universität wien

MASTERARBEIT | MASTER'S THESIS

Titel | Title

Analyzing the Potential of Geographic Knowledge Graphs for Advancing Spatial Capabilities of RAG-based Large Language Model Applications

verfasst von | submitted by

Simon Groß B.Sc.

angestrebter akademischer Grad | in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien | Vienna, 2025

Studienkennzahl lt. Studienblatt | Degree programme code as it appears on the student record sheet:	UA 066 856
Studienrichtung lt. Studienblatt | Degree programme as it appears on the student record sheet:	Masterstudium Kartographie und Geoinformation
Betreut von | Supervisor:	Univ.-Prof. Dr. Krzysztof Janowicz

Contents

List of Figures

List of Tables

List of Abbreviations

AI	Artificial Intelligence
API	Application Programming Interface
EPSG	European Petroleum Survey Group
EU	European Union
FEMA	Federal Emergency Management Agency
FN	False Negative
FP	False Positive
GenAI	Generative Artificial Intelligence
GeoAI	Geo Artificial Intelligence
GeoKG	Geographic Knowledge Graph
GIS	Geographic Information System
GIScience	Geographic Information Science
GPT	Generative Pre-Trained Transformer
GraphRAG	Graph Retrieval Augmented Generation
IRI	Internationalized Resource Identifier
KG	Knowledge Graph
LLM	Large Language Model
LOD	Linked Open Data
NOAA	National Oceanic and Atmospheric Administration
NUTS	Nomenclature of Territorial Units for Statistics
OGC	Open Geospatial Consortium
OWL	Web Ontology Language
QGIS	Quantum GIS / QGIS
RAG	Retrieval Augmented Generation
RCC8	Region Connection Calculus 8
RDF(S)	Resource Description Framework (Schema)
SWRL	Semantic Web Rule Language
SHACL	Shapes Constraint Language
ShEx	Shape Expressions Language
SOSA	Sensors, Observations, Samples, and Actuators
SPARQL	SPARQL Protocol And RDF Query Language
SSN	Semantic Sensor Network Ontology
TP	True Positive
UCSB	University of California, Santa Barbara
URI	Uniform Resource Identifier
USGS	United States Geological Survey
W3C	World Wide Web Consortium
WGS84	World Geodetic System 1984
WKT	Well Known Text

Abstract

Large Language Models (LLM) gained immense popularity in recent years. While exhibiting impressive abilities, also within GIScience, they suffer from so-called hallucinations. Retrieval Augmented Generation (RAG) emerged as an approach to mitigate these by retrieving factual documents and giving them as context to the LLM. GraphRAG is a recent variation the contextual information is retrieved from knowledge graphs (KG) instead. This work assesses the potential of GraphRAG for spatially explicit tasks. Three spatial concepts - topology, directionality and proximity - are analyzed through targeted questions. A GraphRAG system is set up that autonomously retrieves the answers to the questions from the KG. The question together with the KG's ontology are given to a LLM with the instructions to generate a GeoSPARQL query (including functions if applicable) that is executed automatically on a GraphDB instance containing the needed data. During the experiment, multiple parameters (model, temperature, etc.) are recorded and their effects on performance are evaluated. This approach outperforms a non-RAG setup by a wide margin. Overall the F1 scores are 0.81 for RAG compared to 0.37 for non-RAG when using the best performing model-temperature combination. For easier questions the RAG approach reaches F1 scores > 0.95. This study illustrates the potential of GraphRAG with the presented retrieval approach for advancing spatial capabilities of LLMs.

Zusammenfassung

Large Language Models (LLM) haben in den letzten Jahren stark an Popularität gewonnen. Trotz beeindruckender Fähigkeiten, auch in der Geoinformation, leiden sie unter Halluzinationen. Retrieval Augmented Generation (RAG) ist ein Ansatz, diese abzuschwächen, indem korrekte Dokumente abgerufen und als Kontext für das LLM verwendet werden. GraphRAG ist eine neuere Variante, bei der kontextbezogene Informationen stattdessen aus Wissensgraphen abgerufen werden. In dieser Arbeit wird das Potenzial von GraphRAG für räumliche Aufgaben untersucht. Drei räumliche Konzepte - Topologie, Richtungen und Nähe - werden durch gezielte Fragen analysiert. Ein GraphRAG-System wurde eingerichtet, das die Antworten auf die Fragen selbständig aus den Wissensgraphen abruft. Die Fragen werden zusammen mit der Ontologie des Graphen an einen LLM gegeben, mit der Anweisung, eine GeoSPARQL-Abfrage (ggf. mit Funktionen) zu generieren, die automatisch an eine GraphDB-Instanz gesendet wird, die die benötigten Daten enthält. Während des Experiments werden mehrere Parameter (Modell, Temperatur, usw.) aufgezeichnet und ihre Auswirkungen auf die Leistung bewertet. Dieser Ansatz übertrifft ein traditionelles LLM ohne RAG um ein Vielfaches. Insgesamt liegt der F1-Score bei 0,81 für RAG im Vergleich zu 0,37 ohne RAG bei der besten Modell-Temperatur-Kombination. Bei einfacheren Fragen erreicht der RAG-Ansatz F1-Scores $> 0{,}95$. Diese Studie veranschaulicht das Potenzial von GraphRAG mit dem vorgestellten Retrieval-Ansatz zur Verbesserung der räumlichen Fähigkeiten von LLMs.

1 Introduction and Motivation

After the launch of ChatGPT on the 30th of November 2022 by OpenAI, Large Language Models (LLM) experienced an unprecedented growth in popularity in research, industry and everyday life (OpenAI, 2022). It gained one million users in five days and 100 million users after two months making it the fastest technology to reach this milestone at the time (Hu, 2023). Today it boasts over 400 million users and ranks among the top 10 most visited websites worldwide (Duarte, 2025). Soon other companies followed and released their own LLMs, for example Google's Gemini (Pichai & Hassabis, 2023) or the recently published DeepSeek R1 model (DeepSeek, 2025). LLMs are trained on vast amounts of textual data, allowing them to understand and answer to natural language. Applications of LLMs include text generation, translation, summarization, dialogue systems, etc. (Hadi et al., 2024).

However, the shortcomings of these models soon became apparent. Since LLMs generate answers based on probabilities and giant training datasets, the models suffer from "hallucinations" (Lewis et al., 2020). These occur when a model is not able to answer a question and therefore generates an answer that sounds probable but is factually incorrect. Even with the rapid advancements since 2022 hallucinations still remain a risk for LLM based applications (Hadi et al., 2024).

Geographic Information Science (GIScience) and Geographic Artificial Intelligence (GeoAI) research soon took an interest in LLMs. Recent findings prove the ability of LLMs to generate impressive results for spatial tasks, however they also acknowledge problems relating to hallucinations (Roberts et al., 2023). Furthermore, LLMs are biased similar to humans regarding spatial relations between entities (Fulman et al., 2024b), have more accurate results for regions with higher populations (Roberts et al., 2023) and are even biased in sensitive topics such as the expected morality or intelligence of peoples in lower-income regions (Manvi et al., 2024). Furthermore, spatial relationships between objects, for example simple geographic questions like "What are hiking trails near Vienna?", will often give wrong or incomplete results since the LLM is not able to perform operations within a spatial reference system.

Already in 2020, Lewis et al. (2020) presented Retrieval Augmented Generation (RAG) to combine LLMs with a database containing documents of factual information that are retrieved via an embedded vector index to reduce hallucinations. This reduces hallucinations but still relies on unstructured documents containing also information that is irrelevant for the presented task. Recent research therefore explores the idea of incorporating structured data in the form of knowledge graphs (KG) as the underlying database for the RAG approach - this approach is called Graph Retrieval-Augmented Generation (GraphRAG) (Pan et al., 2024; Procko & Ochoa, 2024).

But what exactly are knowledge graphs? They are structured collections of entities and the relationships between them, represented as triples. This means that KGs not only store data but also its semantics, making them a powerful tool for modeling information and defining its meaning in a machine-readable format. Another essential component are ontologies, which provide formal definitions of a domain. They define the structure of the KG also using triples, specifying entity classes, possible relationships, and their constraints, enabling interpretation and reasoning over the data within the knowledge graph (Hogan et al., 2022). Therefore, if an ontology is understood, the data within the knowledge graph can be understood as well. Standards on how KGs and ontologies are represented (RDF) and queried (SPARQL) are defined by the W3C (see section 2.3.1) (RDF Working Group, 2014b; Car et al., 2023). These technologies are often referred to under the umbrella term "semantic web" (Hitzler, 2021).
There are general KGs like DBpedia (Auer et al., 2007) or Google's knowledge graph (Google, 2025), but also domain specific ones for language (Fellbaum, 2010), biology (Belleau et al., 2008) or GIScience. In GIScience, KGs and ontologies are already a well researched topic. Examples include ontologies on sensors (Compton et al., 2012), a KG derived from OpenStreetMap data (Dsouza et al., 2021), the geonames dataset (Geonames, 2024) or a collection of various spatial datasets for geo-enrichment services Janowicz et al. (2022). Furthermore the Open Geospatial Consortium (OGC) developed the GeoSPARQL standard, that defines the representation and querying of geospatial data in knowledge graphs (Car et al., 2023).
In GIScience, GeoAI has emerged as a prominent topic in recent years. The researched AI models within this domain are spatially explicit, meaning they directly integrate spatial data and its unique concepts into their design and training process (W. Li et al., 2024). By embedding spatial data explicitly for downstream machine learning tasks and incorporating concepts like spatial autocorrelation and heterogeneity these models models are optimized for spatial tasks. Also geographic knowledge graphs are a crucial building block of these GeoAI models (Mai et al., 2022). Furthermore, explicit geospatial data is incorporated into LLM training or fine-tuning to enhance their spatial understanding and capabilities (Deng et al., 2024).
However to create GeoAI models the data has to be embedded and the models have to be trained. Apart from consuming significant resources during the training process raising social and environmental questions, models need to be retrained to reflect new data (Shi et al., 2023). The large size and periodic nature of geospatial datasets therefore requires large computing power for training, running and retraining models.
With growing interest in the combination of KGs and LLMs through GraphRAG in other domains the question arises whether this approach could also be implemented in GIScience and GeoAI. If an LLM can understand an ontology, translate questions

about spatial concepts and data into KG queries and autonomously execute them, it would be able to access up-to-date data without the need to embed it directly into the model and without the need for retraining should the data be changed or even expanded. KGs are a mature and well studied technology, and their geospatial dimension is not only well researched but also clearly defined via standards by accredited institutions. Additionally, triplestores like GraphDB, a database specifically designed for storing and handling KGs and ontologies, provide professional infrastructure to perform analyses with semantic web technologies and support the geospatial standards described before (Ontotext, 2024, 2025a). Apart from computational requirements, the GraphRAG approach also improves the explainability as results are not generated out of a black-box, but the data that is provided to the LLM can be inspected.

Even with these rapid advancements in (RAG-)LLM technology, extensive research on Geographic Knowledge Graphs (GeoKG), and the available infrastructure, the integration of GraphRAG-LLMs and GeoKGs remains underexplored. This work aims to fill this gap.

Additionally, in 2020, Janowicz et al. (2020) formulated a moonshot: "Can we develop an artificial GIS analyst that passes a domain-specific Turing Test by 2030?". After the rapid advancements following the launch of ChatGPT and further LLMs the Space and Time for Knowledge Organization Lab at UCSB reiterated the idea in an announcement raising expectations and calling for contributions for developing this goal further (STKO Lab UC Santa Barbara, 2023). A recent, comprehensive overview on the development of autonomous GIS is provided by Z. Li et al. (2025). Today, some publications already explore autonomous GIS analysts. For example Akinboyewa et al. (2024) present a "GIS copilot" that is able to perform autonomous spatial analyses by generating code based on cataloged GIS functions or Y. Jiang & Yang (2024) use a LLM to generate SQL queries to answer questions on a spatial dataset. Given the proven value of KGs, ontologies and the semantics they provide for data, it stands to reason that they could serve as powerful tools to drive advancements in autonomous GIS analyses and enhance the spatial capabilities of LLMs.

This work first defines the baseline on how well a non-RAG LLM performs given questions about spatial data. A set of 18 questions was constructed that considers three different spatial concepts: topology (with a focus on neighborhood), directionality and proximity. These questions are asked to multiple OpenAI models with different sets of parameters. Then, the same questions are asked to a GraphRAG LLM, enhanced with an ontology that describes the KG containing the data needed to answer the question. The augmented question instructs the LLM to retrieve the information to answer the question from the KG by generating a GeoSPARQL query and also performing spatial analyses based on the GeoSPARQL standard. Also here, different parameters for the

LLM but also for other steps in the RAG workflow are examined.
The remainder of the thesis is structured as follows: The next chapter gives an extensive overview on the state of the art regarding (RAG)-LLMs, the semantic web and GraphRAG and puts these technologies in a geospatial context. Section 3 formulates the resulting research questions. After, the software and data used in this thesis are presented. Section 5 gives a detailed view on how the experiments were conducted. The results are presented in section 6 and are followed by the discussion in section 7. Finally, the thesis concludes and provides an outlook for future work in section 8.

2 State of the Art

2.1 Large Language Models

Language modeling is the process of predicting the next word in a text, generating coherent human language. Over the course of the years, different approaches to modeling language were analyzed, namely Statistical Language Models, Machine Learning Models and Deep learning models (Hadi et al., 2024). However, with the introduction of the transformer model, language modeling capabilities grew significantly and gave rise to modern LLMs like OpenAI's GPT series. Today's models are trained on vast amount of data and are able to understand complex and extensive contexts (Hadi et al., 2024).

In order for machine learning models to learn from natural language, it is first chunked into "tokens". These can be but do not have to be words in text. Using models like word2vec or GloVe, tokens are transformed into continuous vectors that represent the semantics of the word - its embedding. The difference between the vectors for Berlin and Germany might then be similar to the difference between the ones for Vienna and Austria (Mikolov et al., 2013; Pennington et al., 2014). The transformer model then introduced the concept of "attention" to language models. The vector embedding of a token is now not only dependent of the token itself, but is also influenced by the previous tokens. Therefore the model can learn how words interact with each other and can grasp deeper context of longer texts (Vaswani et al., 2017).

LLMs are a kind of Generative AI (GenAI). Their capabilities in generating human-like texts are based on large text datasets coupled with today's extensive computing power and data processing capabilities. In the span of only a few years the parameter sizes of LLMs have grown from a few hundred million for GPT-1 to 1.7 trillion for GPT-4. These parameters allow the model to deeply understand and generate human language (Hadi et al., 2024).

Models can be optimized to generate better output depending on the use case. They can be fine-tuned with additional training data, or prompt-engineering techniques can be applied. Here, the LLM is given a set of instructions in natural language about the desired answers, e.g. "Format your answer in markdown". A similar method is single-shot or few-shot training, where the LLM is provided with a few examples of how the output should look like. This provides the LLM with a frame of reference instead of generating an answer without any context (zero-shot prompting) (Hadi et al., 2024).

A crucial part of LLMs (and other machine learning models) is the softmax function. Essentially, this function provides the probability scores of the next possible tokens

of the LLM's output by normalizing each possible token's score. Given the softmax function (Hinton et al., 2015):

$$q_i = \frac{e^{z_i/T}}{\sum_j e^{z_j/T}} \tag{1}$$

z is the logit (the raw score) of a given token. The function turns this score into a probability q_i by normalizing it. The temperature T adjusts theses probabilities. A low temperature value leads to a "sharp" probability distribution, strongly favoring the most likely tokens, while a high temperature value produces a "soft" distribution with he opposite effect (Hinton et al., 2015). This controls how deterministic the LLMs output is. Low values correspond to deterministic output, while higher values make it more random (OpenAI, 2025).

Hadi et al. (2024) give an extensive overview on the domains where the impact of LLMs was studied. These include scientific advancements in education, healthcare, finance or text-modeling, or the LLMs' possible impacts on labor markets. The concrete tasks of LLMs range from text generation or classification to summarization and dialogue systems. Furthermore, the field of question answering and the generation of code is especially relevant since it is the main focus of this work.

LLMs also pose significant challenges: First, LLMs require vast amounts of training data and resources. This data can be biased, or copyrighted material and requires large amounts of computational power to be processed. This consumes significant financial and environmental resources. These costs associated with training and fine-tuning LLMs also make it more difficult to update a model with new knowledge (Hadi et al., 2024). Apart from these challenges Hadi et al. (2024) also address risks associated with the models, mainly the inherent biases of the models, the lack of explainability of LLMs due to their large parameter sizes, and hallucinations, which occur when a model makes up incorrect statements.

2.2 Retrieval Augmented Generation

Especially for knowledge intensive tasks, hallucinations were soon identified as a central problem of LLMs. While a model's parameters are generally able to store factual information, it underperforms compared to other designs. Also, due to intensive and time-consuming training required for these models, knowledge is cut off after a certain date (Lewis et al., 2020). Lewis et al. (2020) therefore developed the retrieval augmented generation (RAG) approach where a LLMs output is enhanced by non-parametric knowledge from indexed documents.

These documents are chunks of natural language text. In the case of Lewis et al. (2020) they are retrieved from Wikipedia, however it can be any data source. The documents are then encoded into a dense vector representation that is used as an index. Each

prompt to the LLM is encoded in a similar way. However, before generating the answer, the top-k documents with the most similar vectors are *retrieved* for the database. Now the prompt is *augmented* by combining the retrieved documents and the original prompt into one coherent prompt, providing the LLM with relevant context. The LLM can then *generate* answers for knowledge intensive tasks grounded on the documents (Lewis et al., 2020; Y. Gao et al., 2023). Since the documents can be changed quickly, the model is not constrained by a cut-off date and information can simply be updated in the database (Lewis et al., 2020; Y. Gao et al., 2023).

Y. Gao et al. (2023) provide a comprehensive overview on RAG and its more advanced iterations. They also outline the development of modular RAG, where the presented approach is enhanced by new modules (multiple data sources, LLM generated code for retrieval, etc.) and new patterns (new combinations of interactions, prompt rewriting, sub-queries). For example, multiple data sources like KGs and a vectorstore are combined, and multiple queries, with different instructions are sent and possible orchestrated through the LLM to provide it with the best context for generation. Z. Jiang et al. (2023) let the LLM generate temporary next sentences and only include RAG if the sentence includes low probability tokens, leading to improved question answering. Chen et al. (2024) address potential shortcomings of RAG-LLMs and propose a benchmark to test how they handle common problems, like noise, wrong or missing information in the retrieved documents or the ability to retrieve answers from multiple documents. Salemi & Zamani (2024) elaborate on how to evaluate the retrieval part of RAG systems and propose a new approach generating relevance labels for each document based on its performance. Fine-tuning as another approach for domain-specific question answering is compared with RAG, concluding with better performance for RAG. However, the potential of combining the two approaches is also highlighted, leading to even better outcomes (Soudani et al., 2024).

However, unstructured, indexed text-based documents are not the only way of grounding for RAG-LLMs. Y. Gao et al. (2023) present multiple studies using knowledge graphs as their data source. The following section provides a comprehensive overview on knowledge graphs and the section after describes their role in RAG.

2.3 The Semantic Web

In 2001, a popular vision of the semantic web was published in the Scientific American (Hendler et al., 2001). It is described as an extension of the World Wide Web, designed to make data machine-readable by structuring information with semantics. Unlike the traditional web, where content is primarily intended for human consumption, the semantic web uses technologies such as the Resource Description Framework (RDF), Web Ontology Language (OWL), and the SPARQL query language to enable computers

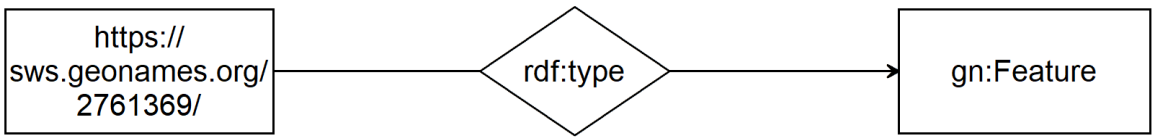
to link, interpret and retrieve data across different domains (Hendler et al., 2001). This vision would allow software agents to perform complex tasks such as querying databases, interpreting relationships, and extracting relevant information from web pages with minimal human intervention (Hendler et al., 2001). A broader overview on the semantic web as a field of research and its surrounding technologies is given by Hitzler (2021). He identifies three key concepts surrounding the semantic web: Ontologies, Knowledge Graphs and Linked Data each of which will be discussed in the following chapters. First, however, the Resource Description Framework is presented. It describes the standards that are commonly used to build KGs and is therefore crucial to understand before explaining the latter.

2.3.1 The Resource Description Framework

The basis for the semantic web is the Resource Description Framework (RDF) defined by the World Wide Web Consortium (W3C). They describe it as "a standard model for data interchange on the Web" RDF Working Group (2014b). It is used to define resources and their properties in a structured way and interlink them using subject-predicate-object statements called triples. The triples' elements can be Internationalized Resource Identifiers (IRI), blank nodes or literals of a datatype (RDF Working Group, 2014a). There are multiple syntaxes to define RDF triples. In this work only the "Turtle" (W3C, 2024) syntax will be used as it is closest to natural language which is suitable for writing but also for LLM comprehension. If one wants for example to define that Vienna is an instance of the "Feature" class defined by Geonames, a corresponding triple would look like this:

```
https://sws.geonames.org/2761369/ http://www.w3.org/1999/02/22-rdf-
    syntax-ns#type
https://www.geonames.org/ontology#Feature .
```

The first IRI corresponds to Vienna, where 2761369 is its ID in the Geonames dataset. The second IRI points to the definition of the relation "type" and the third IRI corresponds to the "Feature" class. The end of a triple is marked by a period. Visually it can be represented like this, with prefixes (see below) already in use:



IRIs are unique identifiers of a resource, these tend to look similar to URLs, however they do not have to be (but often are) associated with a web page. Literals are data

values with a specific data type. These can only be the object of a triple. For example a triple stating the population size of Vienna would be:

```
https://sws.geonames.org/2761369/ https://www.geonames.org/ontology
    #population
"1691468"^^http://www.w3.org/2001/XMLSchema#integer .
```

Or visually:

The first IRI is again Vienna. The second IRI is the relation ”population” defined by Geonames and the object (this time not an IRI but a literal) is the number of inhabitants in quotation marks followed by ^^ and the IRI of the datatype associated with this literal.
To simplify the long IRIs, prefixes (also called namespaces) are used. In Turtle a prefix is denoted at the beginning of the file using `@prefix [prefix name]: <[base IRI]>` and can then be used in the file. Prefixes are also used throughout this work within the text and are denoted in the list of prefixes in table 12 (Appendix).

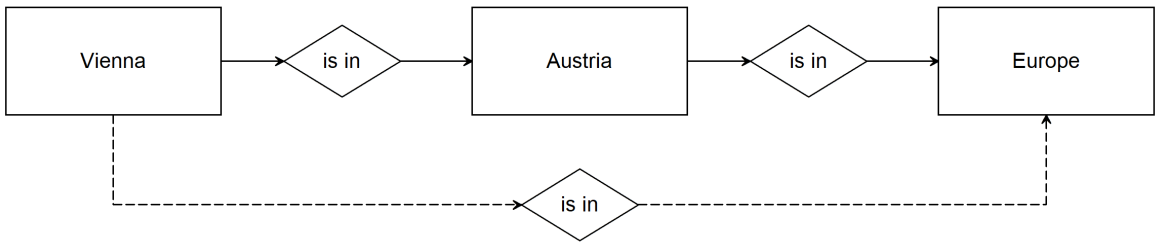
```
@prefix gn: <https://www.geonames.org/ontology#>
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>

https://sws.geonames.org/2761369/ rdf:type gn:Feature .

https://sws.geonames.org/2761369/ gn:population "1691468"^^xsd:
    integer .
```

Note that the IRI for Vienna is not abbreviated since it does not match the exact prefix defined above. But what exactly is a `gn:Feature` or what does the relation `gn:population` mean precisely? Answering these questions requires a formal definition of the understanding of these elements. In the semantic web these definitions are formalized by ontologies.

2.3.2 Ontologies

Uschold & Gruninger (1996) define an ontology as an ”explicit account of a shared

understanding [...] in a given subject area". Different ideas, understandings and implementations of concepts hinder collaboration and progress. Ontologies formalize knowledge, can be reused in similar domains and be interconnected with each other (Uschold & Gruninger, 1996). They are therefore a central part of the semantic web. A key aspect of ontologies is the use of explicitly defined vocabulary for entities and relations in the domain to be modeled (Horrocks, 2008).
However, in order to create ontologies and their vocabulary, a language is needed to formally describe them. While RDF(S) is suitable to describe data and simple relationships, it lacks the ability to model complex relationships. The Web Ontology Language (OWL) was therefore designed on top of RDF(S) to express more sophisticated models, while also using triples. OWL defines for example classes (`owl:Class`), properties (`owl:ObjectProperty`, `owl:DatatypeProperty`) and individuals that are members of classes. Additionally, OWL provides the ability to model cardinality (see figure 1, "population" relation) and value constraints or logical relationships between classes and therefore supports reasoning. For example Vienna is in Austria which is in Europe, therefore Vienna is in Europe if "is in" is defined as transitive (W3C, 2024):



For properties one of the most important aspects is what can be the subject and object of a property. This is modeled by `rdfs:domain` that specifies the possible classes for the subject and `rdfs:range` for the object. Based on these elements, the ontology defines the conceptual layer. Figure 1 shows a subset of the Geonames ontology (Geonames, 2024). Here, `gn:Feature` is a class. The class has three object properties with `rdfs:domain` and `rdfs:range` `gn:Feature`. The `rdfs:domain` and `rdfs:range` declarations on properties such as `gn:nearby` enable inference about the types of the subject and object. In this case, they indicate that both are instances of `gn:Feature`. Therefore, given a triple like `?something gn:nearby ?somethingElse`, a reasoning engine can infer that both `?something` and `?somethingElse` are of type `gn:Feature`. Also examples for a transitive property (parent feature) and symmetric property (neighbor) are given. A Feature can also have a feature code of class `gn:Code` shown as "Code" in figure 1. The number within the "Code" circle (690) stands for the number of individuals that are `rdf:type gn:Code` according to the current RDF version of the geonames ontology (Geonames, 2025c). The green properties are data properties that

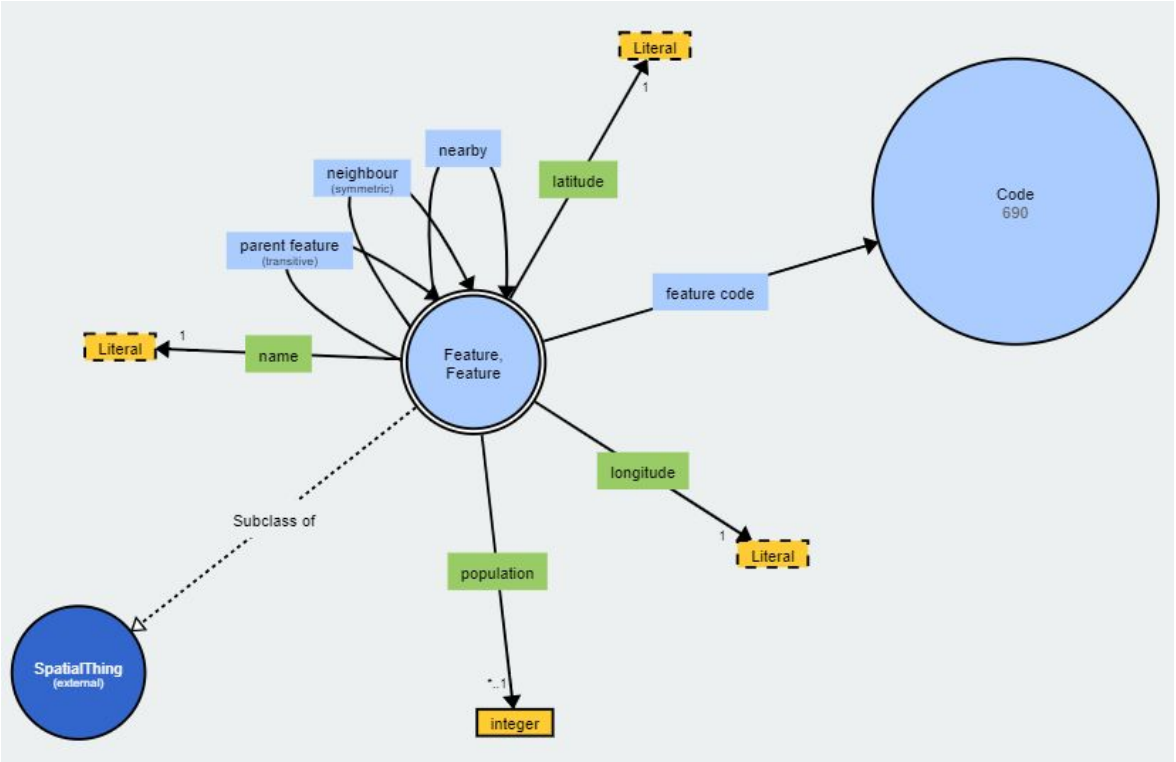


Figure 1: Light Blue: A subset of classes and properties of the Geonames ontology, Visualization by WebVOWL (Geonames, 2025c; VisualDataWeb, 2025).

have a literal as `rdfs:range`. For `gn:population` the datatype is specified to be an integer. Lastly, there is a connection to elements of another ontology (Spatial Thing). This illustrates how there are interconnection between ontologies in practice.
Ontologies can be used for automating entailment, meaning that new knowledge can be inferred from the data by reasoning over the classes and properties in the ontology and making use of the constraints defined within them (Hogan et al., 2022). Besides transitive and symmetric properties there are also asymmetric, reflexive and irreflexive properties. Furthermore, classes can be specified as equivalent, subclasses, inverses, or disjoint from other classes, enhancing the clarity of the defined ontologies (Hogan et al., 2022). While ontologies describe the conceptual layer, knowledge graphs are the actual data modeled within the specific ontology. These are described in the following chapter.

2.3.3 Knowledge Graphs

Hogan et al. (2022) define Knowledge Graphs (KG) as ”a graph of data intended to accumulate and convey knowledge of the real world, whose nodes represent entities of interest and whose edges represent potentially different relations between these entities”. Even though KGs and ontologies are fundamentally different concepts, they both use triples to describe the subject, predicates and objects that they consist of. While

the ontology models the understanding of the topic domain, the knowledge graph models the actual data. It can therefore also be called a ”data graph” (Hogan et al., 2022). The Knowledge Graph records explicit knowledge which can be for a private entity or maintained publicly (e.g. DBpedia (Auer et al., 2007), or Geonames as a geospatial example (Geonames, 2024)). This explicit knowledge is stored in a directed, edge-labeled graph, consisting of (if modeled with RDF, see section 2.3.1) URIs, Literals or Blank Nodes. Hierarchies are not explicitly needed, but can be modelled as well (Hogan et al., 2022). Ji et al. (2022) Identify four main research directions for KGs.
First, knowledge representation learning focuses on how entities and relations in triples can be represented as vectors in a continuous space. This allows for the use of embedding models to transform the elements of a knowledge graph into dense vector representations, which can then be used to estimate the likelihood of potential triples or for training machine learning models for other downstream tasks. There are different types of embedding models. Translational models approximate use the vector of the subject and add the vector ob the predicate to approximate the vector of a potential object Hogan et al. (2022). Convolutional Neural Networks use more sophisticated embeddings, e.g. from local interactions, concatenated triples or hypernetworks to model semantic interactions in the KG. Graph Neural Networks rely more on the structure of the KG, taking into account e.g. the direction of relations or the neighborhood of a subject (Ji et al., 2022). More detailed explanations are given in Ji et al. (2022). The embeddings derived from these processes are used in link prediction tasks, to expand existing knowledge graphs with further information (M. Wang et al., 2021). This already ties into the second research direction identified by Ji et al. (2022): Knowledge Acquisition. Here, triples are classified, new entities are discovered and new relations are identified. Furthermore the shape of the graph can be analyzed to for example, identifying communities, calculate centrality, assess graph connectivity or similarity of different nodes, or for graph summarization (Hogan et al., 2022).
Temporal KGs are a more recent research direction where the time dimension is taken into account, this enables to analyze temporal patters and logic, but also requires temporal embedding processes (Ji et al., 2022). Finally, examples that build on KGs are provided. These can be applications in natural language processing, question answering for single- or multi-hop questions or recommender systems.
To retrieve the data in a knowledge graph, it has to be queried. SPARQL is the query language specified by the W3C (Car et al., 2023) to query RDF graphs, however there are more query languages e.g. Cypher for the Neo4j database (Neo4j, 2025). These languages look for triple patters and return all variables that match the defined patterns. These can be trivial, e.g. `?city gn:name "Vienna"` will return the city that is named ”Vienna”, or more complex queries that handle sophisticated patterns (se-

lections, unions, differences, negations) and inferences like disjunctions, or transitives Hogan et al. (2022). Furthermore, the graphs can be validated by defining so-called "shapes". These shapes defines constraints that apply to this graph and against which the data can be validated. Prominent languages for shapes for RDF graphs are the Shapes Constrain Language (SHACL) (W3C, 2017) or the Shape Expressions Language (ShEx) (W3C, 2025).
Finally, the KG can be partitioned into different subgraphs or "named graphs" that put their respective subgraph into a specific context, e.g. a temporal scope that this named graph addresses (Hogan et al., 2022). These are also implemented as a powerful functionality in the GraphDB graph database (Ontotext, 2024).

2.3.4 Linked (Open) Data

Publishing datasets by itself without additional information, semantics and interconnections limits its machine-readability and isolates the data from the rest of the world. Linked Open Data (LOD) is a framework to publish data and linking it to other datasets using semantic web technologies. Berners-Lee (2006) outlines four principles for publishing LOD:

1. Using Uniform Resource Identifiers (URI, similar to IRI) as names for things.
2. Using URIs with http, to let users look up those names on a webpage.
3. Providing additional information for each URI using RDF and SPARQL standards
4. Including links to other URIs to provide related information

The resource description framework describes the semantics of a dataset, its properties and interconnections, and SPARQL is used to execute queries over the data (Bizer et al., 2009).
If, for example a website references another one through a hyperlink, the meaning of this link is unknown. Using sematic web technologies, the link (or predicate) has a label and the two pages have types. This allows a machine reading this link to infer more information on how the pages are connected (Kuhn et al., 2014). This is not only applicable for webpages but can be implemented for all kinds of data silos, linking them together and enabling them to build a web of interconnected data.
Arguably the most prominent example in LOD is DBpedia, an initiative to structure Wikipedia data to make it machine-readable (Auer et al., 2007). Through LOD, DBpedia is connected to many other datasets for various domains, allowing the datasets to complement each other. Other examples include Wikidata, a community built open

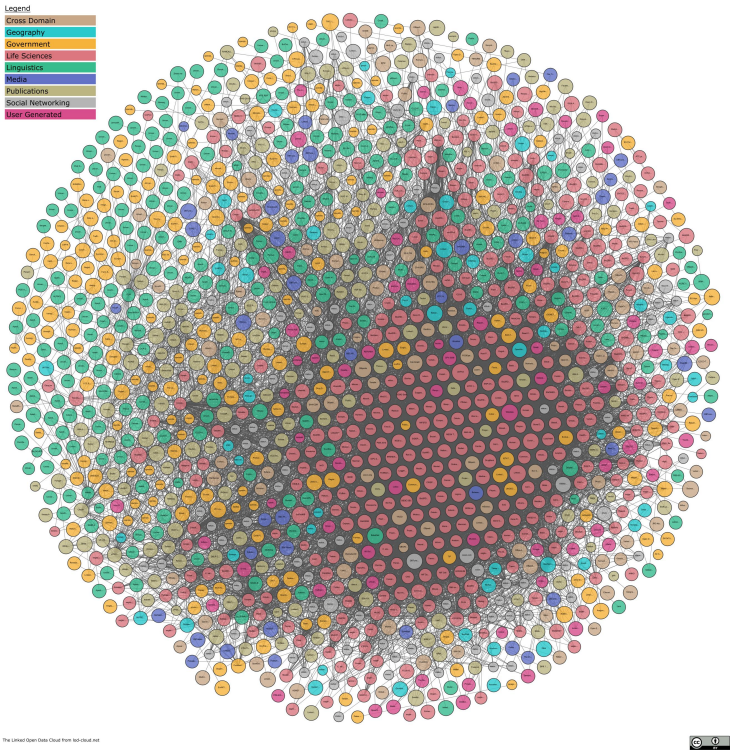


Figure 2: The Linked Open Data cloud (McCrae, 2024).

knowledge base (Erxleben et al., 2014), Europeana, linking cultural heritage objects from European museums and enhancing them with semantic information (Bernhard & Antoine, 2011), Bio2RDF, linking life science data (Belleau et al., 2008), Geonames, a gazeteer of geographic entity names (Geonames, 2024) and many more. The different linked datasets and their connections among each other are visualized on the Linked Open Data Cloud in figure 2. Today there are over 1300 linked open datasets from various domains including linguistics, life sciences, geography, etc. (McCrae, 2024).

2.4 Graph RAG

Section 2.1 gave an overview on LLMs, section 2.2 describe RAG as a way to improve LLM performance and the previous section illustrated semantic web technologies such as KGs and ontologies. GraphRAG now combines the RAG approach with semantic web technologies. Instead of retrieving unstructured documents via vector similarity to give context to the LLM, it retrieves structured data from a knowledge graph and gives this data as context to the LLM.

One of the main disadvantages of the RAG approach presented in section 2.2 is that the retireval of multiple similar unstructured documents also retrieves lots of noise. Especially of domain specific, deep and expert-level questions this can have siginificant impact on model performance (Procko & Ochoa, 2024). Knowledge Graphs provide a source for highly structured, concise and often domain specific information. This can

provide the LLM with less noisy context to generate better answers (Procko & Ochoa, 2024). Furthermore the more concise context leads to a significant reduction in tokens to be processed by the LLM which is less resource intensive.
The main task therefore is to retrieve this relevant information from the KG. Multiple tactics are explored in literature. Most of the times these revolve around extracting relevant subgraphs from the KG to give to the LLM. The graphs can for example be embedded into a dense vector space (see section 2.3.3). Embedding the question and then retrieving similar subgraphs, their community or top-k triples, can select appropriate information from the graph (Procko & Ochoa, 2024).
Baek et al. (2023) use this approach, embedding the graph and fetching top-k similar triples to the question, but also using a re-ranker that is trained to explicitly distinguish confusing results that seem similar but are not. Xu et al. (2024) use historical tickets from LinkedIN customer service as a KG and enhance them with LLM generated summaries that are used for similarity based retrieval. W. Liu et al. (2020) detect entities in the questions and then expand them through their related triples by changing the original sentence with the retrieved information.
A very different approach is using the LLM itself to generate a query to the KG directly, therefore retrieving the data without the use of vector embedding and semantic similarity. X. Wang et al. (2023) detect entities and their relations and then automatically generate Python code to retrieve the relevant information from the KG. In bioinformatic research, Emonet et al. (2024) prompt an LLM to automatically generate SPARQL queries over multiple KGs (federated queries). Additionally, they validate their queries against the ShEx schema (see section 2.3.3) of the endpoint they are executed on and provide the LLM with errors to improve the queries. However they also employ vector similarity to retrieve similar queries from a collection to provide examples to the LLM. In a recent paper Kosten et al. (2025) evaluate different prompting techniques for better LLM based SPARQL query generation. They also provide the LLM with the ontology of the KG, giving it the appropriate knowledge of the structure of the graph and therefore improving generation capabilities.
This approach is also employed in industry. GraphDB provides the GraphRAG-style experimental feature "Talk to Your Graph" where multiple tactics are implemented. Here the SPARQL query is generated based on a provided ontology or with a DESCRIBE query that gives information on the data in the graph (Ontotext, 2025b). Also LangChain, one of the most successful frameworks for building RAG applications integrates this approach in their libraries (LangChain, 2024).
This work explores the approach of using LLM based query generation for KG retrieval, without making use of vector embeddings or semantic similarity directly. Therefore, any subsequent mention of "GraphRAG" refers to this approach.

2.5 LLMs, AI and the Semantic Web in a Geospatial Context

Research into AI, LLMs and semantic web technologies has been progressing rapidly in the last years. This work looks at recent techniques in LLM research and applies them to a geospatial context. Anselin (1989) famously asked "What is Special about Spatial Data?". The first law of geography states that "everything is related to everything else, but near things are more related than distant things" (Tobler, 1979). Location, area, distance and interaction play a crucial role in the realm of spatial data. Additionally, concepts like spatial autocorrelaction and spatial heterogeneity are unique to GIScience. Furthermore, Anselin (1989) discusses the data-driven vs. the model driven approach to analyzing spatial data and highlights common errors associated with it, namely measurement errors and specification errors. Measurement errors are also discussed in the context of this thesis in section 7.4.3.
On a more recent note, Shi et al. (2023) addresses the large nature of spatial datasets and models and their implications. Since much data has a spatial or spatio-temporal perspective, large models and vast amounts of computing power are needed to process this data. This has environmental consequences that disproportionately affect populations that benefit less from these models.

2.5.1 GIScience and LLMs

So how do Large Language Models perform in the world of geography and GIScience? Roberts et al. (2023) thoroughly analyze the capabilities of LLMs to perform spatial tasks of various difficulties. They prompt OpenAIs GPT 4 model for various descriptive and application-centric tasks. They find strong capabilities of the model to recall socio-economic (Population, Life-expectancy, etc.) and quantitative indicators (area, elevation, etc.) of countries, cities and geographic features. However they also find accuracy being biased towards more populated cities and countries. For the more application centric tasks (planing a travel initiary, drawing shipping rountes, recalling country-shapes, etc.) they also find strong capabilities that are however prone to inaccuracies and hallucinations.
Deng et al. (2024) on the other hand trained a LLM as a foundation model for geosciences. They base their model on the LLaMA-7B LLM (Touvron et al., 2023) and fine-tune it in two stages. First, the create a Geoscience literature corpus composed of Wikipedia and scientific geoscience articles organized in the GAKG knowledge graph (Deng et al., 2021). Subsequently the use input, output pairs and human annotated content (instruction fine-tuning) to further boost their models answering capabilities in the field of geosciences.
LLMs however also exhibit clear biases in various geographic dimensions. As mentioned

above, LLMs abilities to recall facts tend to be more accurate for bigger countries or cities (Roberts et al., 2023). More geography specific biases could also be observed for LLMs. Fulman et al. (2024b) analyses four biases typical for human perception of geography and spatial relations: Hierarchical bias (locations are grouped with their respective administrative regions), proximity bias (distances within the same category are underestimated), rotation bias (rotating geographic features to fit cardinal directions) and alignment bias (aligning two geographic regions e.g. on the same latitude). They find that the LLM performs significantly worse for questions were one of these biases is supected, suggesting that LLMs tend to reproduce geographic biases found in humans (Fulman et al., 2024b). Beyond biases in geographic relation perception, LLMs also exhibit geographical biases in general topics like estimating temperature, precipitation, and population density, often performing worse in weaker economic regions, especially Africa. Also sensitive topics like the morality or intelligence of residents are biased, discriminating against poorer regions (Manvi et al., 2024).
In a recent study Z. Liu et al. (2025) formalize the the diversity of the geographic aspect of data, models and LLMs. They then evaluate six different LLMs through knowledge probing, measuring their geographic diversity. Their findings suggest "more geographic diversity for concept extension than for concept intension" (describe "a person" vs "people" from country X) (Z. Liu et al., 2025). Also they find varying geographic diversity dependent on the concept (more diversity when asking about "forests" then "people". Scale also plays a role since asking about a specific country rather than a continent resulted in higher diversity. Interestingly, they also observe less diversity in more advanced models, likely due to recent efforts in reducing hallucinations.

2.5.2 Geospatial Artificial Intelligence

Geospatail Artificial Intelligence (GeoAI) is much more than just the geospatial perspective on LLMs. The wide availability of large quantities of geospatial and spatio-temporal data (e.g. from regular earth observations, large crowdsourced datasets, progressivly higher granularity, etc.) constitutes the need for artificial intelligence models that can make sense of this data (W. Li et al., 2024). Geospatial analyses profit and build on more general AI research, however the aforementioned unique aspects of spatial data (e.g. spatial autocorrelation and heterogeneity) need to be taken into consideration when building GeoAI models (W. Li et al., 2024). This necessitates incorporating explicit spatial information in these models resulting in spatially explicit AI models (Janowicz et al., 2020). A model is spatially explicit if it has one of the following characteristics Janowicz et al. (2020); Goodchild (2001):

1. Invariance: Moving the studied phenomenon to a new location changes the outcome.

2. Representation: The model contains a spatial representation(s) of the studied phonemena like coordinates, relations or place-names.

3. Formulation: Spatial concepts like neighborhood, proximity, spatial autocorrelation, etc. are formulated in the model (see Kuhn (2012)).

4. Outcome: The spatial structure of the output differs from the input.

To create these models, GIScience domain knowledge and a deep understanding of spatial characteristics (such as the georeferencing, spatial resolutions and projections) are needed. Also the incorporation of multi-modal (text, vector, raster) and multi-source data presumes GIScience expertise (W. Li et al., 2024). W. Li et al. (2024) also report on the development of a new geospatial foundational model, Prithvi, and its possibilty to shift the motivation for model development from only commercial value to open science. Finally, they open the discussion to implement GeoAI as its own field of research, basing it on four pillars: Predictability, Interpretability, Reproducibility and Social Responsibility (W. Li et al., 2024).
These AI models can serve different purposes. In the domain of question answering and summarization, GeoAI models may need to take into account personal data of the one asking the question or for summarization they need to decide what consitutes a good summary (Janowicz et al., 2020). In social sensing, GeoAI models are able to digest the vast amount of personal data left online. This can be passive data like shared photos, posts that are often associated with spatial (meta)data or actively contributed data. Janowicz et al. (2020) also raise the point of transparent datasets and reproducibility in GeoAI research, advocating for well documented research, public source code availability and the sharing of data, while also acknowledged the additional burden on researches that often goes unrewarded.
In order for machine learning models to be trained, data needs to be encoded in a vector embedding space. For spatial explicit AI models, this needs to be done for geographic data. Encoders can preserve multiple spatial characteristics, such as the preservation of distance (close points have similar vectors) or Anisotropicity (directional awareness) and should be balanced between performance and the overfitting (Mai et al., 2022).
Janowicz et al. (2020) also formulate the moonshot: "Can we develop an artificial GIS analyst that passes a domain-specific Turing Test by 2030?". In the years passed since, significant progress was made towards this goal. With the gigantic progress in the domain of LLMs, new approaches were developed that go in this direction. Z. Li & Ning (2023) use an LLM at the core that generates a solution graph for a problem, implements the code for each operation and then puts the operations together producing figures and maps for problems like analyzing COVID-19 death rates, or human mobility patterns. Responding to this research, Akinboyewa et al. (2024) implement a "GIS

copilot" in QGIS that is able to analyze tasks, select cataloged GIS functions and tools from QGIS, generate code, handle errors and finally produce results. They examine three levels of difficulty, where the agent performed well for simple and intermediate tasks and lower for hard ones. Similarly, Mansourian & Oucheikh (2024) assess natural language based GIS analyses using a custom fine-tuned model for generating code that controls QGIS functions. They are also able to perform simple tasks like finding nearby locations in OpenStreetMap data or more sophisticated network analyses.

2.5.3 Geospatial semantic web

Geography is one of the most wildly represented domains in the semantic web community and within linked open data (McCrae, 2024). Kuhn et al. (2014) describe the role of GIScience in the context of the semantic web and linked open data. They argue that all data can be put into a spatial perspective by linking it to geographic datasets via predicates. Furthermore, metadata often has a spatial aspect which puts an additional spatial layer on many datasets. While a traditional database schema only describes the structure of the data, ontologies, as a core part of semantic web technologies, also describe the intended meaning of the data (Kuhn et al., 2014). On a more practical note, Janowicz et al. (2022) state that 80 % of a given projects resources are taken up by "gaining situational awareness", resulting in the so-called "data acquisition bottleneck". Furthermore, possible solutions like geo-enrichment services pose challenges like the isolated nature of their data sources. These challenges can be addressed using GeoKGs (Janowicz et al., 2022).
In order to realize this vision, the Open Geospatial Consortium defined the GeoSPARQL standard, an extension of SPARQL to harmonize the spatial domain with W3C standards on semantic web technologies (Car et al., 2023). This standard is then implemented into enterprise technology such as GraphDB to enable spatial functions directly within triplestores (Ontotext, 2024).
S. Wang et al. (2019) propose a formalization of geographic knowledge graphs, arguing a "well-structured geographic knowledge is a benefit to all kinds of geospatial applications, because formalization is the foundation of geospatial big data computing, mining, and visualization". They highlight the evolutionary nature of geographic phenomena such as storms and propose tactics to incorporate these into GeoKG design.
One of the most profound geospatial datasets is OpenStreetMap, a free, crowdsourced map of the world (OpenStreetMap, 2025). Dsouza et al. (2021) designed an approach to transform this dataset into a GeoKG, named WorldKG. They identify a lack of geographic coverage in general purpose knowledge graphs that they address by linking OpenStreetMap entities to other KGs. Their ontology provides semantics to the loosely defined key-value pairs in OpenStreetMap and maps them to DBpedia classes.

Furthermore, they also link geographic entities to their DBpedia entries and add provenance information. All of it is modeled according to OGC standards (Dsouza et al., 2021).
SOSA, the ontology for sensors, observations, samples, and actuators is another example of a geospatial ontology (Janowicz et al., 2019). As a joint effort by the OGC and W3C, it builds upon its predecessor, the SSN ontology, to model domains like the internet of things or environmental processes based on sensor observations and defined procedures using an event-centric perspective.
In an earlier publication Keßler et al. (2009) already propose integrating real-time sensor and user data for context-aware retrieval of geographic information. They highlight that OWL alone lacks support for free variables and overcome this limitation by leveraging the Semantic Web Rule Language (SWRL). By extending SWRL with a custom built-in function, they also enable real-time access to sensor observations. Through a case study focused on personalized surf spot recommendations, they demonstrate how dynamic environmental variables (e.g., weather, tide, wave height) and personal user attributes (e.g., surfing skill level) can be combined to support reasoning and dynamic reclassification of geographic entities.
Similarly, the KnowWhereGraph is a GeoKG that combines various data-silos like the USGS, NOAA or FEMA, using aforementioned technologies like RDF, GeoSPARQL or SOSA to provide a semantic layer, connecting these silos (Janowicz et al., 2022). Using a hierarchical global grid as a spatial reference system, the KnowWhereGraph allows for services built on top, e.g. geo-enrichment services for finding experts for disaster relief or retrieving disaster affected polygons (Z. Liu, Gu, et al., 2022) or supply-chain management in the food industry (Janowicz et al., 2022).
As mentioned in the previous chapter, data needs to be embedded into vector spaces in order to be used by (Geo)AI models. For GeoKG embedding Mai et al. (2020) raise multiple problems. Many KG embedding models ignore literals, omitting crucial information or are only populated sparesly from the beginning. They therefore propose a spatially aware KG embedding model that is also able to infer new relations using link prediction and is extended by spatial encodings for spatial objects in the graph.
To bridge the gap between GeoKGs and GeoAI two possibilities emerge. Either the data in the knowledge graph is encoded and used for training models directly using techniques summarized in Mai et al. (2022) and employed for example in Mai et al. (2020). Another possibility is to to automatically query the knowledge graph without using embeddings and ingesting the retrieved data directly into a LLM prompt. This is the approach used in this thesis.

3 Research Questions

The previous chapter presented three research directions that progressed rapidly during the last years. Large Language Models, semantic web technologies and their geospatial dimensions. Especially the rise of LLMs opened the door for applications in autonomous analyses, including in GIScience (Akinboyewa et al., 2024; Y. Jiang & Yang, 2024). However, these approaches are still missing the semantic layer. Already, recent publications explore and prove the value of combining KGs and LLMs using the GraphRAG approach, reducing hallucinations of LLMs and grounding them with external knowledge sources (X. Wang et al., 2023; Emonet et al., 2024). KGs structure knowledge but are more difficult to access or use, LLMs are easy to use but lack the structured, explainable knowledge and are regarded as black boxes (Pan et al., 2024; Procko & Ochoa, 2024). Therefore, combining these two technologies could unlock significant potential that remain underexplored in GIScience. While GeoKGs are already part of the pre-training step of some custom GeoLLMs (Deng et al., 2024), combining GeoKGs and LLMs using RAG remains a research gap. Furthermore, as pointed out by Shi et al. (2023), geospatial data requires disproportionately large models resulting in significant energy consumption. Also, the ever-evolving nature of geodata requires models to be retrained periodically, further exacerbating computing power requirements. Describing the data using ontologies and providing an LLM with these could circumvent this bottleneck, allowing for autonomous analysis on up-to-date data without direct integration into LLM or GeoAI model pre-training.
From a practical perspective, GraphDB and its GeoSPARQL integration offer the technical foundation to merge GIScience and GraphRAG on a larger and production-ready scale. Additionally, the GeoSPARQL integration also enables the use of geofunctions such as buffers, distances or topological relations, allowing for geoprocessing within GraphDB to infer new relations between entities.
If a LLM is capable to access and process (geo)data in KGs autonomously, the model does not need to be trained or retrained on the data. Furthermore, if the model can reliably make sense of ontologies describing data, new data sources can be made available to the model just by updating the ontology and connecting the endpoints.
The presented approach in this study uses data modeled with RDF in a KG in GraphDB and a custom ontology describing the data. Three geographic concepts are analyzed: topology, directionality and proximity. The hypothesis is that, using the correct set of parameters (e.g. the LLM, temperature settings, an optimized ontology, correct prompts, etc.) the GraphRAG approach can substantially outperform a non-RAG LLM for spatially explicit tasks. Two research questions are presented:

1. **RQ1** What are current limitations of non-RAG LLMs in performing spatially

explicit tasks, and how can these limitations be quantified using metrics such as precision, recall, and F1 score?

2. **RQ2** To what extent can a GraphRAG-based approach address limitations of non-RAG LLMs for spatially explicit tasks and what role do different parameters (LLM, temperature, prompt design, ontology richness, and few-shotting) play in improving results as measured by precision, recall and F1 score?

As the research and technical foundations for GraphRAG LLMs in GIScience are present, this study aims to clarify, if the approach using autonomous query generation based on a given ontology is a viable method for retrieving geodata from knowledge graphs. The following chapters describe the software and data used in this experiment.

4 Used Software and Data

4.1 Software

For combining and refining the different ontologies, Protégé was used. Protégé is "a free, open-source ontology editor" compliant with W3C standards, developed by Standford University and one of the most commonly used ontology management tools (Stanford University, 2025).
To store the data for the experiment and to run queries, GraphDB was used. GraphDB, developed by Ontotext, "is an enterprise ready Semantic Graph Database, compliant with W3C Standards". They provide a free version of their database that was used for this study, running on a local Windows machine (Ontotext, 2024). A major advantage of GraphDB is the native integration of GeoSPARQL that allows for using geospatial functions without requiring additional setup (Ontotext, 2025a).
OpenAI's GPT-4 models were used as LLMs. They provide a simple API for accessing their models. For this study, gpt-4o-2024-08-06, gpt-4o-mini-2024-07-18 and a custom fine-tuned model (see Section 5.6) were used. Additionally, OpenAI awarded 200 $ in funding for this project which covered all costs. The models were used to generate answers to define the experiment questions' baseline and to generate GeoSPARQL queries for comparing this baseline to a GraphRAG approach that are executed in GraphDB.
The code for this study was written in Python using a multitude of packages that are listed in the provided GitHub repository which also stores the full code to replicate this study (Groß, 2025). The different steps for this study are presented in nine jupyter notebooks. Python facilitates the studies infrastructure, connecting GraphDB, OpenAI, the custom ontology and the experiment questions and is used to analyze and visualize the results.
Even though the package LangChain was not used for its functionality, it provided prompt-templates for sending the RAG requests to the LLM (see Section 5.5) (LangChain, 2024).
Quantum GIS (QGIS), an open-source geographic information system, was used to get an overview of the projects data, perform pre-processing steps and to create some of the presented maps (QGIS, 2025).
Finally, Diagram Designer provided the ability to illustrate workflows and the custom ontology (see figure 7) (logicnet, 2023).

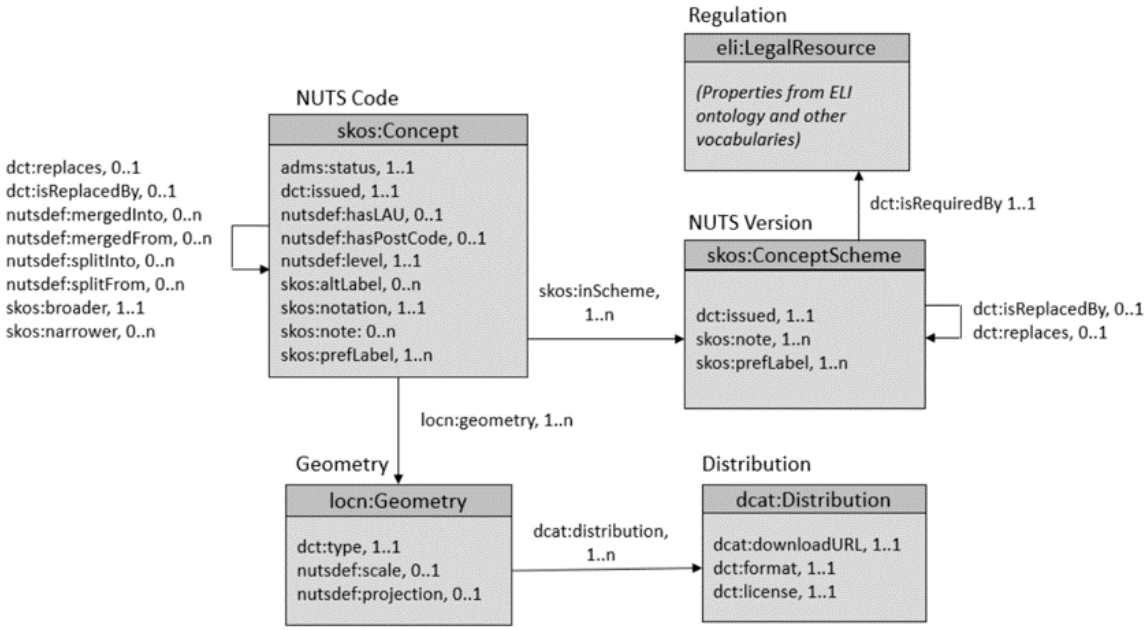


Figure 3: The ontology describing the NUTS dataset. Reused from Dekkers & Novacean (2018).

4.2 Data Sources and Cleaning

4.2.1 NUTS regions

The European NUTS regions (**N**omenclature of **T**erritorial **U**nits for **S**tatistics) are geographic divisions of the European Union's territory at three granularity levels (NUTS 1, NUTS 2, NUTS 3) (Eurostat, 2024). These regions are defined using RDF and are published on the website as a taxonomy in a .rdf file (Eurostat, 2024). The ontology of this taxonomy is defined by (Dekkers & Novacean, 2018), however the definition is only provided in the PDF document and not via an ontology file. The NUTS ontology is visualized in figure 3.

Apart from their availability as RDF, the NUTS region are the main geographic areas for statistical data on an European level and are therefore a suitable candidate for testing an AI approach for data-heavy question answering. The analytical nature of the NUTS regions, their code-based identified system and their relatively little presence in everyday life and media makes them arguably less represented in LLM training data and therefore harder to grasp for the model.

The NUTS regions were downloaded and stored in GraphDB, however there are certain attributes of the dataset that need to be addressed. The latest version from 2024 (NUTS regions are updated in a 3 year cycle) also contains deprecated regions. This includes for example all regions in the United Kingdom due to BREXIT, but also some NUTS level 2 regions in Portugal or level 3 regions all around the EU. These are due to

updates in the regions' definitions. During the time of writing, the spatial definitions of these changed regions are not yet provided with the RDF dataset, therefore the deprecated regions are part of the dataset. However they were excluded when populating the question with examples. A NUTS region subject in Turtle format looks like this:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
@prefix skos: <http://www.w3.org/2004/02/skos/core#>
@prefix nutsdef: <http://data.europa.eu/nuts/>
@prefix geo: <http://www.opengis.net/ont/geosparql#>

<http://data.europa.eu/nuts/code/AT13> rdf:type skos:Concept
 nutsdef:level "2";
 skos:notation "AT13";
 geo:hasGeometry <http://geonuts.eu/geometry/AT13_10m_ETRS89_geom
    >;
 skos:altLabel "Wien" .
```

For GraphDB to execute spatial functions, the subjects need to be compliant to the GeoSPARQL standard (Ontotext, 2025a). However, when loading the RDF file from Eurostat, the actual geometries of the NUTS regions are not provided directly. Instead they are provided via a link to their respective hosted GeoJSON files, with varying projections and scales of the geometries (see figure 3, `dcat:downloadURL`). A Python script was written that downloads the GeoJSON files from the link provided in the graph and transforms them into GeoSPARQL compliant `geo:Geometry` objects that are then attached to their respective NUTS region (for more details see Notebook 01, in the GitHub repository, Groß (2025)). An attached `geo:Geometry` object looks like this:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
@prefix skos: <http://www.w3.org/2004/02/skos/core#>
@prefix geo: <http://www.opengis.net/ont/geosparql#>

<http://geonuts.eu/geometry/AT13_10m_ETRS89_geom> rdf:type geo:
   Geometry;
 geo:asWKT "POLYGON ((4810026.3159 2802881.3099000007, ...,
    4810026.3159 2802881.3099000007))"^^geo:wktLiteral .
```

Furthermore, some smaller data engineering issues were handled. All data is stored in a GraphDB repository.

4.2.2 Geonames

Geonames is an openly available database of place names that ”contains over 25 million geographical names and consists of over 12 million unique features” (Geonames, 2024). Places can be geographical features like mountains or lakes, or man-made objects like monuments or cities. The database is categorized into 9 feature classes with 681 subdivisions (feature codes) (Geonames, 2025b). Geonames also maintains an ontology and the data is available in RDF format. Additionally, their data is available via an API. For this study however, only populated places like cities and villages were considered. These are available as a .csv dataset, ”cities500.csv”, that contains all places with at least 500 inhabitants (Geonames, 2025a). This table of populated places was then filtered to only contain places inside or near available NUTS geometries. The CSV file was then transformed into GeoSPARQL compliant triples using the same projection as for the NUTS regions. Similar to the NUTS data, further data-handling is described in Notebook 01 in detail. A Geonames `gn:Feature` can look like this:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
@prefix skos: <http://www.w3.org/2004/02/skos/core#>
@prefix geo: <http://www.opengis.net/ont/geosparql#>

<https://sws.geonames.org/2761369/> rdf:type gn:Feature;
  geo:hasGeometry <http://geonuts.eu/geometry/gn_2761369_geom>;
  gn:alternateName "Vindobona", "Wien";
  gn:geonamesID "2761369";
  gn:name "Vienna";
  gn:population 1691468 .
```

A major drawback that needs to be addressed is the data quality of the Geonames population values. These seem to be wrong or out of date. For example Munich only has around 1.2 million inhabitants in contrast to its actual population of around 1.5 Mio. (1.2 Mio. was at around 2005) or Vienna at 1.7 million instead of the current 2 Mio. (Wikipedia, 2025). Since population values are mainly asked in a comparative way in this study this drawback was not further considered.
Geonames cities and NUTS regions are downloaded, their coordinates are transformed into GeoSPARQL compliant `geo:Geometry` objects and are stored in a GraphDB repository. The exact definition of the data is defined by an ontology that is described in chapter 5.3.

5 Methodology

5.1 Experimental Setup

This work aims to analyze the potential of using the previously described semantic web technologies to advance the spatial capabilities of GraphRAG enhanced LLM systems. Two datasets that are already integrated into the semantic web are used. The European NUTS regions (Eurostat, 2024) and the Geonames city dataset (Geonames, 2024). The two datasets are stored in a local GraphDB instance. 18 questions about this data explore three different geographical concepts: topology/neighborhood, directionality and proximity. These questions are populated with concrete examples. Furthermore, for each of the 18 questions a ground truth SPARQL query is defined that is populated with the same examples and then executed to get the correct answers for each question. To answer **RQ1** the populated questions are sent directly to the OpenAI LLMs. The LLMs' result is the answer to the question which is then automatically evaluated using NLP techniques and compared to the ground truth results.
For the RAG approach the questions are augmented using prompt engineering techniques and a custom ontology that describes the data and associated functions. These are then sent to the OpenAI LLMs which then generate a SPARQL query that is cleaned and executed on the data in GraphDB. If the generated query yields results, these are analyzed for accuracy by comparing it to the ground truth results as well. The results of the RAG approach can then be compared to the non-RAG results to answer **RQ2**.
In both approaches the 18 questions are populated with many different examples and are affected by lots of parameters during the whole workflow. These parameters are described in the subsequent sections (different ontologies, language models and their temperatures, etc.). Each of them is recorded for each example question to evaluate their effects on accuracy. Figure 4 shows a detailed diagram of this workflow.

5.2 Investigated Spatial Concepts

To analyze the spatial capabilities of the LLM three different spatial concepts were chosen. Topology with a special focus on neighborhood relations, directionality which constitutes the four cardinal directions as well as the four intercardinal directions, and proximity. The three examples were chosen due to their relative simplicity and their possible implementation through geof: functions as well as their relevance in hypothetical use-cases. They are topology (with a focus on neighborhood), directionality and proximity. The three concepts are presented in the following chapters.

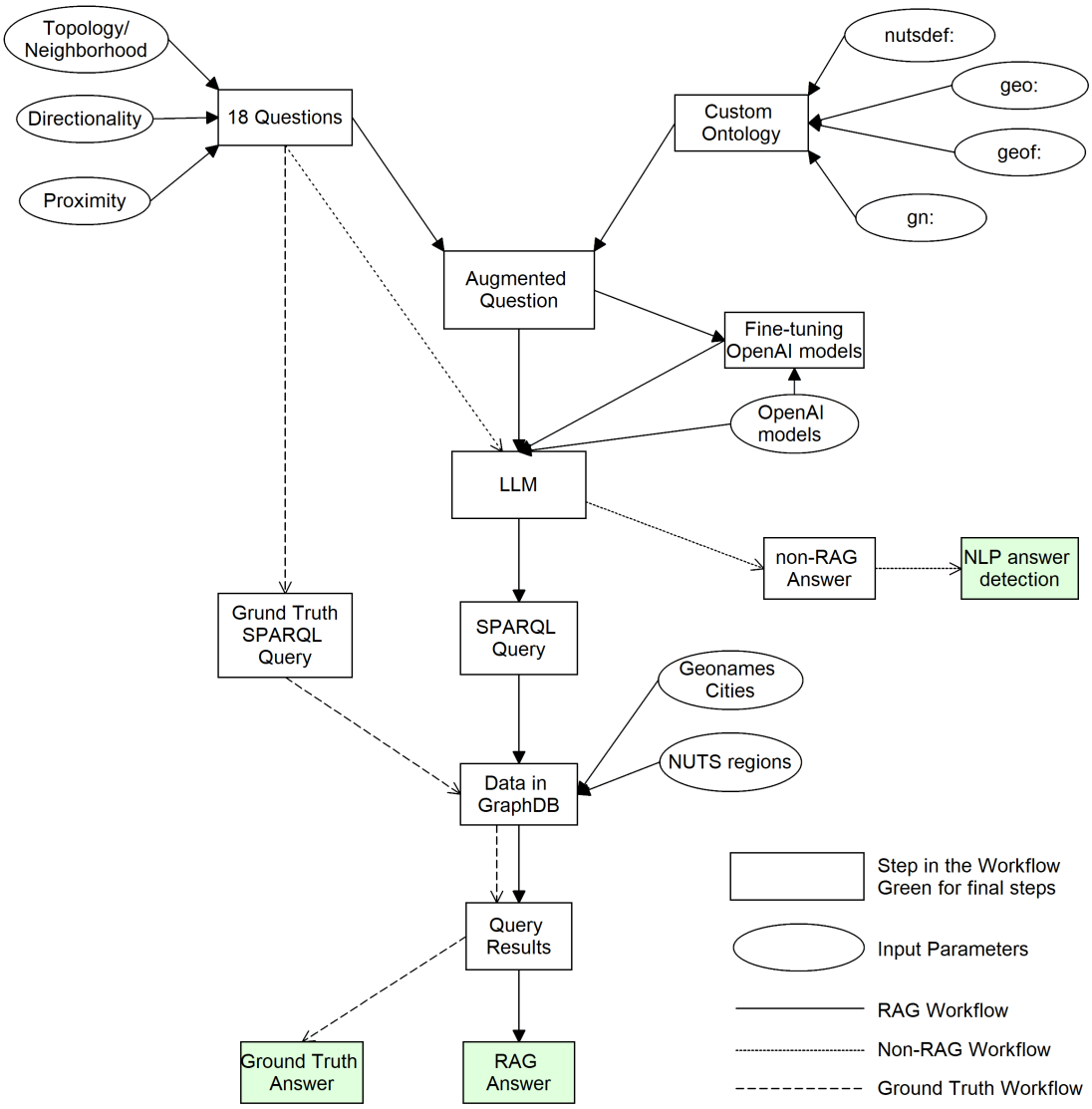


Figure 4: The workflow of the experiment depicted using Diagram Designer (logicnet, 2023). Final outputs are depicted in green.

5.2.1 Topology/Neighborhood

The first concept is containment. It evaluates whether the LLM can accurately determine if a specific city or NUTS region located within a given NUTS region. Neighborhood is another simple topological concept. Two regions are neighbors of each other if their boundaries touch in at least one point, however they are not if one regions is within the other. The proper RCC8 relation would be region 1 is externally connected to region 2 (QSRlib, 2025). Figure 5 shows an example of all regions that are neighbors of the region AT22 and have the same NUTS level. Neighborhoods are important spatial concepts. Possible use-cases are analyzing phenomena between neighboring regions, e.g commuting or migration patters, or calculating statistical concepts like spatial autocorrelation. If an autonomous LLM is prompted to perform such a task, it

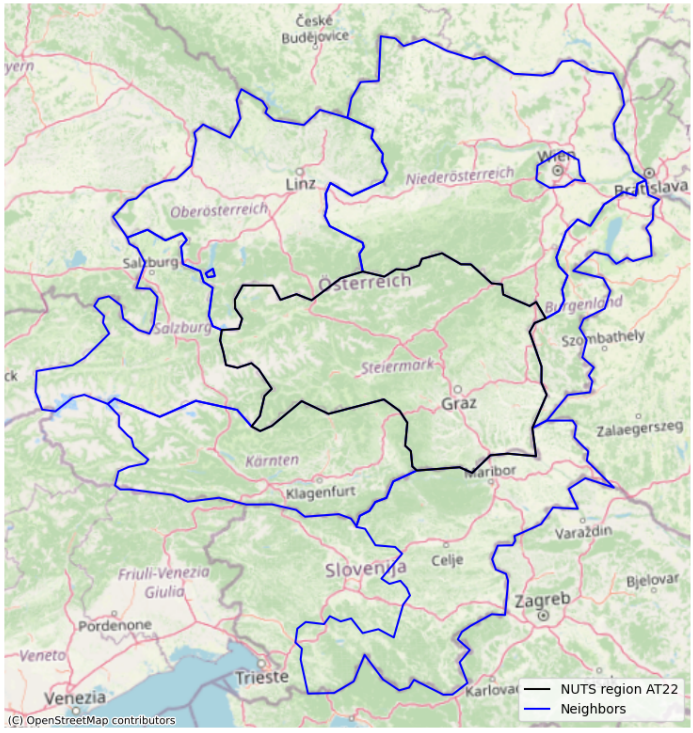


Figure 5: All neighbors of the region AT22.

must be capable to knowing what regions are neighboring each other. This work analyzes if a GraphRAG-based LLM, equipped with the necessary tools performs better in understanding these relations than without RAG.

5.2.2 Directionality

The second concept is directionality. This is represented by the cardinal directions "north", "west", "east" and "south" or the intercardinal directions "northwest", "northeast", "southwest" and "southeast". These are harder to define. Goyal & Egenhofer (2000) formally define directions and all their edge-cases. The definition of directions in this work are based on their findings, however the edge-cases (e.g. if a point has the exact same latitude as one of the latitude bounds of the region it is compared to) are not implemented since they are highly unlikely given the presented data. Let $G1_{minlat}, G1_{maxlat}, G1_{minlon}, G1_{maxlon} = bounds(Geometry1)$, which define the bounds of Geometry1 and $G2_{minlat}, G2_{maxlat}, G2_{minlon}, G2_{maxlon} = bounds(Geometry2)$ which define the bounds of Geometry2. The function $getDirection(Geometry1, Geometry2)$ returns the direction to which Geometry1 is from Geometry2. Equation 2 defines the

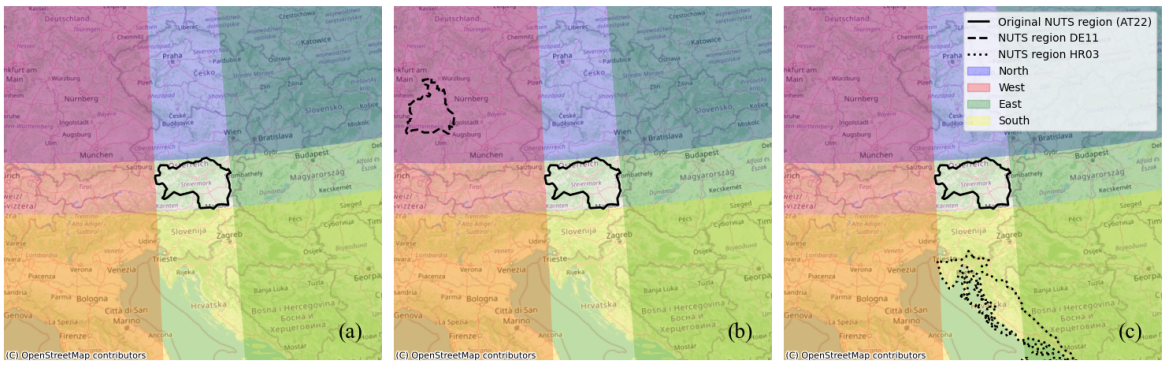


Figure 6: Directions defined from the Region AT22. Overlapping Polygons show areas of intercardinal directions.

directions formally.

$$direction(Geometry1, Geometry2) = \begin{cases} North, & \text{if } G1_{maxlat} > G2_{maxlat}, \\ West, & \text{if } G1_{minlon} < G2_{minlon}, \\ East, & \text{if } G1_{maxlon} > G2_{maxlon}, \\ South, & \text{if } G1_{minlat} < G2_{minlat}, \\ Northwest, & \text{if } North \wedge West, \\ Northeast, & \text{if } North \wedge East, \\ Southwest, & \text{if } South \wedge West, \\ Southeast, & \text{if } South \wedge East, \\ WithinBounds, & \text{otherwise.} \end{cases} \tag{2}$$

Figure 6 **(a)** shows the defined directions from the original region AT22 (Geometry2 in Equation 2). In Figure 6 **(b)** the Polygon in question (Geometry2, DE11) is within the northern and western parts of Geometry1. This results in $direction(DE11, AT22) = \{North, West, Northwest\}$. Analogous, the example **(c)** results in $direction(HR03, AT22) = \{West, East, South, Southwest, Southeast\}$. Here, even the tiny part of HR03 west of the western longitude bounds of AT22 is enough to be considered "West" of AT22. These cases highlight the counterintuitive situations that can result when formally defining concepts like directionality. However, in order to perform computational analyses a clear definition has to be provided for results to be consistent. Edge cases like this are very likely to not be caught by a non-RAG LLM and therefore highlight the necessity for this study. Calculating directions is not supported in GraphDB and is also not defined by GeoSPARQL standards. How this was implemented is described in section 5.3.1.

5.2.3 Proximity

The last investigated concept is proximity. In natural language, vague terms such as “near” or “close” are commonly used to describe spatial relationships. However, these expressions are inherently imprecise and challenging to define in a computational context. There are efforts to model fuzzy spatial concepts like vague cognitive regions using data-driven approaches (S. Gao et al., 2017), however these would be unfeasible to implement in the context of SPARQL and semantic web technologies. Therefore in this study proximity is simply defined by a certain distance threshold. The experiment questions therefore ask for clear distances like *”What is the largest city within 400 km of the NUTS region RO41?”*. These crisp boundaries are hard for an LLM to evaluate since it is not able to perform actual distance calculations or to even know the coordinates of most geometries. Therefore, an LLM might answer this question with a city that is even 500 km away, since it is not aware of the accurate distance. In GeoSPARQL the functions `geof:distance` or `geof:buffer` could be used to achieve this.

5.3 Ontology Development

As described in section 2.3.2 an ontology is a conceptual model of a domain. In this study, the ontology combines the domains modeled by Geonames, the NUTS regions and the technical parts from GeoSPARQL and GeoSPARQL geofunctions. It therefore describes the data and how it is presented in the knowledge graph. This ontology forms the core of the workflow of this study. It is modeled by combining the different existing ones (NUTS regions, GeoSPARQL, GeoSPARQL geofunctions and Geonames) into a new one while only considering the necessary parts of each. The created ontology will subsequently be referred to as **”geonuts”**. The geonuts ontology is given to the LLM in Turtle format together with each question. Only through the ontology, the LLM is capable of generating the respective (Geo)SPARQL queries. Since the ontology is being sent with every question, it has the be as small as possible to be efficiently grasped by the LLM. While common ontology Turtle serializations typically fall well within the context window of modern LLMs, pre-processing and experimentation revealed that query generation becomes significantly more feasible and reliable when irrelevant information is minimized. This is why only the necessary parts of the ontology are actually part of the final Turtle file. Code 2 (Appendix) shows this Turtle file, figure 7 shows the final version in a visual way. It consists of five parts.

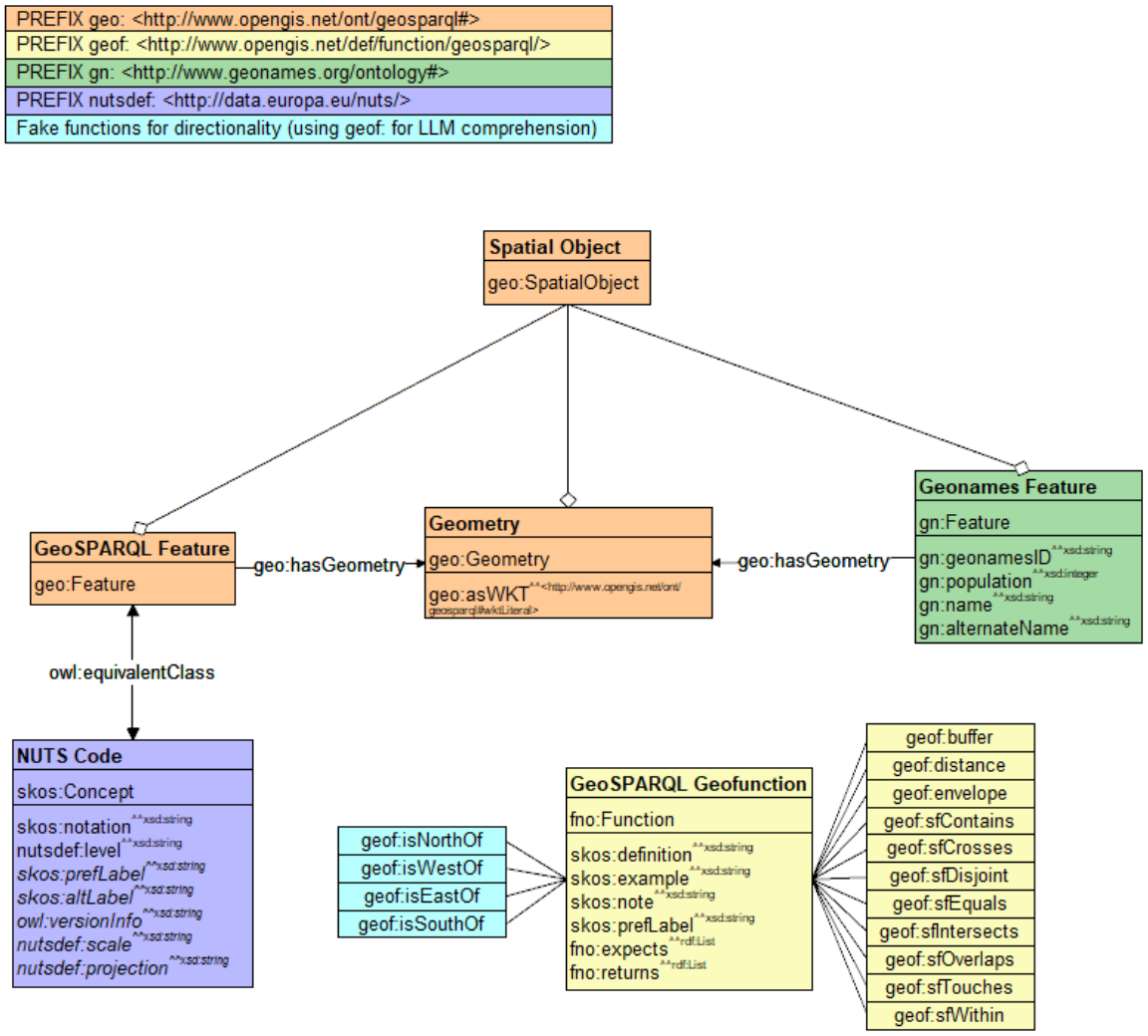


Figure 7: The geonuts ontology. The different colors show the different ontologies that this ontology borrows from. Each rectangle represents a `owl:Class`, the text in **bold** the `skos:prefLabel` or `rdfs:label` and the IRI below the `rdf:type` of the class. Below are the `owl:DatatypeProperties`s for this class. The "fake" functions and the alignments of the ontologies are developed for this study. Own depiction in Diagram Designer.

5.3.1 Ontology Components

GeoSPARQL. GeoSPARQL (prefix geo:, depicted in orange) represents the technical part of the ontology. It defines the class `geo:Feature` that is a subclass of `geo:SpatialObject`. Also `geo:Feature`s have an object `geo:Geometry` that stores the actual coordinates via a literal with the data type `geo:wktLiteral`.

NUTS. Dark blue represents the part taken from the NUTS ontology. Figure 3 shows the original extent of the NUTS schema. Only the relations and attributes defined in figure 7 were used in the geonuts ontology. The actual geographic data that is provided in figure 3 via the `dcat:downloadURL` relation is transformed into a `geo:Geometry` object. Therefore NUTS Codes are an `owl:equivalentClass` to `geo.Feature`. The NUTS ontology is modeled mainly using the skos: namespace, their own attributes are using the prefix nutsdef:.

Geonames. Parts taken from the Geonames ontology are represented in green. Here only the `gn:Feature` class and five attributes were used for this experiment as they are of interest in the example questions. The original Geonames ontology can be found on their website (Geonames, 2025c). It uses the prefix gn:.

Geofunctions The functions used in this study are the GeoSPARQL geofunctions in yellow using the prefix geof:. The GeoSPARQL geofunctions are also defined by the OGC, however only the depicted subset of functions were actually used in the study. The chosen functions are supported in GraphDB by enabling the GeoSPARQL plugin. For this work the eight simple feature functions for topological relations (`geof:sf[FUNCTION]`) as well as `geof:buffer`, `geof:distance` and `geof:envelope` (equivalent to a bounding box) were chosen. All questions asked in the study can be answered with these functions. An example usage of a GeoSPARQL function is provided in code 5 and code 6 (both in Appendix).
Apart from geofunctions, GraphDB also provides magic predicates for some relations, e.g. one could ask for `?geometry1 geo:sfTouches ?geometry2` even if the `geo:sfTouches` relation does not exist between the geometries. The reason why these were not used in this work is described in section 5.3.5.

"Fake" Functions To investigate directionality, there are no functions yet defined by the OGC. Therefore, four "fake" functions were imagined that provide this functionality. These functions are defined in the geonuts ontology under the same geof: prefix to ensure easy understanding for the LLM. In the ontology they use the same attributes as their real counterparts. These functions are `geof:isDIRECTIONOf(geo:wktLiteral,`

`geo:wktLiteral)` where $DIRECTION \in [North, South, East, West]$. The function returns "$true$"^^$^{xsd:boolean}$ if the first input is to the $DIRECTION$ of the second input. If a LLM generates a SPARQL request using one of these functions. They are replaced using a Python script before being executed. Virtual polygons are calculated using the bounds of the second attribute in geographical coordinates (EPSG 4326). For example when analyzing if the NUTS region "ITF" (SUD, southern Italy) is south of the NUTS region "SE3" (Norra Sverige, northern Sweden), the lower latitude bound of "SE3" is extracted and transformed into polygon spanning everything that has a lower latitude than the southernmost point of "SE3". The resulting polygon is then reprojected back into the EPSG 3035 projection to be compatible with the coordinates in GraphDB and is then used to perform an intersection with the "ITF" polygon. If this intersection returns true, "ITF" is south of "SE3". The process is visualized in figure 8.
On a SPARQL level, the line `geof:isSouthOf(?itfWKT, ?se3WKT)` is simply replaced with `geof:sfIntersects(?itfWKT, ?southWKT)` and the calculated polygon is bound as a `geo:wktLiteral` beforehand using `BIND("POLYGON((..))"geo:wktLiteral as ?southWKT`. The implementation of this function can be found on this work's GitHub repository (Groß, 2025). While this is not a native function in GraphDB or the geof: ontology this approach was used to illustrate the potential of using more geospatial functions with SPARQL even if they are not implemented yet.

5.3.2 Ontology Alignment

As described in section 2.3.2, different ontologies can be combined using ontology alignment. In the case of geonuts, only the relevant parts have to be aligned. First, `geo:Feature` and `gn:Feature` can be considered equivalent classes. This is also clearly defined by the OGC in the GeoSPARQL standard specification (Car et al., 2023). Therefore, `gn:Feature` is also a subclass of `geo:SpatialObject` and can use the same relation `geo:hasGeometry` to attach a `geo:Geometry` object. It was however not explicitly stated in the ontology and the GraphDB instance that `gn:Feature owl:equivalentClass geo:Feature`. This is because these elements have to be distinguished by a generated SPARQL query and within the database, so that when querying for a `gn:Feature`, no `skos:Concept`s get queried. When actually setting this equivalence in the ontology, the LLM-generated query will often confuse the two classes, hence it was left out.
The NUTS Codes however were aligned with the `geo:Feature` class since if it is only one of `gn:Feature` or `skos:Concept` it does not matter. This step was mainly taken to clarify the spatial dimension of the NUTS codes for the LLM, if this is really necessary is a matter of discussion.

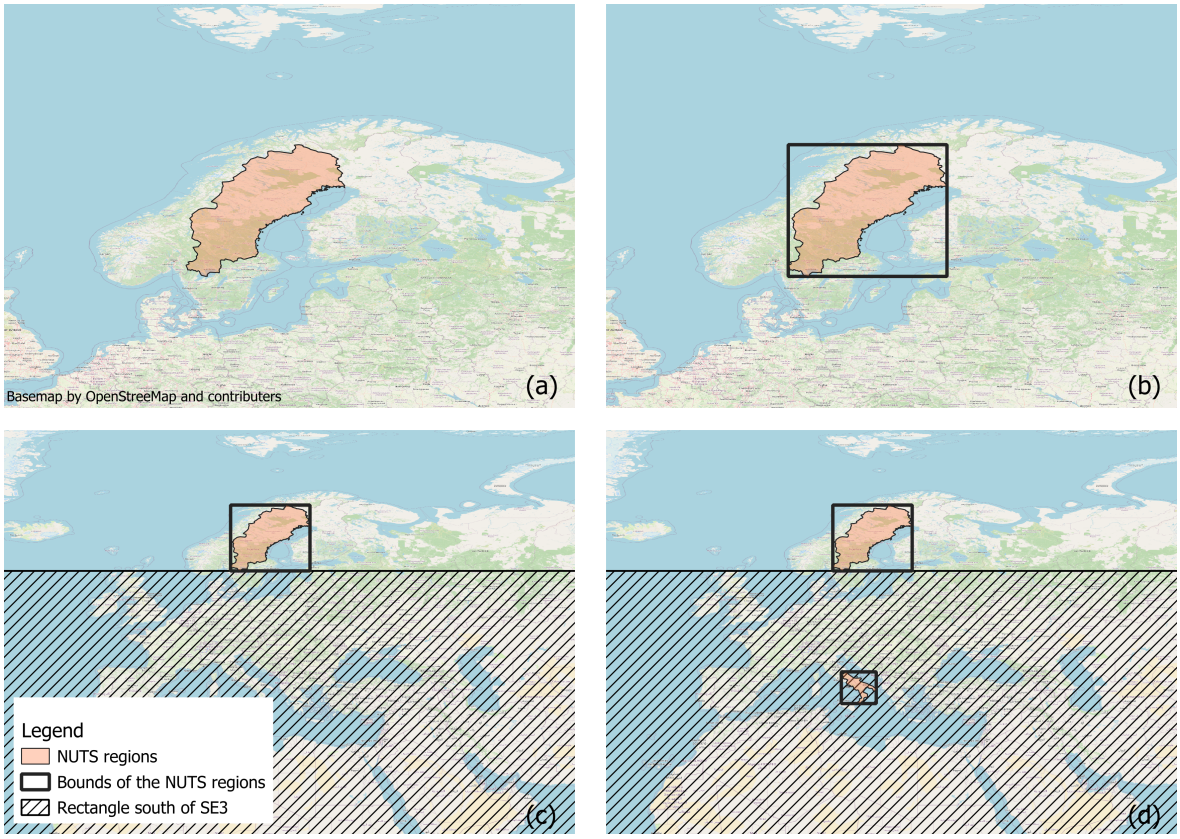


Figure 8: The figure shows the process of defining one of the direction polygons that is later used to perform an intersection with. First the bounds of the selected geometry **(a)** are extracted based on geographic coordinates **(b)**. Then, depending on the direction (here "south") a rectangle is created that encompasses everything with a lower latitude then the minimum latitude of the geometry **(c)**. This rectangle is then projected back into the EPSG 3035 coordinate system and used with `geof:sfIntersects` to assess if the other polygon is within this rectangle **(d)**. Maps made in QGIS (QGIS, 2025).

For the LLM to use GeoSPARQL geofunctions, it must understand their syntax, what parameters they expect and the data types of those parameters. These functions are typically defined as instances of `fno:Function` according to the Function Ontology (De Meester et al., 2020). Here, each function references a `rdf:List` of input and output parameters using the `fno:expects` and `fno:returns` predicates. Each parameter must then be separately described as a subject with additional triples. In Turtle this would be modeled as follows (OGC, 2024):

```
PREFIX geof: <http://www.opengis.net/def/function/geosparql/>
PREFIX geofp: <http://www.opengis.net/def/function/parameter/
    geosparql/>
PREFIX geofo: <http://www.opengis.net/def/function/output/geosparql
    />

geof:sfWithin
    fno:Function;
    fno:expects (geof:sfWithin_param1 geof:sfWithin_param2 ) ;
    fno:returns (geof:sfWithin_output ) ;

geofp:sfWithin_param1 a fno:Parameter ;
    fno:type geo:wktLiteral, geo:gmlLiteral, geo:geoJSONLiteral,
        geo:kmlLiteral, geo:dggsLiteral ;
    fno:required "true"^^xsd:boolean ;
    skos:prefLabel "geom1"@en .

geofp:sfWithin_param2 a fno:Parameter ;
    fno:type geo:wktLiteral, geo:gmlLiteral, geo:geoJSONLiteral,
        geo:kmlLiteral, geo:dggsLiteral ;
    fno:required "true"^^xsd:boolean ;
    skos:prefLabel "geom2"@en .

geofo:sfWithin_output a fno:Parameter ;
    fno:required "true"^^xsd:boolean ;
    fno:type xsd:boolean .
```

Using these clear and compliant, but arguably verbose, definitions proved to be rather difficult for the LLM to understand. Especially when defining multiple functions using this approach, they would significantly increase the size and complexity of the ontology. To address this, the geonuts ontology simplifies the representation by directly assigning the expected and returned datatypes as literals using the `fno:expects` and `fno:returns` predicates:

```
geof:buffer a fno:Function ;
    fno:expects "(geo:wktLiteral xsd:double)" ;
    fno:returns "(geo:wktLiteral)" .
```

This omits some information, but shortens the definitions significantly and ultimately enables the LLM to generate better SPARQL queries: It is acknowledged that this modifies the original GeoSPARQL geofunction definitions and violates domain/range specifications for `fno:expects` and `fno:returns`, however after thoroughly testing different options, this turned out to be the most promising approach. The design of the ontology was an iterative process that is too extensive to be described here in all detail. The crucial point is that the geonuts ontology was not only designed to model the domain of geographic data but to be as understandable as possible for an LLM to generate correct SPARQL queries. Therefore, it was important to keep it as concise and simple as possible. The following chapter explains how the ontology was systematically reduced and prepared for LLM compatibility.

5.3.3 Ontology Adjustments and Pruning

As visible in figure 7 only a small subset of the original ontologies was kept. This was performed by loading all original ontologies (apart from geof:) into Protégé and manually removing all unnecessary components. The geof: ontology was loaded into Python (see Notebook 03 for details, Groß (2025)). Here all irrelevant parts were removed and the `fno:expects` and `fno:returns` relations were changed. Additionally, two predicates were added for each geof: function. A `skos:note` to redundantly clarify that the function needs a `geo:wktLiteral` as input and a `skos:example` that provides an example of how the function can be used within a SPARQL query. This is necessary since by far the most common mistake of LLM query generation turned out to be using the `geo:Geometry` objects as inputs for functions instead of the `geo:wktLiterals`. A similar `skos:note` was also added for the `geo:Geometry` subject. The exact workflow is visible in Notebook 03 (Groß, 2025). The final ontology serialized in Turtle format is provided with code 2 in the Appendix.

5.3.4 Multiple Ontology Versions

For the experiment three different versions of this ontology were used. These however do not differ semantically but only in how much information was provided with each component. The first ontology (V1) is the one described in the section before (see code 2), the second ontology (V2) only differs slightly and removed a few less important relations for this study, namely `owl:disjointWith`, `rdfs:subPropertyOf` and

skos:historyNote.
Ontology V3 removes all natural language from the ontology except for labels. This still defines all important aspects of the ontology but shortens it significantly. The experiment with V3 provides insights in how much the LLM relies on descriptions in natural language for understanding ontologies. Also the previously described tips for using geo:wktLiterals are not provided in this version.

5.3.5 Why Use Functions instead of Magic Predicates?

As described in Section 5.3.1, magic predicates for geospatial relations were not used. This has multiple reasons. First, the ontology needs to be as simple and short as possible for the LLM to be able to understand it. When using geo: magic predicates for simple feature relations (geo:sfTouches, geo:sfWithin, etc.) that use geo:Geometry objects as domain and range and geof: functions that use geo:wktLiterals as attributes, the LLM has problems to accurately and consistently differentiate between these nuances.
Second, the functions take a geo:wktLiteral as an input for geographic data. These functions can work with WGS84 coordinates when the projection URI https://www.opengis.net/def/crs/OGC/1.3/CRS84 is specified in the geo:wktLiteral before the WKT string (for more details see GraphDB documentation, (Ontotext, 2025a)). However, at the time of writing, the function geof:buffer does not work with geographic coordinate systems (alanr, 2024). To circumvent this problem, this study was performed using the EPSG 3035 projected coordinate system for Europe. This way, geofunctions can be used without projection issues. Magic predicates on the other hand always use a spatial index that is derived from WGS84 coordinates. To avoid projection issues and to make everything easier to understand for the LLM the whole study was therefore performed using exclusively geof: geofunctions as geo:wktLiterals in EPSG 3035.
A third reason is consistency for future analysis. If performing more sophisticated spatial (or non-spatial) queries using functions in SPARQL, magic predicates might not be implemented (as for example with the direction functions), the function needs more than two inputs or the output is not a boolean. To make the results more valuable for a broader spectrum, functions instead of magic predicates were used.

5.4 Experiment Questions

Based on analyzed datasets, the spatial concepts to be investigated, and the created geonuts ontology, a set of 18 questions was created. These question are only about facts that are answerable with the provided dataset and the ontology. The questions

are grouped into six categories: simple topology, neighborhood, boolean directionality (questions that are answerable with true/false), open directionality (questions where the answer is an (inter)cardinal direction), proximity and combinations of different categories. The full list of questions can be found in the Appendix (table 15). One example is a simple question checking a neighborhood relation *"What regions of the same level are neighbors of the NUTS region CODE?"*, a proximity relationship *"What is the largest city within BIGDISTANCE km of the NUTS region CODE?"* or a combination of multiple aspects *"What NUTS regions share a border with the region CODE in the DIRECTION on the same nuts level?"*. The parts in capitals (except for NUTS) are placeholders, that are filled with actual examples from the database.
The placeholders are CODE (a NUTS code like "DED43"), CITY (the name of a city in the Geonames dataset within or near the borders of all NUTS regions, e.g. "Vienna"), DIRECTION (a cardinal direction like "north", etc. or an intercardinal direction like "northwest", etc.), BIGDISTANCE (everything between 200 and 700 in a step of 50 followed by km) and CCONDITION (a short sentence restricting the size of cities requested, like *"that have more than 120000 residents"*).
The questions were populated in six different ways consisting of different levels of the chosen NUTS codes, different size minima for chosen cities and asking for cardinal or intercardinal directions. The rest of the placeholders were populated randomly from a given selection. Notebook 05 shows this process in detail. A populated question could be *"What NUTS regions share a border with the region DE3 in the east on the same nuts level?"* These populated questions were then either sent directly to the LLM for the non-RAG experiment or embedded into a prompt for analyzing the RAG capabilities. The following chapter describes these prompts.

5.5 Prompt Engineering and Few-Shot Training

While prompt engineering is crucial for developing RAG-LLMs it is important to note, that this was not the main part of this work. When prompting an LLM there is the system message and the user message. The system message sets the context for the LLM and normally is something similar to *"You are a helpful assistant"*. The user message is then the actual prompt the user is sending. For the non-RAG experiment only a system messages was created, since the user message is the populated questions. This was the created system message:

```
You are a helpful assistant that answers questions about the European NUTS regions
    and cities in Europe.
The user will ask questions about topological relationships between the NUTS
    regions and the cities.
If the question asks for NUTS regions your answer should ONLY contain ALL the
```

applicable NUTS codes, for example "DED, DED4, DED43" or "PL91"
If the question asks a true or false question answer ONLY with "true" or "false".
If the question asks for a direction your answer should ONLY contain the applicable direction, for example "northwest" or "south".
If the question asks for a city or multiple cities your answer should ONLY contain the name of the applicable city or ALL applicable cities, for example "London" or "Weilheim i. OB, Deutenhausen, Marnbach".
Do not provide any explanations or apologies.

The prompt sets the context for the task and provides accurate instructions and examples on how the answer should look like.
For the RAG experiment, four different system messages and two different templates for the user messages were created. For the first two system messages the first user message template was used and for the second two system messages the other user message template was used. The main difference between the templates is that one specifies that the LLM should write a "SPARQL SELECT" query (this is the original message from LangChain) while the second version only specifies a "SPARQL" query, leaving the option for the LLM to generate "ASK" queries for true/false questions. This is because during experimenting, it was noticed that the LLM occasionally generates ASK queries and therefore should not be arbitrarily restricted from doing so. These messages and their respective system messages can be found in the Appendix (code 3) or in the Notebook 06. The system messages tells the LLM to translate natural language questions about geospatial relations (the populated questions) into SPARQL queries based on a given ontology. Three of the four messages provide an additional hint to the LLM to use the geof: functions with the `geo:wktLiteral` instead of the `geo:Geometry` object. The user message then inserts the whole ontology in Turtle format and the populated question into the prompt template where it is also explained in detail how the answer should look like. The user message therefore contains the whole ontology every time which makes the message quite long. This template was copied from the provided templates in the LangChain Python package (LangChain, 2024).
As explained in Section 2.1, few-shotting provides the llm with some examples of the answer before asking the actual question. For this approach new system and user message templates were created (see code 4, Appendix). Then two examples were provided where a question is asked and the correct SPARQL query is provided. To avoid sending the whole ontology multiple times the ontology is embedded in the system message this time, while the populated question is still part of the user message. A whole example of a few-shot setup is also provided in code 4 (Appendix).

5.6 Models, Temperature and Fine-Tuning

Apart from the different ontology versions and prompting techniques two parameters lie within the actual language model. First the temperature, explained in Section 2.1, is defined. Five different values were analyzed in this study: 0, 0.33, 0.66, 1 and 1.33. Higher values are presumed to not yield meaningful results. The preselected value for the public ChatGPT interface is 0.7, for generating code lower temperature are generally recommended all across online tips and forums. Another parameter is the actual models. Only models by OpenAI were analyzed, since they provided a research grant for this study to use their models free of charge. The models are "gpt-4o-2024-08-06", now referred to as gpt-4o (the date has to be specified since OpenAI can point the "gpt-4o" parameter to newer versions in the future) , "gpt-4o-mini-2024-07-18" now referred to as gpt-4o-mini and a custom fine-tuned model based on "gpt-4o-mini-2024-07-18" that will now be referred to as mini-fine-tuned.

Gpt-4o is OpenAI's broadly available flagship model apart from the newest reasoning models like o1. It provides a 128,000 tokens context window and over 200 billion parameters. Gpt-4o-mini is a much smaller, faster and cheaper version of gpt-4o. More details on the models are available on OpenAI documentation pages (OpenAI, 2025).

The mini-fine-tuned model was trained on a custom selection of 12 questions that are quite similar to the 18 study questions. For these 12 questions, the correct SPARQL queries were defined manually. Then for each question 2 alternative wordings were defined that are semantically identical. Therefore 36 question-answer pairs were created (see Notebook 04, Groß (2025)). The questions have the same structure as described in section 5.5 using a simplified system message, the same user message and the custom ontology V1. These are saved in a .jsonl file which is uploaded to the OpenAI dashboard. Here, using the gpt-4o-mini model as a base model, the dataset is selected and the fine-tuning is performed using the preselected parameters. The process can be inspected in Notebook 04. It is important to note that the creating a fine-tuned model is not the main scope of this study. Therefore, the process is not rigorous and has lots of potential for improvement. Furthermore, there is a possible issue of overfitting, since the questions in the training dataset are quite similar to the ones in the experiment. Nonetheless, the results have some interesting implications that should not be withhold.

5.7 Details of Experiment Execution

The final question dataset is constructed as follows. There are 18 base questions $N_Q = 18$. Then there are 5 further parameters that affect LLM generation: six experiment conditions $N_E = 6$ (see Section 5.4), three ontologies $N_O = 3$, three models

$N_M = 3$, five temperatures $N_T = 5$ and four different system messages $N_{SM} = 4$ with two user messages dependent on the them. These constitute $N_{noFewShotting}$. For the number of experiments using few-shotting ($N_{fewShotting}$) the system messages were different. To achieve similar group sizes, the number of few-shotting examples is defined to be equivalent to one user-message group (or two system-messages groups). The total number of experiment questions can therefore be defined as follows:

$$\begin{aligned} & N_{Experiments} = N_{NoFewShotting} + N_{fewShotting} \\ =& (N_Q * N_E * N_O * N_M * N_T * N_{SM}) + (N_Q * N_E * N_O * N_M * N_T * 2) \\ =& (18 * 6 * 3 * 3 * 5 * 4) + (18 * 6 * 3 * 3 * 5 * 2) \\ =& 29160 \end{aligned} \tag{3}$$

In total, 28668 example questions were analyzed that were organized in a dataframe in Python (See Notebook 05 and Notebook 06). The remaining questions were removed due to projection issues leading to inaccuracies for bounding box creation (see section 7.4.3). All these questions were populated with examples after multiplying them for the different parameters to achieve a maximum variety in analyzed NUTS regions and cities. Each populated question was transformed into an API request to OpenAI using the respective parameters. Through the randomness of populating the questions the same question might be asked multiple times using different parameters. The generated SPARQL queries are then cleaned for basic generation mistakes. This includes for example checking if the prefixes are defined correctly or if there are no leading triple backticks (```) in the beginning or end of the generated answer. The generated queries that contain a `geof:isDIRECTIONOf(?wkt1, ?wkt2)` function call are replaced with an intersection between the first parameter of the function (?wkt1) and a polygon representing the direction in question based on the second parameter (?wkt2) (see section 5.3.1). The SPARQL queries are automatically sent to GraphDB. Finally, it is recorded, if the query can actually be executed and if that is the case, the answers from GraphDB are saved for each of the $N_{Experiments}$ questions.
Since the semantics of a questions do not change when changing the parameters, ground truth SPARQL queries only have to be defined for the 18 original questions with the same placeholders. These queries can then be populated similarly to the original questions resulting in a ground truth query for each of the $N_{Experiments}$ questions that is also executed in GraphDB. This gives the ground truth results to evaluate the RAG and non-RAG based answers (see Groß (2025) /functions/correct_queries.py).
For the latter all $N_{Experiments}$ questions are sent directly to the LLM with the system message described in Section 5.5. However, all the other parameters are ignored except for the model and the temperature since they are also relevant for the non-RAG approach. Since the resulting answers are not a query that can be executed and are

therefore not structured, rule-based natural language processing is applied to detect the relevant NUTS codes, cities, directions or true/false statements in the answer. For detecting cities, a generated answer is split into different cities, then it is cleaned for any special characters and finally it is compared to all the cities' labels (and if necessary alternative labels) in the geonames dataset. For example for the question *"What are cities within 8 kilometers to the west of the NUTS region DEA19?"* the generated answer was "Essen, Mülheim an der Ruhr, Oberhausen". This answer was split and cleaned of special characters, resulting in [oberhausen, mülheimanderruhr, essen]. Comparing these strings to the cleaned `gn:name` and `gn:alternateName` values in the geonames dataset yields a match for oberhausen and essen for a `gn:name` and mühlheimanderruhr for a `gn:alternateName`.
The details can be inspected in the GitHub repository (Groß (2025) Notebook 07). Finally, the RAG answers and non-RAG results can be compared to the ground truth.

6 Results

6.1 RQ1: What are current limitations of non-RAG LLMs in performing spatially explicit tasks, and how can these limitations be quantified using metrics such as precision, recall, and F1 score?

Technical Variables Table 1 shows accuracy values for the non-RAG experiment depending on the two relevant technical parameters (model and temperature). The mini-fine-tuned model is omitted, since it is fine-tuned to generate SPARQL queries and not to simply answer the questions. Furthermore, the hallucination rates for questions with NUTS regions in their answers are shown. This rate could only be assessed for NUTS regions and not for cities since it is difficult to assess which words are meant to be cities. The results show peak F1 scores of 0.37 for the gpt-4o model with a temperature setting of 0, while the gpt-4o-mini model achieves a maximum of 0.31. Interestingly, the scores do not drop much with higher temperatures which could be explained by the clear system message restricting creativity and demanding short answers. Similarly, the hallucination rates for NUTS codes are lower for the bigger model, but do not get much worse with higher temperature values.

Model	Temperature	F1	Precision	Recall	Hallucination Rate
gpt-4o	0.00	<u>0.37</u>	<u>0.43</u>	<u>0.37</u>	0.10
	0.33	0.36	0.42	0.35	0.11
	0.66	0.33	0.39	0.32	0.07
	1.00	0.32	0.40	0.31	0.08
	1.33	0.31	0.38	0.30	0.10
gpt-4o-mini	0.00	0.31	0.35	0.30	0.20
	0.33	0.30	0.35	0.29	0.19
	0.66	0.29	0.34	0.28	0.19
	1.00	0.29	0.35	0.29	0.19
	1.33	0.28	0.34	0.28	<u>0.25</u>

Table 1: Result table for the non-RAG experiment depending on model variables.

Three Examples Table 13 (Appendix) shows three examples of generated answers (model=gpt-4o, temperature=0). The first one is a simple true/false question where the model wrongly identified CZ06 to be northwest of Bayreuth. This failure is already quite significant since Bayreuth is a fairly big German city and the Czech Republic is clearly to the east of (most of) Germany without any hierarchical or rotational bias suspected. The second example is a simple, even redundant, question since there is only the region itself inside the NUTS region RO321 (level 3 NUTS regions are the smallest level). The LLM correctly identified this without being confused by the triviality of the question.

Question three requires a more complex answer, listing all cities within a distance of 450 km of the region AT3. Figure 9 shows the answers on a map. It is clear that the LLM missed quite a few cities. While it got almost all fitting cities in south-western Germany and Switzerland as well as many in Italy and Slovenia right, it clearly had problems identifying the fitting cities in Eastern Germany, Czech Republic and Poland as well as on the outer parts of the buffer. Most notably it also missed Prague, Budapest and even Vienna, three major capitals, one of the country containing the NUTS region in question! It did however not miss any city within the actual region.

Another interesting observation is that there are no false-positives outside the buffer, only due to the population criteria of at least 120,000 inhabitants (Klagenfurt, Mulhouse, Nancy all have less than 120,000 inhabitants when defined by city population, not agglomeration, but were listed by the LLM). Although this is not reflected when looking at all examples where false positives are prevalent (see figure 10).

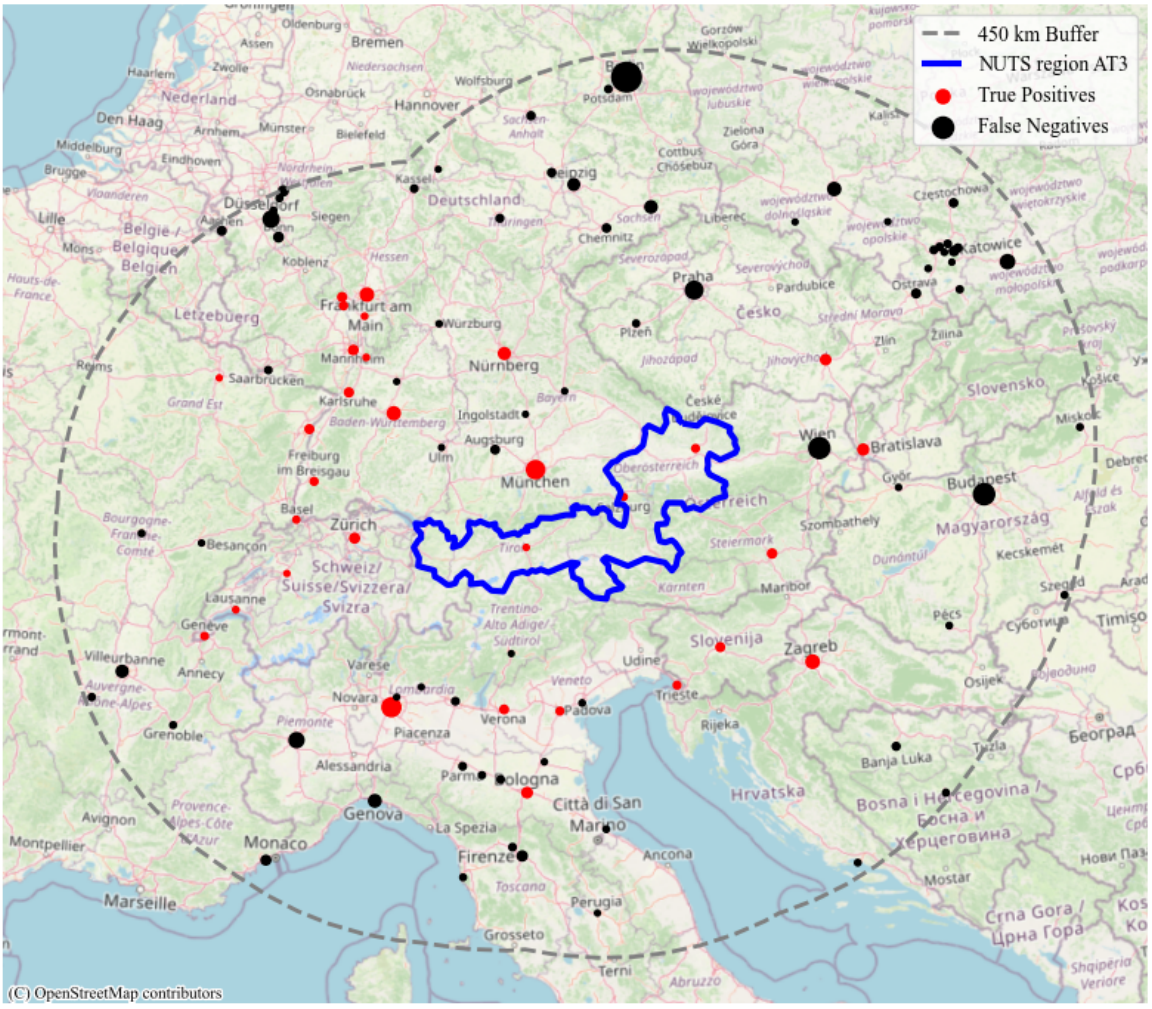


Figure 9: This map shows the answers to the question "What are all cities with more than a 120 k people and are less than 450 km from the NUTS region AT3?". The cities sizes depend on their population. See text for an interpretation of the results.

Performance by Distance for City in Buffer Question This example can also be visualized in a more general way. Figure 10 shows the distribution (Kernel Density Estimation) of true positives, false positives and false negatives for the question "What are all cities that are bigger than 99,999 and are less than 250 km from the NUTS region CODE?" where all NUTS region are of level 2. Figure 10 **(a)** shows this distribution for all examples, **(b)** only for the gpt-4o model with temperature values below 0.5. The plots clearly show that there are more false negatives (cities that the model missed) right before the 250 km limit. The false positives show a bump right around the limit as well as shortly before 500 km. Also there are some false positives that are very far away from the actual limit. There are also interesting differences between the two figures. First, when choosing the better model parameters (gpt-4o and temperature < 0.5), the true positive values are more evenly distributed across the 250 km space, meaning that it can identify cities more evenly within the 250 km range. This would be expected from a better model. Even though the model misses very few cities that are close to the region in question, it does miss significantly more that are close to the boundary. Also the bumps for the false positives are similar, but more pronounced here. Interestingly, the overall performance is worse when looking only at the larger model with smaller temperature values (F1 = 0.14 vs. F1 = 0.18) which stands in contrast to the rest of the questions.

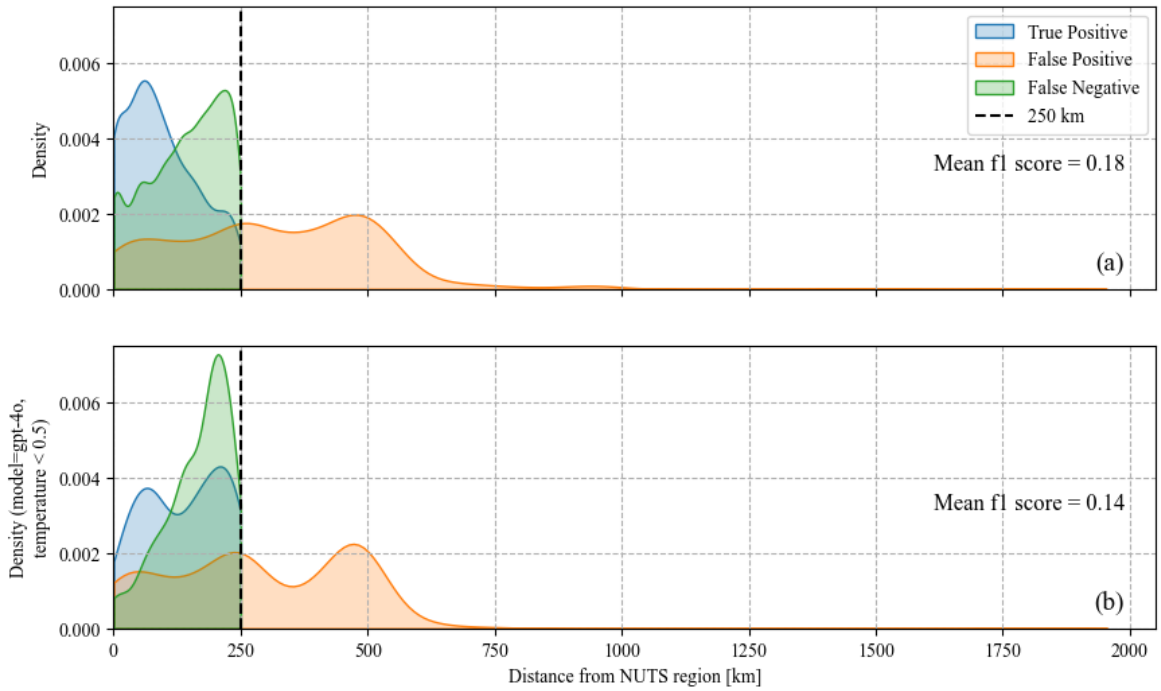


Figure 10: This plot shows the distribution of TP, FP and FNs for the question "What are all cities that are bigger than 99,999 and are less than 250 km from the NUTS region CODE?". Figure **(b)** is restricted to examples from the gpt-4o model with temperature $<$ 0.5.

Question Category Table 2 shows accuracy values depending on the question categories. The non-rag approach performs best with true/false directional and simple topology questions (0.53 and 0.5 resp.). Neighborhood and open directional questions are not much worse with F1 scores at approximately 0.42. Proximity questions show a significant drop in F1 scores down to 0.26 and the combinative questions were almost never answered correctly. When comparing the values to the best model-temperature combination (see Table 3), the categories improve around 0.05 to 0.15 points per category, except for true/false directional, where they actually drop, and for the combinations where they do improve but only slightly.

Question Category	F1	Precision	Recall	Hallucination Rate
simple topology	0.50	0.63	0.47	0.13
neighbors	0.42	0.48	0.41	0.20
directions bool	0.53	0.53	0.53	-
directions open	0.41	0.42	0.41	-
proximity	0.26	0.36	0.24	-
combinations	0.06	0.08	0.05	0.11

Table 2: Results for the non-RAG experiment based on question categories.

Question Category	F1	Precision	Recall	Hallucination Rate
simple topology	0.64	0.80	0.57	0.06
neighbors	0.51	0.54	0.53	0.18
directions bool	0.47	0.47	0.47	-
directions open	0.51	0.52	0.50	-
proximity	0.32	0.38	0.33	-
combinations	0.08	0.14	0.06	0.04

Table 3: Results for the non-RAG experiment based on question categories only using model=gpt-4o and temperature=0 (best technical parameters)

Concrete Question Table 15 (Appendix) shows the results depending on the actual questions asked. Here the most accurate question is *"Is the NUTS region CODE bordering the region CODE?"* - a simple true/false question followed by *"What NUTS regions are within the region CODE?"* - a simple topological question. Also here the worst questions are by far the ones combining spatial concepts.

Maps Figure 17 (Appendix) shows how the F1 score varies between different cities and NUTS regions. The maps show only examples that were present at least 6 times in the dataset. The left maps **(a)** and **(c)** show the F1 scores when the respective city or region was used as examples in the question, the right **(b)** and **(d)** maps show these if they would have been the correct answers. No distinct spatial patterns can be observed. Also there is no significant connection between the size of a city and their accuracy values. Therefore, when asking the LLM questions about larger cities, its performance is not significantly better than when asked about smaller cities (Pearson's $R \approx 0$, $p \gg 0.05$). Likewise, when the correct answers involve larger cities, the LLM does not perform significantly better than when the correct answers involve smaller cities (Pearson's $R \approx 0$, $p \gg 0.05$).
The NUTS regions shown on the map are all of level 2, however the pattern does not

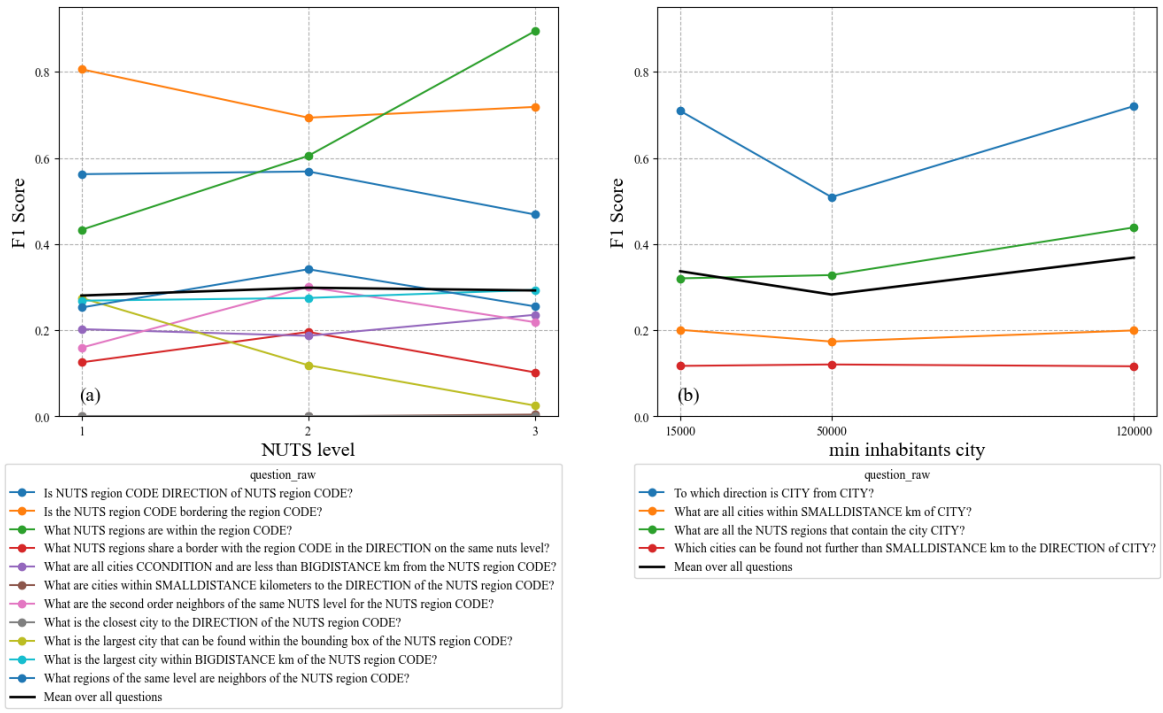


Figure 11: The plots show F1 scores for individual questions dependent on the NUTS level **(a)** and on minimal population size for the input cities **(b)**

.

vary with level 1 or 3. Furthermore, the hallucination rate also does not seem to have a spatial pattern (see figure 18, Appendix).

Non Technical Parameters (Level, Population Size) Figure 11 illustrates the variation in results when examining different examples for question input. Figure 11 **(a)** shows how results of questions about NUTS regions differ, when asking for different NUTS levels. Over all relevant questions, no trend can be observed (Spearman's R = 0.00, p » 0.05). Even when looking at the questions separately they are barely affected by the chosen NUTS level. The only question showing a clear positive trend with a higher level is *"What NUTS regions are within the region CODE?"*. This however can be easily explained by the fact that higher level NUTS regions have less regions within them (level 3 regions only contain themselves, see table 13 row 2) and are therefore easier to answer with higher levels. Figure 11 **(b)** shows the same plot but dependent on the minimum population size of the city chosen for the question. Here, also almost no trend can be observed (Pearson's R = 0.05, p < 0.05). Also here the different questions are affected in different ways by the chosen minimum population size.

Conclusion In conclusion, the following results can be observed for **RQ1**: When splitting by model variables, the larger model (gpt-4o) with the lowest temperature performs best where F1 = 0.37, however scores do not drop much with higher temper-

atures. For the question *"What are all cities that are bigger than 99,999 and are less than 250 km from the NUTS region CODE?"*, the LLM has more difficulties correctly identifying cities around the 250 km limit but also around the 500 km mark. The model's performance is strongly dependent on the specific questions asked resulting in much higher scores for easier questions and very low scores for questions combining spatial concepts. Finally the performance is not, or only barely, influenced by geographical location, the chosen NUTS level or city population size. The following chapter shows, how the RAG-based approach performs compared to this baseline.

6.2 RQ2: To what extent can a GraphRAG-based approach address limitations of non-RAG LLMs for spatially explicit tasks and what role do different parameters (LLM, temperature, prompt design, ontology richness, and few-shotting) play in improving results as measured by precision, recall and F1 score?

For the RAG-based experiment there are more technical variables to consider than just the model and the temperature. Apart from that the mini-fine-tuned model also plays a role here. Furthermore it is important to point out that the accuracy values are based directly on the retrieved data. The answers are not sent to a model a second time.

Technical variables of the LLM Table 4 shows the performance values similar to table 1. Since the database does only contain existing NUTS codes, there cannot be a hallucination rate here. Compared to the non-RAG approach a clear difference can be observed. The best setup (model = gpt-4o, temperature = 0.00/0.33) more than doubles the F1 score. While gpt-4o-mini generally performs weaker than non-RAG, the mini-fine-tuned model is not far behind gpt-4o and outperforms the other models in the non-RAG approach by a wide margin. Also the effect of temperature on performance is noticeable, letting F1 scores drop 0.22 points for the mini-fine-tuned model, 0.14 for gpt-4o and only 0.06 for gpt-4o-mini.

Model	Temperature	F1	Difference to non-RAG
gpt-4o	0.00	0.81	0.44
	0.33	0.81	0.45
	0.66	0.78	0.45
	1.00	0.73	0.40
	1.33	0.67	0.36
gpt-4o-mini	0.00	0.28	-0.03
	0.33	0.26	-0.03
	0.66	0.26	-0.03
	1.00	0.25	-0.04
	1.33	0.22	-0.06
mini-fine-tuned	0.00	0.65	-
	0.33	0.65	-
	0.66	0.63	-
	1.33	0.43	-

Table 4: Result table for the RAG experiment depending on model and temperature. Precision and recall are almost always identical to F1 score (see subsequent tables) so they are omitted here and differences to non-RAG are shown instead. The mini-fine-tuned model is not compared to the non-RAG approach as it is only fine-tuned to generate SPARQL queries.

Model	Template	F1	Precision	Recall
gpt-4o	few-shotting	0.82	0.81	0.82
	template v1	0.72	0.72	0.73
	template v2	0.77	0.77	0.78
gpt-4o-mini	few-shotting	0.52	0.52	0.54
	template v1	0.12	0.12	0.12
	template v2	0.17	0.17	0.18
mini-fine-tuned	few_shotting	0.72	0.73	0.73
	template v1	0.50	0.51	0.51
	template v2	0.52	0.53	0.53

Table 5: Results based on the used system and user message. System messages without tips about using `geo:wktLiteral` were excluded here to ensure comparability. The difference between v1 and v2 is the added sentence "The questions are about the location and topology of the graphs elements" for v2 in the user message (see code 3)

Template	F1	Precision	Recall
few-shotting	0.82	0.81	0.82
template v1	0.81	0.81	0.82
template v2	0.83	0.82	0.83

Table 6: Results based on the used system and user message (temperature < 0.5, model=gpt-4o).

Technical Variables for RAG (messages, tips, ontology, few-shotting) The table looking at all technical variables (model, temperature, system message, message template and ontology version) can be found in the Notebook 09 in the GitHub repository (Groß, 2025). Table 5 shows how the different system and user message templates affect performance. The template version 1 and 2 only differ slightly in their user-message (the version 2 does not specify a "SPARQL SELECT" query, only a "SPARQL" query, leaving the option for the LLM to generate ASK queries), their system messages differ more. All prompting messages can be found in code 3 (Appendix). To ensure compatibility only messages with tips were used for table 5. Since the LLM generated almost no ASK queries with template 2 (< 0.5 %), this cannot explain differences in performance. Therefore differences have to be traced back to the different system messages. Here the main difference lies in a tip to the LLM that the questions asked are *"about the location and topology of the graphs elements"*. When looking at table 5 the difference in performance between templates is not huge, but noticeable. Furthermore, the difference is bigger for gpt-4o-mini.
Few-shotting performs best for both models, however the effect on the gpt-4o-mini model is much more pronounced then with the gpt-4o-model. Interestingly, the few-shotting approach also produced around 10 % ASK queries. These were neither specified in the few-shot examples nor in the system message (see code 4, Appendix for few-shot templates and examples). Looking only at the best model parameters (model=gpt-4o, temperature < 0.5, table 6), all templates are closer together and few-shotting loses its edge over the other templates.

Model	System Message	F1	Precision	Recall
gpt-4o	no tips	0.67	0.67	0.68
	tips	0.72	0.72	0.73
gpt-4o-mini	no tips	0.04	0.04	0.04
	tips	0.12	0.12	0.12
mini-fine-tuned	no tips	0.54	0.55	0.55
	tips	0.50	0.51	0.51

Table 7: Results based on the system message including tips or not based on template version 1.

Model	System Message	F1	Precision	Recall
gpt-4o	no tips	0.75	0.75	0.75
	tips	0.81	0.81	0.82

Table 8: Results based on the system message including tips or not based on template version 1 (temperature < 0.5, model=gpt-4o).

Comparing only the two system messages (one including tips about using geofunctions only with `geo:asWKT` literals and the other not) with otherwise identical parameters using only the template version 1, shows that results only improve slightly (+0.05) with tips for the gpt-4o model. For the gpt-4o-mini model the improvements are slightly stronger with improvements of 0.08 (see table 7). The best model setup (table 8) shows similar slight improvements.
Table 9 shows how performance varies when using the different ontology versions. Here,

Model	Few-shotting	Ontology	F1	Precision	Recall
gpt-4o	False	Ontology V1	0.79	0.78	0.80
		Ontology V2	0.75	0.75	0.76
		Ontology V3	0.66	0.66	0.66
	True	Ontology V1	0.82	0.82	0.83
		Ontology V2	0.81	0.81	0.82
		Ontology V3	0.82	0.82	0.82
gpt-4o-mini	False	Ontology V1	0.13	0.13	0.13
		Ontology V2	0.12	0.12	0.13
		Ontology V3	0.12	0.12	0.13
	True	Ontology V1	0.53	0.52	0.55
		Ontology V2	0.51	0.50	0.53
		Ontology V3	0.52	0.52	0.54

Table 9: Results based on the used ontology split by model and few-shotting.

the mini-fine-tuned model is omitted as well, since it is fine-tuned only on the first ontology version. Ontology version 1 and 2 do only differ slightly (see section 5.3). Only a few triples that are unnecessary for SPARQL generation are removed for version 2. The version 3 does not contain any natural language triples except for labels. This still provides all necessary information to generate correct SPARQL queries but gives the LLM less contextual information. For gpt-4o-mini, version 3 performs quite similar to the others. When using gpt-4o, a larger drop for F1 scores from the ontology version 1 and 2 to version 3 can be observed (0.79 and 0.75 to 0.66).
Few-shotting barely has an effect on gpt-4o, except for the ontology version 3, completely removing the drop described before. For version 1 few-shotting even reduces performance by a slight margin. For gpt-4o-mini the improvements for employing few-shotting are substantial with an improvement for F1 scores of around 0.4 for all ontologies similar to the templates described above. The maximum F1 score of 0.89 was achieved with the following parameters: **Gpt-4o for the model, temperature = 0, ontology version 2 and template version 2** (see Notebook 09 in the code for details).

Three examples Table 14 (Appendix) shows the same three examples from RQ1 (see section 6.1) and the respective results from the RAG approach. For the first two examples from table 9 the RAG approach generates queries that produce the correct answers. The more complex third example was also generated correctly. In the Appendix code 5 shows the generated query, which can be compared to the ground truth query, code 6 . This comparison shows that the generated query is barely different for the ground truth, it is syntactically and semantically correct. This showcases the

ability of LLMs to generate SPARQL queries usable for answering complex questions on data when provided with the relevant ontology.
Figure 12 mirrors figure 10 for the RAG approach. It clearly shows that the accuracy of the results do not vary depending on the distance of the city in question. Only figure 12 **(b)** shows slightly more true positives compared to false negatives further from the region. This approach is therefore independent from the inherent spatial understanding in the LLM and can use the clearly defined spatial concepts in the ontology and the data in the graph if the query is correct. False positives do not exist for the RAG approach for this question.

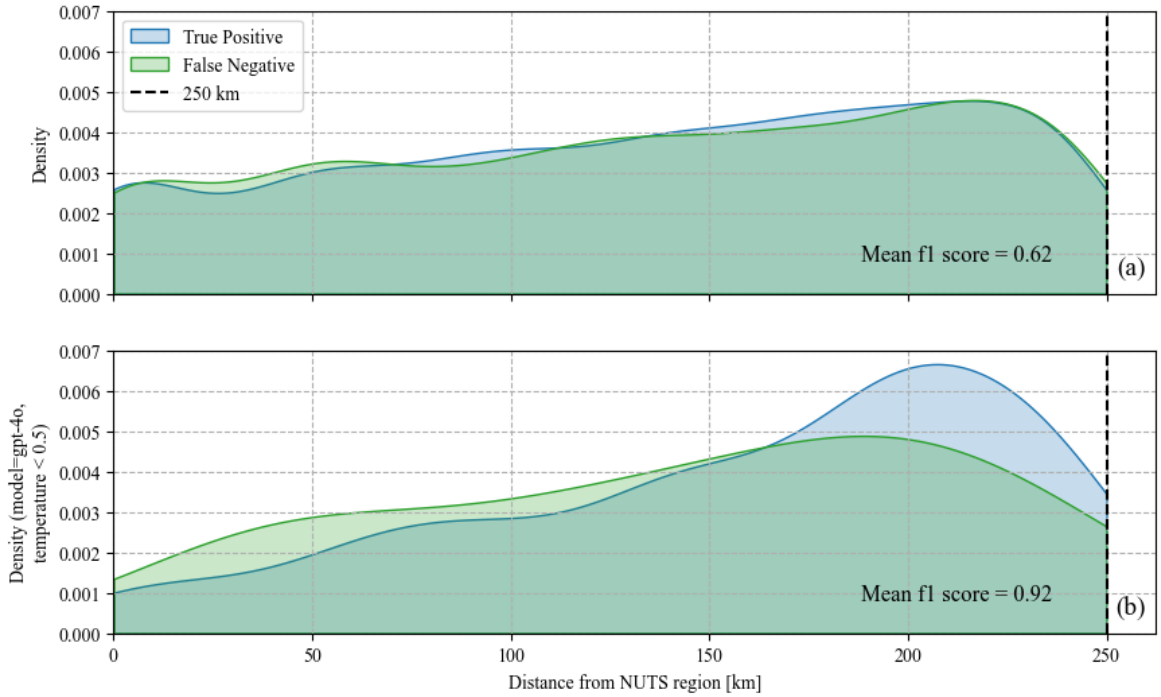


Figure 12: This plot shows the distribution of TP, FP and FNs for the question *"What are all cities that are bigger than 99,999 and are less than 250 km from the NUTS region CODE?"* for the RAG approach (excluding mini-fine-tuned), similar to figure 10. Figure **(b)** is restricted to examples from the gpt-4o model with temperature < 0.5.

Question Category Table 10 and table 11 show the F1 scores and their differences to the non-RAG experiment based on the question category. For all question categories the RAG approach performs better, except for the open direction questions. Using the best model / temperature combination (table 11), the RAG approach performs even stronger. The differences are especially visible for true/false direction, proximity and the combinations questions. For the simpler categories, this approach even achieves F1 scores of > 0.95 for the best combination highlighting the potential of GraphRAG for geospatial questions. Also the more sophisticated combination questions perform

comparatively well in this approach with F1 scores of 0.44 over all models and 0.72 for gpt-4o with temperature = 0.

Question Category	F1	Difference to non-RAG
simple topology	0.53	0.03
neighbors	0.45	0.03
directions bool	0.75	0.21
directions open	0.29	-0.14
proximity	0.58	0.32
combinations	0.44	0.39

Table 10: Results for the RAG experiment based on question categories.

Question Category	F1	Difference to non-RAG
simple topology	0.96	0.32
neighbors	0.84	0.34
directions bool	0.99	0.52
directions open	0.41	-0.10
proximity	0.89	0.56
combinations	0.72	0.64

Table 11: Results for the RAG experiment based on question categories only using model=gpt-4o and temperature=0 (best technical parameters)

Concrete Question Table 16 (Appendix) shows F1 scores and their differences to the non-RAG experiment based on the individual questions. The results confirm the ones described above. Every question performs better in the RAG experiment except for *"To which direction is CITY from CITY?"*, which performs much worse. Interestingly the other question with open direction results (*"CITY is to which direction of NUTS region CODE?"*) still performs better with RAG. Therefore the query generation most likely had problems differentiating correctly between two cities compared to a city and a NUTS region. The question *"What is the largest city that can be found within the bounding box of the NUTS region CODE?"* even has an F1 of 1, meaning for this subset, all 216 queries were generated correctly, resulting in an improvement over the non-RAG approach of 0.88! The more technical nature of the question most likely makes it difficult for the non-RAG LLM to know the answer, since it does not perform spatial operations. Here the value of the RAG approach to provide more spatial capabilities is clearly shown.

Failure Rates During the RAG-approach the overwhelming majority of F1 scores are either 0 or 1. In 45 % of cases the F1 score is 0, in 49 % it is 1. Only for 6.7 % of all questions the RAG approach generates a query that gives a partially correct answer. This happens almost exclusively when asking for cities near a city and the city in question is included in the answer resulting in an F1 score slightly below 1. The non-RAG approach, while having a similar rate of answers with F1 = 0 (55 %), also has F1 score between 0 and 1 for 21 % of all answers. This results in the overall lower scores for the non-RAG approach. The RAG approach therefore has the advantage that there are much less wrong answers, only no answers. Furthermore, the mean F1

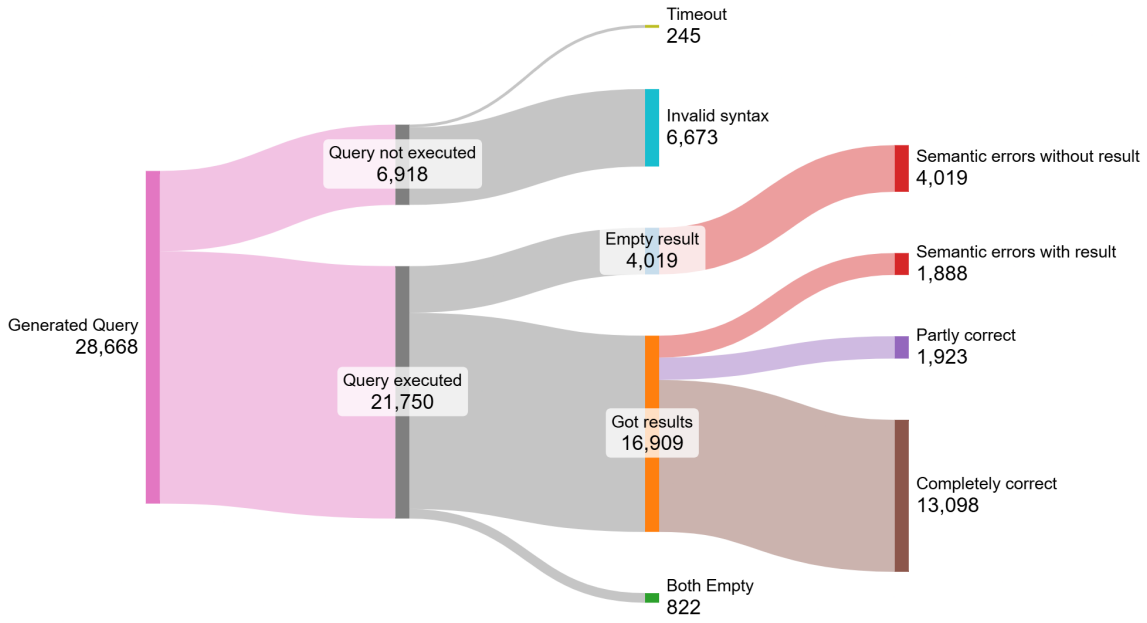


Figure 13: The Sankey diagram shows the flow of the 28668 examples for the RAG workflow. Some queries could not be executed, resulting in timeouts or syntax errors. The other queries were executed and either resulted in no results, incorrect, partly correct or completely correct results. Also, for some questions the ground truth answers and the generated were empty (Figure by SankeyMATIC.com).

score out of all "successful" query generations (meaning the ones that were executed and have results of the correct type) is 0.90.

Reasons for Failure So why do 45 % of the queries give an F1 score of 0? Figure 13 shows a Sankey diagram, explaining at which point in the workflow, the generated query failed or succeeded. First, around 1/4th of the queries could not be executed. This can either be due to an error in the generated syntax or due to a timeout. Timeouts mostly occur when the generated query has also some semantic errors resulting in too many geographical operations to be performed, e.g. when failing to specify the city or NUTS region's name.

The successfully executed queries can either be empty or not. Empty results that are supposed to be empty (when the ground truth is also empty) are hard to assess since it is not entirely clear if the query was semantically correct, however upon manually inspecting the results they show to be generally correct. This happens mostly for the question *"What are cities within 8 km kilometers to the DIRECTION of the NUTS region CODE?"* since the definition for directions can lead to no cities meeting this criteria, especially for bigger NUTS regions and intercardinal directions.

However if the result is empty but the respective ground truth is not, there have to be

semantic errors in the query.
Cases that produce a non-empty result, can either be wrong, partly correct or completely correct, with the majority falling into the last category. This underlines the argument from above stating that when the RAG-approach manages to generate an executable query, it is in most cases a correct and complete answer! When taking only questions that the LLM produced an executable query for, the F1 score for the best model parameters (model=gpt-4o, temperature=0.33 ontology=version 1/version 3, few-shotting=True) the F1 score is 0.97. A RAG-approach with the correct setup and efficient few-shot training could therefore answer questions employing the three analyzed spatial concepts correctly in around 75 % of cases, while giving no answer for 25 % of the time (when no executable query is generated).

Conclusion In conclusion, the following results for **RQ2** can be observed: Similar to the non-RAG approach, the larger model (gpt-4o) with the lowest temperature produced the best results, with the mini-fine-tuned model not too far behind. Prompt-engineering and few-shotting are important for the mini and mini-fine-tuned model, less for the gpt-4o model, especially when using lower temperature values (see table 5 and 6. Removing natural language from ontologies impacts performance, but much less when combining it with few-shotting.
If a generated SPARQL query can be executed, the results are in most cases either empty or correct. Looking at only successfully executed queries and their best parameter setup results in a F1 score of 0.97.

7 Discussion

7.1 Results in Context

This work analyzed how LLMs perform when answering questions relating to three spatial concepts: topology, directionality and proximity. While the first research question (**RQ1**) focused on capabilities of unmodified LLMs, **RQ2** analyzed the potential of geospatial GraphRAG-LLMs to bridge gaps in the spatial capabilities of LLMs. The experiment was based on 18 questions based on the three spatial concepts and their combinations. These questions were populated with different examples from European NUTS regions and cities in the geonames dataset. A multitude of variables were examined for their impact on performance, e.g. the concrete LLM or the models temperature for **RQ1**. For **RQ2** even more variables were taken into account, for example a custom fine-tuned model, the prompt templates or the ontology used to describe the data. While answers to the questions were generated directly by the LLM for **RQ1**, it generated SPARQL requests and sent them to a graph database for **RQ2**. While the approaches differ fundamentally, their performances were compared using the direct answer or the results from the database and comparing them to the results from a ground truth query. For the given questions, the non-RAG approach achieves a maximum F1 score of 0.37 for the best model and temperature combination, ranging from 0 to 0.83 depending on the actual question asked. The RAG approach outperforms non-RAG one by a wide margin with a F1 of 0.81, ranging from 0.39 to 1 depending on the question.

7.1.1 Non-RAG approach

The specificity of the questions in this study and their constraint on the data provided by NUTS and geonames make it difficult to set the findings into the context of current literature. Nonetheless, as described in section 2.5.1 there are multiple publications that analyze the spatial understanding of LLMs. Due to the lack of appropriate benchmarks and the limited scope of this thesis, comparisons to other publications are quantitative when reasonable and are complemented by qualitative analysis.
Roberts et al. (2023) prompted gpt-4o for different geographic aspects, asking for socioeconomic parameters of countries, locations of cities or questions on distances. Their task prompting the model with coordinates to give the corresponding city names resulted in much better results for bigger cities. Comparing this to results from questions based on cities in this study (see figure 11 **(b)**) the difference in accuracy when asking questions about bigger cities could not be observed. This could have multiple reasons. The concrete questions asked can already have strong effects on the outcome as evi-

dent in the vastly different performances for each question observed in this study. Also, Roberts et al. (2023) looked at more extreme examples (the most populated and least populated cities in their dataset), while this thesis just looks at three levels of minimum requirements for choosing an example city ranging between 15,000 and 120,000 inhabitants.

Before performing this study it was suspected, that the non-RAG LLM would be more capable when answering questions about cities and regions in more popular regions. As described in Fulman et al. (2024a) and Manvi et al. (2024), (spatial) biases of humans might be present in the textual data that is used to train LLMs. These biases therefore are replicated by the LLMs. Since eastern Europe is generally less represented in media compared to famous destinations like France or the UK, it was suspected, that the non-RAG approach would perform worse when prompted about cities or NUTS regions in eastern Europe. However, this suspicion could not be confirmed. Looking at figure 17, performance is not varying based on the location within Europe.

Generally, the results differ much less based on different examples or even based on model parameters when compared to the concrete questions that are asked. If a question is straightforward and unambiguous, e.g. *"What NUTS regions are within the region CODE?"* or *"Is the NUTS region CODE bordering the region CODE?"*, even the non-RAG approach can answer with comparatively good accuracy. However, with more complexity, uncertainty, technicality and ambiguity the model performs much worse.

Generally, the non-RAG results are in line with expectations. They get worse with rising question complexity, higher model temperature and smaller models. It performs better with true/false questions regarding direction and neighborhood or when retrieving simple containment relations. Questions requiring more sophisticated geospatial understanding (containment within bounding box, combination of geographical concepts) are much harder for a non-RAG LLM to grasp. While biases towards larger cities or higher-level NUTS regions could not be found on the same scale as Roberts et al. (2023) did, a clear hierarchical bias similar to Fulman et al. (2024a) could be observed. While for the ground truth data approximately 77 % of all answers for NUTS regions have the same country code as the region in question, the non-RAG answered these questions 96 % of the time with regions in the same country. However it remains to be discussed if this is more due to geographical hierarchical bias or a linguistic one since the NUTS codes from the same country look more similar and are therefore more likely to be generated.

7.1.2 RAG approach

While the resulting formats look similar for the RAG and non-RAG approach, the methodological approaches are fundamentally different. While non-RAG completely relies on the models internal storage of facts, RAG or more accurately GraphRAG uses the LLM to generate queries to a graph database. Therefore the actual role of the LLM in the workflow shifts completely. This is also in contrast to other RAG approaches. Traditional RAG approaches rely on retrieving unstructured data from a document database, based on semantic similarity as described in section 2.2. Contrary to the explored GraphRAG approach this provides contextual information to the LLM while still relying on its capability to process natural language. While GraphRAG can also be used in a similar way (see section 2.4), this work only explored questions where the complete answer lies in the retrieved data, without using these answers as a subsequent input for another LLM call. The generation part of RAG is therefore out of scope for this study. However, in the context of the analyzed questions, the retrieval of the data is by far the most crucial step in answering the question since the answers exclusively rely on the queried data and the LLM does not have to perform any summarization or information extraction anymore.
Nonetheless, the result of the two approaches can be compared. The RAG approach scores significantly higher for almost every category when the technical variables are optimized. The complexity of a question also has drastic effects on the results, however for RAG this stems from the more complex query that needs to be generated instead of the more complex combination of knowledge required for a non-RAG LLM.
A recent study by Akinboyewa et al. (2024) explores a more comprehensive approach where geoprocessing tools in QGIS are cataloged and used by an informed agent to autonomously perform spatial analyses. They classify their tasks into basic, intermediate and advanced tasks. Similar to this study, their approach performed well with basic and intermediate tasks, while struggling to implement more complex workflows. They also highlight that LLMs need more knowledge about GIScience fundamentals in order to appropriately choose the correct workflows for a task. Other issues they raise are the need for proper data validation and data sourcing. These are tasks where semantic web technologies could play a significant role. As this study shows, describing data and tools using ontologies can be understood by LLMs and could potentially bridge these gaps. Their results range from 95 % successful analysis for basic tasks and 80 % for intermediate to 75 % for advanced tasks.
Recently, Mansourian & Oucheikh (2024) also assess geospatial data analysis with natural language prompts. They use named entity recognition to map them to a geospatial ontology to provide semantic context for the LLM before a GIS tool is used. Similar to Akinboyewa et al. (2024) they use GIS functions from QGIS to execute their analysis.

Additionally, they employed fine-tuning and few-shot learning for more reliable code generation. Their tasks are simple geospatial analyses like finding nearby locations (Pharmacies, Hotels, etc.) matching specific criteria or even more complex network analyses for multiple purposes. Their specifically trained LLM outperforms the vanilla LLM when being used to generate code for data extraction or spatial analyses. Similar to this study, the main task of the LLM is not to retrieve knowledge itself, but to generate code to retrieve and perform spatial analyses on data. Their limitations also include the semantic ambiguity between natural language and technical parameters, e.g. the mode confusion the "highway" tag in OpenStreetMap with the word highway that can be seen as a synonym for road in general. These are issues that can be addressed with the presented workflow when the model is enhance with an ontology during code generation. This provides the LLM with an idea of how the data is modeled not only technically but also semantically. This reduces the risk of running into ambiguity issues such as the "highway" example explained before.

Y. Jiang & Yang (2024) employ a very similar approach to this work, using LLMs to access geospatial databases with SQL, and also providing it with the specific table schemes. Their results also tend to get worse with higher temperature and rising query complexity, reaching a maximum accuracy of 1 for simple "single table spatial queries" and dropping to 0.27 for "spatial join queries" with a temperature setting of 0.8. These results are in line with the presented findings where the F1 score also reaches 1 for the best performing question/model combination and drops significantly for more difficult questions and worse model parameters. Also their questions are quite similar in the usage of geographical concepts, as they ask for example: "What is the closest street to the subway stations in the neighborhood that has the highest population density?", where proximity needs to be assessed and entities need to be filtered by attributes.

Emonet et al. (2024) generate complex SPARQL queries over bioinformatics knowledge graphs. They also try different models and employ a more sophisticated validation approach for fixing failed generation attempts. Their performances range form F1 scores of 0.85/0.91 (with/without validation) for gpt-4o to 0.37/0.37 for gpt-4o-mini. The drop between models therefore can be compared to the ones in this thesis with the same models used.

Apart from the generally better performance of the presented RAG approach, the hierarchical bias for retrieving NUTS regions is completely eliminated. The RAG approach retrieves 75 % of all codes with the same country code compared to 77 % for the ground truth data. Furthermore, while the RAG approach hallucinates around 10 % of all NUTS codes (see table 1), the RAG approach is immune to hallucination since non-existing codes are not in the database, although another type of "hallucination" can occur where the model thinks a query filter is applied by simply naming a variable

in a certain way, see code 1. The important difference is that when the non-RAG approach fails, it can still provide made-up or wrong data, while the RAG approach mostly fails to create correct query syntax or provides empty results (see figure 13). In productive situations no results are surely to be preferred over wrong ones. The following chapters discuss building blocks and the workflow to generate the queries for the RAG approach in more detail.

7.2 Query Generation, Parameter Discussion

For the RAG process, a multitude of parameters were evaluated and different setups were compared. These include three different versions of the ontology, two different prompt setups, three different models (including one fine-tuned model), five different temperatures and few-shotting. All of these parameters affected the results, suggesting that further optimization of each one could improve the results in the future.

Model and Temperature The model had the most significant effect. gpt-4o is simply more capable than the smaller but faster and cheaper gpt-4o-mini. Generating SPARQL queries that depend on a whole ontology Turtle serialization is a complex task that often exceeds the capabilities of the mini model. Other studies analyzing LLMs for (SPARQL) query generation arrive at the same results: better models generate better queries (Emonet et al., 2024; Y. Jiang & Yang, 2024; Lehmann et al., 2023; Diallo et al., 2023). It is therefore safe to assume that future advancements in LLMs may further boost query generation capabilities, especially with recent advancements using the chain-of-thought architecture Wei et al. (2022).
Similarly, the temperature is a parameter where lower values generating better queries (Y. Jiang & Yang, 2024). However, here the effect is not as strong as initially expected. While just setting the temperature to 0 for query generation tasks might be sufficient, more sophisticated approaches could also be evaluated. For example Zhu et al. (2024) propose adaptive temperature setting for LLM coding tasks that could also boost performance for query generation.

Fine-Tuning Fine-tuning is another method that can boost model performance, although it has to be acknowledged that the fine-tuning was not the main task of this work and overfitting cannot be ruled out. Nonetheless, the fine-tuning model that resulted from only 35 question-query pairs was able to outperform the gpt-4o-mini model significantly. Other works found similar boost in performance when tuning the model to better understand SPARQL query generation. Rangel et al. (2024) explore model fine-tuning for SPARQL generation and also with respect to different variable naming strategies or additional comments in their query dataset. However, studies

evaluating fine-tuning mostly test just on their respective example knowledge graph. A more general fine-tuning approach that seeks to build a LLM specifically to generate SPARQL queries from natural language questions based on a given ontology would certainly be a step forward. On the other hand fine-tuning a model just for generating queries for one specific knowledge graph is most likely not worth the effort.

Prompt Engineering and Few-Shotting Two differences in the system message were explored one, one where there was an additional sentence added (*"The questions are about the location and topology of the graphs elements"*) in the models system message (see figure 5, 6) and one where tips were added for query generation were added (*"Keep in mind that geofunctions can only be used with a WKT literal that is attached to a geometry subject: '?geom geo:asWKT ?wkt'"*). Both did boost performance but not as much as the model or temperature values. Both changes did not result into a performance boost of more than 0.08 for the F1 score. Kosten et al. (2025) evaluate different prompting techniques for SPARQL query generation combined with few-shotting and they also include the relevant ontology in the prompt. Their analysis of different prompting frameworks resulted in the best accuracy when simply giving 5 random few-shot examples without any special prompting techniques.
Few-shotting was also analysed in this work. When provided with the standard ontology and the gpt-4o model, few-shotting did not make a substantial difference. Only when using the ontology that did not contain natural language triples (ontology version 3), few-shotting significantly improved the results. However, when using the gpt-4o-mini model, the result improvements from few-shotting were significantly higher. This aligns well with the findings by Kosten et al. (2025) since they use older models that are more similar to the gpt-4o-mini model. Also the result suggest that few-shotting is less relevant with better models and the improvements for the smaller model are not enough to catch up to the results of the better ones. On the other hand it has to be noted that the few-shotting setup only included two shots that were static for every example. These questions could be bad examples, or there are simply more examples needed to see substantial improvements. Furthermore, the ontology was provided within the system message for this few-shotting experiment, therefore the example questions could have affected the models attention to the ontology and therefore affect its generation capabilities. For a clearer picture, more research is needed that focuses on the best few-shotting setup for SPARQL query generation.

Ontologies As described in section 5.3.4 the ontology versions 1 and 2 barely differ, which is also apparent in the similar results. Version 3 does not contain any natural language information. This was part of the experiment to analyze how much the LLM

relies on the definitions, comments and other textual information in the ontology. Since the relationships are still defined clearly, it is possible to understand and apply this ontology also without the natural language. In theory, a LLM should therefore be also able to use an ontology with this information only. The results however clarify that the LLM indeed still relies on the text. For the gpt-4o model the results without natural language were lower, from 0.08 to 0.13 points. Interestingly, this effect completely disappears with few-shotting, suggesting that the LLM can quickly learn this missing information. It is therefore a matter of debate if it is more efficient to include more textual richness in the ontology or rely more on few-shotting. This will most likely depend on the specific ontology, since ontologies other than the geonuts ontology might react differently to the presented approaches, especially in other domains. During the ontology development, making the ontology as small as possible was a major focus. Eliminating classes and properties that were not related to the analyzed spatial concepts was important for the model to understand the critical parts of the ontology. Eliminating the textual clues negatively affected the performance, therefore it would be important to understand which information exactly would boost performance (the definitions, usage examples, etc.). In the future, ontology engineering efforts could therefore specifically target LLM understandability. The ontology versions V1 and V2 already include the hint to use `geo:wktLiterals` which is already a step in this direction. Common misunderstandings could be clarified this way.
However, the ontology analysed in this study is fairly small. For bigger ontologies too much information might come with the expense of diluting the main definitions. As mentioned above, fine-tuning could also play a role here. A fine-tuned model that is trained on multiple ontologies and just focuses on SPARQL generation could boost results even further. And if it is not fine-tuned on a specific ontology, no retraining would be required to use it for new data and new ontologies.

7.3 Query Generation, Procedural Discussion

Most queries that do execute and retrieve results with the correct answer type turn out to give the correct results, however this also depends on the concrete question asked (see table 16, Appendix). While the section above discussed the variables and how they affect query generation, this section discusses the workflow that led to the generation of these queries. One of the main steps that is missing in this work is a failure handling process. While some typical syntactical errors are taken care of, e.g. missing prefixes or some wrong prefix usage, errors can be diverse and unpredictable. Akinboyewa et al. (2024) for example, use an automatic debugging tool that prompts the LLM to review the code and regenerate if issues arise. This approach is also used in the LangChain GraphDB connector that follows a workflow similar to this study

(LangChain, 2024). Failures were mainly mitigated though tips in system messages and the ontology as described in the previous section, however this is an important step in the workflow that could be added in the future.
Other research explores the idea of using controlled natural language instead of SPARQL to circumvent the limited SPARQL generation capabilities of LLMs (Lehmann et al., 2023). They conclude with better generation capabilities when using controlled natural language, however their LLMs are outdated today and modern one might result in different findings. Also, Diallo et al. (2023) explore procedural techniques to boost their performance. They find that the LLM has difficulties using the correct URIs. As a possible solution they use entity detection for the question, and replace the entities with their corresponding URIs before sending them to the LLM or append the correct URIs at the end of the prompt. In this work, generation the false URIs was not identified as a key problem, only in rare cases the LLM generated the wrong full URIs for the prefixes at the beginning of the query. These are fixed by simply comparing the generated prefixes with a list of known ones, e.g. from the ontology file, and replacing them if necessary.
In the aforementioned paper about bioinformatics knowledge graphs, the prompt is not only enhanced with the relevant ontology but also with relevant example queries that are semantically similar to the question asked (Emonet et al., 2024). This results in better query generation and error handling. Additionally, they employ a validator that checks triples in the generated queries against the constraints defined in the endpoints' ShEx shapes. This validator then provides the issues found back to the LLM to help with re-generation. An evolving database of examples that can help an LLM generate and debug queries more efficiently could be a valuable addition for better query generation. Akinboyewa et al. (2024) call this "an experience-based knowledge system" where common problems are cataloged and the LLM can draw from this database when it runs into problems. They describe this in the context of their autonomous GIS bot, however the idea is comparable to the query example database described by Emonet et al. (2024) in the context of SPARQL query generation.
The techniques employed in this thesis also highlight chances and shortcomings in the workflow process. First, the LLM can generate queries that work for the rather specific field of GeoSPARQL bringing natural language based query generation systems to the realm of GIScience. Additionally, the inclusion of the ontology in the prompt successfully allows the LLM to utilize functions that are entirely new to the LLM. The "fake" direction functions are employed and can therefore successfully be used to expand SPARQL functionalities. To the best of my knowledge this is a novel technique that opens the door to build upon existing (Geo)SPARQL functionalities using a middleware that translates new, non existing functions defined in the ontology into

executable queries.
Typical issues that arose when generating queries were the usage of `geo:Geometry` objects for the geof: functions attributes, even though this was gravely reduced with suggestive prompt instructions. Another interesting issue was the usage of specific variable names instead of filtering for attributes. Code 1 shows a SPARQL query generated for the question *"Which cities can be found not further than 8 km to the west of Geiselwind?"* (model=gpt-4o-mini, temperature=0.33). The query would be syntactically and semantically correct if it were not for the missing triple pattern `?geiselwind gn:name "Geiselwind" .`. The LLM did everything correctly but only named the variable instead of actually specifying the associated name.

```
[All Prefixes]

SELECT ?city ?cityWKT
WHERE {
  ?city a gn:Feature .
  ?city geo:hasGeometry ?cityGeom .
  ?cityGeom geo:asWKT ?cityWKT .

  ?geiselwind a gn:Feature ;
            geo:hasGeometry ?geiselwindGeom .
  ?geiselwindGeom geo:asWKT ?geiselwindWKT .

  FILTER(geof:isWestOf(?cityWKT, ?geiselwindWKT) && geof:distance(?
      cityWKT, ?geiselwindWKT) <= 8000)
}
}
```

Code 1: Generated query for question *"Which cities can be found not further than 8 km to the west of Geiselwind?"* (model=gpt-4o-mini, temperature=0.33).

In conclusion there are multiple procedural steps explored in this study and other works that evidently improve the query generation capabilities of LLMs: Giving well crafted prompts to a model, supplying it with the relevant ontology including natural language triples or examples and employing simple or multi-step error handling techniques to fix malformed queries. All these steps could be refined in the future.

7.3.1 The GraphRAG Approach

This study shows that under the presented circumstances, GraphRAG outperforms traditional LLMs for answering spatial questions. On top of better results, GraphRAG provides more advantages. Since this approach does not rely on the LLMs training

data, no retraining is required to update its knowledge. Especially in the domain of GIScience data tends to be large. Shi et al. (2023) already raised this issue, calling for more efficient and transparent AI models, especially in the domain of GeoAI as the periodic retraining of large models has serious environmental consequences. With the GraphRAG approach, the model avoids the need for extensive encoded spatio-temporal data during training and does not require retraining when data is updated. New concepts and functions can be introduced and new data sources can be added by defining them into the ontology. Furthermore, the GraphRAG also reduced the issue of noise. Vector-based RAG that relies on retrieving semantically similar information as context for the LLM will also contain irrelevant information that the LLM has to filter by itself. The GraphRAG approach is also a step towards more explainable AI. LLMs are a black box when they are answering questions, the reasoning behind a generated answer is not provided. This lack of transparency is a clear risk regarding trust, accountability and has further ethical considerations for the use of LLMs (Hadi et al., 2024). For the presented approach, the LLMs answers depend on the data that was retrieved beforehand. It is therefore possible to understand the reasoning behind the answers and to inspect the data that drives the answer.

Furthermore, when comparing the results between the RAG and non-RAG approach the tactics to further boost accuracy would differ significantly. For the non-RAG approach to answer the example questions reliably all relevant spatial information on the data would need to be periodically and reliably encoded into the LLM directly, which would have the environmental implications described above. For the RAG approach, getting better results relies more on perfecting the retrieval process - optimizing ontologies, fine-tuning, few-shotting and prompt-engineering LLMs for better query generation. These are all parameters that would need to be iteratively perfected, however they do not need to be repeated every time the data is updated.

The presented research and its results can also be put in the context of the four pillars of GeoAI presented in section 2.5.2 (W. Li et al., 2024): The presented approach is not entirely **predictable**. Due to the nondeterministic nature of LLMs, I cannot be said beforehand if a SPARQL query will be semantically and syntactically correct. On the other hand, this approach is much more **interpretable** as for example using deep learning models. As this approach accesses the data directly and the performed queries, functions and results can be examined, it is much more transparent then the black box nature of deep learning models. The **reproducibility** pillar can be viewed as a matter of discussion. While not every generated query will be reproducible, the findings of this study would most likely be identical given the large sample size of the presented questions. While the exact numbers are unlikely to be reproduced, the workflow behind this approach can easily be replicated. **Social responsibility** in the

form of sustainability is addressed, since this approach does not require periodic retraining of models, privacy concerns however are dependent on the data sources that the GraphRAG LLM has access to.

7.4 Semantics

7.4.1 Spatial Concepts

One issue to be discussed is the semantics of the researched geographical concepts. Topology, directional and proximity can all have a certain degree of interpretability that play a significant role in understanding the results of this thesis. While containment is a comparatively unambiguous concept, neighborhood is more complex. In the question dataset, the neighborhood concept is asked in different ways. A NUTS region is *bordering* another region, it *is a (second order) neighbor* of a region or it *shares a border with a region*. While *bordering* is a clear instruction on the topological relationship between the two regions, *being a neighbor* is not. Since an LLM is primarily trained on natural language, it might interpret the concept of neighborhood not in its strictly defined geographical context of sharing at least one point of their borders. For example the question *"What regions of the same level are neighbors of the NUTS region DK031?"* was answered by the non-RAG LLM (model=gpt-4o-mini, temperature=0.33) with *"DK032, DK033, DK034"*. While DK033 and DK034 are hallucinated, DK032 is a NUTS region that almost shares a border with DK031. Their geometries are not connected, since DK031 is an island, however there is a bridge between the both. Figure 14 illustrates the situation. In a less semantically restricted conversation these regions might even be considered neighbors, however the LLM might not be suited to make this decision. When the meaning of neighborhood is unambiguously defined in an ontology, the LLM chooses the suitable concept and actual spatial calculations are performed, the process is explainable and unambiguous. This highlights another advantage of using a GraphRAG approach.

Directionality also highlights the semantic issues in spatial questions. While the cardinal and intercardinal directions in a geospatial sense are clearly defined based on existing literature (see section 5.2.2), directions are a much looser concepts in general language. For example Iceland would intuitively be considered to be southeast of Greenland. However, since Greenland has points to the south and to the east of Iceland, this would not be the case when following the presented definitions for (inter)cardinal directions. This was especially clear when asking for cities to an intercardinal direction of a region within a certain distance (*"What are cities within 8 km kilometers to the DIRECTION of the NUTS region CODE?"*). In some cases this might lead to no place fitting this criteria at all since the point to a specific cardinal direction might already

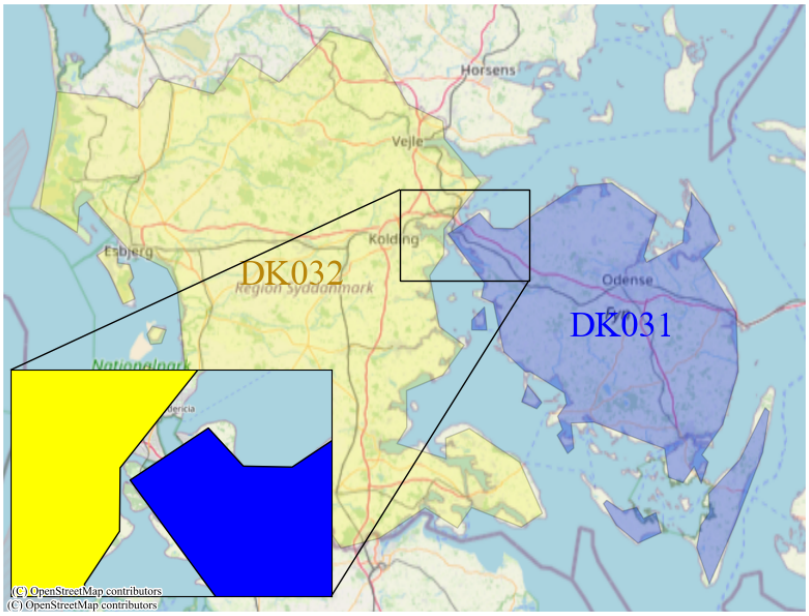


Figure 14: The NUTS regions DK031 and DK032. Their geometries do not touch since DK031 is an island. This illustrates potential semantic issues when asking about neighbors. The question *"What regions of the same level are neighbors of the NUTS region DK031?"* resulted in "DK032" (among others) for the non-RAG approach.

be further away then the specified 8 km. As the cardinal directions often are dependent on the context and are less clearly defined, the RAG approach might not deliver much values here. Since the RAG approach relies on defining clear, unambiguous concepts that leave no room for interpretability, concepts like directionality might be less suited to be implemented here. The RAG approach still performs better for most questions containing directions, however the ground truth data relies on the same, arguably arbitrary, definitions. The interpretation of the results can therefore not solely be based on the numbers but has to take into consideration the actual use-case that the RAG approach would be applied in. Figure 15 shows the regions FI1B and FI1C. The region FI1B is nested in the south of regions FI1C, however the latter actually extends further south than the former due to some islands being part of this region. Therefore:

$$\begin{aligned} &direction(FI1C, FI1B) = \\ &\{North, Northwest, Northeast, West, East, South, Southwest, Southeast\} \end{aligned} \tag{4}$$

while

$$direction(FI1B, FI1C) = \{WithinBounds\} \tag{5}$$

This could be seen as both an advantage and a disadvantage. On the one hand this approach is not flexible depending on context and does not take into account that is most

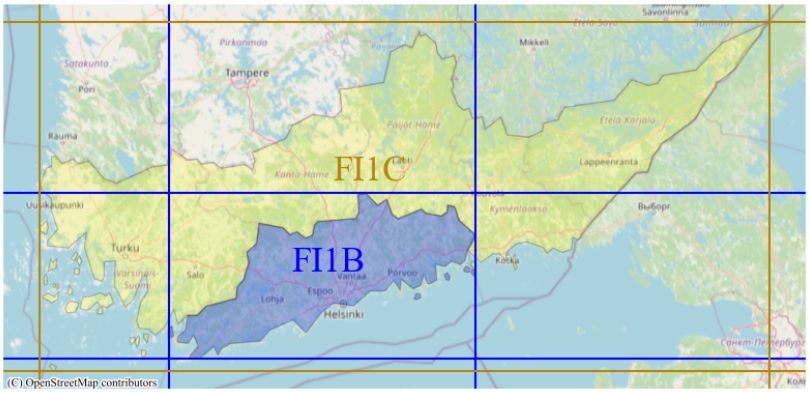


Figure 15: The NUTS regions FI1C and FI1B. FI1B is not south of FI1C since the latter extends further to the south due to some islands. FI1C is therefore to every direction from FI1B, while FI1B is to no direction of FI1C. This highlights semantic questions arising from the RAG approach.

cases FI1B would be considered south of FI1C. On the other hand the RAG approach applies rigorously defined, explainable concepts that can be investigated downstream leading to more explainable processes.

This issue is exacerbated when concepts are combined. Asking *"What NUTS regions share a border with the region SE3 in the southeast on the same nuts level?"* (depicted in figure 19, Appendix) could reasonably be answered with SE1, since leaving SE3 to the east or south in the (more populous) southern parts would most certainly lead you to SE1, however the remote northern parts of SE3 do extend further to the east, invalidating SE1 as an answer.

7.4.2 Questions and Data

Semantic issues can also arise when asking the experiment questions. Many of the analyzed questions, while being asked in a natural way can leave edge cases up to interpretation. For example when retrieving cities with asking *"What are all cities within SMALLDISTANCE km of CITY?"* multiple issues arise. First, it is not specified if this refers to the cities center, its administrative boundaries or to a more vague region that is considered to be this specific city. Also, it does not specify if the city itself is counted when retrieving the answers. In the RAG approach, a city is clearly defined as a `gn:Feature` and a correctly generated query would simply fetch whatever coordinates are defined for the given city. The non-RAG approach on the other hand relies solely on the LLM's internal memory, performing no spatial calculations and in many cases not even knowing the exact coordinates let alone polygon of the city. The two approaches are *fundamentally different*. When asking questions with hard limits and evaluating results based on clear boundaries and criteria, the RAG approach will most definitely give better results - but only when the schema of the underlying data and functions

matches the schema of the expected results.
Another semantic issue is the definition of a city. This can vary from country to country but also has an individual and contextual perspective to it. For example in Denmark cities start with a population of 200 people, while in Japan they start at 50000 (World Bank, 2025). Again the LLM has to rely on which words in its internal memory it associates with the word city, while the RAG approach simply uses entities in the knowledge graph with the `rdf:type gn:Feature`. This gives the RAG approach a considerable advantage as it will only interpret things as cities that are also considered cities by the ground truth as they both access the same data. This is a huge advantage if the underlying knowledge base is correct but can lead to problems if it is not.
The example questions explored in this thesis are simple questions that explore a limited number of spatial concepts and do not rigorously define the semantics of every aspect, be it the spatial concepts or the exact meaning behind the questions or the regions and cities that the questions are about. The non-RAG approach therefore leaves all interpretation of semantic ambiguity to the LLM, while the RAG approach can use the clear definitions of the provided ontology and the functionalities of the database. The ground truth data uses the same ontology and functions that are also supplied to the RAG approach, giving the latter a clear advantage in when comparing the answers to the ground truth. However, this should be viewed as an opportunity! When performing more sophisticated spatial analyses, it is vital to unambiguously define each concept that needs to be used to solve the task. With rising complexity of the tasks, it would be harder and harder to clarify each ambiguity to a LLM. Here, ontologies could play a significant role. The full GeoSPARQL geofunctions ontology (OGC, 2025) already defines a lot more functions compared to the subset used in this study. If GraphRAG can reliably understand these ontologies, make use of their capabilities and transform them into action, it would make artificial GIS analysts as theorized in Janowicz et al. (2020) more realistic. The results presented in this thesis show, that this approach can work on a limited scale and might also work for more complex setups.

7.4.3 Limitations

Data The rise of Large Language Models was next to the enormous leaps in compute power also due to the ability to acquire and process vast amounts of data (Hadi et al., 2024). However this also makes them dependent on the quality of the data. The presented GraphRAG approach poses similar problems. Even with query generation and graph retrieval working reliably in the future and concerns about hallucination and explainability being addressed by (Graph)RAG, the latter is still dependent on the data inside the graph. As already mentioned in section 4.2, the geonames knowledge graph has severe inaccuracies regarding the population values. Also just as a LLM's

internal understanding of facts and data can replicate geospatial and non-geospatial biases (Fulman et al., 2024a; Hadi et al., 2024), geospatial data sources may also replicate these biases (Z. Liu, Janowicz, et al., 2022).
Apart from that there are practical problems with the data used. The area covered by the European Union and therefore its NUTS regions is huge and the geometries that represent them have to be handled on a certain scale. The scale chosen for this thesis (a balance between accuracy and compute power needed) can result in wrong topological relationships at the edge of the NUTS regions. Figure 16 shows two examples. The first one **(a)** shows the pair of cities "Ulm" and "Neu-Ulm" that lie on opposite sides of the Danube river that also marks the border between Baden-Württemberg and Bavaria in Germany. On the larger scale used in this study, the generalized border of the NUTS level 1 regions DE1 and DE2 (Baden-Württemberg and Bavaria) puts Neu-Ulm also barely into DE1, invalidating the topological relationship between the city and the regions. Similarly, figure 16 **(b)** shows a coastal region in Spain (ES512) where cities fall outside of any region due to the generalization of the coastline. This highlights issues regarding data quality and also raises questions on what data to choose regarding computational speed and accuracy. From a FAIR data principles lens (findable, accessible, interoperable, and reusable) some issues need to be acknowledged. First, findability is not an issue, since the used data (Geonames and NUTS) are published through well established institutions. Accessibility, however, presents a limitation: the proprietary LLM used in this study is not freely accessible to the public. Nonetheless, with appropriate adjustments, the study could be reproduced using open-source LLMs. Interoperability might also be impacted since the reused ontologies and the geodata in the KG were modified to be understood by LLMs as easy as possible. Transferring this study to other dataset would need similar pre-processing steps to achieve comparable results. Finally, while reusability of the data is given through the publication on GitHub (Groß, 2025), the non-deterministic nature of LLMs means that identical output cannot be guaranteed, even with a temperature of 0. Apart from data limitations, there are issues in this study's workflow that need to be acknowledged. First, all geospatial calculations were done using the EPSG 3035 coordinate reference system. This is a projected system, which results in distortions, especially for areas further away from the center of the projection. Especially bounding boxes for big NUTS regions can differ significantly from the ones derived from a geographic coordinate system. Areas with a difference of > 5 % of bounding box area were therefore excluded from the bounding box question (see Notebook 07). Using EPSG 4326 could have addressed the issue in this study, however, as explained in section 5.3.5 not all geof: functions are working when using EPSG 4326.
Also, query generation in this study was missing a retry approach. In most other

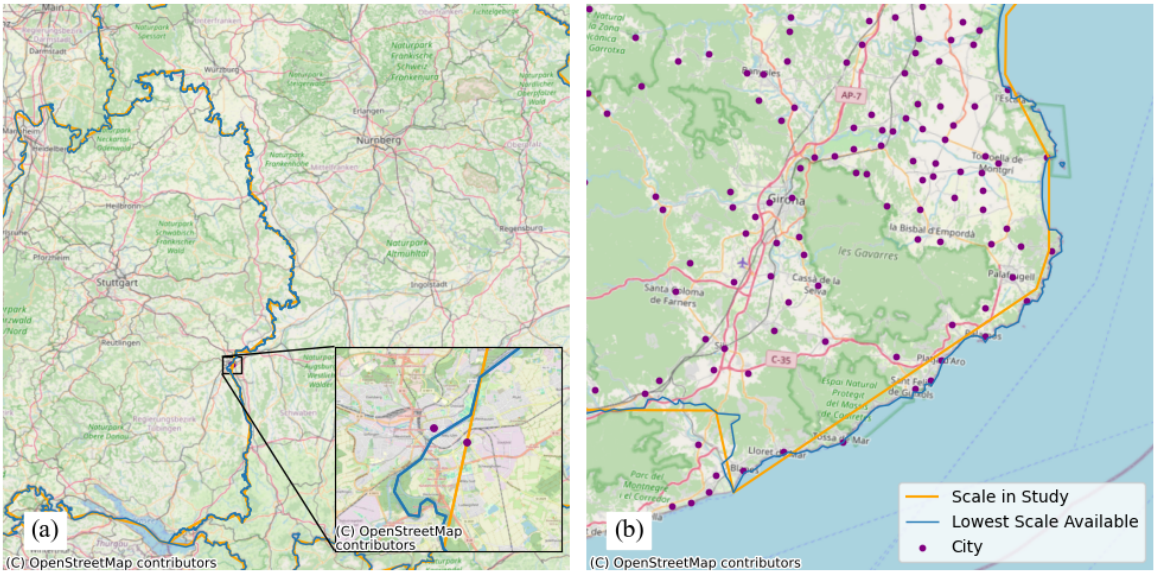


Figure 16: Inaccuracy in topological relationships regarding near-border cities. The city in figure **(a)** on the right (Neu-Ulm in Bavaria, Germany) is wrongly located to the left of the more generalized orange border. In figure **(b)**, some costal cities are not within the more generalized coastline and a few even fall outside the more detailed border.

studies regarding LLM based query generation, failed queries were sent to the LLM again to be evaluated for mistakes. This step is missing in this workflow and should be integrated in similar future processes. Finally, the model fine-tuning and the prompt engineering aspects are only analyzed on a surface level. Especially fine-tuning could have major implications for automated query generation if implemented more rigorously.

On the other hand, the limited setup also highlights the possibility to perform a comprehensive GraphRAG experiment with a single mid-range laptop running the database and orchestrating calls for almost 30,000 questions. Only the LLM calls were processed via API.

8 Conclusion and Future Work

8.1 Conclusion

Through the rapid popularization of LLMs during the last years, many new technical possibilities and research directions were opened up. Retrieval Augmented Generation emerged as one of the most common frameworks to improve LLM performance with many different subcategories being explored. One of those is GraphRAG, where semantic web technologies such as ontologies and knowledge graphs are used as a knowledge base for RAG instead of the more common unstructured documents. However, one of the most critical and difficult steps is the retrieval of relevant data from the knowledge graph. GIScience and GeoAI are well positioned in knowledge graph research, however GraphRAG for GIScience and GeoAI remains unexplored. This work addresses this gap.

The presented approach uses autonomous SPARQL query generation that also includes geofunctions to retrieve spatial information and spatial relationships from a knowledge graph. Three spatial concepts were explored: topology, directionality and proximity, as well as their combinations resulting in 18 questions. The questions are sent to an LLM together with the Turtle serialization of the graphs ontology and with the instructions to translate them into a SPARQL query. This query is then executed via the SPARQL GraphDB endpoint, returning the data needed to answer the question. This approach was compared to asking the question directly to the LLM without any additional modifications relying only on its internal knowledge. This non-RAG approach suffered from hallucinations and performed inconsistently over different questions. Some questions could be answered correctly and reliably while other, more difficult, questions performed badly or it were consistently wrong.

The RAG approach did perform considerably better for almost every question with improvements in F1 scores reaching up to 0.88. Also the RAG approach showed that when a generated query is executable and yields results it is a semantically correct query in most cases and if it fails, it does rarely provide wrong answers, only empty ones. It therefore makes the answers more explainable and reduces the risk of retrieving wrong answers for questions.

Furthermore different parameters and their effects on the results were evaluated. For both approaches different models (gpt-4o, gpt-4o-mini), temperatures (0 to 1.33), etc. were evaluated reaching expected results - bigger models with lower temperatures produce better queries. For the RAG approach, different prompting techniques, ontology versions and a custom fine tuned model were analyzed resulting in performance variations. The setups for prompting and ontology design for autonomous query generation

could still be improved and recent more powerful models could boost performance even further.
GraphRAG through autonomous query generation provides various advantages. Apart from the LLM, no computationally expensive step is needed, functionalities can be expanded by expanding the ontology and the knowledge graph or through providing middleware, similar to the presented approach for handling directionality questions. This work shows that the existing technology provides the basis for implementing GraphRAG LLMs for GIScience and its potential to answer spatially explicit questions more accurately and reliably.

8.2 Future Work

For future research directions many possibilities emerge. Other retrieval techniques such as similarity based ones should be evaluated for RAG-LLM use-cases in GIScience. Research into location-aware knowledge graph embedding is already prevalent (Mai et al., 2020) however the combination with RAG-LLMs is still a research gap. It has to be evaluated if an embedding based retriever can outperform the presented query generation approach.
To gain further insights into this works approach, multiple paths emerge. First, it has to be evaluated how this approach performs at scale. Can much larger ontologies be understood reliably by the LLM? For example the WorldKG ontology by Dsouza et al. (2021) consists of 6060 triples compared to 164 triples for this study's geonuts ontology. These 6060 triples already result in approx. 115000 tokens according to OpenAI's tokenizer (OpenAI, 2025), approaching the context limit of the gpt-4o and gpt-4o-mini models. If multiple large ontologies would be combined for future work, processes need to be developed that preselect ontology subsets (e.g. through semantic similarity) or trim them to use less tokens consistently. Ontology understanding could also be further advanced by fine-tuning LLM models specifically for ontology-based SPARQL generation, whereas the ontology itself could also be optimized further to achieve maximum comprehension by LLMs.
Future implementations of the GraphRAG approach should also include error handling. Most similar studies that use LLMs for query or code generation send multiple attempts to the LLM, if execution fails. This can be simple retries, or more complex ideas like building "experience based knowledge systems" as proposed by Akinboyewa et al. (2024) that draw on a catalog of known issues for error handling.
The approach could also be modified to prompt the LLM to split the workflow into multiple smaller tasks that can then be executed subsequently, resulting in simpler queries with higher success rates. The rigorous description of data using ontologies also forces data to adopt stricter metadata descriptions and standards. If adopting

semantic web technologies for (meta)data management proves to be advantageous for downstream LLM task, it could lead to better data quality for more datasets in the future. Furthermore, the successful use of the custom direction functions proves the possibility to make custom functions available to an LLM, only by describing their usage in the ontology. This opens the door to catalog existing and build new functionalities, and make them available for LLMs only by describing them using semantic web standards.

9 Acknowledgments

ChatGPT has been used as an assistant for coding and for minor language edits such as spellchecking. All text is mine.
OpenAI awarded a grant of 200 $ under the "OpenAI Researcher Access Program" for this thesis which covered all costs for the presented experiments.

10 References

Akinboyewa, T., Li, Z., Ning, H., & Lessani, M. N. (2024). *GIS Copilot: Towards an Autonomous GIS Agent for Spatial Analysis.* arXiv. (Version Number: 4) doi: 10.48550/ARXIV.2411.03205

alanr. (2024). *Does geof:buffer work? If so, can someone demonstrate a working query* [Forum post]. Retrieved 2025-03-12, from `https://stackoverflow.com/q/77773840`

Anselin, L. (1989). *What is special about spatial data? Alternative perspectives on spatial data analysis* (Tech. Rep. No. 89-4). UC Santa Barbara: National Center for Geographic Information and Analysis. Retrieved 2024-11-02, from `https://escholarship.org/uc/item/3ph5k0d4`

Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., & Ives, Z. (2007). DBpedia: A Nucleus for a Web of Open Data. In D. Hutchison et al. (Eds.), *The Semantic Web* (Vol. 4825, pp. 722–735). Berlin, Heidelberg: Springer Berlin Heidelberg. doi: 10.1007/978-3-540-76298-0_52

Baek, J., Aji, A. F., Lehmann, J., & Hwang, S. J. (2023). Direct Fact Retrieval from Knowledge Graphs without Entity Linking.

Belleau, F., Nolin, M.-A., Tourigny, N., Rigault, P., & Morissette, J. (2008). Bio2RDF: towards a mashup to build bioinformatics knowledge systems. *Journal of biomedical informatics*, *41*(5), 706–716. (Publisher: Academic Press) doi: 10.1016/j.jbi.2008.03.004

Berners-Lee, T. (2006). *Linked Data.* Retrieved 2024-06-05, from `https://www.w3.org/DesignIssues/LinkedData.html`

Bernhard, H., & Antoine, I. (2011). data.europeana.eu: The Europeana Linked Open Data Pilot.. doi: 10.23106/DCMI.952135673

Bizer, C., Heath, T., & Berners-Lee, T. (2009). Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems*, *5*(3), 1–22. doi: 10.4018/jswis.2009081901

Car, N. J., Homburg, T., Perry, M., Knibbe, F., Cox, S., Abhayaratna, J., … Janowicz, K. (2023). *OGC GeoSPARQL - A Geographic Query Language for RDF Data.* Retrieved 2024-05-29, from `http://www.opengis.net/doc/IS/geosparql/1.1`

Chen, J., Lin, H., Han, X., & Le Sun. (2024). Benchmarking Large Language Models in Retrieval-Augmented Generation. *Proceedings of the AAAI Conference on Artificial Intelligence*, *38*(16), 17754–17762. (Publisher: Association for the Advancement of Artificial Intelligence (AAAI)) doi: 10.1609/aaai.v38i16.29728

Compton, M., Barnaghi, P., Bermudez, L., García-Castro, R., Corcho, O., Cox, S., … Taylor, K. (2012). The SSN ontology of the W3C semantic sensor network incubator group. *Journal of Web Semantics*, *17*, 25–32. doi: 10.1016/j.websem.2012.05.003

DeepSeek. (2025). *DeepSeek-R1 Release | DeepSeek API Docs.* Retrieved 2025-03-06, from `https://api-docs.deepseek.com/news/news250120`

Dekkers, M., & Novacean, I. (2018). *D04.02.1: NUTS RDF Model revised.* Retrieved 2024-05-14, from `https://joinup.ec.europa.eu/sites/default/files/document/2018-03/D04.02.1%20NUTS%20RDF%20Model%20revised%20v1.00.pdf` (Publication Title: SC353DI07171)

De Meester, B., Seymoens, T., Dimou, A., & Verborgh, R. (2020). Implementation-independent function reuse. *Future Generation Computer Systems*, *110*, 946–959. doi: 10.1016/j.future.2019.10.006

Deng, C., Jia, Y., Xu, H., Zhang, C., Tang, J., Fu, L., … Zhou, C. (2021). GAKG: A Multimodal Geoscience Academic Knowledge Graph. In G. Demartini, G. Zuccon, J. S. Culpepper, Z. Huang, & H. Tong (Eds.), (pp. 4445–4454). Virtual Event Queensland Australia: ACM. doi: 10.1145/3459637.3482003

Deng, C., Zhang, T., He, Z., Chen, Q., Shi, Y., Xu, Y., … He, J. (2024). K2: A Foundation Language Model for Geoscience Knowledge Understanding and Utilization. In L. Angélica, S. Lattanzi, A. Muñoz Medina, L. Akoglu, A. Gionis, & S. Vassilvitskii (Eds.), (pp. 161–170). Merida Mexico: Association for Computing Machinery. doi: 10.1145/3616855.3635772

Diallo, P. A. K. K., Reyd, S., & Zouaq, A. (2023). A Comprehensive Evaluation of Neural SPARQL Query Generation from Natural Language Questions.

Dsouza, A., Tempelmeier, N., Yu, R., Gottschalk, S., & Demidova, E. (2021). WorldKG: A World-Scale Geographic Knowledge Graph. , *221*, 4475–4484. doi: 10.1145/3459637.3482023

Duarte, F. (2025). *Number of ChatGPT Users (Feb 2025).* Retrieved 2025-03-06, from `https://explodingtopics.com/blog/chatgpt-users`

Emonet, V., Bolleman, J., Duvaud, S., de Farias, T. M., & Sima, A. C. (2024). *LLM-based SPARQL Query Generation from Natural Language over Federated Knowledge Graphs.* arXiv. Retrieved 2025-02-16, from https://arxiv.org/abs/2410.06062 (Version Number: 4) doi: 10.48550/ARXIV.2410.06062

Erxleben, F., Günther, M., Krötzsch, M., Mendez, J., & Vrandečić, D. (2014). Introducing Wikidata to the Linked Data Web. In P. Mika et al. (Eds.), *The Semantic Web – ISWC 2014* (Vol. 8796, pp. 50–65). Cham: Springer International Publishing. doi: 10.1007/978-3-319-11964-9_4

Eurostat. (2024). *Nomenclature of Territorial Units for Statistics - EU Vocabularies - Publications Office of the EU.* Retrieved 2025-03-04, from https://op.europa.eu/en/web/eu-vocabularies/dataset/-/resource?uri=http://publications.europa.eu/resource/dataset/nuts

Eurostat. (2024). *NUTS - Nomenclature of territorial units for statistics.* Retrieved 2025-03-04, from https://ec.europa.eu/eurostat/web/nuts

Fellbaum, C. (2010). WordNet. In R. Poli, M. Healy, & A. Kameas (Eds.), *Theory and Applications of Ontology: Computer Applications* (pp. 231–243). Dordrecht: Springer Netherlands. doi: 10.1007/978-90-481-8847-5_10

Fulman, N., Memduhoğlu, A., & Zipf, A. (2024a). *Distortions in Judged Spatial Relations in Large Language Models.* arXiv. Retrieved 2024-11-04, from https://arxiv.org/abs/2401.04218 (Version Number: 2) doi: 10.48550/ARXIV.2401.04218

Fulman, N., Memduhoğlu, A., & Zipf, A. (2024b). Evidence for Systematic Bias in the Spatial Memory of Large Language Models. In G. McDonald, C. Macdonald, & I. Ounis (Eds.), . Glasgow, Scotland.

Gao, S., Janowicz, K., Montello, D. R., Hu, Y., Yang, J.-A., McKenzie, G., … Yan, B. (2017). A data-synthesis-driven method for detecting and extracting vague cognitive regions. *International Journal of Geographical Information Science*, 1–27. doi: 10.1080/13658816.2016.1273357

Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., … Wang, H. (2023). Retrieval-Augmented Generation for Large Language Models: A Survey.

Geonames. (2024). *Geonames.* Retrieved 2024-09-21, from https://www.geonames.org/about.html

Geonames. (2025a). *GeoNames Data Dump.* Retrieved 2025-03-04, from https://download.geonames.org/export/dump/

Geonames. (2025b). *GeoNames Feature Codes.* Retrieved 2025-05-12, from https://www.geonames.org/export/codes.html

Geonames. (2025c). *GeoNames Ontology - Geo Semantic Web.* Retrieved 2025-03-12, from https://www.geonames.org/ontology/documentation.html

Goodchild, M. (2001). Issues in spatially explicit modeling. *Agent-based models of land-use and land-cover change*, 13–17.

Google. (2025). *Enterprise Knowledge Graph documentation.* Retrieved 2025-03-06, from https://cloud.google.com/enterprise-knowledge-graph/docs

Goyal, R. K., & Egenhofer, M. J. (2000). Consistent queries over cardinal directions across different levels of detail. In (pp. 876–880). London, UK: IEEE Comput. Soc. doi: 10.1109/DEXA.2000.875129

Groß, S. (2025). *simon-gross/master_thesis: Spatial GraphRAG.* Zenodo. Retrieved 2025-05-21, from https://zenodo.org/doi/10.5281/zenodo.15484449 doi: 10.5281/ZENODO.15484449

Hadi, M. U., Tashi, Q. A., Shah, A., Qureshi, R., Muneer, A., Irfan, M., … Shah, M. (2024). *Large Language Models: A Comprehensive Survey of its Applications, Challenges, Limitations, and Future Prospects.* Preprints. Retrieved 2024-12-09, from https://www.techrxiv.org/users/618307/articles/682263-large-language-models-a-comprehensive-survey-of-its-applications-challenges-limitations-and-future-prospects?commit=a587f80f85944c90bec6a59264fbe6c0c37ad9bd doi: 10.36227/techrxiv.23589741.v6

Hendler, J., Lassila, O., & Berners-Lee, T. (2001). The Semantic Web. *Scientific American*, *284*(5), 34–43.

Hinton, G., Vinyals, O., & Dean, J. (2015). *Distilling the Knowledge in a Neural Network.* arXiv. Retrieved 2025-05-21, from https://arxiv.org/abs/1503.02531 (Version Number: 1) doi: 10.48550/ARXIV.1503.02531

Hitzler, P. (2021). A review of the semantic web field. *Communications of the ACM*, *64*(2), 76–83. doi: 10.1145/3397512

Hogan, A., Blomqvist, E., Cochez, M., D'amato, C., Melo, G. D., Gutierrez, C., … Zimmermann, A. (2022). Knowledge Graphs. *ACM Computing Surveys*, *54*(4), 1–37. doi: 10.1145/3447772

Horrocks, I. (2008). Ontologies and the semantic web. *Communications of the ACM*, *51*(12), 58–67. doi: 10.1145/1409360.1409377

Hu, K. (2023). ChatGPT sets record for fastest-growing user base - analyst note. *Reuters*.

Janowicz, K., Gao, S., McKenzie, G., Hu, Y., & Bhaduri, B. (2020). GeoAI: spatially explicit artificial intelligence techniques for geographic knowledge discovery and beyond. *International Journal of Geographical Information Science*, *34*(4), 625–636. doi: 10.1080/13658816.2019.1684500

Janowicz, K., Haller, A., Cox, S. J., Le Phuoc, D., & Lefrançois, M. (2019). SOSA: A lightweight ontology for sensors, observations, samples, and actuators. *Journal of Web Semantics*, *56*, 1–10. doi: 10.1016/j.websem.2018.06.003

Janowicz, K., Hitzler, P., Li, W., Rehberger, D., Schildhauer, M., Zhu, R., … Currier, K. (2022). Know, Know Where, KnowWhereGraph: A densely connected, cross-domain knowledge graph and geo-enrichment service stack for applications in environmental intelligence. *AI Magazine*, *43*(1), 30–39. doi: 10.1002/aaai.12043

Ji, S., Pan, S., Cambria, E., Marttinen, P., & Yu, P. S. (2022). A Survey on Knowledge Graphs: Representation, Acquisition, and Applications. *IEEE Transactions on Neural Networks and Learning Systems*, *33*(2), 494–514. doi: 10.1109/TNNLS.2021.3070843

Jiang, Y., & Yang, C. (2024). Is ChatGPT a Good Geospatial Data Analyst? Exploring the Integration of Natural Language into Structured Query Language within a Spatial Database. *ISPRS International Journal of Geo-Information*, *13*(1), 26. doi: 10.3390/ijgi13010026

Jiang, Z., Xu, F. F., Gao, L., Sun, Z., Liu, Q., Dwivedi-Yu, J., … Neubig, G. (2023). Active Retrieval Augmented Generation.

Keßler, C., Raubal, M., & Wosniok, C. (2009). Semantic Rules for Context-Aware Geographical Information Retrieval. In P. Barnaghi, K. Moessner, M. Presser, & S. Meissner (Eds.), *Smart Sensing and Context* (Vol. 5741, pp. 77–92). Berlin, Heidelberg: Springer Berlin Heidelberg. (Series Title: Lecture Notes in Computer Science) doi: 10.1007/978-3-642-04471-7_7

Kosten, C., Nooralahzadeh, F., & Stockinger, K. (2025). Evaluating the effectiveness of prompt engineering for knowledge graph question answering. *Frontiers in Artificial Intelligence*, *7*, 1454258. doi: 10.3389/frai.2024.1454258

Kuhn, W. (2012). Core concepts of spatial information for transdisciplinary research. *International Journal of Geographical Information Science*, *26*(12), 2267–2276. (Publisher: Taylor & Francis) doi: 10.1080/13658816.2012.722637

Kuhn, W., Kauppinen, T., & Janowicz, K. (2014). Linked Data - A Paradigm Shift for Geographic Information Science. In M. Duckham, E. Pebesma, K. Stewart, & A. U. Frank (Eds.), *Geographic Information Science* (pp. 173–186). Cham: Springer International Publishing. doi: 10.1007/978-3-319-11593-1_12

LangChain. (2024). *Introduction.* Retrieved 2024-12-12, from `https://python.langchain.com/docs/get_started/introduction`

Lehmann, J., Gattogi, P., Bhandiwad, D., Ferré, S., & Vahdati, S. (2023). Language Models as Controlled Natural Language Semantic Parsers for Knowledge Graph Question Answering. In (pp. 1348–1356). IOS Press. doi: 10.3233/FAIA230411

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., … Kiela, D. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks.

Li, W., Arundel, S., Gao, S., Goodchild, M., Hu, Y., Wang, S., & Zipf, A. (2024). GeoAI for Science and the Science of GeoAI. *Journal of Spatial Information Science*(29), 1–17. doi: 10.5311/JOSIS.2024.29.349

Li, Z., & Ning, H. (2023). Autonomous GIS: the next-generation AI-powered GIS. *International Journal of Digital Earth*, *16*(2), 4668–4686. doi: 10.1080/17538947.2023.2278895

Li, Z., Ning, H., Gao, S., Janowicz, K., Li, W., Arundel, S. T., … Hodgson, M. E. (2025). *GIScience in the Era of Artificial Intelligence: A Research Agenda Towards Autonomous GIS.* arXiv. Retrieved 2025-05-15, from `https://arxiv.org/abs/2503.23633` (Version Number: 5) doi: 10.48550/ARXIV.2503.23633

Liu, W., Zhou, P., Zhao, Z., Wang, Z., Ju, Q., Deng, H., & Wang, P. (2020). K-BERT: Enabling Language Representation with Knowledge Graph. *Proceedings of the AAAI Conference on Artificial Intelligence*, *34*(03), 2901–2908. doi: 10.1609/aaai.v34i03.5681

Liu, Z., Gu, Z., Thelen, T., Estrecha, S. G., Zhu, R., Fisher, C. K., … Hitzler, P. (2022). Knowledge explorer. In M. Renz & M. Sarwat (Eds.), (pp. 1–4). Seattle Washington: ACM. doi: 10.1145/3557915.3561009

Liu, Z., Janowicz, K., Cai, L., Zhu, R., Mai, G., & Shi, M. (2022). Geoparsing: Solved or Biased? An Evaluation of Geographic Biases in Geoparsing. *AGILE: GIScience Series*, *3*, 1–13. doi: 10.5194/agile-giss-3-9-2022

Liu, Z., Janowicz, K., Majic, I., Shi, M., Fortacz, A., Karimi, M., … Currier, K. (2025). Operationalizing Geographic Diversity for the Evaluation of AI-Generated Content. *Transactions in GIS*, *29*(3), e70057. doi: 10.1111/tgis.70057

logicnet. (2023). *Diagram Designer.* Retrieved 2025-03-04, from `https://sourceforge.net/projects/diagramdesigner/`

Mai, G., Janowicz, K., Cai, L., Zhu, R., Regalia, B., Yan, B., … Lao, N. (2020). SE-KGE : A location-aware Knowledge Graph Embedding model for Geographic Question Answering and Spatial Semantic Lifting. *Transactions in GIS*, *24*(3), 623–655. (Publisher: John Wiley & Sons, Ltd) doi: 10.1111/tgis.12629

Mai, G., Janowicz, K., Hu, Y., Gao, S., Yan, B., Zhu, R., … Lao, N. (2022). A review of location encoding for GeoAI: methods and applications. *International Journal of Geographical Information Science*, *36*(4), 639–673. doi: 10.1080/13658816.2021.2004602

Mansourian, A., & Oucheikh, R. (2024). ChatGeoAI: Enabling Geospatial Analysis for Public through Natural Language, with Large Language Models. *ISPRS International Journal of Geo-Information*, *13*(10), 348. doi: 10.3390/ijgi13100348

Manvi, R., Khanna, S., Burke, M., Lobell, D., & Ermon, S. (2024). *Large Language Models are Geographically Biased.* arXiv. Retrieved 2024-11-04, from `https://arxiv.org/abs/2402.02680` (Version Number: 2) doi: 10.48550/ARXIV.2402.02680

McCrae, J. P. (2024). *The Linked Open Data Cloud.* Retrieved 2024-09-21, from `https://lod-cloud.net/`

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). *Efficient Estimation of Word Representations in Vector Space.* arXiv. Retrieved 2025-03-11, from `http://arxiv.org/abs/1301.3781` (arXiv:1301.3781 [cs]) doi: 10.48550/arXiv.1301.3781

Neo4j. (2025). *What is Cypher - Getting Started.* Retrieved 2025-02-26, from `https://neo4j.com/docs/getting-started/cypher/`

OGC. (2024). *ogc-geosparql GitHub Repository.* Retrieved 2025-05-18, from `https://github.com/opengeospatial/ogc-geosparql/blob/master/vocabularies/functions.ttl`

OGC. (2025). *geoSPARQL Geofunctions.* Retrieved 2025-03-18, from https://defs.opengis.net/vocprez/object?uri=https://www.opengis.net/def/function/geosparql/

Ontotext. (2024). *GraphDB.* Retrieved 2024-04-30, from https://graphdb.ontotext.com/

Ontotext. (2025a). *GeoSPARQL support — GraphDB 10.8 documentation.* Retrieved 2025-03-04, from https://graphdb.ontotext.com/documentation/10.8/geosparql-support.html

Ontotext. (2025b). *Talk to Your Graph — GraphDB 10.8 documentation.* Retrieved 2025-03-09, from https://graphdb.ontotext.com/documentation/10.8/talk-to-graph.html

OpenAI. (2022). *Introducing ChatGPT.* Retrieved 2024-12-08, from https://openai.com/index/chatgpt/

OpenAI. (2025). *OpenAI Platform.* Retrieved 2025-03-11, from https://platform.openai.com

OpenStreetMap. (2025). *OpenStreetMap.* Retrieved 2025-03-03, from https://www.openstreetmap.org/

Pan, S., Luo, L., Wang, Y., Chen, C., Wang, J., & Wu, X. (2024). Unifying Large Language Models and Knowledge Graphs: A Roadmap. *IEEE Transactions on Knowledge and Data Engineering*, 1–20. doi: 10.1109/TKDE.2024.3352100

Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1532–1543). Doha, Qatar: Association for Computational Linguistics. doi: 10.3115/v1/D14-1162

Pichai, S., & Hassabis, D. (2023). *Introducing Gemini: Google's most capable AI model yet.* Retrieved 2025-03-06, from https://blog.google/technology/ai/google-gemini-ai/#sundar-note

Procko, T. T., & Ochoa, O. (2024). Graph Retrieval-Augmented Generation for Large Language Models: A Survey. In *2024 Conference on AI, Science, Engineering, and Technology (AIxSET)* (pp. 166–169). Laguna Hills, CA, USA: IEEE. doi: 10.1109/AIxSET62544.2024.00030

QGIS. (2025). *Spatial without Compromise · QGIS Web Site.* Retrieved 2025-03-04, from https://qgis.org/

QSRlib. (2025). *Region Connection Calculus 8 — QSRlib 0.4.0 documentation.* Retrieved 2025-05-21, from https://qsrlib.readthedocs.io/en/latest/rsts/handwritten/qsrs/rcc8.html

Rangel, J. C., Farias, T. M., Sima, A. C., & Kobayashi, N. (2024). SPARQL Generation: an analysis on fine-tuning OpenLLaMA for Question Answering over a Life Science Knowledge Graph.

RDF Working Group. (2014a). *RDF 1.1 Concepts and Abstract Syntax.* Retrieved 2024-06-01, from https://www.w3.org/TR/rdf11-concepts/#data-model

RDF Working Group. (2014b). *Resource Description Framework (RDF).* Retrieved 2024-06-01, from https://www.w3.org/RDF/

Roberts, J., Lüddecke, T., Das, S., Han, K., & Albanie, S. (2023). GPT4GEO: How a Language Model Sees the World's Geography.

Salemi, A., & Zamani, H. (2024). Evaluating Retrieval Quality in Retrieval-Augmented Generation. In G. Hui Yang, H. Wang, S. Han, C. Hauff, G. Zuccon, & Y. Zhang (Eds.), (pp. 2395–2400). Washington DC USA: ACM. doi: 10.1145/3626772.3657957

Shi, M., Currier, K., Liu, Z., Janowicz, K., Wiedemann, N., Verstegen, J., … Mai, G. (2023). Thinking Geographically about AI Sustainability. *AGILE: GIScience Series*, *4*, 1–7. doi: 10.5194/agile-giss-4-42-2023

Soudani, H., Kanoulas, E., & Hasibi, F. (2024). *Fine Tuning vs. Retrieval Augmented Generation for Less Popular Knowledge.* arXiv. Retrieved 2024-09-21, from http://arxiv.org/abs/2403.01432 (arXiv:2403.01432 [cs])

Stanford University. (2025). *Protégé.* Retrieved 2025-03-04, from https://protege.stanford.edu/

STKO Lab UC Santa Barbara. (2023). *GeoMachina Announcement.* Retrieved 2024-08-31, from https://x.com/STKO_UCSB/status/1697299575487250707

Tobler, W. R. (1979). Cellular Geography. In S. Gale & G. Olsson (Eds.), *Philosophy in Geography* (pp. 379–386). Dordrecht: Springer Netherlands. doi: 10.1007/978-94-009-9394-5_18

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., … Lample, G. (2023). *LLaMA: Open and Efficient Foundation Language Models.* arXiv. Retrieved 2025-03-02, from http://arxiv.org/abs/2302.13971 (arXiv:2302.13971 [cs]) doi: 10.48550/arXiv.2302.13971

Uschold, M., & Gruninger, M. (1996). Ontologies: principles, methods and applications. *The Knowledge Engineering Review*, *11*(2), 93–136. doi: 10.1017/S0269888900007797

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., … Polosukhin, I. (2017). Attention is all you need. In I. Guyon et al. (Eds.), *Advances in neural information processing systems* (Vol. 30). Curran Associates, Inc.

VisualDataWeb. (2025). *WebVOWL.* Retrieved 2025-03-26, from `https://service.tib.eu/webvowl/`

W3C. (2017). *Shapes Constraint Language (SHACL).* Retrieved 2025-02-26, from `https://www.w3.org/TR/shacl/`

W3C. (2024). *RDF 1.1 Turtle.* Retrieved 2024-06-01, from `https://www.w3.org/TR/2014/REC-turtle-20140225/`

W3C. (2025). *Shape Expressions Language 2.1.* Retrieved 2025-03-18, from `https://shex.io/shex-semantics/`

Wang, M., Qiu, L., & Wang, X. (2021). A Survey on Knowledge Graph Embeddings for Link Prediction. *Symmetry*, *13*(3), 485. doi: 10.3390/sym13030485

Wang, S., Zhang, X., Ye, P., Du, M., Lu, Y., & Xue, H. (2019). Geographic Knowledge Graph (GeoKG): A Formalized Geographic Knowledge Representation. *ISPRS International Journal of Geo-Information*, *8*(4), 184. doi: 10.3390/ijgi8040184

Wang, X., Yang, Q., Qiu, Y., Liang, J., He, Q., Gu, Z., … Wang, W. (2023). KnowledGPT: Enhancing Large Language Models with Retrieval and Storage Access on Knowledge Bases. , 21.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., … Zhou, D. (2022). *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models.* arXiv. Retrieved 2025-02-17, from `https://arxiv.org/abs/2201.11903` (Version Number: 6) doi: 10.48550/ARXIV.2201.11903

Wikipedia. (2025). *List of cities in the European Union by population within city limits.* Retrieved 2025-03-11, from `https://en.wikipedia.org/w/index.php?title=List_of_cities_in_the_European_Union_by_population_within_city_limits&oldid=1278619758` (Page Version ID: 1278619758)

World Bank. (2025). *How do we define cities, towns, and rural areas?* Retrieved 2025-03-18, from `https://blogs.worldbank.org/en/sustainablecities/how-do-we-define-cities-towns-and-rural-areas`

Xu, Z., Cruz, M. J., Guevara, M., Wang, T., Deshpande, M., Wang, X., & Li, Z. (2024). Retrieval-Augmented Generation with Knowledge Graphs for Customer Service Question Answering. In (pp. 2905–2909). ACM. doi: 10.1145/3626772.3661370

Zhu, Y., Li, J., Li, G., Zhao, Y., Li, J., Jin, Z., & Mei, H. (2024). Hot or Cold? Adaptive Temperature Sampling for Code Generation with Large Language Models. *Proceedings of the AAAI Conference on Artificial Intelligence*, *38*(1), 437–445. doi: 10.1609/aaai.v38i1.27798

11 Appendix

Prefix	Abbreviated IRIs
dcat:	http://www.w3.org/ns/dcat#
geo:	http://www.opengis.net/ont/geosparql#
fno:	https://w3id.org/function/ontology#
geof:	http://www.opengis.net/def/function/geosparql/
gn:	https://www.geonames.org/ontology#
nutsdef:	http://data.europa.eu/nuts/
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#
skos:	http://www.w3.org/2004/02/skos/core#
xsd:	http://www.w3.org/2001/XMLSchema#

Table 12: List of prefixes used in this study.

```
@prefix fno: <https://w3id.org/function/ontology#> .
@prefix geo: <http://www.opengis.net/ont/geosparql#> .
@prefix geof: <http://www.opengis.net/def/function/geosparql/> .
@prefix gn: <https://www.geonames.org/ontology#> .
@prefix nutsdef: <http://data.europa.eu/nuts/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

nutsdef:level a owl:DatatypeProperty ;
    rdfs:domain skos:Concept ;
    skos:definition "A value that indicates the level of the Code;
        permissible values are '0', '1', '2' or '3'." ;
    skos:note "The Document defining this data property is only available
        as a PDF. No IRI is given and there is no other permanent resource
         for the definition." .

<http://geonuts.eu#> a owl:Ontology .

geof:buffer a fno:Function ;
    skos:definition "A query function that returns a buffer around the
        input geometry."@en ;
```

```
    skos:example "Example usage of a SPARQL query: geof:buffer(?wkt, 1000)
        " ;
    skos:note "When a function expects a geo:wktLiteral it must be a geo:
        wktLiteral, a geo:Geometry object will not work." ;
    skos:prefLabel "buffer"@en ;
    fno:expects "(geo:wktLiteral xsd:double)" ;
    fno:returns "(geo:wktLiteral)" .

geof:distance a fno:Function ;
    skos:definition "A query function that returns the distance between
        the two closest points of the input geometries."@en ;
    skos:example "Example usage of a SPARQL query: geof:distance(?wkt, ?
        wkt2)" ;
    skos:note "When a function expects a geo:wktLiteral it must be a geo:
        wktLiteral, a geo:Geometry object will not work." ;
    skos:prefLabel "distance"@en ;
    fno:expects "(geo:wktLiteral geo:wktLiteral)" ;
    fno:returns "(xsd:double)" .

geof:envelope a fno:Function ;
    skos:definition "A query function that returns the minimum bounding
        rectangle of the input geometry."@en ;
    skos:example "Example usage of a SPARQL query: geof:envelope(?wkt)" ;
    skos:note "When a function expects a geo:wktLiteral it must be a geo:
        wktLiteral, a geo:Geometry object will not work." ;
    skos:prefLabel "envelope"@en ;
    fno:expects "(geo:wktLiteral)" ;
    fno:returns "(geo:wktLiteral)" .

geof:isEastOf a fno:Function ;
    skos:definition "A function that identifies if one geographic object
        is east of another geographic object."@en ;
    skos:example "Example usage in a SPARQL query: geof:isEastOf(?wkt, ?
        wkt2)" ;
    skos:prefLabel "is east of"@en ;
    fno:expects "(geo:wktLiteral geo:wktLiteral)" ;
    fno:returns "(xsd:boolean)" .

geof:isNorthOf a fno:Function ;
    skos:definition "A function that identifies if one geographic object
        is north of another geographic object."@en ;
    skos:example "Example usage in a SPARQL query: geof:isNorthOf(?wkt, ?
        wkt2)" ;
    skos:prefLabel "is north of"@en ;
    fno:expects "(geo:wktLiteral geo:wktLiteral)" ;
    fno:returns "(xsd:boolean)" .
```

```
geof:isSouthOf a fno:Function ;
    skos:definition "A function that identifies if one geographic object
        is south of another geographic object."@en ;
    skos:example "Example usage in a SPARQL query: geof:isSouthOf(?wkt, ?
        wkt2)" ;
    skos:prefLabel "is south of"@en ;
    fno:expects "(geo:wktLiteral geo:wktLiteral)" ;
    fno:returns "(xsd:boolean)" .

geof:isWestOf a fno:Function ;
    skos:definition "A function that identifies if one geographic object
        is west of another geographic object."@en ;
    skos:example "Example usage in a SPARQL query: geof:isWestOf(?wkt, ?
        wkt2)" ;
    skos:prefLabel "is west of"@en ;
    fno:expects "(geo:wktLiteral geo:wktLiteral)" ;
    fno:returns "(xsd:boolean)" .

geof:sfContains a fno:Function ;
    skos:definition """A query function that returns true if the first
        geometry argument spatially contains the second geometry argument.

DE-9IM: T*****FF*"""@en ;
    skos:example "Example usage of a SPARQL query: geof:sfContains(?wkt, ?
        wkt2)" ;
    skos:note "When a function expects a geo:wktLiteral it must be a geo:
        wktLiteral, a geo:Geometry object will not work." ;
    skos:prefLabel "contains"@en ;
    fno:expects "(geo:wktLiteral geo:wktLiteral)" ;
    fno:returns "(xsd:boolean)" .

geof:sfCrosses a fno:Function ;
    skos:definition """A query function that returns true if the first
        geometry argument spatially crosses the second geometry argument.

DE-9IM: T*T***T**"""@en ;
    skos:example "Example usage of a SPARQL query: geof:sfCrosses(?wkt, ?
        wkt2)" ;
    skos:note "When a function expects a geo:wktLiteral it must be a geo:
        wktLiteral, a geo:Geometry object will not work." ;
    skos:prefLabel "crosses"@en ;
    fno:expects "(geo:wktLiteral geo:wktLiteral)" ;
    fno:returns "(xsd:boolean)" .

geof:sfDisjoint a fno:Function ;
```

```
    skos:definition """A query function that returns true if the input
        geometries are disjoint.

DE-9IM: FF*FF****"""@en ;
    skos:example "Example usage of a SPARQL query: geof:sfDisjoint(?wkt, ?
        wkt2)" ;
    skos:note "When a function expects a geo:wktLiteral it must be a geo:
        wktLiteral, a geo:Geometry object will not work." ;
    skos:prefLabel "disjoint"@en ;
    fno:expects "(geo:wktLiteral geo:wktLiteral)" ;
    fno:returns "(xsd:boolean)" .

geof:sfEquals a fno:Function ;
    skos:definition """A query function that returns true if the input
        geometries are equal.

DE-9IM: TFFFTFFFT"""@en ;
    skos:example "Example usage of a SPARQL query: geof:sfEquals(?wkt, ?
        wkt2)" ;
    skos:note "When a function expects a geo:wktLiteral it must be a geo:
        wktLiteral, a geo:Geometry object will not work." ;
    skos:prefLabel "equals"@en ;
    fno:expects "(geo:wktLiteral geo:wktLiteral)" ;
    fno:returns "(xsd:boolean)" .

geof:sfIntersects a fno:Function ;
    skos:definition """A query function that returns true if the input
        geometries intersect.

DE-9IM: T******** ^ *T******* ^ ***T***** ^ ****T**** """@en ;
    skos:example "Example usage of a SPARQL query: geof:sfIntersects(?wkt,
         ?wkt2)" ;
    skos:note "When a function expects a geo:wktLiteral it must be a geo:
        wktLiteral, a geo:Geometry object will not work." ;
    skos:prefLabel "intersects"@en ;
    fno:expects "(geo:wktLiteral geo:wktLiteral)" ;
    fno:returns "(xsd:boolean)" .

geof:sfOverlaps a fno:Function ;
    skos:definition """A query function that returns true if the first
        geometry argument spatially overlaps the second geometry argument.

DE-9IM: T*T***T** """@en ;
    skos:example "Example usage of a SPARQL query: geof:sfOverlaps(?wkt, ?
        wkt2)" ;
```

```
    skos:note "When a function expects a geo:wktLiteral it must be a geo:
        wktLiteral, a geo:Geometry object will not work." ;
    skos:prefLabel "overlaps"@en ;
    fno:expects "(geo:wktLiteral geo:wktLiteral)" ;
    fno:returns "(xsd:boolean)" .

geof:sfTouches a fno:Function ;
    skos:definition """A query function that returns true if the input
        geometries touch.

DE-9IM: FT******* ^ F**T***** ^ F***T****"""@en ;
    skos:example "Example usage of a SPARQL query: geof:sfTouches(?wkt, ?
        wkt2)" ;
    skos:note "When a function expects a geo:wktLiteral it must be a geo:
        wktLiteral, a geo:Geometry object will not work." ;
    skos:prefLabel "touches"@en ;
    fno:expects "(geo:wktLiteral geo:wktLiteral)" ;
    fno:returns "(xsd:boolean)" .

geof:sfWithin a fno:Function ;
    skos:definition """A query function that returns true if the first
        geometry argument is spatially within the second geometry argument
        .

DE-9IM: T*F**F***"""@en ;
    skos:example "Example usage of a SPARQL query: geof:sfWithin(?wkt, ?
        wkt2)" ;
    skos:note "When a function expects a geo:wktLiteral it must be a geo:
        wktLiteral, a geo:Geometry object will not work." ;
    skos:prefLabel "within"@en ;
    fno:expects "(geo:wktLiteral geo:wktLiteral)" ;
    fno:returns "(xsd:boolean)" .

geo:asWKT a owl:DatatypeProperty ;
    rdfs:domain geo:Geometry ;
    rdfs:range geo:wktLiteral ;
    rdfs:subPropertyOf owl:topDataProperty ;
    skos:definition "The WKT serialization of a Geometry"@en ;
    skos:prefLabel "as WKT"@en .

geo:hasGeometry a owl:ObjectProperty ;
    rdfs:domain geo:Feature,
        gn:Feature ;
    rdfs:range geo:Geometry ;
    skos:definition "A spatial representation for a given Feature."@en ;
    skos:prefLabel "has geometry"@en .
```

```
skos:notation a owl:DatatypeProperty ;
    rdfs:label "notation" ;
    rdfs:domain skos:Concept ;
    skos:definition "A notation, also known as classification code, is a
        string of characters such as \"T58.5\" or \"303.4833\" used to
        uniquely identify a concept within the scope of a given concept
        scheme."@en .

gn:geonamesID a owl:DatatypeProperty ;
    rdfs:label "geonames identifier"@en ;
    rdfs:domain gn:Feature ;
    skos:historyNote "Added in version 3.1"@en .

gn:name a owl:DatatypeProperty ;
    rdfs:label "name"@en ;
    rdfs:comment "The main international name of a feature. The value has
        no xml:lang tag."@en ;
    rdfs:domain gn:Feature ;
    rdfs:range xsd:string .

gn:population a owl:DatatypeProperty ;
    rdfs:label "population"@en ;
    rdfs:domain gn:Feature ;
    rdfs:range xsd:integer .

geo:Feature a owl:Class ;
    rdfs:label "GeoSPARQL Feature" ;
    rdfs:subClassOf geo:SpatialObject ;
    owl:disjointWith geo:Geometry ;
    owl:equivalentClass skos:Concept ;
    skos:definition "A discrete spatial phenomenon in a universe of
        discourse."@en ;
    skos:note "A Feature represents a uniquely identifiable phenomenon,
        for example a river or an apple. While such phenomena (and
        therefore the Features used to represent them) are bounded, their
        boundaries may be crisp (e.g., the declared boundaries of a state)
        , vague (e.g., the delineation of a valley versus its neighboring
        mountains), and change with time (e.g., a storm front). While
        discrete in nature, Features may be created from continuous
        observations, such as an isochrone that determines the region that
         can be reached by ambulance within 5 minutes."@en ;
    skos:prefLabel "GeoSPARQL Feature"@en .

geo:wktLiteral a rdfs:Datatype ;
```

```
    skos:definition "A Well-known Text serialization of a Geometry object
        ."@en ;
    skos:prefLabel "Well-known Text Literal"@en .

geo:Geometry a owl:Class ;
    rdfs:subClassOf geo:SpatialObject ;
    skos:definition "A coherent set of direct positions in space. The
        positions are held within a Spatial Reference System (SRS)."@en ;
    skos:note "Geometries should be represented using the geo:wktLiteral
        when using geof:Functions" ;
    skos:prefLabel "Geometry"@en .

skos:Concept a owl:Class ;
    rdfs:label "NUTS Code"@en ;
    rdfs:subClassOf geo:SpatialObject ;
    skos:definition "An idea or notion; a unit of thought."@en .

geo:SpatialObject a owl:Class ;
    skos:definition "Anything spatial (being or having a shape, position
        or an extent)."@en ;
    skos:note "Subclasses of this class are expected to be used for
        instance data."@en ;
    skos:prefLabel "Spatial Object"@en .

gn:Feature a owl:Class ;
    rdfs:label "Geonames Feature"@en ;
    rdfs:comment "A geographical feature"@en ;
    rdfs:subClassOf geo:SpatialObject ;
    skos:historyNote "Modified in version 3.1 : The cardinality
        constraints on feature code, feature class are relaxed on this
        class, and put on the specific subclass Geonames Feature. This
        class and attached properties can therefore be used for features
        not necessarily identified in the Geoanmes data base"@en .
```

Code 2: Full geonuts ontology version 1. Used for creating the fine-tuned model and for one third of the experiment questions. The number of triples in this ontology is 164.

Populated Question	Question Category	Non-RAG Answer	Detected Answers	Ground Truth Answers	F1
Is the NUTS region CZ06 northwest of the city of Bayreuth?	directions bool	true	True	False	0.00
What NUTS regions are within the region RO321?	simple topology	RO321	RO321	RO321	1.00
What are all cities with more than a 120 k people and are less than 450 km from the NUTS region AT3?	proximity	Munich, Nuremberg, Linz, Salzburg, Graz, Innsbruck, Klagenfurt, Ljubljana, Zagreb, Brno, Bratislava, Venice, Verona, Padua, Trieste, Bologna, Milan, Stuttgart, Zurich, Bern, Geneva, Lausanne, Basel, Freiburg im Breisgau, Karlsruhe, Mannheim, Heidelberg, Frankfurt am Main, Wiesbaden, Mainz, Darmstadt, Ludwigshafen am Rhein, Strasbourg, Mulhouse, Freiburg im Breisgau, Metz, Nancy, Luxembourg City.	'trieste', 'heidelberg', 'ljubljana', 'zagreb', 'verona', 'klagenfurtamwörthersee', 'bern', 'freiburg', 'wiesbaden', 'munich', 'zürich', 'padova', 'stuttgart', 'mulhouse', 'mannheim', 'nürnberg', 'genève', 'basel', 'strasbourg', 'darmstadt', 'venice', 'milan', 'graz', 'brno', 'mainz', 'nancy', 'ludwigshafenamrhein', 'linz', 'bologna', 'lausanne', 'bratislava', 'salzburg', 'metz', 'innsbruck', 'frankfurtammain', 'karlsruhe'	'Prato', 'Aachen', 'Dijon', 'Grenoble', 'Dresden', 'Katowice', 'Zenica', 'Zagreb', 'Freiburg', 'Berlin', 'Nice', 'Wuppertal', 'Monza', 'Karlsruhe', 'Lausanne', 'Split', 'Frankfurt am Main', 'Halle (Saale)', 'Opole', 'Augsburg', 'Ingolstadt', 'Prague', 'Vienna', 'Strasbourg', 'Besançon', 'Zabrze', 'Perugia', 'Solingen', 'Basel', 'Köln', 'Bergamo', 'Darmstadt', 'Brno', 'Lyon', 'Verona', 'Göttingen', 'Mestre', 'Stuttgart', 'Magdeburg', 'Florence', 'Livorno', 'Milan', 'Heidelberg', 'Pilsen', 'Brescia', 'Potsdam', 'Graz', 'Zürich', 'Salzburg', 'Wałbrzych', 'Saarbrücken', 'Turin', 'Kassel', 'Nürnberg', 'Rybnik', 'Szeged', 'Chemnitz', "Reggio nell'Emilia", 'Bern', 'Wiesbaden', 'Leverkusen', 'Ostrava', 'Parma', 'Bielsko-Biala', 'Ulm', 'Linz', 'Mannheim', 'Ruda Śląska', 'Kraków', 'Sosnowiec', 'Padova', 'Munich', 'Częstochowa', 'Banja Luka', 'Bonn', 'Ljubljana', 'Pécs', 'Ferrara', 'Mainz', 'Miskolc', 'Metz', 'Budapest', 'Saint-Étienne', 'Rimini', 'Tychy', 'Ludwigshafen am Rhein', 'Heilbronn', 'Genève', 'Villeurbanne', 'Bytom', 'Győr', 'Bologna', 'Trieste', 'Genoa', 'Regensburg', 'Innsbruck', 'Gliwice', 'Leipzig', 'Würzburg', 'Modena', 'Bratislava', 'Wrocław', 'Erfurt', 'Trento'	0.46

Table 13: Three examples of questions and their non-RAG answers (model = gpt-4o, temperature = 0). The values in the column "detected answers" are cleaned to allow easier matching with the "ground truth answers" during the accuracy assessment.

Populated Question	Question Category	Retrieved Answers	Ground Truth Answers	F1
Is the NUTS region CZ06 northwest of the city of Bayreuth?	directions bool	False	False	1.00
What NUTS regions are within the region RO321?	simple topology	RO321	RO321	1.00
What are all cities with more than a 120 k people and are less than 450 km from the NUTS region AT3?	proximity	'Győr', 'Padova', 'Lausanne', 'Monza', 'Innsbruck', 'Ulm', 'Nürnberg', 'Bern', 'Miskolc', 'Livorno', 'Verona', 'Parma', 'Zagreb', 'Opole', 'Bonn', 'Genoa', 'Wrocław', 'Modena', 'Milan', 'Salzburg', 'Dresden', 'Ruda Śląska', 'Ingolstadt', "Reggio nell'Emilia", 'Darmstadt', 'Florence', 'Sosnowiec', 'Grenoble', 'Karlsruhe', 'Wuppertal', 'Szeged', 'Rybnik', 'Heilbronn', 'Würzburg', 'Kraków', 'Bologna', 'Munich', 'Aachen', 'Ostrava', 'Zabrze', 'Pilsen', 'Chemnitz', 'Kassel', 'Solingen', 'Ludwigshafen am Rhein', 'Saint-Étienne', 'Göttingen', 'Bratislava', 'Split', 'Katowice', 'Zenica', 'Augsburg', 'Magdeburg', 'Trento', 'Wałbrzych', 'Turin', 'Frankfurt am Main', 'Tychy', 'Budapest', 'Ferrara', 'Potsdam', 'Wiesbaden', 'Mannheim', 'Villeurbanne', 'Banja Luka', 'Berlin', 'Besançon', 'Mainz', 'Erfurt', 'Basel', 'Vienna', 'Trieste', 'Freiburg', 'Halle (Saale)', 'Lyon', 'Regensburg', 'Prague', 'Stuttgart', 'Mestre', 'Bergamo', 'Metz', 'Bielsko-Biala', 'Prato', 'Graz', 'Strasbourg', 'Heidelberg', 'Leverkusen', 'Köln', 'Dijon', 'Rimini', 'Pécs', 'Genève', 'Ljubljana', 'Perugia', 'Brescia', 'Zürich', 'Gliwice', 'Częstochowa', 'Saarbrücken', 'Brno', 'Linz', 'Nice', 'Leipzig', 'Bytom'	'Győr', 'Padova', 'Lausanne', 'Monza', 'Innsbruck', 'Ulm', 'Nürnberg', 'Bern', 'Miskolc', 'Livorno', 'Verona', 'Parma', 'Zagreb', 'Opole', 'Bonn', 'Genoa', 'Wrocław', 'Modena', 'Milan', 'Salzburg', 'Dresden', 'Ruda Śląska', 'Ingolstadt', "Reggio nell'Emilia", 'Darmstadt', 'Florence', 'Sosnowiec', 'Grenoble', 'Karlsruhe', 'Wuppertal', 'Szeged', 'Rybnik', 'Heilbronn', 'Würzburg', 'Kraków', 'Bologna', 'Munich', 'Aachen', 'Ostrava', 'Zabrze', 'Pilsen', 'Chemnitz', 'Kassel', 'Solingen', 'Ludwigshafen am Rhein', 'Saint-Étienne', 'Göttingen', 'Bratislava', 'Split', 'Katowice', 'Zenica', 'Augsburg', 'Magdeburg', 'Trento', 'Wałbrzych', 'Turin', 'Frankfurt am Main', 'Tychy', 'Budapest', 'Ferrara', 'Potsdam', 'Wiesbaden', 'Mannheim', 'Villeurbanne', 'Banja Luka', 'Berlin', 'Besançon', 'Mainz', 'Erfurt', 'Basel', 'Vienna', 'Trieste', 'Freiburg', 'Halle (Saale)', 'Lyon', 'Regensburg', 'Prague', 'Stuttgart', 'Mestre', 'Bergamo', 'Metz', 'Bielsko-Biala', 'Prato', 'Graz', 'Strasbourg', 'Heidelberg', 'Leverkusen', 'Köln', 'Dijon', 'Rimini', 'Pécs', 'Genève', 'Ljubljana', 'Perugia', 'Brescia', 'Zürich', 'Gliwice', 'Częstochowa', 'Saarbrücken', 'Brno', 'Linz', 'Nice', 'Leipzig', 'Bytom'	1.00

Table 14: Three examples of questions and their RAG answers (model = gpt-4o, temperature = 0)

Raw Question	F1	Precision	Recall	Hallucination Rate
What are all the NUTS regions that contain the city CITY?	0.49	0.72	0.40	0.02
What NUTS regions are within the region CODE?	0.78	0.88	0.75	0.10
Is the NUTS region CODE bordering the region CODE?	0.83	0.83	0.83	-
What regions of the same level are neighbors of the NUTS region CODE?	0.38	0.45	0.39	0.09
What are the second order neighbors of the same NUTS level for the NUTS region CODE?	0.28	0.31	0.35	0.30
Is NUTS region CODE DIRECTION of NUTS region CODE?	0.52	0.52	0.52	-
Is the NUTS region CODE DIRECTION of the city of CITY?	0.52	0.52	0.52	-
To which direction is CITY from CITY?	0.62	0.63	0.62	-
CITY is to which direction of NUTS region CODE?	0.31	0.32	0.30	-
Is CITY within SMALLDISTANCE km of the NUTS region CODE?	0.57	0.57	0.57	-
What are all cities within SMALLDISTANCE km of CITY?	0.24	0.37	0.25	-
What are all cities CCONDITION and are less than BIGDISTANCE km from the NUTS region CODE?	0.27	0.49	0.26	-
What is the largest city that can be found within the bounding box of the NUTS region CODE?	0.15	0.15	0.15	-
What is the largest city within BIGDISTANCE km of the NUTS region CODE?	0.39	0.39	0.39	-
What are cities within SMALLDISTANCE kilometers to the DIRECTION of the NUTS region CODE?	0.00	0.00	0.00	-
Which cities can be found not further than SMALLDISTANCE km to the DIRECTION of CITY?	0.13	0.25	0.10	-
What NUTS regions share a border with the region CODE in the DIRECTION on the same nuts level?	0.15	0.23	0.12	0.02
What is the closest city to the DIRECTION of the NUTS region CODE?	0.00	0.00	0.00	-

Table 15: Results split by actual question for the non-RAG experiment. Parts in capitals are replaced by actual examples (except for NUTS) (model=gpt-4o, temperature < 0.5).

Raw Question	F1	Difference to non-RAG
What are all the NUTS regions that contain the city CITY?	1.00	0.68
What NUTS regions are within the region CODE?	0.91	0.22
Is the NUTS region CODE bordering the region CODE?	0.99	0.25
What regions of the same level are neighbors of the NUTS region CODE?	0.86	0.57
What are the second order neighbors of the same NUTS level for the NUTS region CODE?	0.64	0.40
Is NUTS region CODE DIRECTION of NUTS region CODE?	0.99	0.46
Is the NUTS region CODE DIRECTION of the city of CITY?	0.98	0.43
To which direction is CITY from CITY?	0.39	-0.22
CITY is to which direction of NUTS region CODE?	0.55	0.30
Is CITY within SMALLDISTANCE km of the NUTS region CODE?	0.93	0.41
What are all cities within SMALLDISTANCE km of CITY?	0.66	0.48
What are all cities CCONDITION and are less than BIGDISTANCE km from the NUTS region CODE?	0.92	0.73
What is the largest city that can be found within the bounding box of the NUTS region CODE?	1.00	0.88
What is the largest city within BIGDISTANCE km of the NUTS region CODE?	0.93	0.64
What are cities within SMALLDISTANCE kilometers to the DIRECTION of the NUTS region CODE?	0.85	0.85
Which cities can be found not further than SMALLDISTANCE km to the DIRECTION of CITY?	0.69	0.60
What NUTS regions share a border with the region CODE in the DIRECTION on the same nuts level?	0.62	0.49
What is the closest city to the DIRECTION of the NUTS region CODE?	0.68	0.68

Table 16: Results split by actual question for the RAG experiment. Parts in capitals are replaced by actual examples (except for NUTS) (model=gpt-4o, temperature < 0.5).

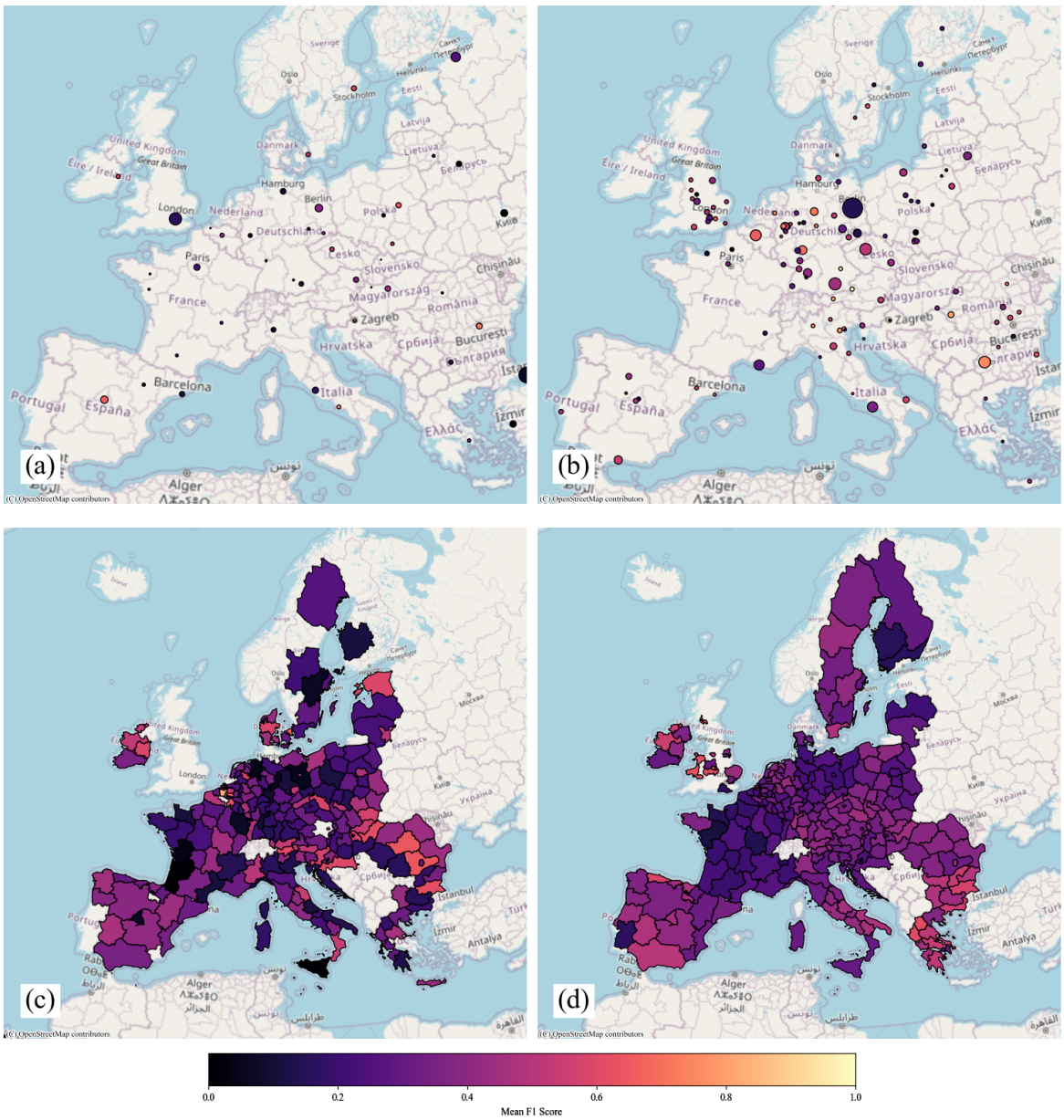


Figure 17: The maps show F1 scores for different cities and NUTS level 2 regions (both n > 6). **(a)** and **(c)** show F1 scores for questions where the respective city or NUTS region was taken as an example in the question. **(b)** and **(d)** show the F1 scores for questions where the respective city or NUTS region would have been the or one of the answers.

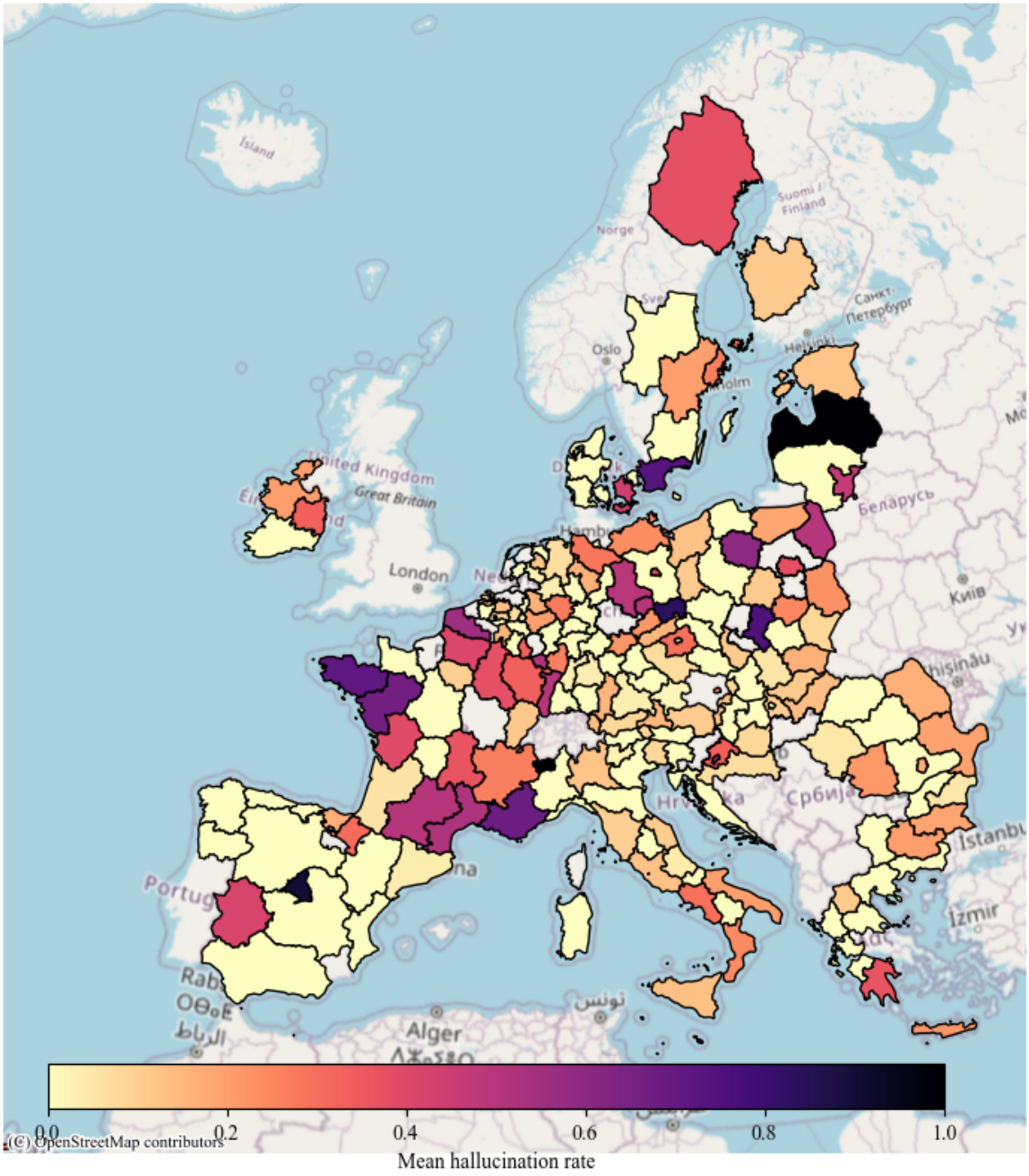


Figure 18: The map shows hallucination rates for questions about each NUTS level 2 region for regions that appeared at least six times as an example.

```
# template_v1, system message with tips:
You are an AI assistant that translates natural language questions about
    geospatial relations into SPARQL queries based on the given Ontology.
     Your answer should only contain the SPARQL query. Keep in mind that
    geofunctions can only be used with a WKT literal that is attached to
    a geometry subject: '?geom geo:asWKT ?wkt'

# template_v1, system message without tips:
You are an AI assistant that translates natural language questions about
    geospatial relations into SPARQL queries based on the given Ontology.
     Your answer should only contain the SPARQL query.

# template_v2, system message with tips:
You are an AI assistant that writes SPARQL queries to a graph database
    based on a user question and the graphs' ontology. The questions are
    about the location and topology of the graphs elements. Your answer
    should only contain the SPARQL query. Keep in mind that geofunctions
    can only be used with a WKT literal that is attached to a geometry
    subject: '?geom geo:asWKT ?wkt'

# template_v1, user message template:
Write a SPARQL SELECT query for querying a graph database.
The ontology schema delimited by triple backticks in Turtle format is:
```
{}
```
Use only the classes and properties provided in the schema to construct
    the SPARQL query.
Do not use any classes or properties that are not explicitly provided in
    the SPARQL query.
Include all necessary prefixes.
Do not include any explanations or apologies in your responses.
Do not wrap the query in backticks.
Do not include any text except the SPARQL query generated.
The question delimited by triple backticks is:
```
{}
```

# template_v2, user message template:
Write a SPARQL query for querying a graph database.
The ontology schema delimited by triple backticks in Turtle format is:
```
{}
```
```

```
```
Use only the classes and properties provided in the schema to construct
    the SPARQL query.
Do not use any classes or properties that are not explicitly provided in
    the SPARQL query.
Include all necessary prefixes.
Do not include any explanations or apologies in your responses.
Do not wrap the query in backticks.
Do not include any text except the SPARQL query generated.
The question delimited by triple backticks is:
```
{}
```
```

**Code 3:** System messages and user message templates for the different template versions. The user message templates are taken for LangChain (2024). The  signal the input of the ontology or the example question.

```

System Message:
You are an AI assistant that writes SPARQL queries to a graph database
    based on a user question and the graphs' ontology.
The questions are about the location and topology of the graphs' elements
    .
Your answer should only contain the SPARQL query.

The ontology schema delimited by triple backticks in Turtle format is:
```
{}
```
Use only the classes and properties provided in the schema to construct
    the SPARQL query.
Do not use any classes or properties that are not explicitly provided in
    the SPARQL query.
Include all necessary prefixes.

User Message:
Write a SPARQL query for querying a graph database.
Do not include any explanations or apologies in your responses.
Do not wrap the query in backticks.
Do not include any text except the SPARQL query generated.
The question delimited by triple backticks is:
```
{}
```
```
```

```
# Example 1:
Question: What are the three largest cities within 40 km of the NUTS
    region PL91?

Answer:
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX geof: <http://www.opengis.net/def/function/geosparql/>
PREFIX gn: <https://www.geonames.org/ontology#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>

SELECT ?cityName
WHERE {
   ?city a gn:Feature ;
       gn:population ?population ;
       gn:name ?cityName ;
       geo:hasGeometry ?cityGeom .
   ?region a skos:Concept ;
       skos:notation "PL91" ;
       geo:hasGeometry ?regionGeom .
   ?cityGeom geo:asWKT ?cityWKT .
   ?regionGeom geo:asWKT ?regionWKT .

   BIND(geof:buffer(?regionWKT, 40000) as ?buffer)
       FILTER(geof:sfWithin(?cityWKT, ?buffer))
}
ORDER BY DESC(?population)
LIMIT 3

# Example 2:
Question: Is the NUTS region DED43 northwest of the NUTS region CZ031?

Answer:
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX geof: <http://www.opengis.net/def/function/geosparql/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>

SELECT ?isNorthwestOf WHERE {
  ?regionA skos:notation "DED43" .
  ?regionB skos:notation "CZ031" .
  ?regionA geo:hasGeometry ?geometryA .
  ?regionB geo:hasGeometry ?geometryB .
  ?geometryA geo:asWKT ?wktA .
  ?geometryB geo:asWKT ?wktB .
    BIND(IF(geof:isNorthOf(?wktA, ?wktB) && geof:isWestOf(?wktA, ?wktB), "
```

```
       True"^^xsd:boolean, "False"^^xsd:boolean) AS ?isNorthwestOf)
}
```

Code 4: System message and user message template for the few-shot experiment. Here the ontology is included in the system message. The signal the input of the ontology or the example question.

```
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX geof: <http://www.opengis.net/def/function/geosparql/>
PREFIX gn: <https://www.geonames.org/ontology#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>

SELECT ?cityName
WHERE {
   ?city a gn:Feature ;
       gn:population ?population ;
       gn:name ?cityName ;
       geo:hasGeometry ?cityGeom .
   FILTER(?population > 120000)

   ?region a skos:Concept ;
       skos:notation "AT3" ;
       geo:hasGeometry ?regionGeom .
   ?cityGeom geo:asWKT ?cityWKT .
   ?regionGeom geo:asWKT ?regionWKT .

   FILTER(geof:distance(?cityWKT, ?regionWKT) < 450000)
}
```

Code 5: Generated query for question *"What are all cities with more than a 120 k people and are less than 450 km from the NUTS region AT3?*"

```
PREFIX gn: <https://www.geonames.org/ontology#>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX geof: <http://www.opengis.net/def/function/geosparql/>

SELECT ?city ?cityName ?population
WHERE {
  ?city a gn:Feature ;
        gn:population ?population ;
        gn:name ?cityName ;
        geo:hasGeometry ?cityGeom .
  ?region a skos:Concept ;
          skos:notation "AT3" ;
          geo:hasGeometry ?regionGeom .
  ?cityGeom geo:asWKT ?cityWKT .
  ?regionGeom geo:asWKT ?regionWKT .
  FILTER(?population > 120000)
  FILTER(geof:distance(?cityWKT, ?regionWKT) < 450000)
}
```

Code 6: Ground truth query for question *"What are all cities with more than a 120 k people and are less than 450 km from the NUTS region AT3?"*

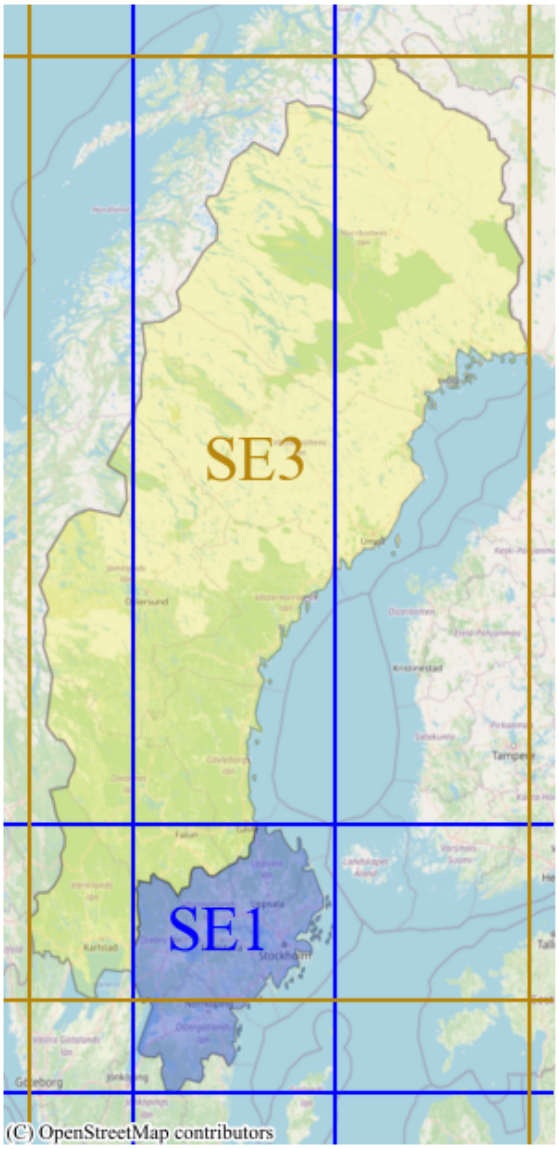


Figure 19: The NUTS regions SE3 and SE1. Region SE1 could be considered to the southeast of region SE3, however SE3 extends further to the east on the more northern parts, leaving SE1 only to the south of SE3.