UNICA

UNIVERSITÀ
DEGLI STUDI
DI CAGLIARI

Ph.D. DEGREE IN
Mathematics and Computer Science

Cycle XXXVIII

TITLE OF THE Ph.D. THESIS

Knowledge Engineering via Large Language Models

Scientific Disciplinary Sector(s)

INFO-01/A

Ph.D. Student	Sandro Gabriele Tiddia
Supervisor	Prof. Salvatore Mario Carta
Co-Supervisor	Dr. Alessandro Sebastian Podda

Final exam. Academic Year 2024/2025
Thesis defence session: February 2026

Abstract

The digital transformation of society has led public institutions and private organizations to embrace digitization, resulting in an unprecedented production of textual documents. Much of this material remains unstructured and primarily intended for human reading and understanding. This thesis investigates how Large Language Models (LLMs) can act as knowledge engineers, transforming raw text into structured, reusable, and queryable information. The central question guiding this work is how far the understanding capabilities of LLMs can automate the organization and retrieval of knowledge traditionally curated by human experts.

The research explores two complementary strategies for structuring textual information, focusing on the construction of Knowledge Graphs (KGs): a schema-first approach, where LLMs infer type systems before populating them with entities, and an extraction-first approach, where entities and relations are identified freely and later organized into a type schema. Beyond structuring, the work also examines LLMs as interfaces for retrieval, evaluating their ability to translate natural language questions into KG queries (SPARQL). Finally, a domain application in healthcare demonstrates how ontology-driven KG modeling and LLM-based event extraction can integrate structured and unstructured electronic health record data.

Collectively, the contributions of this thesis highlight both the promise and the limitations of current LLMs as instruments of knowledge organization, underscoring the need for guided, multi-step, and hybrid human-AI workflows to achieve robust semantic structuring at scale.

Contents

Chapter 1

Introduction

1.1 Motivations

Over the past decades, the progressive digital transformation of society has reshaped how information is created, shared, and stored. Public administrations and private organizations now generate vast quantities of textual content, ranging from reports and manuals to legal acts and online publications. While these documents hold significant informational value, they are typically written for human interpretation. Their expression in natural language poses a challenge for computational processing, limiting the extent to which expert systems can programmatically leverage them for advanced tasks such as information retrieval, conditional search, or reasoning [154].

Within *Artificial Intelligence* (AI), *Natural Language Processing* (NLP) seek to address these challenges by developing computational methods to interpret human language. Recent advances in *Large Language Models* (LLMs) have greatly enhanced the ability to process raw text and perform traditional NLP tasks such as text summarization, sentiment analysis, information extraction, and machine translation [154]. At the same time, LLMs are increasingly extending their role beyond language processing itself, functioning as knowledge bases of factual information and as agents of reasoning, planning, creativity, and scientific discovery [1, 25, 82, 20].

Although powerful, LLMs must be approached with particular care when used as knowledge bases. The implicit and probabilistic way in which they encode knowledge limits the degree to which they can be fully trusted, leading to the possible generation of fabricated or nonfactual information, a phenomenon commonly referred to as hallucination [68, 25]. Recent research has explored how this harmful tendency can be mitigated through factual grounding. By providing LLMs with validated external sources, their factual accuracy can be improved, relying on the model's more robust interpretive abilities rather than on its implicitly encoded knowledge [66].

In this regard, *Retrieval-Augmented Generation* (RAG) has emerged as a practical framework to ground LLMs in reliable external knowledge sources [145, 126]. RAG combines retrieval-based methods with generative models, allowing the model

to access external information during inference rather than relying solely on its internal parameters. This architecture supports the integration of both unstructured corpora (e.g., Wikipedia) and structured repositories (e.g., Wikidata), thereby enhancing factual consistency and interpretability [145].

However, conventional RAG pipelines relying exclusively on unstructured text face inherent limitations. Text chunks retrieved from vector databases may lose fine-grained semantic relations or broader contextual dependencies (for instance, temporal or version-related aspects), leading to potential inaccuracies in the generated response [145]. Such issues are particularly critical when temporal, versioned, or relational information is involved. Structured sources, by contrast, provide explicit semantic organization and relational grounding, making them more dependable for storing and retrieving factual knowledge [126]. For this reason, combining structured and unstructured knowledge has proven especially effective for improving factual grounding and reducing hallucinations in RAG-based systems [145, 126].

Semantic Web technologies have long provided powerful means to represent and organize such structured information. *Knowledge Graphs* (KGs), in particular, capture it through entities with unique identities that correspond to domain concepts, represented as nodes, and through properties describing either attributes of these entities or relations between them, typically expressed as triples or, more generally, n-tuples [130]. KGs have demonstrated significant value across tasks such as recommender systems, question answering, and information retrieval, and they increasingly serve as a structured support for grounding LLMs [126, 130]. However, knowledge acquisition for KG construction remains a significant challenge. Large-scale resources can be created fully automatically only from semi-structured inputs (e.g., Wikidata derived from Wikipedia infoboxes), whereas high-quality expert sources describing domain-specific content often require extensive manual modeling effort (e.g., SNOMED in medicine) [130].

Recent research indicates that LLMs themselves can actively support this challenging structuring process. Owing to their semantic understanding and generative capabilities, they are able to identify and organize relevant concepts and relations directly from unstructured text, thereby facilitating the construction of structured knowledge bases that can subsequently serve as grounding resources for LLM applications [35, 170].

Beyond knowledge structuring, an equally important challenge lies in making structured knowledge accessible and actionable. Traditional query languages such as *SQL* for relational databases or *SPARQL* for KGs enable precise access to stored information, but they require expertise in both the query syntax and the underlying data model. This requirement creates a gap between the expressive potential of structured resources and their effective use in retrieval systems or by non-expert users. LLMs have demonstrated the ability to bridge this gap by translating natural language inputs into formal queries, thereby acting as intermediaries between users and structured data sources [149, 99].

Taken together, these developments highlight how the growing volume of un-

structured textual resources represents an immense yet underexploited source of knowledge. While LLMs can already leverage such data directly through retrieval-augmented approaches, their potential is considerably enhanced when information is expressed in structured, machine-interpretable form. At the same time, LLMs themselves can contribute to this structuring effort and facilitate access to the resulting knowledge through natural language interaction.

1.2 Contributions

Building on these motivations, this work investigates how LLMs can support the automation of knowledge engineering tasks, from the structuring of textual information into Knowledge Graphs to the retrieval of structured knowledge through natural-language interfaces. The contributions span three main dimensions.

Automated knowledge structuring. Two complementary strategies for structuring knowledge from text are investigated. A top-down (schema-first) approach introduces a zero-shot LLM-based pipeline for the unsupervised induction of entity schemas, enabling the derivation of candidate classes and hierarchies without human supervision. A bottom-up (extraction-first) approach proposes a document-level strategy for constructing Knowledge Graphs directly from text, without relying on a predefined schema, through LLM-driven triple extraction and enrichment. The two approaches are compared and further analyzed to explore the automated refinement of ontological hierarchies through LLM-based reasoning.

Knowledge retrieval through natural language. The problem of accessing structured knowledge through natural-language queries is examined. A systematic evaluation of multiple instruction-tuned LLMs is presented for the zero-shot translation of natural-language questions into SPARQL queries. A multi-dimensional evaluation encompassing accuracy, syntax correctness, determinism, and generation time is introduced, together with a taxonomy of common failure modes, providing a reproducible baseline for future research in LLM-based semantic retrieval.

Domain-specific application to electronic health records. LLM-driven extraction methods are applied to the healthcare domain to assess their practical applicability in a sensitive real-world setting. A pipeline is developed to integrate structured and unstructured electronic health record data into a unified, temporally coherent representation, combining an event-based ontology with LLM-based information extraction.

Collectively, these contributions advance the understanding of how LLMs can act as intermediaries between human language and formal knowledge representations.

They show that the same reasoning and generation capabilities enabling textual understanding can also support the construction, refinement, and querying of structured knowledge. By integrating zero-shot modeling, prompt-based reasoning, and ontology-driven design, this work outlines a coherent methodological framework for hybrid, semi-automated knowledge engineering across domains.

1.3 Synopsis

This dissertation combines both original and previously published material. Its chapters are organized to follow a coherent progression from methodological foundations to experimental validation and applied case studies.

Chapter 2 *Background* introduces the general background and theoretical framework that underpin the work. It reviews the fundamentals of Natural Language Processing, language models, information extraction, and Semantic Web technologies, and situates the thesis within the broader landscape of knowledge engineering research.

Chapter 3 *Knowledge Structuring* focuses on methods for the automated structuring of textual information using LLMs. The chapter includes and extends two previously published joint works [24, 23], incorporated with the consent of the co-authors, and presents additional unpublished material.

Chapter 4 *Knowledge Retrieval* examines the problem of accessing structured knowledge through natural-language queries. It builds upon collaborative work accepted for publication in *Neural Computing and Applications*, reproduced with the agreement of the co-authors, and provides an empirical analysis of LLM-based query translation.

Chapter 5 *Application* applies LLM-based information extraction methods to the healthcare domain, addressing the integration of structured and unstructured electronic health records. The chapter is informed by an ongoing collaborative study currently being finalized for submission and included with the consent of all contributors.

Chapter 6 *Conclusion* summarizes the main findings of this work, discusses its limitations and open challenges, and outlines potential directions for future research.

Chapter 2

Background

This chapter presents the theoretical and technological context of the research conducted in this thesis. It first surveys four core areas that provide a foundational basis for the work: *Natural Language Processing* in Section 2.1, *Language Models* in Section 2.2, *Information Extraction* in Section 2.3, and *Semantic Web and Knowledge Graphs* in Section 2.4. Each section introduces the fundamental principles, methods, and challenges of its respective field, highlighting how these areas converge within the broader scope of knowledge engineering. The final Section 2.5 (*Recent Directions in Knowledge Engineering*) surveys recent research in the literature that cuts across these foundational areas, providing an overview of prior work aligned with the three main research directions of the thesis: knowledge structuring, knowledge access, and applications to electronic health records.

2.1 Natural Language Processing

2.1.1 Definition and scope

Natural Language Processing (NLP) is a field of computer science concerned with enabling computers to systematically and computationally analyze, interpret, and generate human language. It encompasses a broad range of methods for representing and understanding linguistic data, such as text segmentation, syntactic and morphological analysis, semantic interpretation. Through these methods, NLP makes it possible for machines to handle natural language as both data and a medium of interaction, supporting applications from information extraction and translation to dialogue systems and knowledge retrieval [79, 42].

Two other terms frequently accompany discussions of NLP, and while they share substantial overlap, their historical and disciplinary orientations differ.

Computational Linguistics (CL) represents the scientific study of language through computational methods. It is rooted in linguistics and seeks to model the structure and principles of human language formally, using algorithms as tools to

test and refine linguistic theory. NLP, in contrast, is the engineering and algorithmic counterpart of this enterprise. NLP focuses on building systems that perform linguistic tasks, analyzing, transforming, or generating text, often drawing on CL insights but ultimately evaluated by empirical performance rather than theoretical adequacy. In this sense, NLP translates the descriptive ambitions of CL into operational technologies [42, 60, 33].

Natural Language Understanding (NLU) denotes the branch of NLP that addresses the problem of meaning, that is, how machines can comprehend language in a way that supports reasoning, inference, and decision-making. Whereas NLP encompasses the entire pipeline from raw linguistic input to output, including structural and statistical processing, NLU focuses on the semantic and pragmatic dimensions of interpretation, mapping words and sentences onto conceptual or logical representations that capture intent and context. It therefore represents the interpretive core of NLP, bridging linguistic form and knowledge-based reasoning [79, 42].

In this thesis, the research focus lies primarily within the NLP perspective, on methods that allow written language to be processed, structured, and made computationally actionable, yet it necessarily incorporates insights about language drawn from CL and NLU to provide a proper context.

2.1.2 Representing linguistic form and structure

Human language is a system of symbols governed by grammar, the set of principles that determine how expressions are formed and interpreted. In linguistics, grammar is commonly analyzed into several interconnected components: *phonetics* and *phonology*, which respectively examine how speech sounds are produced and their systematic patterning; *morphology*, which governs the internal structure of words; *syntax*, which organizes words into phrases and sentences; *semantics*, which assigns meaning to these structures; and *pragmatics*, which studies how context and use contribute to meaning [56, 4]. In the context of written language and our computational aims, three levels are central: *morphology* (surface form), *syntax* (structural organization), and *semantics* (conceptual meaning)[1]. For a computer to process text, each of these levels must be expressed in a machine-operable representation. Although discussed separately, they are inherently interdependent: words are conventional signs whose form is meaningful, and syntactic structure constrains how those meanings combine [4].

At the surface level, text is encoded as sequences of symbols that represent the linguistic units of interest. A basic approach considers words as the primary units, isolated through a process called *word segmentation* [76]. Words are the natural carriers of information for human communication, yet from a computational perspective they are not ideal for modeling meaning. Without providing a system

[1]For the purposes of this dissertation, an intuitive understanding of what *meaning* means is sufficient. I do not attempt to define it formally, but I defer to [69] for a foundational discussion.

with any knowledge of morphology, related forms such as *bank* and *banks*, or *run* and *ran*, would be treated as unrelated units. To mitigate this, early approaches applied simple normalization techniques such as *stemming*[2] to reduce *banks* to *bank*, and *lemmatization*[3], which also normalizes *ran* to *run*. However, such methods discard morphological information that can carry meaning [76, 33]. More recent approaches address this by exploiting the internal structure of words, segmenting them into smaller meaningful units closer to *morphemes*[4], referred to as *tokens*. This process, known as *tokenization*, may help preserve morphological distinctions, for example by producing *bank* and *s* from *banks*, and thereby retaining both the base concept and the morphological marker of plurality [76, 33].

Beyond the surface form, linguistic units are organized into hierarchical and relational structures. These structures are typically represented by assigning morphosyntactic categories to words (e.g., number, tense, case) and by identifying the syntactic relations among them (e.g., agreement, subordination, modification). Different processes are used to produce such annotations, such as part-of-speech (POS) tagging to mark nouns, verbs, and adjectives; constituency parsing to identify noun and verb phrases; and dependency parsing to capture relations like subject, object, or modifier. Such structured representations underlie many NLP tasks and also inform neural models, as they describe how words combine into phrases and sentences, bridging surface form and meaning [76, 118].

Finally, semantic representations operate at the highest level, moving beyond surface form and syntactic structure to model meaning. Representing semantics is widely regarded as one of the more challenging tasks in language processing, as even defining what meaning is proves non-trivial, yet it remains a crucial aspect of knowledge engineering. Symbolic approaches have traditionally represented knowledge and meaning through discrete symbols and explicit rules, whereas modern NLP systems increasingly rely on statistical and numerical representations, typically vectorial, that infer quantitative aspects of language use from data [18, 42].

Many statistical approaches build on the *distributional hypothesis*, a foundational idea stating that the meaning of a word can be inferred from the contexts in which it occurs [44, 18, 42]. From this observation, models such as the bag-of-words (BoW) were introduced to represent the meaning of entire documents. In this representation, a text is modeled as a vector of word counts, ignoring word order but capturing the presence and frequency of the terms appearing in it [18, 42]. Similarly, the meaning of individual words can be represented as vectors derived from counting their occurrences across documents or their co-occurrence with nearby words in sentences throughout a corpus. These explicit count-based representations make it possible to interpret the meaning of each vector dimension directly [18, 42].

However, such count vectors are often scaled (e.g., normalized) or transformed

[2]Reduction of inflected words to a common root through heuristic truncation.
[3]Mapping of words to their canonical dictionary form, known as the *lemma*.
[4]The smallest units of meaning within a word.

(e.g., projected into lower-dimensional spaces) to obtain more compact and abstract representations, known as *embeddings*, since the original spaces are typically high-dimensional and sparse. Approaches following this principle are referred to as frequency-based or count-based embeddings [18, 42].

Most modern methods, however, leverage neural networks to obtain embeddings without any explicit counting, directly mapping linguistic units such as tokens, words, phrases, or sentences to dense vectors in a continuous space. These representations are learned by optimizing specific objectives that encode desired semantic or contextual properties. They can, for instance, be trained so that semantically similar words have nearby vectors according to cosine similarity, or so that sentence embeddings capture broader contextual or retrieval-oriented relations [18, 42]. A common illustrative example of word-level regularities is

$$v(\text{king}) - v(\text{man}) + v(\text{woman}) \approx v(\text{queen}),$$

showing how certain embeddings can capture analogical relations directly from data, and that vectorial representations can be used to express relations through algebraic expressions [108]. This way of representing semantics now forms the foundation of contemporary neural NLP systems.

As previously pointed out, the levels discussed here are inherently interdependent. For example, representing the form as a sequence of words differs from representing it as tokens. Splitting *banks* into *bank* and *s* separates the lexical concept from the morphological feature of plurality, allowing a distributional representation of semantics built upon that symbolic system to have dedicated embeddings for each. Choices of representation thus shape which linguistic properties a model can encode and which aspects of meaning it is able to capture.

2.1.3 Core tasks

Encoding text into computational representations is typically the first step in any NLP pipeline, serving as the foundation for a broad range of tasks that address different aspects of linguistic understanding and production. Building on the preceding discussion of representations, this section outlines a set of core NLP tasks that are commonly used to characterize the scope of the field in practice. Tasks that are more foundational and central to the contributions of this thesis, such as *language modeling* and *information extraction*, receive a dedicated and more in-depth discussion in the following sections.

Text classification. Text classification consists of assigning predefined labels to textual inputs. Typical applications include topic categorization, spam detection, author identification, and sentiment analysis. The latter, a canonical example, aims to determine the affective polarity expressed in a piece of text, such as whether a review or social media post conveys a positive, negative, or neutral attitude. Such

tasks are inherently sensitive to context, subjectivity, and domain-specific language use [90].

Machine translation. Machine translation aims to automatically render text from one language into another while preserving meaning, fluency, and grammatical correctness. It requires handling linguistic divergence across languages, long-range dependencies, ambiguity, and pragmatic, cultural, and contextual factors. Its applications range from cross-lingual communication and information access to multilingual content generation [164].

Text summarization. Text summarization seeks to produce a concise representation of a longer document while preserving its key information and overall coherence. Extractive methods achieve this by selecting and combining segments from the original text, whereas abstractive methods generate new sentences that paraphrase the source content. Beyond compression, summarization requires identifying salient information, avoiding redundancy, and maintaining factual consistency with the input, making it sensitive to discourse structure and contextual interpretation [177].

Question answering. Question answering (QA) addresses the problem of providing explicit answers to natural language queries posed by users. QA systems may operate over structured or unstructured sources and can involve retrieving relevant information, inferring answers through reasoning, or combining both. Traditional QA settings typically assume isolated questions grounded in a given context, such as a document or a knowledge base. Conversational QA extends this formulation by introducing dependencies across multiple turns, requiring the system to maintain dialogue context, resolve references, and handle underspecified or implicit queries [15].

Coreference resolution. Coreference resolution seeks to identify when different expressions in a text refer to the same entity, as in linking *it* to *pizza* (and not *broccoli*) in "*I don't like broccoli, but I do like pizza. It is my favourite food*". By consolidating mentions that denote the same referent, it supports coherence modeling and serves as a crucial preprocessing step for downstream tasks such as information extraction, summarization, and question answering. The task encompasses both entity and event coreference and poses challenges related to ambiguity, long-range dependencies, and implicit semantic relations [96].

Taken together, these tasks exemplify the breadth of NLP. The approaches and techniques developed to address them follow a common historical trajectory, evolving from rule-based text manipulation toward data-driven and neural methods. Large Language Models represent the current culmination of this evolution, as they integrate within a single architecture the capability of modeling linguistic structure, meaning, and context [116, 79].

In this thesis, LLMs are used within the developed pipelines to handle subtasks such as classification, translation, summarization, and, implicitly, coreference resolution.

2.2 Language Models

2.2.1 Language modeling

Having examined how NLP enables computers to process linguistic input, we now turn to the question of how they *model* language itself. *Language modeling* is a foundational task in NLP [111], with the objective of capturing the statistical and structural regularities that govern how words[5] are arranged in natural language. The goal is to estimate the likelihood of linguistic sequences and, in particular, to predict the probability of each possible continuation given a preceding context [179, 42].

Formally, given a sequence of words $w_1, w_2, \ldots, w_{t-1}$, a language model estimates the conditional probability of the next word as

$$P(w_t \mid w_1, w_2, \ldots, w_{t-1}) \tag{2.1}$$

Here P is a probability distribution over the vocabulary V of all known words, mapping each $w_t \in V$ to its probability of occurring at position t. The probability of an entire sequence can thus be modeled using the chain rule:

$$P(w_1, w_2, \ldots, w_T) = \prod_{t=1}^{T} P(w_t \mid w_1, \ldots, w_{t-1}) \tag{2.2}$$

This formulation corresponds to the classical autoregressive setup, where each word is predicted based on all its predecessors [179, 42]. Alternative formulations exist; for example, bidirectional or masked models estimate probabilities of words given their surrounding context rather than only the preceding one, but the general aim remains to model the statistical dependencies that characterize natural language [76, 179].

A *language model* is thus a system that implements the language modeling task by estimating such conditional probability distributions over word sequences.

2.2.2 Approaches to language modeling

Early language models were based on explicit statistical estimation. The most common approach, the *n-gram model*, assumes that the probability of a word depends

[5]Throughout this section, I use the term *word*, as it more naturally represents the constituents of text and sentences. However, the probabilistic principles discussed here apply equally to other linguistic units, including the *tokens* on which most modern models operate.

only on the preceding $n-1$ words observed in a corpus [76, 148]. Under this *Markov assumption*, the conditional probability in Equation (2.1) is approximated as

$$P(w_t \mid w_1, \ldots, w_{t-1}) \approx P(w_t \mid w_{t-n+1}, \ldots, w_{t-1}) \tag{2.3}$$

and estimated directly from frequency counts:

$$P(w_t \mid w_{t-n+1}, \ldots, w_{t-1}) = \frac{\text{count}(w_{t-n+1}, \ldots, w_t)}{\text{count}(w_{t-n+1}, \ldots, w_{t-1})} \tag{2.4}$$

Intuitively, in a trigram model, the probability of a word such as *pizza* in the sequence *I like pizza* corresponds to how frequently that trigram appears relative to all occurrences of its prefix *I like*. To handle unseen or rare sequences, lower-order n-grams and various smoothing techniques are employed. Despite their simplicity, n-gram models face two fundamental limitations: data sparsity and restricted contextual scope, as the model's memory is bounded by the fixed window size $n-1$ [42]. These limitations motivated the transition toward distributed and neural approaches that learn continuous representations of words and can generalize beyond exact n-gram matches [12].

Neural language models address these limitations by parameterizing the conditional distribution in Equation (2.1) through a neural function [12], definable as

$$\mathcal{M}_\theta : (w_1, \ldots, w_{t-1}) \mapsto P_\theta(w_t \mid w_1, \ldots, w_{t-1}) \tag{2.5}$$

where $\mathcal{M}_\theta$ denotes a model with parameters θ that maps a sequence of preceding words to an estimated probability distribution P_θ over the vocabulary V. Specifically, the model outputs a vector $\pi_t = (P_\theta(w_t = v_1), \ldots, P_\theta(w_t = v_{|V|}))$ representing the predicted probabilities for each word $v_i \in V$[6].

Unlike frequency-based modeling, these probabilities are learned implicitly by optimizing the model parameters θ over large text corpora, enabling the model to generalize beyond observed sequences. Neural language modeling typically begins by mapping words to continuous distributed representations through an embedding layer, resulting in a sequence of embeddings that can be numerically processed by a neural architecture to model contextual dependencies [76, 12].

Early architectures employed *Recurrent Neural Networks* (RNNs), which maintain a hidden state summarizing the preceding context. At each time step, this hidden state is updated as a function of the current input and the previous state, creating a strictly sequential dependency across the input sequence [109]. While effective for short-range dependencies, RNNs suffer from vanishing gradients and limited capacity to capture long-distance relations, as information must be propagated through many successive state updates [129]. *Long Short-Term Memory* (LSTM) networks and *Gated Recurrent Units* (GRUs) improved this by introducing gated mechanisms that explicitly regulate the flow of information over time. In

[6]More specific formulations could be provided for particular neural architectures; this general form abstracts away implementation details for clarity.

LSTMs, this is achieved by maintaining a dedicated memory cell that is designed to preserve long-term information, alongside a transient hidden representation, with gates controlling what information is written, retained, or discarded. GRUs adopt a simpler formulation in which gating directly modulates the update of the hidden state, balancing the influence of past and current inputs. Despite these improvements, their inherently sequential computation still constrained training efficiency and parallelization [31].

The introduction of *attention mechanisms* overcame these limitations [29]. Rather than embedding past context into a fixed set of evolving states, as in recurrent architectures, attention assigns pairwise relevance weights between elements of a sequence, allowing each word to selectively attend to other elements directly, without compressing the entire context into a single recurrent state. The *Transformer* architecture [160] is built entirely upon attention-based layers, maintaining a distinct contextual representation for each element in the sequence. These representations are obtained by combining information from all elements according to learned attention weights, rather than by propagating a state over the sequence. As a result, the computation is not inherently sequential and can be fully parallelized during training, while still enabling effective modeling of global contextual relationships.

Transformers can be configured as encoder-decoder models (for sequence-to-sequence tasks such as translation), encoder-only models (for classification and comprehension), or decoder-only models (for autoregressive text generation). The latter configuration underlies most contemporary LLMs, which implement large-scale autoregressive language modeling [179].

2.2.3 Text generation

Autoregressive language models generate text by iteratively selecting the next word w_t from the vocabulary V according to the conditional probability $P_\theta(w_t \mid w_{<t})$ predicted at step t. A *decoding strategy* specifies how this distribution is used to select the next word, balancing determinism and diversity [42].

Greedy decoding. The most straightforward strategy always chooses the most probable continuation:

$$w_t = \arg\max_w P_\theta(w \mid w_{<t})$$

This yields deterministic and locally optimal choices but often leads to repetitive or overly constrained text [42].

Stochastic decoding. Instead of taking the most likely word, stochastic methods sample from the predicted distribution, introducing variability in the output [42]. Before sampling, the distribution can be modified to control its shape.

- **Temperature scaling.**
  A temperature parameter $T > 0$ rescales the distribution as:

$$P_{\theta,T}(w_i) = \frac{P_\theta(w_i \mid w_{<t})^{1/T}}{\sum_{j=1}^{|V|} P_\theta(w_j \mid w_{<t})^{1/T}}$$

  Values of T lower than 1 sharpen the distribution, making sampling more deterministic, while higher values flatten it, increasing randomness [63, 42].

- **Top-p (nucleus) sampling.**
  Alternatively, sampling can be restricted to the smallest subset of words whose cumulative probability exceeds a threshold p. Sampling then occurs only within this dynamically selected nucleus, which preserves coherence while allowing creative variation. Top-p can be used in conjunction with temperature scaling to refine both the shape and scope of the sampling distribution [76, 63].

These decoding strategies reflect the dual role of language modeling: as a probabilistic estimation problem and as a generative process. They determine how models trade off predictability and variability, a key factor in the quality and diversity of generated text.

2.2.4 Large Language Models

Large Language Models (LLMs) are language models characterized by their scale in terms of both parameters and training data. The notion of "large" is relative to technological capabilities and has evolved rapidly: models once considered large contained hundreds of millions of parameters, whereas contemporary models comprise billions or even trillions. Alongside model size, the scale of training corpora has expanded dramatically, drawing from diverse sources such as books, articles, web documents, code, and multimodal data [76, 111].

This increase in scale is associated with improved estimation of the conditional probabilities that underlie language modeling, enabling models not only to capture a broader range of linguistic regularities, but also to encode substantial amounts of world knowledge within their parameters, as evidenced by their ability to retrieve factual information through language-based queries [131].

At sufficient scale, this generative objective can be leveraged to perform a wide variety of tasks by expressing them as conditional text generation problems. Rather than modifying model architectures or training task-specific components, tasks such as translation, summarization, and classification can be specified through natural-language descriptions that guide the generation process. This paradigm was explicitly formalized in early large-scale transfer learning work, which demonstrated that heterogeneous NLP tasks can be unified within a text-to-text framework without altering the underlying language modeling objective [133].

Building on this capacity to interpret task specifications expressed in natural language, subsequent advances have focused on improving the reliability and generality of model behavior in interactive settings. Instruction tuning, which fine-tunes pretrained language models on collections of tasks described via natural-language instructions, has been shown to improve their ability to generalize to unseen tasks in a zero-shot manner [166]. Complementary approaches based on reinforcement learning with human feedback further align model outputs with user preferences and communicative expectations, producing responses that are more helpful, coherent, and contextually appropriate [125].

In parallel, prompting and training strategies that encourage models to generate intermediate textual steps before producing a final answer have been shown to improve performance on multi-step problem-solving tasks. These methods elicit behaviors that resemble an intermediate, explicitly articulated "thought-like" process in text, commonly referred to as *reasoning*. [167, 82]

Their ability to reproduce outward features of intelligence in communicative contexts can be understood in light of long-standing views in linguistics, which emphasize that language use is closely associated with knowledge and reasoning, and that mastery of language is often taken as a hallmark of intelligence [28].

As LLMs have gained prominence by demonstrating increasingly fluent conversational ability and versatile performance across a wide range of tasks, the term *large language model* has sometimes been used in a more specific and even informal sense, moving away from the statistical objective of the language modeling task. Within research communities, it may implicitly refer to large transformer-based architectures designed for autoregressive generation [111]. Outside these communities, the intended meaning often centers on interactive and operational chat-oriented behavior, as these are the contexts in which they are primarily applied and most widely recognized [1].

Despite their successes, LLMs face a wide range of open challenges, including issues of explainability, social bias, and environmental impact [132, 77]. However, in the context of this thesis, challenges more directly related to understanding, such as reasoning and factual reliability, are of utmost importance.

Concerning factuality, their probabilistic nature means that their outputs reflect learned statistical dependencies rather than verified knowledge. Consequently, they may produce text that is fluent yet false, a phenomenon known as *hallucination*. A *hallucination* refers to the generation of statements that lack factual grounding in the training data or in external reality, resulting in the presentation of fabricated or incorrect information. This issue arises because the training objective of next-word prediction optimizes for plausibility rather than truth. Efforts to mitigate hallucinations often involve connecting language models to external knowledge sources or structured databases, enabling them to retrieve and ground their responses in verifiable information [77].

Faulty reasoning also remains a notable weakness. While encouraging models to articulate answers through step-by-step reasoning often improves performance,

LLMs may still fail at multi-step logical inference, producing incorrect answers that can nevertheless appear more credible. Such failures are especially evident in tasks requiring commonsense, spatial reasoning, or quantitative understanding, abilities that depend on embodied experience and interaction with the world [77].

The development of text-only language models is rooted in the *distributional hypothesis*, which assumes that word meaning can be inferred from patterns of co-occurrence with other words. This principle has enabled models to achieve remarkable fluency and contextual sensitivity, yet it constitutes a fundamental simplification [44].

This limitation connects to what [59] described as the *symbol grounding problem*: purely textual systems operate like Searle's *Chinese Room*, manipulating symbols according to rules but lacking any intrinsic understanding of what those symbols denote. Similarly, as [11] observe, the common idea that language models might learn meaning purely from text, as children might learn to communicate by listening to their parents, is misleading. Children do not acquire language solely by hearing words but through rich multimodal interaction, seeing, touching, acting, and sharing attention with others. Without this coupling between linguistic symbols and perceptual or social experience, known as *grounding*, words remain patterns of form rather than carriers of meaning. This lack of grounding explains why earlier text-only models often failed on challenges such as the *Winograd Schema*, where resolving the pronoun *it* in a sentence like "*The suitcase didn't fit through the door because it was too big*" requires commonsense knowledge about how objects behave in the physical world, knowledge that such models lacked.

For these reasons, the ongoing shift toward *large multimodal models*, systems capable of processing not only text but also images, audio, and video, should be regarded as more than a functional innovation or a practical convenience in certain applications. It represents a theoretical step toward anchoring linguistic form in perception and experience, aiming to move from distributional correlation toward genuine understanding [127].

Overall, the continuing progress in improving grounding, reasoning, and factual reliability is paving the way for the increasing use of large language models as supporting tools in tasks that demand structured reasoning, such as knowledge engineering.

2.3 Information Extraction

2.3.1 Definition and tasks

With the computational modeling of language addressed, we now turn to how meaningful information can be derived from it. *Information Extraction* (IE) refers to the process of automatically identifying and structuring information expressed in natural language [76, 33]. Within the broader context of Natural Language Processing and

Knowledge Engineering, it provides the means to derive explicit factual knowledge from textual data [10]. IE thus represents a key step in transforming unstructured or semi-structured text into representations such as databases, ontologies, or knowledge graphs [76, 10].

IE addresses the question of *who did what, to whom, when, and where* [76, 33]. The answer involves a set of complementary subtasks that together support the structuring process and form the core components of IE [33]. Although conceptually straightforward, these subtasks remain computationally demanding due to the ambiguity, variability, and contextual dependency inherent to natural language [76]. Within the broader context of NLP, they are therefore accompanied, sometimes implicitly, by related tasks that provide essential support, such as coreference resolution [33]. The following paragraphs examine the core IE subtasks in greater detail.

Named Entity Recognition. *Named Entity Recognition* (NER) identifies spans of text that refer to named entities, i.e., real-world objects or concepts with a distinguishable identity that can be denoted by a name [76, 33].

Historically, the task focused on a fixed inventory of entity types such as persons (`PER`), organizations (`ORG`), locations (`LOC`), and temporal expressions (`TIME`) [33, 115]. However, a named entity exists independently of such categorical constraints, and the task has gradually evolved into a more general framework adaptable to any referent of interest [76]. Early systems relied on handcrafted rules and gazetteers designed for a limited set of references [33]. These were later replaced by probabilistic sequence models such as Hidden Markov Models and Conditional Random Fields, which helped expand coverage to a broader range of categories [76].

Modern approaches employ neural architectures. They provide a more context-sensitive interpretation that improves generalization and mitigates lexical ambiguity and variation across linguistic forms and domains [89]. Despite these advances, NER remains sensitive to domain shifts, and fine-grained or specialized entity types often require tailored ontological or domain-specific resources [10].

Relation Extraction. *Relation Extraction* (RE) aims to determine the semantic relations that hold between entities, for instance that a person is the author of a book or that an organization is headquartered in a particular location [76, 33]. In many applications, these relations are defined within a predefined schema that specifies the types of links to be identified, such as authorship, employment, or geographic containment [10, 33].

Classical approaches include pattern-based methods that rely on syntactic or lexico-syntactic templates, as well as statistical models trained on annotated examples [33, 8]. While pattern-based systems offer high precision, they typically lack coverage, whereas data-driven models generalize more broadly but are prone to generating spurious relations [76, 33]. The most effective systems combine these

strategies by integrating linguistic patterns, statistical learning, and semantic constraints from external knowledge bases [76, 10].

Event Extraction. *Event Extraction* generalizes relation extraction from static associations to dynamic occurrences. It identifies mentions of events such as meetings, acquisitions, or discoveries, together with their participants, temporal anchors, and contextual modifiers [76, 33].

The task generally involves detecting event triggers, that is, lexical items that signal the occurrence of an event, and assigning argument roles that specify who did what, when, and where. Temporal expressions and their ordering are also essential components, often handled through dedicated tagging and normalization procedures [76, 33].

2.3.2 Approaches and evaluation

The core tasks of IE can be approached under two complementary paradigms [76, 33].

- **Closed IE.** The inventory of entity and relation types is predefined, and extraction is constrained by specific schemas [10, 33]. It is typically framed as a supervised classification or sequence labeling problem [76]. This setting is common in domain-focused applications, such as biomedical or legal IE, where schemas and ontologies restrict the space of possible outputs [10].

- **Open IE.** Systems extract entities and relations directly from text in an unconstrained form [46]. Methods frequently rely on dependency patterns, part-of-speech regularities, and semantic role labeling to identify candidate triples, trading some precision for broader applicability [76].

Closed IE benefits from the availability of focused annotated corpora, which guide learning and reduce ambiguity at evaluation time [33]. Open IE faces additional challenges: it must determine which content is salient, canonicalize heterogeneous surface forms of identical concepts, and type or normalize arguments without prior domain constraints [76, 10]. As a result, open IE requires stronger generalization capabilities and often benefits from domain knowledge even when not explicitly available [10]. The two paradigms also differ in the way their performance can be evaluated [76].

Intrinsic evaluation. Intrinsic evaluation typically reports *precision* and *recall* metrics, which require determining what extracted information is correct and what has been missed. This can be done either by directly assessing the system's output or by comparing it against a complete, manually curated reference (*ground truth*) [76, 33].

In closed IE, a predefined schema constrains the expected label space and information structure. It is therefore easier to determine what falls within scope for direct evaluation and to establish clear matches with the reference [33].

In contrast, open IE lacks such constraints. Because what should be extracted is not explicitly defined, direct evaluation depends heavily on the evaluator's judgment of what constitutes valid information [76]. The absence of a schema also allows for different, yet equally valid, formulations of the same fact, which further complicates comparison with a ground truth. Open IE may rely on semantic similarity measures for more flexible matching, but thresholds and biases must be handled with care [76].

Recent practice sometimes employs model-assisted judging to scale annotation, which requires clear guidelines, calibration, and adjudication. Even with these methods, the fundamental challenge remains: in an open setting, it is difficult to define what the system was expected to extract and, consequently, what counts as correct or missing information [76].

Extrinsic evaluation Extrinsic evaluation measures the utility of extracted structures in downstream tasks such as question answering, knowledge base population, or ontology maintenance [76, 33]. Here the distinction between closed and open settings matters less, provided that outputs are transformed into a form usable by the downstream system [76]. Nevertheless, closed IE often integrates more directly because its outputs already conform to an expected schema and may better fit specific domain tasks, whereas open IE typically requires additional consolidation and alignment [10, 33].

2.3.3 LLM-based extraction

Recent advances in neural architectures have transformed IE into a context-sensitive and multi-stage reasoning process. Pretrained transformer models achieve strong performance on NER, RE, and event extraction, often requiring only modest task-specific supervision or fine-tuning [76].

More recently, *Large Language Models* (LLMs) have been increasingly employed for zero-shot or few-shot IE, formulating extraction as a generation or question-answering task. Instead of explicit pipelines, these models condition text generation on natural-language prompts that describe the extraction schema, producing structured outputs in the desired formats. This paradigm unifies previously distinct subtasks and enables flexible adaptation across domains [76, 178, 170].

However, LLM-based IE also introduces several challenges. Ambiguous text remains difficult to interpret correctly, and models may produce spurious associations between entities. Factual grounding is another concern: while the goal of IE is to extract what is explicitly stated in the source, LLMs may inadvertently supplement or modify information based on their internal knowledge. Although this capacity can sometimes be advantageous, in settings requiring strict fidelity to the source it represents a major limitation [76, 178, 170]. Reproducibility is also affected, as

hosted LLM services typically expose only stochastic decoding, preventing fully deterministic execution, even though local deployments can enforce fixed decoding strategies.

2.4 Semantic Web and Knowledge Graphs

2.4.1 Definitions and principles

Once information has been extracted from text, it must be organized within computational structures that capture entities, relations, and their interconnections in a consistent and reusable form. The *Semantic Web* extends the current Web by providing a framework in which information is given well-defined meaning, enabling data to be shared and reused across applications and domains. Its foundations lie in the formal representation of knowledge through machine-interpretable structures that capture the semantics of concepts and their relations, allowing systems to perform reasoning, integration, and automated discovery. In this sense, the Semantic Web embodies the vision of transforming the Web from a collection of linked documents into a distributed network of linked data and knowledge [61].

An *ontology* provides the conceptual schema for a domain, describing the relevant types of entities, their attributes, and relationships [62, 55]. It serves as both a shared vocabulary and a computational model of a knowledge domain. Ontologies range from lightweight vocabularies with simple hierarchies (such as SKOS [110]) to expressive logical systems like SNOMED CT in medicine [37], where formal constraints and axioms enable precise reasoning over domain data. In the context of the Semantic Web, ontologies thus function as an interface between human knowledge organization and automated reasoning systems [62].

A *knowledge graph* (KG) can be seen as a concrete instantiation of these principles, representing knowledge as a network of interconnected entities and relations that can be built following the schema of an ontology. Each node in the graph corresponds to an entity or concept, while edges encode relations between them, allowing for complex relational patterns that go beyond hierarchical taxonomies [62, 41]. The term encompasses a broad family of systems that merge ideas from semantic web technologies, database research, and artificial intelligence, emphasizing connectivity, semantics, and reasoning over data. While many knowledge graphs are grounded in Semantic Web standards and principles, others adopt property graph models that prioritize flexibility and computational efficiency over formal semantics [62]. Knowledge graphs are particularly suited to integrating heterogeneous sources, as they can combine structured and unstructured information under a uniform graph-based model [41].

The *Google Knowledge Graph* [151] popularized this paradigm of "things, not strings", shifting search from matching words to identifying entities and their relations. It also demonstrated how large-scale knowledge graphs could serve as an

interface between structured representations and user-facing applications such as semantic search and question answering, inspiring the adoption of similar architectures across major web platforms.

2.4.2 Terminology and conceptual boundaries

The terminology surrounding the Semantic Web and Knowledge Graph representation can be perceived as inconsistent and occasionally ambiguous, which may lead to confusion [62, 41]. Clarifying these key terms provides an appropriate starting point.

Ontologies typically define *entities* and *relations*, which knowledge graphs then instantiate. Under a graph paradigm, these correspond respectively to *nodes* and *edges* [62]. Intuitively, an entity can be understood much like the named entities introduced in Section 2.3.1: an object or concept that has a unique identity, while relations represent the connections between such entities.

Ontologies, however, employ a more specific vocabulary. *Classes* define the hierarchy of types that objects or concepts may belong to, while *individuals* are concrete instances of these types, typically introduced by the knowledge graph that instantiates the ontology. *Relations* are referred to as *properties* and are further divided into three kinds: *object properties* (linking individuals to other individuals), *data properties* (linking individuals to literal values), and *annotation properties* (linking any element to textual or metadata information) [55, 103].

From a formal ontological perspective, classes, individuals, and properties each possess a distinct identity defined by the ontology and can therefore all be considered entities within it [55]. In this formal sense, an *entity* is anything with a unique identity, encompassing the ontological definitions themselves. In the practical context of a knowledge graph, however, the term *entity* is often used more narrowly to denote the nodes of the graph, that is, the concrete instances of concepts [62, 41]. This ambiguity is common, but the intended meaning should be clear from context.

Further ambiguities concern the related terms *ontology*, *schema*, *taxonomy*, and *vocabulary*, which differ in scope and formality yet are sometimes used interchangeably to describe structured representations of domain knowledge [50]:

- A *vocabulary* is the simplest representation of a domain. It defines a controlled set of terms relevant to a specific context, without introducing hierarchy or explicit relationships among them [50].

- A *taxonomy* extends this idea by introducing a hierarchical structure that captures parent–child relations between terms, providing a simple conceptual organization of the domain [50].

- A *schema* (or *semantic schema*) defines how entities and their relationships are structured within a domain, focusing on the organization and constraints of data rather than rich semantics. It describes how information is represented

and often serves as a bridge between conceptual and implementation levels. It may also be used in the broader sense of the "formal structure of a domain", which explains why one can refer to an ontology as a schema for a domain [62, 50].

- An *ontology* goes beyond a schema by providing a formal, semantically rich representation of a domain. It includes not only hierarchical and structural aspects but also logical axioms and constraints, such as cardinality, domain and range restrictions, and property characteristics, enabling automated reasoning and inference. Ontologies thus support deeper understanding and interoperability within and across domains [62, 50].

Although ontologies and knowledge graphs are related as schema and instantiation, their distinction might blur in practice [62, 41]. When modeling data, it is possible to find that the resulting structure is referred to interchangeably as an ontology or as a knowledge graph. This depends largely on where one chooses to draw the line between *classes* and *individuals*. If a modeler treats a subclass of a concept as an instance, the structure may be described as a knowledge graph; if the same element is regarded as a class to be further instantiated, the model may instead be considered an ontology. For example, one may ask: *Is Margherita a subclass of Pizza or an individual instance of it?* Formally, ontology languages make the distinction between *classes* and *individuals* syntactically explicit and unambiguous, so such confusion is resolved at implementation time [62, 103].

Another ambiguity concerns whether the schema or ontology is considered part of the resulting knowledge graph or kept distinct from it. In some interpretations, the ontology is integrated into the graph, forming a single structure that contains both schema-level and instance-level elements. In others, the ontology is conceived as separate, providing the schema that defines and constrains the knowledge graph but not belonging to it [62, 41]. This distinction affects how the term *knowledge graph* is understood, whether as a self-contained representation that includes its schema or as an instantiated data layer built upon a distinct ontological model.

A final source of ambiguity concerns the distinction between *knowledge graphs* and *knowledge bases*. These two terms are sometimes used interchangeably, yet they carry slightly different connotations [62]. A *knowledge base* traditionally refers to a repository of facts expressed in a logical form, typically supported by a reasoning component that allows the derivation of new knowledge [62, 19]. A *knowledge graph*, in contrast, emphasizes the graph-based structure of that knowledge, focusing on the connectivity between entities and the use of identifiers that enable integration across heterogeneous sources [62].

2.4.3 Knowledge representation and modeling

The conceptual structures modeled with ontologies and knowledge graphs are implemented in practice through formal representation models and data standards typical

of the Semantic Web. Together, these provide the means to describe and interlink data in a machine-readable way.

To represent the core concept of identity necessary to model entities of any type, the Semantic Web relies on *Uniform Resource Identifiers* (URIs) [14] or, more generally, on *Internationalized Resource Identifiers* (IRIs) [39]. These ensure that entities or concepts can be uniquely and globally referenced, enabling interoperability across distributed systems.

The *Resource Description Framework* (RDF) [81] provides the foundational data model: a directed labeled graph composed of triples of the form (s, p, o), denoting a *subject* s, *predicate* p, and *object* o. Each triple constitutes a *statement*, that is, an assertion expressing that a given subject is related to an object through a specific predicate. Statements model atomic facts about entities and their relationships, providing the building blocks for representing knowledge in a machine-interpretable form. Subjects and predicates are typically identified by IRIs, ensuring unambiguous references across data sources. The simplicity of RDF makes it a flexible abstraction for representing both factual statements and conceptual relationships, forming the structural basis for higher-level reasoning. Extensions such as blank nodes, reification, and named graphs increase its expressiveness, allowing the representation of anonymous entities, statements about statements, and contextualized subgraphs.

The *Web Ontology Language* (OWL) [103] extends the expressivity of RDF by introducing formal modeling of class hierarchies, property constraints, and the logical axioms required for ontology-based reasoning. Its formal semantics are grounded in description logics, which enable automated inference: from explicit assertions contained in a dataset, additional facts can be derived according to logical entailment.

Although these tools provide the representational foundation, modeling real-world knowledge requires addressing additional challenges. Designing and maintaining ontologies or knowledge graphs remains a demanding process, since it requires both domain expertise and formal modeling skills. Large-scale ontology engineering is also costly and time-consuming, which has motivated research on strategies to assist the process [78, 119].

In this context, *ontology learning* and *knowledge graph construction* refer to the automatic or semi-automatic creation of semantic structures from text corpora or existing data sources [181, 21]. The automatically derived structures can then be refined and formalized by human experts, accelerating the creation of domain ontologies and knowledge graphs. The distinction between the two often lies in focus: ontology learning targets the discovery of conceptual structures, such as classes, relations, and constraints, while knowledge graph construction focuses on extracting instance-level facts, typically represented as triples, following either open or predefined schemas [181, 21]. Both approaches leverage methods from information extraction to identify candidate entities and relations [10]. A central challenge lies in determining which elements of a natural-language expression should be represented as formal concepts of a domain. Identifying and isolating relevant concepts and encoding their structure in formal data models is, in itself, an act of semantic

interpretation: representing linguistic content formally presupposes understanding its meaning [76, 10]. Models capable of understanding meaning and performing reasoning are therefore essential in these processes, echoing the challenges discussed for natural language processing where meaning must also be represented, structured, and inferred computationally.

Additionally, while ontologies and knowledge graphs formally represent the meaning and structure of a domain, their symbolic representations have intrinsic limitations [62]. Ontologies model domains through logical definitions of classes and properties, yet part of their meaning remains anchored in the conceptual descriptions and natural-language annotations that accompany those definitions. This dimension of meaning, which is implicit, contextual, and distributed, is difficult to capture through logic alone but can be effectively modeled in continuous vector spaces, as demonstrated by natural-language representations. Embedding-based techniques project entities and relations into such spaces, enabling the representation of semantic similarity and conceptual relatedness beyond explicit logical structures [62, 51]. In this hybrid view, the Semantic Web provides the explicit and verifiable structure, while embeddings capture the latent, language-based dimensions of meaning.

2.4.4 Access and inference

Once information is modeled within semantic representations, having the means to retrieve and exploit it is essential to move from mere data representation to the actual use of knowledge. In this sense, access and inference constitute the operational layer of knowledge representation, enabling the transition from data to information and, ultimately, to knowledge.

Direct access to structured knowledge is typically mediated through formal query languages. In the case of RDF-based graphs, the SPARQL Protocol and RDF Query Language (SPARQL) [64] provides a declarative syntax for retrieving and manipulating data. It allows users to express complex graph pattern queries involving conjunctions, disjunctions, filters, and aggregations. SPARQL thus serves as the operational interface to the Semantic Web, enabling precise, schema-aware access to knowledge graphs. Other query languages have been developed in parallel for graph-based data implemented under different paradigms, such as Cypher [47] for property graphs.

The use of such formal query languages presupposes familiarity with both their syntax and the underlying data model, which creates a barrier for non-expert users. Recent advances in natural language processing and large language models have the potential to overcome this limitation by automatically translating natural-language questions into formal queries [36, 93]. In this setting, the model acts as an intermediary between human language and formal data structures, effectively performing *semantic parsing* and enabling intuitive access to structured knowledge. This convergence of language understanding and semantic data access represents a key step toward making the Semantic Web more accessible and actionable.

Beyond query-based access, reasoning over knowledge graphs extends their utility beyond data storage. Given a well-defined OWL representation, a *reasoner* can compute logical consequences such as class subsumptions, instance memberships, type inheritance, or transitive closure [103, 7]. These operations allow systems to uncover inconsistencies, validate ontological coherence, and infer implicit knowledge. For instance, from the facts that *Rome is the capital of Italy* and *Italy is located in Europe*, a reasoner can infer that *Rome is located in Europe*. Beyond such straightforward cases, reasoning techniques can uncover more complex and non-trivial relations, inferring new axioms and connections that give rise to additional, otherwise implicit, knowledge. This capability supports advanced applications such as question answering, semantic search, and knowledge-based recommendation, all of which rely on a graph's ability to represent and infer relations with semantic coherence [62].

2.5 Recent Directions in Knowledge Engineering

Recent research across the three main directions of this thesis (knowledge structuring, knowledge access, and applications to electronic health records) provides the contextual background for the contributions outlined above. These areas have attracted growing attention, reflecting a collective effort to connect unstructured language with structured, machine-interpretable representations through LLMs. This section briefly reviews the most relevant research directions and works related to the construction, retrieval, and application of structured knowledge.

Knowledge structuring. The first strand of related research concerns the automatic transformation of unstructured text into structured, machine-interpretable knowledge, focusing on ontology learning and knowledge graph construction.

Early work on ontology learning from text introduced semi-automatic systems that combined linguistic analysis and statistical term weighting to extract candidate concepts and relations [100, 32, 143]. Such approaches established the basis for schema induction but generally required extensive linguistic resources and were limited to narrow, predefined domains.

Subsequent studies explored neural architectures for improving concept recognition and relation classification [121, 124, 138, 140], yet these systems still relied on manually annotated data and domain-specific templates. The adoption of large pretrained language models has shifted ontology learning toward more flexible zero-shot and few-shot settings. Generative models can now propose hierarchical structures and candidate relations through prompting, conversational interaction, or iterative refinement [48, 176, 155]. Despite these advances, the automatic creation of coherent, domain-specific ontologies remains difficult, particularly in ensuring conceptual consistency and interpretability.

Parallel efforts in knowledge graph construction have aimed to populate such structures directly from heterogeneous or unstructured data. Classical pipelines combine modules for entity recognition, relation extraction, and entity resolution [27, 117, 40], enabling the semi-automatic generation of graphs in domains such as finance or news [43, 106]. These methods, however, depend on predefined ontologies and extensive human annotation, which limits scalability and adaptability to new contexts.

Recent advances in transformer-based and large language models have alleviated some of these constraints by improving contextual understanding and generalization [163, 58]. LLMs have shown strong zero-shot and few-shot capabilities in information extraction tasks such as entity typing, relation detection, and predicate mapping [3, 168, 87, 162].

Nevertheless, challenges remain in achieving precision and coherence at scale, particularly when extracted content must be integrated into consistent, reusable knowledge graphs.

Knowledge access. A complementary line of research focuses on enabling users to access and query structured data through natural language. Knowledge Graph Question Answering systems translate user questions into executable queries, typically SPARQL, to retrieve information from knowledge graphs.

Early approaches relied on handcrafted rules and templates to map natural-language phrases to ontology elements [137, 122]. Although effective for simple questions, such systems required extensive domain engineering and struggled to generalize beyond predefined patterns. Subsequent work introduced neural sequence-to-sequence architectures that learned direct mappings between questions and SPARQL queries [98, 152, 175]. These models reduced dependence on manual rules but still required large training corpora and remained sensitive to domain and schema variations.

The introduction of pre-trained transformer models extended this paradigm, showing that general-purpose language representations could improve translation accuracy and reduce task-specific supervision [9, 136, 92, 134]. More recently, prompting-based methods have explored the zero- and few-shot capabilities of large language models for SPARQL generation without fine-tuning [80, 107, 84, 45, 141, 6]. These approaches demonstrate that LLMs can interpret and generate structured queries directly from natural language, yet they remain constrained by prompt design, ontology coverage, and limited evaluation benchmarks.

The task of achieving consistent and domain-agnostic query generation therefore continues to represent an open challenge for knowledge access.

Applications to electronic health records. Research on electronic health records (EHRs) has increasingly applied natural language processing to clinical narratives, progress notes, and discharge summaries [65, 88]. These methods support a

range of automation tasks, from entity and relation extraction to document classification, coding, and summarization, thereby enabling downstream applications such as risk prediction and clinical decision support.

The introduction of transformer-based architectures has markedly improved extraction quality, although many studies still rely on task-specific annotation or narrow-domain adaptation [128]. Large language models have since expanded this line of work by demonstrating zero-shot and few-shot capabilities across diverse extraction and reasoning tasks [135]. Applied studies show their use in clinical information extraction [67, 2, 161, 123, 150, 52, 120] and diagnostic support [102, 57, 182, 49, 144], but also beyond direct clinical reasoning, as in coding assistance and structured summarization [72, 105, 5]. Despite these advances, fully reliable and generalizable clinical systems remain elusive, with performance often dependent on retrieval augmentation, fine-tuning, or ontology-aware workflows [97].

In parallel, EHRs have been increasingly represented as knowledge graphs to support interoperability and reasoning. Standards such as FHIR and data models like OMOP-CDM provide a basis for integration but are limited in semantic expressiveness. Knowledge graphs extend these models by linking structured and unstructured information within a unified, temporally aware framework [113, 147, 146, 17, 85].

Taken together, these studies outline a clear trend toward LLM-assisted knowledge engineering, where language models act both as sources of semantic insight and as interfaces to structured data. The methods and experiments developed in the following chapters build directly on this convergence, aiming to integrate automatic structuring, intuitive retrieval, and domain-grounded application within a unified framework.

Chapter 3

Knowledge Structuring

After outlining the theoretical and technological foundations of this work, the focus now moves to the automatic transformation of unstructured text into structured, machine-interpretable knowledge, which represents one of the central challenges of knowledge engineering. This process enables the reuse of the vast amount of information already available in textual form, produced in ever-increasing quantities by public institutions and private organizations as part of the ongoing digital transformation. Much of this material remains unstructured and primarily intended for human interpretation, which makes it largely inaccessible to computational systems and limits its potential reuse for analytical and decision-making purposes.

Knowledge Graphs (KGs) have emerged as a key solution to this problem by providing a formal structure to represent entities, their properties, and relationships in a consistent and interpretable form. However, constructing such graphs automatically from text requires defining an appropriate schema that describes the entities of the domain and populating that schema with the corresponding instances. As discussed in Section 2.4.3, ontology learning research has long sought to automate both processes, bridging the gap between textual content and formalized conceptual structures. The challenge remains substantial, since schema definition and population have traditionally relied on extensive manual work and domain expertise, making them time-consuming, subjective, and prone to incompleteness.

Recent advances in Large Language Models (LLMs) open new opportunities to address this challenge. Their ability to recognize and generalize linguistic patterns across large corpora enables the automatic identification of salient concepts and their organization into structured representations. LLMs can therefore support or even replace parts of the manual design process in ontology and schema construction, complementing established knowledge engineering techniques with scalable language-based inference.

In this thesis, the problem of knowledge structuring is approached through the use of LLMs to induce or infer the conceptual organization underlying textual data. The chapter explores two complementary strategies that differ in their starting assumptions and workflow design, depending on whether the domain schema is defined

before extracting information from text or allowed to emerge from freely extracted content. These two perspectives, the *schema-first* and *extraction-first* approaches, are presented respectively in Sections 3.1 and 3.2. Their comparative discussion is then reported in Section 3.3.

3.1 Schema-first structuring

The *schema-first* strategy approaches knowledge structuring from the perspective that a domain should be formally defined before any extraction of entities or relations is attempted. In this view, the schema acts as the conceptual backbone of a Knowledge Graph, describing its main classes, entity types, and hierarchical relations prior to any instance-level extraction. Once defined, such a schema can constrain and guide the organization of information, ensuring consistency and interpretability across extracted knowledge.

Despite the long-standing interest of ontology learning in this topic, datasets that map textual content to ontological structures remain scarce, as creating them is a challenge in itself. Most available formal resources provide only simple domain definitions, with limited sets of entity types and shallow or no hierarchies. Automatically devising schemas such as vocabularies or shallow taxonomies, rather than full ontologies, represents a practical intermediate objective: they capture the essential conceptual structure of a domain while remaining comparable with the type systems used in information extraction datasets.

The following sections describe an approach that operationalizes this perspective through the use of LLMs for schema induction, combining generative prompting and unsupervised clustering to infer a set of domain categories grounded in textual evidence. Although the resulting structure does not yet represent a complete ontology, it constitutes a first attempt to define the semantics of concepts by introducing natural-language definitions of the extracted categories. This stage forms the initial component of a schema-first pipeline, in which the induced schema can subsequently guide the extraction and population of Knowledge Graphs.

3.1.1 Methodology

The following pipeline operationalizes the *schema-first* perspective by combining LLM-based abstraction with unsupervised clustering to derive a data-grounded vocabulary of domain categories. Figure 3.1 summarizes the process, which comprises four modules: *Keyword Extraction*, *Topic Discovery*, *Topic Clustering*, and *Cluster Labeling*. Each module is described below.

Keyword Extraction

The pipeline aims to generate an exhaustive entity schema. This first module is designed to identify document-level salient terms grounded in the source documents.

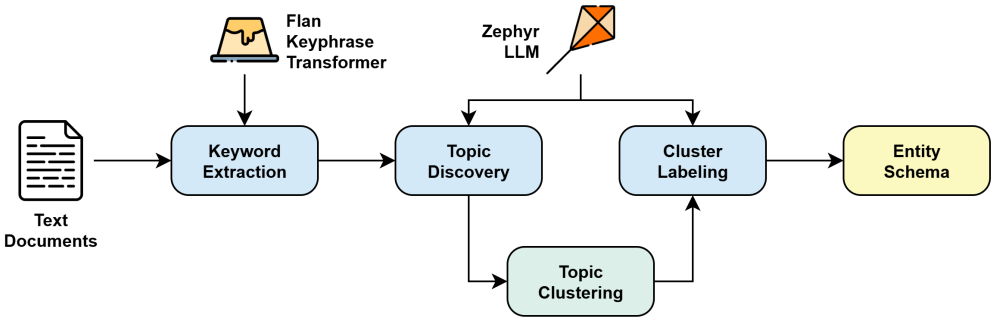

Figure 3.1: Automatic entity schema generation pipeline. Blue modules utilize generative LLMs. Green relies on text embeddings and unsupervised clustering. Yellow represents the output.

Since it must describe the content of the text, a bottom-up approach has been adopted, starting from the identification of concepts that are important and recurrent within the documents of interest. Given a set of documents D, this module produces a set of keywords K containing the salient terms that characterize D. The implementation relies on the *KeyphraseTransformer*[1], a sequence-to-sequence Transformer model fine-tuned from the *FLAN-T5* model [30] to perform keyphrase extraction as a conditional text generation task. Given a document as input, the model encodes the document text and generates a textual list of salient keyphrases as output. For each document $d_i \in D$, a keyword set k_i is extracted by running the *KeyphraseTransformer* on d_i and post-processing the generated output into a set of keywords. All unique keywords across $k_1, \ldots, k_n$ are then merged into the final set K.

Topic Discovery

While keyword extraction focuses on identifying salient document-level terms, topic discovery addresses a distinct sub-task by performing semantic generalization, mapping specific keywords to broader conceptual categories. A keyword set contains terms specific to each document, whereas an entity schema requires broader terms applicable across different contexts. This module generalizes the extracted keywords into more generic topics.

The process relies on the generative LLM *Zephyr* [158], which is instructed through a prompt specifically designed for this task (the full prompt is reported in Section A.1.1). Among the wide range of available LLMs, *Zephyr* is selected as a concrete and reproducible instantiation of the generative component in the proposed pipeline. The choice was motivated by practical considerations, including open-source availability and local executability, rather than by an attempt to

[1]https://github.com/Shivanandroy/KeyPhraseTransformer

identify an optimal model for the task. Preliminary, informal evaluations based on qualitative manual inspection were conducted to assess whether the model was suitable for the intended information extraction task.

Keyword sets k_i are provided to *Zephyr* to obtain hypernyms for the contained terms. The model produces a JSON object formatted as {*original keyword: closest hypernym, ...*}. Topic frequencies are subsequently computed, discarding those below a threshold θ to remove less meaningful terms, as described in Section 3.1.2.

Topic Clustering

Even though the topic discovery module generalizes the terms representing the extracted keywords, the resulting set still contains an excessive number of topics. Many of these are semantically similar and describe the same underlying concept. The purpose of this module is therefore to group semantically related topics into coherent sets, referred to as clusters.

The process begins by aggregating keywords according to their topic labels, obtaining for each topic a set of keywords whose hypernym corresponds to that label.

To perform topic clustering, a feature representation is defined for each topic. Such representation is derived from a *distributed* embedding of the topic and its associated keywords. Given a topic, both its label and all related keywords are projected into an embedding space, as described in Section 3.1.2. The final topic embedding is computed as the weighted average between the label embedding, weighted by w_t, and the centroid of the keyword embeddings, weighted by w_k. This strategy is informed by exploratory, informal analyses based on qualitative inspection, which highlighted two main issues.

- Considering only the embedding of the topic label is suboptimal, as the *FLAN* model often returns correct but overly generic hypernyms. For example, for the keywords *apple*, *banana*, and *orange*, the topic discovery module produces the label *food*, which is correct but semantically distant from the intended concept, whereas *fruit* provides a more appropriate level of abstraction.

- Conversely, considering only keyword embeddings leads to insufficient generalization, risking the grouping of concepts that differ substantially from a semantic perspective.

The weighted strategy addresses these two problems by balancing the semantic contributions of both components, yielding a more accurate and stable topic representation.

For clustering, *Hierarchical Clustering* (HCA) [114] is adopted, as it does not require specifying the number of clusters in advance. Two main strategies exist for HCA: *Agglomerative* (bottom-up) and *Divisive* (top-down). The agglomerative approach is selected, in which all data points start as individual clusters and are iteratively merged according to a similarity metric until a single cluster remains.

Cosine similarity is used as the distance metric, combined with the *complete-linkage* method[2]. An exhaustive search[3] is performed to determine the optimal linkage distance threshold, returning the cluster configuration with the best Silhouette score [142].

Clusters are further refined using the keyword counts introduced in Section 3.1.1 to automatically remove groups insufficiently supported by the original text. For each cluster, only topics associated with at least n keywords are retained. If the cumulative keyword count of a cluster does not reach the threshold m, the cluster is discarded as not sufficiently representative.

Cluster Labeling

After generating clusters, the next step is to assign an expressive and representative label to each group of terms in order to compose the final entity schema. The label acts as an entity type summarizing the conceptual scope of the cluster.

A new prompting step is introduced, in which the LLM *Zephyr* is queried to assign a label to each cluster. The prompt instructs the model to identify a broader concept or category that encompasses all the topics contained in the cluster (the full prompt is reported in Section A.1.2).

The output of this step is the final entity schema, represented as a JSON document containing the generated labels and their corresponding abstract descriptions.

3.1.2 Experiments

The experimental phase evaluates the effectiveness of the proposed schema induction pipeline. It provides details about the implementation, the parameter configuration used to run the process, and the datasets adopted for validation. The assessment includes both qualitative and quantitative analyses of the schemas extracted from the selected corpora.

Experimental Setup

The entire pipeline is implemented in *Python*, relying on the *transformer* package[4] for executing LLM-related tasks, *sentence-transformer*[5] for the text embeddings required in the distributed representations of topics, and *scikit-learn*[6] for the implementation of the agglomerative clustering algorithm and the computation of the

[2]Searches for the maximum distance between topics of cluster pairs.

[3]Conducted in the interval $[0, 1]$ with step s, feasible due to the bounded range of the cosine distance.

[4]`https://huggingface.co/docs/transformers/main_classes/pipelines`

[5]`https://huggingface.co/sentence-transformers/all-mpnet-base-v2`

[6]`https://scikit-learn.org/stable/modules/clustering.html#hierarchical-clustering`

Silhouette score[7].

Each parameter described in Section 3.1.1 is set empirically. Weights $w_t = 0.50$ and $w_k = 0.50$ are used to compute the distributed topic representations. Thresholds are set to $\theta = 1$, $n = 3$, and $m = 9$ to maintain a conservative number of clusters. The search step for the linkage distance threshold is set to $s = 0.05$ to identify the configuration with the best Silhouette score.

Datasets

To evaluate the automatic schema generation capabilities of the proposed approach, a set of named entity recognition datasets is selected, each providing a corpus and a ground truth schema. The choice of datasets is guided by the need to include distinct and domain-specific contexts. Several classic NER datasets consist of large news corpora covering a broad range of topics but annotated only with coarse entity types such as `ORG`, `PER`, `LOC`, or `MISC`. These corpora are not well suited for describing content through a simple schema, making them less appropriate for evaluating the proposed pipeline. Instead, three domain-specific datasets are selected as more suitable test cases.

- MIT-RESTAURANTS [95] contains 1520 development sentences related to food and restaurants.
- MIT-MOVIES [95] contains 2442 development sentences related to cinema and movies.
- BIO-NLP 2004 [34] contains 1927 development sentences related to molecular biology.

Each dataset is divided into training, development, and test folds. Only the development folds are used for the experiments. Because the datasets consist of single sentences, and keyword extraction is generally performed at the document level, sentences are grouped and concatenated into texts of 20 sentences each to serve as input to the pipeline.

Evaluation

The evaluation assesses the schemas generated by the proposed pipeline when applied to the datasets described in Section 3.1.2. A valid schema should correctly and comprehensively represent the content of the corpus by providing a coherent set of entity types that reflect the underlying textual structure. Because there is no single way to group and represent the topics encompassed by a corpus, the evaluation method must account for such variability.

Two complementary strategies are adopted:

[7] `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html`

- *Qualitative evaluation*, which consists in manually comparing the extracted schema with the ground truth schema of each dataset. The comparison considers the semantics of the labels and establishes correspondences between entity types across the two schemas to determine whether the automatically extracted schema models the relevant topics appropriately.

- *Quantitative evaluation*, which measures the exhaustiveness of the generated schema with respect to the entities found in the text. The alignment between the source text and the extracted entities is assessed by running a named entity recognition model with both the ground truth and the automatically generated schemas, and comparing their results.

Entity extraction in both evaluation strategies relies on *UniNER* [183], an LLM-based named entity recognition model capable of retrieving entities according to custom type definitions provided as input.

Qualitative evaluation. The qualitative evaluation compares the entity schemas generated by the proposed method with the human-annotated ground truth schemas of the selected datasets. Tables 3.1, 3.2, and 3.3 report these comparisons. The left column lists the ground truth entity types, while the right column presents the types automatically identified by the pipeline.

Ground Truth	Automatically generated types
Rating	#
Location	Urban neighborhood, Urban area
Amenity	Parking service
Cuisine	Mexican cuisine, Chinese cuisine
Hours	#
Price	#
Dish	Food
Restaurant Name	Pizza restaurant, Fast food establishment, Food establishment, Coffee establishment

Table 3.1: Comparison between ground truth entity types and inferred entity types on MIT-RESTAURANTS dataset

Operating from the left column, the schemas are manually aligned so that each row contains equivalent types or topics. When a type from one schema has no corresponding entry in the other, the *hash* symbol (#) is used to denote the absence of a match.

All tables highlight that most ground truth types have a corresponding match in the automatically generated schema. Exceptions include a few types related to raw data, such as numeric values represented by the Ratings, Hours, and Price categories in Table 3.1, as these elements are not retrieved as keywords by the *Keyphrase Transformer* adopted in the pipeline.

Ground Truth	Automatically generated types
Genre	Film genre
Year	#
Plot	#
Average ratings, Rating	Film rating
Actor	Film actor, Human, Female celebrity
Title	Movie, Science fiction movie, Fantasy film, Comedy
Song	#
Character	Fictional character
Review	#
Director	Film director
Trailer	#
#	Cinema, Horror cinema
#	Animated medium

Table 3.2: Comparison between ground truth entity types and inferred entity types on MIT-MOVIES dataset

Ground Truth	Automatically generated types
DNA	Gene regulation, Gene, Regulatory region, Promoter element
Protein	Protein, Protein complex, Protein kinase, Enzyme, Receptor, Growth factor, Transcription factor
Cell Type	Cell biology, Cellular differentiation, Cell adhesion molecule, Cancer
Cell Line	Cell line
RNA	Messenger RNA
#	Immunological activation, Antibody, Cytokine signaling
#	Chemical compound, Phorbol ester
#	Virus
#	Programmed cell death

Table 3.3: Comparison between ground truth entity types and inferred entity types on BIO-NLP 2004 dataset

For types referring to actual entities, the automatically generated schema tends to be more fine-grained. For example, regarding the type Restaurant name in Table 3.1, the pipeline identifies multiple subtypes of establishments, including Pizza restaurant, Fast food establishment, and Coffee establishment. Similarly, the entity type Protein in Table 3.3 corresponds to several more specific concepts, such as Protein kinase, Enzyme, and Growth factor.

In the specific case of the BIO-NLP 2004 dataset (Table 3.3), the approach discovers additional entity types not present in the ground truth. Table 3.4 reports, for each new type, the number of unique entities detected by *UniNER*. Using the automatically generated schema, *UniNER* identifies a total of 269 unique entities

Entity type	Unique entities
Immunological activation	29
Antibody	25
Cytokine signaling	27
Chemical compound	134
Phorbol ester	18
Virus	31
Programmed cell death	5

Table 3.4: Summary of additionally discovered entity types not present in the ground truth of the BIO-NLP 2004 dataset

distributed across seven novel types absent from the ground truth. These results suggest that automatic schema induction methods such as the one proposed here can serve as valuable tools to extend existing datasets and to support human annotators in defining appropriate entity schemas for new domains.

Quantitative evaluation. To quantitatively assess the comprehensiveness of the automatically generated schema, the evaluation employs the *Topical Similarity Score* (TS) introduced by [71]. This metric was originally designed to measure the informational richness of extracted triples relative to the source text. In the present context, it is used to quantify the informational coverage of entities extracted with *UniNER* according to a given schema. A more comprehensive schema allows the NER model to retrieve entities that better reflect the content of the text, thereby achieving a higher TS score and indirectly indicating schema quality and completeness.

The TS score is computed for both the ground truth schema and the automatically generated schema using the same extraction tool to ensure comparability. It is derived from the *Kullback–Leibler divergence* between the probability distribution of latent topics identified in a document and the topics associated with the extracted entities. The topic distribution of a text is modeled through a *Latent Dirichlet Allocation* (LDA) model with N latent topics, denoted as LDA_N. For each document D, the set of extracted entities is defined as E_D, where each entity is encoded as "*entity label* (*type label*)" and concatenated into a single representation. The topical similarity is then computed according to Equation (3.1).

$$TS(D, E_D, N) = \exp\left(-\sum_{i=1}^{k} LDA_N(D)_i \cdot \log\left(\frac{LDA_N(D)_i}{LDA_N(E_D)_i}\right)\right) \tag{3.1}$$

The results show the comparison of topical similarity scores between entities extracted according to the automatically generated schema (AG) and the ground truth schema (GT) for different numbers of latent topics N used in LDA topic modeling.

The reported scores are obtained from a single execution of the schema generation pipeline and the entity extraction tool. Multiple runs yield only minor variations

	N=5	N=10	N=20	N=30	N=40	N=50	N=75	N=100
AG	0.939	0.850	0.630	0.519	0.583	0.438	0.350	0.421
GT	0.930	0.807	0.630	0.514	0.520	0.529	0.382	0.343

Table 3.5: MIT-RESTAURANTS TS score of UniNER entities extracted with the automatic schema (AG) versus the ground-truth schema (GT).

	N=5	N=10	N=20	N=30	N=40	N=50	N=75	N=100
AG	0.908	0.802	0.771	0.680	0.670	0.616	0.557	0.524
GT	0.929	0.845	0.822	0.753	0.705	0.699	0.635	0.571

Table 3.6: MIT-MOVIES TS score of UniNER entities extracted with the automatic schema (AG) versus the ground-truth schema (GT).

due to the probabilistic nature of the generative models involved. A higher TS value indicates stronger alignment between schema-based entities and the underlying text. Since the absolute values of TS are not directly interpretable, the analysis focuses on the relative differences between the AG and GT results.

Across all datasets, the automatically generated schema demonstrates comparable or slightly superior performance to the ground truth, with an average difference of less than 1% in the TS score. This outcome indicates that the automatically induced schemas are both comprehensive and consistent with the semantic structure of the source corpora.

3.1.3 Summary

The schema-first approach presented in this section demonstrates how LLMs can support the induction of simple yet meaningful domain schemas directly from unstructured text. By combining generative prompting and unsupervised clustering, the pipeline produces a vocabulary of entity categories and descriptions grounded in the linguistic evidence of the corpus. The experimental results show that the automatically induced schemas capture the essential conceptual structure of each domain and achieve coverage comparable to human-defined type systems. Although the resulting schemas remain at a preliminary level of abstraction, they constitute a practical foundation for subsequent knowledge extraction and population tasks.

3.2 Extraction-first structuring

Complementing the schema-first strategy of Section 3.1, the extraction-first approach begins by detecting entities and relations directly from text without a predefined schema. The central idea is to let structure emerge from document-grounded mentions and predicates, and only then consolidate types, relations and hierarchy.

	N=5	N=10	N=20	N=30	N=40	N=50	N=75	N=100
AG	0.975	0.930	0.832	0.820	0.790	0.724	0.697	0.630
GT	0.954	0.887	0.755	0.748	0.683	0.668	0.554	0.542

Table 3.7: BIO-NLP 2004 TS score of UniNER entities extracted with the automatic schema (AG) versus the ground-truth schema (GT).

This perspective aligns with open information extraction paradigms, which prioritize the discovery of factual statements expressed in natural language. By operating without a fixed schema, extraction-first methods can capture a broader and potentially more nuanced range of entities and relations, including domain-specific or context-dependent ones that may not be anticipated in predefined data models. At the same time, this flexibility introduces challenges of consistency, redundancy, and semantic alignment, as the resulting information must later be organized and normalized into coherent structures.

In this section, a zero-shot, LLM-driven pipeline is presented to operationalize this strategy at the *document level*. The pipeline extracts entities enriched with textual descriptions and type indicators, then iteratively identifies and canonicalizes relations among them to form predicate-labeled RDF triples. Each step is designed to preserve grounding in the source text, limiting hallucinations and ensuring traceability. The resulting triples form the basis for constructing Knowledge Graphs that are both expressive and adaptable, allowing the emergent schema to be inferred from empirical data rather than predefined design.

3.2.1 Methodology

The following pipeline operationalizes the *extraction-first* perspective by leveraging LLMs for zero-shot information extraction. The process is designed to detect entities and relations directly from text, enriching them with semantic context and iteratively forming predicate-labeled RDF triples. Figure 3.2 summarizes the overall workflow, which consists of two main stages: *Entity Extraction* and *Iterative Triple Extraction*. Each stage is described below. For brevity, explicit prompt templates are omitted, given their number and length. The complete implementation is provided in a public GitHub repository[8].

Entity Extraction

The first stage performs a comprehensive entity characterization, extending beyond the simple identification of text spans. Its objective is to represent each entity mention as a semantically enriched unit composed of (i) an *entity label*, not necessarily corresponding to an exact text span, (ii) a concise *description*, and (iii) a list of *types*

[8] https://github.com/SandroGT/KG-LLM-Prompting

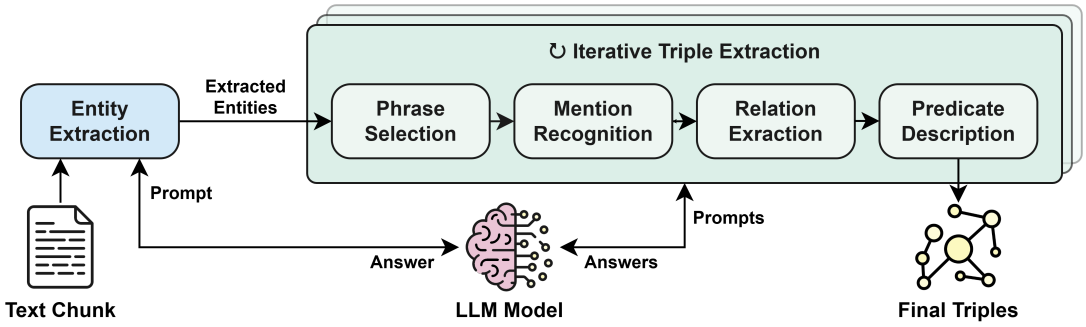

Figure 3.2: High-level architecture of the extraction-first pipeline. The process includes an entity extraction stage and an iterative triple extraction stage. The diagram shows how entity descriptions and types support context-aware relation discovery while constraining hallucinations.

or *hypernyms*. This representation provides contextual and typological information that supports the subsequent structuring stages.

Entities are identified through direct interaction with a large language model. A system prompt defines what constitutes an *entity* and instructs the model to retrieve mentions following the enriched representation above. To ensure consistent behavior, the generation parameters are fixed during all runs. Exploratory qualitative analysis based on manual inspection indicated that providing explicit guidance on what constitutes an entity leads the model to focus more consistently on concrete nouns and named entities, yielding richer and more reliable extractions. The resulting set of entities is denoted as E. The full prompt is reported in Section A.2.1.

Iterative Triple Extraction

The second stage performs an appropriate characterization of triples and predicates. Its purpose is to represent the relations between pairs of entities (subject and object) through suitably defined predicate labels and corresponding textual descriptions that capture the general meaning of the relationship. This design ensures that each extracted relation is both machine-processable and human-interpretable.

The generation of triples is constrained to the previously extracted entities, minimizing hallucinations by ensuring that all relations refer to entities already identified in E. Generating triples that explicitly and exclusively refer to E is challenging for LLMs, especially when E contains many elements, as the model may attempt to introduce unseen entities. To address this, an iterative strategy is adopted, focusing on one entity $e_i \in E$ at each iteration i. This reduces task complexity and increases the accuracy of relation extraction. Each iteration comprises four steps: *Phrase Selection*, *Mention Recognition*, *Relation Extraction*, and *Predicate Description*.

Phrase Selection. Rather than processing the entire text T, a smaller context T_i^G is defined for each entity e_i. This excerpt summarizes the information about e_i and reduces the model's reasoning load, improving reliability. Although several summarization strategies were considered, a practical and simple solution was adopted: using the entity descriptions produced in the previous stage as T_i^G, which naturally capture the local context of each entity. Accordingly, this stage does not rely on an explicit prompt.

Mention Recognition. This step identifies which entities from E are mentioned in T_i^G, defining the subset $E_i^G \subseteq E$. The LLM is prompted with both the complete list E and the text excerpt T_i^G, and instructed to return the same list annotated with a "yes/no" marker for each entity. Those marked "yes" form the subset E_i^G of entities co-occurring with e_i in the local context. The full prompt is reported in Section A.2.2.

Relation Extraction. Relations are then extracted using the restricted content provided by T_i^G and E_i^G. The model is prompted to generate the relations between entities in E_i^G that are expressed in T_i^G, representing them as RDF triples of the form ⟨subject, predicate, object⟩. The prompt explicitly defines the notion of an *expressive predicate*, guiding the model to generate predicates that capture the relationship meaningfully while avoiding overly specific phrasing. This approach encourages predicate canonicalization, improving the reusability and generalization of the resulting triples. The full prompt is reported in Section A.2.3.

Predicate Description. Finally, each unique predicate is described to enhance interpretability. For every predicate identified in the previous step, the LLM is prompted to produce a short textual definition contextualized by the excerpt T_i^G and the corresponding triples R_i^G. The output consists of pairs ⟨predicate, description⟩, which provide human-readable explanations of the extracted relations. The resulting sets of triples and predicate descriptions together form a document-level, semantically enriched representation suitable for Knowledge Graph construction. The full prompt is reported in Section A.2.4.

3.2.2 Experiments

The effectiveness of the extraction-first pipeline is evaluated through a series of experiments involving both human and automatic assessment procedures. The large language model used for the experiments is GPT-3.5 (specifically the GPT-3.5-turbo-0301 release[9]), with the temperature parameter set to zero to approximate deterministic behavior. Although OpenAI does not provide a fully greedy decoding

[9] `https://platform.openai.com/docs/models/gpt-3-5-turbo`

mode, this configuration minimizes stochasticity and aims for consistent outputs across runs.

Datasets

The proposed tool extracts entities and predicates enriched with contextual descriptions derived from in-text knowledge. This task cannot be performed consistently if the source material lacks sufficient detail or contextual information, as often occurs with existing datasets composed of isolated sentences or short paragraphs. To address this limitation, a dedicated dataset was created, denoted as ST_WEB, by collecting complete webpages from the English version of the *Sardegna Turismo* portal[10]. The corpus describes tourist destinations and points of interest in Sardinia and comprises 44 documents (888 sentences) covering topics related to culture and tourism.

To further assess the capability of the approach to enrich existing Knowledge Graphs, an additional dataset was derived from the REBEL corpus[11] [22], consisting of 148 annotated Wikipedia documents (2,803 sentences) on diverse topics. To contain both the computational cost of LLM-based extraction and the effort required for manual evaluation, experiments were conducted on a representative subset of the data. In particular, 20 documents, referred to as REBEL_20, were selected for manual evaluation.

Evaluation

Two complementary types of evaluation are carried out: a human assessment, applied to the ST_WEB and REBEL_20 datasets, and an automated evaluation, applied to the entire REBEL dataset, aimed at measuring the quantity and quality of additional information retrieved by the proposed strategy.

Human Assessment. Human assessors annotate each *entity*, *entity type*, and *triple* as *correct* or *incorrect*. An entity is correct if it is relevant to the textual context and explicitly mentioned in the input text. An entity type is correct if it accurately captures the class and context of the corresponding entity. A triple is correct if its predicate label expresses the relation and description appropriately, and both entities involved have been labeled as correct.

Each correct entity and triple also receives a Boolean annotation a_σ, used to evaluate whether the LLM retrieves the information directly from the text ($a_\sigma = true$) or draws on its internal knowledge ($a_\sigma = false$). Assessors additionally identify a set of *missed entities*, that is, relevant entities present in the input text but not retrieved by the model. This list is defined by considering all entity types having at least two associated entities to form a reference schema.

[10] `https://www.sardegnaturismo.it/en/`

[11] `https://osf.io/4x3r9/?view_only=87e7af84c0564bd1b3eadff23e4b7e54`

For the REBEL dataset, assessors introduce an additional Boolean annotation a_γ to evaluate whether each LLM-generated entity or triple corresponds to a concept annotated in the original dataset ($a_\gamma = true$), even if expressed in a different form, or not ($a_\gamma = false$).

From these annotations, classical confusion matrix entries are derived. Each correct component is a *true positive* (TP), each incorrect item is a *false positive* (FP), and each missed entity is a *false negative* (FN). Missing entities are identified exclusively through human annotations, disregarding those in the REBEL dataset, since its schema may not align with that implicitly adopted by the LLM. Counting them as missing would therefore be inconsistent.

These quantities permit the computation of the well-known metrics of *precision* (P), *recall* (R), and *F-score* (F_1), defined as follows[12]:

$$P = \frac{TP}{TP + FP} \tag{3.2}$$

$$R = \frac{TP}{TP + FN} \tag{3.3}$$

$$F_1 = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{3.4}$$

The capability of the model to infer additional information is assessed through the a_σ annotations, defining a score σ as in eq. (3.5), corresponding to the percentage of correct information derived from the LLM's internal knowledge rather than from the input text. Using the a_γ annotations on the REBEL dataset, the *knowledge enhancement* capability of the tool is quantified by the score γ, defined in eq. (3.6), representing the percentage of correct information not mentioned in the original dataset.

$$\sigma = \frac{TP \wedge \neg a_\sigma}{TP} \tag{3.5}$$

$$\gamma = \frac{TP \wedge \neg a_\gamma}{TP} \tag{3.6}$$

The following metrics are therefore computed:

- P^E, R^E, F_1^E: precision, recall, and F-score for entity extraction;
- P_a^T, P_f^T: precision of entity typing, considering all extracted types for each entity and only the first type, respectively[13];

[12]Note that R and F_1 can be computed only for entities, as the present work considers missing entities exclusively.

[13]The LLM outputs a list of types, which can be evaluated either in full or by considering only its first entry.

- P^R: precision of relation extraction;
- σ^E, σ^R: σ-scores for entity descriptions and relation extraction;
- γ^E, γ^R: γ-scores for entity extraction and relation extraction.

Automated Assessment. An automated evaluation is also conducted to estimate the knowledge enrichment potential by exploiting the ground truth provided in the REBEL dataset. Unlike human evaluation, classical confusion-matrix metrics cannot be applied, as they require exact matches between model-generated and ground-truth triples. Triples produced by LLMs may be correct and informative but represented differently from those in the reference dataset, which traditional metrics would incorrectly classify as false positives. To address this limitation, two complementary metrics introduced by [71] are adopted.

The first metric is the *Topical Similarity Score* (T), which measures the information abundance of the extracted triples compared with the source text. It relies on a Latent Dirichlet Allocation (LDA) model [16] for topic modeling, generating a probability distribution representing the alignment of a text with a set of N abstract topics (LDA_N). All extracted triples, represented as strings, are concatenated into a text T_D, whose LDA representation is compared with that of the source text D as shown in eq. (3.7). A high value of T indicates that the triples and the text exhibit similar alignment with the same abstract topics, reflecting effective information extraction.

$$T = \exp\left(-\sum_{i=1}^{N} LDA_N(D)_i \cdot \log\left(\frac{LDA_N(D)_i}{LDA_N(T_D)_i}\right)\right) \tag{3.7}$$

The second metric is the *Uniqueness Score* (U), which evaluates the diversity of the extracted triples by measuring the percentage of triple pairs that are semantically distinct. Each of the n extracted triples, represented as a string, is encoded as a vector v using embeddings. A pair of triples (v_i, v_j) is considered distinct if their cosine similarity is below a given threshold θ. The final score is computed as in eq. (3.8), where a high U value indicates that the triples convey non-overlapping information.

$$U = \frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{i \neq j}^{n} \left(\begin{cases} 1 & \text{if } CosSim(v_i, v_j) < \theta \\ 0 & \text{otherwise} \end{cases} \right) \tag{3.8}$$

The scores obtained from the model outputs (T_L and U_L) are compared against those derived from the ground truth triples (T_G and U_G). The implementation of these metrics is publicly available in the project repository[14].

[14] `https://github.com/SandroGT/KG-LLM-Prompting/tree/main/dataset`

Results

The extraction-first pipeline produced a total of 761 entities and 640 triples from the ST_WEB dataset, 2,845 entities and 2,024 triples from the REBEL dataset, and 379 entities and 329 triples from REBEL_20. The evaluation focuses exclusively on this system, since, to the best of current knowledge, no other state-of-the-art methods employ an equivalent extraction strategy that would allow for a direct comparison. The results of the manual evaluation are reported in Table 3.8.

Dataset	P^E	R^E	F_1^E	P_a^T	P_f^T	P^R	σ^E	σ^R	γ^E	γ^R
ST_WEB	0.97	0.94	0.96	0.86	0.95	0.75	0.09	0.00	-	-
REBEL_20	0.98	0.90	0.93	0.78	0.94	0.84	0.32	0.07	0.45	0.95

Table 3.8: Evaluation results from manual annotations.

The results demonstrate the effectiveness of the proposed approach, particularly in identifying meaningful entities, as indicated by the high values of P^E, R^E, and F_1^E across both datasets. Entity typing also achieves strong performance, with adequate accuracy when all retrieved types are considered (P_a^T) and substantially higher scores when only the first predicted type is evaluated (P_f^T), showing that the prompting strategy effectively captures entity classes. Relation extraction achieves satisfactory precision, as reflected in the P^R values.

The guidance provided to the LLM toward a text-centric focus proved particularly effective for the ST_WEB dataset, where additional information generated beyond the source text accounts for only 9% of entity descriptions. The REBEL dataset exhibits higher values (32% for entity descriptions and 7% for relations), likely due to the characteristics of its texts, where mentioned entities often lack sufficient contextual detail for generating full descriptions. Refining the entity extraction prompt and the phrase selection strategy is expected to mitigate this issue without altering the overall pipeline design.

The γ scores confirm that the methodology significantly enhances existing Knowledge Graphs, particularly in identifying new relevant entities not originally present in the underlying representation.

For the automated evaluation on the REBEL dataset, the *Topical Similarity* scores reported in Table 3.9 vary with the number of LDA topics (N). The results indicate that the extracted triples are better aligned with the dominant topics of the input texts than those present in the ground truth.

The *Uniqueness* scores, presented in Table 3.10, further assess the diversity of the extracted triples. A total of 2,024 triples were generated compared with 615 annotated in the ground truth. The higher U values obtained for the extracted triples indicate that the proposed method provides a broader and more diverse representation of the available information, maintaining uniqueness even at low similarity thresholds (θ).

	$N = 5$	$N = 10$	$N = 20$	$N = 30$	$N = 40$	$N = 50$	$N = 75$	$N = 100$
T_L	**0.70**	**0.62**	**0.58**	**0.45**	**0.44**	**0.40**	**0.30**	**0.25**
T_G	0.64	0.50	0.51	0.36	0.31	0.29	0.19	0.15

Table 3.9: Topical similarity results comparing extracted triples (T_L) and ground truth triples (T_G) on the REBEL dataset.

	$\theta = 0.70$	$\theta = 0.75$	$\theta = 0.80$	$\theta = 0.85$	$\theta = 0.90$	$\theta = 0.95$
U_L	**0.84**	**0.89**	**0.92**	**0.95**	**0.98**	**0.99**
U_G	0.67	0.77	0.84	0.88	0.93	0.98

Table 3.10: Uniqueness results comparing extracted triples (U_L) and ground truth triples (U_G) on the REBEL dataset.

Figure 3.3 illustrates an example subgraph extracted from the ST_WEB dataset. Yellow nodes correspond to entities, and pink nodes represent their associated types.

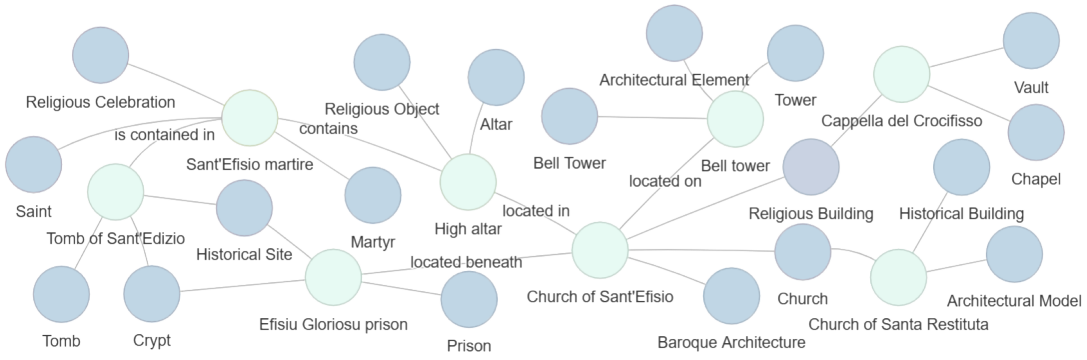

Figure 3.3: Example of an extracted subgraph from the ST_WEB dataset, where yellow nodes represent entities and pink nodes indicate types.

3.2.3 Summary

The extraction-first approach described in this section demonstrates how Large Language Models can directly generate and enrich Knowledge Graphs without predefined schemas. By leveraging zero-shot prompting strategies, the pipeline extracts entities and relations from raw text while providing enriched representations through descriptive and typological information. The resulting triples capture semantically coherent connections between entities and predicates, enabling a more expressive and interpretable structuring of textual knowledge. Experimental results highlight strong performance in entity identification and relation extraction, confirming the

validity of this method for producing detailed and context-aware graph representations. This perspective complements the schema-first strategy by emphasizing data-driven structuring and will be further discussed in the following comparative analysis.

3.3 Comparative analysis and discussion

The *schema-first* and *extraction-first* strategies represent two complementary perspectives on how LLMs can support the transformation of unstructured text into structured knowledge. Their comparison highlights distinct methodological assumptions and reveals a progressive convergence toward a hybrid paradigm.

The schema-first approach assumes that a domain should be formally characterized before any extraction of entities or relations is attempted. This perspective offers an advantage in terms of early semantic consistency, since extracted elements are expected to conform to a shared conceptual frame that facilitates uniform annotation and interpretability. In practice, two main difficulties arise.

First, defining a schema before observing actual content imposes strong top–down assumptions. An LLM cannot be exposed to an entire corpus at once and reliably infer a coherent schema, so the process must be divided into smaller portions and subsequently merged. In earlier experiments, this led to the use of keyword extraction as a practical summary of textual content, which was then aggregated through clustering to derive an initial schema. Although effective in producing broad domain categories, the reliance on keywords limited contextual richness. From a linguistic perspective, keywords can be viewed as raw precursors of entities, providing only partial semantic information.

A second limitation concerns the enforcement of schema constraints during extraction. Constraining an LLM to adhere strictly to a predefined schema requires elaborate prompting and careful conditioning, as models tend to generate new entities or relations beyond the intended scope. Even when explicitly instructed to restrict extraction to a given pre-identified set, LLMs tend to introduce additional elements, as observed in the *Iterative Triple Extraction* phase described in Section 3.2.1. In that stage, the model was instructed to form relations only among a predefined set of entities identified within the same document, yet it occasionally introduced new entities, revealing the intrinsic difficulty of binding generative behavior to fixed symbolic definitions.

By contrast, the extraction-first approach lets entities and relations emerge directly from text without relying on a predefined type system. It leverages the model's capacity to identify semantically salient elements within context, producing richer and more descriptive representations. Because extracted entities already include textual descriptions and implicit type cues, they provide a better foundation for deriving higher-level abstractions in a subsequent stage that mirrors the schema-first logic. In this view, the schema arises as a data-driven generalization built upon the

extracted content rather than as a prior constraint, offering a more natural starting point for knowledge structuring.

3.3.1 Toward hierarchical refinement

Building on these observations, subsequent experiments extend the extraction-first pipeline with a refinement phase designed to infer explicit hierarchies among the extracted types. This stage can be regarded as an evolution of the schema-first approach, where the definition of the schema is reconstructed *a posteriori* from the extracted entities and types rather than induced from keywords. The process remains exploratory and has not yet been systematically evaluated, but it illustrates how the two previously strategies can be integrated within a single workflow.

Preliminary tests were conducted on a subset of geographically themed sentences from the *WikinRE* dataset, pre-classified by an LLM. Consider the example:

```
Ramakuppam is a village in Chittoor district of the
Indian state of Andhra Pradesh.
```

The extraction phase, executed as in the extraction-first approach and using the *Qwen-14B-Instruct* model, identifies entities and their candidate types:

```
Ramakuppam [Village, Location]
Chittoor district [District, Location]
Andhra Pradesh [State, Location]
```

and relations such as:

```
(Ramakuppam, is located in, Chittoor district)
(Chittoor district, is located in, Andhra Pradesh)
```

Across the entire geographic topic, more than $1,500$ entity mentions are extracted, providing a detailed description of the domain that can serve as a foundation for ontology refinement.

Before hierarchical relations are inferred, the type labels are normalized. Each label is divided into words and each word is lemmatized to ensure lexical consistency. This produces roughly 120 unique types, for each of which frequency counts are computed. Examples of frequent and rare types are:

```
Location: 1516
District: 547
Country: 510
...
Neighborhood: 1
Ridge: 1
Street: 1
```

Low-frequency types are pruned, unless they constitute the only label attached to an entity, leaving about 60 dominant categories. These frequencies already hint

at a latent hierarchy: general categories such as `Location` or `Region` appear more frequently than finer ones such as `Village` or `County`.

Type frequency alone, however, is insufficient to define hierarchical relations. To restrict the number of possible comparisons, candidate type pairs are first identified. Two types are considered a candidate pair for further investigation when they co-occur in the same entity annotation (for example, `[Village, Location]` assigned to `Ramakuppam`) or when their embeddings exhibit high cosine similarity. This loose selection favours recall over precision, ensuring that potentially related categories are retained for later reasoning.

Each candidate pair (A, B) is then evaluated by a dedicated reasoning module. An LLM receives both type labels and representative instances to answer a fixed set of four ontological questions, each requiring a binary (`True`/`False`) judgment:

1. (A) is a **hypernym** of (B): every instance of (B) is also an instance of (A). Example: (A) `Vehicle` is a hypernym of (B) `Car`.

2. (A) is a **hyponym** of (B): every instance of (A) is also an instance of (B). Example: (A) `Apple` is a hyponym of (B) `Fruit`.

3. (A) is a **sub-part** of (B): individuals of (A) are parts of individuals of (B). Example: (A) `Singer` is a sub-part of (B) `Musical Band`.

4. (A) is **composed of** (B): individuals of (A) are composed of or structured from individuals of (B). Example: (A) `Triangle` is composed of (B) `Cathetus`.

Expressing these questions through inclusion tests rather than open-ended descriptions reduces ambiguity and constrains generative behaviour. Relations of type (1) and (2) define hierarchical directionality, while answers of type (3) or (4) indicate partonomic connections. If both inclusion questions are answered *True* and no partonomic relation is detected, the two types are considered equivalent. Equivalence groups are subsequently merged, and an additional LLM query selects a canonical label for each group.

The remaining hypernym and hyponym links are then organized into a hierarchical structure. Each relation contributes a directed edge, ensuring that higher-level concepts dominate broader subtrees. An excerpt of the resulting geographic hierarchy is shown in Figure 3.4, where coherent branches emerge from the automatic reasoning process.

Although preliminary, these results already show how instance-grounded reasoning can transform noisy type lists into coherent taxonomic structures. The hierarchies display clearer boundaries and fewer redundant categories than those obtained from keyword clustering, aligning more closely with domain-specific taxonomies. They indicate that abstraction informed by factual extraction yields more interpretable and domain-faithful schemas, combining the contextual grounding of extraction-first with the organizing discipline of schema-first reasoning.

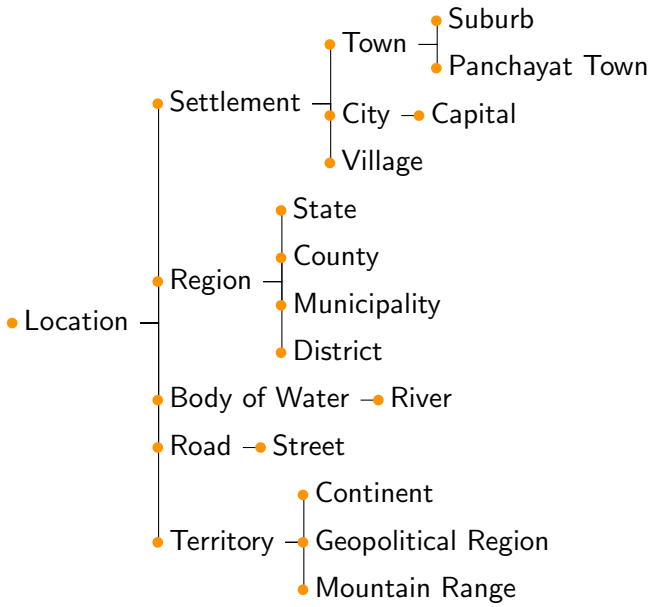

Figure 3.4: Excerpt of the automatically induced geographic hierarchy, obtained on geographic sentences from *wiki-nre*.

3.3.2 Toward ontology induction

The refinement experiments described above represent an initial step toward the broader objective of ontology induction. Beyond the organization of types into hierarchies, the same reasoning principle can be extended to the relational dimension, allowing the construction of complementary hierarchies for predicates.

Each entity and predicate can be enriched with a short natural-language description. For predicates, analysing their typical subjects and objects enables an LLM to infer candidate *domain* and *range* axioms, effectively linking relations to the hierarchical structure of types. Through this process, a coherent conceptual layer emerges above the extracted factual triples, progressively connecting instances, categories, and relations within a unified semantic framework. Such integration marks the transition from a taxonomy or schema toward a true ontology.

It is important to consider this refinement as a support mechanism rather than a fully autonomous process. Ontology engineering remains inherently iterative and benefits from expert supervision. The system therefore aims to provide an informed initial structure that can be manually reviewed and extended, reducing the effort required to construct a coherent conceptual model from unstructured data.

Although the present results are preliminary, they offer a concrete indication that reasoning-driven refinement can enhance the semantic organization of automatically generated knowledge graphs. The experiments suggest that knowledge structuring is most effective when it begins with concrete entity and relation extrac-

tion and proceeds toward increasingly abstract representations guided by reasoning. Such a workflow preserves the contextual grounding of the extraction-first paradigm while reintroducing the organizing discipline of schema-first modeling, pointing to a convergence toward hybrid, reasoning-based pipelines for large-scale ontology construction.

Chapter 4

Knowledge Retrieval

With the structuring of textual information into coherent Knowledge Graphs established, the next challenge concerns how such structured knowledge can be effectively accessed and queried. The practical relevance of Knowledge Graphs depends on the ability to retrieve and exploit the information they encode.

A complementary problem to knowledge structuring lies in making that structured knowledge effectively accessible and usable in a natural and intuitive manner. Traditional query languages such as SPARQL provide a precise and expressive means to interrogate Knowledge Graphs, yet their formal syntax and dependence on domain-specific schemas restrict their use to technically proficient users. Bridging the gap between natural language and formal query syntax thus represents a crucial step toward enabling direct and inclusive access to structured knowledge. Large Language Models (LLMs), with their capacity to interpret and generate both code and structured text, have recently shown promise in translating natural-language questions into executable queries.

This chapter investigates their potential as intermediaries between users and Knowledge Graphs, assessing to what extent general-purpose LLMs can perform the translation from natural language to SPARQL under zero-shot conditions. The analysis characterizes the strengths and limitations of current models and complements the knowledge-structuring experiments discussed in Chapter 3.

4.1 Methodology

The adopted methodology evaluates the capability and consistency of Large Language Models (LLMs) in generating SPARQL queries from natural language inputs. The objective is to assess how effectively different models perform this translation task across domains without domain-specific fine-tuning.

To this end, the following introduce the key formal notions underlying the evaluation approach, provide an overview of the evaluation process, and discuss the prompting strategy design.

4.1.1 Problem formalization

The formal groundwork establishes the basis for the evaluation criteria. It defines the NL-to-SPARQL translation task, the method for comparing the results of two SPARQL queries to determine their equivalence, a voting strategy for selecting the most consistent translated query, and a formal notion of *query correctness*. These aspects are detailed below.

NL-to-SPARQL translation The NL-to-SPARQL task is formalized as a *single-pass translation* problem, defined by a function φ that, given a user's natural language question $\mathcal{Q}_U$ and the associated KG ontology $\mathcal{O}$, returns the expected SPARQL query $\mathcal{Q}_{Exp}$:

$$\mathcal{Q}_{Exp}^{(i)} = \varphi(\mathcal{Q}_U^{(i)}, \mathcal{O}^{(i)}) \quad \forall i \in \mathbb{N}$$

In a multi-domain setting, where both knowledge graph structures and user questions vary, $\mathcal{Q}_U^{(i)}$ denotes the i-th natural language question and $\mathcal{O}^{(i)}$ the corresponding ontology. In this formulation, the function φ is approximated by an LLM guided through carefully designed prompt instructions.

Query equivalence A central aspect of the evaluation concerns comparing LLM-generated SPARQL queries either against each other or against a corresponding ground-truth query. Two queries q_i and q_j are considered *equivalent* if their respective result lists $\mathcal{L}_{q_i}$ and $\mathcal{L}_{q_j}$ are equivalent, that is, $\mathcal{L}_{q_i} \equiv \mathcal{L}_{q_j} \implies q_i \equiv q_j$.

A result list $\mathcal{L}$ is the sequence of records obtained by executing a query q on the respective KG. Each record is a tuple containing the graph variables or expressions stated in the query return statement.

Equivalence between two result lists $\mathcal{L}_{q_i}$ and $\mathcal{L}_{q_j}$ is determined as follows:

1. If the list sizes differ (i.e., $|\mathcal{L}_{q_i}| \neq |\mathcal{L}_{q_j}|$), the result lists are considered *not equivalent* ($\mathcal{L}_{q_i} \not\equiv \mathcal{L}_{q_j}$).

2. When the lists have the same size, including when both are empty, the comparison proceeds record by record. A record is considered equal to another if it contains the same elements, regardless of order. The overall equivalence depends on the query structure: (a) when q_i or q_j includes an `ORDER BY` clause, equivalence holds if records in corresponding positions are equal; (b) when ordering is not specified, equivalence holds if each record in $\mathcal{L}_{q_i}$ has a matching record in $\mathcal{L}_{q_j}$ and vice versa.

If a query fails to execute, resulting in no result list $\mathcal{L}$, equivalence is defined as follows: if only one query fails while the other succeeds, the queries are *not equivalent*; if both fail to execute, they are considered *equivalent*. The latter case applies only when comparing LLM-generated queries, as ground-truth queries are always executed successfully.

Voting strategy Because LLM outputs can vary across runs, multiple generations are used to assess consistency. Each model is prompted k times for each dataset query $q^{(i)}$, producing a set $\Lambda^{(i)} = \{q_1^{(i)}, \ldots, q_k^{(i)}\}$ of generated queries for the corresponding pair $(\mathcal{Q}_U^{(i)}, \mathcal{O}^{(i)})$ of user question and ontology. For a given user question $\mathcal{Q}_U$, the final model output $\mathcal{Q}_C$ is defined as the most frequently occurring query in Λ, according to a function $\mathcal{C}(\Lambda^*, q^*)$ that counts the number of $q_j \in \Lambda^*$ such that $q_j \equiv q^*$. If no query obtains a majority of occurrences in Λ, no answer is assumed for $\mathcal{Q}_U$, meaning that no result list is produced. The selection of $\mathcal{Q}_C$ is therefore formalized as:

$$\mathcal{Q}_C = \begin{cases} \hat{q}, & \text{if } \hat{q} \in S_m,\ |S_m| = 1,\ m > 1 \\ \emptyset, & \text{otherwise} \end{cases}$$

Where:

$$S_m = \{q \in \Lambda \mid \mathcal{C}(\Lambda, q) = m\}, \quad m = \max\{\mathcal{C}(\Lambda, q_1), \ldots, \mathcal{C}(\Lambda, q_k)\}.$$

Query correctness The correctness of a model-generated SPARQL query is defined by its equivalence with the corresponding ground-truth query. For each test question $\mathcal{Q}_U^{(i)}$, the model produces multiple candidate queries, and the final query $\mathcal{Q}_C^{(i)}$, selected through the voting strategy above, is considered *correct* if and only if it is equivalent to the annotated expected query, that is, $\mathcal{Q}_C^{(i)} \equiv \mathcal{Q}_{Exp}^{(i)}$.

4.1.2 Methodology overview

A high-level overview of the evaluation process is shown in Figure 4.1 and described below. The approach relies on a collection of datasets that include knowledge graphs, natural language questions, and corresponding ground-truth (GT) SPARQL queries (highlighted in yellow at the bottom left of the figure). It also involves a set of representative LLMs (in red, top left) and a selection of prompt templates (in orange, center left) used to guide the models in generating SPARQL queries. By iterating over all combinations of these inputs, SPARQL queries are produced and assessed for correctness (shown in green throughout the main body on the right side of the figure). The conceptual steps, numbered and represented by ellipses, are outlined in the following paragraphs.

(1) Formatting Several prompt templates are devised to guide LLMs in the NL-to-SPARQL task. For each selected template, the *Formatting* step constructs a ready-to-use prompt by combining the user's natural language question with ontology information, that is, the classes and relations relevant to the query. The rationale behind the prompting strategy is discussed in Section 4.1.3, while implementation details are provided in Section 4.2.1.

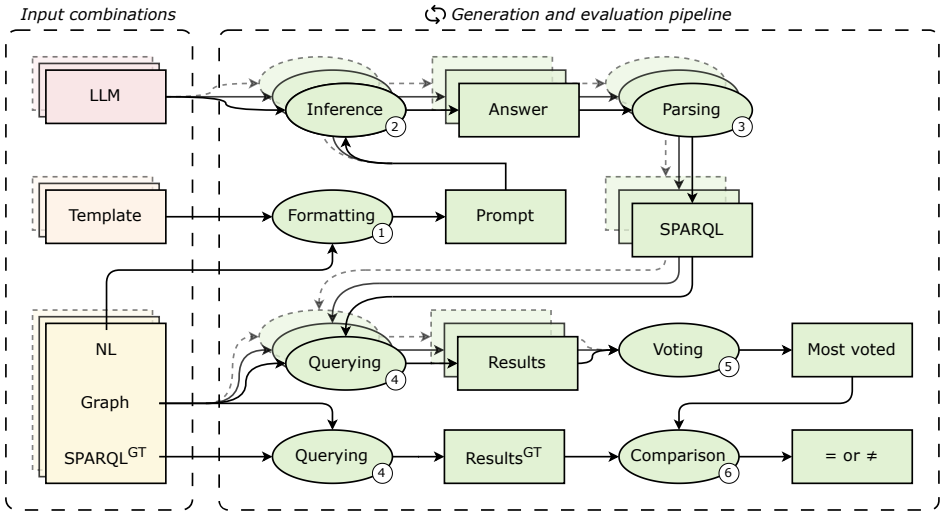

Figure 4.1: Rectangles represent data objects; ellipses represent conceptual steps. NL-to-SPARQL translation and query correctness evaluation across multiple LLMs, domain graphs, and prompt templates.

(2) Inference Each formatted prompt is submitted to the selected LLM, which returns a response containing the generated SPARQL query. Two categories of prompts are used in the experiments: non-reasoning prompts, which directly request a SPARQL query, and reasoning prompts, which encourage the model to produce intermediate reasoning before the final query. The selection of LLMs and their configuration are detailed in Section 4.2.2 and Section 4.2.3, respectively.

(3) Parsing Although the prompts explicitly instruct models to return only the SPARQL query, additional text such as explanations, comments, or code blocks is often included. To ensure robustness and consistency, the *Parsing* step extracts only the portion of the model output that conforms to valid SPARQL syntax, isolating the query from any extraneous content.

(4) Querying To evaluate the quality of a generated SPARQL query, its execution result is compared with that of the corresponding ground-truth query. This is done using a SPARQL engine (described in Section 4.2.6) to execute both queries over the same knowledge graph. Differences between the result sets indicate potential shortcomings in the generated query, such as structural errors or incomplete retrieval.

(5) Voting Each LLM is prompted multiple times per input to assess model consistency and improve reliability. In the experiments, $k = 3$ generations are used for non-reasoning prompts and $k = 1$ for reasoning prompts, balancing reliability with

cost and latency considerations. This produces a set of candidate SPARQL queries, from which a final output is selected through a majority voting mechanism based on query equivalence. The formal definition of this voting procedure is provided in Section 4.1.1.

(6) Comparison The final step verifies whether the selected query correctly reflects the user question. This is achieved by comparing its execution result with that of the ground-truth query, yielding a binary correctness label as defined in Section 4.1.1.

4.1.3 Prompting strategy

To guide LLMs in the NL-to-SPARQL task, prompt templates are defined with placeholders for the user's natural language question and the relevant ontology information, namely the classes and relations referenced in the query. These templates are dynamically filled for each query and graph.

As discussed in Chapter 3, practical applications often begin with unstructured textual data that must be organized into structured, machine-interpretable knowledge graphs through processes such as ontology learning and knowledge graph construction. When such graphs are newly created, typically from proprietary or domain-specific sources, they lack annotated training data or example queries that could support supervised learning or fine-tuned generation. In this context, a zero-shot approach is adopted, avoiding task-specific demonstrations and handcrafted examples.

While few-shot prompting can provide intermediate supervision without full fine-tuning, it still depends on a diverse set of curated examples and an effective retrieval mechanism. Inadequate examples may mislead the model, whereas overly specific ones can reduce the task to pattern replication. Both conditions risk limiting the model's ability to genuinely interpret and generate SPARQL queries, which is the capability under evaluation here.

Although zero-shot prompting may underperform compared to few-shot techniques, it provides a more appropriate framework for assessment. It (i) establishes a simple and reproducible baseline that can later be extended with reasoning-based prompting strategies [82, 167, 165, 172], and (ii) does not rely on task-specific training data, ensuring applicability across domains, unlike fine-tuned models that often overfit to specific graph structures.

In summary, to assess the generalization capabilities of LLMs in the NL-to-SPARQL task under realistic data constraints, a zero-shot prompting strategy is adopted. The design and implementation of the prompt templates employed in the experiments are discussed in Section 4.2.1.

4.2 Experimental setup

The experiments evaluate the performance of various LLMs in the NL-to-SPARQL task. The experimental process involves several components: the design of prompt templates (Section 4.2.1); the selection of models (Section 4.2.2); the configuration of LLMs (Section 4.2.3); the choice and processing of datasets (Sections 4.2.4 and 4.2.5); the inclusion of an additional LLM-based tool as a reference (Section 4.2.8); and the evaluation of model performance using multiple measures (Section 4.2.7). All experiments were implemented in *Python* to ensure reproducibility and consistent execution across models.

4.2.1 Prompt template design

Obtaining an effective zero-shot prompt for evaluation, in accordance with the strategy discussed in Section 4.1.3, is not trivial. Different formulations of instructions can either enhance or degrade output quality. While there exist general guidelines for prompt construction that apply to many LLMs[1], each model has unique characteristics and may require prompt tuning to perform optimally.

Ideally, designing a dedicated prompt for each LLM could maximize performance. However, this would compromise the fairness of comparison by introducing variability across models. Conversely, using a single prompt for all LLMs ensures uniformity but risks favoring some models over others due to potential mismatches in compatibility. To balance these concerns, a small set of prompt templates is defined that maintains consistent structure while offering limited flexibility.

In LLM prompting, the *user* role typically conveys the natural language question, while the *system* role provides contextual instructions that guide the model's behavior. The templates include placeholders for both the user question and the ontology information, differing mainly in the level of detail and structure of the system message:

- (`BASIC`) provides minimal instructions, requesting a syntactically correct query based on the provided question and ontology. It includes only the labels of classes and relations from the input ontology. An example is provided in Section A.3.1.
- (`DETAILED`) offers a more structured guide, outlining specific steps to analyze the question and construct the query. It emphasizes elements such as return values, quantifiers, superlatives, aggregations, and grouping. In addition to class and relation labels, it includes domain and range axioms for each relation. An example is provided in Section A.3.2.
- (`COT`) follows a similar structure to `DETAILED`, but instead of providing static advice, it elicits reasoning from the model. This template encourages the LLM

[1] `https://huggingface.co/docs/transformers/tasks/prompting`

to analyze each step, reflect on its reasoning, and include brief explanatory comments about its decision process. An example is provided in Section A.3.3.

The development of these templates follows a flexible, iterative process common in prompt engineering. Initial drafts were designed to establish distinct levels of detail in the task description, process steps, and ontology format. A small subset of queries, on the order of tens, was used to test the models' adherence to instructions and consistency in output formatting. The iterative refinement focused on improving SPARQL syntax validity and response accuracy across different queries and multiple repetitions of the same input. During this stage, models from the GPT, Llama, and Phi families were evaluated.

The process concluded once further modifications no longer produced significant improvements, leading to the selection of the most contrasting and well-performing templates: `BASIC` and `DETAILED`. The `COT` template was subsequently derived from `DETAILED` by introducing explicit reasoning prompts that encourage the model to explain intermediate steps. This transition from static instructions to a more reflective, reasoning-oriented style was intended to enhance transparency and, potentially, accuracy.

4.2.2 LLM selection

To investigate the capabilities of LLMs in addressing the NL-to-SPARQL problem, a diverse set of instruction-tuned models was selected from several widely used families. The selection balances large-scale, state-of-the-art systems with more lightweight and locally executable alternatives.

The goal of this configuration is to cover a representative range of model sizes, architectural specializations, and deployment paradigms. The selected models include general-purpose, code-optimized, and reasoning-enhanced variants, providing a well-rounded view of performance across different capabilities and training focuses. Although many additional models could have been included, the selection was intentionally limited to maintain a manageable and coherent set for comparison, acknowledging the rapid evolution of the LLM landscape.

The *GPT* family, representing some of the largest and most widely deployed LLMs as a service, is included to reflect the state of the art in large-scale models. While *gpt-3.5-turbo* has been partially superseded by *gpt-4o-mini*, it is retained here due to its extensive use in the literature, which facilitates comparability with previous research.

To extend the analysis beyond commercial APIs, additional instruction-tuned models from other families were considered. These include several open-access models offering competitive performance with reduced computational requirements, enabling local execution and cost-efficient experimentation. The selection also incorporates models with specialized training in coding or reasoning tasks, allowing the

evaluation to explore how such domain-specific optimizations influence the generation of SPARQL queries.

A particularly illustrative example is the *Qwen-32B* series, available in three variants: a general-purpose model, a coder-oriented version, and a reasoning-focused distillation model derived from DeepSeek. This configuration makes it possible to analyze whether coding or reasoning specializations contribute to improved NL-to-SPARQL translation.

The models evaluated in the experiments are summarized in Table 4.1.

Organization	**Model**	**Size**
OpenAI	gpt-3.5-turbo	-
	gpt-4o-mini	-
	gpt-4o	-
Meta	llama-3.1-8b	8B
	llama-3.3-70b	70B
Microsoft	phi-4-14b	14B
Cohere	c4ai-command-r-7b	7B
	c4ai-command-r-32b	32B
MistralAI	codestral-v0.1-22b	22B
	mistral-small-24b	24B
Alibaba	qwen-2.5-32b	32B
	qwen-2.5-coder-32b	32B
DeepSeek	deepseek-v2-coder-16b	16B
	deepseek-r1-qwen-32b	32B

Table 4.1: Models evaluated in the experiments, organized by provider and parameter size (billions). No official size information is available for the GPT family.

4.2.3 LLM settings

Each natural language question and the corresponding knowledge graph were provided to the models using three different prompts derived from the templates discussed in Section 4.2.1. To ensure fair and consistent conditions across models, decoder sampling parameters were standardized by setting *temperature* to 0.01 and *top-p* to 0.01 in all experiments.

The *temperature* and *top-p* parameters control the stochastic behavior of text generation during decoding. Their function and mathematical formulation have been discussed in Section 2.2.3, where they are described as mechanisms that balance determinism and diversity in autoregressive language models.

For the present task, model stochasticity is intentionally minimized to achieve behavior that is as deterministic as possible. Since the objective concerns question answering over structured data rather than open-ended generation, reduced ran-

domness ensures consistent and reproducible outputs. Setting *temperature* = 0 is mathematically undefined and, in practice, handled inconsistently across LLM implementations: some raise errors, others silently adjust the value, and some switch to greedy decoding.

Setting *temperature* = 0 is mathematically undefined and may be handled differently by decoding implementations (discussed in Section 2.2.3) that map model output distributions to token sequences. In practice, some implementations might raise errors, others implicitly adjust the parameter, and others fall back to greedy decoding. To avoid such implementation-dependent behavior, particularly in cases where decoding details are not exposed (e.g., cloud-based APIs), a non-zero temperature is used in these experiments.

In particular, to maintain comparability across systems, a low *temperature*=*top-p*=0.01 was adopted. This choice introduces negligible randomness while preserving compatibility between local and API-based models, including those provided by OpenAI.

Allowing a minimal degree of stochasticity also makes the model's inherent "confidence" observable rather than fully constrained. Each prompt was executed multiple times, as shown in Figure 4.1, to produce multiple candidate queries. The resulting outputs were compared to assess stability using the query equivalence criteria defined in Section 4.1.1.

OpenAI models were accessed through their official API,[2] while all other models were executed locally using the *transformers* [169] or *ollama*[3] libraries with 4-bit weight quantization. While less aggressive quantization levels (e.g., 8-bit or 16-bit) were feasible for some of the smaller models, adopting heterogeneous precision levels would have introduced an additional confounding factor, making cross-model comparisons less direct. Recent empirical studies indicate that modern 4-bit quantization techniques preserve model behavior closely to 8-bit and 16-bit precision on a wide range of language understanding and generation tasks [73]. Accordingly, 4-bit quantization was adopted to promote consistent execution conditions across all locally evaluated models.

4.2.4 Dataset selection

A major requirement of this work, which focuses on assessing LLMs in a zero-shot NL-to-SPARQL setting over diverse, custom-built knowledge graphs, is the selection of appropriate benchmark datasets. To this end, various datasets from the literature on Knowledge Graph Question Answering (KGQA) were analyzed, covering a spectrum of scenarios from large, well-established resources to smaller, domain-specific collections.

This exploration highlighted a key limitation: most widely used KGQA datasets,

[2] `https://platform.openai.com/docs/guides/chat-completions`
[3] `https://github.com/ollama/ollama`

such as *WebQuestions* [13], *LC-QuAD* [157, 38], and *QALD* [159], are tightly coupled with specific and extensively reused knowledge graphs like *Freebase*, *DBpedia*, or *Wikidata*. This coupling restricts their capacity to generalize across different ontologies and graph structures. Conversely, some newer datasets, such as *GrailQA* [54], attempt to reduce this dependence by diversifying training and test distributions, yet still rely on preexisting large-scale KGs [70].

Given these constraints, the focus was placed on datasets that span a broad range of domains and include natural language questions and SPARQL queries exhibiting diverse intents, formulations, and structural patterns. The datasets identified as most suitable for this scenario are:

- **Spider4SPARQL** [83]: a benchmark dataset featuring 166 knowledge graphs across 138 domains, containing 9,693 manually authored natural language questions and 4,721 unique SPARQL queries of varying complexity. The dataset is divided into training and development sets, with the development split, comprising 20 KGs and 1,034 NL–SPARQL pairs, used for testing in the present work.

- **Bestiary** [84]: a smaller dataset built around a single knowledge graph modeling the *Dungeons & Dragons* domain. Although considerably smaller than *Spider4SPARQL*, with 100 queries, it introduces complementary complexity, frequently involving multiple `FILTER` and `BIND` operations.

- **LC-QuAD** [157]: a well-known benchmark based on *DBpedia*, originally comprising 1,000 test queries that cover a broad range of topics and provide rich linguistic variety.

The combination of *Spider4SPARQL* and *Bestiary* offers several advantages:

(i) *Diversity*: the two datasets collectively span a wide range of domains and topics, encouraging model generalization.

(ii) *Uniqueness*: their associated knowledge graphs are not derived from heavily reused sources such as *DBpedia* or *Wikipedia*, which reduces the risk of bias arising from model pretraining exposure and helps ensure that query generation depends on reasoning rather than memorization.

(iii) *Practicality*: their compact ontologies fit entirely within the prompt context, eliminating the need for ontology retrieval or sampling and reducing confounding factors related to template formatting.

To improve completeness and test model robustness under less curated conditions, the *LC-QuAD* dataset is also included. Although it does not fully meet points (ii) and (iii), a lightweight restructuring (described in Section 4.2.5) was applied to

ensure consistency with the experimental assumptions, particularly regarding ontology compactness and prompt integration.

It is nevertheless acknowledged that these assumptions, especially the focus on compact, prompt-fit ontologies, introduce simplifications that may limit the generalizability of the findings. Real-world knowledge graphs often feature substantially more complex and extensive ontologies than those found in the selected datasets. Moreover, while *Spider4SPARQL* covers a wide variety of domains and query complexities, it still exhibits internal regularities in question phrasing and ontology structure. These characteristics, combined with the reliance on grammatically well-formed natural language queries, may reduce both linguistic and structural variability compared with real-world user inputs.

The inclusion of *LC-QuAD* partially mitigates these limitations by introducing greater diversity in SPARQL query forms and ontology terminology, while preserving the natural entity naming conventions and structural inconsistencies typical of real-world graphs such as *DBpedia*. Although this does not completely remove dataset-induced biases, it enables a partial assessment of model robustness under more realistic conditions. A broader discussion of dataset refinement and remaining limitations is provided in Section 4.2.5.

4.2.5 Dataset refinement

Although *Spider4SPARQL* and *Bestiary* are well suited for the present study, as they provide multiple graphs and good variation in query complexity, preliminary and qualitative analyses revealed that for several queries the generated outputs did not consistently match the expected complexity of the corresponding ground-truth queries. In particular, several LLMs exhibited difficulty in aligning concepts mentioned in the natural language questions with the appropriate classes and relations within the ontologies. This behavior often resulted in syntactically valid queries that followed correct structural patterns but misused classes or relations.

A detailed inspection of such cases indicated that these challenges were largely attributable to issues in the original ontology design. In some cases, class and property labels were misleading or ambiguous, while in others the modeling of information deviated from standard knowledge graph practices, such as by encoding information within IRIs. Although these irregularities can occur in real-world settings, they introduced an additional and unintended difficulty that risked obscuring the assessment of LLM query generation capabilities.

To avoid biasing the comparison across models, the prompts were kept fixed, and instead the datasets were refined to improve the clarity and consistency of their underlying ontologies. Each graph from both *Spider4SPARQL* and *Bestiary* was manually reviewed and adjusted to better align with common ontology modeling conventions. The main interventions are summarized below.

- **Spider4SPARQL**: domain axioms were added to support the `DETAILED`

prompt, and ambiguous labels for classes and properties were clarified. A recurring issue involved relations labeled as `ref-entity_id` (e.g., `ref-museum_id`, `ref-people_id`), which caused LLMs to interpret them as links to literal identifiers rather than other entities. To resolve this, the `_id` suffix was removed from all such labels. In other cases, relation names were not misleading but cryptic, such as the attribute `abandoned_yn`, an abbreviation of `abandoned_yes_or_no`, which was renamed to `abandoned_0_1` to indicate its binary encoding.

- **Bestiary**: new predicates were introduced (e.g., `beast_category`) to replace information previously embedded in IRIs, and explicit name attributes were added to most entities to make their identifiers directly queryable. For example, the ground-truth question "*How many types of robots are among creatures?*" originally required a `FILTER regex(str(?creatures), 'robot', 'i')` operation because the relevant information was encoded only in IRIs. The refined ontology exposed this information as explicit properties, simplifying query alignment.

Additionally, two empty graphs with no individuals were removed from *Spider4SPARQL*, and from both datasets all ground-truth queries that failed execution on the adopted SPARQL engine (see Section 4.2.6) due to syntax or structural errors were discarded. The final test folds therefore comprise 18 graphs and 886 queries from *Spider4SPARQL* and 92 queries from *Bestiary*, for a total of 19 graphs and 978 executable queries in the test set.

The refinement process involved both automatic and manual steps, with all modifications tracked systematically to ensure transparency and reproducibility.

LC-QuAD For *LC-QuAD*, a different approach was adopted. Rather than refining the ontology to better conform to LLM-friendly modeling conventions, no manual changes were made to its structure or query formulations. This decision preserves the original naming conventions and modeling patterns typical of *DBpedia*, providing a more realistic evaluation scenario.

To maintain compatibility with the experimental requirement of ontology compactness (see point (iii) in Section 4.2.4), a filtering and restructuring procedure was applied to reduce the original 1,000 test queries to 441. The procedure comprised the following steps:

- **Query executability**: only queries returning non-empty results when executed against *DBpedia* were retained, as the evaluation compares query answers rather than syntax. This filtering step produced the largest reduction in query count.

- **Ontological compactness**: each query was parsed to identify referenced entities, and their `rdf:type` values were mapped to coarse `schema.org` categories.

Queries were grouped by category, and compact sub-ontologies (typically three to ten classes and properties) were constructed to cover each group. Each sub-ontology includes all classes and properties required to execute its associated queries, along with a small amount of additional noise. Oversized categories were split into multiple sub-ontologies (e.g., `person`, `person_2`, etc.), while undersized ones were discarded.

No renaming or manual refinement of classes, properties, or instances was performed, allowing *LC-QuAD* to serve as a more naturalistic benchmark that complements the curated datasets.

4.2.6 Query engine

Preliminary experiments employed the *rdflib*[4] library to query the local graphs from *Spider4SPARQL* and *Bestiary*. However, this setup exhibited significant performance limitations, with execution times for complex queries (including some ground-truth examples) occasionally extending to several minutes. To improve efficiency, the querying process was migrated to *Apache Jena*[5], which substantially reduced execution times to a few seconds even for the most demanding queries. Jena was invoked from Python as a separate process, with the graph file and query passed as input parameters and the resulting answers returned for comparison.

For *LC-QuAD*, the evaluation required execution on *DBpedia*. Queries were therefore submitted to the public *DBpedia* SPARQL endpoint[6] using Python-based HTTP requests.

4.2.7 Evaluation measures

The following measures are used to evaluate the reliability, stability, and efficiency of LLMs in the NL-to-SPARQL generation task.

Accuracy

For each model, accuracy is computed as the ratio of correctly generated queries to the total number of generated queries:

$$accuracy = \frac{number\ of\ correctly\ generated\ queries}{total\ number\ of\ generated\ queries} \tag{4.1}$$

Following the query classification proposed by [83], queries are categorized by complexity based on the logical reasoning required for their formulation, which is often reflected in their structural patterns and constructs. Specifically, four main

[4] `https://pypi.org/project/rdflib/`
[5] `https://downloads.apache.org/jena/`
[6] `https://dbpedia.org/sparql`

categories are distinguished: queries involving a single triple pattern (`1 hop`), those including one or more filtering operations (`filter`), those requiring grouping operations (`group`), and those incorporating set operations such as `minus` or `union` (`minus/union`). Accuracy is computed both globally and per category.

As described in Section 4.2.1, three prompt templates (`BASIC`, `DETAILED`, and `COT`) are used to account for variability in model behavior under different instruction formulations. To avoid tying results too closely to specific prompt phrasing, a combined metric is also reported, denoted as `ENSEMBLE`, in which an answer is considered correct if at least one of the prompts produces a query equivalent to the ground truth.

A commonly used metric in related literature for evaluating query generation quality is BLEU [173, 156, 94], which measures the degree of textual overlap between a generated and a reference query. This metric was not adopted here, since the annotated queries in the *Spider4SPARQL* dataset originate from an automatic conversion of SQL queries from the *Spider* dataset [174]. While syntactically correct and functional, these queries lack formatting and employ procedural variable naming (e.g., `t1`, `t2`), whereas LLM-generated queries typically exhibit meaningful variable names and well-formatted syntax. In light of these differences, form-based comparison metrics such as BLEU are not considered suitable for the present evaluation.

Syntax correctness

Beyond verifying semantic correctness, it is useful to distinguish executable queries from those containing syntactic errors. To this end, the syntax correctness metric measures the proportion of syntactically valid, executable queries among all generated queries:

$$syntax\ correctness = \frac{number\ of\ executable\ queries}{total\ number\ of\ generated\ queries} \tag{4.2}$$

This measure reflects each model's familiarity with the formal syntax of the SPARQL language, independently of the factual correctness of its outputs.

Determinism

By analyzing multiple generations for each question, model consistency can be assessed through the determinism metric. For a given question, the generated queries are grouped according to the equivalence of their results. If all queries produce identical results, they form a single group; otherwise, each distinct result defines a separate group. The determinism score is defined as:

$$determinism = \frac{number\ of\ repetitions - number\ of\ groups}{number\ of\ repetitions - 1} \tag{4.3}$$

A determinism value of 1 indicates that all generated queries yield equivalent results, whereas a value of 0 means that each query produces a different output, with intermediate values reflecting partial stability.

Generation time

For each prompt, the elapsed time between submission and the completion of the model response is recorded. GPT models are executed remotely via OpenAI's infrastructure, while all other models run locally on an *Nvidia RTX A6000* GPU with 48 GB of VRAM.

Although these measurements are not directly comparable, as OpenAI-hosted models rely on undisclosed large-scale infrastructures, generation time remains a relevant factor for real-world question-answering applications. The reported values therefore provide an indicative measure of practical feasibility rather than absolute performance. Local generation times, in particular, should be interpreted as hardware-dependent.

4.2.8 Ontology-agnostic reference model

To provide an additional point of comparison, the evaluation includes *SGPT* [136], a neural model for SPARQL query generation from natural language and one of the most recent and widely adopted approaches to this task. Unlike the prompted LLMs considered in this study, *SGPT* does not take ontology information as input at inference time. Instead, it relies entirely on patterns learned from training data, making its performance dependent on the alignment between training and test distributions.

Originally trained on datasets such as *LC-QUAD2*, *QALD9*, and *VQUANDA*, *SGPT* was adapted to support the *Spider4SPARQL* dataset, which serves here as the primary reference for query complexity and domain diversity. This adaptation required minor modifications to the original source code to enable compatibility with the dataset and to include an additional parameter (`spider4sparql`) for data loading and training.

Although *SGPT* is not directly comparable to prompted LLMs due to its lack of ontology input, it provides a valuable baseline for illustrating the role of ontology context in generalizing across diverse knowledge graphs. As shown in Section 4.3, its limited performance in this setting underscores the challenges faced by models trained on static distributions when applied to structurally novel domains.

4.3 Results

The following section presents the experimental results according to the methodology described in Section 4.1. The evaluation involved running the LLM-based

SPARQL generation pipeline over datasets comprising 20 distinct test graphs and 1,419 natural language–SPARQL query pairs. An additional comparison with the *SGPT* model was conducted on a subset of 18 test graphs from *Spider4SPARQL*. The results are organized by evaluation measure in Sections 4.3.1 to 4.3.3, followed by an analysis of failure cases in Section 4.3.4.

4.3.1 Accuracy analysis

The accuracy analysis evaluates the ability of LLMs to generate SPARQL queries that correctly answer natural language questions. Accuracy was measured for each model across different prompts and datasets. In addition to the three prompting templates (`BASIC`, `DETAILED`, and `COT`) described in Section 4.2.1, a fourth scenario was introduced, denoted as `ENSEMBLE`. In this case, a query is considered correct if at least one of the three prompts produces a valid result and incorrect if none succeed.

Impact of query categories and model size

A graphical overview of the overall accuracy results is shown in Figure 4.2, highlighting accuracy differences among models computed using Equation (4.1) and grouped by query complexity categories as defined in Section 4.2.7.

The analysis focuses on *Spider4SPARQL*, whose broad variability in query complexity and availability of a challenging development split allow for meaningful grouping and an informative comparison with the ontology-agnostic *SGPT* model. Detailed results for the other datasets are reported in the corresponding tables.

Unlike prompted LLMs, *SGPT* cannot accept ontology input, and its performance therefore depends on the degree of overlap between its training data and the target ontology. In practice, *SGPT* generates correct queries only when class and property names in the question closely match those in its training data. These observations confirm that pretrained models of this type require retraining or fine-tuning for each new knowledge graph. Prompted LLMs, by contrast, can dynamically process ontology descriptions at inference time and achieve higher accuracy across multiple datasets without retraining.

This difference does not imply that prompting is inherently superior to task-specific training, but rather highlights the limited generalization capacity of models relying exclusively on static data. Prompting offers a complementary strategy that provides flexibility and adaptability across previously unseen graph structures.

As shown in Figure 4.2, several prompted LLMs achieve accuracy scores exceeding 0.80 for simpler queries and between 0.30 and 0.70 for more complex categories. While there remains significant room for improvement, these values demonstrate substantial zero-shot capabilities for NL-to-SPARQL translation. This performance is especially notable considering the inherent difficulty of the task and the comparatively low accuracy of *SGPT*, which remains below 0.12 in all tested configurations.

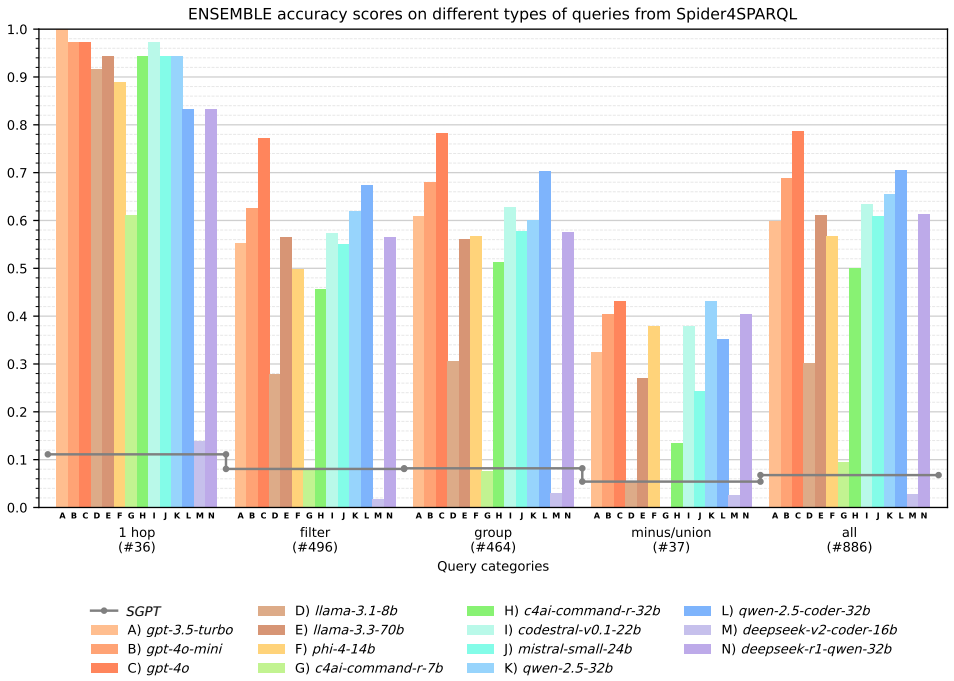

Figure 4.2: Accuracy on the *Spider4SPARQL* dataset, considering as correct those queries for which any of the `BASIC`, `DETAILED`, or `COT` prompts produced valid results (`ENSEMBLE`). The *SGPT* model is included for reference. Models from the same family are represented using the same hue, with luminance variations distinguishing models within each family. Each bar is labeled with a capital letter corresponding to its legend entry.

Tabular accuracy results for all datasets and prompt configurations are presented in Table 4.2, without separating by query category. Both Figure 4.2 and Table 4.2 reveal a consistent trend: larger and more capable models tend to achieve higher accuracy, while smaller models show reduced performance. The highest-scoring models belong to the GPT family, with *gpt-4o* reaching `ENSEMBLE` scores of 0.79 on *Spider4SPARQL*, 0.50 on *Bestiary*, and 0.78 on *LC-QuAD*. Smaller models, such as *c4ai-command-r-7b*, exhibit substantially lower accuracy (0.09, 0.01, and 0.23 respectively). Interestingly, the lowest-performing model overall was not the smallest but *deepseek-v2-coder-16b*, which produced systematic output errors analyzed in Section 4.3.4.

A notable discrepancy was also observed between these results and those reported by [83], the authors of the *Spider4SPARQL* dataset. In this study, *gpt-3.5-turbo* achieved an overall accuracy of 0.60 on the development set, compared to 0.08 reported in the original paper under a zero-shot configuration. This difference emphasizes the potential of large language models when appropriately prompted and paired with refined ontologies. It also illustrates the substantial impact of prompt

	Spider4SPARQL				Bestiary				LC-QuAD			
	BASIC	DETAILED	COT	ENSEMBLE	BASIC	DETAILED	COT	ENSEMBLE	BASIC	DETAILED	COT	ENSEMBLE
gpt-3.5-turbo	0.40	0.49	0.43	0.60	0.29	0.23	0.13	0.34	0.45	0.52	0.43	0.61
gpt-4o-mini	0.45	0.51	0.59	0.69	0.37	0.39	0.21	0.54	0.54	0.48	0.39	0.69
gpt-4o	0.57	0.67	0.71	0.79	0.33	0.33	0.29	0.50	0.65	0.67	0.71	0.78
llama-3.1-8b	0.16	0.10	0.22	0.30	0.01	0.00	0.03	0.03	0.05	0.05	0.07	0.13
llama-3.3-70b	0.20	0.46	0.49	0.61	0.01	0.20	0.15	0.27	0.07	0.12	0.17	0.22
phi-4-14b	0.38	0.25	0.42	0.57	0.10	0.10	0.10	0.20	0.32	0.23	0.23	0.44
c4ai-command-r-7b	0.08	0.02	0.00	0.09	0.01	0.00	0.00	0.01	0.15	0.06	0.07	0.23
c4ai-command-r-32b	0.20	0.37	0.37	0.50	0.13	0.14	0.14	0.30	0.29	0.26	0.30	0.55
codestral-v0.1-22b	0.31	0.48	0.47	0.64	0.17	0.23	0.22	0.41	0.43	0.37	0.54	0.67
mistral-small-24b	0.37	0.42	0.48	0.61	0.20	0.15	0.04	0.27	0.36	0.45	0.33	0.57
qwen-2.5-32b	0.34	0.45	0.53	0.65	0.02	0.01	0.08	0.09	0.45	0.43	0.51	0.71
qwen-2.5-coder-32b	0.47	0.56	0.52	0.71	0.36	0.03	0.23	0.45	0.33	0.51	0.47	0.65
deepseek-v2-coder-16b	0.02	0.00	0.01	0.03	0.02	0.00	0.00	0.02	0.15	0.02	0.03	0.18
deepseek-r1-qwen-32b	0.35	0.42	0.42	0.61	0.12	0.17	0.11	0.29	0.48	0.49	0.41	0.66

Table 4.2: Average accuracy across datasets and prompting styles.

formulation and ontology quality on performance in this task.

Impact of prompt types

Prompting style exerts a clear influence on accuracy. Across the three prompt types, larger and more capable models tend to benefit more consistently from the elaborate formulations provided by the `DETAILED` and `COT` templates. Smaller models, in contrast, often perform better with the shorter and simpler `BASIC` prompt, likely due to their limited ability to exploit additional instructions. Models such as *llama-3.1-8b*, *phi-4-14b*, and *c4ai-command-r-7b* show little or no improvement when switching to more informative prompts, and in some cases their performance even decreases. This indicates that not all architectures can process or benefit from the increased linguistic scaffolding introduced by complex prompt formulations, particularly when these require reasoning behaviors not natively supported by the model.

Overall, the `COT` and `DETAILED` prompts yield broadly comparable results, with no consistently superior option across all model–dataset combinations. In some cases, such as *gpt-4o-mini* on *Spider4SPARQL*, the reasoning-based `COT` formulation achieves higher accuracy, whereas in others, such as *mistral-small-24b* on *LC-QuAD*, the structured but static `DETAILED` prompt performs better. This variability reflects the complex interplay between the reasoning style elicited by a prompt, the internal capabilities of a model, and the linguistic and structural characteristics of the target

dataset.

The ENSEMBLE aggregation mitigates these fluctuations by selecting the best-performing result per query across prompt types, effectively estimating an upper bound for each model's potential. While this abstraction provides an optimistic perspective, the variability observed across prompts underscores the importance of carefully designing instruction formulations when adapting models to new domains or deploying them in heterogeneous settings.

Role of model specializations

An additional dimension of analysis concerns the influence of model specialization, particularly the distinction between general-purpose, coding-oriented, and reasoning-enhanced models. Excluding *deepseek-v2-coder-16b*, which exhibited systematic output errors discussed later, both *codestral-v0.1-22b* and *qwen-2.5-coder-32b* demonstrate stronger and more stable performance compared with their standard counterparts, with only minor exceptions in the *LC-QuAD* dataset. Notably, despite being smaller and older, *codestral-v0.1-22b* slightly outperformed *mistral-small-24b*, achieving ENSEMBLE scores of 0.64 on *Spider4SPARQL*, 0.41 on *Bestiary*, and 0.67 on *LC-QuAD*, compared with 0.61, 0.27, and 0.57 respectively.

Within the Qwen family, three configurations were evaluated: the base *qwen-2.5-32b*, the coding-oriented *qwen-2.5-coder-32b*, and the reasoning-focused *deepseek-r1-qwen-32b*. Although all share the same underlying architecture, they differ in fine-tuning objectives and adaptation strategies. The base model achieved ENSEMBLE scores of 0.65 on *Spider4SPARQL*, 0.09 on *Bestiary*, and 0.71 on *LC-QuAD*. The coding variant improved upon these results, scoring 0.71, 0.45, and 0.65 respectively. The reasoning-oriented version displayed less consistent behavior, with scores of 0.61, 0.29, and 0.66 across the same datasets.

These results suggest that coding specialization consistently enhances performance, particularly on structurally complex datasets such as *Spider4SPARQL* and *Bestiary*. By contrast, reasoning specializations do not yield comparable improvements despite their intended design purpose. In *Bestiary*, the coder variant remains among the top-performing models overall, substantially surpassing both its base and reasoning counterparts.

4.3.2 Determinism and syntax correctness

The results summarized in Tables 4.3 and 4.4 provide insight into model stability and syntactic compliance.

Regarding determinism, all models displayed some degree of response variance, typically within the range of 0.70 to 0.90. In instances where syntax correctness dropped substantially for certain models (such as *c4ai-command-r-7b* and *deepseek-v2-coder-16b*), the determinism score rose to nearly 1.00. This behavior occurred because those models consistently produced invalid outputs; as the metric is based

	Spider4SPARQL		Bestiary		LC-QuAD	
	BASIC	DETAILED	BASIC	DETAILED	BASIC	DETAILED
gpt-3.5-turbo	0.85	0.76	0.73	0.62	0.91	0.87
gpt-4o-mini	0.84	0.77	0.86	0.70	0.86	0.86
gpt-4o	0.87	0.86	0.83	0.73	0.88	0.90
llama-3.1-8b	0.76	0.79	0.82	0.76	0.89	0.79
llama-3.3-70b	0.81	0.78	0.92	0.80	0.91	0.88
phi-4-14b	0.73	0.80	0.71	0.73	0.76	0.74
c4ai-command-r-7b	0.81	0.94	0.80	0.99	0.77	0.77
c4ai-command-r-32b	0.70	0.73	0.77	0.70	0.73	0.79
codestral-v0.1-22b	0.77	0.75	0.68	0.59	0.85	0.82
mistral-small-24b	0.83	0.79	0.70	0.70	0.94	0.84
qwen-2.5-32b	0.76	0.76	0.83	0.74	0.79	0.83
qwen-2.5-coder-32b	0.88	0.83	0.73	0.81	0.87	0.86
deepseek-v2-coder-16b	0.94	0.99	0.82	1.00	0.73	0.91
deepseek-r1-qwen-32b	0.57	0.59	0.53	0.54	0.63	0.69

Table 4.3: Average determinism across datasets and prompt granularities. The COT prompt is excluded, as only one repetition per query was executed.

	Spider4SPARQL			Bestiary			LC-QuAD		
	BASIC	DETAILED	COT	BASIC	DETAILED	COT	BASIC	DETAILED	COT
gpt-3.5-turbo	0.92	0.92	0.90	0.87	0.80	0.74	0.99	0.99	0.98
gpt-4o-mini	0.94	0.91	0.94	0.94	0.86	0.95	0.99	0.97	0.96
gpt-4o	0.99	0.99	0.99	0.95	0.95	0.95	1.00	1.00	1.00
llama-3.1-8b	0.58	0.27	0.44	0.61	0.23	0.48	0.91	0.54	0.84
llama-3.3-70b	0.60	0.79	0.82	0.08	0.57	0.65	0.46	0.88	0.97
phi-4-14b	0.83	0.40	0.60	0.79	0.39	0.55	0.76	0.46	0.57
c4ai-command-r-7b	0.37	0.08	0.02	0.38	0.00	0.01	0.54	0.47	0.38
c4ai-command-r-32b	0.70	0.81	0.79	0.72	0.83	0.76	0.90	0.92	0.92
codestral-v0.1-22b	0.79	0.80	0.83	0.72	0.69	0.75	0.97	0.96	0.94
mistral-small-24b	0.92	0.74	0.79	0.89	0.60	0.23	1.00	0.93	0.77
qwen-2.5-32b	0.70	0.72	0.78	0.20	0.37	0.51	0.77	0.89	0.81
qwen-2.5-coder-32b	0.94	0.85	0.91	0.80	0.19	0.84	0.61	0.91	0.86
deepseek-v2-coder-16b	0.07	0.01	0.01	0.26	0.00	0.05	0.53	0.11	0.29
deepseek-r1-qwen-32b	0.78	0.75	0.69	0.66	0.65	0.48	0.93	0.92	0.81

Table 4.4: Average syntax correctness across datasets and prompt granularities.

on result comparison, consistent failure yields artificially high determinism values. The COT prompt is excluded from this evaluation, since only a single execution per query was performed, precluding an assessment of variance.

While errors can influence determinism, the relationship is not necessarily causal. Higher-performing models, such as those in the *GPT* family, *llama-3.3-70b*, and *qwen-2.5-coder-32b*, generally exhibited higher determinism compared with smaller, less accurate systems such as *phi-4-14b* and *codestral-v0.1-22b*. For instance, *codestral-v0.1-22b*, despite slightly outperforming *mistral-small-24b* overall, showed lower syntax correctness and determinism. It is worth noting that locally executed models can always be configured for complete determinism by switching to *argmax* decoding. Across all configurations, the GPT family more consistently generated SPARQL-compliant queries, outperforming other models in terms of syntax correctness.

4.3.3 Inference time

Table 4.5 reports the average execution times, aggregated across datasets, as no significant differences were observed among them. Models queried via OpenAI's API proved the most efficient, achieving average generation times below two seconds for non-reasoning prompts. This performance reflects the robustness of OpenAI's infrastructure, with only moderate increases in latency observed under reasoning-oriented prompts.

	BASIC	DETAILED	COT
gpt-3.5-turbo	0.93	1.15	2.45
gpt-4o-mini	1.46	1.75	6.39
gpt-4o	1.56	1.80	7.77
llama-3.1-8b	3.51	6.40	28.67
llama-3.3-70b	6.29	9.87	34.24
phi-4-14b	2.38	6.32	14.22
c4ai-command-r-7b	2.61	3.89	14.26
c4ai-command-r-32b	3.69	5.38	16.39
codestral-v0.1-22b	4.86	5.98	16.40
mistral-small-24b	3.59	3.81	18.06
qwen-2.5-32b	3.83	5.55	24.48
qwen-2.5-coder-32b	3.55	4.62	25.00
deepseek-v2-coder-16b	8.03	8.72	37.30
deepseek-r1-qwen-32b	19.46	19.37	25.65

Table 4.5: Average generation time (in seconds) across prompt configurations.

Local models generally exhibited longer execution times, with progressive in-

creases from the `BASIC` to the `DETAILED` and `COT` configurations. This trend stems from both the longer instruction sequences and the models' tendency to produce more verbose responses (especially under `DETAILED` prompts), which include extended schema descriptions, and under `COT`, which explicitly elicits reasoning. For the `BASIC` setting, smaller models required approximately three seconds per query, while larger ones such as *llama-3.3-70b* averaged around seven seconds. With `DETAILED` and `COT`, execution times increased to between four and approximately twenty-five seconds.

The *DeepSeek* models constitute a notable exception due to their disproportionately high response times. The reasoning-oriented *deepseek-r1-qwen-32b* averaged about twenty seconds even for `BASIC` and `DETAILED` prompts, largely due to its generation of longer outputs (on average 2,600 tokens per prompt, compared with 150-300 for other models). Similarly, *deepseek-v2-coder-16b* required roughly eight seconds for non-reasoning prompts and thirty-seven seconds under `COT`, despite generating relatively moderate-length responses. These results indicate inefficiencies relative to both model size and token output.

Overall, execution time represents a critical factor for question-answering systems. Depending on available computational resources, using LLMs as a service may offer a more balanced trade-off between latency and answer quality.

4.3.4 Failure analysis

This section examines the model outputs to identify recurring failure patterns and understand where models struggled to generate correct SPARQL queries.

Model-specific failures

Although many error types recur across models, certain systems displayed distinct and reproducible failure modes. This subsection highlights specific cases in which a model consistently failed across datasets, often due to identifiable structural or syntactic issues.

The lowest-performing model in Table 4.2 was not the smallest one but rather *deepseek-v2-coder-16b*, which achieved near-zero accuracy across all datasets and prompt types. An inspection of its outputs revealed that it systematically omitted required prefixes in class and property names, resulting in invalid SPARQL syntax and execution failures. This behavior is reflected in Table 4.4, where the same model shows widespread syntax invalidity.

Another pronounced failure mode occurred with *qwen-2.5-32b*, which achieved competitive results on *Spider4SPARQL* (e.g., `ENSEMBLE` accuracy of 0.54) but nearly collapsed on *Bestiary* (`ENSEMBLE` accuracy of 0.03). In this case, the generated queries often contained incorrect or missing prefixes and poorly adapted structures. Unlike *deepseek-v2-coder-16b*, these errors were not entirely systematic: the model

occasionally produced syntactically valid queries for simpler questions, though these frequently contained semantic or logical inconsistencies.

Smaller models, such as *llama-3.1-8b* and particularly *c4ai-command-r-7b*, also exhibited frequent syntax errors resembling the prefix-omission pattern described above, though in a less consistent manner. Often, class names were correctly prefixed while predicates were not, suggesting partial understanding of the prompt conventions. Such partially malformed queries systematically failed during execution.

These patterns indicate that failure mechanisms vary by model type: some display consistent, systematic breakdowns, others fail intermittently, and some are highly sensitive to the dataset or prompt template. In the present experiments, the most severe and consistent failures (those leading to near-zero accuracy) primarily stemmed from syntactic errors such as incomplete or malformed query structures. Nevertheless, syntax errors represent only the most visible layer of failure. Correcting them does not necessarily resolve deeper semantic issues, as alignment errors between natural-language expressions and ontology elements often persist.

Determining the precise causes of these syntactic failures would require targeted ablation studies or controlled prompting experiments. However, certain explanations can be reasonably excluded. Given the clean and consistent format of the inputs, tokenization problems are unlikely to account for the observed behaviors. Instead, these consistent failures, such as those of *deepseek-v2-coder-16b*, are more plausibly attributed to prompt incompatibilities (e.g., mismatched formatting conventions) or to limited reasoning ability in structured semantic generation tasks such as SPARQL query construction.

Taxonomy of typical failure cases

To better understand the limitations of current LLMs in SPARQL generation, a taxonomy of common failure cases identified in the evaluation is presented. The cases are grouped into eight distinct categories, each reflecting a recurring failure mode.

Category	Description
1. Prefix/omission	Missing or misused prefixes, unresolved IRIs
2. Ontology misalignment	Invented or misused classes or predicates
3. Aggregation logic errors	Incorrect use of aggregates or grouping
4. Filter misapplication	Incorrect logical conditions or set reasoning
5. Structural syntax errors	Violations of SPARQL syntax or grammar
6. Repetition patterns	Runaway generation or infinite clauses
7. Variable binding errors	Undefined or mis-scoped variables
8. Wrong query form	Using SELECT instead of ASK (or vice versa)

Table 4.6: Taxonomy of common SPARQL generation errors observed across models and datasets.

For each category, a short definition and one representative example are provided, including the dataset question and the corresponding model-generated SPARQL query. Since many errors overlap, each example is assigned to the category that best reflects its primary issue. Because queries may be incorrect for multiple reasons beyond the one being discussed, direct comparison with the ground truth is not always meaningful. To maintain conciseness, ground-truth SPARQL queries are omitted.

1. Prefix omission and IRI resolution errors These errors occur when models omit necessary prefixes or apply them incorrectly, producing unresolved IRIs and non-executable queries. They usually cause syntax errors or semantic mismatches when the model assumes an incorrect namespace resolution. This error pattern accounts for many of the systematic failures observed with *deepseek-v2-coder-16b*.

- **Failing question** (*Bestiary*)
  Which creatures that speak a language, but not DraconicL, do have Chaotic-Good alignment?

- **Generated query** (*deepseek-v2-coder-16b*)

```
SELECT ?beast
WHERE {
  ?beast hasAlignment :ChaoticGood.
  ?beast hasLanguages ?language.
  FILTER NOT EXISTS {?beast hasLanguages :DraconicL}
  FILTER EXISTS {?beast spokenBy ?language}
}
```

- **Error**
  The model omits the required prefix for `hasAlignment` and other predicates, causing parsing failure. Even if prefixes are inserted during post-processing, unresolved symbols still lead to execution errors.

2. Ontology misalignment (predicate or class misuse) These errors arise when the model hallucinates class or property names, or misuses existing ones. Generated terms may appear semantically plausible but do not exist in the ontology.

- **Failing question** (*Spider4SPARQL*)
  What are the codes of template types that are not used for any document?

- **Generated query** (*gpt-3.5-turbo*)

```
SELECT DISTINCT ?template_type_code WHERE {
```

```
  ?template_type a :ref_template_types .
  FILTER NOT EXISTS {
    ?template a :templates .
    ?template :ref-template_type_code ?template_type .
  }
}
```

- **Error**
  The model invents classes such as `:ref_template_types` and properties like `:ref-template_type_code`, which are not defined in the ontology.

3. Aggregation logic errors These occur when models misuse SPARQL aggregation constructs. Typical problems include replacing proper aggregation functions (`MIN`, `MAX`) with `ORDER BY` and `LIMIT`, or violating grouping rules.

- **Failing question** (*Spider4SPARQL*)
  What is the maker of the car produced in the earliest year and what year was it?

- **Generated query** (*gpt-4o*)

```
SELECT ?makerName ?earliestYear WHERE {
  ?carData :year ?earliestYear ;
           :id ?carId .
  ?carName :makeid ?carId ;
           :make ?makerName .
}
ORDER BY ASC(?earliestYear)
LIMIT 1
```

- **Error**
  The query uses `ORDER BY` and `LIMIT` to approximate a `MIN` operation, which is incorrect and unreliable when multiple matches exist.

4. Filter misapplication and set-reasoning mistakes These involve incorrect use of `FILTER`, `MINUS`, or `EXISTS`, producing faulty logical interpretations. The model often reasons at the triple level instead of the set level.

- **Failing question** (*Spider4SPARQL*)
  Return the country codes for countries that do not speak English.

- **Generated query** (*gpt-4o*)

```
SELECT DISTINCT ?countryCode
WHERE {
  ?countrylanguage :countrycode ?countryCode .
  ?countrylanguage :language ?language .
  FILTER(?language != "English")
}
```

- **Error**
  The query filters individual language triples, incorrectly retaining countries that speak both English and other languages. The intended meaning requires excluding all countries that speak English, using a `MINUS` or `FILTER NOT EXISTS` construct.

5. Structural syntax errors This category includes violations of SPARQL grammar, such as misuse of `a` (`rdf:type`), invalid nesting, or undefined constructs.

- **Failing question** (*LC-QuAD*)
  How many different people own the Timeform award winners?
- **Generated query** (*c4ai-command-r-7b*)

```
SELECT COUNT(DISTINCT ?person) WHERE {
  ?film a :property/awards :resource/Timeform .
  ?film :ontology/owner ?person .
}
```

- **Error**
  The triple `?film a :property/awards :resource/Timeform` misuses `a` with a compound path, which is syntactically invalid. The `a` keyword must be followed by a single class IRI.

6. Run-on repetition and non-terminating patterns Some models generate excessively long or repeating queries due to token-level loops or confusion during decoding. This produces unparseable outputs that fail to capture any coherent logic.

- **Failing question** (*Bestiary*)
  Are angels characterized by the greater average value of experience points than demons?
- **Generated query** (*llama-3.1-8b*)
  [Omitted for brevity; consisted of over 600 lines with repeating `SELECT`, `BIND`, and structural patterns.]

- **Error**
  The model enters a runaway generation loop, producing redundant clauses that do not terminate meaningfully. Such outputs typically fail to execute due to timeouts or size constraints.

7. Variable scope and binding errors These occur when variables are used without being properly introduced or when their scope is misinterpreted. Such mistakes generally render the query invalid.

- **Failing question** (*Spider4SPARQL*)
  What are the names of the dogs for which the owner has not spent more than 1000 for treatment?

- **Generated query** (*qwen-2.5-coder-32b*)

```
SELECT ?dog_name WHERE {
  ?dog :name ?dog_name .
  ?dog :ref-owner ?owner .
  ?treatment :ref-dog ?dog .
  ?charge :ref-treatment ?treatment .
  FILTER(?charge_amount <= 1000)
}
GROUP BY ?dog_name
HAVING(SUM(?charge_amount) <= 1000)
```

- **Error**
  The variable `?charge_amount` is never bound before use in the `FILTER` and `HAVING` clauses, making the query invalid.

8. Wrong query-form selection These errors involve choosing the wrong SPARQL form (`SELECT`, `ASK`, or `CONSTRUCT`) relative to the user intent. This produces verbose or semantically mismatched outputs.

- **Failing question** (*Bestiary*)
  Do Lorelei and sirens speak the same languages?

- **Generated query** (*mistral-small-24b*)

```
SELECT ?lang1 ?lang2
WHERE {
  ?lorelei :hasName "Lorelei" .
  ?siren :hasName "Siren" .
  ?lorelei :hasLanguages ?lang1 .
  ?siren :hasLanguages ?lang2 .
  FILTER (?lang1 = ?lang2)
```

```
}
```

- **Error**
  The query lists all matching language pairs, while the user's intent is to check whether any overlap exists. The correct form should be `ASK`, as the expected output is a boolean value.

4.4 Summary

This chapter investigated the ability of large language models (LLMs) to translate natural language questions into SPARQL queries in a zero-shot setting over domain-specific knowledge graphs and ontologies. The study was conceived in continuity with the previous chapter, where the automatic construction of such knowledge graphs was explored. The evaluation therefore focused on scenarios in which the graphs or ontologies are newly created and lack training data, reflecting the practical conditions of automatically structured knowledge resources.

The results demonstrate that larger and more capable models consistently achieve higher accuracy, particularly when handling structurally complex queries. Nevertheless, recent open-weight models, including *qwen-2.5-coder-32b*, *llama-3.3-70b*, and *codestral-v0.1-22b*, exhibit competitive performance that approaches that of proprietary models, indicating a narrowing performance gap between open and closed systems.

Prompt formulation emerged as a central factor influencing model behavior. Structured prompting strategies, such as `DETAILED` and `COT`, tend to benefit larger models by encouraging more systematic reasoning, whereas smaller models perform more reliably with the simpler `BASIC` prompt. This pattern reflects differences in the models' ability to process and exploit complex instructions.

Across datasets, results confirm that LLMs can generate syntactically valid and semantically meaningful SPARQL queries even without domain-specific training. The comparison with the *SGPT* reference model emphasizes the advantage of models capable of incorporating ontology information at inference time, which supports broader generalization across distinct knowledge graphs.

The failure analysis identified several recurring error types, including prefix omissions, ontology misalignment, and aggregation or filtering mistakes. These findings highlight the dependence of LLMs on clear and consistent ontology design. Explicit and well-structured ontologies, providing intuitive naming, coherent relations, and standardized modeling, substantially improve LLM interpretability and query accuracy.

Overall, this chapter provided an integrated evaluation of LLM prompting strategies for knowledge graph querying. It established that prompt design, model capacity, and ontology quality jointly determine performance in NL-to-SPARQL transla-

tion, outlining the conditions under which language models can function as effective intermediaries between users and structured knowledge.

Chapter 5

Application

Once textual knowledge has been structured into coherent representations and made accessible through natural-language interfaces, the next step is to apply these principles in a real-world domain. Among the many contexts where large volumes of heterogeneous information are routinely generated, healthcare is particularly emblematic. Electronic Health Records (EHRs) constitute one of the richest and most complex data ecosystems, documenting the medical history of patients across multiple encounters, institutions, and care processes.

EHR systems combine diverse information types, ranging from highly structured administrative and laboratory data to semi-structured forms and fully unstructured clinical narratives. While the structured components enable standardized storage and retrieval, the unstructured notes written by clinicians often contain essential details that are not captured elsewhere, such as observations, justifications, and contextual nuances that are indispensable for understanding patient trajectories. This coexistence of structured and unstructured information mirrors the broader challenges discussed throughout this thesis: how to represent, integrate, and exploit textual knowledge in a machine-interpretable form.

At the same time, the healthcare domain illustrates the critical importance of interoperability. Differences in data models, formats, and terminologies across hospitals and information systems limit the reusability of clinical data and hinder its secondary use for research, monitoring, and decision support. Knowledge graph technologies provide a promising avenue to address these issues, offering a semantic layer capable of integrating heterogeneous sources within a unified representational framework. However, the integration of free-text notes within such structures remains particularly challenging, given their volume, variability, and sensitivity.

The work presented in this chapter explores how Large Language Models (LLMs) and ontology-driven knowledge graph modeling can be combined to address these challenges. Focusing on real-world EHR data represented in the FNUX format, it examines how LLMs can structure unstructured clinical notes into event-based representations that complement and extend the native structured data. By doing so, it demonstrates how the techniques developed in previous chapters can be applied to

a concrete, domain-specific scenario where interoperability, semantics, and temporal reasoning are essential.

5.1 Problem analysis and system architecture

Building on the challenges outlined in the introduction, this section examines the two main issues that shape the proposed architecture. First, a substantial portion of clinically relevant information resides in free-text notes, which limits interoperability and automated use. Second, existing EHR data models are typically based on rigid schemas and lack the flexibility to represent semantically rich aspects of patient data, particularly in capturing temporality and linking to broader biomedical knowledge.

To address the first challenge, recent advances in large language models (LLMs) are leveraged for the identification and structuring of clinically relevant information from unstructured notes, even under zero-shot conditions [135, 67, 161, 123, 52]. These capabilities make LLMs a suitable component in pipelines designed to transform clinical narratives into structured event representations. To address the second challenge, a semantic modeling approach based on knowledge graphs is adopted. An event-based ontology incorporating temporal attributes enables the knowledge graph to capture the evolution of patient health status and to provide a more expressive representation than conventional EHR data models. This approach relies on an ontology-driven mapping process, transforming both natively structured data and LLM-extracted events into RDF triples.

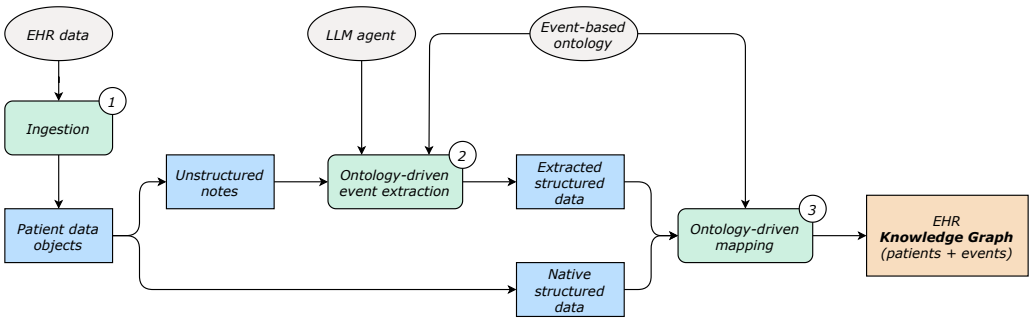

Figure 5.1: Conceptual pipeline for transforming heterogeneous EHR data into a knowledge graph representation. Gray ovals represent external sources (EHR data, LLM, ontology), green rounded boxes represent processing modules, blue rectangles represent intermediate data states, and the orange rectangle indicates the final output.

The resulting pipeline architecture combines these two components. LLMs process clinical notes locally, within a secure and restricted environment, to extract diagnoses and other clinically relevant events, while an ontology-based representation layer integrates these extracted events with the structured portion of the EHR.

The outcome is a knowledge graph that captures both structured and unstructured information in a coherent, temporally aware model. Figure 5.1 illustrates the process at a high level. In module (1), EHR data (whether stored in local databases or accessed through APIs) are ingested and parsed into an internal patient-level representation that facilitates downstream processing. Module (2) constitutes the core of the system: an LLM processes unstructured clinical notes in an ontology-driven event extraction step, while native structured fields are preserved. Finally, module (3) integrates both native and extracted structured data through ontology-driven mapping, producing an event-based knowledge graph representation of the patient records.

In summary, the architecture integrates LLM-based event extraction with ontology-driven semantic modeling, addressing the challenges of unstructured information and temporal representation in EHR data.

5.1.1 Ontology-driven event extraction

As illustrated in Figure 5.1, the pipeline consists of three modules. Modules (1) and (3) perform format-dependent mappings: from the source EHR schema into an internal representation, and subsequently into RDF/OWL. These steps, while essential, are primarily engineering tasks that adapt to the specific data model and ontology. By contrast, module (2) constitutes the core research component, focusing on the extraction of clinically meaningful events from unstructured notes and their alignment with an event-based ontology. This module combines local large language models with an ontology-driven schema to recognize, classify, and structure events such as diagnoses from free-text clinical notes. The following subsections introduce the event-based ontology design principles and describe the extraction component that operationalizes them on clinical notes for integration into the semantic representation layer.

Event-based ontology

To represent clinical information in a temporally coherent way, a general modeling framework is introduced to organize patient data around events. The ontology framework is designed to accommodate heterogeneous EHR information while placing particular emphasis on events and their temporal dimension. Its high-level structure is shown in Figure 5.2, comprising five main categories: *Patient*, *PatientHealthOccurrence*, *ClinicalConcept*, *ClinicalEntity*, and *HealthcareSystemEntity*.

Patient and *PatientHealthOccurrence* capture the temporal aspects of a patient's clinical history. A distinction is made between *ClinicalCourse*, representing aggregated events of care, and *ClinicalEvent*, representing atomic occurrences in time. Events provide a natural unit of representation, as they can be temporally situated and linked to other concepts. For example, a *ProcedureEvent* may register an in-

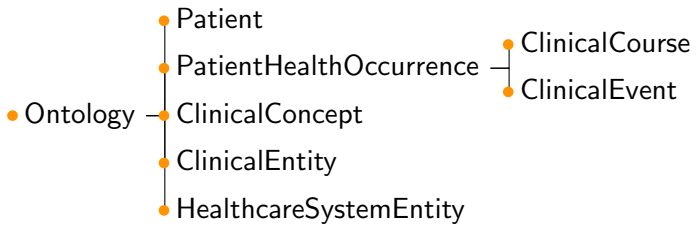

Figure 5.2: High-level organization of the ontology classes.

tervention on a patient and be linked to related concepts such as the affected body part, a medical device used during the procedure, and the healthcare personnel who carried it out. Since these events constitute the main targets of the extraction module, each subclass of *ClinicalEvent* must be supported by appropriate strategies for recognition and structuring from clinical text.

ClinicalConcept represents static or terminological knowledge, such as diagnoses, drugs, medical devices, or prescription reasons. These concepts are not tied to specific patients but provide the controlled vocabularies referenced by events and entities. This is also the level where the local concepts can be linked to widely used biomedical ontologies, such as *SNOMED CT*, *LOINC*, *DRON*, *OGMS*, or *OBI*. Such mappings are key to improving semantic interoperability and facilitating integration with external biomedical knowledge bases.

ClinicalEntity covers concrete objects in the EHR that are tracked in the hospital system, such as a specific drug dispensed, a laboratory result, or a clinical episode. Unlike concepts, which capture definitional knowledge (e.g., the drug "Ibuprofen"), entities represent patient-specific instances (e.g., "Ibuprofen 400 mg with a specific identifier and batch number"). Moreover, they differ from events in that an event may aggregate multiple entities and link them to the patient and additional concepts.

Finally, *HealthcareSystemEntity* includes facilities, healthcare workers, and roles. These classes provide contextual information about the actors and settings in which clinical events occur.

This ontology framework emphasizes the centrality of events while providing a principled way to connect them with static concepts, concrete entities, and healthcare context. It also aligns directly with the extraction tasks in the pipeline, which aim to detect, classify, and structure such events from free-text notes. It is presented as a generic framework: in practice, additional classes must be introduced to capture the specific data structures and coding systems of a given EHR.

LLM-based event recognition and structuring

Module (2) of the pipeline implements the ontology-driven extraction of clinically meaningful events from unstructured notes. The process operates iteratively on one

clinical note[1] and one event type at a time, with each step mediated by a local large language model. The LLM receives the clinical note as input context, and its behavior is guided by dedicated prompts. Prompts for the same step can differ slightly across event types (such as for diagnoses, procedures, or medications) to match the semantics of each type as defined in the ontology.

The extraction process is divided into three successive steps. This modular design improves accuracy while reducing execution time. The first step (classification) is formulated as a binary classification, requiring the model to generate only a single token. The following steps (listing and structuring) progressively increase in complexity but are executed only if the classifier predicts that the note contains at least one mention of the candidate event type. Since LLM latency is mainly proportional to the number of generated tokens, this organization helps to contain inference time. The process is iterated for each note and event type, as illustrated in Figure 5.3, and detailed below.

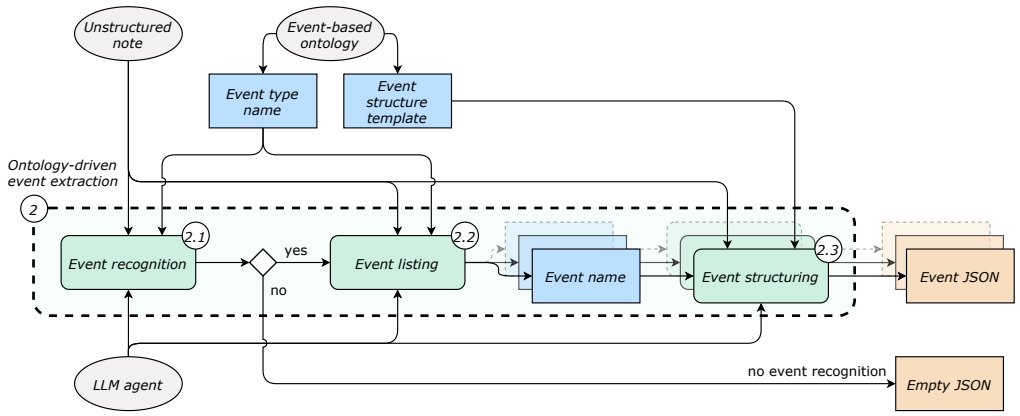

Figure 5.3: Detailed internal processing of module "(2) Ontology-driven event extraction". Gray ovals represent external sources (note, LLM, ontology), green rounded boxes represent processing modules, blue rectangles represent intermediate data states, and the orange rectangle indicates the final output.

Event recognition (2.1). Event recognition classifies the note as mentioning or not a candidate event type. The LLM is prompted with the clinical note and the event type definition from the ontology to decide if any instance is present. The output is a binary decision ("yes"/"no"), which determines whether the extraction process continues for that event type. A bypass option allows subsequent steps to be skipped in the negative case, thereby avoiding token-expensive processing that would otherwise yield only empty results. An instance of the prompt used in this step is reported in Section A.4.1.

[1]Since notes are extracted from an EHR system, the patient to whom each note belongs is already known. This information is not expected to be re-extracted from the note.

Event listing (2.2). Event listing enumerates the explicit mentions of events. If the candidate event type is recognized, the system proceeds to list the explicit mentions of that event in the note. Each mention is assigned a representative name (e.g., for a diagnosis, the name of the condition; for a prescription, the name of the drug). The LLM is prompted with the note and the event type and asked to return an array of strings in JSON format, each corresponding to one distinct event mention. This step deliberately restricts the task to listing surface forms, deferring more complex enrichment to the structuring step. An instance of the prompt used in this step is reported in Section A.4.2.

Event structuring (2.3). Event structuring enriches each event with detailed attributes. Each extracted event name is then independently processed by the structuring module, which enriches and normalizes the event according to the ontology schema. Every event type is associated with a *structure template*, i.e., a JSON object specifying the expected attributes and their format. For example, a diagnosis template may include keys such as *date*, *certainty* (suspected or certain), and *status* (active or cured), with optional extensions for provider information, standardized codes (ICD-10, SNOMED), or other metadata. The LLM receives the note as context, the event name to be structured, and the corresponding template. Its task is to replicate the template, filling values with information extracted from the note, or omitting keys if the information is absent. The output is a structured JSON object for each event mention. An instance of the prompt used in this step is reported in Section A.4.3.

Temporal information. Temporal attributes can be explicitly requested through the structure template. When the note specifies a date (e.g., "diagnosed in 2020"), the LLM is expected to return it in a standardized format (e.g., ISO 8601). When no explicit date is present, the timestamp of the note itself is used. This design choice is generally practical for acute clinical events such as prescriptions or diagnoses but less reliable for chronic conditions or past medical history. Nevertheless, it provides a workable strategy to assign temporal anchors to extracted events when direct evidence is missing.

Output. The outputs of the three steps are aggregated by event type into a list of structured events. Each event is later converted into ontological triples and inserted into the knowledge graph, linked both to the originating patient and grounded with its temporal information. This complements the events reconstructed from the natively structured data available in the EHR.

Discussion. This design emphasizes modularity: recognition, listing, and structuring are decoupled components, each guided by ontology-derived prompts and templates. The iterative processing of individual event mentions ensures focus and

interpretability, while the manually curated schema templates enforce semantic consistency across event types. These design choices allow the extraction process to scale across clinical ontologies and to accommodate evolving definitions of clinically relevant events, while ensuring that the extracted events are readily convertible into ontological triples for integration into the knowledge graph.

5.2 Experiments

The experiments assess the effectiveness of the LLM-based extraction module (2), focusing on two key aspects: the identification of clinically relevant events from free-text notes and the assignment of temporal attributes. These aspects constitute the main methodological contributions of the proposed approach and are therefore the focus of the evaluation.

As a case study, the analysis concentrates on *diagnoses*, a type of clinical event of particular importance in medicine and consistently encoded in EHR systems. Formally, a diagnosis is the process of identifying and characterizing a patient's health condition by recognizing diseases, disorders, or injuries from their signs, symptoms, and test results, and distinguishing them from other possible conditions.[2,3] While historically diagnoses were categorized directly by disease, in contemporary practice and EHR systems they are operationalized more broadly and encoded using classification systems [101], such as ICD for disease-specific coding or ICPC for broader primary care contexts. Defining and structuring diagnoses is therefore both clinically central and methodologically challenging, making them a suitable test case for the pipeline.

The datasets are analyzed in Section 5.2.1, the note processing components in Section 5.2.2, the evaluation methodology in Section 5.2.3, and the results in Section 5.2.4.

5.2.1 Datasets

This subsection discusses the challenges in selecting datasets for event extraction from clinical notes, describes the resources ultimately used in the evaluation, and outlines the preprocessing and descriptive analyses applied to them.

Challenges in dataset selection

Evaluating event extraction from free-text clinical notes is challenging, largely due to the scarcity of viable datasets. Although the pipeline is event-agnostic and can be adapted to recognize different types of events enriched with attributes, two main challenges hamper systematic evaluation.

[2] `https://www.britannica.com/science/diagnosis`

[3] `https://www.cancer.gov/publications/dictionaries/cancer-terms/def/diagnosis`

The first challenge concerns data availability. Many corpora used in prior clinical NLP research remain private and institution-specific. Examples include the *Pediatric Osteosarcoma Pathology Reports* with annotated tumor grades [67], the *RACF Nursing Notes* annotated for agitation [161] and malnutrition risk factors [5], and the *Stanford MOUD Dataset* of opioid use disorder notes [97]. Numerous other studies rely on similarly restricted resources, limiting reproducibility [135, 123, 150].

The second challenge relates to annotation richness. Publicly available corpora such as *MIMIC* [75, 74] are widely used [49, 105] and provide real-world de-identified collections of clinical data from intensive care units, including free-text discharge notes linked to admission diagnoses. Several derived datasets add targeted annotations, for example the *Processed Benchmark Dataset* from BIMSB[4], which assigns specialties to diagnoses (e.g., cardiology, neurology) [49], or *MDACE* [26], which links admission diagnoses to explicit textual evidence under manual supervision [72]. Despite their value, these resources primarily provide flat entity annotations (e.g., diagnosis names), without systematically capturing attributes of interest such as status, temporality, certainty, or severity in the case of diagnoses. In this setting, an event is essentially an entity enriched with temporal and other contextual attributes, which makes the evaluation of event structuring difficult in the absence of such metadata.

A notable exception is the 2012 *i2b2 Temporal Relations Challenge* corpus [153], which annotated discharge summaries with a wide range of clinically relevant events organized into four categories: clinical concepts, clinical departments, evidentials, and occurrences. Each event is labeled with attributes for type, polarity, and modality, and temporal expressions (TIMEX3) are linked to events through temporal relations (TLINKs). Although the schema remains relatively shallow (each event type is enriched only with polarity and modality, and the categorization is not grounded in a formal ontology) this corpus was one of the earliest large-scale resources to support temporal event extraction from clinical notes. However, it is currently not openly hosted and is listed as temporarily unavailable.[5]

Selected datasets and filtering

Since no single dataset fully satisfies the requirements for event structuring and open availability, two sources were selected that complement each other in terms of accessibility and annotation detail:

- the *MDACE* dataset [26], openly available, from which 560 MIMIC-III notes annotated with diagnoses coded in ICD-10 were obtained. While in principle plain MIMIC discharge diagnoses could have been used, the manual evidence annotations in MDACE provide additional assurance that each code corresponds to an explicit mention in the note;

[4] `https://github.com/BIMSBbioinfo/medLLMbenchmark`

[5] `https://portal.dbmi.hms.harvard.edu/projects/n2c2-nlp/`

- the *FNUX* data, private and non-disclosable, obtained through collaboration with Aalborg University Hospital. This dataset comprises the clinical histories of several patients, yielding 1,290 Danish clinical notes with linked diagnoses coded in ICPC-2 and SKS (Sygehusvæsnets Klassifikationssystem). Both notes and diagnoses are associated with dates, enabling evaluation of temporal anchoring within a realistic and heterogeneous EHR setting. Notes without associated diagnoses were excluded.

For both datasets, only the diagnosis names were retained, disregarding the associated ICD-10, ICPC-2, or SKS codes since they are not explicitly mentioned in the text. While ICD and ICPC operationalize diagnoses through their entire code lists, including concepts outside a strict medical definition, the definition of diagnosis events adopted here is based on a concise medical description applicable across coding systems. To ensure consistency with this ontological definition, a filtering procedure was applied to exclude codes that do not correspond to diagnoses in this stricter sense. Both the full annotated set and the filtered subset are retained, providing two complementary views of the data for later evaluation. The filtering rules are as follows:

- **Natural language form**: exclude codes whose description contains terms such as "concern," "fear," or "question," which signal non-diagnostic concepts.

- **ICPC-2 codes**: consist of a one-letter category followed by a two-digit number. Exclude process codes (prefix "–"), social problems (category "Z"), and general codes ("A") below 91. All other categories are retained, but only codes numbered 70 or above.

- **ICD-10 codes**: consist of a one-letter category prefix. Exclude symptom codes ("R"), encounter codes ("Z"), and external causes ("V/W/X/Y"). Retain disease codes ("A–Q"), injury codes ("S/T"), and special-purpose categories ("U").

Together, these datasets allow the examination of complementary aspects of the pipeline: *MDACE*, as an openly available resource that enables reproducible diagnosis extraction experiments, and *FNUX*, as a private but temporally rich EHR collection that allows evaluation in a realistic clinical setting. During preliminary pipeline design and prompt engineering, the *PMC-Patients* dataset [180] was also used, providing a convenient publicly available set of clinical narratives for prototyping.

Descriptive statistics

To provide an overview of the selected datasets, simple statistics are reported together with descriptive observations.

MDACE notes, derived from MIMIC-III, consist exclusively of discharge summaries. These documents summarize an entire patient stay and are therefore long and information-dense, with diagnostic mentions interleaved with a substantial amount of additional medical detail. By contrast, *FNUX* includes a broader range of clinical notes, where discharge summaries are relatively uncommon. Most *FNUX* notes are shorter and more focused, often containing only one or a few diagnoses. This contrast is reflected in the note length distribution shown in Figure 5.4.

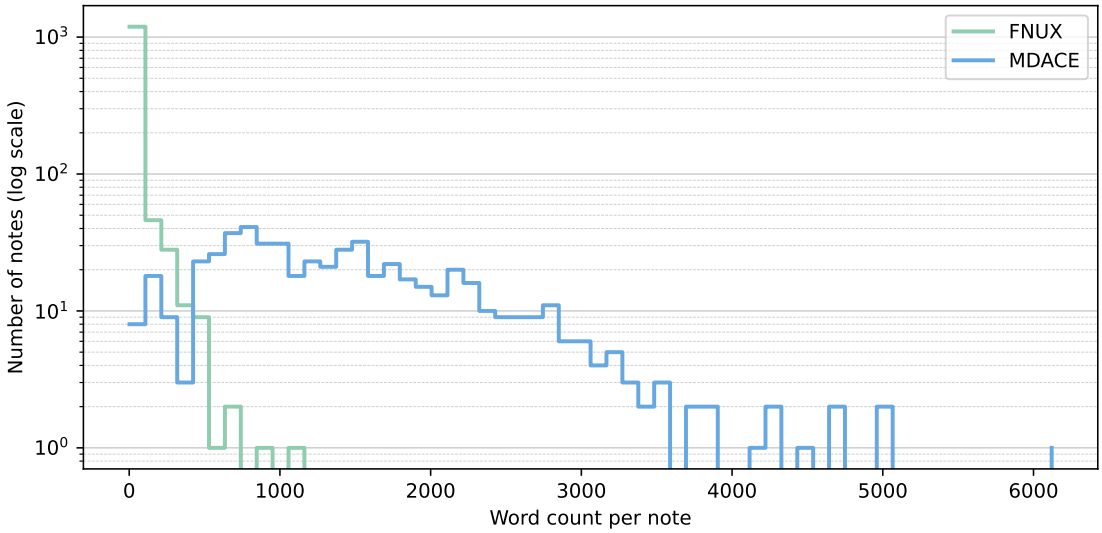

Figure 5.4: Distribution of note lengths in FNUX and MDACE, measured in word counts per note (logarithmic y-scale). MDACE notes are generally longer and exhibit a broader distribution, while FNUX notes are shorter and more concentrated at lower word counts.

Note length also correlates with the number of diagnoses linked to each note. In *FNUX*, the average is close to one diagnosis per note, whereas *MDACE* averages around six. Applying the filtering rules reduces the number of retained diagnoses in both datasets, and in *FNUX* it occasionally leaves notes with no remaining diagnoses. The counts of notes and diagnoses before and after filtering are summarized in Table 5.1.

Dataset	Notes	Diagnoses (all)	Diagnoses (filtered)
MDACE	560	3,319	2,565 (−22.7%)
FNUX	1,290	1,401	649 (−53.7%)

Table 5.1: Counts of notes and diagnoses before and after filtering.

5.2.2 LLM-based note processing

The language model encodes the natural language and medical knowledge required to interpret clinical notes and extract structured medical events. The following describes how the model was chosen and how it was instructed through prompting for the different pipeline steps, focusing on the case of diagnosis events.

Language model choice

Prior literature shows that many studies commonly rely on general-purpose LLM families such as OpenAI's GPT [135, 67, 150, 120, 57, 182, 102, 144, 72, 97], Google DeepMind's Gemini [120, 57, 182, 144], Anthropic's Claude [120, 49, 144], and Meta's Llama [2, 161, 123, 150, 52, 120, 144, 5, 97]. Other families, including Mistral, Cohere, Bard, PaLM, Flan, and BERT, have also been evaluated across a range of clinical NLP tasks. Other families, including Mistral, Cohere, Bard, PaLM, Flan, and BERT, have also been evaluated across a range of clinical NLP tasks.

In this case study, remote-only LLMs such as GPT, Gemini, or Claude, while demonstrating strong performance in many prior evaluations, were not considered viable due to the sensitivity of clinical notes and the requirement for a pipeline deployable entirely on secure local infrastructure. To this end, preliminary experiments were conducted with a set of open-weight models of moderate size (tens of billions of parameters), chosen to balance performance with the feasibility of local deployment.[6] The models explored included Llama 3 (8B, 70B), Mistral 24B, Qwen 2.5 (14B, 32B), and Gemma 3 27B, as well as its medical variant MedGemma [139]. These preliminary tests focused on event recognition and listing (diagnoses, prescriptions, procedures, device exposures, admissions to care), followed by event structuring (e.g., linking prescriptions with drug names, dosages and frequencies, or procedures with the targeted body part and associated devices). Experiments were conducted on notes from the *PMC-Patients* dataset [180], with manual supervision of the outputs. Because this process was labor intensive, we did not run a systematic or extensive comparison across models and we relied mainly on qualitative observations.

Smaller models such as Llama 3 8B struggled to consistently handle event listing and structuring, whereas larger models generally produced more reliable results. No single model clearly outperformed the others in this limited exploratory setting. However, given evidence that domain-specialized LLMs often achieve superior performance in clinical tasks [86], and since *MedGemma* [139] performed strongly in both event extraction and structuring during preliminary evaluations, it was adopted as the core model for the pipeline. Prompts and extraction logic were tuned accordingly, and *MedGemma* is the model formally evaluated in this work.

[6] With 4-bit quantization, the largest model tested required less than 48 GB of memory.

Prompt formulation

The performance of the extraction pipeline depends not only on the underlying LLM but also on how it is instructed. Based on preliminary investigations, a generic prompt structure was defined that can be adapted to each pipeline step and event type. Every prompt includes a {`note`} placeholder, substituted at runtime with the clinical note, and generally follows a common pattern: (i) provide a short introduction describing the note, (ii) formulate the task at a high level (classification, extraction, or structuring), (iii) insert the note text, and (iv) restate the task with more detail, optionally including event-specific guidance.

Initially, fully generic event placeholders for event type and semantic description were tested, retrieved automatically from the ontology. However, these formulations proved too rigid, as effective prompts often benefit from event-specific details or dataset-specific instructions. For example, since *FNUX* notes are predominantly written in Danish but occasionally in English, prompts explicitly warn the model about this variability. Similarly, when a dataset encodes diagnoses in a specific system (e.g., ICD-10), allowing for more tailored instructions may improve performance. A general prompting flow was therefore adopted, complemented by manually written, event-specific instructions to ensure flexibility across different event types.

It was also found advantageous to vary the strictness of prompts across steps. In the recognition stage, the task is deliberately more permissive than in the listing stage: the model is instructed to return "yes" whenever a diagnosis might plausibly be present, thereby reducing false negatives that would otherwise halt the pipeline. This comes at the cost of some additional false positives, but these are subsequently checked in the listing step, which applies stricter criteria to enumerate only explicit mentions and may ultimately return none.

Finally, the structuring step enriches each mention according to an ontology-derived schema. Prompts in this step include a {`json_template`} placeholder, replaced at runtime with a JSON object describing the expected fields (e.g., date, status, certainty). The model is instructed to replicate the template and fill in only attributes explicitly mentioned or clearly implied in the note, omitting absent fields. These schemas must be defined manually for each event type.

5.2.3 Evaluation criteria

The evaluation assesses the effectiveness of the pipeline along two complementary dimensions: (i) the ability to recognize and extract diagnosis mentions from clinical notes, and (ii) the ability to correctly assign temporal anchors to these diagnoses. Both evaluations are performed on the annotated datasets introduced in Section 5.2.1, considering results on both the full set of annotated diagnoses and the filtered subset that retains only strict diagnoses. Extraction of diagnosis names can be assessed on both *MDACE* and *FNUX*, which provide notes annotated with diagnoses. Temporal anchoring, in contrast, can only be assessed on *FNUX*, as it

is the only dataset that includes EHR-based temporal annotations. The following subsections describe the evaluation methodology for each of these tasks.

Diagnosis extraction

The recognition and listing modules were evaluated on the *MDACE* and *FNUX* datasets, each providing clinical notes annotated with diagnoses. Extraction is performed independently on each note, as illustrated in Figure 5.5.

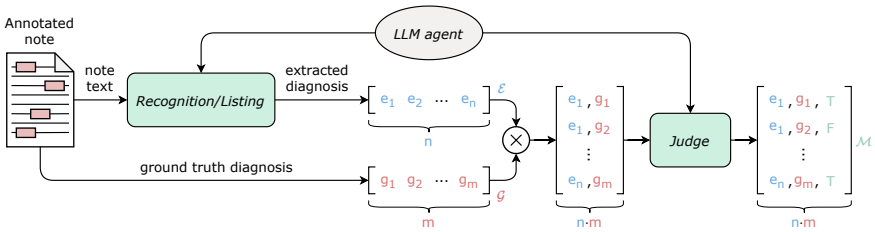

Figure 5.5: Evaluation pipeline for diagnosis extraction. Ground-truth diagnoses are paired with the extracted diagnoses, and their equivalence is assessed by an LLM-based judge.

Given a note, the set of annotated (ground-truth) diagnoses is denoted as $\mathcal{G} = \{g_1, \ldots, g_m\}$ and the set of diagnoses extracted by the system as $\mathcal{E} = \{e_1, \ldots, e_n\}$. To assess correctness, all possible pairs (e_i, g_j) are formed and evaluated to determine whether the two diagnoses are medically equivalent.

A key challenge in this evaluation is that equivalent diagnoses often appear under different surface forms (e.g., "renal failure" vs. "kidney failure"). Exact string matching is too rigid and results in excessive false negatives, whereas embedding-based similarity requires threshold calibration and might inflate false positives. To overcome these limitations, the medical LLM *MedGemma* is adopted as a *judge* to assess whether each (e_i, g_j) pair is medically equivalent. This approach aligns with the emerging "LLM-as-a-Judge" paradigm [53], which has shown that LLMs can provide reliable semantic evaluation at scale. While human annotation remains the gold standard, it is prohibitively costly at the scale of these experiments. In contrast, LLM-based judging offers a practical compromise: more robust than purely lexical or embedding-based methods, and far more scalable than manual curation. To empirically assess its reliability in this specific setting, the judge itself was additionally evaluated against human annotators (Section 5.2.3).

The judge returns a label for each pair, resulting in triples of the form (e_i, g_j, T) or (e_i, g_j, F), where T indicates semantic equivalence and F indicates none. Collectively, these form the set $\mathcal{M}$. From this, precision and recall are defined as:

$$P = \frac{|\{e \in \mathcal{E} : (e, g, T) \in \mathcal{M}\}|}{|\mathcal{E}|} \tag{5.1}$$

$$R = \frac{|\{g \in \mathcal{G} : (e, g, T) \in \mathcal{M}\}|}{|\mathcal{G}|} \tag{5.2}$$

Here, P measures the fraction of extracted diagnoses that are correct (precision), while R measures the fraction of annotated diagnoses successfully recovered (recall). Both metrics are averaged over all notes to yield dataset-level scores. Results are reported on both the full annotated set $\mathcal{G}$ and the filtered subset $\mathcal{G}_f \subset \mathcal{G}$, which retains only strict diagnoses according to the filtering rules described in Section 5.2.1.

Structuring and temporal anchoring

Beyond diagnosis recognition, the evaluation also assesses the temporal anchoring of extracted diagnoses using the *FNUX* dataset, which includes temporal annotations in the source EHR. Temporal assignment follows the procedure described in Section 5.1.1, where dates are derived from the structuring module; if no explicit temporal expression is identified in the note, the note creation date is used as a proxy.

Formally, let $\mathcal{T}(x)$ denote the date assigned to diagnosis x. Temporal accuracy can be assessed only for extracted diagnoses that were judged correct, i.e. equivalence triples $(e, g, T) \in \mathcal{M}$, since only these have a valid ground-truth reference. For each such pair, the predicted date $\mathcal{T}(e)$ is compared with the annotated date $\mathcal{T}(g)$, as illustrated in Figure 5.6.

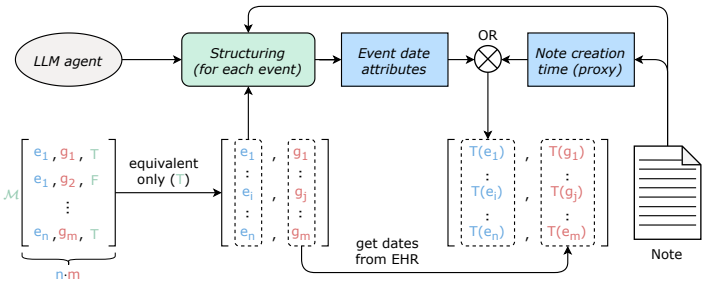

Figure 5.6: Evaluation pipeline for temporal anchoring. For each correctly matched diagnosis pair (e, g, T), the date assigned by the structuring module (or the note date as proxy) is compared with the ground-truth date annotated in the EHR system.

Accuracy is reported at three temporal granularities: exact day (D), within one week (W), and within one month (M). For a tolerance window $h \in \{D, W, M\}$, temporal accuracy is defined as:

$$A_h = \frac{|\{(e, g, T) \in \mathcal{M} : |\mathcal{T}(e) - \mathcal{T}(g)| \leq h\}|}{|\{(e, g, T) \in \mathcal{M}\}|} \tag{5.3}$$

This metric quantifies the proportion of correctly extracted diagnoses whose predicted temporal anchor falls within h days of the annotated ground-truth date.

Reliability of the judge

To assess whether *MedGemma* produces judgments consistent with human reasoning, a focused inter-annotator agreement study was conducted.

From all pairs (e_i, g_j) created for evaluating diagnosis equivalence, 200 were randomly selected for human review. Because non-equivalent pairs were expected to dominate, a stratified sampling strategy was applied based on the LLM judge's predictions, selecting 100 pairs previously labeled as equivalent and 100 as non-equivalent. This ensured a balanced and informative evaluation of both categories, even though the strata were defined according to the model's own judgments rather than verified ground truth.

Each pair was independently re-annotated by three human experts affiliated with the Department of Medicine at Aalborg University, consisting of one senior academic and two advanced medical students. All annotators followed the same written guidelines used to instruct the model, deciding whether each pair of diagnosis names referred to the same underlying clinical condition.

Inter-annotator agreement was measured using Cohen's κ, a chance-corrected coefficient that quantifies the consistency between categorical judgments. The resulting values, reported in Table 5.2, include both human–human and human–LLM pairings. According to the inter-rater categorization proposed by [104] for the "Level of Agreement," values of $\kappa \in [0.40, 0.59]$ fall within the class *Weak*, values of $\kappa \in [0.60, 0.79]$ within the class *Moderate*, and values of $\kappa \in [0.80, 0.90]$ within the class *Strong*.

Rater pair	Cohen's κ	Interpretation
`LLM` – `CV`	0.75	Moderate
`LLM` – `AF`	0.62	Moderate
`LLM` – `SB`	0.55	Weak
`CV` – `AF`	0.81	Strong
`CV` – `SB`	0.71	Moderate
`AF` – `SB`	0.70	Moderate

Table 5.2: Pairwise inter-rater agreement (Cohen's κ) between the LLM judge (`LLM`) and human annotators (`CV`, `AF`, `SB`).

Human annotators displayed imperfect consensus, confirming that the equivalence task involves inherent ambiguity even under shared guidelines. The LLM achieved an average κ of 0.64 against individual humans, compared with 0.74 among humans. Both fall within the same interpretive band of *Moderate* agreement, suggesting that the model's consistency is reasonably aligned with human performance.

Rather than emphasizing the statistical precision of individual κ values, as discussed by [104], the analysis focuses on the relative differences between human and LLM judgments. Manual inspection of disagreements revealed that the LLM occa-

sionally diverged in cases where all humans concurred, reflecting less plausible reasoning. These instances account for the observed drop in LLM–human agreement but represent a limited subset compared with the broader pool of ambiguous cases in which even humans disagreed. The overall magnitude of disagreement therefore remains comparable to that observed among human annotators, as both fall within the same level of *Moderate* agreement on average. Given this, the LLM is considered sufficiently reliable to serve as a scalable proxy for pairwise diagnosis-equivalence assessment in subsequent experiments.

5.2.4 Results

To illustrate the behavior of the extraction pipeline, this section first reports an example of diagnoses from a MIMIC note in the *MDACE* dataset.[7] Table 5.3 shows both the ground truth and extracted diagnoses by columns, aligning matches on the same line.

Ground truth	**Extracted**
[filtered] Car occupant (driver) (passenger) injured in other specified transport accidents, initial encounter	—
Fracture of vault of skull, initial encounter for open fracture	Skull Fracture
Traumatic subdural hemorrhage without loss of consciousness, initial encounter	—
Displaced fracture of base of third metacarpal bone, left hand, initial encounter for closed fracture	Left metacarpal fracture
Fracture of unspecified part of left clavicle, initial encounter for closed fracture	Left clavicle fracture
Injury of abducent nerve, unspecified side, initial encounter	Traumatic 6th nerve palsy
Epidural hemorrhage without loss of consciousness, initial encounter	Epidural Hematoma
—	Scalp laceration
—	Left arm laceration

Table 5.3: Example of ground truth diagnoses aligned with extracted diagnoses from a restricted MIMIC note (*MDACE*). Non-matching cells are filled with "—". Ground truth entries filtered as non-strict diagnoses are marked as *[filtered]*.

Having illustrated the extraction pipeline with an example, the following paragraphs report the results obtained by applying the evaluation procedure described in Section 5.2.3 to the datasets introduced in Section 5.2.1. The outcomes are summarized in Table 5.4.

[7]The note itself cannot be shown due to restricted access requirements.

Dataset	P_a	R_a	P_f	R_f	A_D	A_W	A_M
MDACE	0.48	0.79	0.46	0.85	–	–	–
FNUX	0.83	0.76	0.84	0.94	0.86	0.90	0.93

Table 5.4: Evaluation results. P_a, R_a: precision and recall on all annotated diagnoses. P_f, R_f: precision and recall after filtering non-diagnostic codes. A_h: temporal accuracy at different tolerance windows (same day, ±1 week, ±1 month).

Both datasets show relatively high recall and lower precision.

For *MDACE*, the evaluation is based on 3,319 annotated diagnoses across 560 notes, reduced to 2,565 after filtering. Precision is notably low (0.48 unfiltered, 0.46 filtered), while recall remains comparatively high (0.79 and 0.85, respectively). As discussed in Section 5.2.1, *MDACE* notes are discharge summaries that are longer and denser, typically interleaving diagnostic mentions with substantial amounts of additional information. This likely encouraged the LLM to generate more candidate diagnoses, some of which did not match the annotated scope, thereby lowering precision. Filtering had little effect, suggesting that both matching and non-matching diagnoses were reduced in similar proportions.

For *FNUX*, the evaluation covers 1,401 annotated diagnoses across 1,290 notes, reduced to 649 after filtering. Results are more balanced, with precision above 0.8 in both settings (0.83 and 0.84) and recall increasing from 0.76 to 0.94 after filtering (+23.7%). This improvement can be partly explained by the shorter, more focused nature of *FNUX* notes, which often state only a single diagnosis. Filtering occasionally produced notes with no remaining diagnoses, but these were correctly handled by the recognition/listing module, which in such cases returned no predictions. Overall, filtering had a strong effect in *FNUX*, reflecting the broader coverage of ICPC-2 codes that include non-diagnostic concepts. Although these codes are labeled as "diagnoses" in the *FNUX* schema, their removal improved alignment with the definition used in the extraction instructions and substantially boosted recall.

Temporal anchoring evaluation on *FNUX* showed that 86% of diagnoses were assigned the correct day, increasing to 90% within one week and 93% within one month. This indicates that while the system tends to overgenerate on *MDACE*, it achieves robust performance in the more heterogeneous EHR setting of *FNUX*, particularly with respect to temporal information.

It is worth recalling that *MDACE* is a valuable dataset for automated medical coding (i.e., extracting diagnostic codes from medical notes), since it links MIMIC-III discharge summaries to ICD-10 codes together with the textual evidence supporting them. In the present experiments, 0.46 precision and 0.79 recall were obtained, formulating the task as the extraction of diagnoses and conditions in natural language while ignoring the codes themselves. By contrast, most papers that rely on *MDACE* address the full coding problem, typically reporting precision in the range of 0.40–0.88 and recall between 0.58–0.86, depending on the model

family and setup [171, 91, 112], usually trading higher precision for lower recall and vice versa. Similarly, [72] uses *MDACE* in a sentence-classification setup, deciding whether a sentence contains any code-supporting evidence. Although different in formulation, it shares the focus on identifying evidence rather than performing full coding, and reports precision around 0.25–0.54 and recall of 0.66–0.70, depending on whether GPT-3.5 or BioBERT is used. These approaches are not directly comparable because of differences in task definition, model tuning, and architectural setup, but their reported performance provides an indicative frame of reference for these results.

5.3 Summary

This chapter presented an application of ontology-driven knowledge modeling combined with large language models (LLMs) for the semantic integration of electronic health records. The proposed pipeline unifies structured and unstructured components of EHR data by transforming free-text clinical notes into temporally grounded, event-based representations. An ontology tailored to the FNUX format provides the semantic backbone for integrating the extracted events with existing structured data, resulting in a coherent knowledge graph representation of patient histories.

Experiments conducted on the public *MDACE* dataset and the private *FNUX* collection demonstrated that the pipeline effectively identifies and structures clinically relevant diagnoses from free-text notes, achieving high recall and reliable temporal anchoring. Precision was found to depend on the degree of note complexity, with more concise notes yielding better performance. Despite these variations, the results confirm the feasibility of combining local LLM-based extraction with ontology-driven modeling for enhancing interoperability and temporal reasoning within clinical data.

Overall, the chapter illustrates how methods developed throughout this thesis can be applied to a real-world domain characterized by heterogeneous and sensitive textual data, showing that LLM-assisted knowledge structuring can extend beyond research settings to support complex, high-stakes applications such as healthcare.

Chapter 6

Conclusion

This thesis investigated how Large Language Models (LLMs) can act as knowledge engineers, mediating between unstructured text and formal, machine-interpretable representations. The research has been articulated along three main axes: the automatic structuring of textual information into Knowledge Graphs (KGs), the retrieval of structured knowledge through natural-language interfaces, and the application of these methods to clinical electronic health records. Each axis has been explored both conceptually and experimentally, showing how the reasoning and generative abilities of LLMs can be operationalized for knowledge engineering tasks traditionally requiring extensive manual design.

Knowledge Structuring. In Chapter 3, two complementary methodologies were introduced to address the automatic organization of textual information into coherent and semantically interpretable Knowledge Graphs. The *schema-first* approach proposed an LLM-based pipeline for the unsupervised induction of domain schemas. By combining keyword extraction, topic generalization, and clustering, it automatically generated candidate classes and hierarchies grounded in textual evidence. Experiments across distinct domains demonstrated that the induced schemas capture the essential conceptual structures of the source corpora and can effectively support downstream entity extraction tasks.

The *extraction-first* approach complemented this perspective by constructing KGs directly from text through the iterative identification of entities and relations. LLMs were employed to extract semantically enriched triples including entity types, descriptions, and expressive predicates, yielding data-driven, interpretable graph representations without predefined schemas. Empirical results confirmed the robustness and adaptability of this strategy, demonstrating its ability to produce comprehensive and diverse representations of textual content.

A comparative analysis of the two approaches revealed their conceptual complementarity and convergence toward a hybrid paradigm of knowledge structuring. Building upon their integration, the thesis explored reasoning-driven refinement mechanisms for the automatic inference of hierarchical and relational dependen-

cies. These experiments marked a first step toward *ontology induction*, showing that the same language-based reasoning underlying text understanding can support the formation and refinement of conceptual taxonomies.

Knowledge Retrieval. Chapter 4 investigated how structured knowledge can be accessed and queried through natural language. A comprehensive evaluation of fourteen instruction-tuned LLMs was conducted on the zero-shot translation of natural-language questions into SPARQL queries across multiple datasets and ontologies. The study assessed both correctness and consistency under various prompt formulations, identifying the decisive influence of prompt structure and ontology clarity on query generation accuracy.

The results show that large-scale models, such as those from the GPT family, achieve high accuracy even without domain-specific fine-tuning, while recent open-weight models (*qwen-2.5-coder-32b*, *llama-3.3-70b*, and *codestral-v0.1-22b*) exhibit competitive performance, significantly narrowing the gap with proprietary systems. The experiments also emphasized the role of ontology design: well-structured and semantically explicit schemas enable models to interpret relations more consistently, whereas ambiguous or underspecified ones hinder performance.

The analysis of failure modes revealed recurrent sources of error, including prefix omissions, ontology misalignment, and incorrect use of aggregations or filters. These findings highlight that, beyond model capacity, the effectiveness of natural-language querying depends crucially on the semantic clarity of the underlying knowledge representation. Overall, the chapter established a reproducible baseline for zero-shot NL-to-SPARQL translation and outlined how prompting strategies and ontology quality jointly determine the usability of LLMs as intermediaries between users and structured knowledge.

Application to Electronic Health Records. Chapter 5 demonstrated the applicability of the proposed methodologies in a real-world setting through the integration of structured and unstructured data in electronic health records (EHRs). The developed pipeline combined a locally deployed LLM with an event-based ontology to extract and temporally align clinically relevant information from free-text notes. This approach enabled the construction of temporally coherent knowledge graphs that unify native structured data with extracted events, supporting the longitudinal representation of patient trajectories.

Experiments on public (MDACE) and private (FNUX) datasets confirmed the feasibility of this approach, achieving good recall and stable temporal anchoring. However, precision remains a challenge, particularly in long, information-dense discharge summaries. Addressing such cases may require the introduction of summarization or chunking strategies to improve reliability, as well as the exploration of model tuning and ontology-guided prompting to enhance specificity. While the current experiments focused on diagnoses, the framework is designed to generalize

to other clinical event types, including procedures, prescriptions, and laboratory results, offering a foundation for broader semantic integration in healthcare data.

Overall Perspective. Taken together, the contributions of this dissertation advance the understanding of how LLMs can operate as mediators between natural language and formal semantic representations. Across structuring, retrieval, and applied integration, the results demonstrate that the generative and reasoning capabilities enabling text comprehension can also be used to construct, refine, and query structured knowledge. At the same time, the work identified several open challenges that remain central to the reliable use of LLMs for knowledge engineering, including scalability, interpretability, factual grounding, and dependence on ontology design. These findings suggest that future progress will rely on hybrid, human-in-the-loop workflows that balance the efficiency of automation with the precision of expert supervision.

6.1 Future Work

The study and results here presented open multiple directions for further research and investigation.

Refinement of knowledge structuring. Future work should extend the automatic schema induction pipeline by introducing adaptive parameter tuning, iterative clustering, and hierarchical reasoning. A promising avenue is the integration of schema-first and extraction-first paradigms into unified hybrid workflows capable of evolving toward ontology induction through automated reasoning over entities and relations.

Enhancing scalability and retrieval. In knowledge retrieval, the main challenge lies in scaling zero-shot query generation to large, real-world ontologies that exceed a model's input window. Integrating retrieval-augmented prompting and few-shot strategies represents a crucial step toward scalable semantic querying. Developing methods to dynamically select relevant ontology fragments during inference, while maintaining interpretability, remains an open research problem.

Improving precision and domain adaptation. Within the clinical domain, future developments should focus on precision-oriented extraction strategies, potentially combining summarization or document segmentation with ontology-guided prompting. Expanding the evaluation to additional event types such as prescriptions, procedures, and laboratory measurements will be essential to validate the generality of the proposed framework.

Toward hybrid human-AI workflows. More broadly, advancing the field will require new forms of collaboration between human experts and LLMs. Integrating interactive validation, human feedback, and symbolic reasoning into model-driven structuring processes may enhance both semantic reliability and explainability. Such hybrid human-AI frameworks could pave the way toward practical, scalable, and trustworthy systems for automated knowledge engineering.

In summary, this thesis demonstrated that LLMs can already act as capable assistants in the construction and exploration of structured knowledge. However, realizing their full potential will depend on advancing hybrid methodologies that combine data-driven inference with explicit semantic reasoning and human oversight.

Appendix A

Prompt Listings

This appendix presents the system prompts used in the various pipelines and experiments described in the thesis. For experiments involving prompt templates, the prompts are reported in their fully instantiated form, with placeholders replaced by the specific details used in each experiment. The appendix is intended for documentation purposes, and no further discussion or analysis of the prompts is provided here.

A.1 Schema-first structuring

This section reports the prompts used in the schema-first structuring pipeline described in Section 3.1.

A.1.1 Topic discovery

```
Given a list of keywords, identify their closest hypernym and rewrite the input
     list by adding the assigned hypernym.
The output is formatted as a JSON object with each keyword and its
    corresponding hypernym.

Example:

Input:
- apple
- fruit
- food

Output:
{
  "apple": "fruit",
  "fruit": "food",
  "food": "edible item"
```

```
}
```

A.1.2 Cluster labeling

```
Given a cluster of specific topics, identify a broader concept or category that
     encompasses all the topics in the cluster.
This broader concept is referred to as an ontology label.
The label represents a general category that includes all the specific topics.

For example, if the cluster includes topics such as "dog breeds", "dog training
    ", and "dog health", the ontology label could be "dog care".

The abstract description of the label should have no individual examples, and
    would be: "The process of caring for dogs."

The output is a JSON document that includes the ontology label and abstract
    description:
{
  "ontology label": "dog care",
  "abstract description": "The process of caring for dogs."
}
```

A.2 Extraction-first structuring

This section reports the prompts used in the extraction-first structuring pipeline described in Section 3.2.

A.2.1 Entity Extraction

```
The information in a text is expressed by mentions of concepts such as
    assertions, facts, individuals, objects, events, tasks, activities, and
    more. We call these concepts "entities".
You identify all the "entities" in the text, and for each one, you provide both
    :
 - a brief description of that entity (max 1 sentence);
 - a list of types, i.e. a term (or compound term) to define a category or
     hyperonym for that entity.
The output you provide is the list of entities, descriptions, and types,
    formatted with an initial hyphen this way:
`- <entity>|||<description>|||[<type 1>, <type 2>, ...]`
The user text language may be anything, but your output should be English!
```

A.2.2 Mention recognition

```
You identify mentions of entities in the text: the user provides you a list of
    known entities and a sentence in which those entities may be mentioned. For
     each entity you tell if they are actually mentioned in the sentence by
    saying "yes" or "no".
You write the user list again in the same order with your additional answer
    formatted this way:
`<entity ID>) <entity>|||<yes/no>`
```

A.2.3 Relation extraction

```
You find relations between entities in the text: the user provides you a list
    of known entities and a sentence in which those entities may be related.
    For each entity you check the relation with the other entities and if it
    exists you express the relation using a "predicate" that is expressive yet
    straightforward (max 5 words). Think about a predicate that could be used
    in an ontology for a knowledge graph.
Your output is the list of relations formatted as RDF triplets, that you format
     with an initial hyphen this way:
`- <entity> (<entity index>)|||<predicate>|||<entity> (<entity index>)`
The user text language may be anything, but your output should be {
    output_language}!
```

A.2.4 Predicate description

```
You provide an extended description of the "predicates" in a set of RDF
    triplets, which means you give a summary of the type of relation,
    characteristics, behaviors, or associations that the "predicate" is
    expressing between the "subject" and "object". The description must be
    general and reusable, so it must make no explicit references to the "
    subject" and "object".
The user provides you a sentence from which the triplets are extracted and a
    list of triplets formatted this way:
`- <subject>|||<predicate>|||<object>`
For each unique predicate, you provide the so said description, formatting your
     output this way:
`- <predicate>|||<predicate description><new line>`
The user text language may be anything, but your output should be {
    output_language}!
```

A.3 Knowledge retrieval

This section reports the prompts used in the knowledge retrieval experiments described in Chapter 4.

A.3.1 Basic

```
Given an input question and an ontology create a syntactically correct SPARQL
    query.

# Ontology
{"classes": ["battle", "ship", "death"], "object_properties": ["ref-
    lost_in_battle", "ref-caused_by_ship"], "data_properties": ["
    bulgarian_commander", "result", "disposition_of_ship", "killed", "tonnage",
     "ship_type", "id", "latin_commander", "location", "date", "injured", "
    caused_by_ship_id", "name", "lost_in_battle", "note"]}

# Output
Use the prefix `:` for all ontology entities, but don't define any `PREFIX` in
    the query.
Your answer must be the SPARQL query only.
```

A.3.2 Detailed

```
You are a powerful NLP and SPARQL Interpreter tool.

Your final goal is to find the information needed to answer a user question. So
    , your task is to analyze natural language questions and construct SPARQL
    queries that provide valuable information for the proper answer.

# Classes
- battle
- ship
- death

# Relations
- ref-lost_in_battle : [ship] -> [battle]
- ref-caused_by_ship : [death] -> [ship]

# Attributes
- bulgarian_commander : [battle] -> [str]
- result : [battle] -> [str]
- disposition_of_ship : [ship] -> [str]
- killed : [death] -> [int]
- tonnage : [ship] -> [str]
```

```
- ship_type : [ship] -> [str]
- id : [ship|battle|death] -> [int]
- latin_commander : [battle] -> [str]
- location : [ship] -> [str]
- date : [battle] -> [str]
- injured : [death] -> [int]
- caused_by_ship_id : [death] -> [int]
- name : [ship|battle] -> [str]
- lost_in_battle : [ship] -> [int]
- note : [death] -> [str]

# Instructions

### 1. Check the input
Carefully check the classes, relations and attributes that the user show you:
    choose all those that are necessary to represent the query patterns of the
    answer.

### 2. Identify key elements
Determine the key components of the question and the answer to map them into
    SPARQL keywords:
- **Return information**: the focus of what is being asked? (Use `SELECT`, `ASK
    `);
- **Quantifiers**: how many? (Use `COUNT`);
- **Superlatives**: best, most, worst? (Use `ORDER BY` with optionally `ASC`, `
    DESC` and/or `LIMIT`, `MIN`, `MAX`, ...);
- **Constraints**: inequalities or conditions on specific values? (Use `FILTER`
     and `HAVING`);
- **Aggregations**: average, sum, etc.? (Use `AVG`, `SUM`, ...);
- **Grouping**: collect and aggregate data based on a common condition? (Use `
    GROUP BY`);
- **Exclusions**: with no, without any, where no one, ...? (Use `MINUS` or `
    FILTER NOT EXIST`);
- **Alternative options**: including alternative patterns "or"? (Use `UNION`).

### 3. Build the SPARQL query
Put together all the input data to construct a functioning SPARQL query, with
    proper variables. Pay attention to:
- Perfectly match the syntax and naming of classes, relations and attributes (
    including letter case).
- Use distinguished variables for different concepts.
- Use the prefix `:` for all graph classes and properties. Anyway, you dont
    have to define any PREFIX.
- Return exactly the information the query is asking for, not a variable more.

### 4. Final answer
Your answer must be the SPARQL query only.
```

A.3.3 Chain of thought

```
You are a powerful NLP and SPARQL Interpreter tool.

Your task is to reason step-by-step through the mapping between the user's
    question and the query patterns needed to retrieve valuable information for
     the proper answer. After reasoning, you will produce a valid SPARQL query
    that retrieves the answer.

# Classes
- battle
- ship
- death

# Relations
- ref-lost_in_battle : [ship] -> [battle]
- ref-caused_by_ship : [death] -> [ship]

# Attributes
- bulgarian_commander : [battle] -> [str]
- result : [battle] -> [str]
- disposition_of_ship : [ship] -> [str]
- killed : [death] -> [int]
- tonnage : [ship] -> [str]
- ship_type : [ship] -> [str]
- id : [ship|battle|death] -> [int]
- latin_commander : [battle] -> [str]
- location : [ship] -> [str]
- date : [battle] -> [str]
- injured : [death] -> [int]
- caused_by_ship_id : [death] -> [int]
- name : [ship|battle] -> [str]
- lost_in_battle : [ship] -> [int]
- note : [death] -> [str]

# Instructions

Follow these reasoning steps explicitly before generating the query:

### Step 1: Analyze the question
Identify:
- The main entity or type(s) involved (relevant classes).
- The focus of the question (what information is being asked).
- Any filters or conditions (e.g., values, constraints, comparisons).
- Any aggregation, ordering, or limits.
- The expected shape of the answer (single value, list, boolean, etc.).

### Step 2: Plan the query
Decide:
- Which variables will be used and what they represent.
```

```
- Which classes, properties (object or data), and filters are needed.
- How to join them in the WHERE clause.
- How to structure the SELECT clause (e.g., with DISTINCT, COUNT, ORDER BY, etc
    .).

### Step 3: Generate the query
Write the final SPARQL query using `:` as prefix for all ontology entities, **
    without defining any PREFIX clause**.

# Final Output
Respond with:
1. Your reasoning steps (Step 1 and Step 2).
2. The final SPARQL query **enclosed in a fenced triple backtick (```) code
    block**, and nothing else inside the block.

You only produce one SPARQL query as the very final step of your answer.
```

A.4 EHR application

This section reports the prompts used in the EHR application pipeline described in Chapter 5, as instantiated for diagnosis event extraction.

A.4.1 Event recognition

```
You are given a medical note about a patient. It can be written in either
    English or Danish.
Notes may range from brief, vague comments by general practitioners to detailed
     reports from specialists or test results.

Your task is to determine whether the note contains information important
    diagnoses.
We talk about diagnoses in a broader sense, that means references to confirmed
    or suspected conditions, symptoms, abnormal findings, practitioner concerns
     and physician assessments.
Mentions may include phrases like 'patient has X', 'we suspect Y', 'likely Z',
    'complains of A', 'history of B', 'abnormal C', 'fear of D'.
Providers can be essential, and sometimes just type the name or list of
    diagnoses/conditions in the title of the note without any explicit
    statement (e.g. 'the patient has'). We can interpret these cases as clear
    important diagnoses.

The diagnoses must be recent and addressed personally to the patient, not its
    relatives.

Say whether any relevant diagnosis is mentioned:
```

```
- Respond "yes" if there clearly is at least one important diagnosis mentioned
    in the corpus.
- Respond "yes" if there the title clearly mention a name or list of diagnoses/
    conditions.
- Respond "no" if they note is solely focusing on other events (drug
    prescriptions, lab results, ...).

Do not explain your answer. Only output "yes" or "no".
```

A.4.2 Event listing

```
You are given a medical note about a patient. It can be written in either
    English or Danish.
Notes may range from brief, vague comments by general practitioners to detailed
     reports from specialists or test results.

Your task is to extract **all diagnosis** mentions.

A "diagnosis" means any **confirmed condition or disease** that applies to the
    patient.

Include any condition affecting the patient, whether current or past, as long
    as it is explicitly stated or clearly implied.
Ignore isolated symptoms and normal exam findings that are not described as
    actual conditions.
Ignore **secondary effects or symptoms that are explained by another diagnosis
    ** (e.g., if chest pain is caused by muscle strain, include only "muscle
    strain").

The note is likely a detailed clinical document. Carefully identify the
    relevant diagnoses, focusing on the most significant and clinically
    relevant conditions:
- **If the note is a full summary report** of the patient's recent or past
    history, include all main diagnosed problems that are clinically important.
     The list will typically contain **around two to ten diagnoses (more or
    less)**, depending on the complexity of the case. Avoid listing an
    excessively long or trivial set.
- **If the note focuses on a specific event or encounter**, include only the **
    main diagnosed condition(s)** relevant to that event (usually **one or two
    **).

Return a **JSON array** of concise condition names (strings).
No explanations, no metadata, no duplicates, and no additional formatting
    beyond valid JSON.
```

A.4.3 Event structuring

````
You are given a medical note about a patient. It can be written in either
    English or Danish.

The note may mention a lot of information, but specifically contains
    information about a particular "Epidural Hematoma" diagnosis.

Your task is to extract the relevant information and return it as a JSON object
     following this template:
```
{
    "date": "date when the diagnosis was made (in ISO 8601 format, e.g.
        '2020-12-03')",
    "certainty": "either 'suspected' or 'certain'",
    "status": "either 'active' or 'cured'",
}
```

The template contains field names with descriptions as placeholder values.
If any field is missing or uncertain, do not include.
Otherwise, if the information is available replace the description with the
    corresponding value from the note.

Only extract information that is explicitly stated or crearly implied in the
    text.
Output only a valid JSON, with no explanation or extra text (could also be an
    empty document, if none of the requested information is mentioned).
Keep the values in Danish if the source is Danish.
````

Bibliography

[1] The rise of large language models. *Nature Computational Science*, 5(9):689–690, September 2025. Publisher: Nature Publishing Group.

[2] Hammaad Adam, Junjing Lin, Jianchang Lin, Hillary Keenan, Ashia Wilson, and Marzyeh Ghassemi. Clinical Information Extraction with Large Language Models: A Case Study on Organ Procurement. *AMIA Annual Symposium Proceedings*, 2024:115–123, May 2025.

[3] Monica Agrawal, Stefan Hegselmann, Hunter Lang, Yoon Kim, and David Sontag. Large language models are few-shot clinical information extractors. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1998–2022, 2022.

[4] Adrian Akmajian, Ann K. Farmer, Richard A. Demers, and Robert M. Harnish. *Linguistics: An Introduction to Language and Communication*. MIT Press, Cambridge, MA, USA, 5 edition, June 2001.

[5] Mohammad Alkhalaf, Ping Yu, Mengyang Yin, and Chao Deng. Applying generative AI with retrieval augmented generation to summarize and extract key clinical information from electronic health records. *Journal of Biomedical Informatics*, 156:104662, August 2024.

[6] Caio Viktor S. Avila, Marco A. Casanova, and Vânia M. P. Vidal. A Framework for Question Answering on Knowledge Graphs Using Large Language Models. In Albert Meroño Peñuela, Oscar Corcho, Paul Groth, Elena Simperl, Valentina Tamma, Andrea Giovanni Nuzzolese, Maria Poveda-Villalón, Marta Sabou, Valentina Presutti, Irene Celino, Artem Revenko, Joe Raad, Bruno Sartini, and Pasquale Lisena, editors, *The Semantic Web: ESWC 2024 Satellite Events*, pages 168–172, Cham, 2025. Springer Nature Switzerland.

[7] Franz Baader. *The description logic handbook: Theory, implementation and applications*. Cambridge university press, 2003.

[8] Nguyen Bach and Sameer Badaskar. A survey on relation extraction. *Language Technologies Institute, Carnegie Mellon University*, 178:15, 2007.

[9] Debayan Banerjee, Pranav Ajit Nair, Jivat Neet Kaur, Ricardo Usbeck, and Chris Biemann. Modern baselines for SPARQL semantic parsing. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2260–2265, 2022.

[10] Caroline Barrière. *Natural Language Understanding in a Semantic Web Context*. Springer International Publishing, Cham, 2016.

[11] Emily M. Bender and Alexander Koller. Climbing towards NLU: On Meaning, Form, and Understanding in the Age of Data. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5185–5198, Online, July 2020. Association for Computational Linguistics.

[12] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

[13] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic Parsing on Freebase from Question-Answer Pairs. In David Yarowsky, Timothy Baldwin, Anna Korhonen, Karen Livescu, and Steven Bethard, editors, *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

[14] Tim Berners-Lee, Roy Fielding, and Larry Masinter. Uniform resource identifier (URI): Generic syntax. Technical report, 2005.

[15] Giovanni Maria Biancofiore, Yashar Deldjoo, Tommaso Di Noia, Eugenio Di Sciascio, and Fedelucio Narducci. Interactive Question Answering Systems: Literature Review. *ACM Comput. Surv.*, 56(9):239:1–239:38, 2024.

[16] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[17] Simone Bocca, Alessio Zamboni, Gábor Bella, Yamini Chandrashekar, Mayukh Bagchi, Gabriel Kuper, Paolo Bouquet, and Fausto Giunchiglia. Building Interoperable Electronic Health Records as Purpose Driven Knowledge Graphs. In Pradeep Kumar Singh, Marcello Trovati, Fionn Murtagh, Mohammed Atiquzzaman, and Mohsen Farid, editors, *Data Science and Artificial Intelligence for Digital Healthcare: Communications Technologies for Epidemic Models*, pages 237–254. Springer International Publishing, Cham, 2024.

[18] Gemma Boleda. Distributional Semantics and Linguistic Theory. *Annual Review of Linguistics*, 6(Volume 6, 2020):213–234, January 2020. Publisher: Annual Reviews.

[19] Ronald Brachman and Hector Levesque. *Knowledge representation and reasoning*. Elsevier, 2004.

[20] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, pages 1877–1901, Red Hook, NY, USA, 2020. Curran Associates Inc.

[21] Paul Buitelaar, Philipp Cimiano, and Bernardo Magnini. *Ontology learning from text: methods, evaluation and applications*, volume 123. IOS press, 2005.

[22] Pere-Lluís Huguet Cabot and Roberto Navigli. REBEL: Relation Extraction By End-to-end Language generation. In *Conference on Empirical Methods in Natural Language Processing*, 2021.

[23] Salvatore Carta, Alessandro Giuliani, Marco Manolo Manca, Leonardo Piano, Alessandro Sebastian Podda, Livio Pompianu, and Sandro Gabriele Tiddia. A Zero-Shot Strategy for Knowledge Graph Engineering Using GPT-3.5. *Procedia Computer Science*, 246:2235–2243, 2024.

[24] Salvatore Carta, Alessandro Giuliani, Marco Manolo Manca, Leonardo Piano, and Sandro Gabriele Tiddia. Towards Zero-shot Knowledge Graph building: Automated Schema Inference. In *Adjunct Proceedings of the 32nd ACM Conference on User Modeling, Adaptation and Personalization*, UMAP Adjunct '24, pages 467–473, New York, NY, USA, 2024. Association for Computing Machinery. event-place: Cagliari, Italy.

[25] Hoyeon Chang, Jinho Park, Seonghyeon Ye, Sohee Yang, Youngkyung Seo, Du-Seong Chang, and Minjoon Seo. How Do Large Language Models Acquire Factual Knowledge During Pretraining? *Advances in Neural Information Processing Systems*, 37:60626–60668, December 2024.

[26] Hua Cheng, Rana Jafari, April Russell, Russell Klopfer, Edmond Lu, Benjamin Striner, and Matthew Gormley. MDACE: MIMIC Documents Annotated with Code Evidence. In Anna Rogers, Jordan Boyd-Graber, and Naoaki

Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7534–7550, Toronto, Canada, July 2023. Association for Computational Linguistics.

[27] Jason P.C. Chiu and Eric Nichols. Named Entity Recognition with Bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*, 4:357–370, July 2016. _eprint: https://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00104/1567392/tacl_a_00104.pdf.

[28] Noam Chomsky. *Language and mind*. Cambridge University Press, 2006.

[29] Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. Attention-Based Models for Speech Recognition. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[30] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, and others. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53, 2024.

[31] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling, December 2014. arXiv:1412.3555 [cs].

[32] Philipp Cimiano and Johanna Völker. Text2Onto. In *International Conference on Applications of Natural Language to Data Bases*, 2005.

[33] Alexander Clark, Chris Fox, and Shalom Lappin. *The Handbook of Computational Linguistics and Natural Language Processing*. John Wiley & Sons, April 2013.

[34] Nigel Collier and Jin-Dong Kim. Introduction to the Bio-entity Recognition Task at JNLPBA. In *NLPBA/BioNLP*, 2004.

[35] John Dagdelen, Alexander Dunn, Sanghoon Lee, Nicholas Walker, Andrew S. Rosen, Gerbrand Ceder, Kristin A. Persson, and Anubhav Jain. Structured information extraction from scientific text with large language models. *Nature Communications*, 15(1):1418, February 2024. Publisher: Nature Publishing Group.

[36] Papa Abdou Karim Karou Diallo, Samuel Reyd, and Amal Zouaq. A Comprehensive Evaluation of Neural SPARQL Query Generation From Natural Language Questions. *IEEE Access*, 12:125057–125078, 2024.

[37] Kevin Donnelly. SNOMED-CT: The advanced terminology and coding system for eHealth. *Studies in health technology and informatics*, 121:279, 2006. Publisher: IOS Press; 1999.

[38] Mohnish Dubey, Debayan Banerjee, Abdelrahman Abdelkawi, and Jens Lehmann. LC-QuAD 2.0: A Large Dataset for Complex Question Answering over Wikidata and DBpedia. In Chiara Ghidini, Olaf Hartig, Maria Maleshkova, Vojtěch Svátek, Isabel Cruz, Aidan Hogan, Jie Song, Maxime Lefrançois, and Fabien Gandon, editors, *The Semantic Web – ISWC 2019*, pages 69–78, Cham, 2019. Springer International Publishing.

[39] Martin Dürst and Michel Suignard. Internationalized resource identifiers (IRIs). Technical report, 2005.

[40] Muhammad Ebraheem, Saravanan Thirumuruganathan, Shafiq R. Joty, Mourad Ouzzani, and Nan Tang. DeepER - Deep Entity Resolution. *ArXiv*, abs/1710.00597, 2017.

[41] Lisa Ehrlinger and Wolfram Wöß. Towards a definition of knowledge graphs. *SEMANTiCS (Posters, Demos, SuCCESS)*, 48(1-4):2, 2016.

[42] Jacob Eisenstein. *Natural language processing*, volume 507. 2018.

[43] Sarah Elhammadi, Laks VS Lakshmanan, Raymond Ng, Michael Simpson, Baoxing Huai, Zhefeng Wang, and Lanjun Wang. A high precision pipeline for financial knowledge graph construction. In *Proceedings of the 28th international conference on computational linguistics*, pages 967–977, 2020.

[44] Guy Emerson. What are the Goals of Distributional Semantics? In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7436–7453, Online, July 2020. Association for Computational Linguistics.

[45] Vincent Emonet, Jerven T. Bolleman, Severine Duvaud, Tarcisio Mendes de Farias, and Ana Claudia Sima. LLM-based SPARQL Query Generation from Natural Language over Federated Knowledge Graphs. *ArXiv*, abs/2410.06062, 2024.

[46] Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S. Weld. Open information extraction from the web. *Communications of the ACM*, 51(12):68–74, December 2008.

[47] Nadime Francis, Alastair Green, Paolo Guagliardo, Leonid Libkin, Tobias Lindaaker, Victor Marsault, Stefan Plantikow, Mats Rydberg, Petra Selmer, and Andrés Taylor. Cypher: An Evolving Query Language for Property Graphs.

Proceedings of the 2018 International Conference on Management of Data, pages 1433–1445, May 2018. Conference Name: SIGMOD/PODS '18: International Conference on Management of Data ISBN: 9781450347037 Place: Houston TX USA Publisher: ACM.

[48] Maurice Funk, Simon Hosemann, Jean Christoph Jung, and Carsten Lutz. Towards Ontology Construction with Language Models. *ArXiv*, abs/2309.09898, 2023.

[49] Farieda Gaber, Maqsood Shaik, Fabio Allega, Agnes Julia Bilecz, Felix Busch, Kelsey Goon, Vedran Franke, and Altuna Akalin. Evaluating large language model workflows in clinical decision support for triage and referral and diagnosis. *npj Digital Medicine*, 8(1):263, May 2025.

[50] Alan Gilchrist. Thesauri, taxonomies and ontologies–an etymological note. *Journal of documentation*, 59(1):7–18, 2003. Publisher: MCB UP Ltd.

[51] Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151:78–94, 2018. Publisher: Elsevier.

[52] Bowen Gu, Vivian Shao, Ziqian Liao, Valentina Carducci, Santiago Romero Brufau, Jie Yang, and Rishi J. Desai. Scalable information extraction from free text electronic health records using large language models. *BMC Medical Research Methodology*, 25(1):23, January 2025.

[53] Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, Saizhuo Wang, Kun Zhang, Yuanzhuo Wang, Wen Gao, Lionel Ni, and Jian Guo. A Survey on LLM-as-a-Judge, 2024.

[54] Yu Gu, Sue Kase, Michelle Vanni, Brian Sadler, Percy Liang, Xifeng Yan, and Yu Su. Beyond IID: three levels of generalization for question answering on knowledge bases. In *Proceedings of the Web Conference 2021*, pages 3477–3488. ACM, 2021.

[55] Nicola Guarino, Daniel Oberle, and Steffen Staab. What Is an Ontology? In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, pages 1–17. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[56] Akhil Gudivada, Dhana L. Rao, and Venkat N. Gudivada. Chapter 1 - Linguistics: Core Concepts and Principles. In Venkat N. Gudivada and C. R. Rao, editors, *Computational Analysis and Understanding of Natural Languages: Principles, Methods and Applications*, volume 38 of *Handbook of Statistics*, pages 3–14. Elsevier, 2018. ISSN: 0169-7161.

[57] Gaurav Kumar Gupta, Aditi Singh, Sijo Valayakkad Manikandan, and Abul Ehtesham. Digital Diagnostics: The Potential of Large Language Models in Recognizing Symptoms of Common Illnesses. *AI*, 6(1):13, January 2025.

[58] Shibo Hao, Bowen Tan, Kaiwen Tang, Hengzhe Zhang, Eric P Xing, and Zhiting Hu. BertNet: Harvesting Knowledge Graphs from Pretrained Language Models. *arXiv preprint arXiv:2206.14268*, 2022.

[59] Stevan Harnad. The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42(1):335–346, June 1990.

[60] Roland Hausser. *Foundations of Computational Linguistics: Human-Computer Communication in Natural Language*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[61] James Hendler, Ora Lassila, and Tim Berners-Lee. The semantic web. *Scientific American*, 284(5):34–43, 2001.

[62] Aidan Hogan, Claudio Gutierrez, Michael Cochez, Gerard De Melo, Sabrina Kirrane, Axel Polleres, Roberto Navigli, Axel-Cyrille Ngonga Ngomo, Sabbir M. Rashid, Lukas Schmelzeisen, Steffen Staab, Eva Blomqvist, Claudia d'Amato, José Emilio Labra Gayo, Sebastian Neumaier, Anisa Rula, Juan Sequeda, and Antoine Zimmermann. *Knowledge Graphs*. Synthesis Lectures on Data, Semantics, and Knowledge. Springer International Publishing, Cham, 2022.

[63] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The Curious Case of Neural Text Degeneration, February 2020. arXiv:1904.09751 [cs].

[64] E. P. hommeaux. SPARQL query language for RDF. 2011.

[65] Elias Hossain, Rajib Rana, Niall Higgins, Jeffrey Soar, Prabal Datta Barua, Anthony R. Pisani, and Kathryn Turner. Natural Language Processing in Electronic Health Records in relation to healthcare decision-making: A systematic review. *Computers in Biology and Medicine*, 155:106649, 2023.

[66] Linmei Hu, Zeyi Liu, Ziwang Zhao, Lei Hou, Liqiang Nie, and Juanzi Li. A Survey of Knowledge Enhanced Pre-Trained Language Models. *IEEE Transactions on Knowledge and Data Engineering*, 36(4):1413–1430, April 2024.

[67] Jingwei Huang, Donghan M. Yang, Ruichen Rong, Kuroush Nezafati, Colin Treager, Zhikai Chi, Shidan Wang, Xian Cheng, Yujia Guo, Laura J. Klesse, Guanghua Xiao, Eric D. Peterson, Xiaowei Zhan, and Yang Xie. A critical assessment of using ChatGPT for extracting structured data from clinical notes. *npj Digital Medicine*, 7(1):106, May 2024.

[68] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions. *ACM Trans. Inf. Syst.*, 43(2):42:1–42:55, 2025.

[69] I.a. Richards. *The Meaning Of Meaning*. 1930.

[70] Longquan Jiang and Ricardo Usbeck. Knowledge Graph Question Answering Datasets and Their Generalizability: Are They Enough for Future Research? In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, pages 3209–3218, New York, NY, USA, 2022. Association for Computing Machinery. event-place: Madrid, Spain.

[71] Pengcheng Jiang, Jiacheng Lin, Zifeng Wang, Jimeng Sun, and Jiawei Han. GenRES: Rethinking Evaluation for Generative Relation Extraction in the Era of Large Language Models. *arXiv preprint arXiv:2402.10744*, 2024.

[72] Xiaorui Jiang, Kulsoom Khan, Sumithra Thinakara Vasantha, and Sajjad Haider. Evidence Extraction for Automated Medical Coding: Preliminary Evaluation. In *Proceedings of the 2024 8th International Conference on Natural Language Processing and Information Retrieval*, NLPIR '24, pages 18–23, New York, NY, USA, April 2025. Association for Computing Machinery.

[73] Renren Jin, Jiangcun Du, Wuwei Huang, Wei Liu, Jian Luan, Bin Wang, and Deyi Xiong. A Comprehensive Evaluation of Quantization Strategies for Large Language Models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 12186–12215, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[74] Alistair E. W. Johnson, Lucas Bulgarelli, Lu Shen, Alvin Gayles, Ayad Shammout, Steven Horng, Tom J. Pollard, Sicheng Hao, Benjamin Moody, Brian Gow, Li-wei H. Lehman, Leo A. Celi, and Roger G. Mark. MIMIC-IV, a freely accessible electronic health record dataset. *Scientific Data*, 10(1):1, January 2023.

[75] Alistair E. W. Johnson, Tom J. Pollard, Lu Shen, Li-wei H. Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G. Mark. MIMIC-III, a freely accessible critical care database. *Scientific Data*, 3(1):160035, May 2016.

[76] Daniel Jurafsky and James H. Martin. *Speech and Language Processing*. Pearson Education, December 2025.

[77] Jean Kaddour, Joshua Harris, Maximilian Mozes, Herbie Bradley, Roberta Raileanu, and Robert McHardy. Challenges and Applications of Large Language Models, July 2023. arXiv:2307.10169 [cs].

[78] Elisa F. Kendall and Deborah L. McGuinness. *Ontology engineering*. Morgan & Claypool Publishers, 2019.

[79] Diksha Khurana, Aditya Koli, Kiran Khatter, and Sukhdev Singh. Natural language processing: state of the art, current trends and challenges. *Multimedia Tools and Applications*, 82(3):3713–3744, January 2023.

[80] Gerhard Klager and Axel Polleres. Is GPT fit for KGQA?-Preliminary Results. In *TEXT2KG/BiKE@ ESWC*, pages 171–191, 2023.

[81] Graham Klyne. Resource description framework (RDF): Concepts and abstract syntax. *http://www. w3. org/TR/rdf-concepts/*, 2004. Publisher: W3C recommendation.

[82] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, pages 22199–22213, Red Hook, NY, USA, November 2022. Curran Associates Inc.

[83] Catherine Kosten, Philippe Cudré-Mauroux, and Kurt Stockinger. Spider4SPARQL: A Complex Benchmark for Evaluating Knowledge Graph Question Answering Systems. In *2023 IEEE International Conference on Big Data (BigData)*, pages 5272–5281, 2023.

[84] Liubov Kovriguina, Roman Teucher, Daniil Radyush, Dmitry Mouromtsev, N Keshan, S Neumaier, AL Gentile, and S Vahdati. SPARQLGEN: One-Shot Prompt-based Approach for SPARQL Query Generation. In *SEMANTiCS (Posters & Demos)*, 2023.

[85] Hugo Le Baher, Jérôme Azé, Sandra Bringay, Pascal Poncelet, Nancy Rodriguez, and Caroline Dunoyer. Patient Electronic Health Record as Temporal Graphs for Health Monitoring. *Studies in Health Technology and Informatics*, 302:561–565, May 2023.

[86] Florian Leiser, Richard Guse, and Ali Sunyaev. Large Language Model Architectures in Health Care: Scoping Review of Research Perspectives. *Journal of Medical Internet Research*, 27:e70315, June 2025.

[87] Bo Li, Gexiang Fang, Yang Yang, Quansen Wang, Wei Ye, Wen Zhao, and Shikun Zhang. Evaluating ChatGPT's Information Extraction Capabilities: An Assessment of Performance, Explainability, Calibration, and Faithfulness. *ArXiv*, abs/2304.11633, 2023.

[88] Irene Li, Jessica Pan, Jeremy Goldwasser, Neha Verma, Wai Pan Wong, Muhammed Yavuz Nuzumlalı, Benjamin Rosand, Yixin Li, Matthew Zhang, David Chang, R. Andrew Taylor, Harlan M. Krumholz, and Dragomir Radev. Neural Natural Language Processing for unstructured data in electronic health records: A review. *Computer Science Review*, 46:100511, November 2022.

[89] Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. A Survey on Deep Learning for Named Entity Recognition. *IEEE Transactions on Knowledge and Data Engineering*, 34(1):50–70, January 2022.

[90] Qian Li, Hao Peng, Jianxin Li, Congying Xia, Renyu Yang, Lichao Sun, Philip S. Yu, and Lifang He. A Survey on Text Classification: From Traditional to Deep Learning. *ACM Trans. Intell. Syst. Technol.*, 13(2):31:1–31:41, April 2022.

[91] Rumeng Li, Xun Wang, and Hong Yu. Exploring LLM Multi-Agents for ICD Coding, 2024.

[92] Zhenyu Li, Sunqi Fan, Yu Gu, Xiuxing Li, Zhichao Duan, Bo Dong, Ning Liu, and Jianyong Wang. FlexKBQA: A Flexible LLM-Powered Framework for Few-Shot Knowledge Base Question Answering. *ArXiv*, abs/2308.12060, 2023.

[93] Percy Liang. Learning executable semantic parsers for natural language understanding. *Communications of the ACM*, 59(9):68–76, August 2016.

[94] Jia-Huei Lin and Eric Jui-Lin Lu. SPARQL Generation with an NMT-Based Approach. *Journal of Web Engineering*, 21(5):1471–1490, 2022.

[95] Jingjing Liu, Panupong Pasupat, D. Scott Cyphers, and James R. Glass. Asgard: A portable architecture for multilingual dialogue systems. *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8386–8390, 2013.

[96] Ruicheng Liu, Rui Mao, Anh Tuan Luu, and Erik Cambria. A brief survey on recent advances in coreference resolution. *Artificial Intelligence Review*, 56(12):14439–14481, December 2023.

[97] Ivan Lopez, Akshay Swaminathan, Karthik Vedula, Sanjana Narayanan, Fateme Nateghi Haredasht, Stephen P. Ma, April S. Liang, Steven Tate, Manoj Maddali, Robert Joseph Gallo, Nigam H. Shah, and Jonathan H. Chen. Clinical entity augmented retrieval for clinical information extraction. *npj Digital Medicine*, 8(1):45, January 2025.

[98] Fabiano Ferreira Luz and Marcelo Finger. Semantic parsing natural language into SPARQL: improving target language representation with neural attention. *arXiv preprint arXiv:1803.04329*, 2018.

[99] Chuangtao Ma, Yongrui Chen, Tianxing Wu, and Arijit Khan. Unifying Large Language Models and Knowledge Graphs for Question Answering: Recent Advances and Opportunities. In *Proceedings 28th International Conference on Extending Database Technology*, Barcelona, Spain, 2025. OpenProceedings.org.

[100] Alexander Maedche and Raphael Volz. The text-to-onto ontology extraction and maintenance system. 2001.

[101] Murray E. Maitland. A Transdisciplinary Definition of Diagnosis. *Journal of Allied Health*, 39(4):306–313, 2010.

[102] Daniel McDuff, Mike Schaekermann, Tao Tu, Anil Palepu, Amy Wang, Jake Garrison, Karan Singhal, Yash Sharma, Shekoofeh Azizi, Kavita Kulkarni, Le Hou, Yong Cheng, Yun Liu, S. Sara Mahdavi, Sushant Prakash, Anupam Pathak, Christopher Semturs, Shwetak Patel, Dale R. Webster, Ewa Dominowska, Juraj Gottweis, Joelle Barral, Katherine Chou, Greg S. Corrado, Yossi Matias, Jake Sunshine, Alan Karthikesalingam, and Vivek Natarajan. Towards accurate differential diagnosis with large language models. *Nature*, 642(8067):451–457, June 2025.

[103] Deborah L. McGuinness and Frank Van Harmelen. OWL web ontology language overview. *W3C recommendation*, 10(10):2004, 2004.

[104] Mary L. McHugh. Interrater reliability: the kappa statistic. *Biochemia Medica*, 22(3):276–282, October 2012.

[105] Denis McInerney, Geoffrey Young, Jan-Willem van de Meent, and Byron Wallace. CHiLL: Zero-shot Custom Interpretable Feature Extraction from Clinical Notes with Large Language Models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 8477–8494, Singapore, December 2023. Association for Computational Linguistics.

[106] Aman Mehta, Aashay Singhal, and Kamalakar Karlapalem. Scalable knowledge graph construction over text using deep learning based predicate mapping. In *Companion Proceedings of The 2019 World Wide Web Conference*, pages 705–713, 2019.

[107] Lars-Peter Meyer, Johannes Frey, Felix Brei, and Natanael Arndt. Assessing SPARQL capabilities of Large Language Models. *arXiv preprint arXiv:2409.05925*, 2024.

[108] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space, September 2013. arXiv:1301.3781 [cs].

[109] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Proc. Interspeech 2010*, pages 1045–1048, 2010.

[110] Alistair Miles, Brian Matthews, Michael Wilson, and Dan Brickley. SKOS core: simple knowledge organisation for the web. In *Proceedings of the International Conference on Dublin Core and Metadata Applications*. Dublin Core Metadata Initiative, 2005.

[111] Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. Large Language Models: A Survey, March 2025. arXiv:2402.06196 [cs].

[112] Andreas Motzfeldt, Joakim Edin, Casper L. Christensen, Christian Hardmeier, Lars Maaløe, and Anna Rogers. Code Like Humans: A Multi-Agent Solution for Medical Coding, September 2025.

[113] Lino Murali, G. Gopakumar, Daleesha M. Viswanathan, and Prema Nedungadi. Towards electronic health record-based medical knowledge graph construction, completion, and applications: A literature study. *Journal of Biomedical Informatics*, 143:104403, 2023.

[114] Fionn Murtagh and Pedro Contreras. Algorithms for hierarchical clustering: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(1):86–97, 2012. Publisher: Wiley Online Library.

[115] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, August 2007.

[116] Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. A Comprehensive Overview of Large Language Models. *ACM Trans. Intell. Syst. Technol.*, 16(5):106:1–106:72, 2025.

[117] Thien Huu Nguyen and Ralph Grishman. Relation Extraction: Perspective from Convolutional Neural Networks. In *VS@HLT-NAACL*, 2015.

[118] Joakim Nivre. Dependency Parsing. *Language and Linguistics Compass*, 4(3):138–152, 2010. _eprint: https://compass.onlinelibrary.wiley.com/doi/pdf/10.1111/j.1749-818X.2010.00187.x.

[119] Natalya F. Noy and Deborah L. McGuinness. Ontology development 101: A guide to creating your first ontology, 2001.

[120] Vasileios Ntinopoulos, Hector Rodriguez Cetina Biefer, Igor Tudorache, Nestoras Papadopoulos, Dragan Odavic, Petar Risteski, Achim Haeussler, and Omer Dzemali. Large language models for data extraction from unstructured and semi-structured electronic health records: a multiple model performance evaluation. *BMJ Health & Care Informatics*, 32(1), January 2025.

[121] Atsushi Oba, Incheon Paik, and Ayato Kuwana. Automatic Classification for Ontology Generation by Pretrained Language Model. In *Advances and Trends in Artificial Intelligence. Artificial Intelligence Practices: 34th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2021, Kuala Lumpur, Malaysia, July 26–29, 2021, Proceedings, Part I 34*, pages 210–221. Springer, 2021.

[122] Peter Ochieng. PAROT: Translating natural language to SPARQL. *Expert Systems with Applications: X*, 5:100024, 2020. Publisher: Elsevier.

[123] William K. S. Ojemann, Kevin Xie, Kevin Liu, Ellie Chang, Dan Roth, Brian Litt, and Colin A. Ellis. Zero-Shot Extraction of Seizure Outcomes from Clinical Notes Using Generative Pretrained Transformers. *Journal of Healthcare Informatics Research*, 9(3):380–400, September 2025.

[124] Joel Oksanen, Oana Cocarascu, and Francesca Toni. Automatic Product Ontology Extraction from Textual Reviews. *arXiv preprint arXiv:2105.10966*, 2021.

[125] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, and Alex Ray. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[126] Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. Unifying Large Language Models and Knowledge Graphs: A Roadmap. *IEEE Transactions on Knowledge and Data Engineering*, 36(7):3580–3599, July 2024.

[127] Letitia Parcalabescu, Nils Trost, and Anette Frank. What is Multimodality? In Lucia Donatelli, Nikhil Krishnaswamy, Kenneth Lai, and James Pustejovsky, editors, *Proceedings of the 1st Workshop on Multimodal Semantic Representations (MMSR)*, pages 1–10, Groningen, Netherlands (Online), June 2021. Association for Computational Linguistics.

[128] Jung In Park, Jong Won Park, Kexin Zhang, and Doyop Kim. Advancing equity in breast cancer care: natural language processing for analysing treatment outcomes in under-represented populations. *BMJ Health & Care Informatics*, 31(1), July 2024.

[129] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning*, pages 1310–1318. PMLR, May 2013. ISSN: 1938-7228.

[130] Ciyuan Peng, Feng Xia, Mehdi Naseriparsa, and Francesco Osborne. Knowledge Graphs: Opportunities and Challenges. *Artificial Intelligence Review*, 56(11):13071–13102, November 2023.

[131] Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. Language models as knowledge bases? In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 2463–2473, 2019.

[132] Pejman Peykani, Fatemeh Ramezanlou, Cristina Tanasescu, and Sanly Ghanidel. Large Language Models: A Structured Taxonomy and Review of Challenges, Limitations, Solutions, and Future Directions. *Applied Sciences*, 15(14):8103, January 2025. Publisher: Multidisciplinary Digital Publishing Institute.

[133] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

[134] Julio Cesar Rangel Reyes, Tarcisio Mendes de Farias, Ana Claudia Sima, and Norio Kobayashi. SPARQL Generation: an analysis on fine-tuning OpenLLaMA for Question Answering over a Life Science Knowledge Graph. *ArXiv*, abs/2402.04627, 2024.

[135] Tiago Rodrigues and Carla Teixeira Lopes. Harnessing Large Language Models for Clinical Information Extraction: A Systematic Literature Review. *ACM Trans. Comput. Healthcare*, 2025.

[136] Md Rashad Al Hasan Rony, Uttam Kumar, Roman Teucher, Liubov Kovriguina, and Jens Lehmann. SGPT: A Generative Approach for SPARQL Query Generation From Natural Language Questions. *IEEE Access*, 10:70712–70723, 2022.

[137] Malte Sander, Ulli Waltinger, Mikhail Roshchin, and Thomas Runkler. Ontology-based translation of natural language queries to SPARQL. In *2014 AAAI fall symposium series*, 2014.

[138] Krithikha Sanju Saravanan and Velammal Bhagavathiappan. Innovative agricultural ontology construction using NLP methodologies and graph neural

network. *Engineering Science and Technology, an International Journal*, 52:101675, 2024. Publisher: Elsevier.

[139] Andrew Sellergren, Sahar Kazemzadeh, Tiam Jaroensri, Atilla Kiraly, Madeleine Traverse, Timo Kohlberger, Shawn Xu, Fayaz Jamil, Cían Hughes, Charles Lau, Justin Chen, Fereshteh Mahvar, Liron Yatziv, Tiffany Chen, Bram Sterling, Stefanie Anna Baby, Susanna Maria Baby, Jeremy Lai, Samuel Schmidgall, Lu Yang, Kejia Chen, Per Bjornsson, Shashir Reddy, Ryan Brush, Kenneth Philbrick, Mercy Asiedu, Ines Mezerreg, Howard Hu, Howard Yang, Richa Tiwari, Sunny Jansen, Preeti Singh, Yun Liu, Shekoofeh Azizi, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Riviere, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean-bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Elena Buchatskaya, Jean-Baptiste Alayrac, Dmitry Lepikhin, Vlad Feinberg, Sebastian Borgeaud, Alek Andreev, Cassidy Hardin, Robert Dadashi, Léonard Hussenot, Armand Joulin, Olivier Bachem, Yossi Matias, Katherine Chou, Avinatan Hassidim, Kavi Goel, Clement Farabet, Joelle Barral, Tris Warkentin, Jonathon Shlens, David Fleet, Victor Cotruta, Omar Sanseviero, Gus Martins, Phoebe Kirk, Anand Rao, Shravya Shetty, David F. Steiner, Can Kirmizibayrak, Rory Pilgrim, Daniel Golden, and Lin Yang. MedGemma Technical Report, July 2025.

[140] Seungmin Seo, Byungkook Oh, Eunju Jo, Sanghak Lee, Dongho Lee, Kyong-Ho Lee, Donghoon Shin, and Yeonsoo Lee. Active Learning for Knowledge Graph Schema Expansion. *IEEE Transactions on Knowledge and Data Engineering*, 34(12):5610–5620, 2022.

[141] Mili Shah, Joyce Cahoon, Mirco Milletari, Jing Tian, Fotis Psallidas, Andreas Mueller, and Nick Litombe. Improving LLM-based KGQA for multi-hop Question Answering with implicit reasoning in few-shot examples. In Russa Biswas, Lucie-Aimée Kaffee, Oshin Agarwal, Pasquale Minervini, Sameer Singh, and Gerard de Melo, editors, *Proceedings of the 1st Workshop on Knowledge Graphs and Large Language Models (KaLLM 2024)*, pages 125–135, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[142] Ketan Rajshekhar Shahapure and Charles Nicholas. Cluster quality analysis using silhouette score. In *2020 IEEE 7th international conference on data science and advanced analytics (DSAA)*, pages 747–748. IEEE, 2020.

[143] Mehrnoush Shamsfard and Ahmad Abdollahzadeh Barforoush. Learning ontologies from natural language texts. *International journal of human-computer studies*, 60(1):17–63, 2004. Publisher: Elsevier.

[144] Guxue Shan, Xiaonan Chen, Chen Wang, Li Liu, Yuanjing Gu, Huiping Jiang, and Tingqi Shi. Comparing Diagnostic Accuracy of Clinical Professionals

and Large Language Models: Systematic Review and Meta-Analysis. *JMIR Medical Informatics*, 13(1):e64963, April 2025.

[145] Richard Shan and Tony Shan. Retrieval-Augmented Generation Architecture Framework: Harnessing the Power of RAG. In Ruifeng Xu, Huan Chen, Yirui Wu, and Liang-Jie Zhang, editors, *Cognitive Computing - ICCC 2024*, pages 88–104, Cham, 2025. Springer Nature Switzerland.

[146] Yong Shang, Yu Tian, Kewei Lyu, Tianshu Zhou, Ping Zhang, Jianghua Chen, and Jingsong Li. Electronic Health Record–Oriented Knowledge Graph System for Collaborative Clinical Decision Support Using Multicenter Fragmented Medical Data: Design and Application Study. *Journal of Medical Internet Research*, 26(1):e54263, July 2024.

[147] Yong Shang, Yu Tian, Min Zhou, Tianshu Zhou, Kewei Lyu, Zhixiao Wang, Ran Xin, Tingbo Liang, Shiqiang Zhu, and Jingsong Li. EHR-Oriented Knowledge Graph System: Toward Efficient Utilization of Non-Used Information Buried in Routine Clinical Practice. *IEEE journal of biomedical and health informatics*, 25(7):2463–2475, July 2021.

[148] C. E. Shannon. Prediction and entropy of printed English. *The Bell System Technical Journal*, 30(1):50–64, January 1951.

[149] Liang Shi, Zhengju Tang, Nan Zhang, Xiaotong Zhang, and Zhi Yang. A Survey on Employing Large Language Models for Text-to-SQL Tasks. *ACM Comput. Surv.*, 58(2):54:1–54:37, 2025.

[150] Seiji Shimizu, Tomohiro Nishiyama, Hiroyuki Nagai, Shoko Wakamiya, and Eiji Aramaki. Toward Cross-Hospital Deployment of Natural Language Processing Systems: Model Development and Validation of Fine-Tuned Large Language Models for Disease Name Recognition in Japanese. *JMIR Medical Informatics*, 13(1):e76773, July 2025.

[151] Amit Singhal. Introducing the Knowledge Graph: things, not strings, May 2012.

[152] Tommaso Soru, Edgard Marx, André Valdestilhas, Diego Esteves, Diego Moussallem, and Gustavo Publio. Neural machine translation for query construction and composition. *arXiv preprint arXiv:1806.10478*, 2018.

[153] Weiyi Sun, Anna Rumshisky, and Ozlem Uzuner. Evaluating temporal relations in clinical text: 2012 i2b2 Challenge. *Journal of the American Medical Informatics Association : JAMIA*, 20(5):806–813, September 2013.

[154] Supriyono, Aji Prasetya Wibawa, Suyono, and Fachrul Kurniawan. Advancements in natural language processing: Implications, challenges, and future directions. *Telematics and Informatics Reports*, 16:100173, December 2024.

[155] Milena Trajanoska, Riste Stojanov, and Dimitar Trajanov. Enhancing knowledge graph construction using large language models. *arXiv preprint arXiv:2305.04676*, 2023.

[156] Hieu Tran, Long Phan, James Anibal, Binh T. Nguyen, and Truong-Son Nguyen. SPBERT: an Efficient Pre-training BERT on SPARQL Queries for Question Answering over Knowledge Graphs. In Teddy Mantoro, Minho Lee, Media Anugerah Ayu, Kok Wai Wong, and Achmad Nizar Hidayanto, editors, *Neural Information Processing*, pages 512–523, Cham, 2021. Springer International Publishing.

[157] Priyansh Trivedi, Gaurav Maheshwari, Mohnish Dubey, and Jens Lehmann. LC-QuAD: A Corpus for Complex Question Answering over Knowledge Graphs. In Claudia d'Amato, Miriam Fernandez, Valentina Tamma, Freddy Lecue, Philippe Cudré-Mauroux, Juan Sequeda, Christoph Lange, and Jeff Heflin, editors, *The Semantic Web – ISWC 2017*, pages 210–218, Cham, 2017. Springer International Publishing.

[158] Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro von Werra, Clémentine Fourrier, Nathan Habib, and others. Zephyr: Direct distillation of lm alignment. *arXiv preprint arXiv:2310.16944*, 2023.

[159] Ricardo Usbeck, Xi Yan, Aleksandr Perevalov, Longquan Jiang, Julius Schulz, Angelie Kraft, Cedric Möller, Junbo Huang, Jan Reineke, Axel-Cyrille Ngonga Ngomo, Muhammad Saleem, and Andreas Both. QALD-10 – The 10th challenge on question answering over linked data: Shifting from DBpedia to Wikidata as a KG for KGQA. *Semantic Web*, 15(6):2193–2207, 2024. _eprint: https://journals.sagepub.com/doi/pdf/10.3233/SW-233471.

[160] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[161] Dinithi Vithanage, Chao Deng, Lei Wang, Mengyang Yin, Mohammad Alkhalaf, Zhenyu Zhang, Yunshu Zhu, and Ping Yu. Adapting Generative Large Language Models for Information Extraction from Unstructured Electronic Health Records in Residential Aged Care: A Comparative Analysis of Training Approaches. *Journal of Healthcare Informatics Research*, 9(2):191–219, June 2025.

[162] Zhen Wan, Fei Cheng, Zhuoyuan Mao, Qianying Liu, Haiyue Song, Jiwei Li, and Sadao Kurohashi. GPT-RE: In-context Learning for Relation Extraction using Large Language Models. *ArXiv*, abs/2305.02105, 2023.

[163] Chenguang Wang, Xiao Liu, and Dawn Song. Language models are open knowledge graphs. *arXiv preprint arXiv:2010.11967*, 2020.

[164] Haifeng Wang, Hua Wu, Zhongjun He, Liang Huang, and Kenneth Ward Church. Progress in Machine Translation. *Engineering*, 18:143–153, November 2022.

[165] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

[166] Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. Finetuned Language Models Are Zero-Shot Learners, February 2022. arXiv:2109.01652 [cs].

[167] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and others. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[168] Xiang Wei, Xingyu Cui, Ning Cheng, Xiaobin Wang, Xin Zhang, Shen Huang, Pengjun Xie, Jinan Xu, Yufeng Chen, Meishan Zhang, Yong Jiang, and Wenjuan Han. Zero-Shot Information Extraction via Chatting with ChatGPT. *ArXiv*, abs/2302.10205, 2023.

[169] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics.

[170] Derong Xu, Wei Chen, Wenjun Peng, Chao Zhang, Tong Xu, Xiangyu Zhao, Xian Wu, Yefeng Zheng, Yang Wang, and Enhong Chen. Large language models for generative information extraction: a survey. *Frontiers of Computer Science*, 18(6):186357, November 2024.

[171] Zhichao Yang, Sanjit Singh Batra, Joel Stremmel, and Eran Halperin. Surpassing GPT-4 Medical Coding with a Two-Stage Approach, 2023.

[172] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of Thoughts: Deliberate Problem Solving with Large Language Models, 2023. _eprint: 2305.10601.

[173] Xiaoyu Yin, Dagmar Gromann, and Sebastian Rudolph. Neural machine translating from natural language to SPARQL. *Future Generation Computer Systems*, 117:510–519, 2021.

[174] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium, 2018. Association for Computational Linguistics.

[175] Hamid Zafar, Giulio Napolitano, and Jens Lehmann. Formal query generation for question answering over knowledge bases. In *European semantic web conference*, pages 714–728. Springer, 2018.

[176] Bohui Zhang, Valentina Anita Carriero, Katrin Schreiberhuber, Stefani Tsaneva, Lucía Sánchez González, Jongmo Kim, and Jacopo de Berardinis. OntoChat: a Framework for Conversational Ontology Engineering using Language Models. *arXiv preprint arXiv:2403.05921*, 2024.

[177] Haopeng Zhang, Philip S. Yu, and Jiawei Zhang. A Systematic Survey of Text Summarization: From Statistical Methods to Large Language Models. *ACM Comput. Surv.*, 57(11):277:1–277:41, 2025.

[178] Zikang Zhang, Wangjie You, Tianci Wu, Xinrui Wang, Juntao Li, and Min Zhang. A Survey of Generative Information Extraction. In Owen Rambow, Leo Wanner, Marianna Apidianaki, Hend Al-Khalifa, Barbara Di Eugenio, and Steven Schockaert, editors, *Proceedings of the 31st International Conference on Computational Linguistics*, pages 4840–4870, Abu Dhabi, UAE, January 2025. Association for Computational Linguistics.

[179] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, and Zican Dong. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 1(2), 2023.

[180] Zhengyun Zhao, Qiao Jin, Fangyuan Chen, Tuorui Peng, and Sheng Yu. A large-scale dataset of patient summaries for retrieval-based clinical decision support systems. *Scientific Data*, 10(1):909, December 2023.

[181] Lingfeng Zhong, Jia Wu, Qian Li, Hao Peng, and Xindong Wu. A Comprehensive Survey on Automatic Knowledge Graph Construction. *ACM Computing Surveys*, 56(4):1–62, April 2024.

[182] Shuang Zhou, Zidu Xu, Mian Zhang, Chunpu Xu, Yawen Guo, Zaifu Zhan, Yi Fang, Sirui Ding, Jiashuo Wang, Kaishuai Xu, Liqiao Xia, Jeremy Yeung, Daochen Zha, Dongming Cai, Genevieve B. Melton, Mingquan Lin, and Rui

Zhang. Large language models for disease diagnosis: a scoping review. *npj Artificial Intelligence*, 1(1):9, June 2025.

[183] Wenxuan Zhou, Sheng Zhang, Yu Gu, Muhao Chen, and Hoifung Poon. UniversalNER: Targeted Distillation from Large Language Models for Open Named Entity Recognition. *ArXiv*, abs/2308.03279, 2023.