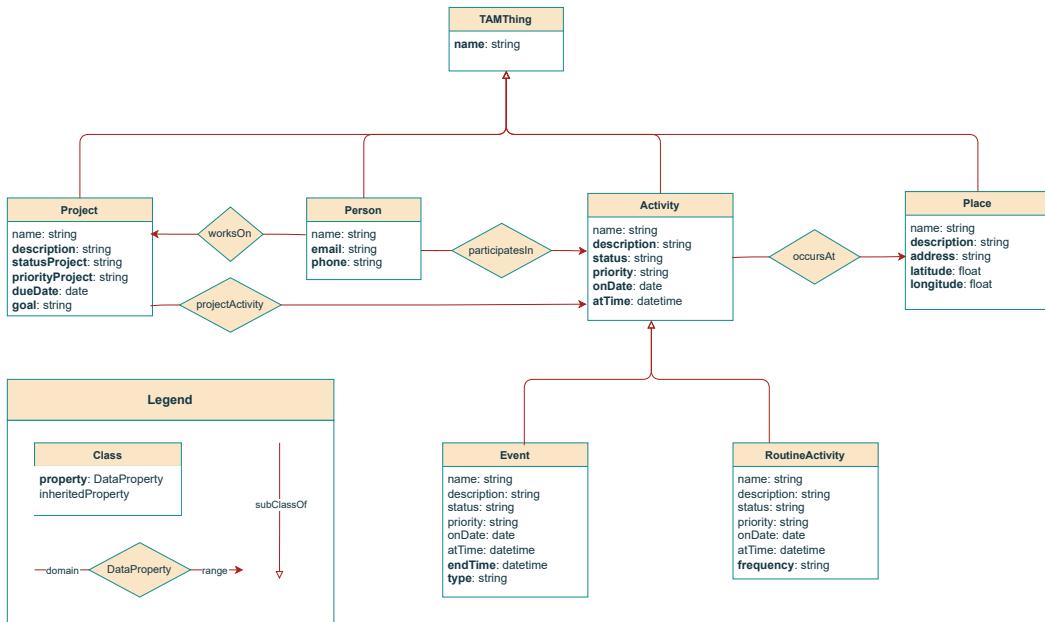


Figure 3.6: Conceptual structure of TAMOntology in UML notation

**Classes** The model is based on a superclass, **TAMThing**, from which all conceptual entities in the domain are derived. From this common root, the ontology defines six main classes: **Activity**, **Event**, **RoutineActivity**, **Person**, **Place**, and **Project**.

### 3. Original contribution to the solution of the problem

---

Class	Description
TAMThing	General superclass from which all entities are derived. Facilitates future extension of the ontology.
Activity	Action to be performed or performed by the user. Can be isolated or part of a project or routine.
Event	Scheduled activity with a specific start, end, and type. It is a subclass of <b>Activity</b> .
RoutineActivity	Recurring activity over time (e.g., taking a medicine every day). It is a subclass of <b>Activity</b> .
Project	Organized set of activities aimed at achieving a goal (e.g., writing a thesis).
Person	Main user or subjects involved in activities and projects. Extends the <b>foaf:Person</b> class.
Place	Physical or virtual place where an activity or event takes place.

Table 3.1: Main conceptual classes defined in TAMOntology

**Properties** Each class is associated with a set of attributes, defined as Date-typeProperty, that describe its characteristics. The main properties by class are listed below:

Class	Main Properties
TAMThing	name: string
Activity	description: string, status: string, priority: string, onDate: date, atTime: datetime
Event	inherits from Activity, plus endTime: datetime, type: string
RoutineActivity	inherits from Activity, plus frequency: string
Project	description: string, statusProject: string, priorityProject: string, dueDate: date, goal: string
Person	email: string, phone: string
Place	description: string, address: string, latitude: float, longitude: float

Table 3.2: Datatype properties associated with each main class in TAMOntology

**Relationship** Semantic relationships between entities are modeled through ObjectProperty. They represent links between instances of different classes and enable navigation and reasoning within the knowledge graph.

### 3. Original contribution to the solution of the problem

Relationship	Domain	Range	Description
occursAt	Activity	Place	Indicates where an activity or event takes place.
participatesIn	Person	Activity	Links a person to an activity or event in which they participate.
projectActivity	Project	Activity	Links a project to its component activities.
worksOn	Person	Project	Indicates that a person works on a project.

Table 3.3: Semantic relationships (ObjectProperty) defined in TAMOntology

All relationships and properties are designed to support semantic inheritance: for example, a **RoutineActivity** automatically inherits the properties defined for the **Activity** class, and instances of **Event** can be linked to a **Place** through the occursAt relationship without the need for duplication.

### Resulting Ontology

The current version of the ontology is labeled as 1.0 and is identified by IRI: <http://www.semanticweb.org/alexandra.tarantino/tam> was developed using the semantic modeling tool Protégé, one of the most popular tools for designing and validating OWL ontologies. At the end of the modeling phase, the ontology was exported in the Turtle(.ttl) format, chosen for its readability and compatibility with automatic processing tools.

It is annotated with descriptive metadata such as the author's name, subject scope, and a textual description of the goals and application domain, as visible in Figure 3.7.

The screenshot shows the Protégé interface with the following details:

- Ontology header:**
  - Ontology IRI: <http://www.semanticweb.org/alexandra.tarantino/tam>
  - Ontology Version IRI: [e.g. http://www.semanticweb.org/alexandra.tarantino/tam/1.0.0](http://www.semanticweb.org/alexandra.tarantino/tam/1.0.0)
- Annotations:**
  - rdfs:label [language: en]: TAM: An ontology for Time Assistive Management
  - rdfs:comment [language: en]: TAM (Time Assistive Management) is an ontology designed to support the structured representation of time-based activities, scheduled events, and assistive interactions for users requiring time management assistance. It models key concepts such as activity, place, event, project and their relationship.
  - dcterms:creator: Alexandra Tarantino
  - dcterms:hasVersion [type: xsd:decimal]: 1.0
  - dcterms:subject [language: en]: TAM: An ontology for Time Assistive Management
  - dcterms:subject [language: en]: Time management, scheduling, assistive technologies, semantic web, knowledge representation, activity
- Metrics:**

Metrics	Count
Axiom	157
Logical axiom count	57
Declaration axioms count	37
Class count	8
Object property count	6
Data property count	18
Individual count	0
Annotation Property count	7
- Class axioms:**

Class Axioms	Count
SubClassOf	6
EquivalentClasses	1
DisjointClasses	0
GCI count	0
Hidden GCI Count	1
- Object property axioms:**

Object Property Axioms	Count
SubObjectPropertyOf	0
EquivalentObjectProperties	0
InverseObjectProperties	0
DisjointObjectProperties	0
FunctionalObjectProperty	0

Figure 3.7: Ontology metadata and definition preview in Protégé

At the structural level, the ontology consists of:

- **8** main conceptual classes (Activity, Event, RoutineActivity, Project, Place, Person, TAMThing, foaf:Person);
- **6** object properties (ObjectProperty);
- **18** datatype properties (DatatypeProperty);
- **7** annotation properties;
- **6** subclass axioms (SubClassOf);
- **1** equivalent class (equivalentClass);
- **0** individuals.

Altogether, the ontology contains **157 axioms** and **57 logical axioms**, including declarations of classes, properties, domains, ranges, and relationships between entities.

This structure, while contained, is designed to provide a solid, extensible and immediately usable semantic foundation to guide the populating and navigation of the knowledge graph. The adoption of a standard such as OWL also ensures its portability, interoperability, and the possibility of future integration with external vocabularies or Linked Data-based systems.

### 3.2.3 Ontology Parser

In the proposed system, a key component is the Ontology Parser, responsible for the automatic transformation of the *TAMOntology* ontology, written in RD-F/Turtle format, into a structured representation compatible with the entity extraction component in the *KG Construction* module. This step is essential for guiding the recognition and extraction of relevant entities and relationships within the user's textual inputs, ensuring semantic consistency with the reference ontological model.

The parser was implemented in Python using the `rdflib` package, and is responsible for extracting from the `.ttl` files all elements relevant to semantic modeling: classes, properties (both datatype and object), and relationships between entities.

The extraction logic is articulated in several stages:

### *3. Original contribution to the solution of the problem*

---

1. **Extraction of classes (entities):** the parser identifies all RDF subjects that represent entities of type `owl:Class`, extracting their `label` and `comment`. Each class is transformed into a `SchemaEntity` object, which constitutes the basis of the semantic structure.
2. **Hierarchy management:** inheritance links are identified via the `rdfs:subClassOf` property, building a map of subclasses that enables the propagation of properties to child classes.
3. **Extraction of simple (datatype) properties:** the parser detects all `owl:DatatypeProperty` elements and extracts, for each, the `domain`, `label`, `comment`, and `type` (e.g., string, int, date). The RDF type is converted into a standardized format through a mapping that aligns XSD types with the types used in Neo4j.
4. **Extension of properties to subclasses:** if a property is defined for a class, it is automatically inherited by all its subclasses, respecting the ontology semantics.
5. **Extraction of relationships (object properties):** for each `owl:ObjectProperty`, the parser extracts the `domain`, `range`, `name`, and `description`. A potential semantic schema of the relationships is then built, combining all class pairs that can be linked by the same relation.

The final result of the transformation is a Python structure consisting of:

- a list of entities (`SchemaEntity`), each with its properties (`SchemaProperty`);
- a list of relationships (`SchemaRelation`);
- a `potential_schema`, that is, an abstract representation of the possible relationships between entities.

#### **3.2.4 KG Construction**

The KG Construction module is responsible for transforming textual input provided by the user into structured knowledge, which is then stored within the knowledge graph. It is a central element of the system, as it enables the translation of unstructured data into a formal format, consistent with the application domain and queryable through semantic queries.

The entire pipeline was built in Python using the `neo4j-graphag` package, which provides a set of modular, customizable components for building knowledge graphs from natural language text. Each component of the pipeline is separately configurable and can be combined with others to suit different contexts and uses.

The knowledge construction process is depicted in the diagram in Figure 3.8, where the individual steps, the modules involved, and their respective inputs and outputs are illustrated. The pipeline is organized according to a sequence of transformations, each of which performs a specific operation on the data.

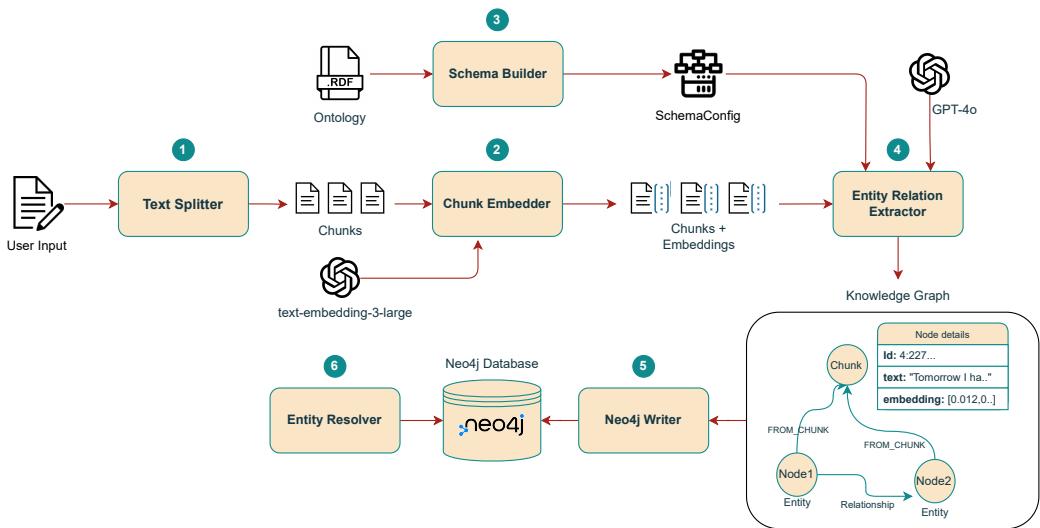


Figure 3.8: Knowledge Graph Construction Pipeline

## 1 - Text Splitter

The first component of the pipeline is the Text Splitter, whose job is to split the input text into smaller, more manageable text units called *chunks*. This process is particularly useful for ensuring the effectiveness of subsequent embedding and semantic extraction steps, especially when working with language models that have limitations in the amount of text that can be processed in a single request.

The input to this module is a single natural language text provided by the user, usually a sentence or short description (e.g., *"Tomorrow I have to buy colors"*). However, the infrastructure is designed to allow the user the freedom to enter longer, more articulate text as well, such as entire descriptions of days or weekly

### *3. Original contribution to the solution of the problem*

---

plans. The output of the module is a sequence of text portions, each representing a segment of the original text divided according to the set configuration.

In this specific case, the module adopted for chunk splitting is the `FixedSizeSplitter` from the `neo4j-graphrag` package, which operates by dividing the text into fixed-size chunks. The two configurable parameters are:

- `chunk_size`: indicates the maximum length (in characters) of each chunk.  
In the current configuration, it is set to **4000 characters**;
- `chunk_overlap`: defines the number of characters to overlap between one chunk and the next, so as to maintain semantic continuity between segments. In this case, it is set to **200 characters**.

## **2 - Chunk Embedder**

Once the textual chunks are obtained, each one is processed by the `TextChunkEmbedder` component, provided by the `neo4j-graphrag` package. This module is responsible for transforming each text segment into a numerical vector representation, called an *embedding*.

Embeddings are dense vectors that capture the semantic features of the textual content, allowing the system to compare and correlate information not only on the basis of linguistic form, but also deep meaning.

An embedding model deals with mapping texts into a continuous vector space in which similar concepts are found to be close to each other. This makes it possible to assess semantic similarity between sentences even if they are expressed with different words. The output of the module is a list of embeddings, each associated with a chunk, which will be used by the GraphRAG component of the system to enable vector search within the graph.

An embedding model provided by OpenAI, specifically the `text-embedding-3-large` model, was chosen for the implementation of this step. This is one of the latest and best performing models available in the OpenAI platform, designed to provide high-precision semantic representations optimized for retrieval, clustering, reasoning, and generation tasks.

### 3 - Schema Builder

Before entities and relationships can be extracted from the obtained chunks, it is necessary to define the semantic schema that will guide the extraction operation. This task is entrusted to the `SchemaBuilder` component of the `neo4j_graphrag` package, which is executed by passing as input the schema definition generated by the *Ontology Parser* module.

The Ontology Parser translates the TAM ontology, modeled in OWL/Turtle format, into a structured representation in Python that describes the entities, properties, and relationships of the application domain. This representation is used as the basis for the `SchemaBuilder`, which formally constructs the schema to be provided to the language model to guide semantic extraction.

The schema consists of three main classes provided by the `neo4j_graphrag` package:

- `SchemaEntity`, which represents an entity type and describes its relevant properties;
- `SchemaProperty`, which defines a single property associated with an entity, specifying the name and type of the expected value;
- `SchemaRelation`, which describes a semantic relationship between two entities, possibly enriched with additional properties.

The schema thus constructed ensures that, during the extraction phase, the system is able to recognize in the text only the structures predicted by the ontology, ignoring extraneous or out-of-domain elements. This mechanism improves the semantic accuracy of the system and reduces the risk of errors when entering information into the graph.

### 4 - Entity and Relation Extractor

Once textual chunks have been enriched with vector representations and associated with a semantic reference schema, they are processed by the `LLMEntityRelationExtractor` component, provided by the `neo4j_graphrag` package.

### *3. Original contribution to the solution of the problem*

---

This module uses a generative language model to parse each segment of text, with the goal of identifying relevant semantic entities and relationships in a way that conforms to the schema provided as input.

The output of the extraction has two levels:

1. **Lexical graph** It is the structured representation of the original text and its blocks. It includes:

- a **Document** node, representing the textual input provided by the user, with a property indicating its path or identifier;
- a set of **Chunk** nodes, each of which contains a **text** property and associated **embedding** property;
- **NEXT\_CHUNK** relationships between chunks, which preserve the sequential order of the text;
- **FROM\_DOCUMENT** relationships that link each chunk to the document node from which it was generated.

This graph allows persistent and structured storage of user input, enabling advanced features such as **vector search** and navigation between consecutive blocks of text.

2. **Domain graph** In addition to the text structure, the module builds the domain graph, which is the conceptual representation of the knowledge extracted from the text.

The **Entity** nodes, mapped to the classes of the TAM ontology, and the semantic relationships connecting them are generated, subject to domain and range constraints, thus building a structured and consistent representation of the information given as input.

The two graphs are linked together through **FROM\_CHUNK** relationships, which associate each extracted entity with the chunk from which it was generated. This linkage maintains direct traceability between the original text entered by the user and the derived structured information, allowing both contextual retrieval and explainability of the responses produced by the system.

This stage represents a critical moment in the pipeline, as it determines the

### *3. Original contribution to the solution of the problem*

---

quality and consistency of the information feeding the Knowledge Graph and affects the system's performance in answering questions.

This phase represents a critical step in the pipeline because it directly affects the quality and consistency of the data in the Knowledge Graph and, consequently, the performance of the system in generating correct and relevant answers.

#### **5 - Neo4j Writer**

Once the extraction is complete, the data is entered into the Neo4j database via the `Neo4jWriter` component, provided by the `neo4j_graphrag` package. This module is responsible for writing to the graph not only the semantic entities and relations that make up the domain graph, but also the nodes and relations of the lexical graph, which represent the text structure and associated embeddings.

The module uses the official Neo4j driver to establish a direct connection and manage data entry.

To optimize performance, you can adjust the `batch_size` parameter, which defines the maximum number of nodes or relations written per batch of operations. The default value is 1000, but it can be customized according to the volume of data and resources available in the system.

#### **6 - SinglePropertyExactMatch Resolver**

After the write phase in the graph, a duplicate entity resolution process is performed. During writing, in fact, the `Neo4jWriter` creates a new node for each entity identified in the text, without checking whether that same entity was already present.

To address this problem, the system uses the `SinglePropertyExactMatchResolver` component, included in the `neo4j_graphrag` package. This module parses the nodes with label `_Entity_` present in the graphrag, identifying and matching nodes that share the same value in the `name` property. Running the resolver results in the automatic replacement of duplicate nodes with a single consolidated instance.

The component can be customized to accommodate different domains or more

complex resolving strategies, such as by including other distinguishing properties or comparisons based on semantic similarity or embedding.

This approach, while simple, proved effective in the prototyping phase, reducing redundancy and improving the quality of the knowledge base. Going forward, the system may be extended with more sophisticated resolvers capable of handling more complex ambiguities and ensuring finer disambiguation of entities.

### **3.2.5 GraphRAG**

The GraphRAG module allows the user to ask questions in natural language and obtain answers based on the knowledge contained in the graph. This component implements a Graph Retrieval-Augmented Generation approach, as described in Section 2.5, in which answer generation is driven by a structured information context extracted from a knowledge graph.

To realize this mechanism, the `neo4j-graphrag` package was used, which simplifies the implementation of the GraphRAG paradigm by offering modular components already integrated with Neo4j. Specifically, the package allows for the selection of:

- a **retriever**, responsible for retrieving the context from the graph;
- an **LLM**, responsible for generating the response from the extracted context.

The process consists of two main steps illustrated in Figure 3.9:

#### **1. Context Retrieval**

Starting with the user's query, the retriever, in our case the `HybridCypherRetriever`, queries the Neo4j database to extract the most relevant information. The result is a structured context, composed of entities, relationships and properties, obtained by combining semantic search and graph traversal. The detailed operation of the retriever used will be described in the next section.

#### **2. Answer Generation**

The retrieved context is then passed to a generative model (LLM) along with a custom prompt, which guides the formulation of the answer. The model

### 3. Original contribution to the solution of the problem

processes the information received and generates a coherent, question-centered text based on verifiable data.

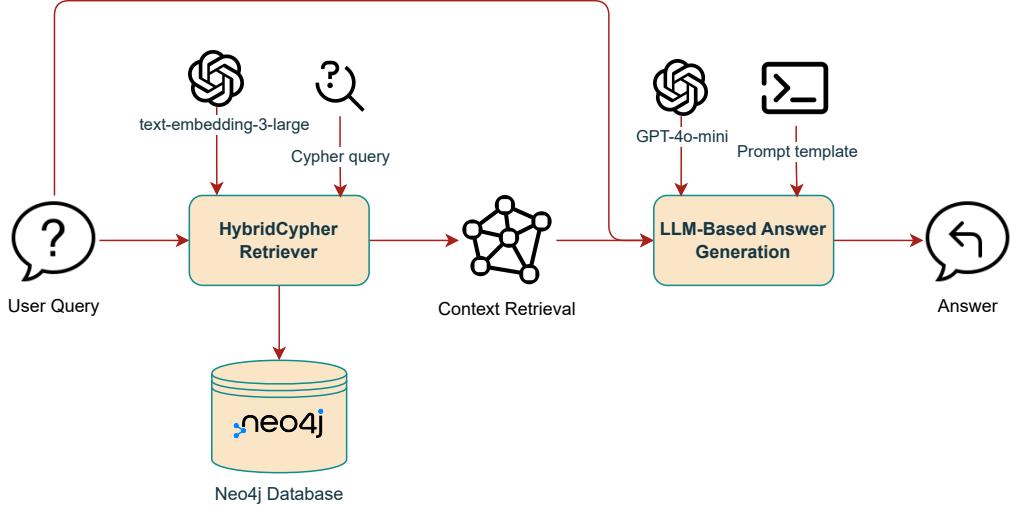


Figure 3.9: GraphRAG architecture: context retrieval and answer generation

As analyzed in Section 2.5.1, this approach enables the system to provide more complete, traceable, and efficient responses, overcoming the limitations of traditional RAG techniques based on unstructured documents. Indeed, the use of a structured knowledge graph makes it possible to improve the accuracy of responses, reduce the number of tokens used, increase the relevance of the context provided, and provide greater transparency and control over information sources.

#### HybridCypher Retriever

In the proposed system, information retrieval is handled by the `HybridCypherRetriever` component, included by the `neo4j-graphrag` package.

This module implements a hybrid retrieval strategy, combining the flexibility of **vector search**, the precision of **full-text search**, and the information richness derived from **structured graph traversal**.

As illustrated in Figure 3.10, the process consists of three main steps: initial node selection by combined search, normalization and selection of the most relevant results, and finally semantic enrichment by graph traversal.

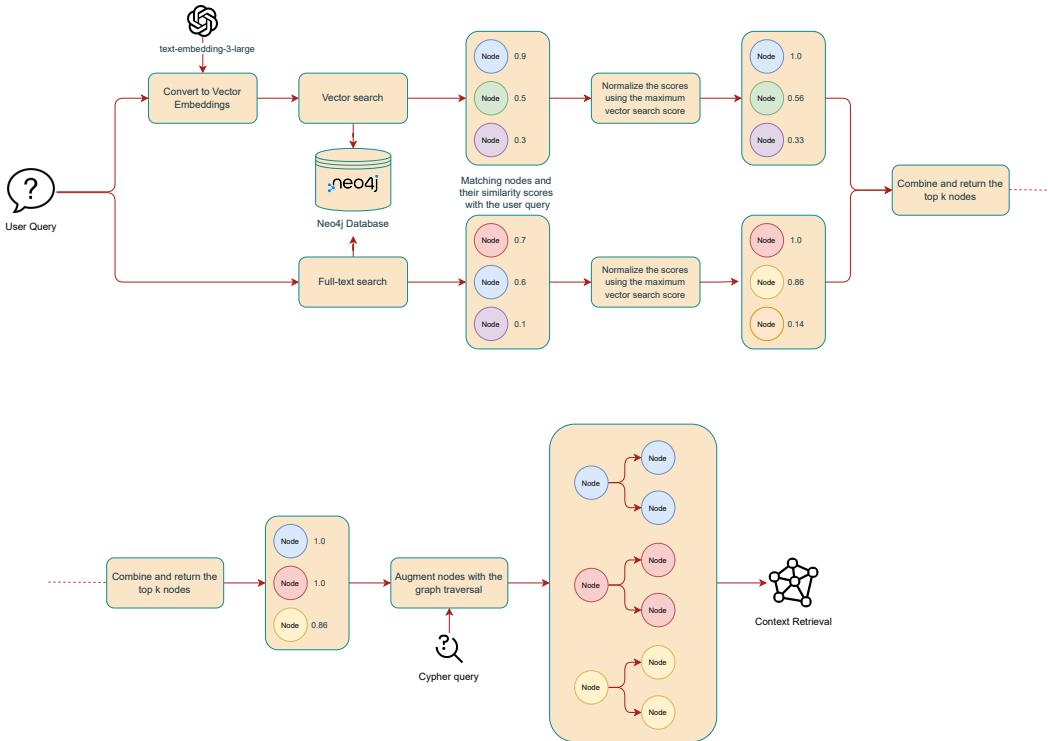


Figure 3.10: HybridCypherRetriever phases

## 1. Vector search

From the user's query, the system generates a text embedding, via the same model used by the Chunk Embedder. This numeric vector is compared with the embeddings stored in the `Chunk` type nodes stored in the graph, containing the user's textual input.

Thanks to the native support offered by Neo4j for vector indexes, the system is able to efficiently compute the semantic similarity between the query and the present content, returning a set of nodes with a score expressing conceptual closeness. This type of search is particularly useful for identifying content even if it is expressed in linguistic forms different from the query, but conceptually related.

## 2. Full-text search

In parallel, a full-text search is performed on the same query. Neo4j allows the creation of text indexes configured on one or more properties, in our case `Chunk.text`. This allows to quickly retrieve nodes that contain specific terms present in the query, such as proper names, dates, or exact expressions.

### *3. Original contribution to the solution of the problem*

---

This mode ensures greater lexical precision, which is particularly useful in cases where the user's query contains exact references (e.g., "*project with deadline 2023-10-20*") that would not be retrieved correctly by a search based only on semantic similarity.

### **3. Combination and normalization of scores**

The results obtained from both searches are normalized against the maximum score for each list, ensuring uniformity between the two metrics. The normalized scores are then combined to obtain a single sorted list of nodes, from which the `top-k` most relevant nodes are selected. These represent the starting nodes for the graph traversal phase.

### **4. Graph traversal**

From the selected nodes, the system performs a graph traversal operation to extend the retrieved context. The goal is to collect all the information conceptually related to the initial nodes, using the semantic structure of the graph.

The traversal used in this phase is not based on a rigid set of relationships predefined by the schema, but traverses neighboring nodes within a limited number of hops (usually 2 or 3), excluding only redundant connections such as the `FROM_CHUNK` relationship. This approach makes it possible to retrieve, from `Chunk` nodes, related semantic entities (e.g., `Project`, `Activity`, `Person`), the relevant properties of those entities, and the relationships that connect them to each other.

This choice is motivated by the fact that the application domain is very narrow and well delimited, focusing on key concepts such as activities, events, projects, places, and people. In this context, every piece of related information is potentially useful for response generation, and there is no need to filter *a priori* the relationships to be traversed.

The integration of vector search, full-text search, and graph traversal allows the system to overcome the limitations of each technique taken individually. Vector search, for example, is not effective for comparing exact values such as dates or strings, while full-text does not capture more complex conceptual relationships. Traversal in the graph adds depth, allowing implicit and structured connections to be retrieved, taking full advantage of the semantics of the knowledge graph.

### *3. Original contribution to the solution of the problem*

---

The end result is a rich, structured information context composed of original textual fragments, entities, relationships, and related properties. This information set is used by the LLM model to generate a response based on the data previously stored by the system in the knowledge graph.

#### **3.2.6 Neo4j Graph**

Neo4j, one of the most widely used platforms for graph database management, was chosen for the representation and management of the system's knowledge base.

In addition to offering an infrastructure suitable for modeling entities and relationships, Neo4j also supports a number of features useful for implementing the GraphRAG paradigm. In particular, Neo4j integrates tools that simplify the implementation of semantic pipelines, including:

- the ability to **store vector embeddings** directly in graph nodes;
- support for the **declaration and management of vector indexes**, used to compute semantic similarities;
- **built-in semantic search**, which allows vector queries to be performed without resorting to external systems (such as FAISS, Weaviate or Pinecone);
- the ability to **create full-text indexes** on text properties useful for performing quick and targeted keyword-based searches.

#### **Neo4j Aura Instance**

In the proposed system, an **AuraDB** instance was used, which allowed the knowledge base to be developed and tested in a convenient way, without burdening local resources and with the possibility of remote access at any time.

Access to the database is through the official Python driver `GraphDatabase.driver`, configuring the `URI`, `USERNAME` and `PASSWORD` of the instance. In addition, Neo4j Aura provides an integrated web interface, accessible via a browser, which effectively supported the development phase. This configuration facilitated verification of the generated graph, response analysis, and debugging during the development phase.

## Graph structure

The knowledge base is organized into two main subgraphs, as anticipated in Section 3.2.4:

- The **lexical graph**, which represents the textual content entered by the user. It includes the **Chunk** and **Document** nodes, associated with the textual fragments and the documents from which they originate, respectively. Each node of type **Chunk** contains the **text** property with the textual content and an **embedding** property that stores the computed vector representation, used in the semantic retrieval phase.
- The **domain graph**, constructed from the **TAMOntology** ontology, includes nodes such as **Person**, **Activity**, **Project**, **Event**, **Place**, and **RoutineActivity**, each with its semantic properties (e.g., **name**, **onDate**, **priority**, **dueDate**, etc.).

The two subgraphs are connected via **FROM\_CHUNK** type relationships, which allow tracing back from user input to associated entities and are exploited in the graph traversal phase for context retrieval.

## Integration and querying

The graph is updated dynamically: with each new textual input from the user, the **KGConstruction** component parses the text and inserts new nodes and relationships or integrates information with those already present.

In the current implementation, the graph is queried exclusively by the **GraphRAG** module via the **HybridCypherRetriever** retriever. This does not exclude the possibility of implementing direct queries via Cypher queries, for example, to view all current activities, all open projects, or to provide an interactive interface based on semantic filtering.

## Extension potentials

One of the main benefits of using a graph database and defining the schema via ontology is the ability to easily extend the application domain, both in terms of new classes (e.g., **Medication**, **Appointment**) and new properties or relationships.

Neo4j also allows the configuration of custom indexes, integrity constraints, and multilevel security profiles. Although these aspects have not yet been implemented in the system, they may be added at a later stage to ensure greater scalability, governance, and data protection.

### **3.2.7 Embedding and GenerativeAI Models**

The system employs next-generation language models to support the main phases of the pipeline: semantic content representation, structured information extraction, and response generation. All models used are provided by OpenAI and integrated via their respective APIs.

#### **Semantic Embedding Calculation**

For generating semantic embeddings, the `text-embedding-3-large` model is employed. This model transforms each text segment into a dense numerical vector that reflects the semantic meaning of the content, enabling comparison between texts even when expressed with different vocabulary or syntax. The generated embeddings are used both for semantic indexing of `Chunk` nodes in the graph and for the retrieval phase.

#### **Entity and Relation Extraction**

The automatic extraction of entities and relationships from the texts is performed using the `gpt-4o` model, through a controlled generation procedure. The model is provided with the chunked input text along with a formal semantic schema (derived from the domain ontology), which defines the recognizable entities, properties, and allowable relationships. For each chunk, the model returns a set of triples representing the structured information to be inserted into the knowledge graph.

#### **Response Generation**

Natural language response generation is carried out using the `gpt-4o-mini` model, selected for its balance between accuracy and computational cost. The model receives as input a static prompt and the context retrieved by the retriever (both in the RAG and GraphRAG variants). The prompt defines the expected structure of the response and the instructions to follow, while the context provides

the model with the information on which to base a coherent, personalized, and verifiable response.

### 3.3 Technological and/or application innovations

The main technological contribution of this work is the integration of semantic support via ontology in the construction and management of the knowledge graph. The **TAMOntology**, designed specifically to represent activities, events, and projects related to time management in people with cognitive difficulties, is not limited to defining a set of concepts, but is used as a reference schema to organize the data within the knowledge graph in a structured way.

Through the **Ontology Parser** module, developed as part of this thesis, the ontology schema is converted into an operational representation that guides the insertion of entities, properties, and relationships extracted from the text, ensuring that they respect the intended structure. This approach ensures semantic consistency throughout all stages of populating and updating the graph, facilitating data interpretation and reuse.

In addition to improving content organization, the use of ontology also enables inference, that is, inferring new information from existing information. Through hierarchies between classes and the explication of relationships, the system may be able to automatically enrich the graph with new connections and implicit properties. This makes the graph more “intelligent,” allowing it to pose more complex queries and obtain consistent results even in the presence of partial data.

The approach taken reflects a recent trend also promoted by Neo4j, particularly in the second season of the *Going Meta* [102] series, posted on YouTube as of September 3, 2024. These videos emphasize the value of ontologies as a guiding tool for building knowledge graphs from unstructured data, highlighting how a well-defined semantic schema can improve the quality, understanding, and usability of data.

A second distinguishing feature of the thesis work is the adoption of the **Graph-based Retrieval-Augmented Generation (GraphRAG)** paradigm for response generation. GraphRAG is a recent approach, introduced by Microsoft

### *3. Original contribution to the solution of the problem*

---

[103] in February 2024, which aims to overcome some of the main critical issues of LLMs, such as the production of hallucinated content, lack of transparency about the origin of the data and the handling of sensitive information, an issue that is particularly relevant in the healthcare and assistive context.

Unlike traditional RAG, which works on unstructured documents, GraphRAG allows direct querying of a knowledge graph, taking advantage of the relational structure and semantic links between entities. This enables the system to provide more accurate, complete and traceable answers built from organized and verifiable data.

From an application perspective, the use of GraphRAG to support people with **Mild Cognitive Impairment (MCI)** is still an under-explored area. To date, there are no solutions in the literature that adopt this approach within **Time Assistive Technologies (TAT)**.

Through the integration of GraphRAG and the use of a semantically structured knowledge graph, the system is able not only to remember future activities or events, but also to reconstruct what the user has already done, monitor current projects, and understand the relationships between different activities. This ability to reconstruct the information context is particularly useful for people with cognitive difficulties, as it enables the user to understand as well as remember.

The system is also designed to adapt progressively to the user, building a personalized profile that allows interactions to be tailored to the user's habits and information. This type of personalization, already discussed in Section 2.2.1, increases the accessibility, relevance of responses, and effectiveness of support provided to people with MCI.

## **3.4 Tools, technologies and models used in the implementation**

Models, technologies, and tools used to implement the different tasks are listed below.

### **3.4.1 Protégé**

Protégé is an open-source editor for designing and managing ontologies, developed primarily for knowledge engineering and the semantic web. It supports standard languages such as OWL and RDF(S), allowing the definition of classes, properties, individuals, and logical rules to model specific domains. It integrates automatic reasoning engines (e.g., HermiT, Pellet) to check logical consistency and infer implicit knowledge. Protégé offers an intuitive graphical interface, is extensible via plugins, and supports team collaboration; in fact, it is available both as a desktop application and as a collaborative web platform (WebProtégé).

### **3.4.2 Neo4j**

Neo4j is a native graph database based on the property graph model, in which data are represented as nodes and relationships with associated attributes. The declarative, GQL-compliant Cypher query language allows complex search patterns to be expressed intuitively, similar to SQL but optimized for graph structures. Neo4j supports storage and search on vector embeddings using specialized indexes, enabling efficient semantic search execution. It also offers advanced features such as parallel query execution, creation of indexes optimized for different data types, and integration with analytics and machine learning tools.

#### **Neo4j Aura**

Neo4j AuraDB is a cloud-based graph database service designed to offer the advanced features of Neo4j with the simplicity and scalability of the cloud. It offers advanced features such as automatic scalability, parallel query execution, and integration with analytics and machine learning tools, all with automatic updates and no need for manual infrastructure management.

### **3.4.3 Python Framework and Package**

The entire system was developed and tested in the Python environment, taking advantage of the high flexibility and wide availability of libraries for natural language processing, graph manipulation, and interaction with language models. Specifically, the following frameworks and packages were used to support the different components of the system.

#### **neo4j-graphrag package**

The `neo4j-graphrag` package is a Python library designed to implement Retrieval-Augmented Generation (RAG) pipelines on knowledge graphs constructed in Neo4j. It provides modular components for automatic extraction of entities and relations from natural language text, generation of semantic embeddings, incremental construction of knowledge graphs, and definition of hybrid retrievers based on Cypher queries and vector search. It supports integration with LLM models, efficient indexing of information and contextualized generation of responses, making it particularly suitable for the development of GraphRAG-based intelligent systems.

#### **RAGAS**

RAGAS (Retrieval-Augmented Generation Assessment) is an open-source framework for automatic evaluation of language model-based RAG systems. It provides standardized metrics to measure key aspects such as relevance of retrieved context, fidelity of responses to sources, and semantic quality of generation. The framework enables both quantitative and qualitative analysis of RAG pipelines, facilitating comparison between models and identification of critical issues at the retrieval or generation stage.

#### **Streamlit**

Streamlit is an open-source framework for creating interactive Web interfaces in Python, designed to simplify the development of prototypes and data visualization tools. It allows dynamic applications to be built using simple Python scripts, automatically updating the interface in response to user input.

### **3.4.4 OpenAI Models**

The OpenAI API provides access to a wide range of advanced language models designed to support natural language processing, text generation and automatic reasoning applications. Among the available models, those used in the system include the following:

#### **text-embedding-3-large**

Specialized model for generating vector embeddings from text. Optimized to represent sentences, documents or queries semantically, it is used for similarity, clustering and retrieval tasks in semantic search pipelines. It supports high-dimensional vector spaces (up to 3072) with excellent computational efficiency.

#### **gpt-4o**

Advanced multimodal model capable of processing textual, visual, and audio input in an integrated manner. It represents the state of the art in generative models, with fast response time and high accuracy, making it suitable for complex conversational systems and interactive applications in real-time environments.

#### **gpt-4o-mini**

Compact and optimized version of the GPT-4o model, designed to maintain good generative quality with reduced computational cost. Ideal for text generation and completion tasks in scenarios requiring high responsiveness and contextualized content, but with limited resources.

# Chapter 4

## Experimental validation and application aspects

### 4.1 Description of evaluation metrics

Once the system described in Chapter 3 had been designed and developed, it was deemed important to evaluate its performance in order to justify the use of knowledge graphs over the traditional RAG methodology, which relies solely on vector semantic search. The objective of the evaluation was to analyze the extent to which the GraphRAG approach, based on querying a structured, traversable knowledge graph, was able to retrieve a more effective information context and generate more accurate, relevant, and verifiable answers than classical RAG solutions.

To carry out this analysis, the **Retrieval-Augmented Generation Assessment Suite (RAGAS) framework** [104] was adopted, a suite of specialized metrics designed to evaluate RAG systems more comprehensively and adherently than metrics traditionally used in natural language processing, such as *BLEU* or *ROUGE*. Indeed, the latter are based on superficial textual comparisons and do not take into account source fidelity, semantic relevance, or actual use of the retrieved context.

RAGAS, on the other hand, focuses on aspects central to the RAG and GraphRAG systems, including:

- the accuracy of the context retrieved by the system,

- the coverage of the information needed to answer the question,
- the fidelity of the answer to the data actually available,
- and the relevance of the answer to the user’s query.

The importance of using these metrics is also emphasized in recent studies and industry articles [81, 105, 106, 107], which point out that LLM and retrieval based systems require evaluation tools that are more specific and related to the accuracy of the information rather than just the linguistic form. Furthermore, Neo4j itself has adopted the RAGAS framework to evaluate the performance of its Knowledge Graph Builder [108], a cloud application that builds a knowledge graph from textual documents and allows queries through RAG and GraphRAG approaches. This confirms the relevance and deployment of the framework also in industrial contexts and on related technical solutions to the one developed in this thesis.

For the evaluation of the system, the four main metrics proposed by the RAGAS framework were selected and divided into two distinct groups reflecting the two basic phases of the RAG paradigm:

- **Metrics related to the retrieval phase:**
  - *Context Precision*
  - *Context Recall*
- **Metrics related to the generation phase:**
  - *Faithfulness*
  - *Answer Relevancy*

This breakdown is represented graphically in Figure 4.1, taken from a technical guide [109], which provides a concise but effective overview of the metrics used to evaluate Retrieval-Augmented Generation-based systems. In addition to these metrics, a complementary metric, called *Answer Similarity*, was computed, which measures the semantic similarity between the response generated by the system and a predefined reference response. This metric, not natively included in RAGAS but useful for qualitative analyses, was introduced to assess how close responses are linguistically and conceptually to an “expected” response, even in

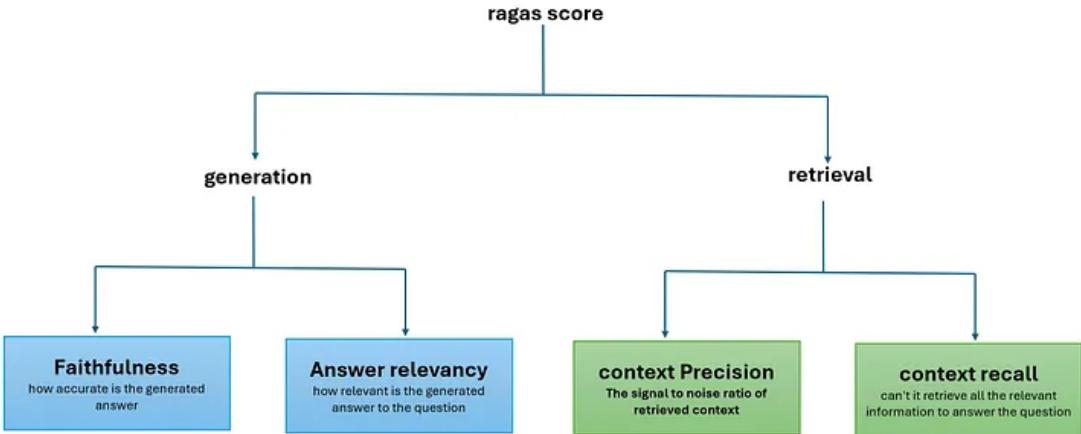


Figure 4.1: RAGAS score

cases where they are expressed differently.

In the following sections, each metric is described in detail, specifying its formal definition, how it is calculated, and the rationale behind its adoption in this paper. For this purpose, we will use a general notation defined as follows:

- **question**: the question posed by the user to the system;
- **context**: the set of documents or nodes returned by the RAG or GraphRAG systems in the retrieval phase and used by the LLM during generation;
- **ground\_truth**: the correct or expected answer, used as a reference for evaluation;
- **answer**: the answer actually generated by the system.

#### 4.1.1 Context Precision

The **Context Precision** metric is used to evaluate how much context retrieved by a RAG or GraphRAG system is actually useful in answering a given question correctly. Specifically, this metric measures the percentage of relevant information items (nodes, sentences, or documents) within the context provided to the model during the response generation phase.

In the present work, we chose to adopt the verification-based calculation mode

using LLM, comparing the retrieved context with the expected correct answer (`ground truth`), and not with the original contexts. This is because, using the GraphRAG approach, the retrieved data do not correspond to static documents, but to nodes and relationships dynamically selected from the knowledge graph. A direct comparison with the reference documents would have led to biased and meaningless evaluations.

The LLM-based mode, on the other hand, allows us to estimate, for each context element, whether it is actually useful for generating a correct answer by comparing it directly with ground truth.

### Evaluation process

The evaluation process is represented in the conceptual diagram in Figure 4.2, taken from the RAGAS guide :

1. The user provides a `question`;
2. The system retrieves a set of  $K$  `context` items;
3. The LLM model compares each context element with the `ground truth`;
4. For each element, a `verdict` (0 = not relevant, 1 = relevant) is returned, accompanied by a `reason` justifying the evaluation.

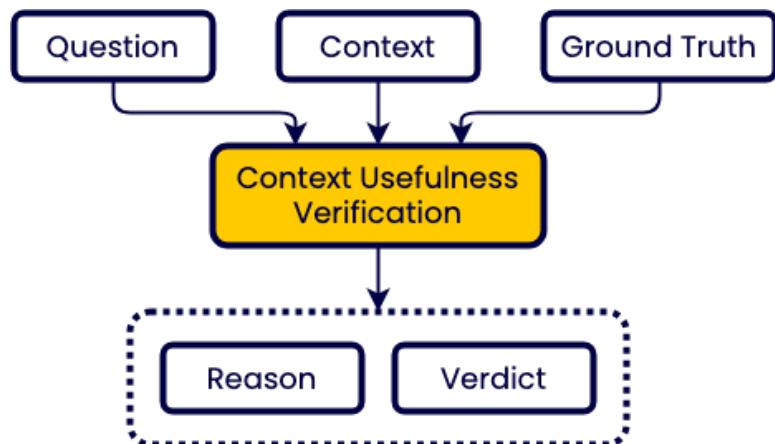


Figure 4.2: Context Precision schema

## Formula

The calculation is based on the concept of **Precision@ $k$** , where  $k$  represents the position of the item in the retrieved context ranked by relevance. The formula is:

$$\text{Precision}@k = \frac{\text{True Positives}@k}{\text{True Positives}@k + \text{False Positives}@k} \quad (4.1)$$

In the case of a ranked list of  $K$  elements, the overall **Context Precision** is defined as the weighted sum of the Precision@ $k$  values associated with the elements judged as relevant:

$$\text{Context Precision}@K = \frac{\sum_{k=1}^K (\text{Precision}@k \times v_k)}{\text{Total number of relevant items in the top K results}} \quad (4.2)$$

where  $v_k$  is equal to 1 if the element at position  $k$  is relevant, and 0 otherwise.

## Interpretation

Elements in the first positions of the context have a greater impact on the final result. This is because each Precision@ $k$  is multiplied by the indicator  $v_k$  and contributes to the sum only if the element is relevant. Consequently, if useful information is retrieved in the first positions, Context Precision will be higher.

This also reflects the behavior of LLMs, which tend to give more weight to the initial information in the prompt.

A high value of Context Precision indicates that the system is able to retrieve little but relevant content, while a low value suggests the presence of a lot of unnecessary or off-topic information in the context provided. This type of evaluation is particularly useful in GraphRAG systems, where the context can be enriched with structured elements, but it is critical to avoid information overload

### 4.1.2 Context Recall

The **Context Recall** metric measures the ability of the system to retrieve all relevant information needed to answer a question, without missing important elements. Unlike Context Precision, it assesses how much of the useful content actually needed was retrieved.

For an LLM-based calculation, RAGAS defines Context Recall as the proportion of sentences (or statements) in the ground truth that can be attributed to the context provided.

$$\text{Context Recall} = \frac{\text{Number of claims supported by the retrieved context}}{\text{Total number of claims in the reference}} \quad (4.3)$$

### Evaluation process

As highlighted in Figure 4.3, the calculation is based on the extraction of statements from the expected response and their classification by the LLM, which evaluates for each statement whether it can be justified (1 = justified, 0 = not justified) by at least one piece of information contained in the retrieved context.

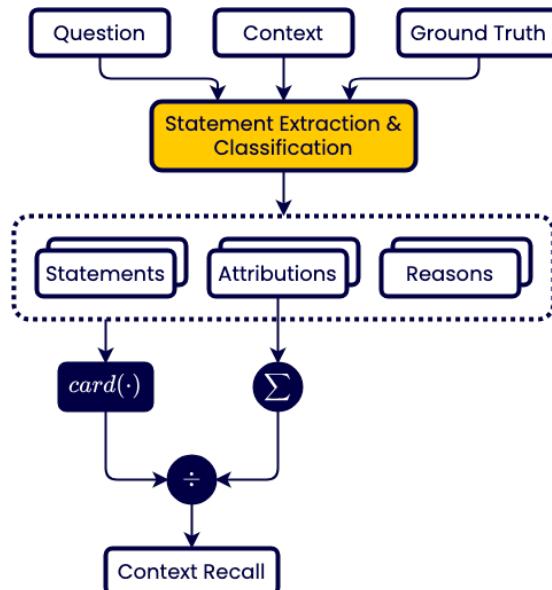


Figure 4.3: Context recall schema

### Interpretation

A high Context Recall value indicates that the system is able to retrieve most of the information needed to generate a complete and correct response. In contrast, a low value suggests that the context is partial or insufficient, and that some parts of the expected response cannot be inferred from the data provided.

In the context of GraphRAG systems, this metric is particularly relevant because it allows us to measure the overall effectiveness of graph traversal: a good traversal should be able to “reach” nodes containing all key information.

### 4.1.3 Faithfulness

The **Faithfulness** metric measures how well a generated response is actually supported by the context provided as input to the model. In other words, it assesses whether the information contained in the response can be inferred, justified, or verified from the data retrieved in the retrieval phase.

This metric is critical for identifying the phenomenon of hallucinations in generative language models, i.e., the inclusion of content that is not supported by the sources, which is a significant risk in sensitive contexts such as health or medical.

#### Assessment Process

Faithfulness assessment is based on a two-step process, supported by two separate prompts provided to an LLM, as illustrated in the diagram in Figure 4.4:

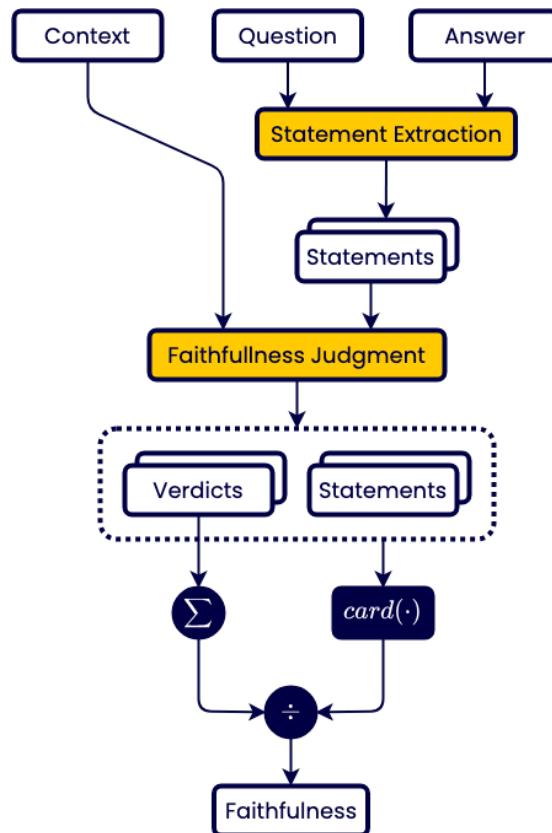


Figure 4.4: Faithfulness schema

#### 1. Extraction of statements:

From the generated response (**answer**), the model extracts a set of explicit or implicit statements, which constitute the set of sentences to be verified.

## 2. Verification against context:

Each statement is compared against the context provided to the model to generate the answer. The LLM assigns a binary verdict to each statement (1 if the statement is justified by the context, 0 if unsupported).

### Formula

The Faithfulness score is calculated as the proportion of statements justified by context to the total number of statements extracted:

$$\text{Faithfulness} = \frac{\sum \text{verdict}}{|\text{Statements}|} \quad (4.4)$$

where:

- $\sum \text{verdict}$  is the number of statements classified as true (value 1);
- $|\text{Statements}|$  is the total number of statements contained in the response.

### Interpretation

A high Faithfulness score indicates that the response produced by the system is fully supported by the information in the retrieved context, without introducing external or unverifiable elements. In contrast, a low value may indicate the presence of inaccurate or unwarranted information, which undermines the overall reliability of the system.

In the case of the proposed GraphRAG-based system, the use of structured and traceable data improves the model's ability to generate justifiable answers, and this metric directly measures its effectiveness.

#### 4.1.4 Answer Relevancy

The **Answer Relevancy** metric measures how relevant the response generated by the system is to the question asked. Unlike Faithfulness, which focuses on the truthfulness of information with respect to context, Answer Relevancy assesses whether the answer is truly centered on the intent of the question, penalizing answers that are incomplete or include redundant information.

## Evaluation Process

The process, illustrated in Figure 4.5, consists of two main steps, both of which are implemented through an LLM:

**1. Generation of a hypothetical question from the generated answer:**

The model receives the **Answer** and **Context**, and generates a new **Generated Question** that, according to the model, could have produced that answer.

**2. Relevance Evaluation:**

- The *semantic similarity* between the Generated Question and the original Question is calculated, using the *cosine similarity between the respective embeddings*.
- In addition, the system classifies the answer as *noncommittal* if it is too general or vague (e.g., "I don't know," "it depends," "it could be..."), assigning a minimum score in that case.

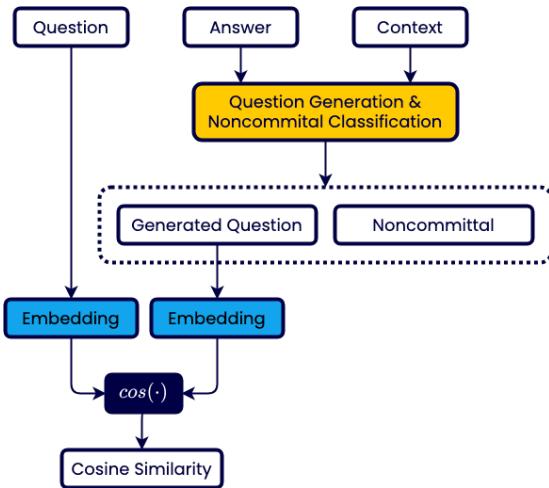


Figure 4.5: Answer Relevancy schema

## Formula

The final score is calculated as the **cosine similarity** between the embeddings of the original question and the generated question, penalized if the answer is classified as non-committal. Formally:

$$\text{Answer Relevancy} = \frac{1}{N} \sum_{i=1}^N \text{cosine similarity}(E_{g_i}, E_o) \quad (4.5)$$

## Cosine Similarity

Cosine similarity is a measure that quantifies how much two vectors are oriented in the same direction in space, regardless of their length. It is commonly used to compare semantic embeddings of texts, as it allows one to assess the similarity between two sentences or documents based on their meaning.

In the context of the *Answer Relevancy* metric, cosine similarity is calculated between the embedding of the original question and that of the question generated from the answer. The formula is:

$$\text{cosine similarity}(E_g, E_o) = \frac{E_g \cdot E_o}{\|E_g\| \cdot \|E_o\|} \quad (4.6)$$

Where:

- $E_g$  is the embedding of the **question generated** by the model from the answer,
- $E_o$  is the embedding of the **original question** posed by the user,
- the numerator represents the **scalar product** between the two vectors,
- the denominator is the product of their **norms (moduli)**.

The resulting value is between **-1 and 1**, where:

- **1** indicates perfectly aligned vectors (maximum similarity),
- **0** indicates orthogonal vectors (no similarity),
- **-1** indicates opposite vectors (maximum negative similarity, rare in texts).

## Interpretation

A high Answer Relevancy value suggests that the answer is accurate and directly related to the question, while a low score may indicate that the content is generic, unclear, or not relevant to the user's intent.

In the context of the proposed system, *Answer Relevancy* helps assess how relevant and focused the answers actually are, valuing those that not only contain correct information, but are also aligned with the user's information needs.

### 4.1.5 Answer Similarity

The **Answer Similarity** metric, also known as *Semantic Similarity*, measures how similar the system-generated answer is semantically to the expected correct answer, regardless of linguistic or syntactic form. Unlike the previous metrics, which assess the relevance or logical correctness of the answer, this metric focuses on overall meaning.

#### Calculation method

The calculation is based on the cosine similarity between the embeddings of the Answer and the Ground Truth, as shown in the diagram in Figure 4.6.

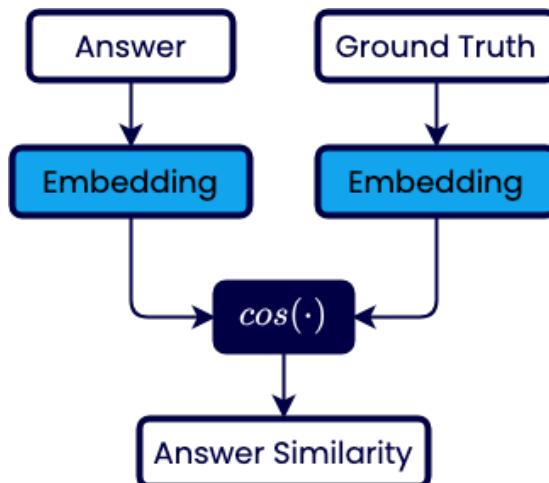


Figure 4.6: Answer Similarity schema

The formula is as follows:

$$\text{Answer Similarity} = \cos(\varphi) = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\| \cdot \|\mathbf{b}\|} \quad (4.7)$$

Where:

- **a** is the embedding of the **generated answer**,
- **b** is the embedding of the **ground truth**,
- the numerator represents the **scalar product** between the two vectors,
- the denominator is the product of the **norms** of the two vectors.

## Interpretation

A high value of *Answer Similarity* indicates that the answer produced is semantically compatible with the expected answer, even if worded differently. It is a simple but effective metric for assessing overall semantic consistency between responses, and can be useful for identifying significant divergences in meaning.

## 4.2 Definition of the Experimental Protocol

In order to rigorously evaluate the performance of the proposed system based on the GraphRAG approach and compare it with that of a traditional RAG-type implementation, a four-step experimental protocol was defined and followed.

The phases of the protocol are:

### 1. User profile construction.

Realistic synthetic profiles inspired by the *MemoryCompanion* dataset were created, simulating users with Mild Cognitive Impairment (MCI). From each profile, 30 natural language text interactions, representative of activities, appointments, routines and projects, were generated and subsequently used to populate the graph.

### 2. Knowledge graph generation.

The interactions were automatically inserted into the graph via the KG Construction pipeline. The graph was constructed in accordance with the TAMOntology ontology and visualized using Neo4j, checking for semantic consistency and correct populating of entities and relationships.

### 3. Creation of the test dataset.

A dataset of 10 expected questions and answers was generated for each profile, following predefined categories (e.g., deadlines, events, routines, project-related activities). This allowed us to evaluate the system's ability to respond to different types of queries.

### 4. Calculation of evaluation metrics.

Calculation of the metrics defined in Section 4.1 was done in an automated manner using the Python library of the RAGAS framework, which allowed direct comparison of the two approaches using the same set of questions,

expected responses, and model configurations.

Specifically, for each question in the test dataset, the following were generated:

- a response using the traditional RAG pipeline, which uses only semantic search;
- a response using the GraphRAG system, which also exploits graph traversal.

Both responses were associated with the information context retrieved from the respective retrievers, and all the expected metrics were automatically computed on this basis.

The results obtained were saved in CSV format for easy analysis and visualization, and also uploaded to the RAGAS interactive dashboard for more immediate and visual comparison.

## 5. Visualization of results.

The values obtained from calculating the metrics were represented graphically using Python libraries, facilitating direct comparison between the two query modes. The graphs highlighted the performance differences between the two approaches and support the analysis discussed in Section 4.4.

All experimental operations were implemented within an interactive Jupyter `test.ipynb` notebook, with the goal of ensuring repeatability and transparency of the tests. To run the notebook and replicate the described protocol, it is necessary to compile an `.env` configuration file containing the essential parameters for connecting to external services and running the system. The parameters to be configured are:

- **NEO4J\_URI** = Specifies the URL for connecting to the Neo4j Aura database, in a format compatible with the official Neo4j driver.
- **NEO4J\_USERNAME** = Username of the Aura instance to be used for authentication to the database.
- **NEO4J\_PASSWORD** = Password associated with the specified user, required to establish connection with Neo4j and access the knowledge base.
- **OPENAI\_API\_KEY** = API key required to access OpenAI services, used in the system for semantic embedding generation, response generation via LLM, automatic (LLM-based) evaluation of RAGAS metrics.

- `ONTOLOGY_FILE` = Path to the `.ttl` (Turtle) file containing the ontology used.

### 4.3 Description of the data used for experimentation

As described in Section 4.2, the evaluation of the system required the creation of two distinct types of datasets:

- a dataset of natural language interactions, used to build the system’s knowledge base (Knowledge Graph);
- a test dataset, consisting of questions and expected answers, used for experimental performance evaluation.

In detail, **10** interaction datasets were generated, each consisting of **30** textual sentences simulating a user’s dialogue with the system. For each such dataset, an associated test dataset was created, consisting of **10** custom questions based on the information contained in the constructed graph. In total, the system was evaluated on **100** questions, tested with both the GraphRAG approach and a traditional RAG implementation.

The amount of data used was defined taking into account several technical and economic constraints:

- the construction of the knowledge graph for each profile involves semantic processing of interactions through a pipeline using LLM models, with significant execution time and cost;
- evaluations with RAGAS also require the use of generative and embedding models, both of which are provided through paid APIs, with costs proportional to the number of requests made;

For these reasons, the choice to work with 10 user profiles and 100 queries in total is an effective compromise between experimental coverage and operational sustainability.

### 4.3.1 User Profiles Generated

To evaluate the performance of the proposed system, it was necessary to have a knowledge graph sufficiently populated with data consistent with respect to the application domain, i.e., the management of daily activities by users with cognitive difficulties. Therefore, it was critical to generate realistic textual inputs that would credibly represent the interactions a user might have with an intelligent time, project, and event monitoring system.

Since there are no structured public datasets for RAG/GraphRAG scenarios in the assistive domain, it was decided to build an ad hoc dataset from the synthetic profiles provided by the MemoryCompanion [19]. This dataset contains detailed descriptions of people with early-stage Alzheimer's disease, designed for use in simulation environments, but containing no real or identifiable data.

From these profiles, which included biographical information, interests, and routines, datasets of textual interactions were automatically generated through the use of large language models (LLMs). For each profile, a detailed prompt was provided to the model, designed to guide the generation of natural language sentences simulating a continuous dialogue between user and system.

Each resulting dataset contains 30 interactions consistent with the provided profile, and describes daily activities, appointments, recurring habits, family relationships, and personal goals, faithfully reflecting the domain of time management in assistive contexts.

#### Example of a profile in the MemoryCompanion dataset.

Each profile in the dataset provided by the authors of MemoryCompanion contains biographical information, medical conditions, personal goals, and support networks, useful for simulating realistic scenarios of interaction with assistive systems.

**Name:** Mateo    **Gender:** Male    **Age:** 45    **Ethnicity:** Italian

**Religion:** Catholic

**Medical Condition:** Early onset Alzheimer's    **First language:**  
Italian, fluent in English

**Family:** Wife, one daughter    **Location:** New York, Mateo was a

chef, running a popular Italian trattoria.

**What's important to him?** Celebrating family and tradition through food.

**What's happening for him at the moment?** He occasionally forgets recipes passed down for generations.

**What is the impact on him?** His confidence in the kitchen wanes.

**What would he like to happen in the future?** Mateo dreams of authoring a cookbook, mixing classic recipes with new culinary innovations.

**What strengths and support networks does he have?** His wife manages the restaurant, and food critics admire his culinary prowess.

### Prompt Used for Generating Interactions

For each profile, 30 textual interactions were generated using a structured prompt-driven LLM model designed to ensure variety, consistency, and semantic coverage with respect to the application domain.

The text prompt used is given below:

Generate a tabular dataset consisting of 30 rows and 3 columns:  
date (date on which the interaction takes place, can also be dates  
that are not close), user, interaction.

Simulate user interactions with an intelligent system designed  
monitor the user's activities, projects, and appointments.

User Profile: [PROFILE]

Detailed instructions for interactions:

- All interactions must be formulated in natural language,  
in the first person, varying in style, form, and verbs used.
- The system can recognize activities, projects, events, routines,  
places, and people, saving associated details in a database.

Mandatory interaction types to include within the dataset  
(spread across the 30 rows):

1. At least one interaction where a new project is added with a

- brief description and due date and priority.
2. Activities with relative time references (e.g., today, tomorrow).
  3. Activities with a specific date, hour and location.
  4. Three appointments specifying date, location, and person (indicates name).
  5. Only one routine activity with an explicit frequency (e.g., every Monday) (mentioned only once).
  6. One activity scheduled on the same day as another existing activity.
  7. At least 3 activities explicitly linked to the newly added project.
  8. Add a medicine or treatment to be taken with some frequency.

Ensure activities, locations, and people named are consistent with the provided user profile.

Output: Return the dataset in a file csv.

The prompt specifies:

- the use of natural language in the first person;
- the coverage of different types of interaction: activities, events, routines, appointments, projects;
- the contextual consistency with the user's biographical profile (places, relationships, interests);
- the requirement to include certain key elements in a distributed manner among the 30 rows.

### **Example of interaction dataset**

Once you run the prompt to generate the interaction dataset what you get is a csv file consisting of the date, user and interaction columns. An example dataset is shown in Table 4.1.

#### 4. Experimental validation and application aspects

---

Date	User	Interaction
2023-10-01	Mateo	Hi I'm Mateo and I want to start a new project: writing a cookbook. I'll call it 'Flavors of Tradition and Innovation'. The due date is December 31, 2024, and it's a high priority.
2023-10-02	Mateo	Today, I need to brainstorm ideas for the cookbook's introduction. I'll do this at home after lunch.
2023-10-03	Mateo	On October 10 2023, I'll visit the library to research traditional Italian recipes for the cookbook 'Flavors of Tradition and Innovation'.
2023-10-04	Mateo	I have an appointment with my editor, Sofia, on October 10, 2023, at 3:00 PM at her office in Manhattan.
2023-10-05	Mateo	Every Monday, I will dedicate two hours to writing a new recipe for the cookbook.
2023-10-06	Mateo	I need to take my medication, Donepezil, every morning after breakfast.
2023-10-07	Mateo	On October 15, 2023, at 10:00 AM, I will meet with a food critic, Marco, at my restaurant to discuss the cookbook 'Flavors of Tradition and Innovation'.
2023-10-08	Mateo	Today, I will test a new recipe for the cookbook: a modern twist on lasagna.
2023-10-09	Mateo	I have a family dinner tonight at 7:00 PM at home. I'll prepare the new lasagna recipe for my wife and daughter.
2023-10-10	Mateo	Met with Sofia today to discuss the cookbook's outline. She loved the concept!
2023-10-11	Mateo	Tomorrow, I will visit the farmers' market to source fresh ingredients for the cookbook recipes.
2023-10-12	Mateo	I need to write a draft of the cookbook's first chapter by October 20, 2023.
2023-10-13	Mateo	Today, I will organize my notes and recipes for the cookbook at my home office.
2023-10-14	Mateo	I have a meeting with my wife at the restaurant on October 16, 2023, at 11:00 AM to discuss the cookbook's budget.
2023-10-15	Mateo	Met with Marco today. He provided valuable feedback on the cookbook's theme.
2023-10-16	Mateo	Discussed the cookbook's budget with my wife. We agreed on a plan.
2023-10-17	Mateo	Today, I will finalize the list of recipes for the cookbook's first section.

Table 4.1: Example of user interaction dataset for KG population.

### 4.3.2 Knowledge Graph construction from User Interactions

In order to test the performance of the system, it was first necessary to construct a knowledge base containing user data, from which the two implemented solutions could perform information retrieval to answer the questions.

Once the datasets of interactions were constructed, these were used as input for the *KG Construction* component, described in Section 3.2.4. This component is responsible for parsing each textual sentence, extracting entities, relationships, and properties according to the semantic schema defined by *TAMOntology*, and inserting them into the knowledge graph implemented on Neo4j.

The following is an application example showing how, from a textual input, the pipeline automatically builds the corresponding nodes and relations in the graph.

#### Example of Knowledge Graph obtained

Given the first interaction as input to the KG Construction component, an initial version of the knowledge graph depicted in Figure 4.7 is obtained:

*"Hi I'm Mateo and I want to start a new project: writing a cookbook. I'll call it 'Flavors of Tradition and Innovation'. The due date is December 31, 2024, and it's a high priority."*

One can observe the construction of the lexical graph with the node of type `Chunk`, containing the textual content in the `text` property and the vector representation of the text in the `embedding` property.

Connected to the `Chunk` type node via the `FROM_CHUNK` relation are two nodes belonging to the **domain graph**, both labeled `_Entity_`, but associated with different ontology classes.

- The node named "Mateo" is of type `Person` and represents the user.
- The node named "Flavors of Tradition and Innovation" is of type `Project` and represents the project the user is working on.

#### 4. Experimental validation and application aspects

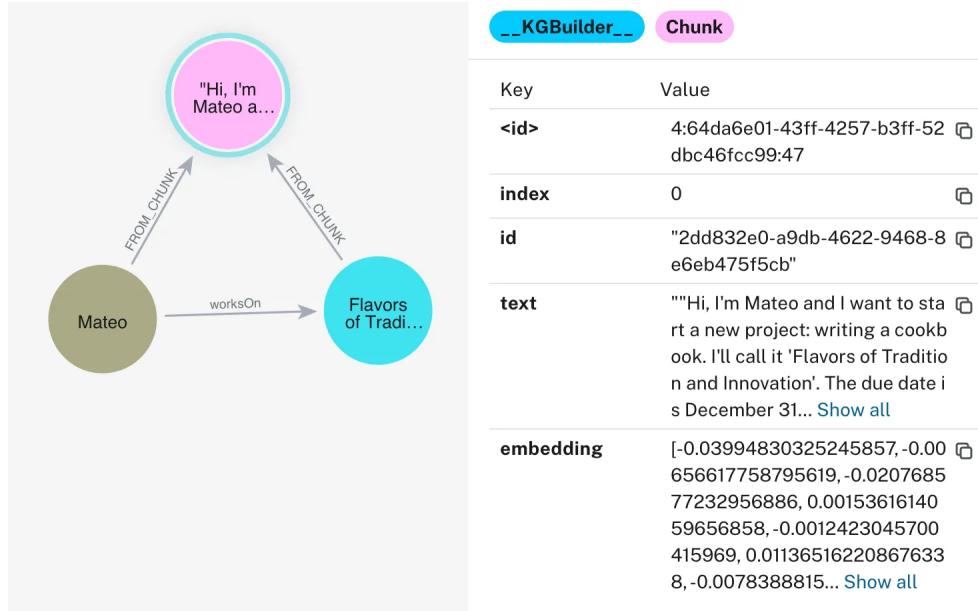


Figure 4.7: KG created from the first user interaction

As shown in Figure 4.8, the **Project** node also includes properties extracted from the text: `priorityProject = "high"` and `dueDate = "2024-12-31"`.

The two nodes are connected by a `worksOn` relationship, which indicates that Mateo is actively involved in the project *Flavors of Tradition and Innovation*.

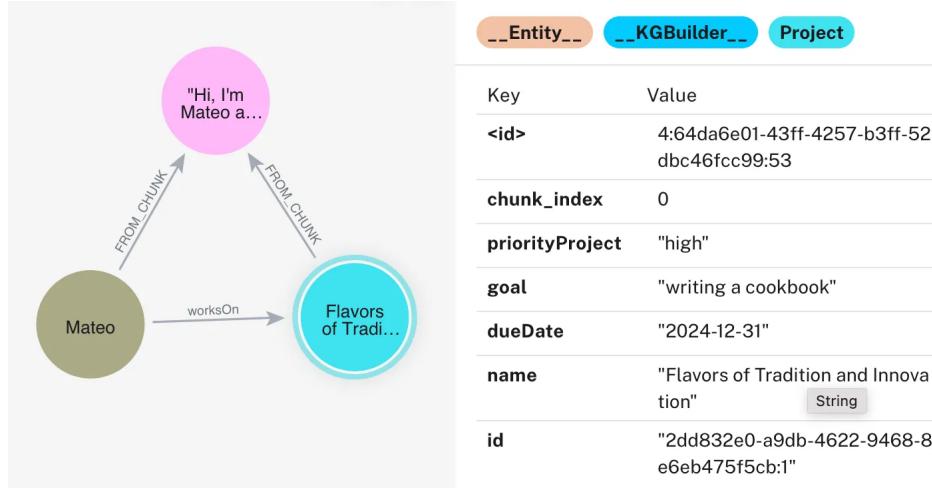


Figure 4.8: Example of properties associated to project node

Next, the system is executed on a second interaction containing specific dates and locations associated with the user:

*"On October 10 2023, I'll visit the library to research traditional Ital-*

ian recipes for the cookbook 'Flavors of Tradition and Innovation'."

Figure 4.9 shows the resulting graph, containing:

- A node of type **Activity** with **name** = "research traditional Italian recipes" and **onDate** = "2023-10-10";
- A node of type **Place** with **name** = "library";
- A **occursAt** relationship between the activity and the place;
- A **participatesIn** relationship between the user and the activity;
- A **projectActivity** relationship between the project and the activity.

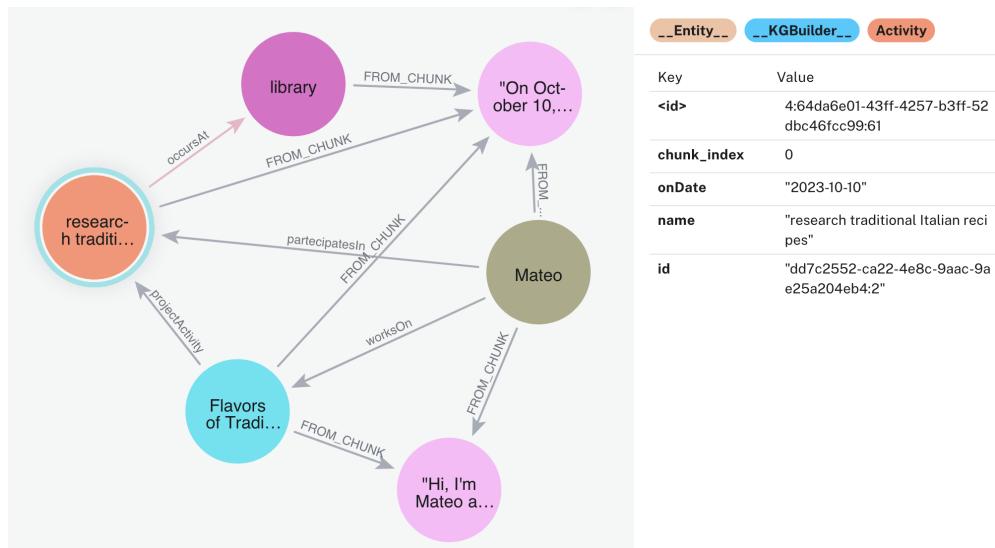


Figure 4.9: Graph generated from second user interaction

After all 30 interactions in the dataset are processed, the final domain graph represented in Figure 4.10 is obtained.

The total number of entities created is 43, including: **Activity**, **Event**, **Person**, **Place**, **Project** and **RoutineActivity**. All entities are visibly related to each other through the relationships defined in the ontology, and they all converge to the reference user **Mateo**.

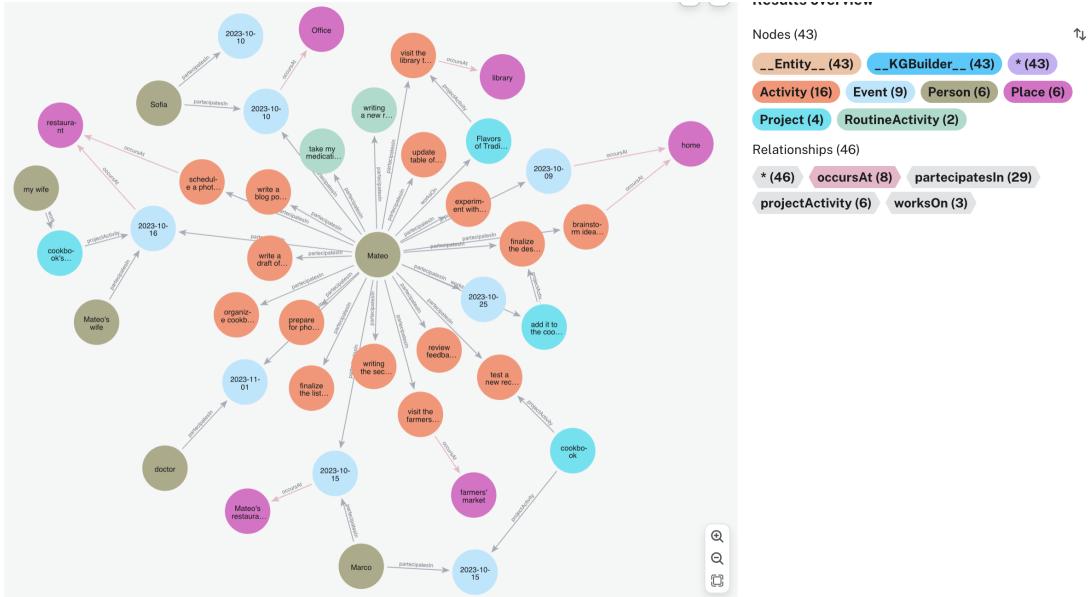


Figure 4.10: Domain graph resulting from processing all 30 interactions

### 4.3.3 Test Dataset

To evaluate the performance of the system in both context retrieval and response generation, a test dataset was constructed for each of the user profiles described in the previous section. Each dataset consists of **10** natural language questions, formulated as if they were actually asked by the user to the system, accompanied by an expected answer derived directly from the information contained in the knowledge graph.

The questions were created manually, following a well-defined pattern of the type of questions that must be present, represented in Table 4.2.

This scheme allowed a comprehensive evaluation of the system on different types of queries. The questions were designed to test both the ability to retrieve relevant information and the quality and relevance of the responses generated. The approach taken also allowed us to test whether the system was able to link together information distributed across multiple interactions and to reconstruct the user's overall context.

## 4. Experimental validation and application aspects

---

Question Type	Example of a question	Expected answer
Deadline for a specific project	What is the deadline for my 'Flavors of Tradition and Innovation' project?	The project is due on December 31, 2024.
Time and location of a specific event	When is my meeting with Sofia to discuss the cookbook?	Your meeting with Sofia is on October 10, 2023, at 3:00 PM at her office in Manhattan.
Planned activities on a specific day	What are my scheduled activities on 2023-10-24?	On 2023-10-24, you will experiment with a new dessert recipe and update the cookbook's table of contents.
Current projects	What are my current projects?	You are currently working on the cookbook project called 'Flavors of Tradition and Innovation'.
Activities or events at a specific location	What activities do I have at my restaurant?	Meeting with Marco on 2023-10-15, meeting with your wife on 2023-10-16, and a photoshoot on 2023-11-05.
Activities or events with a specific person	What activities involve Marco?	You met with Marco on October 15, 2023, at your restaurant to discuss the cookbook.
Routine-type activities and frequency	How often do I write new recipes for the cookbook?	You write a new recipe for the cookbook every Monday for two hours.
Activities related to a specific project	What tasks have I planned for my cookbook project?	You brainstorm the introduction, researched recipes, wrote chapters, finalized a dessert, prepared a photoshoot and wrote a blog post.
Reference to past events	When did I finish the draft of the first chapter?	You finished the draft of the first chapter on 2023-10-20.
Specific future activities	When will I write a blog post about the cookbook's progress?	You will write a blog post about the cookbook's progress on 2023-10-27.

Table 4.2: Examples of questions and expected answers in the test dataset.

## 4.4 Presentation and Analysis of results

After completing the experimental evaluation described in the previous sections, this section presents the results obtained in terms of RAGAS metrics, directly comparing system performance in GraphRAG and RAG modes.

The goal is to provide a quantitative and qualitative analysis of performance, highlighting both the overall benefits of the GraphRAG approach and any limitations or differences observed in individual user profiles.

### 4.4.1 Average comparison of metrics

This section presents the average values of the evaluation metrics described in Section 4.1, calculated on the 10 user profiles for each of the two implemented

approaches. The values were obtained through the RAGAS framework and aggregated from the ratings by individual question.

The results, summarized in the graph in Figure 4.11, are as follows:

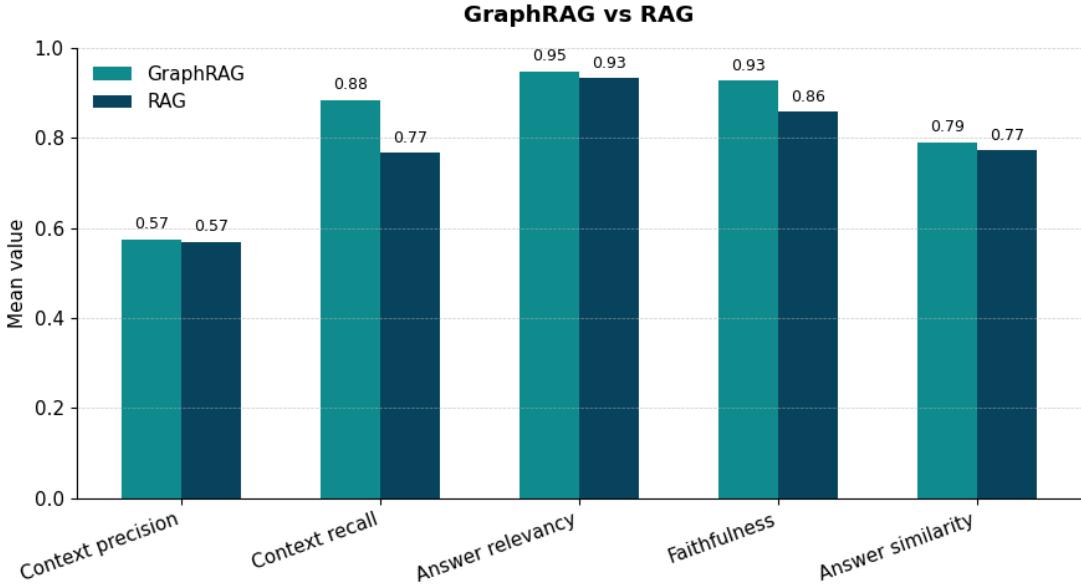


Figure 4.11: Comparison of average metric values between RAG and GraphRAG

### Context Precision

GraphRAG and RAG both achieved a *Context Precision* of 0.57, resulting in a difference of 0.00. This indicates that the two systems are equally accurate in selecting useful elements within the retrieved context. The identical average values reflect the use of the same initial strategy—semantic vector search on Chunk nodes. Importantly, the addition of structured nodes and relationships in GraphRAG does not introduce noise into the selection phase, confirming that semantic enrichment does not compromise precision.

### Context Recall

In terms of *Context Recall*, GraphRAG scored 0.88 compared to 0.77 for RAG, with a notable difference of +0.11. This improvement highlights the effectiveness of graph traversal in retrieving a more comprehensive set of relevant information. The enhanced coverage suggests that GraphRAG is better suited for scenarios in which completeness of information is critical to generating accurate and contextualized responses.

### Answer Relevancy

The *Answer Relevancy* metric yielded a value of 0.95 for GraphRAG and 0.93 for RAG, corresponding to a difference of +0.02. Both systems produced highly relevant answers, but GraphRAG’s slight advantage indicates that it tends to generate responses that are more closely aligned with the user’s intent. This refinement can be attributed to the semantic depth provided by the knowledge graph, which helps narrow the focus of the generation process.

### Faithfulness

For the *Faithfulness* metric, GraphRAG scored 0.93 versus 0.86 for RAG, resulting in a difference of +0.07. This metric measures how well the answer is supported by the retrieved context, and the higher value for GraphRAG indicates greater consistency and adherence to factual data. This is particularly important in domains such as assistive technologies, where the reliability and traceability of responses are essential.

### Answer Similarity

The *Answer Similarity* metric was 0.79 for GraphRAG and 0.76 for RAG, with a difference of +0.03. This result suggests that GraphRAG’s answers, while potentially more verbose or detailed, maintain a high level of semantic alignment with the expected reference answers. The increased similarity confirms that GraphRAG not only generates relevant and faithful content but also produces linguistically and conceptually coherent outputs.

#### 4.4.2 Analysis on individual profiles

To further explore the comparison between GraphRAG and RAG, the distribution of evaluation metrics on individual user profiles was analyzed.

Table 4.3 shows the values of the five main metrics calculated for each profile, in both approaches (G = GraphRAG, R = RAG).

#### 4. Experimental validation and application aspects

---

Profile	Context Precision(G/R)	Context Recall(G/R)	Answer Relevancy(G/R)	Faithfulness(G/R)	Semantic Similarity(G/R)
Mateo	0.56 / 0.53	0.89 / 0.68	0.96 / 0.95	0.90 / 0.68	0.80 / 0.74
Benjamin	0.61 / 0.49	0.88 / 0.72	0.94 / 0.93	0.92 / 0.74	0.76 / 0.75
Ella	0.58 / 0.58	0.83 / 0.75	0.94 / 0.94	0.98 / 0.88	0.85 / 0.81
Javier	0.39 / 0.42	0.81 / 0.63	0.95 / 0.92	0.83 / 0.80	0.73 / 0.69
Liliana	0.57 / 0.59	0.80 / 0.82	0.97 / 0.95	0.93 / 0.82	0.82 / 0.81
Naomi	0.65 / 0.66	0.95 / 0.83	0.96 / 0.94	0.91 / 0.97	0.83 / 0.81
Nathaniel	0.63 / 0.40	0.95 / 0.87	0.95 / 0.93	0.93 / 0.98	0.85 / 0.84
Nina	0.60 / 0.67	0.95 / 0.86	0.96 / 0.95	1.00 / 0.90	0.78 / 0.77
Priyanka	0.56 / 0.57	0.84 / 0.69	0.97 / 0.96	0.92 / 0.84	0.75 / 0.75
Rahul	0.46 / 0.56	0.88 / 0.70	0.97 / 0.94	0.89 / 0.85	0.76 / 0.71
Samuel	0.70 / 0.80	0.95 / 0.88	0.86 / 0.85	0.99 / 1.00	0.76 / 0.80

Table 4.3: Metric comparison on individual user profiles: GraphRAG vs. RAG

The main observations based on these data are as follows:

- **Consistent improvements on Context Recall:** GraphRAG outperforms RAG in all but one profile (*Liliana*), with improvements ranging from +0.07 to +0.22. This confirms that graph traversal allows retrieving more useful information than semantic search alone.
- **Answer Relevancy consistently high:** all profiles obtain values above 0.93 for both modes. GraphRAG outperforms RAG in 9 out of 11 profiles, indicating a greater ability to generate more targeted answers.
- **Average Faithfulness higher with GraphRAG:** In almost all profiles, GraphRAG improves the adherence of responses to context, with noticeable peaks in *Mateo* (+0.22) and *Benjamin* (+0.18). Only *Nathaniel* and *Naomi* show slightly lower, but still high results.
- **Stable Context Precision:** the difference between the two approaches is often marginal. Values remain concentrated between 0.55 and 0.65 in most profiles, with the exception of *Javier* (lower in both cases).
- **Semantic Similarity stable, with slight variations:** scores are very close in the two systems, but GraphRAG outperforms RAG in 9 out of 11 cases. This suggests slightly greater semantic compatibility with the expected response.
- **Mateo:** among the profiles with greatest benefit from GraphRAG, with improvements on all metrics, especially Faithfulness (+0.22) and Recall (+0.21).

- **Nathaniel:** shows the sharpest gap in Context Precision (+0.23), a sign that GraphRAG has selected a much more relevant context.
- **Samuel:** only profile in which RAG reaches the same Faithfulness as GraphRAG (1.00) and outperforms it in Context Precision and Semantic Similarity. However, the difference in the other metrics is minimal.

#### 4.4.3 Qualitative analysis on question types

In addition to the aggregate analysis and comparison by individual profile, a more detailed qualitative study was conducted on some specific types of questions, specifically observing the behavior of the two approaches with respect to **Mateo**'s profile.

##### Deadline for a specific project

- **Question:** What is the deadline for my '*Flavors of Tradition and Innovation*' project?
- **Reference Answer:** The project is due on December 31, 2024.
- **GraphRAG Response:** The deadline for your '*Flavors of Tradition and Innovation*' project is December 31, 2024.
- **RAG Response:** The deadline for your '*Flavors of Tradition and Innovation*' project is December 31, 2024.
- **Comment:** Identical responses from both systems; both correct and perfectly aligned with ground truth.

##### Time of an event

- **Question:** When is my meeting with Sofia to discuss the cookbook?
- **Reference Answer:** Your meeting with Sofia is on October 10, 2023, at 3:00 PM.
- **GraphRAG Response:** You met with Sofia on October 10, 2023, at 3:00 PM to discuss the cookbook's outline.
- **RAG Response:** Your meeting with Sofia to discuss the cookbook was on October 10, 2023, at 3:00 PM.

- **Comment:** Both responses correctly provide the date and time; wording is different but complete in both cases.

### Scheduled activities on a given day

- **Question:** What are my scheduled activities on 2023-10-24?
- **Reference Answer:** On 2023-10-24, you will experiment with a new dessert recipe and update the cookbook's table of contents.
- **GraphRAG Response:** On 2023-10-24, your scheduled activities are:
  - Update Table of Contents - You will update the cookbook's table of contents based on the latest changes.
  - Experiment with a Dessert Recipe - You will experiment with a new dessert recipe for the cookbook.
- **RAG Response:** You do not have any scheduled activities on 2023-10-24 according to the provided context.
- **Comment:** GraphRAG correctly returns both activities scheduled on the given date; RAG, on the other hand, does not identify any activities.

### Current projects

- **Question:** What are my current projects?
- **Reference Answer:** You are currently working on the cookbook project called '*Flavors of Tradition and Innovation*'.
- **GraphRAG Response:** Your current project is writing a cookbook titled "*Flavors of Tradition and Innovation*," which is a high-priority project due on December 31, 2024. Here are some of the specific activities and tasks related to this project:
  - Brainstorm ideas for the cookbook's introduction - Scheduled for October 2, 2023.
  - Visit the library to research traditional Italian recipes - Planned for October 10, 2023.
  - Writing the second chapter of the cookbook - Planned for October 19, 2023.

#### 4. Experimental validation and application aspects

---

- Write a draft of the cookbook's first chapter - Planned for October 20, 2023.
  - Finalize the list of recipes for the cookbook's first section - Planned for October 17, 2023.
  - Experiment with a dessert recipe - Scheduled for October 24, 2023.
  - Update the table of contents - Scheduled for October 24, 2023.
  - Finalize the dessert recipe - Planned for October 29, 2023.
  - Write a blog post about the cookbook's progress - Scheduled for October 27, 2023.
  - Schedule a photoshoot for the cookbook - Planned for November 5, 2023.
  - Prepare for the photoshoot - Planned for November 6, 2023.
  - Organize cookbook materials - Planned for October 30, 2023.
  - Review feedback and make revisions - Planned for October 21, 2023.
  - Visit the farmers' market to source fresh ingredients - Planned for October 12, 2023.
  - Test a new recipe - Scheduled for October 8, 2023.
- **RAG Response:** You are currently working on a cookbook titled "*Flavors of Tradition and Innovation*," which is a high-priority project due on December 31, 2024. Additionally, you have planned to organize your notes and recipes for the cookbook on October 13, 2023, and brainstorm ideas for the cookbook's introduction on October 2, 2023.
  - **Comment:** Both correctly recognize the active project; however, they add unsolicited details. In particular, RAG provides a more concise response, while GraphRAG returns a very detailed list of related activities.

#### Activities or events at a specific location

- **Question:** What activities do I have at my restaurant?
- **Reference Answer:** At your restaurant, you have a meeting with Marco to discuss the cookbook on 2023-10-15, a meeting with your wife to discuss the cookbook's budget on 2023-10-16, and a photoshoot for the cookbook scheduled on 2023-11-05.

- **GraphRAG Response:** At your restaurant, you have several activities planned, including:

- Scheduling a photoshoot for the cookbook on November 5, 2023.
- Meeting with your wife on October 16, 2023, at 11:00 AM to discuss the cookbook's budget.
- Finalizing the dessert recipe on October 29, 2023.
- Writing a blog post about the cookbook's progress on October 27, 2023.
- Experimenting with a dessert recipe on October 24, 2023.
- Organizing cookbook materials on October 30, 2023.
- Preparing for the photoshoot on November 6, 2023.

These activities are part of your ongoing work related to the cookbook you are writing, titled "*Flavors of Tradition and Innovation*."

- **RAG Response:** You have a photoshoot for the cookbook scheduled on November 5, 2023, and a meeting with your wife to discuss the cookbook's budget on October 16, 2023, both at the restaurant.
- **Comment:** RAG returns only events explicitly related to the restaurant, maintaining a good level of accuracy; GraphRAG includes multiple activities, some of which are not clearly located, risking an excess of information.

### Activities or events with a specific person

- **Question:** What activities involve Marco?
- **Reference Answer:** You met with Marco on October 15, 2023, at your restaurant to discuss the cookbook. He provided valuable feedback on its theme.
- **GraphRAG Response:** Marco is involved in the following activities:
  - Meeting with Mateo on October 15, 2023, to provide feedback on the cookbook's theme.
  - Discussing the cookbook "*Flavors of Tradition and Innovation*" with Mateo at Mateo's restaurant on the same date.

- **RAG Response:** The activities involving Marco include a meeting on October 15, 2023, where Mateo met with him to discuss the cookbook's theme and provided valuable feedback. Additionally, there is a scheduled meeting with Marco, a food critic, to discuss the cookbook "*Flavors of Tradition and Innovation.*"
- **Comment:** Both systems correctly recognize interaction with Marco; the difference is only stylistic.

### Routine-type activity and frequency

- **Question:** How often do I write new recipes for the cookbook?
- **Reference Answer:** You write a new recipe for the cookbook every Monday for two hours.
- **GraphRAG Response:** You write a new recipe for the cookbook every Monday, devoting two hours to this task.
- **RAG Response:** You write new recipes for the cookbook every Monday, dedicating two hours to this task.
- **Comment:** Identical responses, perfectly consistent with ground truth.

### Tasks related to a specific project

- **Question:** What tasks have I planned for my cookbook project?
- **Reference Answer:** You have planned several tasks for your cookbook project, including:
  - Brainstorming ideas for the introduction (scheduled for 2023-10-02),
  - Researching traditional Italian recipes at the library (2023-10-04),
  - Writing a new recipe every Monday,
  - Writing the first chapter (draft due by 2023-10-20),
  - Finalizing the list of recipes for the first section (2023-10-17),
  - Writing the second chapter (starting 2023-10-19),
  - Organizing notes and recipes (2023-10-13),

#### 4. Experimental validation and application aspects

---

- Reviewing feedback from Sofia and revising the first chapter (2023-10-21),
  - Updating the table of contents (2023-10-24),
  - Finalizing a dessert recipe (2023-10-29),
  - Organizing all cookbook materials for a photoshoot (2023-10-30),
  - Scheduling a photoshoot for the cookbook (set for 2023-11-05),
  - Writing a blog post about your progress (2023-10-27).
- **GraphRAG Response:** You have planned the following tasks for your cookbook project, "*Flavors of Tradition and Innovation*":
- Organize your notes and recipes (2023-10-13),
  - Finalize the list of recipes for the first section (2023-10-17),
  - Write a blog post about the cookbook's progress (2023-10-27),
  - Update the table of contents (2023-10-24),
  - Experiment with a dessert recipe (2023-10-24),
  - Review feedback and make revisions (2023-10-21),
  - Write a draft of the first chapter (2023-10-20),
  - Write the second chapter (2023-10-19),
  - Finalize the dessert recipe (2023-10-29),
  - Organize cookbook materials (2023-10-30),
  - Schedule a photoshoot (2023-11-05),
  - Prepare for the photoshoot (2023-11-06),
  - Visit the library (2023-10-10),
  - Visit the farmers' market (2023-10-12),
  - Test a new recipe (2023-10-08),
  - Brainstorm introduction ideas (2023-10-02).
- **RAG Response:** You have planned the following tasks for your cookbook project:
- Organize your notes and recipes for the cookbook at your home office (2023-10-13),

- Finalize the list of recipes for the first section (2023-10-17).
- **Comment:** RAG returns only two activities associated with the project, being incomplete; GraphRAG provides a complete list consistent with ground truth.

### Reference to past events

- **Question:** When did I finish the draft of the first chapter?
- **Reference Answer:** You finished the draft of the first chapter on 2023-10-20.
- **GraphRAG Response:** You finished the draft of the first chapter on October 20, 2023.
- **RAG Response:** You finished the draft of the first chapter on October 20, 2023.
- **Comment:** Both responses match ground truth; the date is correct and the responses are formally equivalent.

## 4.5 Assessment of the significance of the results obtained

The analysis conducted highlighted the tangible advantages of the **GraphRAG approach**, which combines the power of generative models with a structured and semantically coherent **knowledge graph**, built on the basis of an **ontology**. The use of the graph allowed the system to retrieve interrelated information, structure it in a queryable way, and generate more informative, traceable, and source-aligned responses compared to the traditional RAG approach.

In the domain of *Time Assistive Technologies*, the system's effectiveness depends not only on the correctness of the information, but also on its relevance, completeness, and reliability. Experimental results demonstrate that GraphRAG provides a concrete contribution across all these aspects.

Below are the observed outcomes for each evaluation metric and their implications for the proposed system:

- **Context Precision** (0.57 for both) shows that GraphRAG, despite extending the context with entities and relationships, maintains a quality comparable to pure semantic search, without introducing noise. This is important because it means that adding structured knowledge does not compromise the accuracy of initial selection.
- **Context Recall** is the metric where GraphRAG achieves the best result (+11%), thanks to the graph's ability to link elements even if not explicitly mentioned in the query. This has a direct impact on information coverage, which is crucial in scenarios where users expect a complete response capable of connecting events, activities, and people.
- **Answer Relevancy** (+2% for GraphRAG) highlights that semantic enrichment improves the alignment of the response with the user's intent. In an assistive system, where it is important to correctly understand requests and return immediate and contextualized answers, this is a key element.
- **Faithfulness** (+7%) is a clear strength: GraphRAG responses better adhere to the data actually present in the knowledge base. This aspect is central to ensuring system reliability, avoiding hallucinated content, and maintaining traceability of information, as required in sensitive applications such as support for people with cognitive impairment.
- **Answer Similarity** (+3%) suggests that GraphRAG responses, although more detailed, are semantically closer to those expected. This confirms the semantic consistency of the approach, even with different phrasings.

# Chapter 5

## Conclusions

The work described in this thesis addressed the problem of time management and daily activities for people with *Mild Cognitive Impairment* (MCI) by proposing a system that allows the user to keep track of his or her activities, appointments, projects, and habits through natural language interaction, and to subsequently query the system to reflect on his or her commitments, obtain clarification, and organize his or her routine, thus helping to compensate for difficulties related to cognitive impairment and to promote greater autonomy in managing daily life. The information provided by the user is processed and stored within a graph of personal knowledge, dynamically constructed according to an ontological scheme that ensures its semantic coherence, interrogability and evolution over time. Through the GraphRAG approach, the system is able to answer questions posed in natural language by retrieving relevant data from the graph, interpreting the connections and generating complete, personalized and verifiable answers. This mechanism enables not only the storage and reorganization of information in a structured form, but also enables inference and traceability processes, which are essential to ensure reliability and usefulness in the answers.

### 5.1 Final considerations

The results obtained confirm that the adoption of a knowledge graph approach, integrated with linguistic models through the GraphRAG paradigm, is an effective solution to address the problem of time management and daily activities in individuals with Mild Cognitive Impairment (MCI). The comparative analysis between GraphRAG and a traditional RAG system highlighted the added value introduced by the semantic and structured component of the knowledge graph.

In particular, the evaluation metrics show significant improvements in information coverage (*Context Recall*) and answer fidelity with respect to context (*Faithfulness*), which are central to an assistive system in which the user must be able to rely on answers that are correct, verifiable, and adherent to his or her experience. *Answer Relevancy* and *Semantic Similarity* are also found to be improved, indicating a greater ability of the system to understand the question and return centered answers.

The use of the knowledge graph makes it possible to connect items that are distant from each other in time and semantic space, offering more articulate and contextualized answers. This approach has proven particularly useful in questions that require the connection of scattered information, such as those about project-related activities or planned events at a specific location.

However, it is precisely the richness of information that can be a critical issue: in some situations, the system tends to include unsolicited details, weighing down the answer or moving away from the expected wording. This aspect highlights the need to refine the generation mechanisms, such as through automatic synthesis techniques or filter modules, to ensure a balance between semantic completeness and relevance.

On an engineering level, the adoption of a semantic infrastructure involves higher costs and complexity than solutions based solely on vector search. Building and updating the graph requires computational resources, attention to ontological consistency, and accurate semantic extraction modules. Indeed, the overall quality of the system is affected by the ability to correctly interpret textual input and populate the graph in a consistent manner.

Despite these challenges, the work demonstrates that the integration of knowledge graphs, ontologies, and natural language processing is a promising direction for the development of personalized assistive systems. The proposed architecture, while designed for the Time Assistive Technologies domain, has modularity and adaptability features that make it extensible to other application domains. Contexts such as digital health, personalized education, decision support, and knowledge management in complex environments such as the enterprise can ben-

efit from the same combination of structured representation, information retrieval and tracking, and conversational interface.

## **5.2 Future improvements**

The designed system provides a solid foundation for assisted management of time and daily activities through the use of a structured, natural language queryable knowledge base. However, several aspects remain that can be further developed to improve its accuracy, robustness, and adaptability in real-world settings. Some possible directions for improvement are outlined below.

### **Ontology expansion**

The ontology currently used is designed to cover a limited set of concepts related to time, daily activities, and the basic structure of user routines. In the future, it could be expanded to represent a broader domain, including new types of entities (e.g., emotional states, needs, environmental contexts) and more nuanced relationships. This would not only allow more items to be recognized during extraction, but also enable more sophisticated semantic inferences, increasing the system's ability to answer complex questions and anticipate latent user needs.

### **Controlling semantic coverage in extraction**

In the graph construction phase, an important improvement concerns the introduction of an automatic verification module that checks that each relevant element of the textual input has indeed been recognized, typed, and stored correctly. This mechanism would help detect any omissions in the extraction pipeline and ensure more complete semantic coverage, reducing information loss during the natural language-to-graph transformation.

### **Improved entity resolution**

Currently, entities that refer to the same reality but are expressed differently can be treated as distinct elements. A more advanced resolver could correctly identify these multiple occurrences (e.g., "Dr. Rossi" and "Doctor Rossi") and merge them into a single node, improving the quality and consistency of the knowledge base. This would also lead to more accurate queries and reduced redundancy within the graph.

### **Cypher retriever and query optimization**

The retrieval module can be improved through dynamic customization of the Cypher query based on the query asked. In particular, queries can be constructed that select only semantically relevant nodes and relationships, automatically filtering out unnecessary elements. This approach would lead to a reduction of noise in the retrieved context, improving the relevance and accuracy of the generated answers.

### **Automatic question classification**

Another possible extension involves the introduction of a module for automatic question classification. Identifying the type of the query (e.g., future appointment, past activity, recurring activity, queries related to a person or place) would allow customizing the behavior of the system in the retrieval and generation phase. For example, some queries might require timely and concise answers, others an articulated summary of related events. Recognizing the type of question from the outset would enable optimization of subsequent steps, improving relevance, clarity, and usefulness of the final response.

### **Traceability of sources in the answer**

The ability to trace the source of the information used in the response is a further improvement, both in terms of perceived reliability for the user and as a tool for internal verification within the system. Currently, the response is generated in a manner consistent with the data in the graph, but does not make its references explicit. Introducing a mechanism that highlights the nodes and relationships from which information is drawn would help increase the transparency of the system and facilitate content validation.

### **Design of an end-user interface**

The current interface is designed as a functional prototype for experimentation. A further step could be the design of an accessible, end-user-oriented interface. This includes introducing speech features (recognition and synthesis), using intuitive graphical displays to represent tasks and projects, and adopting accessibility guidelines to ensure simple, customizable, and barrier-free interaction.

# List of Figures

2.1	Example: nodes in an e-commerce graph . . . . .	27
2.2	Example: relationships in an e-commerce graph. . . . .	27
2.3	Example: classification schema in an e-commerce graph. . . . .	28
2.4	Taxonomy of Knowledge Graph Construction dimensions. . . . .	30
2.5	Comparison between Direct LLM, RAG, and GraphRAG . . . . .	46
2.6	GraphRAG system architecture: indexing, retrieval, and generation.	49
3.1	Digital Diary: input, semantic processing, and querying . . . . .	56
3.2	Logical architecture of the proposed system . . . . .	57
3.3	User Interface Prototype . . . . .	59
3.4	User Interface: Add Information . . . . .	60
3.5	User Interface: Ask Information . . . . .	61
3.6	Conceptual structure of TAMOntology in UML notation . . . . .	64
3.7	Ontology metadata and definition preview in Protégé . . . . .	66
3.8	Knowledge Graph Construction Pipeline . . . . .	69
3.9	GraphRAG architecture: context retrieval and answer generation	75
3.10	HybridCypherRetriever phases . . . . .	76
4.1	RAGAS score . . . . .	88
4.2	Context Precision schema . . . . .	89
4.3	Context recall schema . . . . .	91
4.4	Faithfulness schema . . . . .	92
4.5	Answer Relevancy schema . . . . .	94
4.6	Answer Similarity schema . . . . .	96
4.7	KG created from the first user interaction . . . . .	105
4.8	Example of properties associated to project node . . . . .	105
4.9	Graph generated from second user interaction . . . . .	106
4.10	Domain graph resulting from processing all 30 interactions . . . . .	107
4.11	Comparison of average metric values between RAG and GraphRAG	109

# List of Tables

2.1	Selected definitions of Knowledge Graphs from the literature . . . . .	21
2.2	Comparison of representative open knowledge graphs. . . . .	40
2.3	Examples of Domain-specific Knowledge Graphs (DSKGs) across different application domains. . . . .	42
3.1	Main conceptual classes defined in TAMOntology . . . . .	65
3.2	Datatype properties associated with each main class in TAMOntology . . . . .	65
3.3	Semantic relationships (ObjectProperty) defined in TAMOntology	66
4.1	Example of user interaction dataset for KG population. . . . .	103
4.2	Examples of questions and expected answers in the test dataset. .	108
4.3	Metric comparison on individual user profiles: GraphRAG vs. RAG111	

# Bibliography

- [1] National Institute of Neurological Disorders and Stroke. *Dementias: Information Page*. 2024. URL: <https://www.ninds.nih.gov/health-information/disorders/dementias>.
- [2] Sarah C. Maaß et al. “Timing deficiencies in amnestic Mild Cognitive Impairment: Disentangling clock and memory processes”. In: *Behavioural Brain Research* 373 (2019), p. 112110. ISSN: 0166-4328. DOI: <https://doi.org/10.1016/j.bbr.2019.112110>. URL: <https://www.sciencedirect.com/science/article/pii/S0166432819304954>.
- [3] Dan Zhu, Abdullah Al Mahmud, and Weijia Liu. “Examining behaviour change techniques (BCTs) in technology-based interventions for enhancing social participation in people with mild cognitive impairment (MCI) or dementia: a scoping review protocol”. In: *BMJ Open* 14 (2024), e078188. DOI: [10.1136/bmjopen-2023-078188](https://doi.org/10.1136/bmjopen-2023-078188). URL: <https://bmjopen.bmjjournals.com/content/14/1/e078188>.
- [4] Alex Gillespie, Catherine Best, and Brian O’Neill. “Cognitive Function and Assistive Technology for Cognition: A Systematic Review”. In: *Journal of the International Neuropsychological Society : JINS* 18 (Dec. 2011), pp. 1–19. DOI: [10.1017/S1355617711001548](https://doi.org/10.1017/S1355617711001548).
- [5] K. Yousaf et al. “A comprehensive study of mobile-health based assistive technology for the healthcare of dementia and Alzheimer’s disease (AD)”. In: *Health Care Management Science* 23.2 (2020). Epub 2019 Jun 20, pp. 287–309. DOI: [10.1007/s10729-019-09486-0](https://doi.org/10.1007/s10729-019-09486-0).
- [6] Eleonore Bayen et al. “Reduction in Fall Rate in Dementia Managed Care Through Video Incident Review: Pilot Study”. In: *Journal of Medical Internet Research* 19.10 (2017), e339. DOI: [10.2196/jmir.8095](https://doi.org/10.2196/jmir.8095). URL: <https://www.jmir.org/2017/10/e339>.

## BIBLIOGRAPHY

---

- [7] Mustafa Atee, Kreshnik Hoti, and Jeffery D. Hughes. “A technical note on the PainChek™ system: a web portal and mobile medical device for assessing pain in people with dementia”. In: *Frontiers in Aging Neuroscience* 10 (2018), p. 117. DOI: [10.3389/fnagi.2018.00117](https://doi.org/10.3389/fnagi.2018.00117). URL: <https://doi.org/10.3389/fnagi.2018.00117>.
- [8] Nicolas Farina et al. “Acceptability and feasibility of wearing activity monitors in community-dwelling older adults with dementia”. In: *International Journal of Geriatric Psychiatry* 34.4 (2019), pp. 617–624. DOI: [10.1002/gps.5064](https://doi.org/10.1002/gps.5064). URL: <https://doi.org/10.1002/gps.5064>.
- [9] Shahriar Nirjon et al. “MOBI-COG: A Mobile Application for Instant Screening of Dementia Using the Mini-Cog Test”. In: *Proceedings of the Wireless Health 2014 on National Institutes of Health*. ACM, 2014. DOI: [10.1145/2668883.2668889](https://doi.org/10.1145/2668883.2668889). URL: <https://doi.org/10.1145/2668883.2668889>.
- [10] Shireen Sindi et al. “The CAIDE dementia risk score app: the development of an evidence-based mobile application to predict the risk of dementia”. In: *Alzheimer's & Dementia: Diagnosis, Assessment & Disease Monitoring* 1.3 (2015), pp. 328–333. DOI: [10.1016/j.dadm.2015.06.005](https://doi.org/10.1016/j.dadm.2015.06.005). URL: <https://doi.org/10.1016/j.dadm.2015.06.005>.
- [11] Maurice Mulvenna et al. “Behavioural Usage Analysis of a Reminiscing App for People Living with Dementia and their Carers”. In: *Proceedings of the European Conference on Cognitive Ergonomics 2017*. ACM, 2017. DOI: [10.1145/3121283.3121289](https://doi.org/10.1145/3121283.3121289). URL: <https://doi.org/10.1145/3121283.3121289>.
- [12] Holger Megges et al. “Technology for home dementia care: a prototype locating system put to the test”. In: *Alzheimer's & Dementia: Translational Research & Clinical Interventions* 3.3 (2017), pp. 332–338. DOI: [10.1016/j.trci.2017.04.004](https://doi.org/10.1016/j.trci.2017.04.004). URL: <https://doi.org/10.1016/j.trci.2017.04.004>.
- [13] Hyejin Park and Jiwon Ha. “Effect of digital technology interventions for cognitive function improvement in mild cognitive impairment and dementia: A systematic review and meta-analysis”. In: *Research in Nursing & Health* 47.4 (Aug. 2024). Epub 2024 Apr 3, pp. 409–422. DOI: [10.1002/nur.22383](https://doi.org/10.1002/nur.22383). URL: <https://doi.org/10.1002/nur.22383>.

## BIBLIOGRAPHY

---

- [14] Yasushi Nishiura et al. “Effectiveness of using assistive technology for time orientation and memory, in older adults with or without dementia”. In: *Disability and Rehabilitation: Assistive Technology* 16.5 (July 2021). Epub 2019 Aug 19, pp. 472–478. DOI: [10.1080/17483107.2019.1650299](https://doi.org/10.1080/17483107.2019.1650299). URL: <https://doi.org/10.1080/17483107.2019.1650299>.
- [15] Dsouza et al. “Assistive devices to support daily time management of persons with dementia: the Indian experience”. In: *Alzheimer’s Dementia* 20 (Jan. 2025). DOI: [10.1002/alz.095235](https://doi.org/10.1002/alz.095235).
- [16] Baudin et al. “Informal carers’ experiences in everyday life and the use of digital assistive technology for time management in persons with dementia or mild cognitive impairment”. In: *BMC Geriatrics* 24 (Apr. 2024). DOI: [10.1186/s12877-024-04979-2](https://doi.org/10.1186/s12877-024-04979-2).
- [17] Björg Thordardottir et al. “Acceptance and Use of Innovative Assistive Technologies among People with Cognitive Impairment and Their Caregivers: A Systematic Review”. In: *BioMed Research International* 2019 (2019), p. 9196729. DOI: [10.1155/2019/9196729](https://doi.org/10.1155/2019/9196729). URL: <https://doi.org/10.1155/2019/9196729>.
- [18] Joshi et al. “AiZi Assist Application – A Tool for Alzheimer Disease Patients”. In: *Sirjana Journal* 54.3 (2024). ISSN: 2455-1058, pp. 314–319.
- [19] Lifei Zheng, Yeonie Heo, and Yi Fang. *MemoryCompanion: A Smart Health-care Solution to Empower Efficient Alzheimer’s Care Via Unleashing Generative AI*. 2023. arXiv: [2311.14730 \[cs.CL\]](https://arxiv.org/abs/2311.14730). URL: <https://arxiv.org/abs/2311.14730>.
- [20] Nima Nasiri. “A Language-Model-Based Chatbot that Considers the User’s Personality Profile and Emotions to Support Caregivers of People with Dementia”. Master’s thesis. Provo, Utah: Brigham Young University, 2023. URL: <https://scholarsarchive.byu.edu/etd/9862/>.
- [21] Wordh Hasan et al. “CareCompanion: A Personalized Virtual Assistant for Enhancing Support and Independence in ADRD Patients and Older Adults”. In: May 2024, pp. 1–10. DOI: [10.1109/SmartNets61466.2024.10577734](https://doi.org/10.1109/SmartNets61466.2024.10577734).
- [22] Thomas Gruber. “A Translational Approach to Portable Ontologies”. In: *Knowledge Acquisition* 5 (June 1993), pp. 199–220. DOI: [10.1006/knac.1993.1008](https://doi.org/10.1006/knac.1993.1008).

## BIBLIOGRAPHY

---

- [23] Thabet Slimani. “A Study Investigating Typical Concepts and Guidelines for Ontology Building”. In: (Sept. 2015).
- [24] John F. Sowa. “Ontologies for Knowledge Sharing”. In: *Manuscript of the invited talk at Terminology and Knowledge Engineering Congress (TKE '96)*. Manuscript. Vienna, 1996.
- [25] Nicola Guarino. “Formal Ontologies and Information Systems”. In: *Proceedings of FOIS'98*. Ed. by I. P. Amsterdam. Trento, Italy, 1998, pp. 3–15.
- [26] Robert Neches et al. “Enabling technology for knowledge sharing”. In: *AI Magazine* 12.3 (1991), pp. 16–36.
- [27] Bob Wielinga and A. Th. Schreiber. “Reusable and Shareable Knowledge Bases: A European Perspective”. In: *Proceedings of the International Conference on Building and Sharing of Very Large-Scale Knowledge Bases (KB&KS'93)*. Ed. by K. Fuchi. Tokyo, Japan, Dec. 1, 1993, pp. 103–115.
- [28] Tim Berners-Lee, James Hendler, and Ora Lassila. “The Semantic Web: A New Form of Web Content That is Meaningful to Computers Will Unleash a Revolution of New Possibilities”. In: *ScientificAmerican.com* (May 2001).
- [29] Li Ding et al. “Using Ontologies in the Semantic Web: A Survey”. In: *Ontologies: A Handbook of Principles, Concepts and Applications in Information Systems*. Ed. by Raj Sharman, Rajiv Kishore, and Ram Ramesh. Boston, MA: Springer US, 2007, pp. 79–113. ISBN: 978-0-387-37022-4. DOI: [10.1007/978-0-387-37022-4\\_4](https://doi.org/10.1007/978-0-387-37022-4_4). URL: [https://doi.org/10.1007/978-0-387-37022-4\\_4](https://doi.org/10.1007/978-0-387-37022-4_4).
- [30] S. Staab and M. Maedche. *Axioms are Objects, too — Ontology Engineering beyond the Modeling of Concepts and Relations*. Technical Report 400. Institute AIFB, Karlsruhe University, 2000.
- [31] M. Uschold and M. Gruninger. “Ontologies: Principles, Methods, and Applications”. In: *Knowledge Engineering Review* 11.2 (1996), pp. 93–155.
- [32] Marcia Lei Zeng and Lois Mai Chan. “Trends and Issues in Establishing Interoperability Among Knowledge Organization Systems”. In: *Journal of the American Society for Information Science and Technology* 59.3 (2008), pp. 378–390. DOI: [10.1002/asi.20746](https://doi.org/10.1002/asi.20746). URL: <https://doi.org/10.1002/asi.20746>.

## BIBLIOGRAPHY

---

- [33] W3C. *Resource Description Framework (RDF)*. 2014. URL: <https://www.w3.org/RDF/>.
- [34] W3C. *RDF Schema 1.1*. 2014. URL: <https://www.w3.org/TR/rdf-schema/>.
- [35] W3C. *OWL 2 Web Ontology Language: Document Overview*. 2012. URL: <https://www.w3.org/OWL/>.
- [36] W3C. *SKOS Simple Knowledge Organization System Reference*. 2009. URL: <https://www.w3.org/TR/skos-reference/>.
- [37] Lisa Ehrlinger and Wolfram Wöß. “Towards a Definition of Knowledge Graphs”. In: Sept. 2016.
- [38] Heiko Paulheim. “Knowledge Graph Refinement: A Survey of Approaches and Evaluation Methods”. In: *Semantic Web* (2016), pp. 1–20. URL: <https://www.semantic-web-journal.net/system/files/swj1222.pdf>.
- [39] Matthias Kroetsch and Gerhard Weikum. *Journal of Web Semantics: Special Issue on Knowledge Graphs*. <http://www.websemanticsjournal.org/index.php/ps/announcement/view/19>. Aug. 2016.
- [40] Andreas Blumauer. *From Taxonomies over Ontologies to Knowledge Graphs*. <https://blog.semantic-web.at/2014/07/15/from-taxonomies-over-ontologies-to-knowledge-graphs>. Accessed: 2016-08. July 2014.
- [41] Michael Färber et al. “Linked Data Quality of DBpedia, Freebase, OpenCyc, Wikidata, and YAGO”. In: *Semantic Web Journal* (2016). Revised version, under review. URL: <http://www.semantic-web-journal.net/content/linked-data-quality-dbpedia-freebase-opencyc-wikidata-and-yago>.
- [42] Jay Pujara et al. “Knowledge Graph Identification”. In: *Proceedings of the 12th International Semantic Web Conference - Part I (ISWC '13)*. New York, USA: Springer, 2013, pp. 542–557.
- [43] Bilal Abu-Salih. *Domain-specific Knowledge Graphs: A survey*. 2021. arXiv: 2011.00235 [cs.DB]. URL: <https://arxiv.org/abs/2011.00235>.
- [44] Google Inc. *Introducing the Knowledge Graph: things, not strings*. 2012. URL: <https://blog.google/products/search/introducing-knowledge-graph-things-not/>.

- [45] Marvin Hofer et al. *Construction of Knowledge Graphs: State and Challenges*. Feb. 2023. DOI: [10.2139/ssrn.4605059](https://doi.org/10.2139/ssrn.4605059).
- [46] Renzo Angles. “The Property Graph Database Model”. In: *Proceedings of the 12th Alberto Mendelzon International Workshop on Foundations of Data Management (AMW 2018)*. 2018.
- [47] Nicole Francis et al. “Cypher: An Evolving Query Language for Property Graphs”. In: *Proceedings of the 2018 International Conference on Management of Data (SIGMOD 2018)*. Ed. by Gautam Das, Christopher M. Jermaine, and Philip A. Bernstein. Houston, TX, USA: ACM, 2018, pp. 1433–1445. DOI: [10.1145/3183713.3190657](https://doi.org/10.1145/3183713.3190657). URL: <https://doi.org/10.1145/3183713.3190657>.
- [48] Marko A. Rodriguez. “The Gremlin Graph Traversal Machine and Language (Invited Talk)”. In: *Proceedings of the 15th Symposium on Database Programming Languages*. ACM, 2015. DOI: [10.1145/2815072.2815073](https://doi.org/10.1145/2815072.2815073). URL: <https://doi.org/10.1145/2815072.2815073>.
- [49] Oscar van Rest et al. “PGQL: A Property Graph Query Language”. In: *Proceedings of the Fourth International Workshop on Graph Data Management Experiences and Systems (GRADES)*. Ed. by Peter A. Boncz and Josep Larriba-Pey. Redwood Shores, CA, USA: ACM, 2016, p. 7. DOI: [10.1145/2960414.2960421](https://doi.org/10.1145/2960414.2960421). URL: <https://doi.org/10.1145/2960414.2960421>.
- [50] Renzo Angles et al. “G-CORE: A Core for Future Graph Query Languages”. In: *Proceedings of the 2018 International Conference on Management of Data (SIGMOD 2018)*. Ed. by Gautam Das, Christopher M. Jermaine, and Philip A. Bernstein. Houston, TX, USA: ACM, 2018, pp. 1421–1432. DOI: [10.1145/3183713.3190654](https://doi.org/10.1145/3183713.3190654). URL: <https://doi.org/10.1145/3183713.3190654>.
- [51] Neo4j. *What is a Knowledge Graph?* 2024. URL: <https://neo4j.com/blog/knowledge-graph/what-is-knowledge-graph/>.
- [52] Tareq Al-Moslmi et al. “Named Entity Extraction for Knowledge Graphs: A Literature Overview”. In: *IEEE Access* 8 (2020), pp. 32862–32881.
- [53] Arijit Khan. *Knowledge Graphs Querying*. 2023. arXiv: [2305.14485 \[cs.DB\]](https://arxiv.org/abs/2305.14485). URL: <https://arxiv.org/abs/2305.14485>.

- [54] Jing Yang, Weijie Yao, and Wen Zhang. “Keyword search on large graphs: a survey”. In: *Data Science and Engineering* 6.2 (2021), pp. 142–162.
- [55] Davide Mottin et al. “Exemplar Queries: Give Me an Example of What You Need”. In: *Proceedings of the VLDB Endowment (VLDB)* 7.5 (2014), pp. 365–376.
- [56] Nandish Jayaram et al. “Querying Knowledge Graphs by Example Entity Tuples”. In: *IEEE Transactions on Knowledge and Data Engineering* 27.10 (2015), pp. 2797–2811.
- [57] Yannis Tzitzikas, Nikos Manolis, and Panos Papadakos. “Faceted Exploration of RDF/S Datasets: A Survey”. In: *Journal of Intelligent Information Systems* 48.2 (2017), pp. 329–364.
- [58] Sourav S. Bhowmick and Bongki Choi. “Data-Driven Visual Query Interfaces for Graphs: Past, Present, and (Near) Future”. In: *Proceedings of the 2022 ACM SIGMOD International Conference on Management of Data*. ACM, 2022.
- [59] Lushan Han, Tim Finin, and Anupam Joshi. “Gorelations: An Intuitive Query System for DBpedia”. In: *Proceedings of the International Joint Conference on Semantic Technology (JIST)*. 2011.
- [60] Niranjan Chakraborty et al. “Introduction to Neural Network-Based Question Answering over Knowledge Graphs”. In: *WIREs Data Mining and Knowledge Discovery* 11.3 (2021). DOI: [10.1002/widm.1395](https://doi.org/10.1002/widm.1395).
- [61] Hao Zhou et al. “Commonsense Knowledge Aware Conversation Generation with Graph Attention”. In: *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*. International Joint Conferences on Artificial Intelligence Organization, 2018, pp. 4623–4629.
- [62] Veton Kepuska and Gamal Bohouta. “Next-Generation of Virtual Personal Assistants (Microsoft Cortana, Apple Siri, Amazon Alexa and Google Home)”. In: *Proceedings of the IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, 2018, pp. 99–103. DOI: [10.1109/CCWC.2018.8301638](https://doi.org/10.1109/CCWC.2018.8301638).
- [63] Ruobing Wang et al. “Knowledge Graph Embedding via Graph Attenuated Attention Networks”. In: *IEEE Access* 8 (2019), pp. 5212–5224. DOI: [10.1109/ACCESS.2019.2962239](https://doi.org/10.1109/ACCESS.2019.2962239).

## BIBLIOGRAPHY

---

- [64] Rose Catherine and William Cohen. “Personalized Recommendations Using Knowledge Graphs: A Probabilistic Logic Programming Approach”. In: *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys)*. ACM, 2016, pp. 325–332.
- [65] Shimaa Elnagar and Hugh R. Weistroffer. “Introducing Knowledge Graphs to Decision Support Systems Design”. In: *Proceedings of the EuroSymposium on Systems Analysis and Design*. Springer, 2019.
- [66] An Yang et al. “Enhancing Pre-trained Language Representations with Rich Knowledge for Machine Reading Comprehension”. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*. 2019, pp. 2346–2357.
- [67] Daniel S. Himmelstein et al. “Systematic integration of biomedical knowledge prioritizes drugs for repurposing”. In: *eLife* 6 (2017), e26726. DOI: [10.7554/eLife.26726](https://doi.org/10.7554/eLife.26726).
- [68] He Xing et al. “RDBridge: a knowledge graph of rare diseases based on large-scale text mining”. In: *Bioinformatics* 39.7 (2023), btad440. DOI: [10.1093/bioinformatics/btad440](https://doi.org/10.1093/bioinformatics/btad440).
- [69] Prerana Sanjay Kulkarni et al. *HeCiX: Integrating Knowledge Graphs and Large Language Models for Biomedical Research*. 2024. arXiv: [2407.14030](https://arxiv.org/abs/2407.14030) [cs.CL]. URL: <https://arxiv.org/abs/2407.14030>.
- [70] Penghe Chen et al. “KnowEdu: A System to Construct Knowledge Graph for Education”. In: *IEEE Access* PP (May 2018), pp. 1–1. DOI: [10.1109/ACCESS.2018.2839607](https://doi.org/10.1109/ACCESS.2018.2839607).
- [71] Bowen Zhao et al. *EDUKG: a Heterogeneous Sustainable K-12 Educational Knowledge Graph*. 2022. arXiv: [2210.12228](https://arxiv.org/abs/2210.12228) [cs.CL]. URL: <https://arxiv.org/abs/2210.12228>.
- [72] Xiaohui Victor Li. *FinDKG: A Financial Domain Knowledge Graph*. <https://github.com/xiaohui-victor-li/FinDKG>. Accessed: 2024-04-30. 2021.
- [73] Zhipeng Xu and Ryutaro Ichise. “FinCaKG-Onto: the financial expertise depiction via causality knowledge graph and domain ontology”. In: *Applied Intelligence* 55 (2025), p. 461. DOI: [10.1007/s10489-025-06247-1](https://doi.org/10.1007/s10489-025-06247-1).

## BIBLIOGRAPHY

---

- [74] Xiaohui Victor Li and collaborators. *FinKG-JP: Knowledge Graph Construction for Financial Applications in Japan*. <https://confit.atlas.jp/guide/event/jsai2023/subject/1U4-IS-1a-03/detail>. Presented at the JSAI 2023 Conference. 2023.
- [75] Sudip Mittal, Anupam Joshi, and Tim Finin. *Cyber-All-Intel: An AI for Security related Threat Intelligence*. 2019. arXiv: [1905.02895 \[cs.AI\]](https://arxiv.org/abs/1905.02895). URL: <https://arxiv.org/abs/1905.02895>.
- [76] Unified Cyber Ontology (UCO) Working Group. *Unified Cyber Ontology (UCO)*. <https://unifiedcyberontology.org/>. Accessed: 2024-04-30. 2024.
- [77] Erdogan Dogdu and collaborators. *UCKG: Unified Cybersecurity Knowledge Graph*. <https://github.com/edogdu/UCKG>. Accessed: 2024-04-30. 2020.
- [78] Tinatin Osmonova, Alexey Tikhonov, and Ivan P. Yamshchikov. *Knowledge Graph Representation for Political Information Sources*. 2024. arXiv: [2404.03437 \[cs.CL\]](https://arxiv.org/abs/2404.03437). URL: <https://arxiv.org/abs/2404.03437>.
- [79] Wei Chen et al. “Opinion-aware Knowledge Graph for Political Ideology Detection”. In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*. 2017, pp. 3647–3653. DOI: [10.24963/ijcai.2017/510](https://doi.org/10.24963/ijcai.2017/510). URL: <https://doi.org/10.24963/ijcai.2017/510>.
- [80] Xinyi Mou et al. “Unifying Local and Global Knowledge: Empowering Large Language Models as Political Experts with Knowledge Graphs”. In: *Proceedings of the ACM Web Conference 2024*. WWW ’24. Singapore, Singapore: Association for Computing Machinery, 2024, pp. 2603–2614. ISBN: 9798400701719. DOI: [10.1145/3589334.3645616](https://doi.org/10.1145/3589334.3645616). URL: <https://doi.org/10.1145/3589334.3645616>.
- [81] Sihan Song et al. “TravelRAG: A Tourist Attraction Retrieval Framework Based on Multi-Layer Knowledge Graph”. In: *ISPRS International Journal of Geo-Information* 13.11 (2024). ISSN: 2220-9964. DOI: [10.3390/ijgi13110414](https://doi.org/10.3390/ijgi13110414). URL: <https://www.mdpi.com/2220-9964/13/11/414>.
- [82] Qi Zhang, Zongwei Ma, Peng Zhang, et al. “Mobility Knowledge Graph: Review and Its Application in Public Transport”. In: *Transportation* 52 (2025), pp. 1119–1145. DOI: [10.1007/s11116-023-10451-8](https://doi.org/10.1007/s11116-023-10451-8).

## BIBLIOGRAPHY

---

- [83] Dinghe Xiao et al. “A Practice of Tourism Knowledge Graph Construction based on Heterogeneous Information”. eng. In: *Proceedings of the 19th Chinese National Conference on Computational Linguistics*. Ed. by Maosong Sun et al. Haikou, China: Chinese Information Processing Society of China, Oct. 2020, pp. 939–949. URL: <https://aclanthology.org/2020.ccl-1.87/>.
- [84] Wenqi Fan et al. “A Survey on RAG Meeting LLMs: Towards Retrieval-Augmented Large Language Models”. In: (2024). arXiv: [2405.06211 \[cs.CL\]](https://arxiv.org/abs/2405.06211). URL: <https://arxiv.org/abs/2405.06211>.
- [85] Yunfan Gao et al. “Retrieval-Augmented Generation for Large Language Models: A Survey”. In: (2024). arXiv: [2312.10997 \[cs.CL\]](https://arxiv.org/abs/2312.10997). URL: <https://arxiv.org/abs/2312.10997>.
- [86] Yucheng Hu and Yuxing Lu. “RAG and RAU: A Survey on Retrieval-Augmented Language Model in Natural Language Processing”. In: (2024). arXiv: [2404.19543 \[cs.CL\]](https://arxiv.org/abs/2404.19543). URL: <https://arxiv.org/abs/2404.19543>.
- [87] OpenAI. *GPT-4 Technical Report*. Tech. rep. OpenAI, 2024. arXiv: [2303.08774 \[cs.CL\]](https://arxiv.org/abs/2303.08774). URL: <https://arxiv.org/abs/2303.08774>.
- [88] Abhimanyu Dubey, Abhinav Jauhri, et al. “The Llama 3 Herd of Models”. In: (2024). arXiv: [2407.21783 \[cs.AI\]](https://arxiv.org/abs/2407.21783). URL: <https://arxiv.org/abs/2407.21783>.
- [89] An Yang, Baosong Yang, et al. “Qwen2 Technical Report”. In: (2024). arXiv: [2407.10671 \[cs.CL\]](https://arxiv.org/abs/2407.10671). URL: <https://arxiv.org/abs/2407.10671>.
- [90] Lei Huang et al. “A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions”. In: (2023). arXiv: [2311.05232 \[cs.CL\]](https://arxiv.org/abs/2311.05232). URL: <https://arxiv.org/abs/2311.05232>.
- [91] Darren Edge et al. *From Local to Global: A Graph RAG Approach to Query-Focused Summarization*. 2025. arXiv: [2404.16130 \[cs.CL\]](https://arxiv.org/abs/2404.16130). URL: <https://arxiv.org/abs/2404.16130>.
- [92] Yuntong Hu et al. “GRAG: Graph Retrieval-Augmented Generation”. In: (2024). arXiv: [2405.16506 \[cs.LG\]](https://arxiv.org/abs/2405.16506). URL: <https://arxiv.org/abs/2405.16506>.
- [93] Boci Peng et al. *Graph Retrieval-Augmented Generation: A Survey*. 2024. arXiv: [2408.08921 \[cs.AI\]](https://arxiv.org/abs/2408.08921). URL: <https://arxiv.org/abs/2408.08921>.

- [94] Costas Mavromatis and George Karypis. “GNN-RAG: Graph Neural Retrieval for Large Language Model Reasoning”. In: (2024). arXiv: [2405.20139 \[cs.CL\]](https://arxiv.org/abs/2405.20139). URL: <https://arxiv.org/abs/2405.20139>.
- [95] Data.world. *Generative AI Benchmark: Increasing the Accuracy of LLMs in the Enterprise with a Knowledge Graph*. 2023. URL: <https://data.world/blog/generative-ai-benchmark-increasing-the-accuracy-of-langs-in-the-enterprise-with-a-knowledge-graph/>.
- [96] Zhentao Xu et al. “Retrieval-Augmented Generation with Knowledge Graphs for Customer Service Question Answering”. In: *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’24. Washington DC, USA: ACM, 2024, pp. 2905–2909.
- [97] Wenqi Fan et al. “Graph Machine Learning in the Era of Large Language Models (LLMs)”. In: (2024). arXiv: [2404.14928 \[cs.LG\]](https://arxiv.org/abs/2404.14928). URL: <https://arxiv.org/abs/2404.14928>.
- [98] Bin Fu et al. “A Survey on Complex Question Answering over Knowledge Base: Recent Advances and Challenges”. In: (2020). arXiv: [2007.13069 \[cs.CL\]](https://arxiv.org/abs/2007.13069). URL: <https://arxiv.org/abs/2007.13069>.
- [99] Dawei Li et al. *DALK: Dynamic Co-Augmentation of LLMs and KG to answer Alzheimer’s Disease Questions with Scientific Literature*. 2024. arXiv: [2405.04819 \[cs.CL\]](https://arxiv.org/abs/2405.04819). URL: <https://arxiv.org/abs/2405.04819>.
- [100] Naoki Matsumoto et al. “KRAGEN: a knowledge graph-enhanced RAG framework for biomedical problem solving using large language models”. In: *Bioinformatics* 40.6 (2024), btae353. DOI: [10.1093/bioinformatics/btae353](https://doi.org/10.1093/bioinformatics/btae353).
- [101] Natalya F. Noy and Deborah L. McGuinness. *Ontology Development 101: A Guide to Creating Your First Ontology*. 2001. URL: [https://protege.stanford.edu/publications/ontology\\_development/ontology101-noy-mcguinness.html](https://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html).
- [102] Neo4j. *Going Meta: A Series on Graphs, Semantics, and Knowledge*. Video series. Accessed: 2025-05-10. 2024. URL: <https://neo4j.com/video/going-meta-a-series-on-graphs-semantics-and-knowledge/>.

## BIBLIOGRAPHY

---

- [103] Microsoft Research. *GraphRAG: Unlocking LLM Discovery on Narrative Private Data*. Accessed: 2025-05-10. 2024. URL: <https://www.microsoft.com/en-us/research/blog/graphrag-unlocking-llm-discovery-on-narrative-private-data/>.
- [104] Shahul Es et al. *Ragas: Automated Evaluation of Retrieval Augmented Generation*. 2025. arXiv: 2309.15217 [cs.CL]. URL: <https://arxiv.org/abs/2309.15217>.
- [105] Ahmet Yasin Aytar, Kemal Kilic, and Kamer Kaya. *A Retrieval-Augmented Generation Framework for Academic Literature Navigation in Data Science*. 2024. arXiv: 2412.15404 [cs.IR]. URL: <https://arxiv.org/abs/2412.15404>.
- [106] Bhaskarjit Sarmah et al. *HybridRAG: Integrating Knowledge Graphs and Vector Retrieval Augmented Generation for Efficient Information Extraction*. 2024. arXiv: 2408.04948 [cs.CL]. URL: <https://arxiv.org/abs/2408.04948>.
- [107] Kahyun Jeon and Ghang Lee. “Hybrid large language model approach for prompt and sensitive defect management: A comparative analysis of hybrid, non-hybrid, and GraphRAG approaches”. In: *Advanced Engineering Informatics* 64 (2025), p. 103076. ISSN: 1474-0346. DOI: <https://doi.org/10.1016/j.aei.2024.103076>. URL: <https://www.sciencedirect.com/science/article/pii/S1474034624007274>.
- [108] Neo4j. *Announcing the Release of LLM Knowledge Graph Builder*. Accessed: 2025-05-10. 2024. URL: <https://neo4j.com/blog/developer/llm-knowledge-graph-builder-release/>.
- [109] Kaarthick S. *RAGAS for RAG in LLMs: A Comprehensive Guide to Evaluation Metrics*. Accessed: 2025-05-10. 2024. URL: <https://dkaarthick.medium.com/ragas-for-rag-in-llms-a-comprehensive-guide-to-evaluation-metrics-3aca142d6e38>.