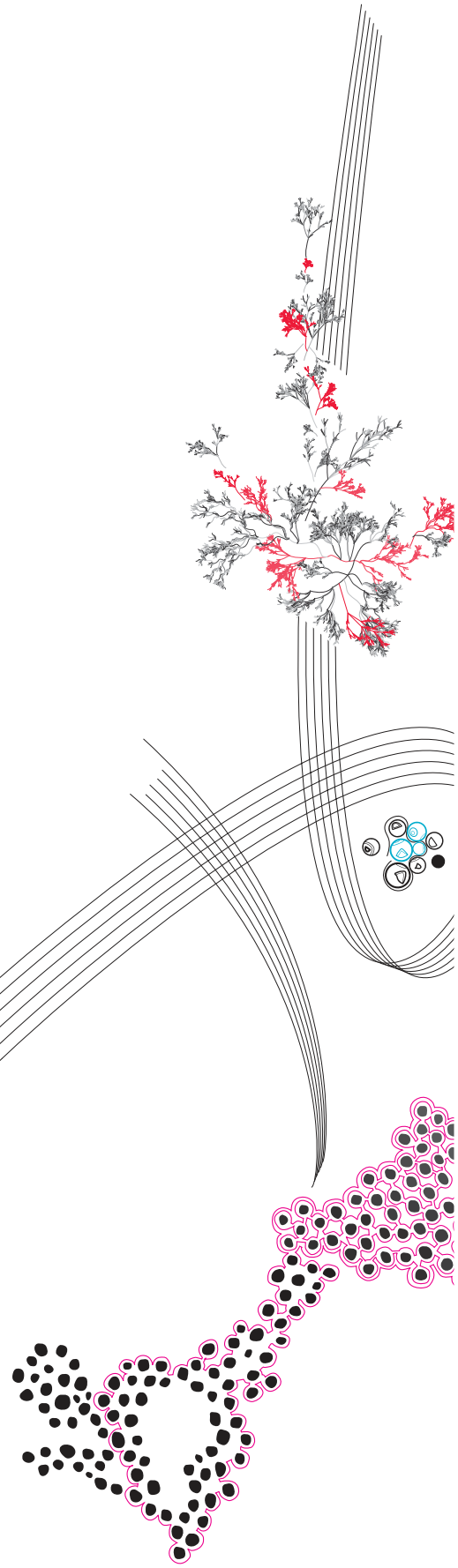

MSc Business Information Technology
Master Thesis

KNOCK: Knowledge-driven Ontology and Chain-of-Thought-based Knowledge Graph Question Answering

Bolin Huang

Supervisor:
dr. A. Abhishta
dr. L.O. Bonino da Silva Santos
A. Kolk BSc.

March, 2025

Business Information Technology
Faculty of Electrical Engineering,
Mathematics and Computer Science,
University of Twente

UNIVERSITY OF TWENTE.

Acknowledgments

This thesis marks the end of my master's study in Business Information Technology. The journey began more than two years ago when I arrived at the University of Twente in a transition from a business student to a practitioner focusing more on tech. My interest in information technology started to sprout during my bachelor years in business studies when data-driven approaches emerged to make such an impact on all businesses across different industries. Witnessing how IT had reshaped the redefined the business process inspired me to dive into the study and research of this domain. It is not an easy journey from the beginning, as I face numerous new concepts and knowledge frameworks. It is also not easy to come to a new culture and live, change and adapt to the new environment. Despite all these difficulties, embarking on an audit of this journey, I am so proud of how far I have come.

I would like to express my deep gratitude to my academic supervisors at the University of Twente, Abhishta and Luiz. They have always accompanied me during my thesis and provided timely guidance. They give tips on thinking and solving problems from different perspectives, which often benefit me. I have learned a lot from them for being academically rigorous and conducting a project systematically. I would also like to express my deep thanks to my supervisor at Kadaster, Anjo. He has always inspired me with his expertise during the project and offered me extra motivation. His patience and guidance have helped me tremendously. Kudos to Lexi, who taught me a lot about linked data with her practical experience. I also want to thank Janneke Michielsen and Erwin Folmer for offering me the opportunity to complete my thesis internship at the Data Science Team of Kadaster.

The unwavering support of my friends has been my driving force. I sincerely thank everyone for their integral support to me throughout this journey. I would like to extend my heartfelt thanks to Jialiang Liang and Fengpeng Huang for always being there to share both my delight and sorrow. Their companionship has been invaluable in helping me navigate through the toughest times. I am also grateful to Akash Ramakrishnan and Amal Reji, who made my life in Enschede memorable. I cherish all the moments with them in our cooking sessions, trips and parties. The happy times with them have been my source of joy and solace.

The following chapters present my research on this topic, and I sincerely hope you find reading it enjoyable and insightful. I also hope that, in some small way, it contributes meaningfully to the domain.

Bolin Huang

Enschede, March 2025

Abstract

Knowledge graph question answering(KGQA) has garnered significant research interests and has evolved quickly in both academia and industry. It combines the strengths of artificial intelligence and linked data to connect humans with heterogeneous knowledge-intensive data sources in an interactive way. However, the current approaches of KGQA are mostly difficult to adapt to knowledge graph with different schemas universally. A lot of approaches also rely on substantial training to build the question answering model, which is difficult to maintain as the data sources iteratively update. In this thesis, we propose KNOCK, which stands for **KN**owledge-driven **O**ntology and **C**hain-of-Thought-based **K**GQA, an approach utilizing the synergy strengths of ontology embedding and Chain-of-Thought(CoT) prompting to construct a Retrieval-augmented Generation(RAG) system to enhance the performance of question answering on knowledge graphs. Via a bibliometric survey, we define our methodological framework and implement our prototype modeling in a CRISP-DM process. We ground our research in the question answering system of Kadaster Knowledge Graph(KKG), a knowledge database of the Dutch national cadastral and mapping agency. By analyzing the current question answering system of KKG, we identify its limitations and implement our approach on KKG to verify its feasibility. By conducting this research project, we aim to validate our approach and answer how we can establish a KGQA system more efficiently with less training cost and ensure the system is more resilient to knowledge source update. This research also makes contributions to the KGQA application in the domain of cadastral data management and geographical information system(GIS).

Keywords: Knowledge Graph Question Answering, Ontology Embedding, Chain-of-Thought Prompting, Retrieval-augmented Generation, Cadastre

Contents

List of Figures

List of Tables

List of Abbreviations

KG	Knowledge Graph
KB	Knowledge Base
KGQA	Knowledge Graph Question Answering
KBQA	Knowledge Base Question Answering
QA	Question Answering
NLP	Natural Language Processing
KKG	Kadaster Knowledge Graph
LLM	Large Language Model
CoT	Chain-of-Thought
RAG	Retrieval-augmented Generation
CRISP-DM	Cross-Industry Standard Process for Data Mining
W3C	World Wide Web Consortium
OWL	Web Ontology Language
RDF	Resource Description Framework
SHACL	Shapes Constraint Language
XML	Extensible Markup Language
JVM	Java Virtual Machine
PDCA	Plan-Do-Check-Act

Chapter 1

Introduction

A Knowledge Graph(KG) is a graph-structured database using a knowledge representation model to represent a collection of facts [1]. It can be interfaced to perform downstream tasks such as asking questions about the facts and inferring new knowledge from the existing facts [2]. The knowledge retrieval from a KG for question answering is termed Knowledge Graph Question Answering(KGQA). While answering simple questions on KGs has been extensively studied and has achieved significant success in practice, the performance of KGQA over complicated questions, such as those requiring multi-hop reasoning, is still challenging when grounded in real-world KGs [3], despite the rapid advances of the language models. Lots of prior research has focused on i.i.d. generalization, for example using the distribution of questions in the training set that is identical to the distribution of questions in the test set. but in practical KGQA, the limitations of i.i.d distribution are conspicuous. Gu et al. have pointed out that for a large-scale knowledge base, it is difficult to train the realistic KGQA system only with i.i.d. generalization, because accurately capturing the true user distribution in the knowledge graph is challenging, and randomly sampling training examples from the vast data space would result in significant data inefficiency. [4]. The challenge of training a KGQA system is also highlighted by the fact that the distribution shift between the training set and the test set is ubiquitous [5], especially when the data heterogeneity of real-world KGs is considered.

KGs often have complex schemas, including entities, relations, and their interconnections. These schemas define the data structure of the KGs. Successful KGQA must perform robust entity linking to accurately identify the correct entities within the data model based on the given question. Additionally, precise relation extraction is essential to discern the relationships between these entities. Achieving this requires advanced semantic parsing and query generation capabilities to transform natural language questions into machine-readable logical forms [6], like structured queries including join, filter, aggregate and other operations, which are non-trivial over a knowledge graph. To generate satisfying answers to the questions, these entities and relations need to be organized into logical reasoning in the form of natural language. A common problem of the structure of the knowledge graph is that they are often incomplete, missing some entities and relations [7] [8]. This usually adds complications to relation extraction in the KGQA tasks. It is also noted that the questions can sometimes be ambiguous, containing varied implications in different contexts [9] [10]. For example, in the question "What is LLM?", LLM can refer to either *Large Language Model*, or *Master of Laws*, depending on the context of the question. The

robust KGQA systems are supposed to be resilient to these ambiguities.

1.1 Research Background

Kadaster is the Dutch national organization in charge of land registry, national mapping and other public cadastral affairs in the Netherlands. It keeps track of data on buildings, addresses, plots of land, roads and underground networks of cables and pipes, as well as other geographical information for the Netherlands. Besides, it also registers data about moveable properties such as ships, aircrafts and the rights and rights holders thereof [11]. Moreover, Kadaster acts as an advisory body for land-use issues and national spatial data infrastructures [12].

Kadaster has been employing linked data and semantic web technologies in practice and conducting state-of-the-art research in these domains for years. It has built its own knowledge graphs to represent open data sources in practice [13]. As a government administration, Kadaster is responsible for providing open cadastral data sources including BAG(Dutch: Basisregistratie Adressen en Gebouwen, English: Base Registration of Addresses and Buildings. BAG consists of all addresses and buildings and their properties in the Netherlands, such as year of construction, surface area, energy labels, purpose of use and location on the map) [11] [14], BRK(Dutch: Basisregistratie Kadaster, English: Base Registration of Land Registry. BRK consists the cadastral registration of immovable property and rights in rem and the cadastral map showing the location of the cadastral parcels (including parcel number) and the boundaries of the state, provinces and municipalities) [15], BRT(Dutch: Basisregistratie Topografie, English: Base Registration of Topography. BRT contains digital topographical files at different scales, in which both the drawn up maps and the object-oriented files are available as open data) [16] [17] and BGT(Dutch: Basisregistratie Grootschalige Topografie, English: Base Registration of Large-Scale Topography. BGT is a standardized uniform detailed digital map of the whole Netherlands with objects such as buildings, roads, water, railway lines and greenery recorded in an unambiguous manner) [18]to the general public [19]. These data are regularly accessed by municipalities, provinces, and other government institutions [20]. In addition, public stakeholders, including real estate brokers, accountants, and ordinary people, can also benefit from the data for different purposes. This implies that any user, who has questions about their house, their environment, or the geoinformation of the Netherlands as a whole, can benefit from this data portal.

However, it is unrealistic to expect every data user to wield professional expertise in knowledge graph, linked data and API. Thus ease of use of these open data must be achieved. One solution to streamline the utilization of the data is to use the chatbot. A chatbot is an intelligent conversational computer system that simulates human chat in order to provide online assistance and guidance automatically [21]. It enables user to ask questions in natural language to obtain answers in the same format, containing information from the pertinent data sources. It usually has a user-friendly interface in order to make users feel relaxed and interactive. The chatbot has been employed for some time in customer service and support. It is suitable for users to retrieve information in an easier manner. It is especially helpful when the information is stored in a strictly structured model, such as a knowledge graph, which usually requires users to perform information extraction with sufficient linked data expertise if the traditional query methods are used. With a properly integrated chatbot, users can circumvent the traditional methods and

still obtain target information from these structured graph databases without necessarily being experts in semantic web technologies.

Therefore, it is inspired for Kadaster to have developed its own chatbot integrated with the Kadaster Knowledge Graph(KKG). The chatbot is named Loki(Acronym for Location-based Kadaster Information, Dutch: Locatie-gebaseerde Kadaster Informatieverstrekking), and it is currently deployed on the website of Kadaster Labs for open access. As the front-end web interface. Loki connects users with an underpinning question answering system, and this system built on KKG is the key underlying factor that makes QA on KKG successful. The question answering system of Loki is deployed on a T5 language model [22]. When the user enters a question, the text serves as input to the question answering system. Then the pre-trained question answering system coverts the text into a machine-readable logical form, in this case, a SPARQL query, and use the SPARQL query to query related information from KKG, The queried structured information is eventually transformed back to text as the output answer to the user. However, the current question answering system still faces significant limitations and performance bottlenecks. We will further discuss these challenges in the following sections.

1.2 Problem Statement

In this section, the current challenges and limitations faced by the existing question answering system of Loki are outlined. The section serves to identify the gaps in the systems ability to handle complex and novel queries, and proposes potential solutions to enhance its performance. It consists of two subsection. The subsection Status Quo provides an overview of the systems current capabilities and challenges, while regarding these challenges, the subsection Research Question formulates the core research question and associated subquestions that guide this study. Together, these subsections establish the foundation for the research objectives and methodologies discussed in subsequent chapters.

1.2.1 Status Quo

The current question answering system is validated to perform well in i.i.d generalization for the pre-defined questions. The pre-defined questions are part of the question-SPARQL pairs in the training data of the question answering system. These question-SPARQL pairs are manually constructed for the chatbot, a process that is highly labor-intensive and necessitates the involvement of linked data experts. Only questions that share the same schemas with the question-SAPRQL pairs in the training data can be answered by the current question answering system. Consequently, the system is yet to be capable of answering questions that are excluded from the question-SAPRQL pairs in the training data. In other words, it performs insufficiently when facing questions with novel compositions of schemas or unseen schemas, providing very limited flexibility. In addition, the compositions of the vast entities and relations of KKG are enormous, leading to a very large number of possible cadastral questions based on the linked data of KKG, while the manually designed question-SPARQL pairs can only include a tiny fraction of these possible questions. Therefore, the utilization of the graph data in the current question answering system is low. To enhance the systems robustness and flexibility in handling a broader range of queries related to cadastral data in the KKG, it is essential to enable automated inference of the data schema, otherwise the system will have to rely on more manually crafted question-SPARQL pairs, which is labor-intensive and thus impractical because given the scale of a KKG-sized knowledge graph, it is infeasible to manually

enumerate all possible questions and generate corresponding SPARQL queries for each, highlighting the necessity of automated schema inference.

An intuitive solution is to feed all the semantic elements of KKG as the training data to fine-tune a language model so that it can learn all the schemas and their combinations in the knowledge graph. The fine-tuned language model can therefore serve as the question answering system. The limitations of this solution are evident. KKG is a dynamic knowledge graph in a frequent update cycle. New cadastral data is coming in every day to update the knowledge graph instances. The one-time fine-tuning is unable to handle the following updated data. To ensure the question answering system is up-to-date and can give correct answers, we will have to keep fine-tuning the model every day and maintain version control of the implementation. This is not the priority of Kadaster and it incurs a considerable amount of extra cost as well. Another intuitive solution is to use basic prompt engineering as it is resilient against data updates without iterated training hassles. More specifically, it is to instruct a language model with a prompt that entails all the knowledge graph entities and relations of KKG. It works towards a similar effect to fine-tuning and enables the LLM to perform inference on the knowledge graph elements with the prompt. However, there are billions of instances and sematic triples in KKG. Considering the number of triples in this one big knowledge graph and the token limitations of the prompt in most LLMs, it is impossible and infeasible to feed the complete instance data and their exhaustive pertinent questions in the prompt.

As for the language model, although T5 is a decent language model, Kadaster expects to use a newer and more powerful LLM to improve the question answering system. Current prevailing LLMs, such as GPT [23], LLaMA [24] and Gemini [25] have displayed excellent capabilities in sequence-to-sequence text generation and question answering. Therefore, we should also look into implementing a stronger LLM instead of T5 on the task.

1.2.2 Motivation

Embarking on an audit of these limitations and challenges, it is motivated to develop a system to accommodate the robust inference on the knowledge graph under the circumstance that the graph data is dynamically updated. Prompt engineering can thus still be a good strategy. Due to prompt token limitations and the vast graph size, the entire knowledge graph cannot be included in the prompt. Therefore, it is necessary to construct a compact subgraph that retains the essential semantic and structural properties of the KKG, but with a compressed size. The problem is how we can use a knowledge-intensive and still compact-in-size subgraph that functions as an underlying layer of the full knowledge graph, to improve the inference of the LLM on the KKG linked data, through which we enable the system to better accommodate the question answering task, even for novel question schemas.

For a right subgraph extraction that creates the representative subgraph of a large knowledge graph, it is imperative to distinguish the key elements of the knowledge graph. Gutiérrez-Basulto and Schockaert have highlighted that ontology serves as rules to define the dependencies among difference relations in the knowledge graph, and the embedding of ontology with geometric model is more compatible than some prevailing KG embedding approaches [26]. Blagec et al. [27] have showed that in an ontology-based knowledge graph, the ontology model is an underlying component and ontology design plays an important role in the construction of knowledge graph. [28] has explicitly clarify how ontology works

as a foundation layer of knowledge graph. And in [29], Chah explored Google Knowledge Graph from an interpretation of its underlying ontology, which underpins the semantic search feature of Google Search Engine. These research findings indicate that the ontology is the underlying layer and schema of a knowledge graph and captures the key connection of the knowledge representation model. Meanwhile, when the new cadastral data comes in, it usually updates the instance data but the ontology data remains static. So it is immune from data dynamics. Hence in this research project, we can extract the ontology of KKG to construct the representative subgraph. It is noted that in the context of KKG, the ontology refers to the concepts coming from potentially different domain ontologies, vocabularies and taxonomies and how these concepts are interrelated. These domain ontologies represent and classify the instance data of KKG. The concepts may also contain a small amount of necessary name individual data. Considering the enormous size of instance data in KKG, but the very lightweight size of the ontology, which is still highly informative, this remains as a pragmatic solution for subgraph extraction.

After the subgraph extraction, the system needs to match the question content to the subgraph elements, namely ontology elements in this context, when the chatbot user poses a question in the system. This is the core and also the bottleneck of Knowledge Graph Question Answering. The automatic matching and natural language processing of this step should be executed in the LLM. However, considering artificial intelligence cannot inherently comprehend natural language and the semantic meaning of ontological axioms in the same way humans do, the questions and ontology must be vectorized into mathematical representations that are machine-readable, which are also referred to as embeddings in the context of machine learning. Afterwards, it is possible to compare the similarity of the word embeddings of the questions and the ontology embedding in a mathematical sense for a good matching performance. Therefore, we need to investigate and identify the most suitable embedding techniques for the ontology and the associated questions.

With the suitable embedding model, the ontology embedding and word embedding can be accomplished and matched correctly. In this correct matching case, the next step to look for the pertinent instance data in the knowledge graph can begin. We are supposed to use a SPARQL query here to query the instance data from the graph. When a question is entered by the user, our embedding model can identify the most relevant ontology elements the question refers to. In order to generate the SPARQL query correctly, we need to construct a proper prompt with this embedding matching result. The pre-defined prompt should guide the LLM to formulate the SPARQL query based on the ontology elements identified by the embedding matching and the instance data involved in the question. The ontology can function as an external retrievable data source to facilitate the SPARQL generation within the LLM, which mirrors the operational principles of Retrival-augmented Generation(RAG) [30]. In order to design and refine the prompts to effectively guide the query generation, it is essential that we refer to the cutting-edged prompt engineering techniques to design the most suitable prompt for our model. We propose using Chain-of-Thought Prompting [31]. Research should be done to review how well these prompt engineering approaches can serve the task. If the model can stably produce accurate SPARQL queries for the given question, it enables the following step in which the LLM formulates the correct output answer to the question based on the query results.

1.2.3 Research Questions

Accordingly, we can formulate the core research question of this research based on the unsolved obstacles we are facing at the moment:

Given the current challenges in Kadaster Knowledge Graph(KKG), how can we utilize ontology embedding and CoT prompting to build a resilient RAG system to solve a KGQA problem?

We have also defined our subquestions based on the insights we learned from our preliminary research work to serve the core research question:

1. How can the ontology elements of KKG be accurately aligned with relevant natural language questions using ontology embeddings?

2. How can we design effective RAG and prompts with an LLM to generate accurate SPARQL queries against the knowledge graph?

3. How can the SPARQL queries generated in the previous step, along with the corresponding questions, be utilized to improve the KGQA system?

To solve these questions, we will first conduct a bibliometric survey to systematically study the literature of the prior research. For this data science project, we are going to leverage the CRISP-DM framework as the implementation methodology. Then we will design the experimental strategy in the CRISP-DM cycle for our KNOCK approach based on what we learn from the prior research and the problem context at Kadaster. We will develop the prototype of a new question answering system with ontology embedding and CoT prompting, following our designed methodology and experimental framework. After the establishment of the prototype, we conduct the experiment by testing KKG-related questions. We will implement a comprehensive evaluation to assess the results of the experiment and provide validation to our proposed approach.

This thesis is organized in the following structure: Chapter 2 specifies the related work in the prior research. Chapter 3 articulates the methodology we implement for this research and formulates the experimental set-up of the prototyping process. Chapter 4 presents and discusses the experimental results and the implications we discover from the research. Chapter 5 provides a conclusive review of the research project, summarizing key findings to address the research questions and proposing the potential directions for future research.

Chapter 2

Literature Review

Our research is grounded on the question answering system on a structured knowledge database, namely the Kadaster Knowledge Graph(KKG). There are often synonymous terms used interchangeably by scholars to refer to this research domain, such as Knowledge Base Question Answering [1] [3] [6] [32] [5] [33], Knowledge Graph Question Answering [34] [35] [36] [8] [37] [38], as well as their acronyms KBQA and KGQA. These terms should be considered as criteria for the preliminary selection of related works. In addition, a key component of our research is exploring how to perform accurate transformation from natural language questions to a logical form(SPARQL query) in the question answering system. This marks the prior research in the domain of semantic parsing as a significant reference for our research.

Since the information needed for question answering is structured and stored in KKG, there are organized interlinked entities to represent the ontologies of the information model. However, it is noted that KKG contains an enormous size of linked data with billions of triples. It is thus impossible to load the full knowledge graph with all instances into the prompt of an LLM to generate the SPARQL queries. Our alternative is to construct a compact-size subgraph from KKG and use the ontologies of KKG as the subgraph components. The subgraph should consist of sufficient higher-level ontology information like classes, properties, et cetera of KKG. The subgraph should be well curated to make sure its ontology size is appropriate. On one hand, if it is too large, it will incur more noise and a higher computational cost in inference temporally and financially. On the other hand, if its size is too small, it doesn't contain sufficient information to represent KKG. After constructing the compact subgraph with representative ontologies, we need to pair the subgraph with the natural language questions set to instruct the LLM to generate valid SPARQL queries against KKG. The subgraph is comprised of selected KKG ontologies while the question set consists of natural language text. These two representations can not be compared and matched directly. They need to be converted into mathematical representations in lower dimensional space to analyze the semantic similarity. This step requires transforming the ontology into feature vectors like ontology embeddings, and storing the vectors in a vector database. While a question is answered in the LLM, the question is also transformed into feature vectors like word embeddings, and compared with ontology vectors stored in the vector database to find out what specific ontology the question is referring to. Therefore, researching how to transform ontology into vectors in a machine-understandable and reusable manner and store them in a vector space is key to the

following ontology match, thus we place Ontology Embedding into the related works. Once the relevant ontology components are matched, this information is given to the prompt for the generation of SPARQL query. Hence, prompt engineering techniques are needed in this knowledge-intensive task. As we propose using CoT prompting, prior research on the domain of CoT prompting will be examined. We also use Retrieval-augmented Generation(RAG) [30] here, with the ontology vector database serving as an external data source to guide the LLM with additional knowledge. Accordingly, Retrieval-augmented Generation serves as an important part of the related works as well.

2.1 Literature Review Methodology

Bibliometric survey is an important statistical analysis to the existing literature of a certain batch of research topics over time [39]. A bibliometric survey is implemented for the literature study of the related work. In this section, we provide a concise summary of the bibliometric survey. A complete review of the bibliometric survey can be found in the Appendix.

2.1.1 Motivation and Purpose of the Bibliometric Survey

Bibliometric survey offers perceptions into quantifiable aspects of scientific publications, like the number of an author's publications and citations. These indicators effectively help researchers with capturing scholarly output in their research domain [40]. Basically the purpose of a bibliometric survey consists of three part.

2.1.1.1 Analyzing Existing Literature

Citation analysis on currently existing publications is a commonly performed bibliometric survey method [41]. Bibliometric survey allows researchers to systematically analyze the body of scientific literature within a specific field. By examining patterns, trends, and relationships, researchers gain insights into the intellectual landscape. Key aspects of analysis include identifying influential authors, journals, and research topics. Researchers can also explore citation patterns to understand how knowledge flows across publications.

2.1.1.2 Identifying Trends

The purpose of a bibliometric survey extends beyond mere analysis of existing literature. It also helps identify emerging trends and shifts in research focus. By analyzing publication patterns over time, researchers can pinpoint areas of growth or decline. Trend identification aids policymakers, funding agencies, and scholars in making informed decisions. For instance, it can guide resource allocation or highlight areas needing further investigation.

2.1.1.3 Highlighting Key Contributions

Moreover, bibliometric survey allows us to recognize impactful research contributions. This includes identifying highly cited papers, influential authors, and groundbreaking studies. Highlighting key contributions fosters collaboration, encourages further research, and acknowledges the work of scholars who have significantly shaped their fields.

2.1.2 Selection Criteria

The literature selection and analysis are conducted with VOSviewer [42], a prevalent tool for bibliometric survey and systematic literature review. We further implement topic modeling to cluster publications in different topics via BERTopic [43]. With the results of topic modeling, we have formulated the following selection criteria for the literature selection. Each criterion has its representative literature documents. These criteria are demonstrated in Table 2.1.

Document Selection Criteria
The documents should be associated with Knowledge Graph Question Answering(KGQA) & Knowledge Base Question Answering(KBQA).
The documents should focus on knowledge or ontology embedding.
The document should investigate the application of LLM and prompt engineering techniques to enhance the accuracy of output in KGQA systems.

TABLE 2.1: Selection criteria

2.2 Related Work

The bibliometric survey facilitates a systematic review of relevant topics in existing research. By analyzing the literature in these areas, we derive insights that inform the development of our research framework and experimental methodology.

2.2.1 Knowledge Graph Question Answering

In both academic and industrial scenarios, KBs and KGs are widely implemented as a type of expert system with a structured data model to address complex knowledge-intensive problems via reasoning through the bodies of knowledge [44]. Both [45] and [46] have proposed novel embedding methods for entity linking(EL) in knowledge base and knowledge graph. These general methods are not only tailored for question answering but also for other knowledge-intensive downstream tasks such as knowledge base population, content analysis and relation extraction. Luo et al. [47] have proposed integrating a relation-aware attention network with the BERT model to enhance the entity linking and relation detection processes in knowledge-intensive question answering, which is supposed to strengthen the association of the questions and the knowledge base facts. It has attained state-of-the-art accuracy when evaluated on the SimpleQuestion dataset. To tackle the disambiguation problem in entity linking, Zhu et al. [48], Zwicklbauer et al. [49] [50] have proposed embedding approaches to decrease ambiguity while mapping web text to the entities in knowledge graph. The research of Mai et al. [51] [52] highlights the combination of knowledge graph and geographic question answering in this cluster. They even go one step ahead to encode geographic coordinates into the knowledge graph and perform knowledge graph embedding afterwards to tackle questions related to distance and locations in the QA system, which we don't have in KKG yet. Pan et al. [53] have proposed a roadmap to facilitate the synergistic effect of KG and LLM, exploring the potentials of enhancing one with the other. For knowledge graph reasoning, Bi et al. [54] have proposed a neural network model for unrestricted multi-hop question answering to solve both one-to-many mapping

reasoning and many-to-one mapping reasoning issues. They later have also proposed a reward integration and policy evaluation to boost correct reasoning in KGQA [55]. Furthermore, for the multiple relations related to one identical entity, namely a one-to-many dilemma, Zhu et al. [56] have proposed a two-level hierarchical reinforcement learning approach to enhance multi-hop reasoning in KG. Similarly, Wang et al. [57] have proposed a deep reinforcement learning framework for knowledge graph reasoning as well. Wang et al. [58] have employed a knowledge graph prompting technique to handle multi-document question answering task, exploring the graph construction and graph traversal modules. Besides, Guo et al. [59] have introduced a framework to utilize ChatGPT and Llama2 in conjunction with knowledge graphs to facilitate precise and interpretable multi-hop reasoning in KGQA. Jiao et al. [60] have proposed a semantic matching approach to boost the effectiveness of the logical forms generated from the input questions. It is ensured that the logical form is coupled with the knowledge graph for valid query execution in question answering.

2.2.2 Subgraph Extraction and Ontology Embedding

There are substantial research outputs on advanced subgraph extraction from knowledge graphs. Yow et al. [61] have summarized and discussed the cutting-edged machine-learning-based subgraph extraction techniques. Aghaei et al. [62] have discussed three types of tailored subgraph extraction approaches , namely, filter-based approaches, heuristic-based approaches and neural-based approaches, and proposed a subgraph extraction technique based on Personal Page Rank algorithm for knowledge graph question answering system. Sun et al. [63] have proposed PullNet, a weakly supervised integrated framework that uses graph CNN to construct question-specific subgraph and retrieve answers on subgraphs for KGQA. Zhang et al. [64] have argued the limitations of heuristic- or reasoning-based subgraph extraction approaches and introduced a novel trainable subgraph retriever (SR) that operates independently of the subsequent reasoning process. This decoupling facilitates a modular framework that can be seamlessly integrated with any subgraph-oriented KBQA model. It significantly outperforms existing retrieval methods in both retrieval accuracy and question-answering performance. These approaches all utilized complex mathematical reasoning or machine learning techniques to curate a subgraph and answer the query based on this subgraph. They have respectively achieved outstanding performance on academic benchmarks. As for ontology embedding, Ristoski et al. [65] have proposed the graound-breaking Rdf2vec methodology to embed RDF graphs. And Portisch and Paulheim have advanced the framework of Rdf2vec by introducing novel random walk strategies to generate embeddings that emphasize either similarity or relatedness [66]. For OWL ontology embedding, Smaili et al. [67] have developed the Onto2vec approach for the ontology embedding in biological knowledge representation such as protein-protein interaction. Later they enhanced the method and introduced Opa2vec [68], which leverages the ontology metadata more effectively to better generate vector representations for various types of biomedical ontology. Based on the work of Onto2vec and Opa2vec, Zhapa-Camacho et al. [69] developed mOWL, a full python implementation for machine learning tasks with OWL ontology, including ontology embedding and zero-shot learning. In addtion, Chen et al. [70] have also introduced the method OWL2vec* tailored for OWL ontology embedding, which leverages the graph structure, lexical information and logical constructors of the ontology for embeddings. These approaches are well evaluated and prove to be inspiring for similar ontology embedding tasks on KKG. We can utilize and tailor these methods on the embedding of our ontology for effective knowledge extraction.

As for word embedding, there are already quite a lot of mature solutions. One prevailing example is word2vec [71], which is also leveraged in a lot of ontology embedding apporaches for semantic analysis and learning. Due to the limitation of bag-of-words model, Mikolov et al. further advanced word2vec to better word embedding in sentences and documents [72]. Other prevalent methods include BERT [73], and recently emerging in-context learning LLMs such as GPT [74] and Claude [75]. Different embedding models can be studied and evaluated to select the most suitable embedding model, or models combination, to perform the embedding task.

2.2.3 Chain-of-Thought Prompting

Wei et al. [31] have researched how CoT prompting can effectively boost the reasoning ability of LLMs. Gilbert et al. [76], regarding the token limitation issues in prompt engineering, have researched on the performance of using compressed text information to prompt an LLM. Nguyen et al. [77] have studied the potential of using large language models such as Llama, T5 or Mistras with CoT prompting for semantic parsing tasks, namely to transform natural language questions into SPARQL queries for semantic graph search. Ye and Durrett [78] have proposed an explanation selection method specifically designed for CoT prompting, aiming at enhancing the quality of the prompts to achieve superior question answering performance. Zhang et al. [79] have researched on leveraging CoT prompting and LLM-restructured small data to train relatively compact-sized "baby" language model to outperform the vanilla large language model like RoBERTa. Chen et al. [80] have researched and examined the different performance of prompts on ChatGPT-series models, with or without CoT prompting, on classification and reasoning tasks of biomedical applications, and compared the results with the ones generated by fine-tuning transformer models on the same applications. Zahera et al.[81] have proposed to utilize CoT prompting for the translation from natural language questions to SPARQL queries. They integrate CoT prompting with the LLM LlaMA2-code to form the framework CoT-SPARQL to generate SPARQL queries for natural language questions and achieve state-of-the-art results on QALD-10 and QALD-9 datasets.

2.2.4 Retrieval-augmented Generation

Muludi et al. [82] have utilized GPT-3.5 Turbo to construct and evaluate the performance of RAG to enhance QA and reduce hallucination via external knowledge base support. Ding et al. have proposed a [83] strategy to utilize RAG to solve the fine-tuning challenge of a large language model while facing dynamic data update and facilitate the reproduction of factual information. Roychowdhury et al. [84] have leveraged LLM and RAG for a QA system in the domain of telecom. Suess et al. [85] have proposed a RAG conversational question answering system for car-specific questions, implying that RAG is very useful in boosting the domain-specific question answering system. Ryu et al. [86] have also highlighted the difficulties inherent in domain-specific question answering, especially in the domain where hallucination and misinformation are highly risky, such as the legal domain. They have proposed a RAG technique to evaluate the LLM-generated texts associated with legal domain against legal documents. Urban and Binnig [87] have proposed an approach to enhance RAG strategy, in which they expand databases with LLM to transform queries into execution plans so that it can process multi-modal data in a scalable manner.

2.3 Implications of the Literature Review

This chapter has thoroughly reviewed the pertinent literature of prior research closely associated with our research question. Current approaches to KGQA often focus on the question answering over the prevalent open knowledge graphs, such as Wikidata [88] and DBpedia [89]. These open knowledge graphs are neatly structured and follow strict paradigms. In practice, knowledge graphs can have distinct schemas from these prestige open graphs and face data quality bottlenecks, often caused by missing graph elements. The current approaches often struggle with adaptability across knowledge graphs with varying schemas. Additionally, many methods depend on extensive training to construct the question-answering model, making maintenance challenging as data sources undergo iterative updates. There has been limited research tailored to solve this problem. Also, as the rapid advancement in LLMs and prompt engineering consistently gives impressive performance on question answering tasks, the potential of combining the strengths of ontology embedding and CoT prompting to construct an RAG system to address KGQA problem has nevertheless not been explored yet. We strive to bridge the gap in the domain by integrating the ontology-based inference approach with the advancements in CoT prompting and RAG to construct a KGQA application with a robust performance and economic training cost. The methodology and experimental set-up details of our approach will be presented in the following chapter.

Chapter 3

Methodology and Experimental Set-up

This chapter outlines the methodological approach used in this research, mainly the Cross-Industry Standard Process for Data Mining (CRISP-DM), and the experimental set-up process of implementing the methodological approach to build our QA system artifact.

3.1 Methodology

The CRISP-DM methodology provides a robust structure for addressing the research objectives while ensuring both data science rigor and practical relevance. It offers a systematic framework for designing and evaluating information systems artifacts. Integrating CRISP-DM into our project allows for iterative development, evaluation, and refinement of the proposed solution, aligning with the data-driven nature of the research problem and the need for practical applicability in Kadaster's problem senario of KGQA.

3.1.1 Cross-industry Standard Process for Data Mining

CRISP-DM is a widely adopted methodology that provides a structured approach to data mining and data science projects. Originally developed in the late 1990s by a consortium of leading data mining users and suppliers [90], CRISP-DM has become a de facto standard for organizing and executing data-driven projects across various industries [91]. Its popularity stems from its flexibility, robustness, and comprehensive nature, making it suitable for both academic research and practical applications. It consists of six phases: business understanding, data understanding, data preparation, modelling, evaluation and deployment. Its iterative nature also accommodate flexibility and continuous improvement of the project, aligning well with the often cyclical process of academic research and industrial development. In the context of our research project focusing on knowledge graph question answering, with both academic and industrial components, CRISP-DM can provide a valuable structure for organizing and executing the work.

1. **Business Understanding**
   This initial phase focuses on understanding the project objectives and requirements from a business perspective, for which we clearly define the research objectives and industrial applications of the knowledge graph question answering system. This can

include identifying specific use cases, determining evaluation metrics, and establishing project constraints.

2. **Data Understanding**
   In this phase it is supposed to collect and explore the data to gain insights into the data characteristics such as data structure and data quality. It is also important to identify the potential issues and understanding the datas limitations and strengths. During the Data Understanding phase, we would explore available knowledge graphs, the representativeness of ontology in the graph, question-answer datasets,and any other relevant data sources. This phase might involve analyzing the structure and content of existing knowledge graphs, as well as examining the characteristics of typical questions in the domain of interest.

3. **Data Preparation**
   Data preparation involves the meticulous cleaning and transformation of raw data before it undergoes processing and analysis. This critical preliminary step often includes reformatting data, rectifying errors, and merging multiple datasets to enhance the overall data quality and richness. In our case it would encompass tasks such as formatting ontology data and complementing data annotations, preprocessing question-answer pairs, and potentially augmenting existing graph datasets. This phase also involves techniques like entity linking or relation extraction to enhance the knowledge graph.

4. **Modelling**
   The Modeling phase would firstly focus on implementing embedding algorithms that can effectively traverse and extract information from the ontology of knowledge graphs. This could include experimenting with various ontology embedding methods, and reasoning approaches. After that we may apply various natural language processing (NLP) and machine learning techniques to utilize large language models(LLM) that can accurately answer questions using the knowledge graph. This phase involves experimenting with different algorithms, such as transformer models or graph neural networks, and tuning parameters to achieve the best results.

5. **Evaluation**
   In the Evaluation phase, the performance of the developed models would be assessed using established metrics such as accuracy, F1 score, or mean reciprocal rank. The models should also be examined to check whether it meets the business success criteria set up in step 1. This phase would also encompasses analyzing errors and identifying areas for improvement. In general, evaluation step reviews the work accomplished and check whether there is anything overlooked or whether all steps were properly executed. Based on the assessment, findings are summarized, and necessary corrections are made if needed. Subsequently, decisions are made regarding the next steps: Based on the previous three tasks, determine whether to proceed to deployment, iterate further, or initiate new projects.

6. **Deployment**
   Finally, the Deployment phase would involve publishing and presenting our results to our stakeholders, sharing findings with the academic community, and implementing the models in practical applications. From an industrial perspective, this means integrating the enhanced KGQA system into practical applications of Kadaster, whether

for further industrial use or scientific research. The phase may include scaling the system, optimizing for real-time performance, or developing user interfaces. Furthermore, monitoring its performance, and making necessary adjustments based on user feedback are necessary for further research.

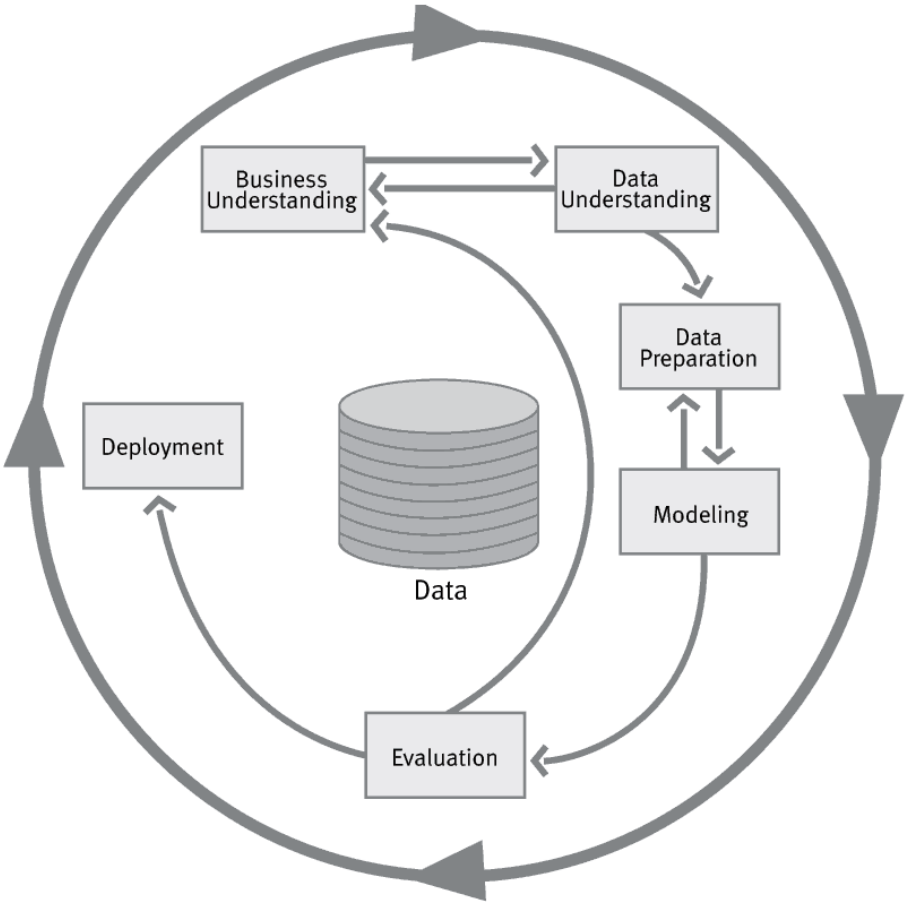

FIGURE 3.1: CRISP-DM Process Model

3.2 Experimental Set-up

This section presents a detailed account of the experimental set-up designed to build the process of combining ontology embedding and LLM to boost the KGQA performance on a real-case knowledge graph and evaluate the efficiency of this integrated approach on KGQA tasks. Our experimental design aims to address several key challenges in the domain:

1. The effective retrieval of relevant information from the extensive ontology data of the knowledge graph.(The selection of embedded ontology)

2. The integration of semantic knowledge represented in the ontology data with state-of-the-art large language models(LLMs).(Ontology embedding and the RAG chain design)

3. The generation of accurate and contextually appropriate responses to diverse query types.(The design of the CoT prompt and the design of the few-shots learning).

To tackle these challenges, we have devised a comprehensive experimental framework that leverages the strengths of ontology embedding, RAG, and CoT prompting. This approach allows us to harness the semantic richness of knowledge graphs while benefiting from the natural language understanding capabilities of large language models. The approach is also aligned with our research methodology process, namely CRISP-DM, providing a systematic framework for our experimental process.

The CRISP-DM methodology consists of six main phases: Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation, and Deployment. We have adapted this process to suit the specific needs of our KGQA research, ensuring a comprehensive and rigorous experimental approach.

3.2.1 Business Understanding

In light of KGQA from a utilitarian perspective, a common application is to use the graph-structured knowledge model to store the interlinked axioms of entities and the relationships that underlie these entities. Knowledge graph highlights the relationships between concepts and things and has been frequently connected to linked open data initiatives [92], as well as large-scale data analytics [93]. First coined by E. W. Schneider in 1972 [94], the term has drawn ubiquitous commercial attention since Google announced its own Knowledge Graph and integrated it into Google search engine in 2012 [95]. Since then, a lot of companies have implemented their own knowledge graphs or knowledge bases. It's closely related to the application of artificial intelligence. For example, Apple utilizes knowledge graph to augment its famous digital assistant Siri; Microsoft also uses knowledge graph to reinforce its search engine Bing to compete with Google [96]. Other cases include Amazon's Alexia [97], Meta's Facebook Graph Search [98], as well as LinkedIn knowledge graph [99]. In these cases, a knowledge graph is often used as a service-oriented information repository to offer support to the front-end agent that interacts directly with the customers, such as an intelligent assistant, chatbot, or search engine. As more and more tech companies attach importance to the development and employment of knowledge graph and ontology technology, the trending sheds light on the promising business value of KGQA.

Kadaster owns a huge amount of open cadastral data of the Netherlands and also takes in charge of a lot of cadastral data from other Dutch governmental sectors. A lot of Dutch government administrations benefit from the data of Kadaster. For example, if a new highway is going to be built, the topographical data and real-estate data on the way must be analyzed for prior planning and decision-making [14]. These data are collected and managed by Kadaster. Additionally, construction and utility companies also have a big demand for basic registration data about buildings and underground pipelines. Accordingly, it's quite important to streamline the retrieval of these data from Kadaster Knowledge Graph. It's also noted that it would be very convenient and cost-effective if the retrieval process only requires minimal domain knowledge in linked data and knowledge base, and provides a user-friendly interface to enable people with only basic computer science knowledge to use it without hassles so that users can accomplish the information retrieval without much assistance from the professional linked data specialists of Kadaster. Hence, it would be essential to build a versatile intelligent question answering system to extract information accurately in the context of KGQA. With such an implementation, it will promote the service quality of Kadaster and foster the collaboration between kadaster and other Dutch government administrations, construction companies and environmental service vendors, facilitate data usage, and improve the data quality in Kadaster.

3.2.2 Data Understanding

The version of KKG we used is 2022.7.6. Its linked data consists of ontology and instance data. The instance data constitutes more than 99% in size of the knowledge graph. There are eight different OWL (Web Ontology Language) ontologies in KKG and each of them is represented as one RDF graph model[1]. The ontologies are listed as follows:

- imbrt(graph:model-imbrt)
- imbaglv(graph:model-imbaglv)
- sor-ext(graph:model-sor-ext)
- sor(graph:model-sor)
- nen3610(graph:model-nen3610)
- kad(graph:model-kad)
- tijdelijk(graph:model-tijdelijk)
- cbs-wkb(graph:model-cbs-wkb)

These eight ontologies, together with their number of knowledge statements and the last updated time are shown in Figure 3.2. The reason of the multiple ontologies is that the data of KKG comes from different datasets owned by different data owners. KKG owns different datasets from the following five data sources:

1. The National Facility of the Base Register of dresses and Buildings(Dutch: De Landelijke Voorziening van de Basisregistratie Adressen en Gebouwen (BAG)).
2. The Top10NL of the Base register of topography(Dutch: De Top10NL van de Basisregistratie Topografie (BRT)).
3. The National Facility of the Base Register of Large-Scale Topography(Dutch: De Landelijke Voorziening van de Basisregistratie Grootschalige Topografie (BGT)).
4. The Open Part of the Base Land Register(Dutch: Het open deel van de Basisregistratie Kadaster (BRK)).
5. The Public Law Restrictions(Dutch: De Publiekrechtelijke Beperkingen (PB)).

Besides, it also hosts a dataset from the data source of Statistics Netherlands(Dutch: Centraal Bureau voor de Statistiek(CBS)). KKG uses the following national and international linked data standards:

- GeoSPARQL (OGC).
- NEN 3610 (Geonovum).
- SHACL, OWL, PROV, RDF, RDFS, SKOS (W3C).

[1] https://data.labs.kadaster.nl/dst/kkg/graphs

So we can also see ontology like graph:model-nen3610 that represents the linked data standard in the ontologies of KKG. As we emphasize ontology data, we mainly analyze and process the ontology graph models here. Each ontology is stored in RDF structure in the triple store of KKG, as Figure 3.3 shows a part of the ontology "graph:model-sor". The ontology data is stored in node relations with a triple relationship(subject-predicate-object) instead of in the table related through keys like in a relational database. Each ontology can be extracted in the RDF serialization format of TriG or Turtle, both providing a good readability of the ontology. However, although TriG and Turtle are also popular serialization formats, they are newer and not as universally supported as RDF/XML. The latter has been around longer and is more entrenched in legacy systems. Furthermore, RDF/XML benefits more from the extensive ecosystem of XML tools and technologies, making it easier to integrate with XML parsers and XML-compatible systems. It is therefore better to transform the ontology into RDF/XML format for the following process of ontology embedding and RAG chain construction. It is also not the best option to handle eight ontologies separately, so it would be logical to integrate and merge the ontologies into one big ontology for processing since they are compatible with one another.

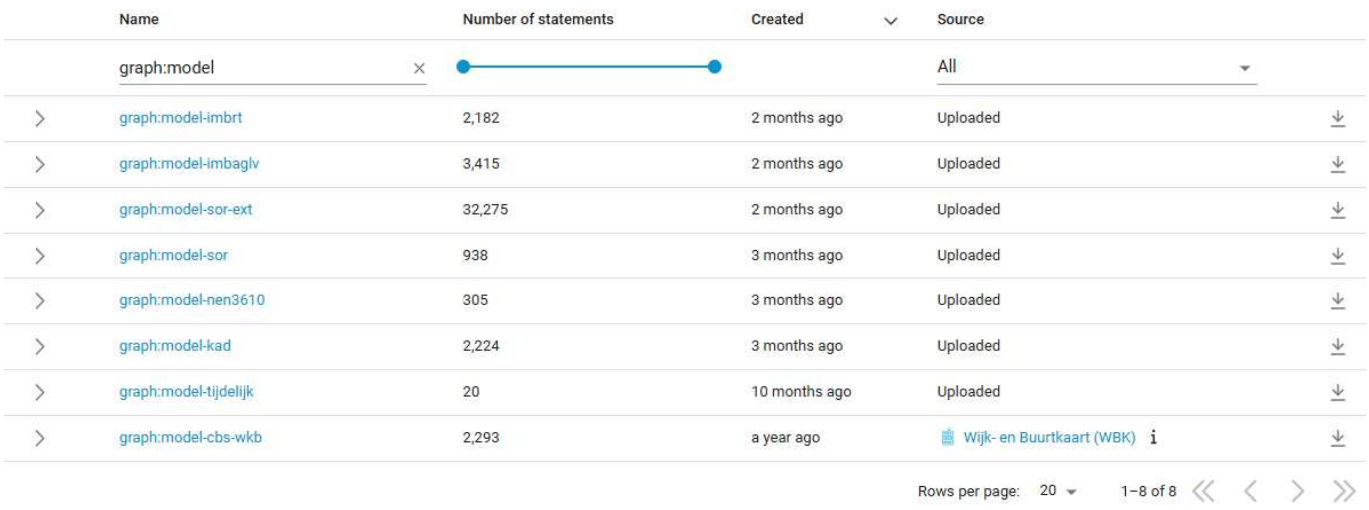

FIGURE 3.2: Graph models of KKG

In the KKG ontology, there are various IRIs that denote the namespaces of different ontology elements, such as a class, property, individual or concept. These IRIs have their corresponding prefixes as a concise representations of the namespaces. Both the prefixes and IRIs are key to the reference of ontology data in the SPARQL queries and the knowledge graph. We have listed all the pertinent prefixes and their IRIs used in the ontology, which will also be useful in the question answering system, in Table 3.1 below. In the following sections, when a certain ontology element is mentioned, we use the prefix form to refer to the mentioned element instead of using the full IRI. This provides better conciseness and readability in the data reference.

3.2.3 Data Preparation

The data contained in the ontology sor-ext is irrelevant to the question answering system, so it is excluded from the ontology selection. The remaining seven ontologies are selected for further data preparation. The download of the ontologies sor, nen3610, kad,

Subject	Predicate	Object	Graph
			https://data.kkg.kadaster.nl/... ×
kad:plusStatus	rdf:type	owl:ObjectProperty	*graph:model-sor*
kad:plusStatus	rdfs:isDefinedBy	https://data.kkg.kadaster.nl/kad/model/	*graph:model-sor*
kad:plusStatus	rdfs:range	kad:PlusStatus	*graph:model-sor*
kad:plusStatus	skos:prefLabel	plus status	*graph:model-sor*
kad:relatieveHoogteligging	rdf:type	owl:DatatypeProperty	*graph:model-sor*
kad:relatieveHoogteligging	rdfs:domain	sor:Gebouw	*graph:model-sor*
kad:relatieveHoogteligging	rdfs:isDefinedBy	https://data.kkg.kadaster.nl/kad/model/	*graph:model-sor*
kad:relatieveHoogteligging	rdfs:range	xsd:integer	*graph:model-sor*
kad:relatieveHoogteligging	skos:prefLabel	relatieve hoogteligging	*graph:model-sor*
kad_shp:Gebouw_plusStatus	rdf:type	sh:PropertyShape	*graph:model-sor*
kad_shp:Gebouw_plusStatus	sh:class	kad:PlusStatus	*graph:model-sor*
kad_shp:Gebouw_plusStatus	sh:in	[...e9f]	*graph:model-sor*
kad_shp:Gebouw_plusStatus	sh:maxCount	1	*graph:model-sor*
kad_shp:Gebouw_plusStatus	sh:path	kad:plusStatus	*graph:model-sor*
kad_shp:Gebouw_relatieveHoogteligging	rdf:type	sh:PropertyShape	*graph:model-sor*

FIGURE 3.3: Some ontology triples of graph:model-sor

Prefix	Full IRI
owl	http://www.w3.org/2002/07/owl#
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs	http://www.w3.org/2000/01/rdf-schema#
xsd	http://www.w3.org/2001/XMLSchema#
nen3610	https://data.kkg.kadaster.nl/nen3610/model/def/
nen3610-shp	https://data.kkg.kadaster.nl/nen3610/model/shp/
skos	http://www.w3.org/2004/02/skos/core#
bag	http://bag.basisregistraties.overheid.nl/def/bag#
bag-begrip	http://bag.basisregistraties.overheid.nl/id/begrip/
brt	http://brt.basisregistraties.overheid.nl/def/top10nl#
wbk	https://data.labs.kadaster.nl/cbs/wbk/vocab/
geo	http://www.opengis.net/ont/geosparql#
kad	https://data.kkg.kadaster.nl/kad/model/def/
kad-con	https://data.kkg.kadaster.nl/kad/model/con/
kad-shp	https://data.kkg.kadaster.nl/kad/model/shp/
sor	https://data.kkg.kadaster.nl/sor/model/def/
sor-con	https://data.kkg.kadaster.nl/sor/model/con/
sor-shp	https://data.kkg.kadaster.nl/sor/model/shp/
bgt	http://bgt.basisregistraties.overheid.nl/def/bgt#
bgt-pand	http://bgt.basisregistraties.overheid.nl/id/pand/
bnode	https://data.kkg.kadaster.nl/well-known/genid/
brt	http://brt.basisregistraties.overheid.nl/def/top10nl#
brt-gebouw	http://brt.basisregistraties.overheid.nl/id/gebouw/
brt-scheme	http://brt.basisregistraties.overheid.nl/top10nl/id/scheme/
brt-shp	http://brt.basisregistraties.overheid.nl/top10nl/id/shape/
foaf	http://xmlns.com/foaf/0.1/
gebouw	https://data.kkg.kadaster.nl/id/gebouw/
bouwzone	https://data.kkg.kadaster.nl/id/gebouwzone/
gemeente	https://data.kkg.kadaster.nl/id/gemeente/
mim	http://bp4mc2.org/def/mim#
purl	http://purl.org/linked-data/cube#
prov	http://www.w3.org/ns/prov#
gml	http://www.opengis.net/ont/gml#
time	http://www.w3.org/2006/time#
shacl	http://www.w3.org/ns/shacl#
schema	https://schema.org/
triplydb	https://triplydb.com/Triply/value/

TABLE 3.1: Namespace Prefixes and Full IRIs

tijdelijk and cbs-wkb was done on September 26th, 2024 and the download of the ontologies imbrt and imbaglv was done on Octorber 4th, 2024. All ontologies were extracted and serialized in TriG format. Then they were all transformed into RDF/XML format with Protégé [100]. The ontologies were then merged together into one ontology of a single file with ROBOT[1] [101] of version 1.9.6 running on JVM version 11. The merged ontology will be named **kkg_onto** in the following chapters. Based on

[1] https://robot.obolibrary.org/

Table 3.1, a succinct identifier is assigned to each ontology element according to the namespace of the element and its detailed local name for a concise data reference and an enhancement of readability. For example, for the ontology class denoted by the IRI "`https://data.kkg.kadaster.nl/kad/model/def/Gebouwtype`", we use the prefix "kad" to represent the namespace "`https://data.kkg.kadaster.nl/kad/model/def/`" and the specific local name "Gebouwtype" that denotes the class name, to form a succinct "kad:Gebouwtype" as the identifier of this class. Similarly, for object property "`http://bag.basisregistraties.overheid.nl/def/bag#maaktDeelUitVan`" and data property "`http://bag.basisregistraties.overheid.nl/def/bag#gebruiksdoel`", we have "bag:maaktDeelUitVan" and "bag:gebruiksdoel" respectively as their identifier. This will be used in ontology embedding and and RAG chain built-up as well. We will also use the concise identifier to refer to a certain ontology element in the following sections.

The data was still not ready yet since some ontology axioms were still missing or provided incorrect information that would worsen the embedding quality, thus it must be further preprocessed. We did some model enrichment work here to complement and correct some information in kkg_onto. Specifically, rdfs:comment, which is an ontology axiom that provides a human-readable definition and explanation to an ontology entity, was incorrectly defined in the class nen3610:FunctioneleRuimte(English: Functional Space). We then modified the rdfs:comment of the class nen3610:FunctioneleRuimte and added a new class named TransportRuimte(English: Transport Space) with the same prefix nen3610 as nen3610:TransportRuimte. The former annotation of nen3610:FunctioneleRuimte in its rdfs:comment was given to nen3610:TransportRuimte, because it actually denoted the definition of nen3610:TransportRuimte and was not precise/representative enough to be a definition of the nen3610:FunctioneleRuimte.

Likewise, an empty class kad:Gebruiksdoel is removed. The reason is that there are two classes with the same name but with difference namespaces, namely kad:Gebruiksdoel and sor: Gebruiksdoel. After careful examine, it is plausible that only sor:Gebruiksdoel serves for all ontology entities related to the concept Gebruiksdoel(English: Use Purpose) and kad:Gebruiksdoel is not connected to any instances in the ontology. It is an isolated empty class and therefore is removed from the ontology class. In OWL ontologies, object properties and data properties play crucial roles in defining relationships between entities and their attributes. Object properties link individuals (instances of classes) to other individuals, while data properties link individuals to data values (literals) that are also set as class instances. Through object properties and data properties, we connect class instances to constitute triple relations. In the ontology object property, there is an ontology axiom called rdfs:range, which refers to an ontology class whose instance is the object of the triple relation, and another ontology axiom called rdfs:domain. which refers to an ontology class whose instance is the subject of the triple relation. The object property itself serves as the predicate of the triple relation. Data properties also work in a similar way. Thus, the class kad:Gebruiksdoel is also deleted from the rdfs:range of the object property sor:gebruiksdoel because it contains no instances as the triple objects. Similarly, in the data property sor:oppervlakte (English: surface area), two rdfs:range existed, namely, xs:nonNegativeInteger and xs:positiveInteger. They indicated that in the triple relation where sor:oppervlakte was a predicate, the object must be an instance of xs:nonNegativeInteger and xs:positiveInteger simultaneously. However, in XML schema, xs:nonNegativeInteger includes all integers that are zero or positive (0, 1, 2, ...), while xs:positiveInteger includes only positive integers (1, 2, ...). So xs:positiveInteger is a subset of xs:nonNegativeInteger and specifying both as the ranges of the same property is

redundant. In addition, it is also recommended to have a single rdfs:range for each property instead of multiple rdfs:range values to avoid complications in data validation and reasoning [102] [103]. Considering the surface area of a building can be any non-negative values, the redundant xs:positiveInteger was removed from the rdfs:range of the data property sor:oppervlakte.

It is also observed that in the ontology, the object property geo:sfWithin has the class geo:SpatialObject as both rdfs:domain and rdfs:range, which means that both the subject and object in this triple relation are instances of the class geo:SpatialObject. The object property geo:sfWithin is part of the GeoSPARQL standard, which is used for representing and querying geospatial data on the semantic web. Specifically, geo:sfWithin is a topological property that indicates a spatial relationship where one geometry is within another geometry. In KKG, geo:sfWithin plays an important role in the instance level to build the connections between different national administrative geometries. For example, all the instances of sor:Gemeente(English: municipality) lies within instances of sor:Provincie(English: province) through geo:sfWithin. Similarly, via geo:sfWithin, an instance of wbk:Wijk(English: district) lies within an instance of wbk:Gemeente, an instance of wbk:Buurt(English: neighborhood) lies within an instance of wbk:Wijk, an instance of sor:Gebouw(English: Building) lies within an instance of wbk:Buurt, an instance of sor:Perceel(English: plot, a small piece of land) also lies within an instance of sor:Gemeente. The triple relation connected by geo:sfWithin were well defined in instance data. However, on the ontology level, the classes sor:Provincie, sor:Gemeente, wbk:Gemeente, wbk:Wijk, wbk:Buurt, sor:Gebouw and sor:Perceel are not connected via any properties to construct triple relations, although their intrinsic relations are widely expressed on the instance level.

Therefore, for model enrichment, the triple relations between these classes that indicate national administrative geometries should be constructed on the ontology level to be aligned with the pattern of the instance data, which will help with the ontology embedding and model reasoning. Our method is to fill the gap in the class, not in the object property since the property geo:sfWithin already has an rdfs:domain and an rdfs:range. We add a parent class geo:SpatialObject respectively to these seven administrative geometry classes. Now they are all subclasses of geo:SpatialObject. Through the relation between geo:SpatialObject and geo:sfWithin, the parent class enrichment serves as a foundation of the triple relation build-up between corresponding administrative geometry classes on the ontology level to conform to the existing relations in instance data. Listing 3.1 shows the class sor:Provincie after its parent class geo:SpatialObject is complemented.

Now we seem to have a plausible chain from a very fundamental geometry gebouw(building) and perceel(plot) to a high-level administrative geometry provincie(province). But it is noted that the classes sor:Gemeente and wbk:Gemeente are different, and they don't lie within each other since they refer to the same administrative level but belong to different ontologies. On the instance level, each sor:Gemeente instance is connected to the corresponding gemeente instance of wbk:Gemeente with the property owl:sameAs. But the triple relation is still missing on the ontology level. To reflect this relation on the ontology level, an owl:equivalentClass property has been added to the class sor:Gemeente. An updated class sor:Gemeente is shown in Listing 3.2. The chain that links gebouw(building) and perceel(plot) to provincie is thus complete as shown in Figure 3.4. The refinement of the chain to construct a better ontology embedding for the syntactically and semantically accurate creation of SPARQL queries will be further discussed in the Modeling Section.

Listing 3.1: The Enriched Class sor:Provincie

```
<!-- https://data.kkg.kadaster.nl/sor/model/def/Provincie -->

<Class rdf:about="https://data.kkg.kadaster.nl/sor/model/def/
   Provincie">
    <rdfs:subClassOf rdf:resource="https://data.kkg.kadaster.nl/sor
       /model/def/Overheidsorganisatie"/>
    <rdfs:subClassOf rdf:resource="http://www.opengis.net/ont/
       geosparql#SpatialObject"/>
    <terms:subject rdf:resource="http://brk.basisregistraties.
       overheid.nl/id/begrip/Provincie"/>
    <rdfs:comment xml:lang="nl">Een provincie is een openbaar
       lichaam met zelfbestuur en bevoegdheid binnen een gebied
       kleiner dan die van het Rijk.</rdfs:comment>
    <rdfs:isDefinedBy rdf:resource="https://data.kkg.kadaster.nl/
       sor/model/"/>
    <rdfs:label xml:lang="nl">Provincie</rdfs:label>
</Class>
```

Listing 3.2: The Enriched Class sor:Gemeente

```
<!-- https://data.kkg.kadaster.nl/sor/model/def/Gemeente -->

<Class rdf:about="https://data.kkg.kadaster.nl/sor/model/def/
   Gemeente">
    <rdfs:subClassOf rdf:resource="https://data.kkg.kadaster.nl/sor
       /model/def/Overheidsorganisatie"/>
    <rdfs:subClassOf rdf:resource="http://www.opengis.net/ont/
       geosparql#SpatialObject"/>
    <owl:equivalentClass rdf:resource="https://data.labs.kadaster.
       nl/cbs/wbk/vocab/Gemeente"/>
    <terms:subject rdf:resource="http://brk.basisregistraties.
       overheid.nl/id/begrip/Gemeente"/>
    <rdfs:comment xml:lang="nl">Afgebakend gedeelte van het
       grondgebied van Nederland, onder zeggenschap van een
       openbaar lichaam met diverse bestuurlijke taken, ingesteld
       op basis van artikel 123 van de Grondwet en de Gemeentewet
       .</rdfs:comment>
    <rdfs:isDefinedBy rdf:resource="https://data.kkg.kadaster.nl/
       sor/model/"/>
    <rdfs:label xml:lang="nl">Gemeente</rdfs:label>
</Class>
```

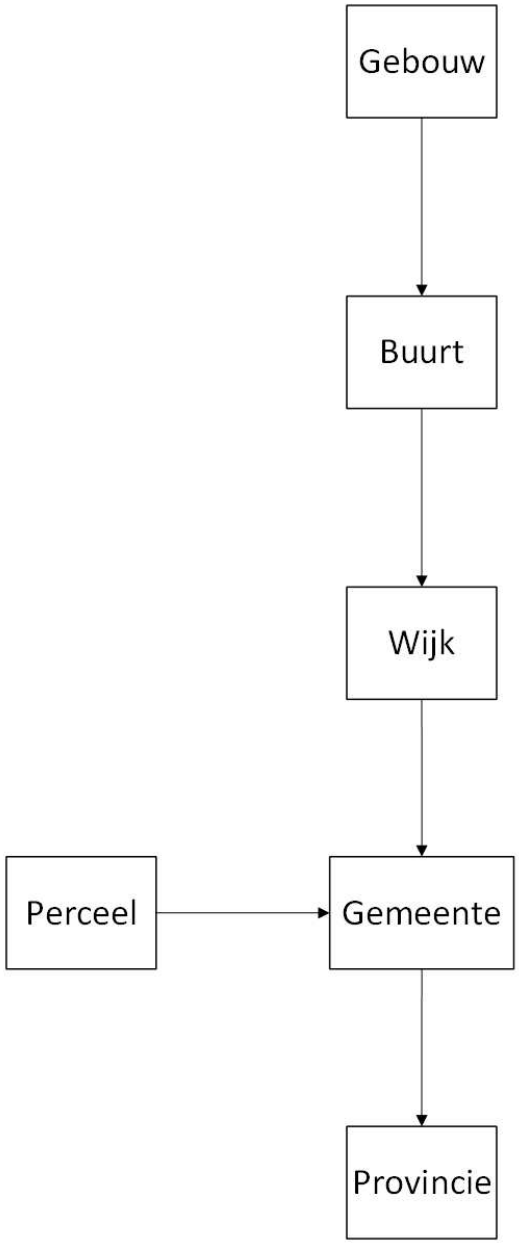

FIGURE 3.4: Geometry Chain: From Building&Plot to Province

3.2.4 Modeling

The preliminary stages in data understanding and preparation set the foundation for the ontology embedding and the RAG chain construction of our question answering system modeling.

3.2.4.1 Ontology Embedding

The purpose of ontology embedding is to vectorize the ontology data of KKG for the similarity matching between ontology elements and question elements. The vectorized ontology data will be stored in a vector database as an external data source for the LLM to generate solution SPARQL queries to the questions, using Retrieval-augmented Generation. The ontology elements will also be the key elements in the generated SPARQL queries. This process is a significant part of the RAG chain construction. We have prepared the ontology data for the embedding in Data Preparation. But not all the ontologies in hand have to be embedded, we still need to design a pattern to contain the more representative information of the ontologies in the embedding and neglect the other parts. The same embedding model will be used for both the ontologies and the questions to make sure their vectors are comparable and matchable. The embedding of the questions is easier since the questions are short pieces of texts in natural language and the current NLP techniques are good enough to handle the cases. The embedding of the ontologies are more tricky, as they are in the form of RDF/XML, as demonstrated in Listing 3.1 and 3.2. Not all information in the XML-structured ontology element are useful for the SPARQL generation. If everything of one ontology element is treated as plain text and embedded like a natural language question without pre-filtering, there is useless or redundant information stored in the embedding, which incurs too much noise. Therefore, it is necessary to select the most useful axioms from an ontology element as the input of ontology embedding.

Not all the ontology elements are useful for the SPARQL query generation and our question answer system as well. In an owl ontology, there are usually several types of key ontology elements: Classes, Individuals(Instances), and Properties. This is also the case of our ontology. Classes represent abstract concepts or categories of entities within a domain, such as Person, Animal, or Building. They are used to group individuals (instances) that share common characteristics. They are fundamental components of an ontology and are used to define the structure and semantics of the data. [104] Individuals, on the other hand, represent specific objects or entities within the domain of interest. They are the concrete instances of classes and can have properties that describe their attributes and relationships with other individuals. Although most of the instance data of KKG are in different data sets from the ontologies, there are some typical individuals included in our KKG ontologies. Properties represent relationships between individuals or between individuals and data values. They are crucial for defining the structure and semantics of the ontology by specifying how entities are connected and what attributes they possess. There are three types of properties in our ontologies, including: Object Properties, Data Properties and Annotation Properties. Object Properties link individuals to other individuals. Data Properties link individuals to data values and data types. Annotation properties, such as rdfs:comment and rdfs:label, provide metadata about classes, properties or individuals. In our question answering task, we concentrate on SPARQL queries whose clauses rely heavily on the triple patterns that are mainly constituted and specified by ontology classes and properties. Consequently, our focus is on the embedding of

these two types of ontology elements. The knowledge graph is constructed with graph nodes(entities) and edges(relations). To traverse through the graph and connect different entities in the query, we need the property paths to involve repeated traversal of edges. A property path, in the context of RDF graph and SPARQL queries, represents a potential route between two nodes within a graph. A simple instance of this is a property path of length one, which corresponds to a single triple relation. The ends of such paths can be either RDF terms or variables. Property paths facilitate more succinct representations of certain SPARQL basic graph patterns and enhance the capability to determine the connectivity between two resources through paths of arbitrary lengths [105]. An example of a property path of our ontologies can be seen in the last line of the WHERE clause of the SPARQL query below. In this example, a property path, which contains properties such as owl:sameAs and geo:sfWithin, connect the instances of class sor:Gebouw to the instances of class sor:Gemeente. The two class nodes are connected with a 4-step property path, because there are other intermediate classes, namely wbk:Gemeente, wbk:Wijk, wbk:Buurt between these two classes, which can be connected directly with properties such as owl:sameAs and geo:sfWithin in a triple pattern, as we discuss in the geometry chain example in Data Preparation. In this context, a property path can actually be regarded as a "chain". It indicates the feature of the knowledge graph and linked data that despite not being adjacent to each other, these data nodes can be linked to each other as long as there are sufficient properties to construct a property path, no matter how many steps the path will take. And this feature can be utilized to query a data node in SPARQL.

```
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX provincie: <https://data.kkg.kadaster.nl/id/provincie/>
PREFIX sor: <https://data.kkg.kadaster.nl/sor/model/def/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>

SELECT (COUNT(DISTINCT ?geb) as ?aantal)
WHERE {
    ?geb a sor:Gebouw;
         sor:oorspronkelijkBouwjaar ?bo.

    ?gemeente geo:sfWithin provincie:0023.
    ?gemeente owl:sameAs/^geo:sfWithin/^geo:sfWithin/^geo:sfWithin
       ?geb.
}
LIMIT 9999
```

This analysis emphasizes that the classes and properties form an interconnected ecosystem. It is therefore logical to consider them together in the ontology embedding. More precisely, we should start the embedding from properties instead of isolated classes, because through the rdfs:range and rdfs:domain of a property, we can traverse to the classes and find property paths to expand the network across the whole ontology graph. Eventually, we discover all useful links to connect every possible nodes. This information is vectorized and stored in our vector database as the retrieval data of the RAG chain. The LLM can reason over these data to understand all the possible property paths and generate SPARQL queries against any path of an arbitrary length to query the linked data of any node and any edge on this path. This potentially broadens the spectrum of questions that can be answered, thereby enhancing the system's flexibility.

We use PGVector, version 0.3.6, as our vector store. In PGVector, a single data point unit is typically referred to as a document. A document contains page content and metadata.

Page content refers to the actual content or data stored in a vector format, such as the text of a file, the content of a webpage, or any other data that has been converted into a vector representation. A metadata, on the other hand, is additional information that describes the page content. We start the embedding from properties, including object properties and data properties. And we only make embedding for the useful ontology axioms of a property. We need to decide the useful axioms to be embedded in a property, then try to organize the information of these ontology axioms in human-readable text. Next, text embedding can be implemented to vectorize these texts. Accordingly, these embedding vectors will be stored in page content. Listing 3.3 demonstrates a property example of the object property kad:gebouwtype. It is important that we can expand the property path from a property to classes and other properties. Via rdfs:domain, we can connect to the subject class sor:Gebouwzone of the triple relation in which this property serves as the predicate, while via rdfs:range we can connect to the object class kad:Gebouwtype of the same triple relation. Similarly, in the rest of the object properties and data properties, we can trace to other classes through rdfs:domain and rdfs:range so these two axioms should be included in the embedding data. Leveraging the robust reasoning capabilities of large language models (LLMs), it is plausible to incorporate these triple patterns into the embedding process to build interconnected property paths. This integration can assist LLMs in generating SPARQL queries to effectively answer questions. In this context, each triple relation should be represented in a single document of PGVector. Therefore, a piece of text containing the triple relation is constructed for each document.

Listing 3.3: Object Property: kad:gebouwtype

```
<!-- https://data.kkg.kadaster.nl/kad/model/def/gebouwtype -->

<ObjectProperty rdf:about="https://data.kkg.kadaster.nl/kad/model/
    def/gebouwtype">
     <rdfs:domain rdf:resource="https://data.kkg.kadaster.nl/sor/
        model/def/Gebouwzone"/>
     <rdfs:range rdf:resource="https://data.kkg.kadaster.nl/kad/
        model/def/Gebouwtype"/>
     <terms:source rdf:datatype="http://www.w3.org/2001/XMLSchema#
        anyURI">https://kadaster.github.io/imbrt/#attribuut-type-
        gebouw</terms:source>
     <rdfs:isDefinedBy rdf:resource="https://data.kkg.kadaster.nl/
        kad/model/"/>
     <skos:definition xml:lang="nl">Het type gebouw, het doel
        waarvoor de bebouwing gebruikt wordt (gaat worden / werd).</
        skos:definition>
     <skos:prefLabel xml:lang="nl">gebouwtype</skos:prefLabel>
     <skos:scopeNote xml:lang="nl">Indien de betreffende bebouwing
        onderdeel uitmaakt van een groter geheel, wordt het
        gebouwdeel "uitgesneden" uit de omliggende bebouwing en
        voorzien van de bijbehorende attribuutwaarde. Indien het
        gebouwdeel niet exact is aan te geven, wordt een op zich
        zelf staand gebouw aangebracht aan de rand van het complete
        gebouw (op de ware plek of bij de dichtsbij gelegen ingang
        van het gebouw).</skos:scopeNote>
</ObjectProperty>
```

The text includes a declarative sentence that functions as a statement of the triple rela-

tion in natural language, and a Turtle-like [106] triple pattern that clearly specifies the subject, predicate and object of the triple relation. The text is set as the page content of the document. In the Turtle-like triple pattern, we use the concise identifier to represent the ontology element, which gives details about the namespace and the name of the resource(class, property or individual). When we extract the triple pattern from a property, the subject class and object class are also traced in the ontology and their necessary axioms, such as identifiers and labels are extracted as well. The sentence is more likely to be an explanatory statement to the triple pattern that facilitate the understanding and reasoning of LLMs. It is basically a simple declarative sentence structured linguistically by a subject, a predicate and an object, which are the subject class, the property and the object class respectively. For a concise formulation of the sentence, we prioritize to use the label(either rdfs:label [102] or skos:prefLabel [107]) of the ontology element to denote the element itself instead of using the element's identifier in the declarative sentence, since the label provides a comprehensive but simple description to the element without the namespace prefix, which we already have in the Turtle-like triple pattern. This reduces noise and redundant information in the embedding. In addition, the metadata of the document can provide annotations to the triple relations, so we clearly specify the information of the elements of the triple pattern embedded in this document in the document's metadata, including the role(subject,predicate,object) this ontology plays in the triple relation, and its unique identifier. The document example of the triple relation we extracted from the object property kad:gebouwtype can be seen in the following box. But not all the ontology elements have a label. In case an element doesn't have a label, we use its name of resource, namely its identifier without the namespace prefix, as its label. This is usually very similar to the label according to the feature of KKG. For example, if the class sor:Gebouwzone doesn't originally have a label in the ontology, we extract "Gebouwzone" from "sor:Gebouwzone" to serve as its label.

Document Containing A Triple Pattern

```
page_content='Gebouwzone gebouwtype Gebouwtype,
subject:<sor:Gebouwzone>predicate:<kad:gebouwtype>object:<kad:Gebouwtype>'
metadata={
'subject': 'sor:Gebouwzone',
'predicate': 'kad:gebouwtype',
'object': 'kad:Gebouwtype',
'id': 'sor:Gebouwzone kad:gebouwtype kad:Gebouwtype;'
}
```

The triple pattern extraction is implemented in all object properties and data properties to construct as complete property paths as possible. Each triple relation is embedded and stored in a document in PGVector. Besides building property paths from object properties and data properties, the subclass relations are also important for the completion of the property paths, since subclasses inherently remain the properties of their parent classes. If a parent class is connected on the property path, then its subclasses should be connected as well. This also facilitates the expansion of the property paths network. We try to elaborate this layer of relation to the LLMs through our embedding data and promote the LLMs' understanding of this logical relation over its SPARQL inference. The similar embedding pattern is implemented to extract the subclass relations. We start from ontology classes, then try to find out the ontology axioms that state potential subclass relations. We

construct vector documents of these subclass relations, which are constituted of page content comprised of a manually designed text to unambiguously specify the subclass relation, and the metadata of the pertinent classes. An example of class sor:Verblijfsobject is displayed in Listing 3.4. It is clearly stated that class sor:Verblijfsobject is the subclass of class kad:AdresseerbaarObject in the axiom rdfs:subClassof. Therefore, we can extract the subclass relation and track the details of parent classes from the ontology axiom rdfs:subClassof. We make embedding and store the embedding of one subclass relation in a single document. In the page content, we construct a declarative text that makes a clear statement of the subclass relation. It can be seen below in the example. In pursuit of simplicity, this text only contains this sentence. It is clear enough for the LLMs to interpret the subclass relations. We do not have to set up any triple patterns to elaborate this relation. Despite the conciseness of the label of an ontology element, we decide to use the unique identifier of the class instead of its label in the text. This is because the namespace prefix is crucial for the class identification. Classes from different namespaces may share the identical resource names and labels, such as sor:Verblijfsobject and bag:Verblijfsobject. However, the unique identification of a class is determined by the combination of its namespace and resource name. Therefore, if we do not have a triple pattern here to specify the namespace of a class, the namespace prefixes have to be incorporated in the declarative sentence to prevent ambiguity and incorrect connections.

LISTING 3.4: Class: sor:Verblijfsobject

```
<!-- https://data.kkg.kadaster.nl/sor/model/def/Verblijfsobject -->

<Class rdf:about="https://data.kkg.kadaster.nl/sor/model/def/
    Verblijfsobject">
     <rdfs:subClassOf rdf:resource="https://data.kkg.kadaster.nl/kad
        /model/def/AdresseerbaarObject"/>
     <terms:subject rdf:resource="http://bag.basisregistraties.
        overheid.nl/id/begrip/Verblijfsobject"/>
     <rdfs:comment xml:lang="nl">De kleinste binnen één of meer
        gebouwen gelegen eenheid van gebruik die ontsloten wordt via
         een eigen afsluitbare toegang vanaf de openbare weg, een
        erf of een gedeelde verkeersruimte, onderwerp kan zijn van
        goederenrechtelijke rechtshandelingen en in functioneel
        opzicht zelfstandig is.</rdfs:comment>
     <rdfs:isDefinedBy rdf:resource="https://data.kkg.kadaster.nl/
        sor/model/"/>
     <rdfs:label xml:lang="nl">Verblijfsobject</rdfs:label>
</Class>
```

Document of a Subclass Relation

```
page_content='sor:Verblijfsobject is a subclass of kad:AdresseerbaarObject'
metadata={
'class_id': 'sor:Verblijfsobject',
'class_label': 'verblijfsobject',
'parentClass_id': 'kad:AdresseerbaarObject',
'parentClass_label': 'Adresseerbaar object'
}
```

Besides classes and properties, some individuals should also be considered in the embedding according to the requirements and preferences of users from Kadaster because these data individuals are associated with the questions that this question answering system is expected to answer. These individuals are mainly the instances of the classes sor:Gebruiksdoel, kad:Gebouwtype and bag:Gebruiksdoel. They are pertinent to important geographical entities associated with the cadastral query, such as kerken(English: churches), ziekenhuizen(English: hospitals), watertoren(English: water towers). We use the similar embedding method to construct documents for these individuals. In the text, the key point is to correctly specify that these individuals are the instances of their corresponding classes and ensure the accurate interpretation from the reasoning so that the LLMs can make correct relation expression in the clauses of generated SPARQL queries.

Listing 3.5 shows an example of the individual kad-con:gemeentehuis(English: town hall). The class information of an individual can be found in the property rdf:type. So for individual kad-con:gemeentehuis, it is an instance of both classes skos:Concept and kad:Gebouwtype. This information will automatically be extracted and fill in the declarative statement in page content of the document. In page content we also list other relevant ontology axioms of the individuals, such as its label and definition. This provide more information about the individuals and help LLMs understand and distinguish different instances in the query generation. It is especially helpful because individuals, offering more details, provide more specific information than classes and properties, thus they are of a higher granularity. This allows many different ways to ask the same questions regarding the individuals. For example, if a question is asked about the towl hall: Waar is het dichtstbijzijnde gemeentehuis vanaf mijn adres Drienerlolaan 5, 7522 NB Enschede?(English: Where is the closest town hall from my address Drienerlolaan 5, 7522 NB Enschede?), besides "gemeentehuis", the users can also use other words like "raadhuis" and "stadhuis" to refer to a town hall in Dutch language. Providing the definition of an individual is thus informative for the LLMs to process liguistic synonyms of the same entities and different paraphrases of the same questions. The definitions are extracted from the ontology axioms rdfs:comment, skos:definition or shacl:description. The labels are extracted for the ontology axioms rdfs:label, skos:prefLabel or shacl:name. We use the package rdflib[1] here to streamline the ontological information retrieval of individuals. When the rdf:type information is extracted with rdflib, it will automatically add the class owl:NamedIndividual, because every owl individual is an instance of the class owl:NamedIndividual. But this information is redundant, so we erase the class owl:NamedIndividual from each document. A constructed document of the individual kad-con:gemeentehuis is demonstrated below in the example.

The embedding patterns have so far incorporated the key information of classes, properties and selected individuals, which forms the underlying layer of KKG and serves as the key foundation to write SPARQL queries against this knowledge graph. Besides the standard patterns extracted from the ontology and vectorized in the embedding, some customized data should also be added to the vector store to align with the data preparation process to provide better clarification. Precisely, these special customized embedding is mainly related to the process of building property paths between province class and building/plot class on the ontology level. Since we fill the gap of the property paths through adding parent class geo:SpatialObject to these administrative geometry classes, the relations are reflected through the embedding of the subclass relations. But they are still not yet directly incorporated in the triple pattern embedding. To address this, it is logical to add

[1] https://github.com/RDFLib/rdflib

Listing 3.5: Individual: kad-con:gemeentehuis

```
<!-- https://data.kkg.kadaster.nl/kad/model/con/gemeentehuis -->

<NamedIndividual rdf:about="https://data.kkg.kadaster.nl/kad/model/
    con/gemeentehuis">
     <rdf:type rdf:resource="http://www.w3.org/2004/02/skos/core#
        Concept"/>
     <rdf:type rdf:resource="https://data.kkg.kadaster.nl/kad/model/
        def/Gebouwtype"/>
     <rdfs:isDefinedBy rdf:resource="https://data.kkg.kadaster.nl/
        kad/model/"/>
     <skos:definition xml:lang="nl">Gebouw waarin de gemeentelijke
        secretarie gevestigd is.</skos:definition>
     <skos:inScheme rdf:resource="https://data.kkg.kadaster.nl/kad/
        model/scheme/Gebouwtype"/>
     <skos:prefLabel xml:lang="nl">gemeentehuis</skos:prefLabel>
     <skos:scopeNote xml:lang="nl">Dependances van deze secretarie
        worden als stadskantoor, hulpsecretarie aangegeven. Indien
        het gemeentehuis omwille van ruimtegebrek over verschillende
         gebouwen is verspreid, is er geen sprake van dependances.</
        skos:scopeNote>
</NamedIndividual>
```

Document of an Individual

```
page_content="gemeentehuis is an individual of class ['skos:Concept',
'kad:Gebouwtype'],
{
'individual': 'kad-con:gemeentehuis',
'rdf:type': ['skos:Concept', 'kad:Gebouwtype'],
'label': 'gemeentehuis',
'definition': 'Gebouw waarin de gemeentelijke secretarie gevestigd is.'
}"
metadata={
'individual': 'kad-con:gemeentehuis',
'rdf:type': ['skos:Concept', 'kad:Gebouwtype'],
'label': 'gemeentehuis',
'definition': 'Gebouw waarin de gemeentelijke secretarie gevestigd is.'
}
```

the triple pattern relations manually to the embedding. We use the similar method of extracting the triple relations from owl properties to fill the document of these relations. An example of the document of the triple relation between the class sor:Gemeente and the class sor:Provincie is shown in the box below. The complete property path between the class sor:Gebouw/sor:Perceel and sor:Provincie is addressed in the triple pattern embedding and demonstrated in Figure 3.5. To check the full documents of the cusomized triple patterns, please check the Appendix.

Document of a Customized Triple Pattern

```
page_content='Gemeente within Provincie,
subject:<sor:Gemeente>predicate:<geo:sfWithin>object:<sor:Provincie>'
metadata={
'subject': 'sor:Gemeente',
'predicate': 'geo:sfWithin',
'object': 'sor:Provincie',
'id': 'sor:Gemeente geo:sfWithin sor:Provincie;'
}
```

This subsection basically introduces the specific methods and process of the ontology embedding, which is the significant foundation of building our question answering system in the RAG technique. The following section will introduce the prompt engineering part of the RAG chain, which is another prerequisite of our question answering system.

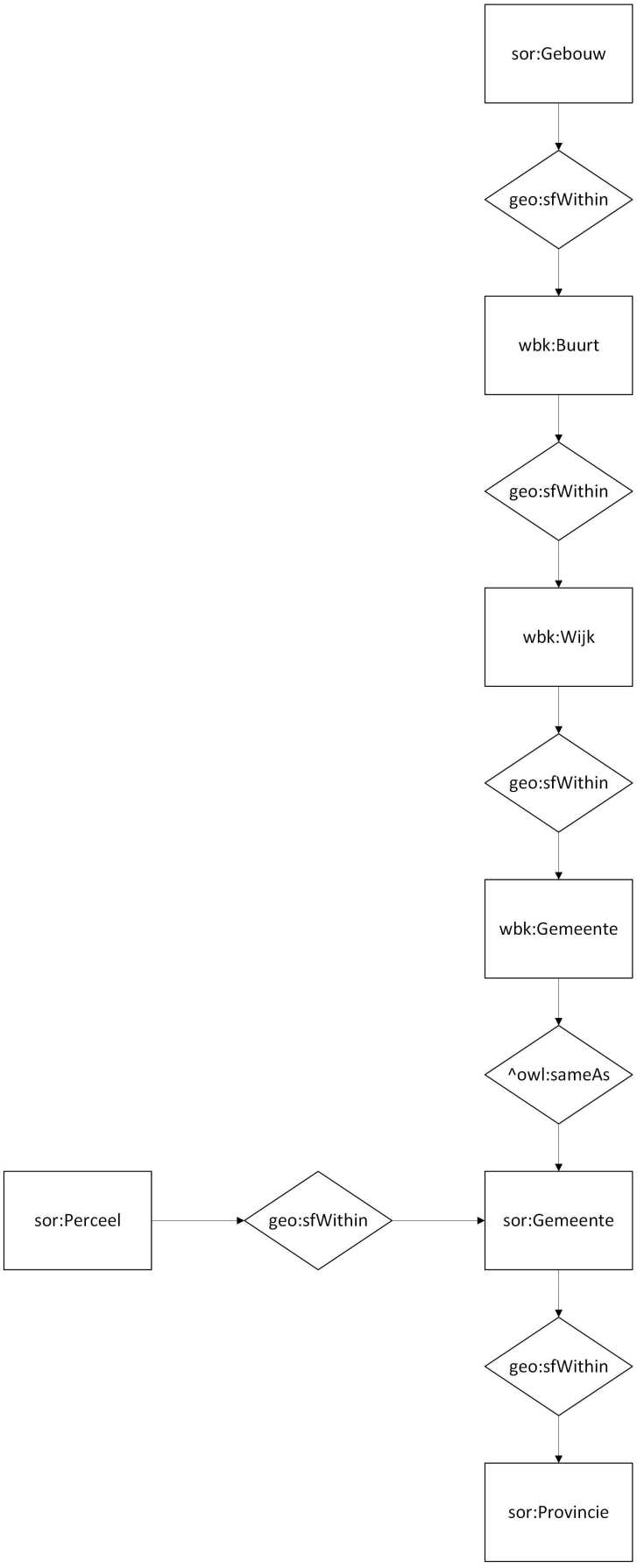

FIGURE 3.5: The Complete Property Path: From Building&Plot to Province

3.2.4.2 Prompt Engineering

Prompt engineering plays a critical role in optimizing the performance of the LLMs by shaping how these models interpret and respond to tasks. Since LLMs rely on contextual information provided in their inputs, a well-designed prompt can guide the model's attention, elicit reasoning, and improve task-specific outcomes [108]. By structuring prompts effectively, the model behavior can be aligned with our desired outputs, reducing ambiguity and enhancing accuracy [109]. In this section, we introduce how we use Chain-of-Thought(CoT) prompting as the main prompt engineering technique with the assistance of few-shot learning to design the system prompt of the question answering system. [31].

3.2.4.2.1 Chain-of-Thought Prompting

CoT prompting helps LLMs handle complex tasks by breaking them down into intermediate reasoning steps. It allows models to tackle problems step-by-step, which improves their ability to reason through complex challenges, and thus is particularly useful for tasks that require detailed reasoning, such as problem-solving, research, and analysis, suiting the features of our SPARQL query generation assignment. With CoT prompting, the model has the potential to maintain a clear line of thought while generating the SPARQL queries based on the user requirement, leading to better performance.

It is essential that the the prompt is designed to be stepwise aligned with the logical thinking process of how human semantic experts write the SPARQL queries for corresponding questions. First of all, the task background should be given regarding the purpose of our question answering system. The LLM is instructed that it is a semantic web expert and it is required to write SPARQL code for input user queries that are formatted in natural language. Since the system is mainly designed for Dutch users and most of the user queries are expected to be in Dutch, the model is also instructed that it masters the Dutch language to better activate its Dutch reasoning ability. Additionally, it is necessary to specify the data source available to the LLM and the methodology the LLM should employ to manipulate the data. Specifically, the LLM will be informed that it has access to the cadastral ontology data that we have processed and stored in the embedding, which is related to the Dutch cadastral system. And the LLM is required to use RAG technique, in which the cadastral ontology data serves as the retrieval data, to perform the SPARQL query generation task. Moreover, to prevent model hallucination, the LLM is instructed to only reply on the given cadastral ontology data, the context of the prompt and the user input to solve the problem, prohibiting the use of other unprovided information, in case the LLM generates answers from its irrelevant pre-trained knowledge. The task introduction of the CoT prompt is demonstrated in Figure 3.6. It also highlights the essence of the CoT prompting with the classical entry "Let's think step by step".

In the step-wise prompt, the LLM is instructed to break down the problem and approach the solution step by step. For a general interpretation and analysis on the input question, the LLM should make sure it understands the linguistic implication of the whole question explicitly, which is the prerequisite of the following steps. Then the prompt should instruct the LLM to identify any relevant geographical location in the question context. This is important because the questions are cadastral-related and are thus usually pertinent to a certain geographical location of the Netherlands. Also to prevent hallucination and make sure all the namespace prefixes and ontology elements used in the generated SPARQL queries actually exist in our ontology, we prompt the LLM that the questions are in Dutch and it should only use the ontology elements and relations we provide in the retrieval data

FIGURE 3.6: Task Introduction of the CoT Prompt

of ontology embedding to perform the task. Since most LLMs still have limited SPARQL ability [110], this step needs to be prompted more specifically to instruct what ontology elements the LLM should utilize and how it can traverse over the ontology data to write queries to connect ontology elements that are steps away for the construction of the correct property paths. We give very direct prompt here to inform the LLM to implement the property paths technique of SPARQL in this step, and also give an analogy with the recursive query of SQL(which is similar to the property paths in SPARQL) to facilitate the LLM's understanding on the utilization of property paths. We are convinced that the pre-trained LLM already have profound knowledge on the syntax of SPARQL and SQL, which will assist its interpretation of our instruction.

In case the LLM hallucinates with the English translation of the local names of the ontology elements(for example sor:Provincie becomes sor:Province)that are supposed to be Dutch cadastral terms in the generated SPARQL queries, which sometimes indeed happens and leads to a mismatch on the local name of the ontology elements of KKG and thus returns no valid results from the query execution, we add in this prompt step to instruct the LLM to name all the variables with only Dutch terms. Although the name of variables doesnt matter with the validity of the SPARQL query, adding this snippet in the prompt does drastically reduce the hallucinations and boost the performance. The reason is yet to be revealed and worth further research and inspection.

With these two steps, the LLM should be able to generate some SPARQL queries snippet based on the given information but they are not complete for execution yet. Furthermore, it is also important to ensure the correct SPARQL syntax in query generation. When a SELECT or a COUNT clause is executed, it is expected to select or count the distinct objects

and avoid duplication. Therefore, the LLM is prompt to use DISTINCT in a SELECT and COUNT clause. The SPARQL queries are not executable without the imperative prefix declarations. So we prompt the LLM to add the necessary prefix declarations on top of the generated query. What prefix declarations to add is based on the elements used in the SAPRQL query and is independently judged and executed by the LLM regarding our prompt.

With these four steps, we ensure the LLM can generate executable SPARQL queries, but still not guarantee that the queries can return plausible results. There are still some question-wise and data-wise details worth considering in the generation process. For example, the label, of the Dutch place that is a geographical instance in KKG, is usually the name of the place with the suffix @nl to represent that the name is in Dutch language. Therefore, the instance "province:0026" denoting the Province of Utrecht is an instance of the class "sor:Provincie" and this instance has a label "Utrecht@nl". The SPARQL query has to state this label with the place name and the @nl suffix to shoot a match to the correct instance. We instruct the LLM about this to handle the label in the query properly. Besides this, it is expected that the users don't always make semantically and grammatically correct questions. They may have typos in the name of places when they input a question. The current question answering system has not yet performed stably against the input with typos. To handle these cases, we instruct the LLM to identify the correct place name and locate the target instance data despite encountering possible typos in the word. The pre-trained knowledge of the LLM can help with this issue. When a question expects the answer to return a specific name of place, the name should be obtained from the label of the instance as well, and thus the query should be designed to ensure that the execution result will contain the target label.

Similarly, polysemy can occur in questions, resulting in ambiguity when multiple instances share the same name designation in their label or local name, and the LLM should be prompted to discern which instance is the most pertinent to the question. For example, "Utrecht" can denote the "Woonplaats(English: place of resident) Utrecht" or the "Gemeente(English: municipality) Utrecht" or the "Provincie(English: province) Utrecht". When "Utrecht" is asked in the question, it should be analyzed based on the question context to interpret what it exactly means to locate the correct instance data. For example, if the user asks "How many hospitals are there in the Province of Utrech?"(Dutch: Hoeveel ziekenhuizen zijn er in de provincie Utrecht?), based on the context, the LLM will recognize that Utrecht in the question refers to the utrecht instance in the province class instead of the one in the municipality class. This issue is also partially caused by the property of the Dutch cadastral system. For many cities in the Netherlands, there are "Place of Residence"(Dutch: woonplaats) and "Municipality"(Dutch: gemeente) with the same name and they can both refer to the city itself, like what the Utrecht example indicates. However, although the two concepts sound quite similar to each other, they refer to different geographical areas of the city, despite the two areas usually having overlapping components. The two concepts denote two different data elements and serve different functions in practice. When the user asks "What is the average surface area of the houses in Utrecht?"(Dutch: Wat is de gemiddelde oppervlakte van de huizen in Utrecht?), it is very ambiguous whether the user wants to know the data of the Place of Resident Utrecht or the Municipality Utrecht. Thus we prompt the LLM that if the user gives a city name without specifying it is a place of resident or a municipality, interpret it as a place of resident by default. But if the user has explicitly indicated that it is a Place of Residence or a Municipality, then it it should be interpreted as referring to the respective instance of the

place of residence or the municipality. This analytical process is explicitly demonstrated with the utrecht example in Figure 3.7.

FIGURE 3.7: The Polysemy of "Utrecht"

Another similar problem is the synonym issue where the users can use multiple words to represent the same concept. For instance, the word "plot" in Dutch is commonly referred to as "perceel", but it can also be represented by the terms 'tuin' or 'kavel' to convey the same concept.". For the term "construction year", which is a property of building(Dutch: gebouw), users can use the Dutch word "bouwjaar" or its synonyms "oorspronkelijk bouwjaar" and "leeftijd" interchangeably to denote this concept in the questions. We integrate a few-shot learning paradigm in the CoT to prompt the LLM to learn to generate the SPARQL queries following the few-shot examples. The few-shot examples are designed based on the thinking process aligned with our CoT prompt. At the beginning, we try to break down the few-shot examples and in pieces and simulate the inference of how each piece of the examples are designed based on the thinking process of the CoT prompt, and we insert the pieces of query snippets in the prompt in the format of plain text. However, this doesn't produce the effect we pursue and the generated SPARQL queries to the new input questions are just invalid parody of the provided examples, performing poorly in the test with zero accuracy.

By themselves, language models can't take actions - they just output text. A big use case for LangChain is creating agents. Agents are systems that use LLMs as reasoning engines to determine which actions to take and the inputs necessary to perform the action. After executing actions, the results can be fed back into the LLM to determine whether more

actions are needed, or whether it is okay to finish. This is often achieved via tool-calling

Then we modify the method to accommodate the few-shot examples into the prompt with the utilization of LangChain agents. The design details of the few-shot learning and the integration of the few-shot examples are further explained in the Few-shot Learning. Embarking on an audit of the current steps we have for the CoT prompting, there is still concern that what if the user input question seeks the cadastral information of a certain location but misses a clear indication of the exact location address. It can result from the lack of proficiency of using a chatbot or can simply result from the recklessness of the user. In this case, we do not want the LLM to enforce to generate any random SPARQL queries and give unreasonable answers. Therefore, it is prompted that it should review the question and independently judge whether the location information provided in the question is sufficient to answer the question. If it is not, then the LLM is not supposed to generate any SPARQL query, but simply answer that it can not help with this question. This is also the case for the input questions that are not cadastral-related. Before the last step, we remind the LLM to review the generated query and check whether all the necessary prefix declarations are complemented property at the top of the query. In the final step, we prompt the LLM to only give the generated query as the output without adding any other context or description. This benefits the automation of the following procedure of processing and executing the output SPARQL queries to obtain the queried results.

This step-by-step design of the CoT prompt is tailored specifically for our Dutch cadastral question-answering task. For the complete version of the prompt, please refer to the Appendix.

3.2.4.2.2 Few-shot Learning

We have tried to integrate our CoT prompting with zero-shot, one-shot, and few-shot learning, respectively. It is identified that both zero-shot learning and one -shot learning perform poorly on the current questions and are unable to marginally surpass the performance of the current question answering system of Loki.

Few-shot learning(FSL) enables GenAI systems to adapt to new tasks or domains with minimal labeled data, addressing the limitations of traditional models that require extensive annotated datasets. FSL is particularly beneficial for domain-specific question answering systems, such as geographical, medical or legal applications [111], where labeled data is scarce, which is a similar situation in our case. Therefore, it is a promising prompt engineering technique for our domain-specific cadastral question answering system. It allows the pre-trained language model to quickly adapt to our specialized cadastral context without requiring costly and time-consuming annotation processes. Furthermore, FSL fosters efficiency by reducing computational overhead associated with retraining models on extensive datasets, making it ideal for the applications of Kadaster since the new cadastral data is input and updated frequently.

We integrate FSL with CoT prompting to utilize the combined power of two state-of-the-art prompt engineering techniques to enhance the question answering. We use a five-shot learning for the implementation. The few-shot examples are designed to align with the thinking process of the CoT prompt to instruct the LLM to generate SPARQL queries with the identical logical expression. One of the few-shot examples adapted to answer the question "How many buildings are there on the street Zandweg in the municipality of

Maasdriel that were built before 1980?" is shown in Figure 3.13. The original question is asked in Dutch. It is identified that the target location of this question is the street Zandweg in the municipality of Maasdriel. From the interrogative word "How many", we understand that the question seeks a number of count, so the query should begin with a SELECT on the COUNT clause. Variables are named in Dutch based on our prompt instruction. The variable **geb** refers to **building(gebouw)** in Dutch and the variable **aantal** refers to the Dutch word for **number**. The duplications are supposed to be eliminated so a DISTINCT clause is needed. The method is to count the number of buildings on the target street, and then filter the buildings to only remain those that were built before 1980. In the prompt, we instruct the LLM to use the property path technique to query the data. Following this approach, we start from the target street Zandweg, an instance with the Dutch label "Zandweg" of the class sor:OpenbareRuimte(English: public space, which means street in KKG). The house number, which is "nummeraanduiding" in Dutch and in short, na as the variable name, constructs a triple relation with the street instance of the class sor:Openbareruimte via the object property sor:ligtAan(English: lies on). This is logical since a house number inherently lies on a street. Therefore, a query snippet can be written as is shown in Query Snippet 1.

```
SELECT (COUNT(DISTINCT ?geb) as ?aantal)
WHERE {{
  ?openbareruimte a sor:OpenbareRuimte;
        skos:prefLabel "Zandweg"@nl;
        ^sor:ligtAan ?na.

  ?na a sor:Nummeraanduiding.
```

FIGURE 3.8: Query Snippet 1

Furthermore, the house number is connected to a residence object(Dutch: Verblijfsobject) of the class sor:Verblijfsobject on a triple relation with the object property sor:hoofdadres(English: main address). It is noted that the term Verblijfsobject in KKG refers to a place with only one independent address, such as an apartment, a house with a single address, or a factory with a single address. It doesn't have to be a place of residence(although that is the literal translation of this Dutch word in English). And a building with multiple addresses is not one Verblijfsobject but contains multiple Verblijfsobject. A place of residence in KKG, on the other hand, is a Verblijfsobect with a living function(woonfunctie) according to the property of the knowledge graph. From a Verblijfsobject, the property path continues forward to sor:Gebouw(English: building) through the object property sor:maaktDeelUitVan(English: is part of). A building(gebouw) has a data property sor:oorspronkelijkBouwjaar(English: original year of construction), through which a building instance possesses a numerical property of its construction year. This value is denoted by the variable bo, an abbreviation of the Dutch term "bouwjaar", meaning construction year in English. It will be used to filter the buildings with the anticipated year of construction. The SPARQL query can thus extend with the following snippet shown in Query Snippet 2. We also add a property geo:hasGeometry into the query of sor:Gebouw to offer more flexibility for building-related property path extension for other pertinent questions.

```
?na a sor:Nummeraanduiding;
  ^sor:hoofdadres ?vbo.

?vbo a sor:Verblijfsobject;
  sor:oppervlakte ?wo;
  sor:maaktDeelUitVan ?geb.

?geb a sor:Gebouw;
  sor:oorspronkelijkBouwjaar ?bo;
  geo:hasGeometry [
    geo:asWKT ?geo_wgs84;
    rdfs:isDefinedBy bag:
  ].
```

FIGURE 3.9: Query Snippet 2

The location of interest in the question is the Municipality of Maasdriel. From the building class sor:Gebouw, we have a property path to step by step connect to the municipality class sor:Gemeente, via the classes of neighborhood(buurt), district(wijk) and another municipality class wbk:Gemeente, as shown in Query Snippet 3. In the end, the Filter clause is applied to select buildings that were built before the year of 1980, as shown in Query Snippet 4.

A thorough check is conducted on the query clause details to add all the necessary prefix declarations on the top of the query. This is designed in Query Snippet 5. In this snippet, we also add some extra prefix declarations that are not necessarily needed in this SPARQL query. This is because the inference of the LLM on checking the prefix declarations are not perfect. It is noted that sometimes it misses one prefix declarations and causes the syntax error of the query. Therefore we add some common-used prefix declarations in the examples to ensure the integrity of the generated SPARQL code. According to the prompt instruction, only the query itself is presented in the answer, without extra descriptive text. Together, the code snippets constitute the complete SPARQL query demonstrated in Figure 3.13 to answer the question "How many buildings are there on the street Zandweg in the municipality of Maasdriel that were built before 1980?". With similar examples, the five-shot learning instructs our LLM to embark on a deeper understanding on how to generate proper SPARQL queries for the questions following the CoT process.

To check the complete version of the five-shot learning examples, including the questions and their query solutions, please refer to the Appendix.

```
?geb a sor:Gebouw;
  sor:oorspronkelijkBouwjaar ?bo;
   geo:hasGeometry [
     geo:asWKT ?geo_wgs84;
     rdfs:isDefinedBy bag:
    ].

?wbk_buurt a wbk:Buurt;
     ^geo:sfWithin ?geb;
      geo:sfWithin ?wbk_wijk.

?wbk_wijk a wbk:Wijk;
     geo:sfWithin ?wbk_gemeente.

?wbk_gemeente a wbk:Gemeente;
     ^owl:sameAs ?gemeente.

?gemeente a sor:Gemeente;
     owl:sameAs ?wbk_gemeente;
     skos:prefLabel "Maasdriel"@nl.
```

FIGURE 3.10: Query Snippet 3

```
FILTER (?bo < "1980"^^xsd:gYear)
```

FIGURE 3.11: Query Snippet 4

```
PREFIX bag: <http://bag.basisregistraties.overheid.nl/def/bag#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX sor: <https://data.kkg.kadaster.nl/sor/model/def/>
PREFIX sor-con: <https://data.kkg.kadaster.nl/sor/model/con/>
PREFIX kad: <https://data.kkg.kadaster.nl/kad/model/def/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
prefix xsd: <http://www.w3.org/2001/XMLSchema#>
prefix owl: <http://www.w3.org/2002/07/owl#>
prefix wbk: <https://data.labs.kadaster.nl/cbs/wbk/vocab/>
```

FIGURE 3.12: Query Snippet 5

Question:
Hoeveel panden staan er aan de straat zandweg in de gemeente Maasdriel die voor 1980 zijn gebouwd?
(English: How many buildings are there on the street Zandweg in the municipality of Maasdriel that were built before 1980?)

Example query solution:

```
PREFIX bag: <http://bag.basisregistraties.overheid.nl/def/bag#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX sor: <https://data.kkg.kadaster.nl/sor/model/def/>
PREFIX sor-con: <https://data.kkg.kadaster.nl/sor/model/con/>
PREFIX kad: <https://data.kkg.kadaster.nl/kad/model/def/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
prefix xsd: <http://www.w3.org/2001/XMLSchema#>
prefix owl: <http://www.w3.org/2002/07/owl#>
prefix wbk: <https://data.labs.kadaster.nl/cbs/wbk/vocab/>

SELECT (COUNT(DISTINCT ?geb) as ?aantal)
WHERE {{
   ?openbareruimte a sor:OpenbareRuimte;
               skos:prefLabel "Zandweg"@nl;
               ^sor:ligtAan ?na.

   ?na a sor:Nummeraanduiding;
      ^sor:hoofdadres ?vbo.

   ?vbo a sor:Verblijfsobject;
      sor:oppervlakte ?wo;
      sor:maaktDeelUitVan ?geb.

   ?geb a sor:Gebouw;
     sor:oorspronkelijkBouwjaar ?bo;
      geo:hasGeometry [
         geo:asWKT ?geo_wgs84;
         rdfs:isDefinedBy bag:
      ].

   ?wbk_buurt a wbk:Buurt;
         ^geo:sfWithin ?geb;
          geo:sfWithin ?wbk_wijk.

   ?wbk_wijk a wbk:Wijk;
         geo:sfWithin ?wbk_gemeente.

   ?wbk_gemeente a wbk:Gemeente;
         ^owl:sameAs ?gemeente.

   ?gemeente a sor:Gemeente;
         owl:sameAs ?wbk_gemeente;
         skos:prefLabel "Maasdriel"@nl.

   FILTER (?bo < "1980"^^xsd:gYear)
}}
LIMIT 9999
```

FIGURE 3.13: Few-shot Example

3.2.4.3 Retrieval-augmented Generation

We briefly introduce how we constructed the RAG chain to facilitate the QA process. We have already set up our system prompt. We have the language model, which is GPT4-32k, for the QA system. It is quite smooth to build the RAG with existing packages. We use the framework RetrievalQA of Langchain to construct the RAG chain. It has built-in functions and methods to receive the user-defined prompt and language model as arguments. For this Confactual QA task, we expect the system to be more deterministic instead of creative, so the temperature is set as 0. More technical details regarding the RAG chain construction are available in the repository of this project.

3.2.5 Evaluation

To evaluate the performance of our question answering system artifact, a systematic validation methodology should be implemented. The key requirement for the designed prototype is that it is able to answer a broader spectrum of cadastral questions accurately within a fair execution time. The validation is expected to validate the accuracy of different aspects of the system.

We utilize three types of questions to validate the designed artifact.

1. Basic Retrieval Questions: Simple queries that can already be answered by Loki at the moment. These questions are directly derived from the existing question-SPARQL pairs of the training set of the current question answering system of Loki(hereafter referred to as the old system). According to Kadaster, the old system is expected to answer these questions well enough. Each question has a corresponding SPARQL query(hereafter referred to as the ground truth query) to query the result from KKG to answer the question. By testing these questions, it is demonstrated that the designed system can at least have equivalent performance on the tasks of the current question-answering system and therefore the designed system's function as a proof of concept is validated.

2. Edge Case Questions: Ambiguous questions, typos, or incomplete information. These questions can be directly adapted from the basic retrieval questions by adding typos or giving ambiguous information in the text. By testing these questions, it is validated that the designed system is resilient against complications, offering a more robust performance facing different contexts compared to the current question-answering system.

3. Novel Retrieval Questions: More complicated cadastral questions, which can not be answered by the current version of Loki but are in line with the interest of the users. By testing these questions, the novelty of the proof of concept is validated and its potential to outperform the current question answering system can be evaluated.

To ensure the robustness of the designed system, each question is tested at least three times against the artifact. This rigorous testing process verifies that the generated queries are consistently correct in both semantic and syntactic aspects, thereby preventing the validation bias from the hallucinated results by the LLM [112].

For basic retrieval questions, we examine the validity of our system from the following perspectives.

1. Query accuracy.

    - Exact Match (EM): Percentage of generated queries that exactly match the ground truth queries.

    - Syntax Validity: Ensure all generated queries are syntactically correct.

2. Output accuracy

    - Semantic Equivalence: Check if the generated query produces the same result as the ground truth query (even if the syntax differs).

- Result Accuracy: Compare the results of the generated query with the expected results.

Aligned with the CRISP-DM circle, we implement PDCA(for Plan, Do, Check, Act) [113], a continuous improvement process for the iterative development and improvement of our prototype in the evaluation process. We set up the objective of our experiment and conduct the experiment by testing the questions on our new system for evaluation. As our evaluation goes on, we may encounter results that do not meet the expected outcomes and indicate system flaws. We continue to improve the system based on the revealed issues and iterate the testing from the beginning. We continue this iterative development and improvement process until we effectively rectify solvable system flaws in the prototype so that it is deliverable to verify the validity of our KGQA approach.

The PDCA process is implemented iteratively in four stages as follows and is demonstrated in Figure 3.14:

1. Plan

   We define the objective of our continuous improvement process and the necessary steps to deliver the results.

2. Do

   We implement the testing and obtain the execution results for analysis.

3. Check

   We evaluate the results in terms of system performance. The data and results obtained from the implementation phase are systematically assessed. The collected data is compared to the expected outcomes to identify similarities and differences.

4. Act

   We progress on process improvement by utilizing records from the "do" and "check" phases to identify and analyze system deficiencies. The root causes of these deficiencies are systematically investigated, identified, and addressed through system modifications. Depending on the results, we can decide to standardize and stabilize the changes or begin the cycle again. Planning for the next cycle should proceed with a better baseline. Work in the next do phase should not create a recurrence of the identified deficiencies. If such issues persist, the corrective actions are ineffective.

The evaluation results are further discussed in Chapter 4.

3.2.6 Deployment

We use GPT-4 32k as the LLM for this task. The embedding model of the designed question answering system is text-embedding-3-large[1]. Since it is a SPARQL generation task, the question answering is supposed to be more coherent and less creative. Therefore, the temperature of the question answering system is set as zero. So far, the system has only generated the SPARQL query without executing the query. We build an automated

[1] https://platform.openai.com/docs/guides/embeddings

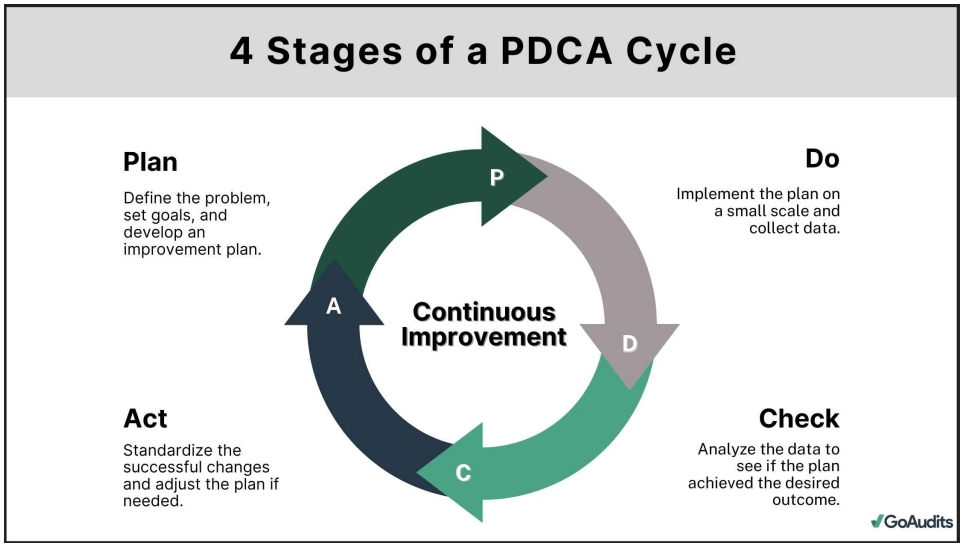

FIGURE 3.14: The PDCA Cycle

process to obtain the query output and execute it via the API of KKG and gain the execution result. By integrating this process with the query generator, we have a complete process for a question answering system demo.

The designed question answering system only serves as a proof of concept for our idea of integrating ontology embedding and prompt engineering to enhance knowledge graph question answering, based on the case of KKG. Therefore, this project only works as a demo. It is not fully ready for a commercial deployment in an industrial environment. For a large-scale deployment, further research and analysis are needed to improve the performance and the user experience of such an artifact. This will be further discussed in future research.

Chapter 4

Result and Discussion

We conducted the experiment between November 21 and December 4, 2024. This section presents the results of the experiment and our discussion on the implications of the results. For basic retrieval questions, thirty questions were evaluated using our newly developed question-answering system (hereafter referred to as the new system). These questions were sourced from six distinct categories within the question sets of the old system of Loki. The six question categories are brt closest query, brt filter query, location query, filter query, property query and top 10 nl query respectively. More elaboration of the categories can be found in Table 4.1. Within one category, there are many similar questions, which are paraphrased and asked in different formulations, but pursue the same SPARQL queries and output answers. Therefore, selecting five random questions from each category is deemed sufficient to represent the corresponding category. For our evaluation, we randomly select five questions from each of the six categories, resulting in an evaluation question set of thirty questions.

The evaluation of basic retrieval questions are to assess whether the new system can fulfill the fundamental functions of the old system. If the evaluation outcome is positive, the initial potential of replacing the old system with the new system to boost the performance of Loki is verified.

4.1 Basic Retrieval Question Results

We input the questions into the new system to obtain the generated SPARQL query snippets. Then we execute the queries through an API of the query executor of KKG to obtain the query results, which will be the answers to the questions. We carry out our validation based on query accuracy and output accuracy, according to our evaluation approach.

4.1.1 Query Accuracy

For query accuracy, we evaluate with three metrics: Exact Match(EM), Syntax Validity and Semantic Equivalence.

Question Category	Total Number of Questions	Category Details
brt closest query	46	This category contains questions regarding the distance from the address of interest of the user to a specific brt property.
brt filter query	173	This category contains questions regarding the characteristics of a user-specified brt property that meets a certain condition filter.
location query	33	This category contains questions regarding a certain type of buildings or plots in a user-specified location granularity.
filter query	18	This category contains questions regarding buildings that meet a filter of their time of construction in a user-specified location.
property query	55	This category contains questions regarding a location granularity specified by a user-given address.
top 10 nl query	20	This category contains questions regarding the brt properties in TOP10NL, a topographic dataset that provides detailed geographic information at a 1:10,000 scale and serves as a foundation for topographic maps.

Table 4.1: The Basic Retrieval Question Category

4.1.1.1 Exact Match

This metric quantifies the proportion of generated queries that exactly match the ground truth queries. Since the uniqueness of the CoT process and the example queries in the few-shot learning are designed to complement and reinforce each other and they perform in a way distinctly different from the question-SPARQL pairs of the old system, the hypothesis is that the generated queries of the new system are very unlikely to be the same as the ground truth queries of the old system syntactically. The experiment have also proved the hypothesis to be correct. Not a single query of the new system can exactly be identical to the ground truth query in syntax. However, it is noted that we do not have to require the generated queries to match the ground truth queries exactly in all syntactic details. For the same question, the generated query proves correct if it does not contain any syntactic errors and can return the same result as the ground truth queries from the execution. Therefore, EM does not matter in this context and we will not focus on verifying it in the following evaluations.

4.1.1.2 Syntax Validaity

This metric measures whether the generated queries of the new system are correct in SPARQL syntax. By executing a query with the query executor of KKG, the query with syntactic errors will raise and return the syntax error warning. The evaluation has demonstrated that, for all basic retrieval questions, the new system consistently generates queries that adhere to the correct SPARQL syntax. Consequently, this validates the system's robustness in producing syntactically correct SPARQL queries.

4.1.2 Output Accuracy

Once the query is syntactically correct, it can be executed successfully through the API of KKG and return a valid answer. We need to verify if this answer output is the accurate output we expect. For output accuracy, we evaluate with two metrics: Semantic Equivalence and Result Accuracy.

4.1.2.1 Semantic Equivalence

This metric measures whether the query generated by the new system will return the same query result as the ground truth query for a given question. Via thirty questions evaluated in the new system, we obtained thirty generated queries. Within these thirty queries, twenty-two of them return the same results as the ground truth queries. Among the rest of the eight queries, four of them return the correct results but these four questions can not be answered correctly by the old system. For three of them, no query is returned from the old system and the remaining one returns a zero answer with an empty JSON array. This is a system flaw because the old system is supposed to be able to answer these questions. One query returns a result different from the ground truth query but the result are justifiable to be correct. One query can return the correct result but the performance of the new system on this question is unstable. Hallucinations happened two times within three consecutive tests and generated incorrect SPARQL queries. We consider it as a failure to pass the hallucination test and conclude that our new system cannot answer this question, although it is sometimes correct. This question cannot be answered by the old system as well so both results are incorrect. Two queries return the incorrect results while the ground truth queries of the old system return correct results for these two questions.

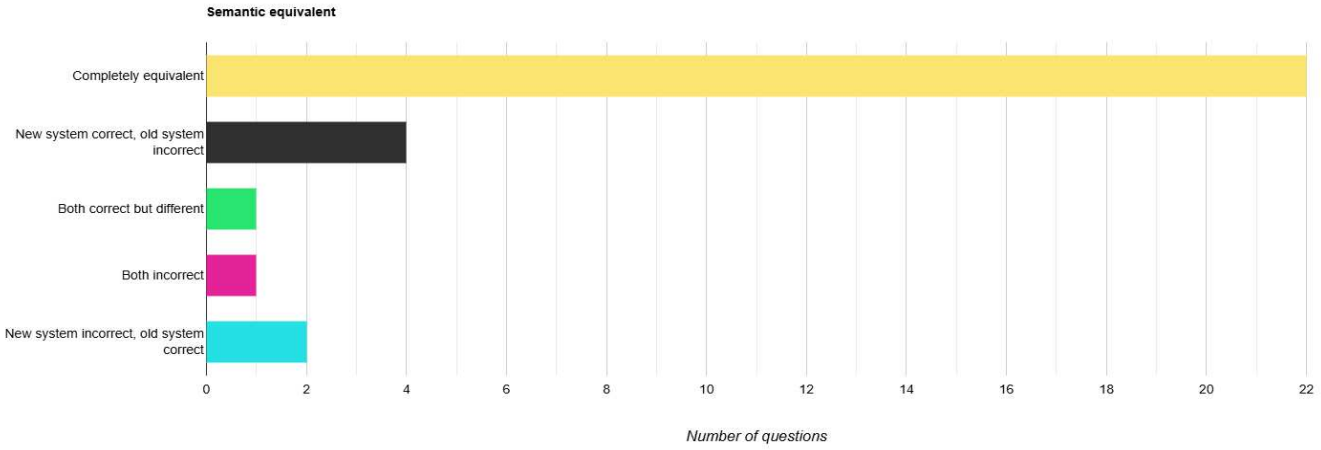

FIGURE 4.1: Semantic Equivalence Results of Basic Retrieval Questions

4.1.2.2 Result Accuracy

We have compared the equivalence of the results with the results from the old system. In most cases, the results from the old system are the expected results. However, some questions are open to more than one possible answer, or the questions cannot be answered by the old system. Therefore, in addition to semantic equivalence of the results, we also need to verify the result accuracy of each question for the new system. This metric evaluates the accuracy of the results from the generated queries by comparing them to the expected results. If a result from the new system is equivalent to the result from the old system for a given question, the result and the SPARQL query are considered accurate. In cases where the old system is unable to resolve the questions, we examine the details of our generated queries and the structure of KKG to verify the integrity of these results. In total, we have twenty-seven accurate queries out of thirty, with the other three being inaccurate ones.

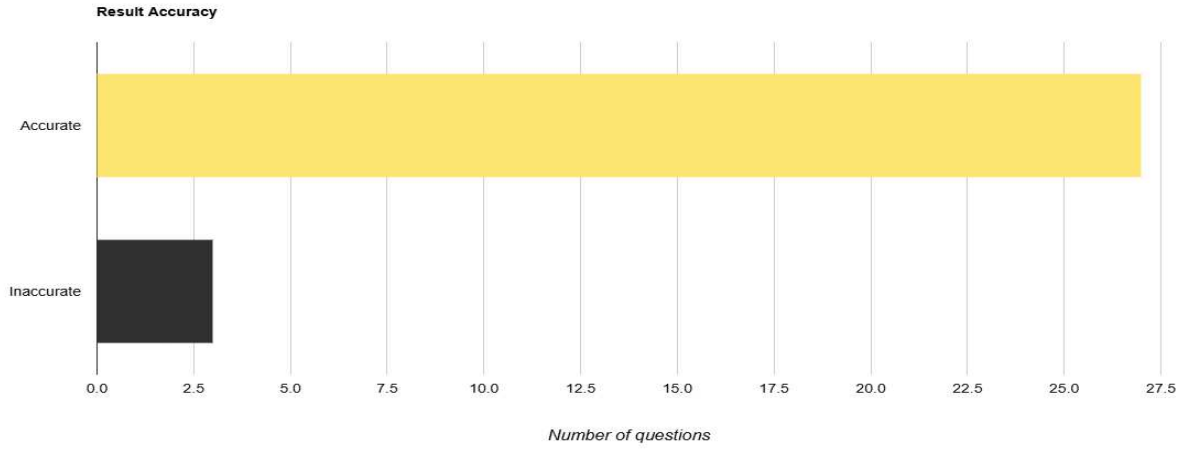

FIGURE 4.2: Result Accuracy of Basic Retrieval Questions

4.2 Edge Case Question Question Results

We evaluate how well the new system handles the edge cases, such as ambiguous information or typos in the question text. Ambiguous information and typo cases are not exclusive to each other as a question can be ambiguous because of typos. In general, the edge case question refers to questions that are not articulated in an unambiguous and Dutch-grammatically correct way.

We have tested five questions regarding edge cases. We primarily generate edge case questions by modifying the wording or phrases of basic retrieval questions that our new system can answer accurately. For example, since cadastral questions are very likely related to a certain location or address, we add typos in the location names. The experiment shows that, if we replace the correct location name "Enschede" with a typo "Enshcede" in the question, the new system can still return the accurate answer pertinent to the Dutch municipality Enschede while the old system cannot answer the question anymore. Similar experiment has been conducted on other Dutch municipalities such as Hengelo and Zwolle, and we have perceived the same situation. An exception we observe is Amsterdam. Even with typos in the spelling of Amsterdam, the old system can answer the questions correctly as the new system. All the tested edge case questions and their original questions can be viewed in Table 4.2.

4.2.1 Query Accuracy

All the queries generated by the new system against the edge case questions are syntactically valid. There are no syntax errors in these queries. However, except for the question "How many buildings are there in Amsterdam that were built between 1980 and 1985?", the old system cannot generate an accurate query against the edge cases we test, or generate invalid queries. The invalid queries generated by the old system are usually syntactically correct, but semantically inaccurate.

4.2.2 Output Accuracy

The queries of the new system are all executed successfully and return non-empty answers to all the tested edge case questions. We compare the answers with the answers of the selected basic retrieval questions, from which our edge case questions are adapted. They are the same answers. Therefore, they meet the requirements of both semantic equivalence and result accuracy. The old system can sometimes generate syntactically correct queries to edge cases but these queries are semantically inaccurate so that they only return empty answers from the execution. Hence, in terms of semantic equivalence, only the answers to the Amsterdam-related question are completely equivalent in the new system and the old system. Regarding the rest of the four edge case questions, the new system's answers are correct, whereas the old system's answers are not. This is demonstrated in Figure 4.3. As for result accuracy, all the answers of the new systems are accurate as presented in Figure 4.4.

The performance gap can be attributed to the reasoning abilities of different language models. One one hand, the GPT-4-32k model we utilize on the new system has a strong reasoning ability. The pre-trained LLM already possesses a substantial knowledge base before we insert a CoT prompt to instruct it. And we have also included typo handling in our CoT prompt. Together with in-context learning, the LLM is able to catch the accurate interpretation of the question, even if there are typos on certain words. This robust

Edge Case Question	English Translation	Edge Case Type
Hoe hoog is de afstand tot de ziekenhuis (van mijn adres Hengelosestraat 100, Enshcede)?	How far is the hospital (from my address Hengelosestraat 100, Enschede)?	Typo. "Enshcede" should be "Enschede".
Wat voor kerken zijn er dichtbij vanaf Deldenerstraat 134, Henglo?	What churches are close to Deldenerstraat 134, Henglo?	Typo. "Henglo" should be "Hengelo".
Welke politiebureau is het dichstbij vanaf Grote Markt 18, Zwoole?	Which police station is closest from Grote Markt 18, Zwolle?	Typo. "Zwoole" should be "Zwolle". "dichstbij" should be "dichtstbij".
Geef me alle woningen in Utrecht.	Give me all the houses in Utrecht.	Ambiguous information. "Utrecht" can refer to the province, the municipality or the place of residence of the same name. If not specified, we define it to refer to the place of residence, namely "woonplaats" in Dutch.
Hoeveel gebouwen zijn er in Ammsdetam die tussen 1980 en 1985 zijn gebouwd?	How many buildings are there in Amsterdam that were built between 1980 and 1985?	Typo. "Ammsdetam" should be "Amsterdam".

Table 4.2: Edge Case Questions

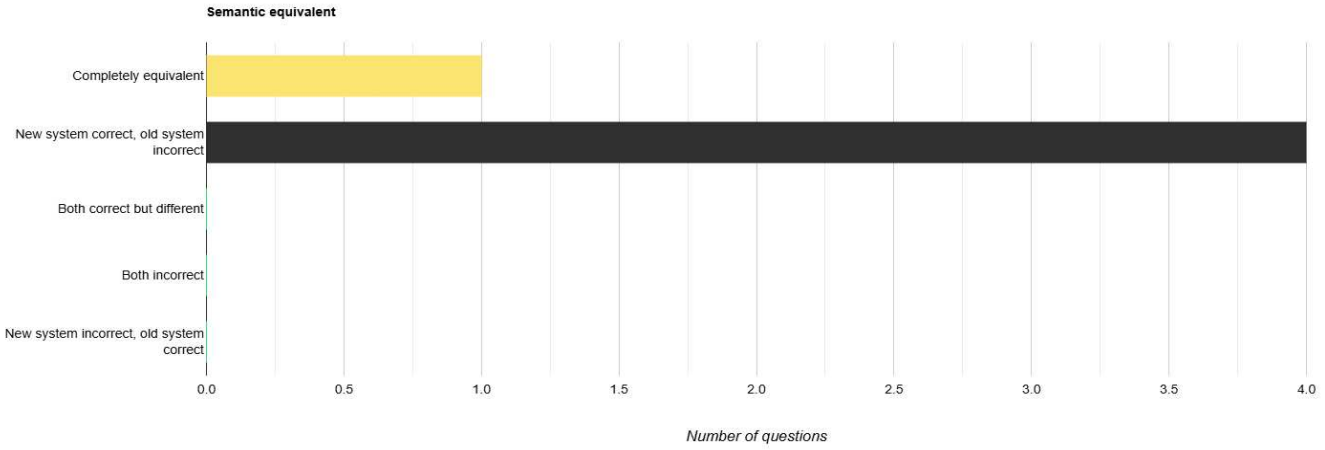

FIGURE 4.3: Semantic Equivalence Results of Edge Case Questions

reasoning capability of LLMs is also observed by Saporov and He [114]. On the other hand, the old system is using a T5 model, which has a relatively limited reasoning capability. In addition, the old system's training set with the fixed combination of the question-SPARQL pairs may not be as resilient in handling typos or ambiguous information in the question as CoT prompting.

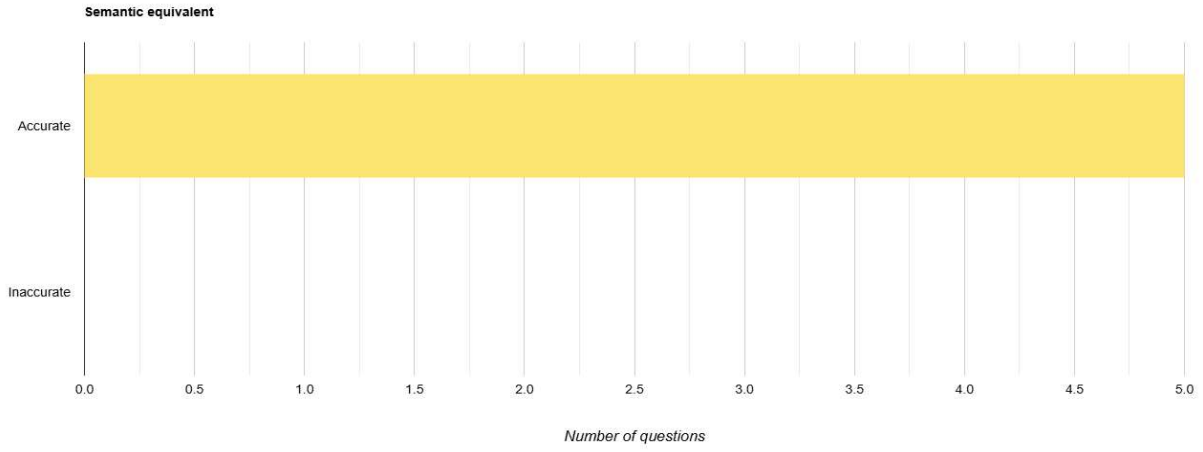

FIGURE 4.4: Result Accuracy of Edge Case Questions

4.3 Novel Retrieval Questions and New Features

We have also tested new questions aligned with the interest of Kadaster against our new system. These questions can not be answered by the old system yet. We will also discuss some new features that we develop on the new system. These new features not only facilitate flexible customizations of the questions but also strengthen the performance stability compared to the old system. With these features, the new system is enabled to answer a wider range of questions according to the needs of users.

The tested new questions can be seen in Table 4.3. There are more new questions to be answered, because technically our ontology embedding methodology, in combination with the CoT prompting and the strong reasoning ability of GPT-4-32k, should enable the new system to answer many new questions that can be concatenated from the property paths, which means almost any questions relevant to the classes, properties, and selected individuals that are on the interconnected property paths. However, we cannot enumerate all the possible new questions, so we just utilize the following new questions as representative questions for testing, because the answers to these types of questions are helpful to the daily work of Kadaster or other public stakeholders but the current methods to retrieve this information are more time-consuming.

Original Question	English Translation	Answer
Welke gemeente heeft de meest verblijfsobjecten met kantoorfunctie in de provincie Gelderland?	Which municipality has the most residential objects with office function in the province of Gelderland?	Arnhem
Welke gemeente heeft de meest openbare ruimte in de provincie Overijssel?	Which municipality has the most public space in the province of Overijssel?	Enschede
Wat is de gemiddelde oppervlakte van huizen (met woonfunctie) in Apeldoorn?	What is the average surface area of houses (with residential function) in Apeldoorn?	116.646366977664688
Welke woonplaats heeft de meeste gebouwen die voor 1700 zijn gebouwd?	Which place of residence has most buildings built before 1700?	Amsterdam
Hoeveel gebouwen zijn er in Amsterdam en Rotterdam die voor 1800 zijn gebouwd?	How many buildings are there in Amsterdam and Rotterdam that were built before 1800?	5130
Geef me de top 3 woonplaats met de meeste gebouwen die voor 1700 zijn gebouwd.	Give me the top 3 places of residence with the most buildings built for 1700.	Amsterdam, Utrecht, Leiden

TABLE 4.3: Novel Retrieval Questions

For the novel retrieval questions, we do not evaluate with the matrix we use for basic questions and edge case questions, because the old system can not answer these questions anymore. We validate the new system by comparing the results with the knowledge graph data. The full queries of the tested novel retrieval questions can be found in the Appendix.

As for new features, our new system allows questions to be expressed in more flexible ways. In the one-to-one training-based old system, only questions in the fixed question set can be answered. With the new system, we do not have to design the questions manually with a large amount of labor, as the reasoning ability of the LLM has drastically broadened the spectrum of answerable questions. More examples can be seen in Table . For example, in the question "How many buildings are there in {city} that were built {filter}{year}?"(Dutch: Hoeveel gebouwen zijn er in {city} die {filter}{year} gebouwd?), the old system can only accommodate conventional ways of asking the question in the pre-defined question structure, such as "How many buildings are there in Rotterdam that were built before 1900?"(Dutch: Hoeveel gebouwen zijn er in Rotterdam die vóór 1900 zijn gebouwd?) or "How many buildings are there in Amsterdam that were built between 1980 and 1985?"(Dutch: Hoeveel gebouwen zijn er in Amsterdam die tussen 1980 en 1985 zijn gebouwd?). The "between"(Dutch: tussen) in the question by default does not include the data of the boundary year in the old system, which are the year 1980 and 1985 in the case. In the new system, it includes the boundary years by default, and we can flexibly customize whether to include the boundary or not by specifying "including" or "excluding" in the question, such as "How many buildings are there in Amsterdam that were built between 1980 and 1985, excluding 1980 and 1985?"(Dutch: Hoeveel gebouwen zijn er in Amsterdam die tussen 1980 en 1985 zijn gebouwd, exclusief 1980 en 1985?). Or we can specify to include one boundary and exclude the other. We can also ask the question in different rephrasings thanks to the strong reasoning ability of the LLM, which is not possible in the old system if these rephrasings are not included in the training set. We can specify more wanted details in the question, so that the query will return more information. For example, in the question "What are the buildings in The Hague built after 1990?"(Dutch: Wat zijn de panden in 's-Gravenhage met bouwjaar na 1990?), we can try to ask the system to return more property details of the filtered buildings according to our needs by adapting the question to "What are the buildings in The Hague built after 1990? Also provide the year of construction, surface area and wgs84-coordinates of these buildings."(Dutch: Wat zijn de panden in 's-Gravenhage met bouwjaar na 1990? Geef ook de bouwjaar, oppervlakte en wgs84-coördinaten van deze panden.). By adding more details, we obtain the answer including the year of construction, surface area and wgs84 coordinates of these buildings. We can also access to the information of other building properties by further customizing the question.

In addition to facilitating question customization, the new system also exhibits robust multilingual capabilities. Although we presume most of the users of Loki will be in the Netherlands, which implies that the questions will be mainly asked in standard Dutch, the new system accommodates more languages besides Dutch. In our limited experiment, it demonstrates a consistently high proficiency in answering questions asked in Frisian, which is another language in the Netherlands, and also the official language of the Dutch province of Friesland. It can also answer questions in English and German. The multilingual feature is partly thanks to the multilingual reasoning ability of the pre-trained LLM, and partly thanks to our instruction in the prompt. We ensure in the CoT prompt that for any input questions related to Dutch cadastre, the new system can progress the inference to the accurate ontology and instance elements in KKG, which are mainly formulated in standard Dutch. This pattern in the prompt was initially designed to render the new system more fault-tolerant to typos and grammatical errors in the questions and not to infer the questions to irrelevant or non-existent knowledge graphs. Enhanced by the built-in multilingual reasoning ability of GPT-4-32k, its ability to process Dutch

cadastral questions presented in multiple languages is significantly improved. As a result, the new system accommodates a broader range of users from diverse cultural backgrounds by supporting multiple languages. And this is achieved simply through proper prompt engineering. No costly retraining on the language model is needed.

The new system also exhibits a consistently stable performance to, which is superior to the old system. The old system performs very unstably in answering some basic questions of certain municipalities, such as Nijmegen, Ede and Zwolle. The reason why the old system has a problem with answering questions related to these municipalities is unknown. But the new system has no problem in solving questions of these municipalities and the data of these municipalities are in KKG. This implies that there can be errors in the training process of the old system. The new system demonstrates stable performance on the basic retrieval questions across difference instances while the old system has difficulty in certain instances in the knowledge graph. This proves that the new system demonstrates a more robust performance than the old system.

Chapter 5

Conclusion

This chapter presents the conclusion of our research project by recapping and answering the core research question and the subquestions. In this research, we formulated the KGQA problem based on the current gap in the domain and the bottlenecks of KGQA in the practical scenario of Kadaster. We conducted a bibliometric survey to complete a systematic review of the literature of the prior research in the relevant domains. Based on the study of the related work, we designed our methodological framework and experimental strategy with a CRISP-DM cycle. Then we implemented the prototyping and experiment. Intertwined with the prototyping, we conducted a systematic evaluation on the experimental results to continue improving the performance and prove the validity of our proposed approach. We will illustrate our critical findings and elaborate on the prospects for the future work of this research.

5.1 Recapping the Research Questions

According to the current situation and bottlenecks faced by the KGQA system at Kadaster, we outlined the following core research questions and subquestions to be solved. We will present the answers to our subquestions one by one grounded in our critical findings and together they constitute the answer to the core research question.

Given the current challenges in Kadaster Knowledge Graph(KKG), how can we utilize ontology embedding and CoT prompting to build a resilient RAG system to solve a KGQA problem?

1. How can the ontology elements of KKG be accurately aligned with relevant natural language questions using ontology embeddings?

2. How can we design effective RAG and prompts with an LLM to generate accurate SPARQL queries against the knowledge graph?

3. How can the SPARQL queries generated in the previous step, along with the corresponding questions, be utilized to improve the KGQA system?

Answering SQ1: How can the ontology elements of KKG be accurately aligned with relevant natural language questions using ontology embeddings? We pre-

pared the ontology data that was suitable for our task in Data Preparation. We have extracted the ontology concepts from the ontology of KKG based on a designed schema and embedded the concepts into vector representations. These concepts encompass the key information derived from the domain ontologies, vocabularies and taxonomies and how these entities are interrelated in KKG. We designed a property path schema to illustrate the triple relations between different entities for better query inference. We have implemented the same embedding model for the word embedding of the questions in the question answering system to ensure that the questions will be accurately aligned with relevant ontology elements of KKG based on their semantic similarity in the identical vector space. We elaborate the details of this part in Ontology Embedding in Chapter 3.

Answering SQ2: How can we design effective RAG and prompts with an LLM to generate accurate SPARQL queries against the knowledge graph? We have constructed a retrieval dataset by storing the ontology embedding we created in SQ1 in a vector database. We utilized this dataset as the retrieval data of the RAG system and designed a customized CoT prompt specialized in our task. We integrated a few-shot learning section in the CoT prompt to better guide the RAG system to generate accurate SPARQL queries. We carefully designed the learning examples of the few-shot learning to align them with the CoT process to achieve a synergetic effect. For the language model, we have used the LLM(GPT-4-32k) aligned with our embedding model. We formulate the experimental details of this section in Prompt Engineering in Chapter 3.

Answering SQ3: How can the SPARQL queries generated in the previous step, along with the corresponding questions, be utilized to improve the KGQA system? We have streamlined a process to execute the SPARQL queries generated from SQ2 automatically with the API of a built-in SPARQL executor of the KKG and return the execution results as the answers to the questions. We have evaluated the SPARQL accuracy and answer accuracy to validate the feasibility of our approach. We have also tested novel questions based on the combination of the existing schemas of KKG on our system to verify its adaptivity and robustness to improve and replace the current KGQA system at Kadaster. The experimental results and the discussion of the evaluation are presented in Chapter 4.

5.2 Contribution

We have had an overview on the needs of ease-of-use knowledge retrieval at Kadaster and performed an audit on its current KGQA system. The current system is built with training on a manually-designed question-SPARQL set. We have identified that the performance of the current approach is too limited to fulfill the expectation of sufficient knowledge retrieval function because of its lack of inference ability and failure in solving questions from schema combinations. We have proposed the idea of establishing a new system based on a new approach combining ontology embedding and CoT prompting to enhance the question answering system. And we have designed a prototype to evaluate the approach and validate the feasibility of our idea on the KKG. Via the validation, we have verified that it is possible to improve the quality of KGQA by clearly presenting the knowledge graph schema with the ontology and strengthening the reasoning of the language model.

5.3 Future Research

This thesis has researched validating the idea of utilizing ontology embedding and CoT prompting to enhance a KGQA system based on KKG. For the prototyping, we have only developed a backend for the question answering system, which is sufficient to verify the feasibility of this idea. But for the implementation in a production environment, we have to make the system ease-of-use and provide potential users with a comfortable user experience. A user-friendly frontend should therefore be further developed for and integrated with our KGQA system and the Kadaster infrastructure in future research. More work like dockerization can be done to further develop the prototype into a software application for stable deployment and distribution. Moreover, answering one question takes the new system around 10-25 seconds, depending on the question's complexity and the performance of the cloud infrastructure. More optimization work in system build-up can be done to shorten the time.

Since this is a KGQA system specialized in the cadastral domain, a majority of the questions are location-related, which requires the user to give a location or an address of interest in the question input(which can also be viewed in the evaluated question set in the Appendix). In our designed system, the method of handling the input location tp implement a direct location inference on the knowledge graph via the generated SPARQL query. To ensure the query details are aligned with the knowledge graph structure, we give concrete instruction to the LLM in the few-shot learning examples. As a result, a limitation bound to this feature is that it strictly requires the user to enter the location in the question in a fixed format, which is the same format as the location registered in the Dutch cadastral system. An example is that for an address containing the Dutch province of North Holland, whose legislative name in the Dutch cadastral system is "Noord-Holland" with a dash between two words. If the users ask a question with this address, they have to enter "Noord-Holland" to refer to the province. Entering "Noord Holland" or "Noordholland" without the dash, or an uncapitalized form will not work. This also applies to other levels of location. We expect the system to be more flexible and not limit the users too much on this aspect in the future. A better way to implement this is to use a location server, allowing the location of interest to be auto-detected from the question and matched to the pertinent location in KKG even if the entered spelling is a bit different. This is supposed to be further improved in future research. Furthermore, higher knowledge graph quality will also improve the KGQA. As we utilize ontology embedding and knowledge graph schema inference as the foundation of KGQA, a more complete knowledge graph will facilitate better inference and reasoning in the knowledge graph. At the moment Kadaster Knowledeg Graph still suffers from incomplete knowledge graph elements and relations, which is a common issue in the domain of knowledge graph and knowledge base. It requires a lot of work on data preprocessing before ontology embedding. KKG also accommodates different semantic web languages such as RDF(which provides the basic structure for representing and linking data in the triple formats), OWL(which defines the ontology and enable inference and relationships) and SHACL(which validates RDF data against predefined rules to ensure it is structured correctly.). The ontology embedding method, along with other ontology embedding or knowledge graph embedding approaches in prior research, can not be directly applied to the SHACL data, which is still a constraint in the domain and is worth further research. Above all, the validation of the proof-of-concept was only conducted in KKG. Future research should explore its applicability to other KGs such as DBpedia and Wikidata to fully harness its potential.

Reference

[1] Yunshi Lan et al. “A survey on complex knowledge base question answering: Methods, challenges and solutions”. In: *arXiv preprint arXiv:2105.11644* (2021).

[2] P Russel Norvig and S Artificial Intelligence. “A modern approach”. In: *Prentice Hall Upper Saddle River, NJ, USA: Rani, M., Nayak, R., & Vyas, OP (2015). An ontology-based adaptive personalized e-learning system, assisted by software agents on cloud storage. Knowledge-Based Systems* 90 (2002), pp. 33–48.

[3] Yunshi Lan et al. “Complex knowledge base question answering: A survey”. In: *IEEE Transactions on Knowledge and Data Engineering* (2022).

[4] Yu Gu et al. “Beyond IID: three levels of generalization for question answering on knowledge bases”. In: *Proceedings of the Web Conference 2021*. 2021, pp. 3477–3488.

[5] Yiheng Shu and Zhiwei Yu. “Distribution Shifts Are Bottlenecks: Extensive Evaluation for Grounding Language Models to Knowledge Bases”. In: *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop*. 2024, pp. 71–88.

[6] Donghan Yu et al. “Decaf: Joint decoding of answers and logical forms for question answering over knowledge bases”. In: *arXiv preprint arXiv:2210.00063* (2022).

[7] Saiping Guan et al. “Shared embedding based neural networks for knowledge graph completion”. In: *Proceedings of the 27th ACM international conference on information and knowledge management*. 2018, pp. 247–256.

[8] Apoorv Saxena, Adrian Kochsiek, and Rainer Gemulla. “Sequence-to-sequence knowledge graph completion and question answering”. In: *arXiv preprint arXiv:2203.10321* (2022).

[9] Philipp Christmann et al. “Look before you hop: Conversational question answering over knowledge graphs using judicious context expansion”. In: *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 2019, pp. 729–738.

[10] Pavan Kapanipathi et al. “Leveraging Abstract Meaning Representation for Knowledge Base Question Answering”. In: *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. Ed. by Chengqing Zong et al. Online: Association for Computational Linguistics, Aug. 2021, pp. 3884–3894. DOI: 10.18653/v1/2021.findings-acl.339. URL: https://aclanthology.org/2021.findings-acl.339.

[11] Kadaster. *Het Kadaster registreert, beheert en faciliteert - Kadaster.nl particulier — kadaster.nl*. https://www.kadaster.nl/over-ons/het-kadaster/wat-doet-het-kadaster. [Accessed 14-03-2024].

[12] Kadaster. *About us - Kadaster.nl particulier — kadaster.nl*. https://www.kadaster.nl/about-us. [Accessed 14-03-2024].

[13] Kadaster. *Kadaster Labs — labs.kadaster.nl*. https://labs.kadaster.nl/. [Accessed 05-03-2024].
[14] Kadaster. *Alles over de BAG - Kadaster.nl zakelijk — kadaster.nl*. https://www.kadaster.nl/zakelijk/registraties/basisregistraties/bag. [Accessed 14-03-2024].
[15] Kadaster. *Waar bestaat de BRK uit? - Kadaster.nl zakelijk — kadaster.nl*. https://www.kadaster.nl/zakelijk/registraties/basisregistraties/brk. [Accessed 14-03-2024].
[16] Kadaster. *Geschiedenis Kadaster; overzicht jaartallen en gebeurtenissen - Kadaster.nl particulier — kadaster.nl*. https://www.kadaster.nl/over-ons/het-kadaster/geschiedenis/mijlpalen. [Accessed 14-03-2024].
[17] *Basisregistratie Topografie (BRT) - Kadaster.nl zakelijk — kadaster.nl*. https://www.kadaster.nl/zakelijk/registraties/basisregistraties/brt. [Accessed 15-03-2024].
[18] Kadaster. *Basisregistratie Grootschalige Topografie (BGT) - Kadaster.nl zakelijk — kadaster.nl*. https://www.kadaster.nl/zakelijk/registraties/basisregistraties/bgt. [Accessed 14-03-2024].
[19] Kadaster. *Loki in het algoritmeregister - Kadaster.nl particulier — kadaster.nl*. https://www.kadaster.nl/over-ons/beleid/algoritmeregister/loki. [Accessed 05-03-2024].
[20] Kadaster. *Kadaster Labs — labs.kadaster.nl*. https://labs.kadaster.nl/cases/loki. [Accessed 05-03-2024].
[21] Guendalina Caldarini, Sardar Jaf, and Kenneth McGarry. "A literature survey of recent advances in chatbots". In: *Information* 13.1 (2022), p. 41.
[22] Colin Raffel et al. "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". In: *Journal of Machine Learning Research* 21.140 (2020), pp. 1–67. URL: http://jmlr.org/papers/v21/20-074.html.
[23] Alec Radford et al. "Improving language understanding by generative pre-training". In: (2018).
[24] Hugo Touvron et al. "Llama: Open and efficient foundation language models". In: *arXiv preprint arXiv:2302.13971* (2023).
[25] Gemini Team et al. "Gemini: a family of highly capable multimodal models". In: *arXiv preprint arXiv:2312.11805* (2023).
[26] Víctor Gutiérrez-Basulto and Steven Schockaert. "From Knowledge Graph Embedding to Ontology Embedding? An Analysis of the Compatibility between Vector Space Representations and Rules". In: *International Conference on Principles of Knowledge Representation and Reasoning*. 2018. URL: https://api.semanticscholar.org/CorpusID:51935922.
[27] Kathrin Blagec et al. "A curated, ontology-based, large-scale knowledge graph of artificial intelligence tasks and benchmarks". In: *Scientific Data* 9.1 (2022), p. 322.
[28] Fatima N Al-Aswadi, Huah Yong Chan, and Keng Hoon Gan. "From ontology to knowledge graph trend: ontology as foundation layer for knowledge graph". In: *Iberoamerican Knowledge Graphs and Semantic Web Conference*. Springer. 2022, pp. 330–340.
[29] Niel Chah. "OK Google, What Is Your Ontology? Or: Exploring Freebase Classification to Understand Google's Knowledge Graph". In: *arXiv preprint arXiv:1805.03885* (2018).

[30] Patrick Lewis et al. “Retrieval-augmented generation for knowledge-intensive nlp tasks”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 9459–9474.

[31] Jason Wei et al. “Chain-of-thought prompting elicits reasoning in large language models”. In: *Advances in neural information processing systems* 35 (2022), pp. 24824–24837.

[32] Yu Gu et al. “Knowledge base question answering: A semantic parsing perspective”. In: *arXiv preprint arXiv:2209.04994* (2022).

[33] Kangqi Luo et al. “Knowledge base question answering via encoding of complex query graphs”. In: *Proceedings of the 2018 conference on empirical methods in natural language processing*. 2018, pp. 2185–2194.

[34] Apoorv Saxena, Aditay Tripathi, and Partha Talukdar. “Improving multi-hop question answering over knowledge graphs using knowledge base embeddings”. In: *Proceedings of the 58th annual meeting of the association for computational linguistics*. 2020, pp. 4498–4507.

[35] Weiqiang Jin et al. “Improving embedded knowledge graph multi-hop question answering by introducing relational chain reasoning”. In: *Data Mining and Knowledge Discovery* 37.1 (2023), pp. 255–288.

[36] Lihui Liu et al. “Joint knowledge graph completion and question answering”. In: *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*. 2022, pp. 1098–1108.

[37] Yunqi Qiu et al. “Stepwise reasoning for multi-relation question answering over knowledge graph with weak supervision”. In: *Proceedings of the 13th international conference on web search and data mining*. 2020, pp. 474–482.

[38] Aleksandr Perevalov et al. “Knowledge graph question answering leaderboard: A community resource to prevent a replication crisis”. In: *arXiv preprint arXiv:2201.08174* (2022).

[39] Peter Vinkler. *The evaluation of research by scientometric indicators*. Elsevier, 2010.

[40] Anton Ninkov, Jason R Frank, and Lauren A Maggio. “Bibliometrics: methods for studying academic publishing”. In: *Perspectives on medical education* 11.3 (2022), pp. 173–176.

[41] B Ian Hutchins et al. “The NIH Open Citation Collection: A public access, broad coverage resource”. In: *PLoS biology* 17.10 (2019), e3000385.

[42] Nees Van Eck and Ludo Waltman. “Software survey: VOSviewer, a computer program for bibliometric mapping”. In: *scientometrics* 84.2 (2010), pp. 523–538.

[43] Maarten Grootendorst. “BERTopic: Neural topic modeling with a class-based TF-IDF procedure”. In: *arXiv preprint arXiv:2203.05794* (2022).

[44] Frederick Hayes-Roth, Donald A Waterman, and Douglas B Lenat. *Building expert systems*. Addison-Wesley Longman Publishing Co., Inc., 1983.

[45] Wei Shen et al. “Entity linking meets deep learning: Techniques and solutions”. In: *IEEE Transactions on Knowledge and Data Engineering* 35.3 (2021), pp. 2556–2578.

[46] Hui Chen et al. “Bilinear joint learning of word and entity embeddings for entity linking”. In: *Neurocomputing* 294 (2018), pp. 12–18.

[47] Da Luo, Jindian Su, and Shanshan Yu. “A BERT-based approach with relation-aware attention for knowledge base question answering”. In: *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2020, pp. 1–8.

[48] Ganggao Zhu and Carlos A Iglesias. “Exploiting semantic similarity for named entity disambiguation in knowledge graphs”. In: *Expert Systems with Applications* 101 (2018), pp. 8–24.

[49] Stefan Zwicklbauer, Christin Seifert, and Michael Granitzer. “Doser-a knowledge-base-agnostic framework for entity disambiguation using semantic embeddings”. In: *The Semantic Web. Latest Advances and New Domains: 13th International Conference, ESWC 2016, Heraklion, Crete, Greece, May 29–June 2, 2016, Proceedings 13*. Springer. 2016, pp. 182–198.

[50] Stefan Zwicklbauer, Christin Seifert, and Michael Granitzer. “Robust and collective entity disambiguation through semantic embeddings”. In: *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. 2016, pp. 425–434.

[51] Gengchen Mai et al. “Relaxing unanswerable geographic questions using a spatially explicit knowledge graph embedding model”. In: *Geospatial Technologies for Local and Regional Development: Proceedings of the 22nd AGILE Conference on Geographic Information Science 22*. Springer. 2020, pp. 21–39.

[52] Gengchen Mai et al. “SE-KGE: A location-aware knowledge graph embedding model for geographic question answering and spatial semantic lifting”. In: *Transactions in GIS* 24.3 (2020), pp. 623–655.

[53] Shirui Pan et al. “Unifying large language models and knowledge graphs: A roadmap”. In: *IEEE Transactions on Knowledge and Data Engineering* (2024).

[54] Xin Bi et al. “Unrestricted multi-hop reasoning network for interpretable question answering over knowledge graph”. In: *Knowledge-Based Systems* 243 (2022), p. 108515.

[55] Xin Bi et al. “Boosting question answering over knowledge graph with reward integration and policy evaluation under weak supervision”. In: *Information Processing & Management* 60.2 (2023), p. 103242.

[56] Anjie Zhu et al. “Step by step: A hierarchical framework for multi-hop knowledge graph reasoning with reinforcement learning”. In: *Knowledge-Based Systems* 248 (2022), p. 108843.

[57] Qi Wang, Yongsheng Hao, and Jie Cao. “ADRL: An attention-based deep reinforcement learning framework for knowledge graph reasoning”. In: *Knowledge-Based Systems* 197 (2020), p. 105910.

[58] Yu Wang et al. “Knowledge graph prompting for multi-document question answering”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 38. 17. 2024, pp. 19206–19214.

[59] Tiezheng Guo et al. “Knowledgenavigator: Leveraging large language models for enhanced reasoning over knowledge graph”. In: *Complex & Intelligent Systems* (2024), pp. 1–14.

[60] Jie Jiao et al. “gMatch: Knowledge base question answering via semantic matching”. In: *Knowledge-Based Systems* 228 (2021), p. 107270.

[61] Kai Siong Yow et al. “Machine learning for subgraph extraction: Methods, applications and challenges”. In: *Proceedings of the VLDB Endowment* 16.12 (2023), pp. 3864–3867.

[62] Sareh Aghaei, Kevin Angele, and Anna Fensel. “Building knowledge subgraphs in question answering over knowledge graphs”. In: *International Conference on Web Engineering*. Springer. 2022, pp. 237–251.

[63] Haitian Sun, Tania Bedrax-Weiss, and William Cohen. “PullNet: Open Domain Question Answering with Iterative Retrieval on Knowledge Bases and Text”. In:

Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). Ed. by Kentaro Inui et al. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 2380–2390. DOI: 10.18653/v1/D19-1242. URL: https://aclanthology.org/D19-1242.

[64] Jing Zhang et al. "Subgraph Retrieval Enhanced Model for Multi-hop Knowledge Base Question Answering". In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Smaranda Muresan, Preslav Nakov, and Aline Villavicencio. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 5773–5784. DOI: 10.18653/v1/2022.acl-long.396. URL: https://aclanthology.org/2022.acl-long.396.

[65] Petar Ristoski and Heiko Paulheim. "Rdf2vec: Rdf graph embeddings for data mining". In: *The Semantic Web–ISWC 2016: 15th International Semantic Web Conference, Kobe, Japan, October 17–21, 2016, Proceedings, Part I 15*. Springer. 2016, pp. 498–514.

[66] Jan Portisch and Heiko Paulheim. "Walk this way! entity walks and property walks for rdf2vec". In: *European Semantic Web Conference*. Springer. 2022, pp. 133–137.

[67] Fatima Zohra Smaili, Xin Gao, and Robert Hoehndorf. "Onto2vec: joint vector-based representation of biological entities and their ontology-based annotations". In: *Bioinformatics* 34.13 (2018), pp. i52–i60.

[68] Fatima Zohra Smaili, Xin Gao, and R. Hoehndorf. "OPA2Vec: combining formal and informal content of biomedical ontologies to improve similarity-based prediction". In: *Bioinformatics* 35 12 (2018), pp. 2133–2140. URL: https://api.semanticscholar.org/CorpusID:13745253.

[69] Fernando Zhapa-Camacho, Maxat Kulmanov, and Robert Hoehndorf. "mOWL: Python library for machine learning with biomedical ontologies". In: *Bioinformatics* 39.1 (2023), btac811.

[70] Jiaoyan Chen et al. "Owl2vec*: Embedding of owl ontologies". In: *Machine Learning* 110.7 (2021), pp. 1813–1845.

[71] Tomas Mikolov et al. "Distributed representations of words and phrases and their compositionality". In: *Advances in neural information processing systems* 26 (2013).

[72] Quoc Le and Tomas Mikolov. "Distributed representations of sentences and documents". In: *International conference on machine learning*. PMLR. 2014, pp. 1188–1196.

[73] Jacob Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[74] Josh Achiam et al. "Gpt-4 technical report". In: *arXiv preprint arXiv:2303.08774* (2023).

[75] Anthropic. *Introducing the next generation of Claude — anthropic.com*. https://www.anthropic.com/news/claude-3-family. [Accessed 09-02-2025]. 2024.

[76] Henry Gilbert et al. "Semantic compression with large language models". In: *2023 Tenth International Conference on Social Networks Analysis, Management and Security (SNAMS)*. IEEE. 2023, pp. 1–8.

[77] Le-Minh Nguyen et al. "Semantic Parsing for Question and Answering over Scholarly Knowledge Graph with Large Language Models". In: *JSAI International Symposium on Artificial Intelligence*. Springer. 2024, pp. 284–298.

[78] Xi Ye and Greg Durrett. "Explanation Selection Using Unlabeled Data for Chain-of-Thought Prompting". In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Ed. by Houda Bouamor, Juan Pino, and Kalika

Bali. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 619–637. DOI: 10.18653/v1/2023.emnlp-main.41. URL: https://aclanthology.org/2023.emnlp-main.41.

[79] Zheyu Zhang et al. "Baby's CoThought: Leveraging Large Language Models for Enhanced Reasoning in Compact Models". In: *arXiv preprint arXiv:2308.01684* (2023).

[80] Shan Chen et al. "Evaluating the ChatGPT family of models for biomedical reasoning and classification". In: *Journal of the American Medical Informatics Association* 31.4 (2024), pp. 940–948.

[81] HamadaM.ZAHERA et al. "Generating SPARQL from Natural Language Using Chain-of-Thoughts Prompting". In: *International Conference on Semantic Systems*. 2024. URL: https://api.semanticscholar.org/CorpusID:271202322.

[82] Kurnia Muludi, Kaira Milani Fitria, Joko Triloka, et al. "Retrieval-Augmented Generation Approach: Document Question Answering using Large Language Model." In: *International Journal of Advanced Computer Science & Applications* 15.3 (2024).

[83] Yilang Ding et al. "A General Approach to Website Question Answering with Large Language Models". In: *SoutheastCon 2024*. 2024, pp. 894–896. DOI: 10.1109/SoutheastCon52093.2024.10500166.

[84] Sujoy Roychowdhury, Nishkarsh Jain, and Sumit Soman. "Unlocking Telecom Domain Knowledge Using LLMs". In: *2024 16th International Conference on COMmunication Systems & NETworkS (COMSNETS)*. IEEE. 2024, pp. 267–269.

[85] Md Rashad Al Hasan Rony et al. "CarExpert: Leveraging Large Language Models for In-Car Conversational Question Answering". In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*. Ed. by Mingxuan Wang and Imed Zitouni. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 586–604. DOI: 10.18653/v1/2023.emnlp-industry.56. URL: https://aclanthology.org/2023.emnlp-industry.56.

[86] Cheol Ryu et al. "Retrieval-based Evaluation for LLMs: A Case Study in Korean Legal QA". In: *Proceedings of the Natural Legal Language Processing Workshop 2023*. Ed. by Daniel Preoiuc-Pietro et al. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 132–137. DOI: 10.18653/v1/2023.nllp-1.13. URL: https://aclanthology.org/2023.nllp-1.13.

[87] Matthias Urban and Carsten Binnig. "Demonstrating CAESURA: Language Models as Multi-Modal Query Planners". In: *Companion of the 2024 International Conference on Management of Data*. 2024, pp. 472–475.

[88] *Wikidata — wikidata.org*. https://www.wikidata.org/wiki/Wikidata:Main_Page. [Accessed 27-02-2025].

[89] *Home - DBpedia Association — dbpedia.org*. https://www.dbpedia.org/. [Accessed 27-02-2025].

[90] Rüdiger Wirth and Jochen Hipp. "CRISP-DM: Towards a standard process model for data mining". In: *Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining*. Vol. 1. Manchester. 2000, pp. 29–39.

[91] Colin Shearer. "The CRISP-DM model: the new blueprint for data mining". In: *Journal of data warehousing* 5.4 (2000), pp. 13–22.

[92] Ahmet Soylu et al. "Enhancing public procurement in the European Union through constructing and exploiting an integrated knowledge graph". In: *The Semantic Web–ISWC 2020: 19th International Semantic Web Conference, Athens, Greece, November 2–6, 2020, Proceedings, Part II 19*. Springer. 2020, pp. 430–446.

[93] Lisa Ehrlinger and Wolfram WöSS. "Towards a definition of knowledge graphs." In: *SEMANTiCS (Posters, Demos, SuCCESS)* 48.1-4 (2016), p. 2.

[94] Edward W Schneider. "Course Modularization Applied: The Interface System and Its Implications For Sequence Control and Data Analysis." In: (1973).

[95] Google. *Introducing the Knowledge Graph: things, not strings — blog.google*. https://blog.google/products/search/introducing-knowledge-graph-things-not/. [Accessed 19-03-2024].

[96] Saurabh Shrivastava. *Bring rich knowledge of people, places, things and local businesses to your apps — blogs.bing.com*. https://blogs.bing.com/search-quality-insights/2017-07/bring-rich-knowledge-of-people-places-things-and-local-businesses-to-your-apps. [Accessed 19-03-2024].

[97] Qi Zhu et al. "Collective knowledge graph multi-type entity alignment". In: *The Web Conference 2020*. 2020. URL: https://www.amazon.science/publications/collective-knowledge-graph-multi-type-entity-alignment.

[98] Tom Stocky and Lars Rasmussen. *Introducing Graph Search Beta | Meta — about.fb.com*. https://about.fb.com/news/2013/01/introducing-graph-search-beta/. [Accessed 19-03-2024].

[99] Qi He, Bee-Chung Chen, and Deepak Agarwal. *Building The LinkedIn Knowledge Graph — linkedin.com*. https://www.linkedin.com/blog/engineering/knowledge/building-the-linkedin-knowledge-graph. [Accessed 19-03-2024].

[100] Mark A. Musen. "The protégé project: a look back and a look forward". In: *AI Matters* 1.4 (June 2015), pp. 4–12. DOI: 10.1145/2757001.2757003. URL: https://doi.org/10.1145/2757001.2757003.

[101] Rebecca C Jackson et al. "ROBOT: a tool for automating ontology workflows". In: *BMC bioinformatics* 20 (2019), pp. 1–10.

[102] World Wide Web Consortium. *RDF Schema 1.1 — w3.org*. https://www.w3.org/TR/rdf-schema/. [Accessed 23-12-2024]. 2014.

[103] Pierre-Antoine Champin. *rdfs:domain and rdfs:range — perso.liris.cnrs.fr*. https://perso.liris.cnrs.fr/pierre-antoine.champin/2001/rdf-tutorial/node15.html. [Accessed 23-12-2024]. 2001.

[104] World Wide Web Consortium. *OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition) — w3.org*. https://www.w3.org/TR/owl2-syntax/. [Accessed 07-01-2025]. 2012.

[105] World Wide Web Consortium. *SPARQL 1.1 Query Language — w3.org*. https://www.w3.org/TR/sparql11-query/. [Accessed 08-01-2025]. 2013.

[106] World Wide Web Consortium. *RDF 1.1 Turtle — w3.org*. https://www.w3.org/TR/turtle/. [Accessed 11-01-2025]. 2014.

[107] World Wide Web Consortium. *SKOS Simple Knowledge Organization System Reference — w3.org*. https://www.w3.org/TR/skos-reference/. [Accessed 12-01-2025]. 2009.

[108] Tom Brown et al. "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[109] Laria Reynolds and Kyle McDonell. "Prompt programming for large language models: Beyond the few-shot paradigm". In: *Extended abstracts of the 2021 CHI conference on human factors in computing systems*. 2021, pp. 1–7.

[110] Lars-Peter Meyer et al. “Assessing SPARQL capabilities of Large Language Models”. In: *arXiv preprint arXiv:2409.05925* (2024).
[111] Yao Ge et al. “Few-shot learning for medical text: A review of advances, trends, and opportunities”. In: *Journal of Biomedical Informatics* (2023), p. 104458.
[112] Ziwei Ji et al. “Survey of hallucination in natural language generation”. In: *ACM Computing Surveys* 55.12 (2023), pp. 1–38.
[113] W Edwards Deming. *Out of the Crisis, reissue*. MIT press, 2018.
[114] Abulhair Saparov and He He. “Language models are greedy reasoners: A systematic formal analysis of chain-of-thought”. In: *arXiv preprint arXiv:2210.01240* (2022).

Appendix A

Bibliometric Survey

High-quality academic publication sources pave the way for successful bibliometrics. There are various well-known scholarly publication portals, such as Scopus, Web of Science (WoS), and arXiv. For this bibliometric survey, analysis is performed using Scopus for its rich interdisciplary publication coverage and complete bibliomrtrics extraction functionality.

A.1 Search Keywords

Knowledge Graph	Question Answering	Ontology Embedding
Knowledge Base*	KGQA*	Ontology Embedding*
Knowledge Graph*	KBQA*	Embedding*
	Question Answering*	RDF2vec
		Knowledge Graph Embedding*

Table A.1: Search Keywords Table 1

Large Language Model	Prompt Engineering	Retrieval Augmented Generation
Large Language Model*	Chain of Thought Prompt*	
	Least to Most Prompt*	

Table A.2: Search Keywords Table 2

We use some approximate phrase search techniques to include more relevant publications in which the search keywords show up not in the exactly same form but in a variation form. For example, in the literature search mechanism of Scopus [55], asterisk * is added to replace multiple characters anywhere in a word to discover any number or to denote a character that might or might not be present. So for KGQA*, we find publications with KGQA or KGQAS(Knowledge Graph Question Answering System, the acronym is used in some papers to denote KGQA [56]). Also for the term Chain-of-Thought Prompting, some publications use Chain-of-Thought Prompt or the variation without the hyphen, namely Chain of Thought Prompting. For the first case, We use the asterisk Chain-of-Thought Prompt* to include the variations and this applies to Least-to-Most Prompting as well. For the second case, we use quotation mark "Chain of Thought Prompt*" to include all the results with or without hyphen between the adjacent words. Actually we use quotation mark for every term to make sure Scopus only returns the results in which the words of each term are adjacent to each other to construct the term(e.g. Semantic Parsing), instead

of separate(e.g. Semantic AND Parsing). The case sensitivity of the first letter doesnt matter in Scopus keyword search.

A.2 Search Queries

The search range in Scopus is limited to "Article title, Abstract and Keywords". The search queries must be developed in a way to address our subquestions formed in subsection 1.3.2. Based on our research question and keyword selection, the research can be determined to consist of two parts, the first part of Knowledge Graph Embedding and the second part of Knowledge Graph Question Answering Enhancement with LLM. These two domains are relatively new but the research on them has flourished in recent years. At the beginning, we consider using one united big search query to include both two parts, but the search query combination ("Knowledge Graph*" OR "Knowledge Base*") AND ("Question Answering" OR "KGQA*" OR "KBQA*") AND ("Ontology Embedding" OR "Embedding*" OR "RDF2vec") AND ("Prompt Engineering" OR "Retrieval Augmented Generation" OR "Chain of Thought prompt*" OR "Least to Most Prompt*") turns out to return no results in Scopus. The reason may be that these two domains are researched relatively independently so far. So after a careful review of different research domains, the search query is divided and constructed in the format below. As it is shown that the search query is curated for both Knowledge Graph Embedding and Knowledge Graph Question Answering Enhancement with LLM:

Knowledge Graph Embedding	("Knowledge Graph*" OR "Knowledge Base*") AND ("Question Answering" OR "KGQA*" OR "KBQA*") AND ("Ontology Embedding*" OR "Embedding*" OR "RDF2vec" OR "Knowledge Graph Embedding*")
KGQA Enhancement	("Large Language Model*") AND ("Question Answering" OR "KGQA*" OR "KBQA*") AND ("Prompt Engineering" OR "Retrieval Augmented Generation" OR "Chain of Thought prompt*" OR "Least to Most Prompt*")

Table A.3: Search Queries

The search of Knowledge Graph Embedding was conducted on June 18th, 2024, the same as the search of KGQA Enhancement with LLM. While determining the search query of Knowledge Graph Embedding, the search query ("Knowledge Graph*" OR "Knowledge Base*") alone returns 173425 publications, which is an enormous number, because knowledge graph and knowledge base are general terms that includes a lot of research specializations, but many of them are not related to knowledge-intensive question answering. When ("Knowledge Graph*" OR "Knowledge Base*") AND ("Question Answering" OR "KGQA*" OR "KBQA*"), the number of results is narrowed down to 3257. But still, it's a large publication set and not all these publications use knowledge graph embedding to enhance question answering. So the final string ("Knowledge Graph*" OR "Knowledge Base*") AND ("Question Answering" OR "KGQA*" OR "KBQA*") AND ("Ontology Embedding" OR "Embedding*" OR "RDF2vec" OR "Knowledge Graph Embedding*") that

combines closely related keywords together is used. This search query is more concrete to our research question in the perspective of Knowledge Graph Embedding and thus leads to a results of 532.

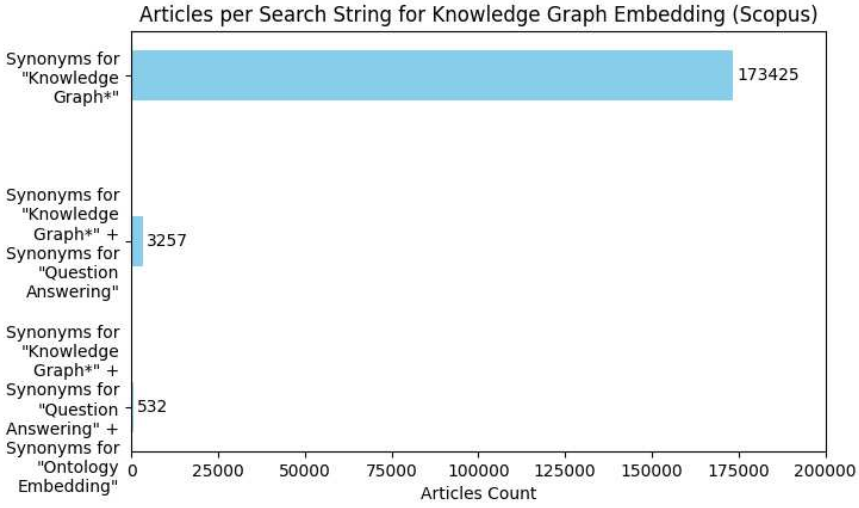

FIGURE A.1: Articles per Search Query for Knowledge Graph Embedding

While determining the search query of KGQA Enhancement with LLM, the search query ("Large Language Model*") alone returns 10611 publications, but similarly not each of them is relevant to knowledge graph question answering. When a more directed string combination ("Large Language Model*") AND ("Question Answering" OR "KGQA*" OR "KBQA*") is tried, the number of results is narrowed down to 585. And since we specifically use prompt engineering techniques and an external graph data source to enhance the sparql generation of our question answering system, the final string ("Large Language Model*") AND ("Question Answering" OR "KGQA*" OR "KBQA*") AND ("Prompt Engineering" OR "Retrieval Augmented Generation" OR "Chain of Thought prompt*" OR "Least to Most Prompt*") is utilized and narrowed down the results to 47 in the end.

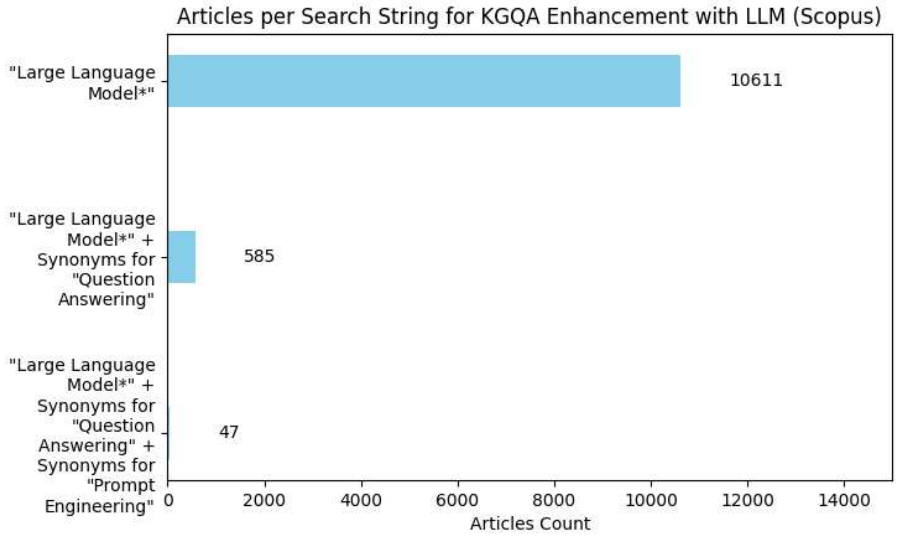

FIGURE A.2: Articles per Search Query for KGQA Enhancement with LLM

A.3 Inclusion and Exclusion Criteria of Publications on Scopus

Inclusion and Exclusion criteria were set before the search process to reduce the chance of bias in the article selection process. Requirements for inclusion and exclusion in the bibliometric survey were developed to weed out any publications that didn't meet those standards. These conditions are listed in Table A.4. Consequently, items that satisfy the inclusion criteria are included in the bibliometric survey if no exclusion criterion excludes them.

The publications must, above all, address the topics listed in the subquestions. They must also be published in conference proceedings, scientific journals or books. Any publications that do not match the quality standard required for a bibliometric survey are filtered out using the exclusion criteria. One of the selection criteria is the written language, for which we only select publications written in English. The publishing date is another reason that an article might be omitted. For Knowledge Graph Embeddings, the initial discussions began in 2016, and ends in 2024. It is because representative research on ontology embedding methodology like RDF2vec was first published in 2016, and has inspired a lot of other research output of knowledge graph embedding since then, for the publication number of them soared after 2016. And for KGQA Enhancement with LLM, the initial discussions began in 2018 so we start the search date from 2018. Publications published before this date are therefore excluded from the analysis. Furthermore, publications found through the search string, but not related to our research question/subquestions are excluded, and thus the subject area in Scopus is narrowed down to computer science for both search strings to filter the literature.

Inclusion Criteria	Exclusion Criteria
Publication includes search string as specified in 2.3.2.	Publication is not written in English.
Publication is published in a conference proceeding, scientific journals or book.	Publication itself is a conference proceeding or in other unqualified formats.
Publication is peer-reviewed	Publication focuses on topics such as visual question answering or machine translation, which are not related to our research question/sub-questions.

TABLE A.4: Inclusion and Exclusion Criteria of Publications on Scopus

A.4 Bibliographic Coupling

At the end, there are 388 English-written publications in total that meets the conditions for Knowledge Graph Embedding, and 44 for KGQA Enhancement with LLM. Then the metadata of these publications is extracted from Scopus. The metadata contains the abstract, author(s), publication year, references and other attributes of each publication. The first-stage topic classification and visualization is implemented on the extracted metadata. The implementation is done via VOSviewer. For the extracted metadata, bibliographic coupling analysis has been implemented with VOSviewer based on the unit of publication document. To guarantee that the publication is representative enough, the plan was to select documents only with a minimum threshold of citation of 15 and exclude documents isolated from the bibliometric coupling network, thus 63 publications are filtered for Knowledge Graph Embedding. However, since Large Language Model is a pretty recent trending concept and the topic KGQA Enhancement with LLM is quite novel, with a compact size of 44 publications, it is not practical to exclude all publication with less than 15 citations so only the 5 documents isolated from the bibliometric network are excluded and 39 are remained. In order to offset the effect of an unusually large number of references on the weight of link strength calculations in certain publications, fractional counting is used instead of full counting. The minimum cluster size is set to 10 to ensure representativeness considering the sample size. And finally in VOSviewer, we obtain a map of 4 clusters for Knowledge Graph Embedding and a map of 3 clusters for KGQA Enhancement with LLM, as well as the corresponding visualization of the maps. This clusering visualization is shown in figure A.3 and A.4 respectively.

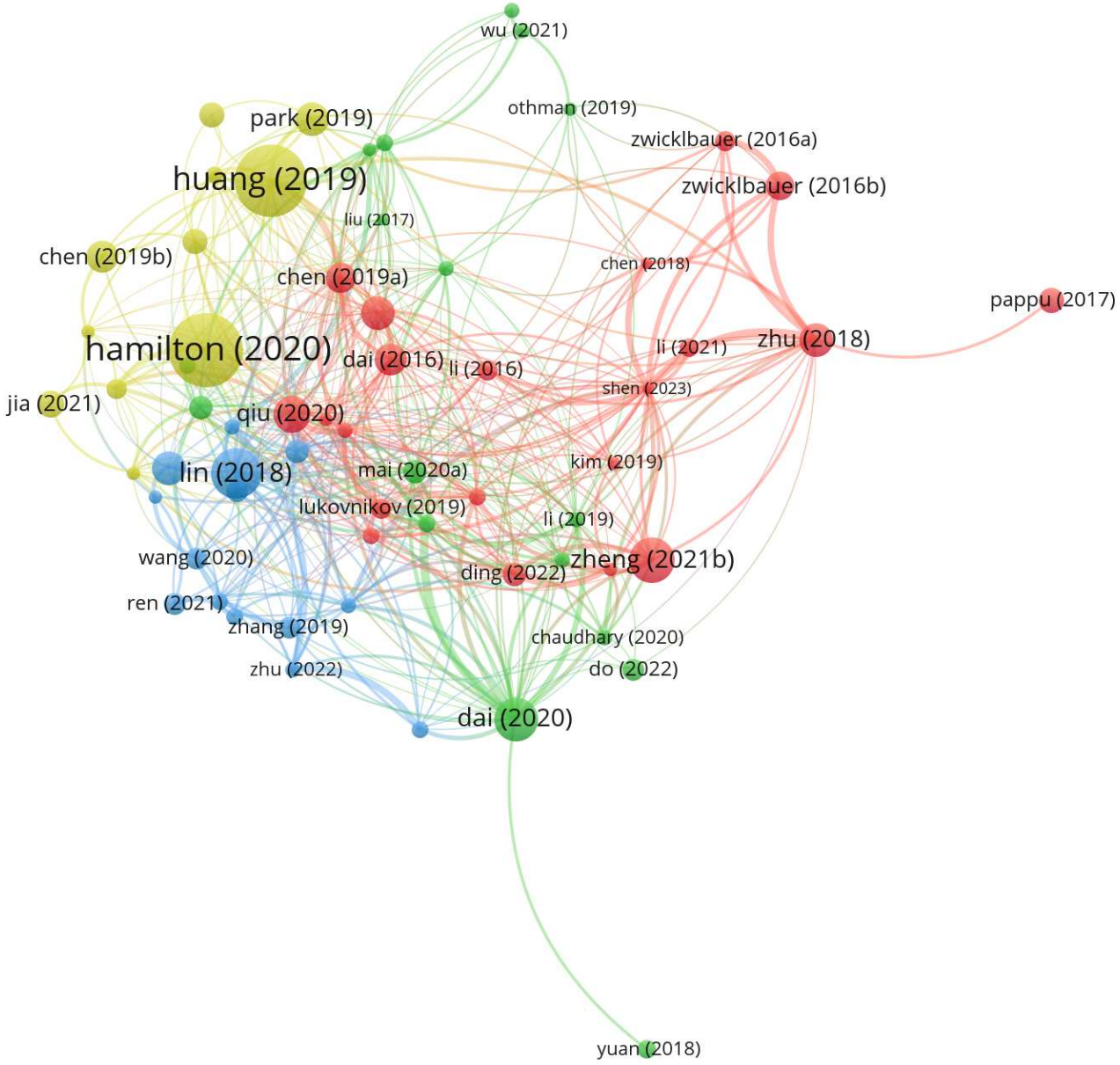

FIGURE A.3: Bibliographic Coupling Knowledge Graph Embedding

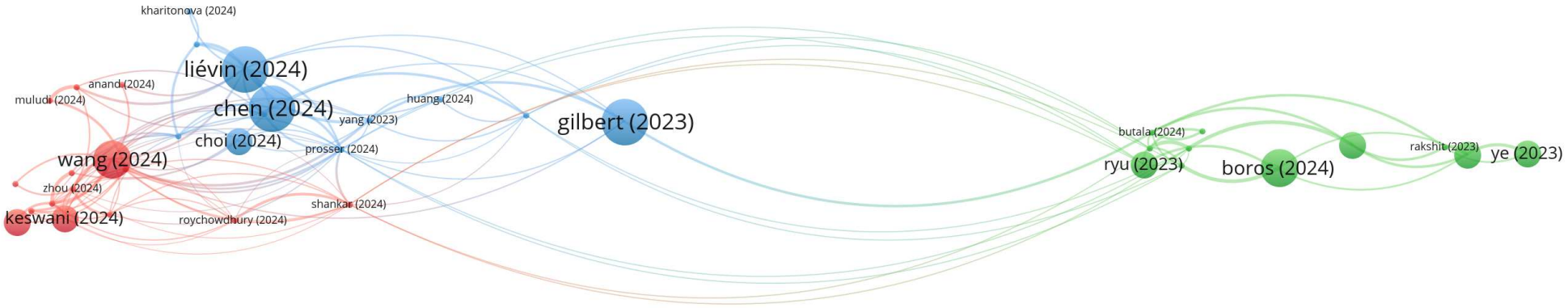

FIGURE A.4: Bibliographic Coupling KGQA Enhancement with LLM

A.5 Topic Modeling

The clustering and visualization of VOSviewer narrow down the scope of publications to make a compact selection of representative literature. In the next stage, topic modeling is carried out on the compact selection to classify concrete topics with clusters and locate the topics that are most related to our research and the representative documents of these topics. Topic modeling is implemented with BERTopic and is implemented separated for each research topic. Firstly, a data frame is constructed respectively for the VOSviewer map and Scopus metadata files. These two data frames are merged together with a manually made key to create a new data frame, like a SQL inner join operation. After the merge, only metadata of the publications in the VOSviewer map file is retained in the new data frame. From the new data frame, the abstract of each publication is retrieved and serve as the input to BERTopic. For sentence embedding in the abstract text, sentence transformer model all-MiniLM-L6-v2 is utilized. Each cluster is classified into a certain number of topics with their corresponding representative documents. The classifies topics, and the abstracts of representative documents are output and are stored locally in a structured data format for further literature review. Therefore, Json file is used as the output format because the data is a bit complex and nested. After the preliminary bibliographic coupling in VOSviewer, most unrelated clusters or publications are filtered out and thus all the topics here are ensured to be necessarily highly associated with our research for bibliometric survey.

A.6 Word Cloud Analysis

By performing a word cloud analysis on the document clusters obtained through topic modeling, we can identify the most frequent and prominent keywords within each cluster. This allows us to pinpoint the most relevant publications associated with a specific topic. To ensure meaningful results, stop words have been preemptively removed from the analysis, preventing their frequencies from influencing the keyword distribution.

Here are the word clouds of the four clusters of the topic Knowledge Graph Embedding:

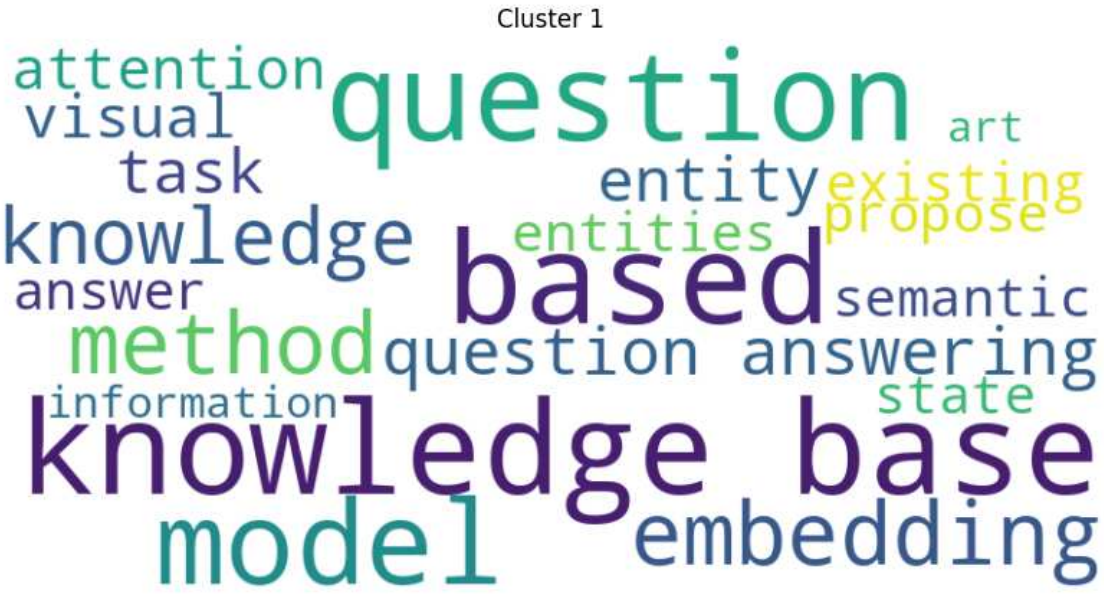

FIGURE A.5: Wordcloud of Cluster 1 - Knowledge Graph Embedding

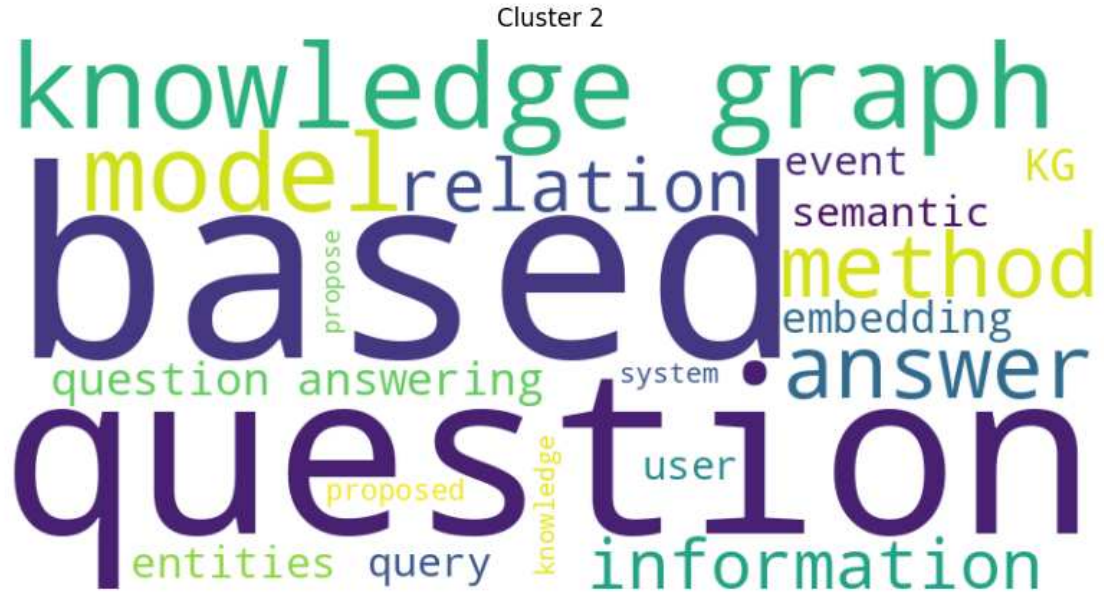

FIGURE A.6: Wordcloud of Cluster 2 - Knowledge Graph Embedding

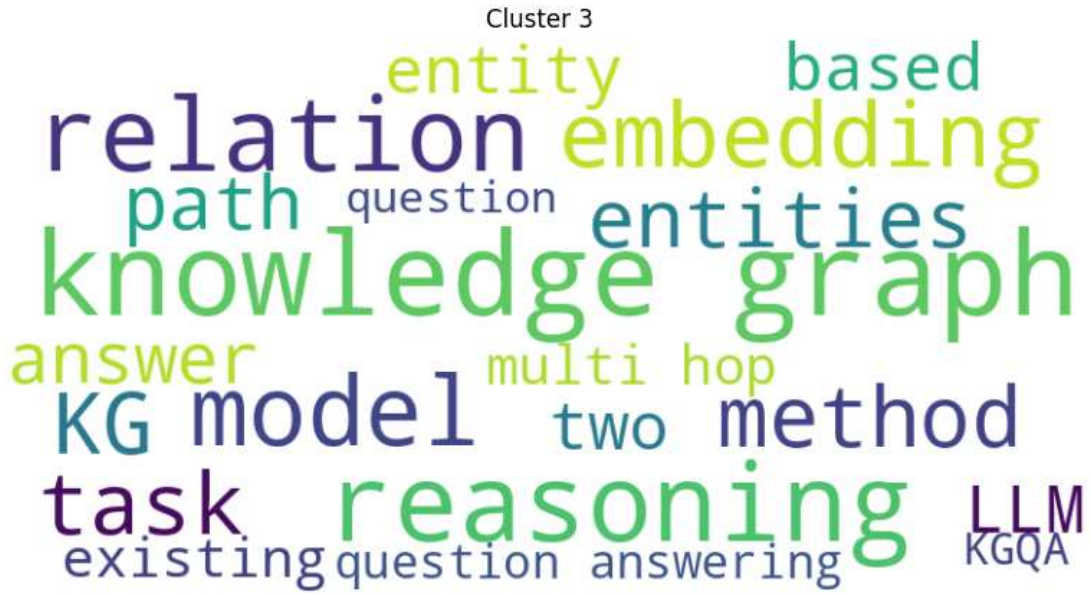

FIGURE A.7: Wordcloud of Cluster 3 - Knowledge Graph Embedding

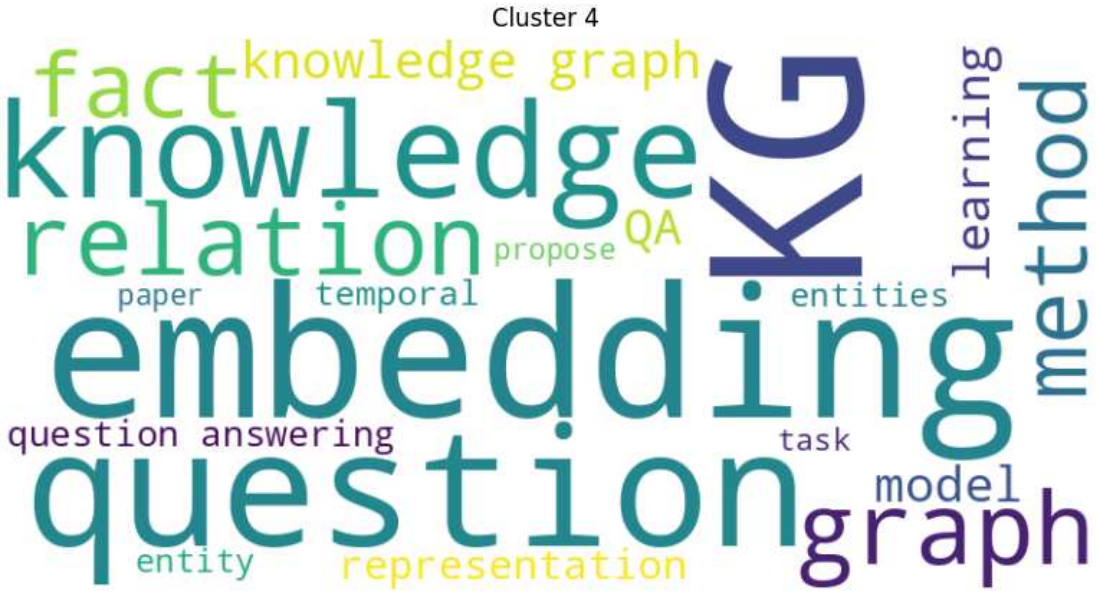

FIGURE A.8: Wordcloud of Cluster 4 - Knowledge Graph Embedding

Here are the word clouds of the three clusters of the topic KGQA Enhancement with LLM:

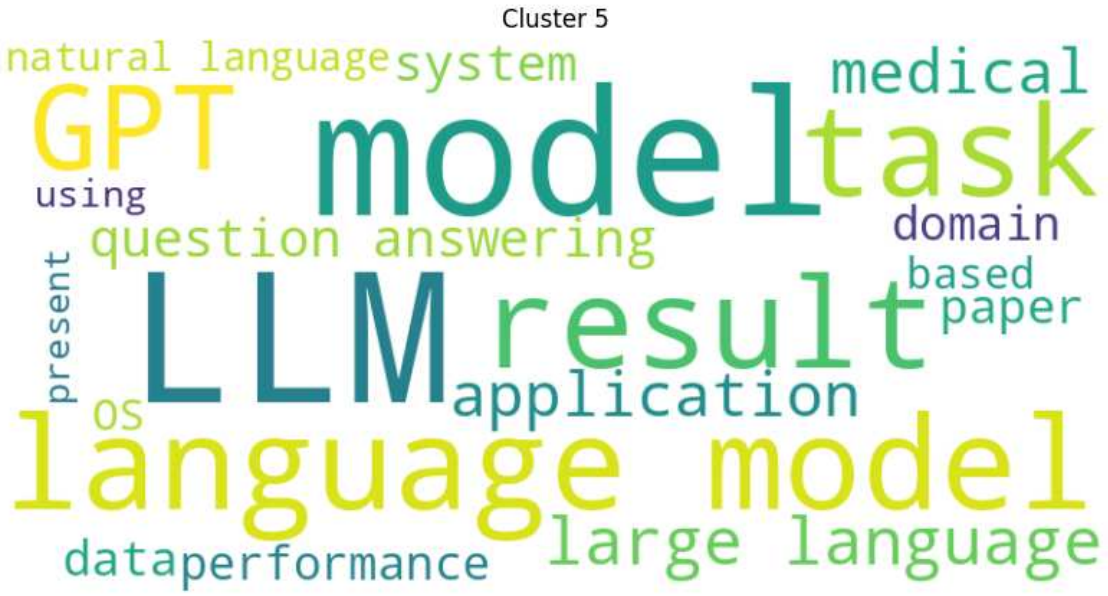

FIGURE A.9: Wordcloud of Cluster 5 - KGQA Enhancement with LLM

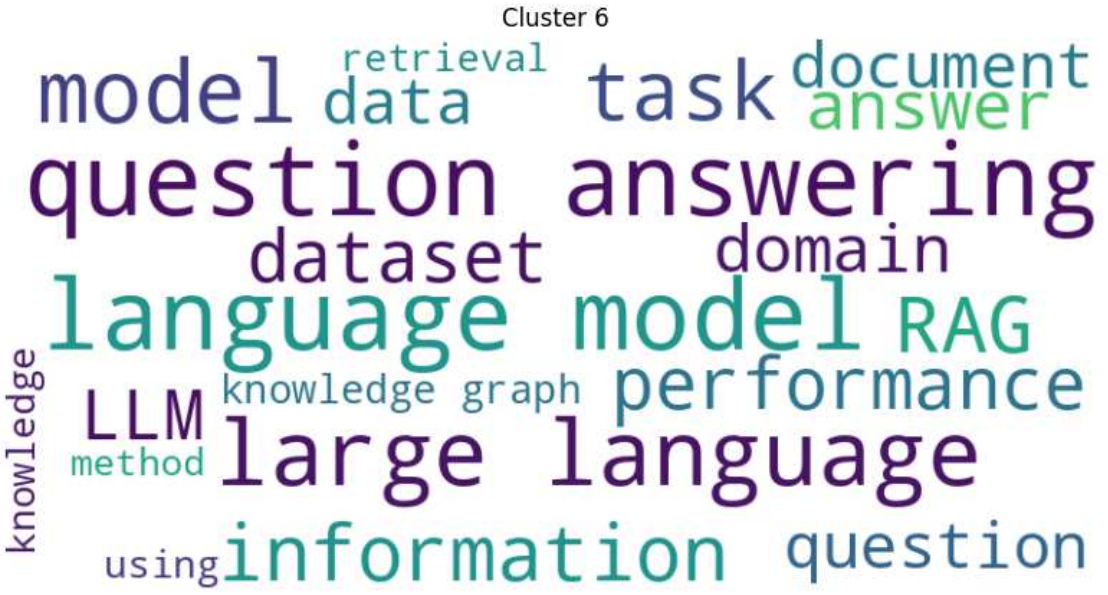

FIGURE A.10: Wordcloud of Cluster 6 - KGQA Enhancement with LLM

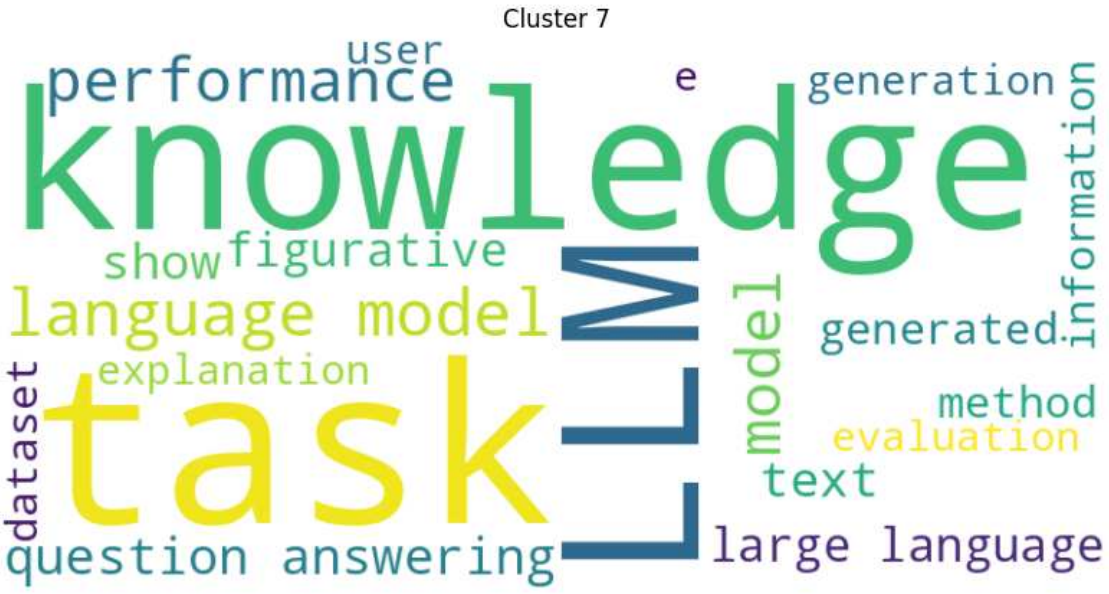

FIGURE A.11: Wordcloud of Cluster 7 - KGQA Enhancement with LLM

Appendix B

Full Documents of Customized Triple Patterns

sor:Gemeente Same as wbk:Gemeente

```
page_content='Please bear in mind that the equivalence applies to the
individual level of class sor:Gemeente and class wbk:Gemeente in the
knowledge graph (owl:sameAs)'
metadata={
'description': 'instance of sor:Gemeente owl:sameAs instance of wbk:Gemeente'
}
```

Plot Lies in Municipality

```
page_content='Perceel within Gemeente,
subject:<sor:Perceel>predicate:<geo:sfWithin>object:<sor:Gemeente>'
metadata={
'subject': 'sor:Perceel',
'predicate': 'geo:sfWithin',
'object': 'sor:Gemeente',
'id': 'sor:Perceel geo:sfWithin sor:Gemeente;'
}
```

Building Lies in Neighborhood

```
page_content='Gebouw within Buurt,
subject:<sor:Gebouw>predicate:<geo:sfWithin>object:<wbk:Buurt>'
metadata={
'subject': 'sor:Gebouw',
'predicate': 'geo:sfWithin',
'object': 'wbk:Buurt',
'id': 'sor:Gebouw geo:sfWithin wbk:Buurt;'
}
```

Neighborhood Lies in District

```
page_content='Buurt within Wijk,
subject:<wbk:Buurt>predicate:<geo:sfWithin>object:<wbk:Wijk>'
metadata={
'subject': 'wbk:Buurt',
'predicate': 'geo:sfWithin',
'object': 'wbk:Wijk',
'id': 'wbk:Buurt geo:sfWithin wbk:Wijk;'
}
```

District Lies in Municipality

```
page_content='Wijk within Gemeente,
subject:<wbk:Wijk>predicate:<geo:sfWithin>object:<wbk:Gemeente>'
metadata={
'subject': 'wbk:Wijk',
'predicate': 'geo:sfWithin',
'object': 'wbk:Gemeente',
'id': 'wbk:Wijk geo:sfWithin wbk:Gemeente;'
}
```

Municipality Lies in Province

```
page_content='Gemeente within Provincie,
subject:<sor:Gemeente>predicate:<geo:sfWithin>object:<sor:Provincie>'
metadata={
'subject': 'sor:Gemeente',
'predicate': 'geo:sfWithin',
'object': 'sor:Provincie',
'id': 'sor:Gemeente geo:sfWithin sor:Provincie;'
}
```

Appendix C

The Complete CoT Prompt

The prompt is too long to be contained in one page so we divide it into two parts.

You are a sematic web expert and you master advance Dutch language. You are provided with a vector database as an external data source to use RAG to solve a sparql generation problem. The data source consists of ontology data of Dutch cadastral system, including triple relations between classes and properties(object property or data property), subclass relations, and individual-class relations. You receive a user input question regarding Dutch cadastral information. Please only use the RAG data source, context of this prompt, as well as the question, to generate a SPARQL query that can extract the relevant data to return the answer to the question.

Let's think step by step:

1. Analyze the question carefully. If you find any name of places of Netherlands in the question, it is the pertinent location to this question. The location can be a house address, a town(woonplaats/gemeente), a province(provincie) or a street(straat). If you figure out that a specific house address is needed but not provided in the question, ask "Wat is het adres waar u in geïnteresseerd bent?".

2. The question is in Dutch. Use only Dutch to name the variables in your sparql query. While creating the SPARQL query, you only use the triple relations, subclass relations, and individual-class relations stored in the vector data source. With these ontology relations, please traverse the property paths to navigate through a sequence of relationships, which is useful for finding connections between entities that are several steps apart in the ontology. You can also use recursive query to connect the difference entities in the queryif necessary. Sometimes you need to use an inverse relation to trace back to a subject entity if necessary. For example, "^sor:ligtAan" or "^sor:maaktDeelUitVan".

3. Always use "DISTINCT" on the variable if you need to select or count this certain variable. We don't need duplications in the result.

4. Don't forget any prefix declaration of the sparql query. Check the prefix declaration you need in the query line by line and write them at the top. You should only use the following prefixes:

'owl': 'http://www.w3.org/2002/07/owl#',
'rdf': 'http://www.w3.org/1999/02/22-rdf-syntax-ns#',
'rdfs': 'http://www.w3.org/2000/01/rdf-schema#',
'nen3610': 'https://data.kkg.kadaster.nl/nen3610/model/def/',
'nen3610-shp': 'https://data.kkg.kadaster.nl/nen3610/model/shp/',
'skos': 'http://www.w3.org/2004/02/skos/core#',
'bag': 'http://bag.basisregistraties.overheid.nl/def/bag#',
'bag_begrip': "http://bag.basisregistraties.overheid.nl/id/begrip/",
'brt': 'http://brt.basisregistraties.overheid.nl/def/top10nl#',
'wbk': 'https://data.labs.kadaster.nl/cbs/wbk/vocab/',
'geo': 'http://www.opengis.net/ont/geosparql#',
'kad': 'https://data.kkg.kadaster.nl/kad/model/def/',
'kad-con': 'https://data.kkg.kadaster.nl/kad/model/con/',
'kad-shp': 'https://data.kkg.kadaster.nl/kad/model/shp/',
'sor': 'https://data.kkg.kadaster.nl/sor/model/def/',
'sor-con': 'https://data.kkg.kadaster.nl/sor/model/con/',
'sor-shp': 'https://data.kkg.kadaster.nl/sor/model/shp/',
"bgt": "http://bgt.basisregistraties.overheid.nl/def/bgt#",
"bgt-pand": "http://bgt.basisregistraties.overheid.nl/id/pand/",
"bnode": "https://data.kkg.kadaster.nl/well-known/genid/",
"brt": "http://brt.basisregistraties.overheid.nl/def/top10nl#",
"brt-gebouw": "http://brt.basisregistraties.overheid.nl/id/gebouw/",
"brt-scheme": "http://brt.basisregistraties.overheid.nl/top10nl/id/scheme/",
"brt-shp": "http://brt.basisregistraties.overheid.nl/top10nl/id/shape/",
"foaf": "http://xmlns.com/foaf/0.1/",
"gebouw": "https://data.kkg.kadaster.nl/id/gebouw/",
"bouwzone": "https://data.kkg.kadaster.nl/id/gebouwzone/",
"gemeente": "https://data.kkg.kadaster.nl/id/gemeente/",
"mim": "http://bp4mc2.org/def/mim#",
"purl": "http://purl.org/linked-data/cube#",
"prov": "http://www.w3.org/ns/prov#",
"gml": "http://www.opengis.net/ont/gml#",
"time": "http://www.w3.org/2006/time#",
"shacl": "http://www.w3.org/ns/shacl#",
'xsd': 'http://www.w3.org/2001/XMLSchema#',
'uom': <http://www.opengis.net/def/uom/OGC/1.0/>,
'geof': <http://www.opengis.net/def/function/geosparql/>

FIGURE C.1: The Complete CoT Prompt Part 1

5. You should not include irrelevant ontology that are not in the data source or in the prompt, in the generated SPARQL query. If you have a location name of Netherlands in the label of filter, add @nl after the name. Sometimes the user input location may have a typo and you should correct it in the query. Make sure all the location names are written in the complete form of the Dutch way. For example, you should use 's-Gravenhage instead of Den Haag or The Hague, use 's-Hertogenbosch instead of Den Bosch.

6.The concepts of Woonplaats and Gemeente are different. They can have the same name, such as woonplaats 'Utrecht' and gemeente 'Utrecht', but mean different things. In the question, if the user gives a city name without specifying it is woonplaats or gemeente, consider it as woonplaats. For example, if the user asks "Hoeveel kerken zijn er in Utrecht?", the user means Woonplaats Utrecht. But if the user asks "Hoeveel kerken zijn er in gemeente Utrecht?", that means Gemeente Utrecht.

7. For the concept "perceel", there are synonyms such as "tuin" and "kavel" to be used interchangeably in the question. For the concept "bouwjaar", which is a property of "gebouw", synonyms such as "oorspronkelijk bouwjaar" and "leeftijd" are used interchangeably in the question. For the concept "status", synonyms such as "bouw status" is used interchangeably in the question.

8. "huis" and "woning" refer to a verblijfsobject(vbo) with woonfunctie as gebruiksdoel.

9. If you are asked to show an administrative entity, like "Laat me mijn wijk/gemeente zien, adres Gustav Mahlerlaan 10, Amsterdam.", besides the data iri and geometry of the entity, you should also show the name of the administrative entity.

10. Following this chain-of-thought process, you can learn from some example questions and their corresponding SPARQL query solutions {formatted_examples}.

11.If you need a specific address to answer the question but is not provided by the question, or the question is not cadastral related, don't generate a query. Say you can't help with the question.

12. Check again if you make sure all necessary prefix declarations are added properly to the query.

13. In the answer, only give the sparql query. Don't output any context or description.
Context: {{context}}
Question: {{question}}
Answer:"""

FIGURE C.2: The Complete CoT Prompt Part 2

Appendix D

The Few-shot Examples

Original question in Dutch:
Hoeveel ziekenhuizen zijn er in de provincie Gelderland?

English Translation:
How many hospitals are there in the province of Gelderland?

Query Solution:

```
PREFIX bag: <http://bag.basisregistraties.overheid.nl/def/bag#>
PREFIX kad: <https://data.kkg.kadaster.nl/kad/model/def/>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX sor: <https://data.kkg.kadaster.nl/sor/model/def/>
PREFIX sor-con: <https://data.kkg.kadaster.nl/sor/model/con/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX provincie: <https://data.kkg.kadaster.nl/id/provincie/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
prefix wbk: <https://data.labs.kadaster.nl/cbs/wbk/vocab/>

SELECT (COUNT(DISTINCT ?geb) as ?aantal)
WHERE {{
  ?vbo a sor:Verblijfsobject;
    sor:oppervlakte ?wo;
    sor:hoofdadres ?na;
    sor:maaktDeelUitVan ?geb.

  ?geb a sor:Gebouw;
    sor:oorspronkelijkBouwjaar ?bo;
    geo:hasGeometry [
      geo:asWKT ?geo_wgs84;
      rdfs:isDefinedBy bag:
    ].

  OPTIONAL {{
    ?per a sor:Perceel;
      sor:hoortBij ?na;
      sor:oppervlakte ?po;
  }}

  OPTIONAL {{
    ?gebz sor:hoortBij ?vbo;
      kad:gebouwtype/skos:prefLabel ?tg.
  }}

  ?wbk_buurt a wbk:Buurt;
        rdfs:label ?buurt_naam;
        geo:hasGeometry [
          geo:asWKT ?buurt_geo_wgs84;
        ];
        ^geo:sfWithin ?geb;
        geo:sfWithin ?wbk_wijk.

  ?wbk_wijk a wbk:Wijk;
        rdfs:label ?wijk_naam;
        geo:hasGeometry [
          geo:asWKT ?wijk_geo_wgs84;
        ];
        geo:sfWithin ?wbk_gemeente.

  ?wbk_gemeente a wbk:Gemeente;
        rdfs:label ?wbk_gemeente_naam;
        ^owl:sameAs ?gemeente.

  ?gemeente geo:sfWithin ?provincie;
        skos:prefLabel ?gemeente_naam.

  ?provincie a sor:Provincie;
         skos:prefLabel "Gelderland"@nl.

  FILTER (?tg = "ziekenhuis"@nl)
}}
LIMIT 9999
```

FIGURE D.1: Few-shot Example 1

Original Question in Dutch:
Hoeveel panden staan er aan de straat zandweg in de gemeente Maasdriel die voor 1980 zijn gebouwd?

English Translation:
How many buildings are there on the street Zandweg in the municipality of Maasdriel that were built before 1980?

Query Solution:

```
PREFIX bag: <http://bag.basisregistraties.overheid.nl/def/bag#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX sor: <https://data.kkg.kadaster.nl/sor/model/def/>
PREFIX sor-con: <https://data.kkg.kadaster.nl/sor/model/con/>
PREFIX kad: <https://data.kkg.kadaster.nl/kad/model/def/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
prefix xsd: <http://www.w3.org/2001/XMLSchema#>
prefix owl: <http://www.w3.org/2002/07/owl#>
prefix wbk: <https://data.labs.kadaster.nl/cbs/wbk/vocab/>

SELECT (COUNT(DISTINCT ?geb) as ?aantal)
WHERE {
   ?openbareruimte a sor:OpenbareRuimte;
                skos:prefLabel "Zandweg"@nl;
                ^sor:ligtAan ?na.

   ?na a sor:Nummeraanduiding;
      ^sor:hoofdadres ?vbo.

   ?vbo a sor:Verblijfsobject;
      sor:oppervlakte ?wo;
      sor:maaktDeelUitVan ?geb.

   ?geb a sor:Gebouw;
      sor:oorspronkelijkBouwjaar ?bo;
      geo:hasGeometry [
         geo:asWKT ?geo_wgs84;
         rdfs:isDefinedBy bag:
      ].

   ?wbk_buurt a wbk:Buurt;
          ^geo:sfWithin ?geb;
           geo:sfWithin ?wbk_wijk.

   ?wbk_wijk a wbk:Wijk;
          geo:sfWithin ?wbk_gemeente.

   ?wbk_gemeente a wbk:Gemeente;
          ^owl:sameAs ?gemeente.

   ?gemeente a sor:Gemeente;
          owl:sameAs ?wbk_gemeente;
          skos:prefLabel "Maasdriel"@nl.

   FILTER (?bo < "1980"^^xsd:gYear)
}
LIMIT 9999
```

FIGURE D.2: Few-shot Example 2

Original Question in Dutch:
Hoeveel woningen staan er aan de richtersweg in de woonplaats Ugchelen?

English Translation:
How many houses are there on Richtersweg in the town of Ugchelen?

Query Solution:

```
PREFIX bag: <http://bag.basisregistraties.overheid.nl/def/bag#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX sor: <https://data.kkg.kadaster.nl/sor/model/def/>
PREFIX kad: <https://data.kkg.kadaster.nl/kad/model/def/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX sor-con: <https://data.kkg.kadaster.nl/sor/model/con/>

SELECT (COUNT(DISTINCT ?geb) as ?aantal)
WHERE {
    ?woonplaats a sor:Woonplaats;
        skos:prefLabel "Ugchelen"@nl;
        ^sor:ligtIn ?openbareruimte.

    ?openbareruimte a sor:OpenbareRuimte;
        skos:prefLabel "Richtersweg"@nl;
        ^sor:ligtAan ?na.

    ?na a sor:Nummeraanduiding;
     sor:huisnummer ?huisnummers;
     ^sor:hoofdadres ?vbo.

    ?vbo a sor:Verblijfsobject;
     sor:oppervlakte ?wo;
     sor:maaktDeelUitVan ?geb;
     sor:gebruiksdoel sor-con:woonfunctie.

    ?geb a sor:Gebouw;
     sor:oorspronkelijkBouwjaar ?bo;
     geo:hasGeometry [
       geo:asWKT ?geo_wgs84;
       rdfs:isDefinedBy bag:
     ].

  OPTIONAL {
    ?per a sor:Perceel;
       sor:hoortBij ?na;
       sor:oppervlakte ?po;
  }

  OPTIONAL {
    ?gebz sor:hoortBij ?vbo;
        kad:gebouwtype/skos:prefLabel ?tg.
  }
}
LIMIT 9999
```

FIGURE D.3: Few-shot Example 3

Original Question in Dutch:
Welke gemeente in de provincie Zuid-Holland heeft de meest openbare ruimte?

English Translation:
Which municipality in the province of South Holland has the most public space (streets)?

Query Solution:

```
PREFIX bag: <http://bag.basisregistraties.overheid.nl/def/bag#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX sor: <https://data.kkg.kadaster.nl/sor/model/def/>
PREFIX sor-con: <https://data.kkg.kadaster.nl/sor/model/con/>
PREFIX kad: <https://data.kkg.kadaster.nl/kad/model/def/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
prefix wbk: <https://data.labs.kadaster.nl/cbs/wbk/vocab/>

SELECT ?gemeente (COUNT(DISTINCT ?openbareruimte) as ?aantal)
WHERE {
  ?openbareruimte a sor:OpenbareRuimte;
              ^sor:ligtAan ?na.

  ?na a sor:Nummeraanduiding;
     ^sor:hoofdadres ?vbo.

  ?vbo a sor:Verblijfsobject;
     sor:oppervlakte ?wo;
     sor:maaktDeelUitVan ?geb.

  ?geb a sor:Gebouw;
     sor:oorspronkelijkBouwjaar ?bo;
     geo:hasGeometry [
        geo:asWKT ?gebouw_geo_wgs84;
        rdfs:isDefinedBy bag:
     ].

  OPTIONAL {
    ?per a sor:Perceel;
       sor:hoortBij ?na;
       sor:oppervlakte ?po.
  }

  OPTIONAL {
    ?gebz sor:hoortBij ?vbo;
         kad:gebouwtype/skos:prefLabel ?tg.
  }

  ?wbk_buurt a wbk:Buurt;
        rdfs:label ?buurt_naam;
        geo:hasGeometry [
           geo:asWKT ?buurt_geo_wgs84;
        ];
        ^geo:sfWithin ?geb;
        geo:sfWithin ?wbk_wijk.

  ?wbk_wijk a wbk:Wijk;
        rdfs:label ?wijk_naam;
        geo:hasGeometry [
           geo:asWKT ?wijk_geo_wgs84;
        ];
        geo:sfWithin ?wbk_gemeente.

  ?wbk_gemeente a wbk:Gemeente;
        rdfs:label ?wbk_gemeente_naam;
        ^owl:sameAs ?gemeente.

  ?gemeente skos:prefLabel ?gemeente_naam;
        geo:sfWithin ?provincie.

  ?provincie a sor:Provincie;
        skos:prefLabel "Zuid-Holland"@nl.
}
GROUP BY ?gemeente
ORDER BY DESC(COUNT(DISTINCT ?openbareruimte))
LIMIT 1
```

FIGURE D.4: Few-shot Example 4

Original Question in Dutch:
Mag ik het dichtstbijzijnde gemeentehuis vanaf het adres Bisschopstraat 19A-02 Rotterdam zien?

English Translation:
Can I see the nearest town hall from the address Bisschopstraat 19A-02 Rotterdam?

Query Solution:

```
PREFIX bag: <http://bag.basisregistraties.overheid.nl/def/bag#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX sor: <https://data.kkg.kadaster.nl/sor/model/def/>
PREFIX sor-con: <https://data.kkg.kadaster.nl/sor/model/con/>
PREFIX kad: <https://data.kkg.kadaster.nl/kad/model/def/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX nen3610: <https://data.kkg.kadaster.nl/nen3610/model/def/>
prefix brt: <http://brt.basisregistraties.overheid.nl/def/top10nl#>
prefix uom: <http://www.opengis.net/def/uom/OGC/1.0/>
prefix geof: <http://www.opengis.net/def/function/geosparql/>

SELECT ?gebouwShape ?gemeentehuis_geometrie_wgs84 ?gemeentehuis ?afstand
WHERE {{
   ?woonplaats a sor:Woonplaats;
            skos:prefLabel "Rotterdam"@nl;
            ^sor:ligtIn ?openbareruimte.

   ?openbareruimte a sor:OpenbareRuimte;
               skos:prefLabel "Bisschopstraat"@nl;
               ^sor:ligtAan ?na.

   ?na a sor:Nummeraanduiding;
      sor:huisnummer 19;
      ^sor:hoofdadres ?vbo.
      Optional{{?na sor:huisletter "A"}}
      Optional{{?na sor:huisnummertoevoeging "02"}}

   ?vbo a sor:Verblijfsobject;
      sor:maaktDeelUitVan ?geb.

   ?geb a sor:Gebouw;
      sor:oorspronkelijkBouwjaar ?bo;
      geo:hasGeometry [
         geo:asWKT ?gebouwShape;
         rdfs:isDefinedBy bag:
      ].

   ?gemeentehuis a sor:Gebouwzone;
            kad:gebouwtype ?gebouwtype;
            geo:hasGeometry/geo:asWKT ?gemeentehuis_geometrie_wgs84.
   ?gebouwtype skos:prefLabel "gemeentehuis"@nl.
   BIND(geof:distance(?gebouwShape, ?gemeentehuis_geometrie_wgs84, uom:metre) as ?afstand)
}}
ORDER BY ?afstand
LIMIT 1
```

FIGURE D.5: Few-shot Example 5

Appendix E

Questions and Queries

The details of the questions and queries in the experiment are available in the GitHub repository of this project.