NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCES
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

MSc THESIS

Geospatial Question Answering with GeoQA3

Sergios - Anestis A. Kefalidis

Supervisor: **Manolis Koubarakis,** Professor

ATHENS

MARCH 2024

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΔΙΠΛΩΜΑΤΙΚΉ ΕΡΓΑΣΙΑ

Απάντηση γεοχωρικών ερωτήσεων με το GeoQA3

Σέργιος - Ανέστης Α. Κεφαλίδης

Επιβλέπων: **Μανόλης Κουμπαράκης,** Καθηγητής

ΑΘΗΝΑ

ΜΑΡΤΙΟΣ 2024

MSc THESIS

Geospatial Question Answering with GeoQA3

Sergios - Anestis A. Kefalidis
S.N.: 7115112200036

SUPERVISOR: **Manolis Koubarakis,** Professor

EXAMINATION COMMITTEE: **Manolis Koubarakis**, Professor NKUA
Alex Delis, Professor NKUA
Alexandros Ntoulas, Assistant Professor NKUA

Examination Date: 15 March, 2024

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Απάντηση γεοχωρικών ερωτήσεων με το GeoQA3

Σέργιος - Ανέστης Α. Κεφαλίδης
Α.Μ.: 7115112200036

ΕΠΙΒΛΕΠΩΝ: **Μανόλης Κουμπαράκης,** Καθηγητής

ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ: **Μανόλης Κουμπαράκης**, Καθηγητής ΕΚΠΑ
Αλέξης Δελής, Καθηγητής ΕΚΠΑ
Αλέξανδρος Ντούλας, Επίκουρος Καθηγητής ΕΚΠΑ

Ημερομηνία Εξέτασης: 15 Μάρτιος 2024

ABSTRACT

Question answering (QA) is a computer science discipline within the fields of information retrieval and natural language processing that is concerned with building systems that answer questions posed in natural language. Question answering is not just a scientific challenge, question answering techniques have seen widespread use and adoption in the industry. They are used in all well-known search engines (e.g., Google and Bing) and digital assistants (e.g., Siri, Alexa, and Google Assistant) for answering questions like "What's the time in New York?" or "How many children did Albert Einstein have?".

Today, users often are interested in posing questions or information requests with a *geospatial dimension* to search engines, chatbots and virtual personal assistants. However, such technologies, like Google and ChatGPT, still struggle to give immediate and precise answers to complex geographic questions of the form: "Which rivers longer than 10kms cross London and at least one other city?", "How many churches exist in a 2-mile radius of the city center of Austin, Texas?", "Which countries border Greece, have the euro as their currency and their population is greater than the population of Greece".

In this thesis, we deal with the problem of answering such questions over *geospatial knowledge graphs* i.e., knowledge graphs (KGs) which represent knowledge about *geographic features* or simply *features* in the terminology of GIS systems. Geospatial knowledge in KGs is encoded using latitude/longitude pairs representing the center of features (as e.g., in DBpedia and YAGO2), but also more detailed geometries (e.g., lines, polygons) since these are more appropriate for modeling the geometries of features such as rivers, roads, countries etc. (as in Wikidata, YAGO2geo, WorldKG and KnowWhereGraph).

We present the geospatial QA system GeoQA3 which is based on GeoQA and its revision GeoQA2. GeoQA3 represents a radical evolution of the GeoQA family of engines, introducing Large Language Models in the question-answering pipeline to improve natural language understanding and facilitate dynamic query generation. This allows GeoQA3 to understand and correctly answer a larger variety of questions. The engine is available as open source. As its predecessor, it targets the union of the knowledge graph YAGO2 and the geospatial knowledge graph YAGO2geo.

SUBJECT AREA: Geographic Information Systems

KEYWORDS: Geospatial Question Answering, Geospatial Data, Neural Network, Knowledge Graph, SPARQL

ΠΕΡΙΛΗΨΗ

Η απάντηση ερωτημάτων είναι μία περιοχή της επιστήμης των υπολογιστής που ασχολείται με την κατασκευή συστημάτων που απαντούν ερωτήσεις που δίνονται σε φυσική γλώσσα. Δεν αποτελεί απλά ένα τομέα επιστημονικής αναζήτησης, τεχνικές απάντησης ερωτημάτων έχουν χρησιμοποιηθεί ευρεύως από μεγάλες εταιρείες λογισμικού. Χρησιμοποιούνται σε όλες τις γνωστές μηχανές αναζήτησης (π.χ. Google και Bing) και σε ψηφιακούς βοηθούς (π.χ. Siri, Alexa και Goggle Assistant) για να απαντούν σε ερωτήσεις όπως "Τι ώρα είναι στη Νέα Υόρκη;" και "Πόσα παιδιά είχε ο Άλμπερτ Αϊνστάιν;".

Σήμερα, οι χρήτες συχνά θέλουν να κάνουν ερωτήσεις με γεωχωρική χροιά σε μηχανές αναζήτησης, chatbots και ψηφιακούς βοηθούς. Όμως συστήματα όπως το Google Search και το ChatGPT αδυνατούν να δώσουν ακριβής απαντήσεις σε πολύπλοκες γεωχωρικές ερωτήσεις της μορφής: "Ποιά ποτάμια που είναι μεγαλύτερα από 10 χιλιόμετρα παιρνούν από το Λονδίνο και τουλάχιστον μία ακόμα πόλη;", "Πόσες εκκλησίες υπάρχουν σε ακτίνα 2 μιλίων από το κέντρο της πόλης του Όστιν στο Τέξας", "Ποιές γειτονικές χώρες της Ελλάδας χρησιμοποιούν το ευρώ και έχουν πληθυσμό μεγαλύτερο από την Ελλάδα;".

Σε αυτη τη διπλωματική εργασία αντιμετωπίζουμε το πρόβλημα της απάντησης τέτοιων ερωτήσεων που μπορούν να απαντηθούν απο γεωχωρικούς γράφους γνώσεων που αναπαριστούν γεωγραφικά σημεία ενδιαφέροντος. Η γεωχωρική γνώση στους γράφους γνώσεων κωδικοποιείται με τη χρήση συντεταγμένων (όπως σε DBpedia και YAGO2) αλλά και με λεπτομερέστες γεωμετρίες (π.χ. γραμμές, πολύγωνα) που είναι πιο ταιριαστές αναπαραστάσεις για δρόμους, ποτάμια, χώρες και λοιπά (όπως σε Wikidata, YAGO2geo, WorldKG και KnowWhereGraph).

Παρουσιάζουμε το γεωχωρικό σύστημα απάντησης ερωτημάτων GeoQA3 που βασίζεται στα συστήματα GeoQA και GeoQA2. Το GeoQA3 είναι μία ριζική εξέλιξη για τα συστήματα GeoQA, μιας και κάνει χρήση Μεγάλων Γλωσσικών Μοντέλων στην αρχιτεκτονική του για βελτιωμένη κατανόηση της φυσικής γλώσσας και την ικανότητα να παράγει εκτελέσημα ερωτήματα δυναμικά. Αυτό επιτρέπει στο GeoQA3 να κατανοεί και να απαντάει ένα μεγάλυτερο εύρος ερωτήσεων. Το σύστημα είνα προσφέρεται ως λογισμικό ανοιχτού κώδικα. Όπως και το GeoQA2 στοχεύει στην απάντηση ερωτημάτων με τη χρήση των YAGO2 και YAGO2geo.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Γεωγραφικα Πληροφοριακα Συστηματα

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Απάντηση Γεωχωρικών Ερωτημάτων, Γεωχωρικά Δεδομένα, Νευρωνικά Δίκτυα, Γράφος Γνώσης, SPARQL

A mi novia.

CONTENTS

LIST OF FIGURES

PREFACE

This thesis is part of my postgraduate studies in the Department of Informatics and Telecommunications of the University of Athens. Specifically, this is my last remaining obligation before I complete my master's degree.

I want to thank my supervisor, Professor Manolis Koubarakis for giving me the opportunity to work in his research group and helping me make my first steps as an aspiring computer science researcher. His guidance, generosity and jolly attitude have been a source of joy throughout my time in this group.

I want to thank my colleagues, co-researchers and friends at the AI Team of the University of Athens for their invaluable help and camaraderie during this time. Special mention to Kostas, Konstantinos, Despina and Mariangela, whom I cannot imagine the last 18 months without, and for that I am grateful.

Last but not least, I want to thank my family and friends who have always helped and supported me.

1. INTRODUCTION

1.1 Problem description

The number of data sources in private environments, enterprises, and the Web is increasing continuously, increasing the effort of making data accessible. One important means of making data accessible is *question answering (QA)*, which provides a natural language interface for common users to express their information needs [25]. A significant fraction of the available data on the Web is geospatial, and this fraction is growing by at least 20% per year [35].

Today users often are interested in posing questions or information requests with a *geospatial dimension* to search engines, chatbots and virtual personal assistants. However, such technologies, like Google and ChatGPT, still struggle to give immediate and precise answers to complex geographic questions of the form: "Which rivers cross London?", "How many churches exist in a 2-mile radius of the city center of Austin, Texas?", "Which countries border Greece, have the euro as their currency and their population is greater than the population of Greece". Answering such questions or information requests requires access to a source of precise data that includes *geospatial* information as well.

There is plenty of such data available today in the form of *geospatial knowledge graphs (KGs)* (e.g., YAGO2 [26], YAGO2geo [30], WorldKG [13] and KnowWhereGraph [28]). The standard way to retrieve knowledge from geospatial KGs (or RDF stores storing linked geospatial data) is by using the query language SPARQL and its geospatial extensions GeoSPARQL and stSPARQL. However, to better serve the needs of non-technical end users, a system that will enable posing geospatial questions in natural language is needed. Although the developments in *question answering (QA)* over structured or unstructured data have been significant during the last few years, the geospatial dimension introduces additional challenges [40]: i) the QA system has to automatically identify the spatial representation (e.g., point, polygon) of the entities to choose, depending on the context of the question; ii) the interpretation of spatial operations and relations is subject to the map scale tied to the question (e.g., the word "near" is interpreted differently for the questions "Which countries are near Greece?" and "Which POIs are near Acropolis?"); iii) the problem of spatial relation recognition can be difficult due to the variability of language (north of Greece vs. northern Greece); iv) calculations among the geometries of the entities mentioned in the question may be required, a process that can be computationally very expensive.

The research community has attempted to address the above challenges by developing geospatial question-answering systems like GeoQA2 and the system of [23]. These systems utilize templates and heuristics to facilitate the query generation process. Although such systems excel in handling a subset of similar natural language questions, their understanding of natural language remains significantly constrained. As a result, their ability to generate appropriate queries for input questions is limited, particularly when these questions cannot be readily mapped to predefined templates. Furthermore, in [31] we observed that pipelines of such systems are prone to error propagation, that is when an incorrect output in one component leads to a faulty query or even complete failure. On the other end of the spectrum, fully neural approaches to geospatial question answering present their own set of limitations. Available GeoSPARQL [44] datasets are small in size and limited in scope, which makes the training and fine-tuning of an end-to-end neural question answering engine a very difficult task. Because of this limitation, such systems are unable to learn and reproduce the structure of complex queries and are prone to mak-

ing syntactical mistakes.

1.2 Introducing GeoQA3

Based on the aforementioned problems, we decided to implement a *hybrid* approach that both ensures the generation of well-formed queries, which is characteristic in rule-based approaches, and is able to understand complex natural language, which is inherent in approaches that utilize deep learning techniques. For the rule-based part, we utilize dependency parsing and a set of heuristics for improved semantic parsing. The large language model part, which was implemented by utilizing Llama 2 [58], enabled the understanding of complex language, while also avoiding error propagation in various scenarios.

In this thesis, we present the geospatial QA system GeoQA3 which answers questions over the KG YAGO2geo. GeoQA3 is based on GeoQA, the first geospatial QA engine proposed by [49] and its revision GeoQA2 in [48]. Even though GeoQA3 reuses components of GeoQA2 it is a fundamentally different engine, built from the ground up to generate queries dynamically, thus removing the need for handwritten templates. GeoQA3 is available as open source[1].

1.3 Thesis Layout

The rest of this thesis is organized as follows. In Chapter 2 we present the basic concepts that geospatial question answering is built on. In Chapter 3 we present an overview of work related to ours. In the following two Chapters, 4 and 5, we present two existing geospatial question answering engines, GeoQA2 [50] and the engine of [23], and how they function. In Chapter 6 we present our engine, GeoQA3. In the final Chapter 7.1 we evaluate the three engines on the dataset GeoQuestions1089 [31].

[1] `https://github.com/AI-team-UoA/GeoQA`

2. BASIC CONCEPTS IN GEOSPATIAL QUESTION ANSWERING

For a better understanding of the rest of this thesis, in this chapter we introduce all necessary notions related to geospatial QA over KGs.

2.1 Knowledge Graphs

A *knowledge graph* is a directed graph $KG = (V, E)$ consisting of a set of vertices V that represent entities, types, and literals, and a set of labeled edges E that connect these vertices [63, 27]. The RDF framework [64] standardized by the W3C is the data model for knowledge graphs. RDF is based on the notion of a triple SPO where S is the subject, P is the predicate, and O is the object. For a knowledge graph KG represented in RDF, we have $S, O \in V$ and $P \in E$. SPARQL[1] is the structured query language for querying RDF. GeoSPARQL [44], standardized by OGC, defines a vocabulary for representing geospatial data in RDF and it is an extension of SPARQL for processing geospatial data. stRDF and stSPARQL [32], proposed independently and around the same time as GeoSPARQL, are also extensions of RDF and SPARQL, respectively, for representing and querying spatial data, but they also support geospatial aggregate functions and offer constructs for representing and processing temporal data.

2.2 Geographic Features

Geospatial or *geographic knowledge* has been studied for many years by researchers in Geography, Geographic Information Systems (GIS), Geographic Information Retrieval (GIR), Databases, Artificial Intelligence and the Semantic Web, and there is a wealth of research results concerning representation, querying and inference for geographic knowledge. In GIS terminology which we use in this paper, a *geographic feature* (or simply *feature*) is an abstraction of a real-world phenomenon and can have various attributes that describe its *thematic* and *spatial* characteristics. For example, the country Greece is a feature, its name and population are thematic attributes, while its location on Earth in terms of polar coordinates is a spatial attribute. Knowledge about the spatial attributes of a feature can be *quantitative* or *qualitative*. For example, the fact that the distance between Athens and Salonika is 502 km is quantitative knowledge, while the fact that river Evros crosses Bulgaria and Turkey is qualitative knowledge.

Qualitative geographic knowledge can be captured by *qualitative binary relations* between the geometries of features. Qualitative geospatial data can be expressed as a property of an entity or an explicit assertion. For example, in the YAGO2geo KG, the resource `yago:Berlin` has the object property `geo:sfWithin` with value `yago:Germany`, hence the question “Is Berlin in Germany?” can be easily answered. *Quantitative geographic knowledge* is, usually, represented using *geometries* (e.g., points, lines and polygons on the Cartesian plane). Hence, in cases where the qualitative geographic knowledge cannot be logically entailed by the KG, polygon-based spatial operations can be performed. For instance, supposing that it is not explicitly included in YAGO2geo that Berlin is located in Germany, it suffices to check if the polygon representing Berlin is within the polygon representing Germany to answer the aforementioned question.

[1] `https://www.w3.org/TR/rdf-sparql-query/`

2.3 Geospatial Questions

A *geospatial question* is a question that requires qualitative or quantitative geographic knowledge to be answered. This definition agrees with the notion of *geographic question*, as defined by [40].

2.4 Summary

In this chapter, we introduced the definitions of knowledge graph, geographic features and geospatial questions. We also explained how qualitative and quantitative geospatial knowledge is stored and used.

3. RELATED WORK

In this chapter, we discuss related work on knowledge graphs, question answering systems, and question answering benchmarks.

3.1 Knowledge Graphs

In this section we survey the most well-known KGs, paying special attention to those that contain a significant amount of geospatial knowledge.

3.1.1 Encyclopedic Knowledge Graphs

Freebase is a collaborative KG built in 2007 [5] by the American company Metaweb. Freebase was bought by Google in 2010 and was used as a basis for the Google KG. In 2014, Freebase was shut down by Google, and its data were moved to Wikidata [56].

Wikidata[1] is a knowledge graph which was created in 2012 [62]. Wikidata is a free, collaborative, multilingual database, that collects structured data to provide support for Wikipedia, Wikimedia Commons, and the other wikis of the Wikimedia movement. It can be read and edited by both humans and machines and contains geospatial information such as point coordinates, polygons, countries, and cities.

DBpedia[2] is another well-known KG that was proposed in 2007 [3] to extract and interconnect structured information from Wikipedia infoboxes. DBpedia organizes this information into RDF triples. The DBpedia 2014 release consists of 3 billion RDF triples, out of which 580 million were extracted from the English edition of Wikipedia and 2.46 billion were extracted from other language editions. The DBpedia knowledge graph has several advantages over other KGs: it covers many domains; it represents real community agreement; it automatically evolves as Wikipedia changes; and it is multilingual.

YAGO [3] is KG proposed in 2007 [53]. It contains more than 2 million entities (like persons, organizations, cities, etc.) and 20 million facts about these entities. YAGO includes data extracted from the categories and infoboxes of Wikipedia, combined with the taxonomy of WordNet. YAGO was manually evaluated and found to have an accuracy of 95% with respect to the extraction source. YAGO classifies each entity into a taxonomy of classes. Every entity is an instance of one or multiple classes. Every class (except the root class) is a subclass of one or multiple classes which yields a hierarchy of classes — the taxonomy.

In YAGO2 [26], the second version of YAGO, geospatial and temporal information is introduced in the KG. Geospatial information in YAGO2 comes from two sources, namely Wikipedia and GeoNames[4]. GeoNames is a gazetteer[5], whose data and accuracy have been studied in [1, 2, 20]. Geospatial information in YAGO2 is represented with the properties `yago2:hasLongitude` and `yago2:hasLatitude` that provide the longitude and latitude of the center of a *geoentity* (i.e., an entity that represents a geographical feature). To simplify the representation and querying of geospatial and temporal knowledge, YAGO2 introduced the *SPOTL* data model where *S* stands for subject, *P* for predicate, *O* for ob-

[1] `https://www.wikidata.org`

[2] `https://www.dbpedia.org/`

[3] `https://yago-knowledge.org/`

[4] `https://www.geonames.org/`

[5] A gazetteer is a geographical dictionary that is used, in most cases, together with a map. Given a name (i.e., a city or a river) a gazetteer gives geospatial information about that name.

ject, *T* for time and *L* for location. The SPOTL model allows for facts of the knowledge graph to be associated with temporal and geospatial information. For example, the fact that Barack Obama was inaugurated as president of the USA can be associated with a place (Washington D.C.) and a date (2009-01-20). However, since YAGO2 relies on GeoNames, it contains only points, so it is not an appropriate KG for many applications e.g., the exact geometry of a country cannot be represented and similarly the geometry of other features such as rivers, lakes etc.

YAGO3 [39] extends YAGO2 with multilingual information. YAGO3 has been constructed from 10 different Wikipedia versions (English, German, French, Dutch, Italian, Spanish, Polish, Romanian, Persian, and Arabic) resulting in having more than 17 million entities (like persons, organizations, cities, etc.) and containing more than 150 million facts about these. YAGO3 combines the clean taxonomy of WordNet with the richness of the Wikipedia category system, assigning the entities to more than 350,000 classes. It is also anchored in time and space since it is an extension of YAGO2.

Another large KG is YAGO4 [57]. Although previous versions of the YAGO KG like YAGO2 and YAGO3 had substantial improvements in scope and size, their focus on Wikipedia infoboxes meant that YAGO did not have the same scale as Freebase or Wikidata. Therefore YAGO4 was created, based on Wikidata, the largest public KG. YAGO4 refines the data of Wikidata by making all entity and property identifiers human-readable. Also, the top-level classes and properties come from `schema.org`, a standard list of classes and properties maintained by a community, combined with `bioschemas.org`. The lower-level classes are a selection of Wikidata classes. YAGO4 also contains semantic constraints in the form of SHACL, a WWW Consortium standard language for enhancing the semantic and technical interoperability of RDF ontologies. These constraints keep the data clean and consistent and allow for logical reasoning.

3.1.2 Geospatial Knowledge Graphs

Only four geospatial KGs are currently available in the literature, namely YAGO2 presented above [26], YAGO2geo [30], WorldKG [13] and KnowWhereGraph [28].

YAGO2geo [30] extended YAGO2 with more complex geometries, namely lines, polygons, and multi-polygons. To minimize the geometric uncertainty [40], YAGO2geo is constructed with precise geographical administrative data provided by official sources (from Greece, the United Kingdom and the Republic of Ireland) and the Global Administrative Areas dataset. Additionally, for naturaly formed geofeatures, such as lakes and streams, the respective YAGO2 geo-entities are further enriched with information from OSM. Later on, data of administrative divisions of the United States of America from NBD[6] was added by [19]. YAGO2geo currently contains 640 thousand polygons and 137 thousand lines and, hence, it can be used to answer questions for which precise geospatial information is required.

WorldKG [13] is a more recent effort to create a geospatial knowledge graph using data from OpenStreetMap (OSM). The first release of WorldKG contains over 100 million geographic entities from 188 countries and over 800 million triples. Overall, the number of geographic entities in WorldKG is two orders of magnitude higher than in Wikidata and DBpedia knowledge graphs. WorldKG models data using the WorldKG ontology which has been constructed by converting the flat OSM schema into a hierarchy. The WorldKG ontology is aligned with the Wikidata and DBpedia ontologies enabling the support of ap-

[6] `https://www.usgs.gov/`

plications utilizing data from all these knowledge graphs.

The most recent geospatial knowledge graph is the KnowWhereGraph [28], a visionary approach to integrating geospatial data silos to offer information e.g., for environmental intelligence. KnowWhereGraph integrates information about extreme events, administrative boundaries, soils, crops, climate, transportation etc. It departs from the representation of geospatial information using geometries like the above knowledge graphs do, and uses instead a discrete global grid called the "S2 Grid System." KnowWhereGraph is currently used in three applications: humanitarian relief, food safety and land valuation.

3.2 Question Answering Benchmarks

In this section, we present datasets that can be used to evaluate question answering engines. In addition, these datasets can be used to train deep learning models that can then be part of QA engines.

3.2.1 Encyclopedic Question Answering Datasets

The datasets WebQuestions [4] (6K questions), SimpleQuestions [6] (100K questions), both targeting Freebase[7], were the first considerably large datasets that appeared in the literature. WebQuestions was created in a forward manner: 100K questions were randomly selected by using the Google Suggest API and, then, by manually keeping the ones that could be answered by Freebase. SimpleQuestions, on the other hand, was created in a backward manner: a set of facts from Freebase were shortlisted and, then, manually annotated with relevant questions by English speakers. In terms of structural complexity, both datasets were simple, containing only factoid questions i.e., questions with a unique answer that can be derived from a single fact (triple) in Freebase. In 2016, WebQuestionsSP [66] (5K questions) was generated from WebQuestions, by providing SPARQL queries for the questions that the annotators could fully process to find the answers (SP stands for Semantic Parsing). Then, WebQuestionsSP was used to generate the benchmark ComplexWebQuestions [54] (35K questions) by sampling question-query pairs and automatically creating more complex SPARQL queries. From these queries, a set of questions was generated automatically by using 687 templates, and, then, the resulting questions were manually reformulated. ComplexWebQuestions contains composition questions, superlatives, and comparatives.

Three other significant benchmarks that contain complex questions are the LC-QuAD [59], LC-QuAD 2.0 [60] and QALD-9 [61] datasets. The LC-QuAD dataset (5K questions) targets DBpedia. Similarly to SimpleQuestions, it was created in a backward manner: the queries were generated semi-automatically by extracting sub-graphs containing triples within a 2-hop distance from a seed entity. The generation of the questions was facilitated automatically, using templates, and, then, refined manually. LC-QuAD was later extended to form LC-QuAD 2.0 (30K questions), which contains questions, their paraphrases, and their corresponding SPARQL queries. QALD-9 was generated manually as part of the latest QALD challenge[8]. It targets DBpedia 2016-10 and contains 558 manually created questions with counts, superlatives, comparatives, and temporal aggregators. The questions are available in 11 different languages and each question is annotated with a manually specified SPARQL query and its output.

[7] SimpleQuestions were later reformulated to target also Wikidata [12].

[8] `http://qald.aksw.org/`

3.2.2 Geospatial Question Answering Datasets

All aforementioned datasets contain encyclopedic questions, but some of them also contain geospatial questions. However, as the datasets are too large, NLP techniques are required to extract them from the full dataset, and, then, check manually that the extracted questions are indeed geospatial (for instance, the question "when was Washington elected" may, falsely, be identified as a geospatial question as Washington is both a state and a person), which is a time-consuming process. A dataset marginally relevant to the geospatial domain is POIReviewQA[9] (20K questions), which contains questions about POIs. It was created by retrieving questions from the "Ask the community" service of Yelp business pages[10] and, then, by manually filtering the ones for which answers are identified in the respective reviews. The dataset contains only pairs of questions with their answers.

Currently, the only datasets focusing on the geospatial domain are GeoQuery [55], GeoAnQu [65], GeoQuestions201 [49] and GeoQuestions1089 [31]. GeoQuery [55] contains 880 handcrafted factoid questions about the U.S. Geography in natural language paired with the corresponding queries in a formal query language (Prolog). GeoAnQu is a corpus of 429 geo-analytic complex and non-factoid questions manually extracted from research papers containing GIS analysis, and GIScience textbooks.

GeoQuestions201, created for the evaluation of GeoQA [49], targets the linked geospatial dataset built from DBpedia and the parts of the datasets GADM and OSM restricted to the United Kingdom and the Republic of Ireland. It contains 201 manually crafted factoid, simple and complex questions, with the respective stSPARQL/GeoSPARQL/SPARQL queries and answers. It has been used to evaluate the engines proposed by [21, 23] and by [37].

The benchmark GeoQuestions1089 is the largest question-answering benchmark focusing on the geospatial domain over a specific KG, that contains factoid, simple and complex questions that may contain aggregates and superlatives. In addition to simple questions like those present in GeoQuestions201, GeoQuestions1089 contains semantically complex questions that require a sophisticated understanding of both natural language and GeoSPARQL to be answered. Furthermore, it expands the geographical area of interest, by including questions about the United States and Greece. This expanded list of countries of interest introduces additional challenges that QA engines must overcome. In this thesis GeoQuestions1089 is used to evaluate GeoQA3 and compare it to existing geospatial question answering systems.

3.3 Geospatial Question Answering Engines

Within the research efforts for the development of natural language processing tools in recent years, an increase of question answering systems has been observed in the literature. [11] in a recently released survey presented a set of QA systems over KGs. Two main research directions are presented in this survey: i) translation of natural language questions into the respective SPARQL queries or, ii) learning to embed the question in the same embedding space with the KG and retrieve the answers by calculating the vector similarities among them. Since then, a plethora of research works has been noted for both approaches or their combination (e.g., [9, 68]). The spatial nature of the geospatial domain, where the geospatial KGs do not always contain or can logically entail the required knowledge, but further algebraic calculations among the polygons may need to be performed,

[9] http://stko.geog.ucsb.edu/poireviewqa/

[10] https://blog.yelp.com/news/qa/

led us to the first approach. Hence, in this section we will present the state-of-the-art in the geospatial QA over KGs by transforming the questions to SPARQL/GeoSPARQL queries.

One of the first efforts in this domain is presented by [55]. They present an inductive logic programming approach for learning a semantic parser and applies its techniques to two areas, one of which is querying geospatial databases. They have experimented with a dataset consisting of 1000 Prolog facts from the U.S. Geography domain, and have also developed a corpus of 880 natural language questions and their corresponding logical queries in Prolog.[11] A part of this corpus was used to train the semantic parser developed by the authors.

The work by [67] is closely related to our work since it presents a system for answering geospatial questions over DBpedia. The system is based on a PostGIS[12] database containing precise geospatial information of features in the United Kingdom provided by Ordnance Survey, a spatial index of DBpedia resources built using their point coordinates, and a SPARQL endpoint storing the DBpedia dataset. The three classes of questions considered are proximity "Find churches within 1 km of the River Thames"), crossing (e.g., "Find the mouths of the rivers that cross Oxford") and containment (e.g., "Find churches in Manchester").

[20] explore the use of DBpedia and Geonames for answering topological queries involving administrative divisions of Switzerland and Scotland (since the authors are very familiar with the administrative geographies of these two countries). The paper contains a detailed discussion of quality issues in linked geospatial data and especially the two linked data sources used by the authors (e.g., incompleteness, inconsistency of data etc.). Finally, the paper considers queries for neighbouring and containing/contained administrative divisions, and measures precision and recall when only one of datasets or both linked datasets are used.

[22], initially, analyzed the natural language question-answer dataset MS MARCOV2.1 [45], which contains questions posed to the Bing search engine and human-generated answers. They concentrated on place-related questions of this dataset and defined a set of patterns that can be used to characterize semantically questions and their answers. In their next work [21], they extended this set of patterns to build GeoSPARQL queries from natural language questions. Specifically, they defined patterns utilizing the semantic characteristics of the questions, according to their constituency and dependency parse tree, and based on these patterns and a set of rules, first order statements are produced. Finally, the respective GeoSPARQL queries are generated from these statements.

[8] presented an approach for translating natural language questions to spatial SQLs through a parameterization process. The authors identified different parameters entity name, entity names, order, distance, number and entity type. Five different types of geographic questions are considered and different templates of the SQL queries are predefined for these different question types. The question is annotated with different parameters present in the question and based on the different parameters present in the question it is classified in to one of the five different question type. The differences between the parameters and spatial function needed to answer the question is the key to the classification of the questions. The templates of the SQL queries are generated from filling templates with the different annotated parameters in the question.

[41] present a spatially explicit transnational knowledge graph embedding model called

[11] http://www.cs.utexas.edu/users/ml/nldata/geoquery.html

[12] http://postgis.net/

TransGeo which utilizes an edge-weighted PageRank and sampling strategy to encode the distance decay into the embedding model training process. This embedding model is further applied to relax and rewrite unanswerable geographic questions. The embedding model is evaluated using two tasks: link prediction as well as query relaxation/rewriting for an approximate answer prediction task. The approach is not to generate SPARQL/GeoSPARQL queries from natural language but considers that the SPARQL/GeoSPARQL query is already generated from natural language and it can not produce any answers and thus rewrites the query and based on both the similarity/relatedness among geographic entities (the distance decay effect) and the nature of the question. The evaluation results over their benchmark dataset shows that spatially explicit embedding model outperforms non-spatial models.

The first QA engine over a KG has been a system for answering geospatial questions over DBpedia [67]. The system is based on a PostGIS database containing precise geospatial information of features in the United Kingdom provided by Ordnance Survey, a spatial index of DBpedia resources built using their point coordinates, and a SPARQL endpoint storing the DBpedia dataset. The three classes of questions considered are proximity (e.g., “Find churches within 1 km of the River Thames”), crossing (e.g., “Find the mouths of the rivers that cross Oxford”) and containment (e.g, “Find churches in Manchester”).

The next geospatial QA engine to be proposed was GeoQA [49] and its revised version [48]. GeoQA can answer geospatial questions over DBpedia interlinked with the parts of GADM and OSM for the United Kingdom and Ireland. GeoQA is implemented as a pipeline of six components (dependency parse tree generator, concept identifier, instance identifier, geospatial relation identifier, property identifier and query generator) using a template-based approach.

[37] proposed to use a neural approach for extracting information from input questions and experimented with Long Short-Term Memory (LSTM) networks and transformers for the implementation of the idea. They also used geospatial semantic graphs for representing input questions. To generate the geospatial semantic graph, they first use a semantic dependency parser to map questions to dependency graphs. Then, they combine the parsing results with the neural encoding results to construct the final geospatial semantic graph. Finally, they use a template-based approach like the one proposed by GeoQA for generating the final GeoSPARQL queries. [37] achieves better results than [49] in producing GeoSPARQL translations of input questions. However, they achieve worse results than [48] given that the GeoQuestions201 benchmark is very small to allow for the successful training of deep learning models. In contrast, neural approaches to non-spatial factoid question answering exhibit excellent results because the used deep learning models have been trained on very large question benchmarks [38].

[23] presents a geographic QA engine that uses pre-trained deep learning models for fine-grained named entity recognition and part-of-speech tagging for extracting encodings of nouns, verbs, adjective etc. from the question and, in that way, discovering e.g., whether a noun is a place name or an event name. Then, they defined patterns utilizing the extracted encodings of the questions, according to their constituency and dependency parse tree, and based on these patterns and a set of rules, first-order logic statements were produced. The elements from first-order logic statements are mapped to the resources and classes of YAGO2geo using different techniques that include string matching from prebuilt Solr indexes, exact matching and cosine similarity of BERT embeddings. Predefined templates are used with set of rules to build the structure of the GeoSAPRQL query and mapped elements are replaced to generate thefinal executable GeoSPARQL query over

the YAGO2geo knowledge graph. We are going to present this engine in more detail in section 5.

[50] presents the geospatial question-answering engine GeoQA2, an evolution of GeoQA that targets the YAGO2geo [30] knowledge graph and integrates constituency parsing in the pipeline to handle more complex questions. We will present this engine in more detail in section 4 since it was used as the starting point for our work.

The rise of foundational models like GPT-4 [47] and Llama2 [58] has not gone unnoticed in the geospatial question-answering space. A number of engines that utilize such models for query generation exist [15] and ChatGeoPT[13]. Both of these engines employ a few-shot learning environment to instruct instruction trained LLMs to generate executable queries over a knowledge base. Even though this approach is seemingly both simple to program and able to generate complex queries no evaluation results are provided. GeoQA3 is, to our knowledge, the first engine that combines classical question-answering techniques with LLMs and is compared to existing geospatial question-answering engines.

At this point we have finished our overview of related work on geospatial question-answering engines, knowledge graphs and datasets. We will continue with the presentation of the engines of our engine, as well as the engines of [23] and [50] with which we compare our work.

3.4 Summary

In this chapter we discussed related work about knowledge graphs, question answering engines and datasets in the domain of geospatial question answering. We make two important observations. First, translating natural language questions to executable SPARQL queries is the natural approach followed by a large majority of geospatial question answering engines. Second, there is a lack of large GeoSPARQL datasets that are suitable for training machine learning models on the task of GeoSPARQL query generation. FInally, we introduced the two engines that will be used as benchmarks for our own, GeoQA2 [50] and the engine of [23].

[13] `https://github.com/earth-genome/ChatGeoPT`

4. THE GEOQA2 QUESTION ANSWERING ENGINE

In this chapter, we present the GeoQA2 engine in detail. GeoQA2 was used as the starting point for the development of GeoQA3 so the two systems share multiple components of their pipelines. A detailed study of GeoQA2 will aid in understanding GeoQA3 and its innovations over its predecessor.

4.1 The conceptual model of GeoQA2

Before we present the pipeline that implements GeoQA2, it is important to explain the conceptual model on which it is based. According to GeoQA2, the world consists of *instances* (e.g., Essex) that belong to *concepts* (e.g., county). Instances can have *thematic* (e.g., population) or *geospatial properties* (e.g., geometry). In addition, instances can be related with *geospatial relations* (e.g., overlaps) to other instances.

In GeoQA2, world knowledge is stored in KG YAGO2geo and there is the following correspondence between YAGO2geo constructs and GeoQA2 terminology (see Figure 4.1):

- Instances in GeoQA2 correspond to geoentities in YAGO2geo.
- Concepts in GeoQA2 correspond to classes in YAGO2geo.
- Properties in GeoQA2 correspond to thematic and geospatial properties of geoentities in YAGO2geo.
- Geospatial relations in GeoQA2 correspond to geospatial relations between geoentities in YAGO2geo.

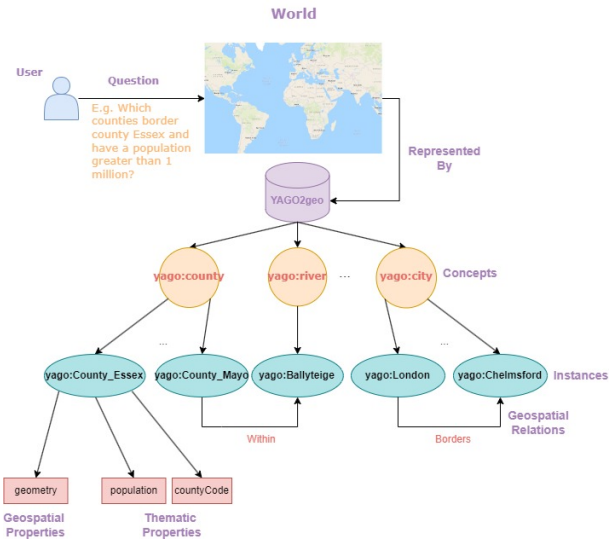

Figure 4.1: The conceptual model of GeoQA2

4.2 The GeoQA2 pipeline

GeoQA2 takes as input a question in natural language (currently only English is supported) and the YAGO2geo KG, and produces one or more answers. Question answering is

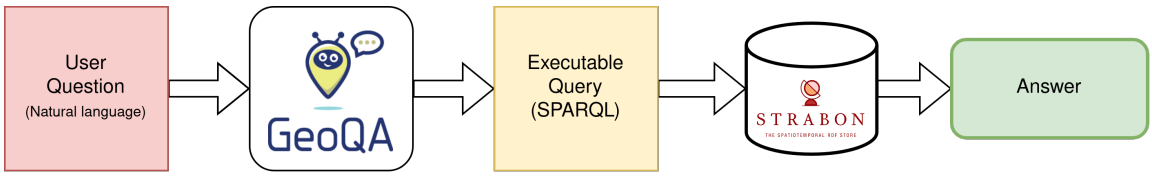

Figure 4.2: The Question Answering process with GeoQA2 and Strabon

performed by translating the input question into a set of SPARQL/GeoSPARQL queries, ranking these queries, and executing the top-ranked query over a YAGO2geo endpoint. An overview of the question-answering process with GeoQA3 is presented in Figure 4.2.

In Figure 4.3 we illustrate the conceptual view of the GeoQA2 pipeline, which contains the following components:

- Dependency and constituency parse tree generator
- Concept identifier
- Instance identifier
- Geospatial relation identifier
- Property identifier
- Query generator

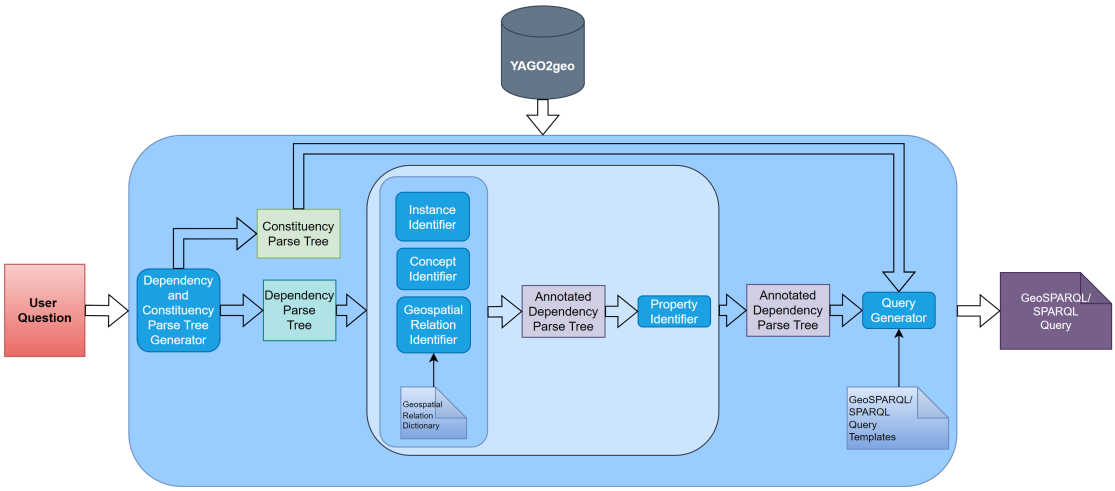

Figure 4.3: The conceptual architecture of GeoQA2

The order in which these components are called in the pipeline is important because some components use the output generated from other components to perform their task. The dependency parse tree generator must be the first in the pipeline as all the other components annotate the respective nodes of the dependency parse tree. The functionality of the concept, instance, and geospatial relation identifiers does not depend on any other component to perform their tasks, thus they can be called in any order in the pipeline. The property identifier uses the outputs from the concept and instance identifiers, thus it must be called only after these two components. The query generator uses the outputs from all

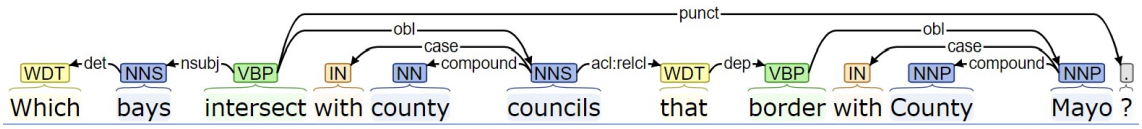

Figure 4.4: The dependency parse tree for the question: "*Which bays intersect with county councils that border with County Mayo?*" as it is generated by CoreNLP.

Table 4.1: Classes from YAGO2geo ontology

Class Label	URI of the class
County Council	`yago2geoo:OSI_County_Council`
Nature Reserve	`yago2geoo:OSM_nature_reserve`
Townland	`yago2geoo:OSI_Townland`
Barony	`yago2geoo:OSI_Barony`
Bay	`yago2geoo:OSM_bay`

the other components to generate queries so it is the last one in the pipeline. Below we present each one of these components in detail.

Dependency parse tree generator. This component carries out part-of-speech tagging and generates a dependency parse tree for the input question using the Stanford CoreNLP toolkit [42]. The dependency parse tree is produced in CoNLL-U format [46]. In Figure 4.4 we provide an example of the dependency parse tree for the question Q: "*Which bays intersect with county councils that border with County Mayo*?" (which will be the question of reference in the rest of this text).

Concept identifier. This component identifies the *types of features (concepts)* present in the input question and maps them to the corresponding classes of the YAGO2geo ontology. These concepts are identified by the elements of the question that are tagged as nouns (NN, NNS, NNP, NNPS) by the dependency parse tree generator. Then, these elements are mapped to the ontology classes using n-grams, as shown in Figure 4.5.

For instance, in the input question Q, it identifies the concepts "county councils" (as well as "county" and "councils") and "bays", as they are tagged as NN, NNS, NNP and maps them to the class `yago2geoo:OSI_County_Council` and `yago2geoo:OSM_bay`, respectively. To simplify the process, we have added labels to the YAGO2geo classes (an excerpt of the YAGO2geo labels with their corresponding classes is presented in Table 4.1)[1]. Hence, the concept identifier iterates through this list of class labels from the ontology, it generates the n-grams (where n is the number of words present in each class label) for Q, and compares the n-grams with the respective class labels. For instance, for the class label "County Council" and the input question Q, the generated 2-grams are: {"*Which bays*", "*bays intersects*", "*intersects with*", "*with county*", "*county councils*", "*councils that*", "*that border*", "*border with*", "*with County*", "*County Mayo*"}. The 2-gram "*county councils*", which has string similarity 0.998 with the class label "county council" (all letters of the class labels are converted to lowercase at the pre-processing phase), is mapped to the class `yago2geoo:OSI_County_Council`. In its final stage, the concept identifier annotates the appropriate node of the dependency parse tree with its results.

[1] The list of prefixes used in the paper can be found at `https://figshare.com/s/584841072d90481e122f`

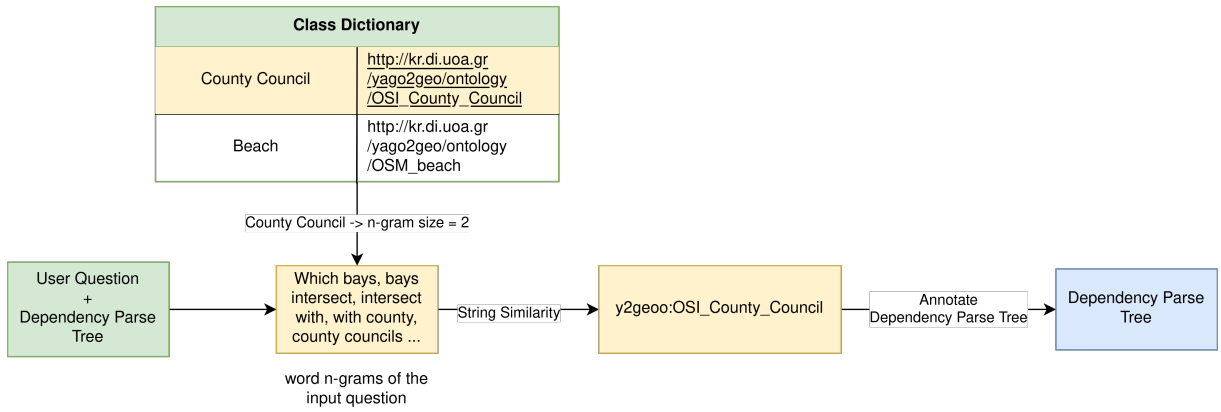

Figure 4.5: GeoQA2 pipeline: Concept identifier

Instance identifier. This component identifies the *features* (*instances*) present in the input question. These can be, for example, the Corfu island or County Mayo. The features are identified by the elements of the question that are tagged as (NN, NNS, NNP) by the dependency parse tree generator. Then, these elements are mapped to YAGO2geo resources using an entity recognition and disambiguation tool, as shown in Figure 4.6.

In previous work [48] we tested a set of well-known tools for named entity recognition and/or disambiguation over GeoQuestions201 [49] and concluded in TagMeDisambiguate [16], which had the best performance. As TagMeDisambiguate links the identified instances with instances only from Wikipedia (hence, from YAGO2 as well [26]), we, also, query YAGO2geo to disambiguate the instances that are contained in YAGO2geo, but not in YAGO2 (e.g., instances in GADM), along with the total number of YAGO2geo triples that contain these instances.

In the running example, once the term “County Mayo” is identified from TagMeDiambiguate, it is mapped to `yago2geor:geoentity_Mayo_3302545`, which is found by executing the SPARQL query `SELECT DINSTICT ?x WHERE{?x yago: hasName "County Mayo"@en}` over the YAGO2geo endpoint.

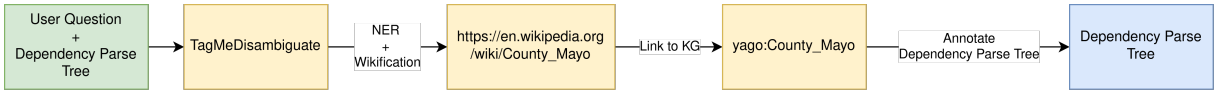

Figure 4.6: GeoQA2 pipeline: Instance identifier

Geospatial relation identifier. Geospatial questions often include some qualitative geospatial relation, such as “borders”, or some quantitative ones, such as “at most 2km”. The current implementation supports the geospatial relations and their synonyms shown on Table 4.2. These include topological, distance, and cardinal direction relations [14, 18, 51].

Similarly to the previous modules, this module first identifies the geospatial relations in the input question, based on the POS tags {VB, IN, VP, VBP, VBZ}, generated by the dependency parse tree. Then, it maps them (or their synonyms) to the respective spatial function of the GeoSPARQL or stSPARQL vocabulary according to Table 4.2. In the running example question, Q, the geospatial relations “intersect” and “border” are identified from their POS tag (VBP) in the dependency tree, and they are mapped to the spatial functions `geof:sfIntersects` and `geof:sfTouches` of the GeoSPARQL vocabulary. In its final stage, the geospatial relation identifier annotates the appropriate node of the dependency parse tree with its results, as shown in Figure 4.7.

The semantics of the topological relations are according to the Dimensionally Extended 9-Intersection Model (DE9IM) [52]. Distance relations, such as “close to” and “near”, are translated into quantitative distance relations based on the concept identified earlier by the concept identifier (e.g., when asking for “hotels near a place”, “near” is taken to mean at most 1 kilometer). The semantics of cardinal direction relations are the usual ones, i.e., a relation A north of B is interpreted by considering the bounding box of the reference region B and the partition of the plane in nine areas that is induced by it [51]. The same semantics are implemented by the Strabon system and its query language stSPARQL, which is used as the back-end geospatial RDF store [33] (GeoSPARQL does not support any cardinal direction functions or relations, therefore stSPARQL is used instead).

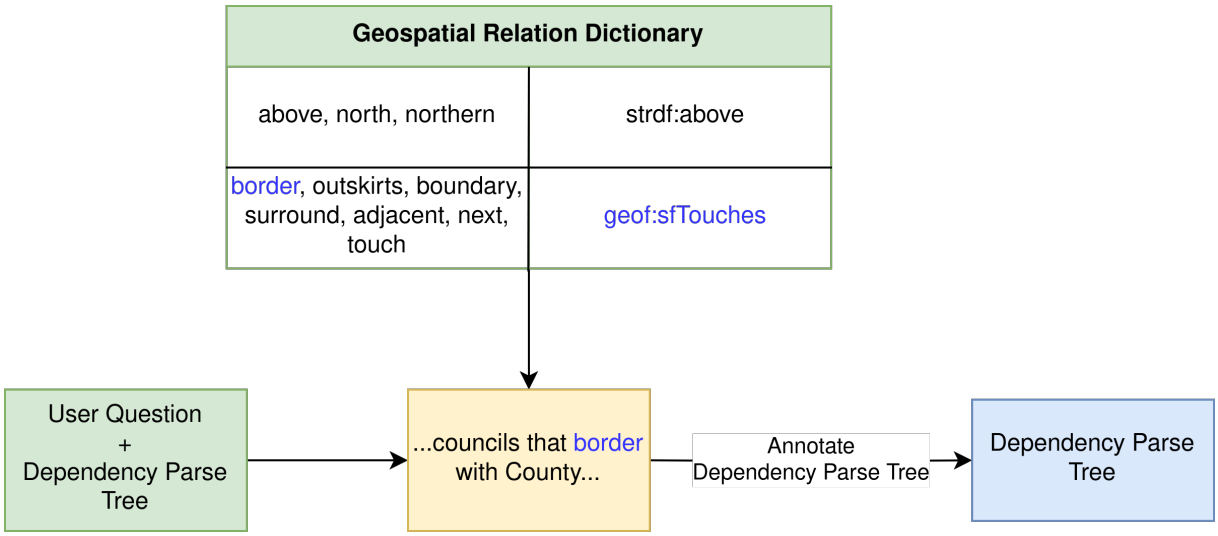

Figure 4.7: GeoQA2 pipeline: Geospatial relation identifier

Property Identifier. The property identifier module identifies *attributes of types of features* and *attributes of features* specified by the user in input questions and maps them to the corresponding properties in YAGO2geo. For instance, for the question “Which village in Rhodes has the biggest population?”, the “population” attribute of the type of feature “village” is required.

This module first identifies the properties in the input question, based on the POS tags {NN, JJ, NNP, NP}, generated by the dependency parse tree. Additionally, from the concepts identified by the concept identifier it performs pattern matching between the filtered terms and the labels of the 1-hop connected relations of the YAGO2geo classes. For instance, for the question “Which village in Rhodes has the biggest population?”, the terms $\mathcal{T} = \{village, Rhodes, population\}$ are filtered. Also, the 1-hop relations connected to the class `yago2geoo:OSM_village`, returned by the concept identifier are selected. Finally, the property identifier performs pattern matching between these relations and the terms in $\mathcal{T}$. This way, the YAGO2geo property `yago2geoo:hasGAG_Population` is retrieved, as shown in Figure 4.8.

For two cases, though, this process is not straightforward: first, when the property is not explicitly mentioned in the question and, second, when the KG does not explicitly contain the implied property. Consider the question “Which is the largest lake in Greece?”. Here, the qualification “by area” is not clearly stated but implied. For such cases, we have defined the rules (not listed here due to space limit) to identify the implied properties. It is to be noted that the list of rules can be extended further without having any impact on the process as required. In particular, the implied property is specified from the classes

Table 4.2: Geospatial relations and their synonyms

Relation Type	Geospatial Relation (With their Synonyms)	Respective Spatial Function
Topological	within : in, inside, is located in, is included in	geof:sfWithin()
	crosses: flows, passes through	geof:sfCrosses()
	borders: at the border of, at the outskirts of, at the boundary of	geof:sfThouches()
	intersects: overlaps	geof:sfIntersects()
Distance	distance: nearby, close to, around	geof:sfDistance()
	near	
	far	
	at most/least x units	
Cardinal Direction	north of: above	strdf:above()
	south of: below	strdf:below()
	east of: right	strdf:right()
	west of: left	strdf:left()
	northeast	strdf:above() && strdf:right()
	northwest	strdf:above() && strdf:left()
	southeast	strdf:below() && strdf:right()
	southwest	strdf:below() && strdf:left()

participating in the question (returned by the concept identifier), the JJS or NN POS tags of the edges of the dependency parse and the implied properties. Hence, in the previous example, to capture the property “area”, after identifying with the concept identifier the class `yago2geoo:OSM_lake`, the property identifier checks if the POS tags of the edges of the dependency parse are annotated as JJS (adjective, superlative, e.g. “biggest”) or NN (noun, singular, e.g. “lake”) and if so, it then checks if any of the keywords {smallest, biggest, largest} appears in the question. If this is the case, according to the table of implied properties, the question is annotated with the “area” property. Supposing, now, that YAGO2geo does not contain any property related to the term “area” for the class `yago2geoo:OSM_lake`, then the areas of the lakes in Greece will be calculated by applying the stSPARQL function `strdf:area()` on their geometries.

Query generator. This module generates the formal query using handcrafted query patterns, templates, and the outputs of the previous modules. In particular, the query generator reformulates the annotated (by the previous components of the pipeline) dependency parse tree and parses it in traversal order. From this process, it identifies the pattern of the question and, then, the respective template. Finally, the GeoSPARQL or SPARQL queries are generated from the templates and the resources identified from the previous modules of the pipeline. If the user question does not match any of the patterns, no query is generated.

We utilized the question patterns defined by [48], which we extended with one more ques-

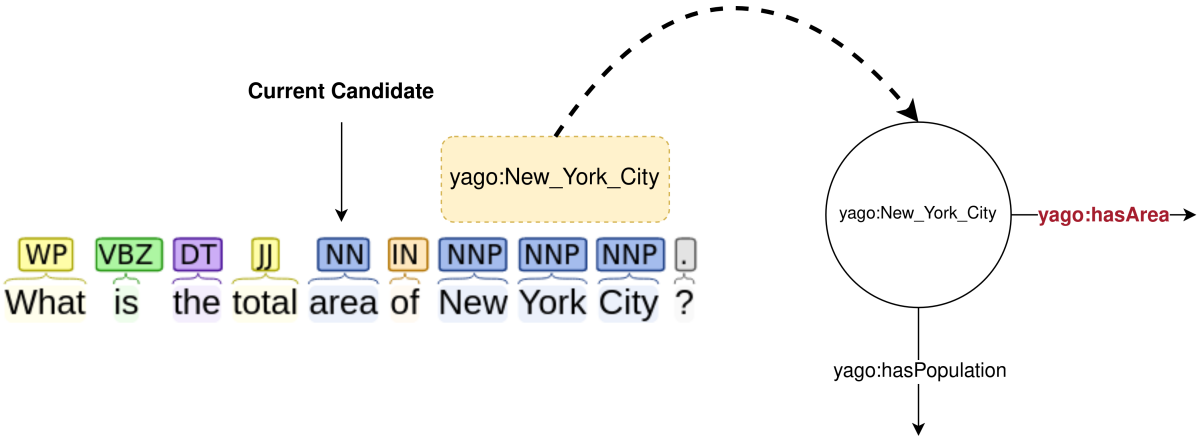

Figure 4.8: GeoQA2 pipeline: Property identifier

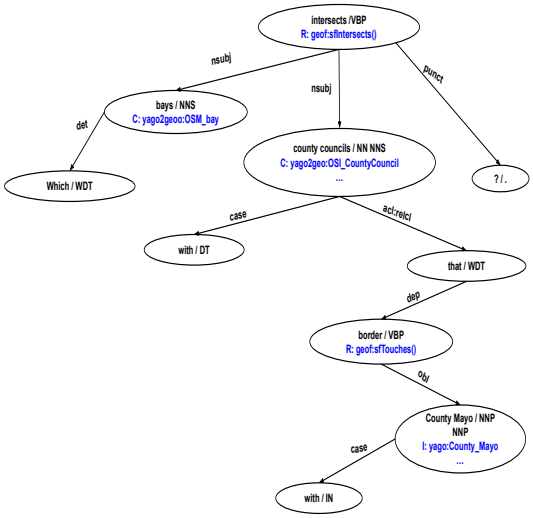

Figure 4.9: Modified dependency parse for the question: "*Which bays intersect with county councils that border with County Mayo?*".

tion pattern (PCRCRI). An incomplete list of question patterns is shown in table 4.3, where "C" stands for "concept", "I" for "instance", "R" for "geospatial relation", "P" for "property" and "N" for "Count of", following the terminology introduced above. Also, in the same tables, the corresponding question type (as defined in Section 7.1), an example question, and the corresponding GeoSPARQL query template is presented for each pattern. Notice that the identification of the intent of the question is implicitly executed through the identification of the proper query template. The query templates contain slots (strings starting with an underscore), which are replaced by the query generator with the outputs of the previous modules.

For instance, the question pattern of the running example question, Q, is CRCRI. The reformulated annotated dependency parse tree is illustrated in Figure 4.9. As it shown, it is annotated with the concepts (C) `yago2geoo:OSM_bay` and `yago2geoo:OSI_County_Council`, the geospatial relations (R) `geof:sfIntersects` and `geof:sfTouches` and the instance (I) `yago2:County_Mayo`. The question pattern is extracted by traversing the tree in order.

To capture more complex questions, containing superlatives, comparatives, or counts,

the query generator produces, also, the constituency parse tree of the input question. Then, the templates are reformulated based on dependency and constituency parse tree, and a number of handcrafted rules. This way, for instance, the query generator will detect that the question “Which Civil Parishes in Ireland have more than 10 townlands?” contains the quantifier phrase (QP) “more than 10”. Subsequently, it will automatically replace the “`SELECT ?x`” with “`SELECT ?x (COUNT(?y) AS ?total)`” and it will add `GROUP BY(?x) HAVING (?total > 10)` at the end of the query generated by the template-based approach.

Lastly, the query generator ranks the generated queries. The ranking system employed is based on the estimated importance of the instances that are used in the generated queries. The importance of an instance is measured as the number of triple patterns that it is a member of, more important instances are likely to have more triple patterns in the knowledge graph. The generated query with the highest total importance is selected. It is to be noted that GeoSPARQL queries are given priority over SPARQL queries in regard to query selection.

4.3 Summary

In this chapter, we presented the complete GeoQA2 pipeline. We examined its pipeline in detail, looking at the functionality of every component. We also discussed the conceptual model of GeoQA2 and how the entire question answering process works.

Table 4.3: A sample of SPARQL/GeoSPARQL templates that are used by GeoQA2.

Pattern	Question Type	Example natural language question	SPARQL/GeoSPARQL Template

IP	A	What is the population of Piraeus?	`SELECT ?property WHERE {`
			`_Instance _Property ?property.`
			`}`

IRI	B	Is Kallithea at the border of Nea Smyrni?	`ASK WHERE {`
			`_Instance1 geo:hasGeometry ?iGeom1.`
			`?iGeom1 geo:asWKT ?iWKT1.`
			`_Instance2 geo:hasGeometry ?iGeom2.`
			`?iGeom2 geo:asWKT ?iWKT2.`
			`FILTER(_Relation(?iWKT1, ?iWKT2))`
			`}`

CRI	C	Which streams cross Limerick?	`SELECT ?x WHERE {`
			`?x rdf:type _Concept; geo:hasGeometry ?xGeom.`
			`?xGeom geo:asWKT ?xWKT.`
			`_Instance geo:hasGeometry ?iGeom.`
			`?iGeom geo:asWKT ?iWKT.`
			`FILTER(_Relation(?xWKT, ?iWKT))`
			`}`

CRC	D	Which parks are within cities?	`SELECT ?x WHERE {`
			`?x rdf:type _Concept1; geo:hasGeometry ?xGeom.`
			`?xGeom geo:asWKT ?xWKT.`
			`?y rdf:type _Concept2; geo:hasGeometry ?yGeom.`
			`?yGeom geo:asWKT ?yWKT.`
			`FILTER(_Relation(?xWKT, ?yWKT))`
			`}`

CRIRI	E	Which beaches are at most 100 km distance to Ormos Vourkari in Greece?	`SELECT ?x WHERE {`
			`?x rdf:type _Concept; geo:hasGeometry ?xGeom.`
			`?xGeom geo:asWKT ?xWKT.`
			`_Instance1 geo:hasGeometry ?i1Geom.`
			`?i1Geom geo:asWKT ?i1WKT.`
			`_Instance2 geo:hasGeometry ?i2Geom.`
			`?i2Geom geo:asWKT ?i2WKT.`
			`FILTER(_Relation1(?xWKT, ?i1WKT) &&`
			`_Relation2(?i1WKT, ?i2WKT) )`
			`}`

5. THE QA ENGINE OF HAMZEI ET AL.

5.1 The engine of Hamzei et al.

Like GeoQA2, the engine of Hamzei et al. [23] takes as input a natural language question and translates it into a GeoSPARQL query targeting a version of YAGO2geo that has been extended with more data from OSM [23]. The engine uses a four-step workflow, shown in Figure 5.1 consisting of encoding extraction, grammatical parsing, intermediate representation generation, and GeoSPARQL query generation. These steps are described below using the question "How many pharmacies are in 200 meter radius of High Street in Oxford?" as an example.

The step of *encoding extraction* extracts certain kinds of information from the question and encodes them using an extension of the *encoding classes* of [24]. These encoding classes offer a rich representational framework that can be used to classify a geospatial question according to what kind of question word it uses (e.g., "how many"), whether semantic categories such as placenames (e.g., "High Street" and "Oxford"), place types (e.g., "pharmacies"), geospatial relations (e.g., "in 200 meter radius of" and "in") etc. are mentioned.[1] The encoding extraction step is implemented as a rule-based system but its POS tagging and named entity recognition components use the pre-trained neural network models of [34, 29] and the large language model BERT [10].

In the *grammatical parsing* step, the engine of Hamzei et al. constructs a constituency parse tree and a dependency parse tree for the input question. In this step, the *intention* of the question is also computed (e.g., "How many pharmacies") through the use of a number of heuristic rules.

The *intermediate representation generation* step uses the information produced by the previous two steps to compute a first-order logic formula corresponding to the input question. For the example question, the formula is

$$Count(x) : Place(High\ Street) \wedge Place(Oxford) \wedge Pharmacy(x) \\ \wedge InRadiusOf(x, HighStreet, 200meter) \wedge In(HighStreet, Oxford)$$

where our notation for first-order logic is the usual note.

The step of *GeoSPARQL query generation* produces a GeoSPARQL query based on the first-order logic formula of the previous step by utilizing YAGO2geo and its ontology. Classes, geospatial relations and properties are identified in similar fashion to GeoQA2. For named features (what GeoQA calls Instances) instead of doing place-name disambiguation, this step relies on string similarity search using an Apache Solr server for identifying instances. This means that the engine lacks a named entity recognition and disambiguation component, like the Instance identifier of GeoQA2. The query structure is generated dynamically by the intermediate, first-order logic representation, contrary to GeoQA no complete SPARQL/GeoSPARQL templates are employed.

The resulting query is subsequently sent to an Apache Jena Fuseki endpoint where YAGO2geo is stored to retrieve the answer(s).

The code for the Hamzei et al. engine is publicly available at [2] while a demo is available at [3].

[1] The conceptual framework of Hamzei et al. [23] is much richer than the one of GeoQA2 and it includes concepts such as events, times etc. but it has not been tested with KGs or datasets involving these concepts.

[2] `https://github.com/hamzeiehsan/Questions-To-GeoSPARQL`

[3] `https://tomko.org/demo/`

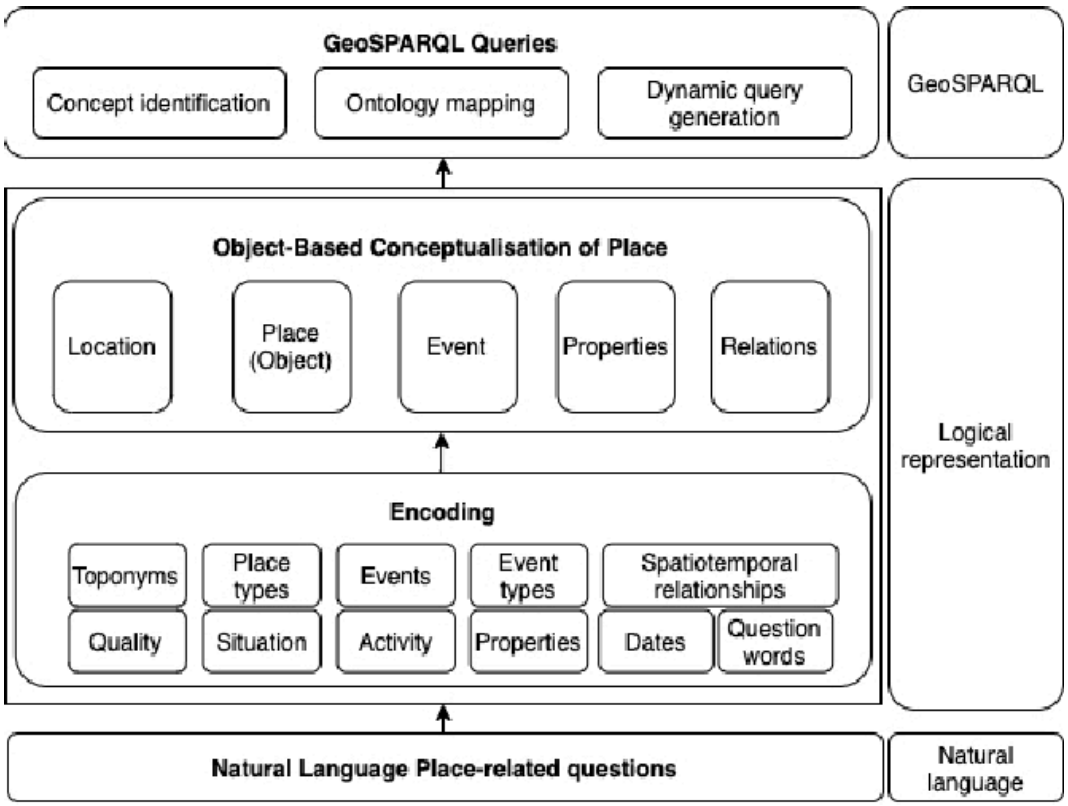

Figure 5.1: The conceptual architecture of the system of Hamzei et al. [23]

5.2 Summary

In this chapter, we presented the question answering engine of Hamzei et al. The engine follows a pipeline architecture and utilizes classical natural language processing techniques to process the input questions and generate executable GeoSPARQL queries over the YAGO2geo KG. Before generating the executable queries, a first-order logic intermediate representation is produced. This intermediate representation is subsequently translated to GeoSPARQL.

6. THE GEOQA3 QUESTION ANSWERING ENGINE

In this section we begin by presenting the architecture and fundamental logic of the GeoQA3 engine. Following that, we provide a comprehensive analysis of the query generation process.

6.1 Overview

GeoQA3, like all engines in the GeoQA family, takes as input a question in natural language and the YAGO2 [26] and YAGO2geo [30] knowledge graphs and produces one or more answers. The natural language question is translated to a SPARQL/GeoSPARQL query which is subsequently executed over a Strabon endpoint that stores both YAGO2 and YAGO2geo knowledge graphs. A notable departure from previous iterations is that GeoQA3 generates a single query.

Like its predecessors, GeoQA3 uses a pipeline architecture, meaning that it consists of a number of components, each of which performs a specific task. Information is propagated from one component to the next. The GeoQA3 QA pipeline, as showcased in 6.1, is split in 4 distinct conceptual steps:

1. WHERE clause generation
2. SELECT/ASK clause generation
3. Query generation
4. Query rewriting

Previous members of the GeoQA family utilized predefined templates for producing queries. This restricted the variety of questions that could be answered by those engines. In order to achieve a more flexible query generation process, in GeoQA3 the WHERE clause is generated dynamically by combining basic SPARQL/GeoSPARQL buildings blocks. This allows for the inclusion of information that can potentially be used in the next steps, and is utilized for handling natural language questions of arbitrary size. The identification of the correct building blocks is handled by the individual components that partake in this step, using a variety of techniques like dependency parsing, string similarity, lemmatization and named entity recognition and disambiguation.

The SELECT/ASK clause is generated by a combined approach that utilizes the large language model Llama2 [58] and dependency parsing. This step was pivotal in removing templates from our architecture and is heavily dependent on the usage of deep learning techniques offered by Llama2, which is responsible for understanding the intended

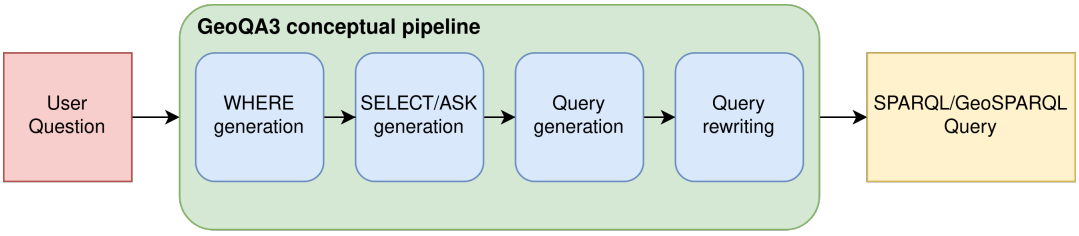

Figure 6.1: Conceptual steps of the GeoQA3 pipeline

return values of the questions. The return values are specified by traversing the dependency parse tree generated by the input question. Utilizing Llama2, allowed our engine to achieve a much more meaningful natural language understanding and provided it with the capability to handle more complex questions.

The query generation step works by combining the SELECT/ASK clause and WHERE clause generated by the previous conceptual steps. In addition, at this stage, any information about superlatives, ordering, grouping and specific number of expected return values is used to enhance the final generated query. All information required for this process is gathered in the aforementioned two steps.

The final conceptual step of our engine is the rewriting of the query generated by the previous step. The rewriting process is implemented, by using a transpiler that was developed by the AI Team of the University of Athens GoST[1], in which GeoSPARQL queries are transformed to their equivalent SPARQL queries, that utilize materialized information in the knowledge graph.

Unlike GeoQA2, GeoQA3 is not implemented using the Qanary [7] framework for pipeline question answering engines. Instead, we opted to rewrite our pipeline from scratch, which has allowed us to achieve a significantly faster query generation speed. This can be explained by the removal of the information propagation overhead that is caused by the extensive use of Stardog[2] as a bridge between components.

6.2 Pipeline

In this section, we provide a detailed analysis of GeoQA3's pipeline as presented in Figure 6.2.

6.2.1 WHERE clause generation

The WHERE clause generation process happens in the following components:

- Instance identifier
- Concept identifier
- Property identifier
- Geospatial relation identifier

As was previously mentioned, the WHERE clause of the generated query is created by combining simple SPARQL/GeoSPARQL blocks. Each component is responsible for identifying a kind of geographic feature, property or geospatial relation and creating the corresponding block. All components annotate the dependency parse tree with information about their discoveries. The functionality of the pipeline's components will be discussed using the question "Is the largest island in the United Kingdom larger than Crete by population?" (Figure 6.3).

[1] https://github.com/AI-team-UoA/GoST
[2] https://www.stardog.com/

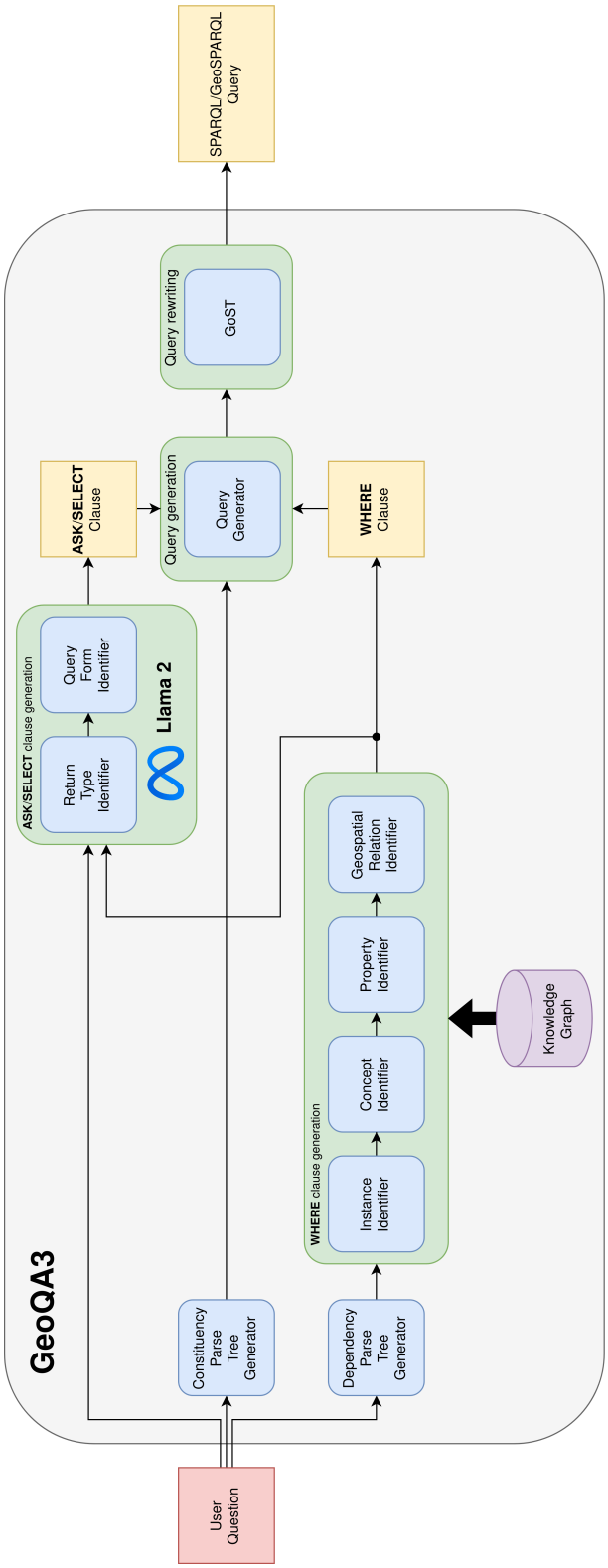

Figure 6.2: The complete GeoQA3 pipeline. The conceptual steps are shown in green.

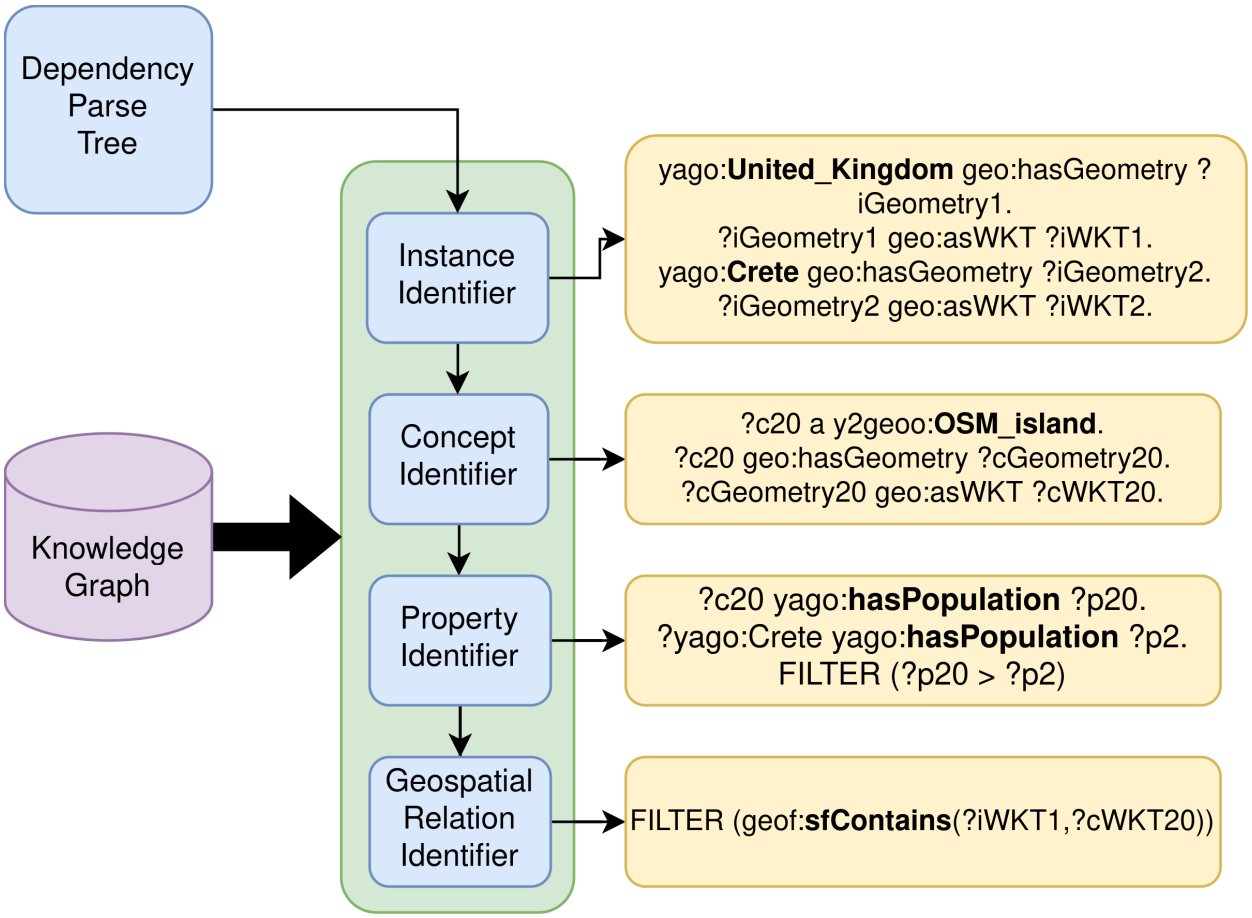

Figure 6.3: WHERE-clause generation.

Instance identifier. Similarly to GeoQA2, the *instance identifier* identifies the *features* (*instances*) present in the input question (e.g., “United Kingdom” and “Crete”). The features are identified using the TagMeDisambiguate tool [16]. Then, these elements are mapped to knowledge graph resources (e.g., `yago:United_Kingdom` and `yago:Crete`).

To facilitate our WHERE-clause generation process and improve the general performance of this component, we introduce the following two important optimizations:

1. For instances that consist of multiple words, e.g., New York City, after identifying the instance, the component modifies the dependency parse tree to merge the three nodes into one. As a consequence, the dependency parse tree is modified as elements are discovered. That applies to all components in this section. Particularly for the instance identifier, this avoids the problem of falsely identifying the word City as a concept in the following step.
2. In addition to identifying the instance, the instance identifier is responsible for creating the block that will be used in the WHERE clause for the identified instance. The generated block has the following structure:

```
<URI> geo:hasGeometry ?iGeomertryID .
?iGeomertryID geo:asWKT ?iWKTID .
```

Concept identifier. Similarly to the instance identifier, the concept identifier has not seen change functionality wise. It identifies the *types of features (concepts)* present in the input question (e.g., “island”) and maps them to the corresponding classes of the YAGO2 or YAGO2geo ontologies (e.g., `y2geoo:OSM_island`). These concepts are identified by the elements of the question that are tagged as nouns (POS tags NN, NNS, NNP and NNPS) during dependency parsing. Then, these elements are mapped to the ontology classes of YAGO2 and YAGO2geo using string matching based on n-grams.

The same two modifications that were applied to the Instance identifier were applied to the Concept identifier. Multi-word concepts lead to the modification of the dependency parse tree. The concept identifier is also responsible for generating the query block that will be used in the WHERE clause for any identified concepts. This has the following structure:

```
?cID a <URI> .
?cID geo:hasGeometry ?cGeometryID .
?cGeometryID geo:asWKT ?cWKTID .
```

Property identifier. The *property identifier* identifies *attributes of features or types of features* specified by the user in input questions and maps them to the corresponding properties in YAGO2 or YAGO2geo. For instance, for the example question the property “population” of type of feature “island” will be identified and mapped to property `yago:hasPopulation`.

In GeoQA2 only one property per question is supported, a limitation of the template-based approach to query generation. For GeoQA3, we wanted to overcome this limitation to be able to answer a wider array of question. For that reason, this component has been completely rewritten.

For each identified concept, we try to match, using string similarity, its properties to the words in the sentence. Matched properties are identified as candidate properties for this

concept. Multiple concepts might have the same candidate property. To combat this, we introduced a heuristic, that selects the closest concepts as the targets to the properties. The proximity heuristic is calculated using the distance of nodes in the dependency parse tree. For tiebreaks, the distance of the words in the natural language question is used. This process is similar for instances inside the question.

In addition, we reworked how superlative and implicit properties are identified in GeoQA3, again to overcome the one-property-per-sentence limitation of GeoQA2. We identify words with JJS part-of-speech tag (adjective, superlative) in our input questions. For each identified word, we compute the nearest dependency parse tree distance feature, whether that feature represents a concept or a property. When it pertains to a property, the information is stored for later use in the query generator as a superlative for that property. Superlatives are expressed through ORDER BY, DESC/ASC, and LIMIT clauses in SPARQL or GeoSPARQL queries. If the feature is a concept, it implies an implicit property. In such cases, as the property is not explicitly mentioned in the natural language question, we generate the property and store it as information to be utilized as a superlative for the generated property in the query generation process. For instance, in the case of an implicit property, consider the natural question 'Which is the largest lake in Greece?' In this context, we can deduce that the word 'largest' when referring to a 'lake' refers to the 'area' property. Thus, we create a property to represent 'area' and store it, along with its corresponding superlatives, for use in the query generation component.

The result of the property identifier has the following structure:

```
INSTANCE/CONCEPT_VARIABLE <URI> ?pID.
```

Geospatial relation identifier. The *geospatial relation identifier* first identifies the geospatial relations (e.g., "in") in the input question, and then maps them to the respective spatial function of the GeoSPARQL or stSPARQL vocabulary (e.g., `geof:sfWithin`) according to a mapping between geospatial relations and stSPARQL/GeoSPARQL functions provided by a dictionary.

This component was completely reworked to facilitate a more dynamic approach to query generation, without using templates, as well as incorporating deep learning techniques to improve natural language understanding, and with it improve overall system performance.

GeoQA3's geospatial relation identifier is implemented in the following manner. For each label of a geospatial relation, for example north of, contains, located, etc., n-grams are generated similarly to the concept identifier. If the relation is matched, the two closest geographic features (instances and concepts) are located using the following equation:

$$distance = dependency\_parse\_tree\_distance + (word\_distance/100)$$

By identifying the two closest features, we assign the closest feature as the first argument and the second closest feature as the second argument of the relation, except in the case of cardinal directions, where the positional order of the features in the sentence is the primary concern. When the geospatial relation pertains to distance, we search for the nearest CD (cardinal number) tag, which typically represents a numerical value. We use this number as the distance (with conversions like kilometers made accordingly). The NER component of CoreNLP [43] played a crucial role in the development of this step. Leveraging this deep learning component enabled us to efficiently locate CD tags and gather relevant information about the numerical values' interpretation within the question. With the NER component, the identifier was able to determine whether the number should

be used as greater, smaller, or equal in the FILTER clause of the query, as needed.

A part of the component's implementation involves how the dependency parse tree is navigated to identify paths to different concepts and instances. Our approach, which improved the performance of this component, was to halt the traversal within the current branch of the dependency parse tree when a geospatial relationship is encountered. This pruning technique assumes that when a geospatial relationship is in close proximity to a concept or instance, it holds a stronger association with that concept or instance. Consequently, there is no need to continue traversing the tree.

Furthermore, in order to implement conjunctions in sentences, the dependency parse tree is traversed to locate ”conj:and” edges which are stored for later usage. After all relationships are located, the edges containing conjunctions are visited to examine the feature that ”conj:and” targets. Utilizing the features and relationships, another relationship of the same type is added by replacing the non-target feature.

The result of the geospatial relation identifier has the following structure:

```
FILTER (<URI> (FIRST_FEATURE, SECOND_FEATURE))
```

and

```
FILTER (geof:distance (FIRST_FEATURE, SECOND_FEATURE, uom:metre)
    {<, >, <=, >=} DISTANCE)
```

6.2.2 SELECT/ASK clause generation

The ASK/SELECT clause generation process happens in the following components:

- Return Type identifier
- Query Form identifier

Return type identifier. The *return type identifier* is responsible for identifying the expected form/type of the answer to the question. Our example question is a yes or no question, so the expected return type is a boolean value. Such questions are answered using ASK statements, while questions that expect row of values are answered using SELECT statements. For SELECT statements, this component is also responsible for predicting the types of the returned values. The supported types are { Name, Coordinates, Number-Property, Number-Count }.

During the development of the return type identifier, we opted to leverage the deep learning framework Llama2 [58] due to its proven efficiency and open-source nature. In implementing this component, we conducted extensive experimentation with various Llama2 settings to enhance our system's performance.

Our trials with Llama2 7B and 13B yielded middling results, as they struggled to fully comprehend the prompts provided. Consequently, we chose to employ Llama2 70B in a few-shot setting instead of a zero-shot setting, as the latter generally failed to deliver satisfactory outcomes. The integration of this system into GeoQA3 now enables us to efficiently generate the return types required to formulate the final SPARQL or GeoSPARQL queries. The prompt we deployed for our system is showcased at the end of this section.

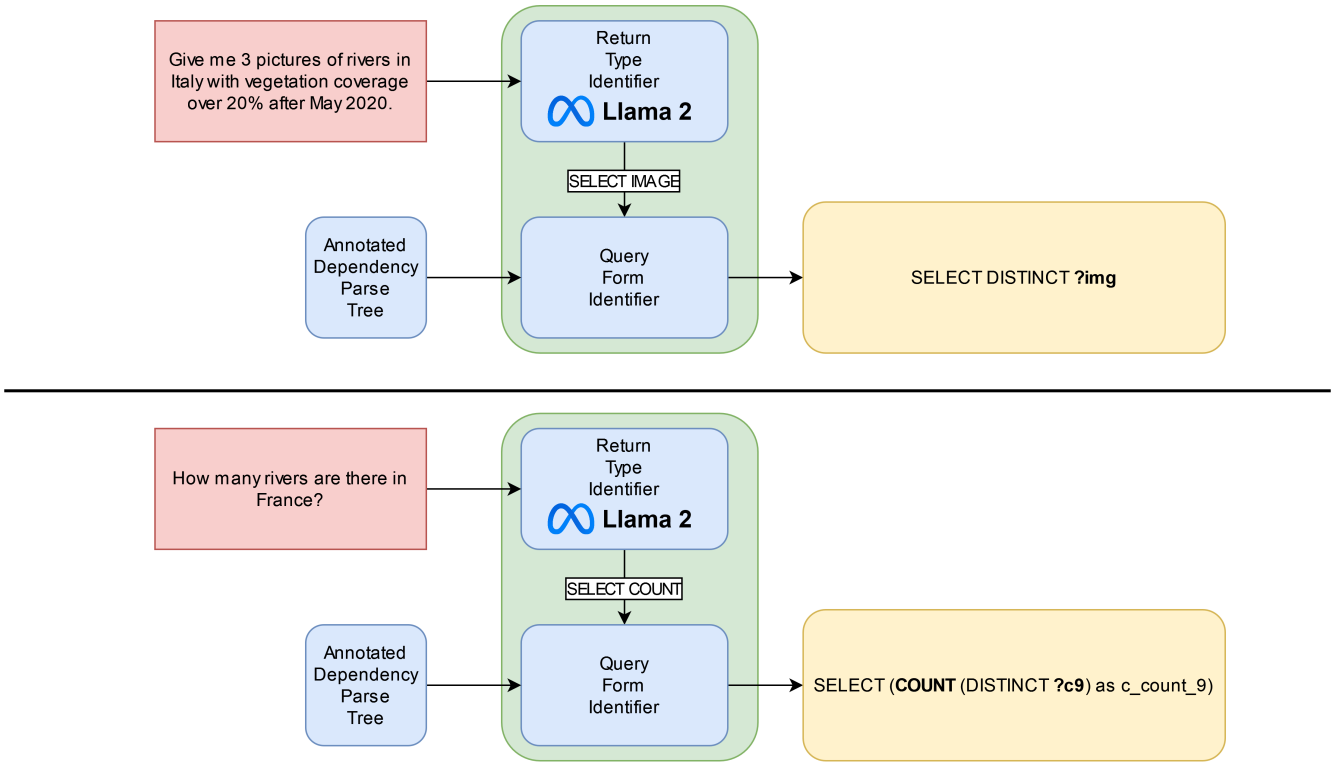

Figure 6.4: SELECT-clause generation.

It's worth noting that we refrained from using Llama2 to determine whether the generated query should be a SELECT or an ASK query. This decision was made because this task can be performed through lexical analysis, which offers both faster execution speed and effectiveness comparable to the results achieved by Llama2.

Query form identifier. The *return type identifier* is responsible for generating the final ASK/SELECT clause, which will be used by the query generator. It takes as input the return types generated by the return type identifier. For each expected return type, we follow an iterative approach as follows: If the type is 'Name,' we search for the next concept. If the type is 'Coordinates,' we seek the next concept or instance. When the return type is 'Number-Property,' we look for the next property, and if it's 'Number-Count,' we search for the next concept. Additionally, we enhance the query by introducing a COUNT aggregation and the necessary GROUP BY clauses.

To determine the 'next' object, we traverse the dependency parse tree. The traversal begins from the Wh-word of the sentence. Wh-words are those that are tagged as WP, WFT, WRB by the CoreNLP part of speech tagger. After locating Wh-words, we identify the objects that are closest based on the dependency parse tree distance.

The prompt used by Llama 70B is presented below:

```
SYSTEM (inside <sys> tags): You are an assistant that can only use the words
{ Name, Coordinates, Number-Property, Number-Count }. Answer appropriately and concis
Instruction (inside <INST> tags):
Select the most fitting expected answer type from these options
{ Name, Coordinates, Number-Property, Number-Count }.
I don't want you to answer the question, but to return the type of the answer. Some e
    QUESTION: Which parks are in New York City?
    ANSWER: Name
    QUESTION: Which towns in England are near rivers?
    ANSWER: Name
    QUESTION: What are the 5 tallest mountains of Mexico?
    ANSWER: Name
    QUESTION: Which is largest lake in Brazil?
    ANSWER: Name
    QUESTION: Which towns are less than 10kms away from the Grand Canyon?
    ANSWER: Name
    QUESTION: In Rhodes, which towns are close to a lake and near a mountain?
    ANSWER: Name
    QUESTION: Where is Greece located?
    ANSWER: Coordinates
    QUESTION: Where is Washington D.C. located?
    ANSWER: Coordinates
    QUESTION: What is the location of Norway?
    ANSWER: Coordinates
    QUESTION: Where is Moscow situated?
    ANSWER: Coordinates
    QUESTION: What are the coordinates of Tokyo?
    ANSWER: Coordinates
    QUESTION: Where is the capital of Spain located?
    ANSWER: Coordinates
    QUESTION: How large is Lake Trichonida?
    ANSWER: Number-Property
    QUESTION: What is the area of France?
    ANSWER: Number-Property
    QUESTION: What is the population density of Athens?
    ANSWER: Number-Property
    QUESTION: What is the average surface area of lakes in Italy?
    ANSWER: Number-Property
    QUESTION: How many lakes are there in Germany?
    ANSWER: Number-Count
    QUESTION: What is the number of mountains in County Sligo?
    ANSWER: Number-Count
    QUESTION: Where is Greece located and how many islands does it have?
    ANSWER: Coordinates Number-Count
    QUESTION: What are the five biggest cities in France and their population?
    ANSWER: Name Number-Property
    QUESTION: Which is the tallest mountain in the US and how tall is it?
    ANSWER: Name Number-Property
```

```
QUESTION: Which Greek regions have more than 3 municipalities and
    how many municipalities do they have?
ANSWER: Name Number-Count
QUESTION: Which is the smallest and which is the largest forest in Turkey?
ANSWER: Name Name
QUESTION: MY_QUESTION
```

6.2.3 Final query generation

The query generator is responsible for generating the final query. Within this stage of the pipeline, it assimilates all the information provided by the preceding components and combines them into a suitable, executable SPARQL or GeoSPARQL query. This component also determines the inclusion of superlatives such as 'the most,' 'the least,' 'the fewest,' and so on. Once again, the dependency parse tree distance is employed to pinpoint the relevant features, and the requisite GROUP BY and ORDER BY clauses are inserted accordingly, if needed.

6.2.4 Query Materialization and Execution

Query materialization and execution is the final component of the GeoQA3 pipeline and is responsible for rewriting GeoSPARQL queries into SPARQL queries, in order to utilize the materialized geospatial relationships inside the knowledge graph. The materialization process and optimization is further discussed in [31].

The rewriting process is achieved by integrating the GoST transpiler, which relies on Apache Jena [17]. GoST leverages the visitor patterns provided by Jena to parse the input GeoSPARQL query and extract the geospatial Boolean functions. It handles FILTERs within the query, processing and consolidating them into a single filter. This filter, in turn, is used to structure the query into a tree-like arrangement, encompassing UNION clauses for logical ORs within the FILTER and EXCEPT clauses for logical NOTs. The inherent conjunctive structure of triples is maintained within each branch. In the final, this tree-like structure is assembled into a query using UNION and EXCEPT clauses. Importantly, the final query omits any Boolean geospatial relationships that are already materialized within the knowledge graph and replaces them with the appropriate triples.

After the generation and the rewriting of the query, its execution happens in a Strabon triple store endpoint. To the best of our knowledge, GeoQA3 is the first engine to integrate this functionality in its pipeline.

6.3 Summary

In this chapter, we presented our engine GeoQA3 and how it combines classical natural language processing techniques, like those used in GeoQA2 and the engine of Hamzei et al., to generate the WHERE-clause with modern neural techniques to generate the ASK/SELECT-clause. This way our pipeline uses a hybrid approach to query generation aiming to achieve a more sophisticated understanding of natural language while maintaining the ability to generate well-formed queries. GeoQA3 is the first engine to forego handwritten query templates, making it able to tackle a wider array of questions. It is also, to the best of our knowledge, the first question answering that utilizes materialized geospatial relations.

7. EVALUATION

In this chapter we present an evaluation of GeoQA3 on the dataset GeoQuestions1089 and a comparison to GeoQA2 and the engine of Hamzei et al. We start this chapter by shortly introducing the GeoQuestions1089 dataset and

7.1 The GeoQuestions1089 dataset

The GeoQuestions1089 dataset consists 1089 triples of questions-queries-answers. The questions were crowdsourced by students of the University of Athens. The dataset is split in two parts, GeoQuestions$_C$ (1017 questions) and GeoQuestions$_W$ (72 questions) both of which target the union of YAGO2 and YAGO2geo.

The dataset GeoQuestions$_C$ was checked by the authors. Each question (query) was checked both grammatically and syntactically, using Grammarly ([1]) and QuillBot ([2]). The resulting set contained 1017 question-query-answer triples.

GeoQuestions$_W$ consists of the elements of GeoQuestions$_C$ whose questions originally had spelling, grammar or syntax mistakes. The end goal of benchmarking how capable QA engines are at handling incorrect input.

Following the categorization of [49], we can see that the questions of dataset GeoQuestions1089 fall under the following categories:[3]

A. Asking for a thematic or a spatial attribute of a feature, e.g., *"Where is Loch Goil located?"*. In GeoQA2, these questions can be answered by posing a SPARQL query to YAGO2geo.

B. Asking whether a feature is in a geospatial relation with another feature or features, e.g., *"Is Liverpool east of Ireland?"*. The geospatial relation in this example question is a cardinal direction one (east of). Other geospatial relations in this category of questions include topological ("borders") or distance ("near" or "at most 2km from").

C. Asking for features of a given class that are in a geospatial relation with another feature. E.g., *"Which counties border county Lincolnshire?"* or *"Which hotels in Belfast are at most 2km from George Best Belfast City Airport?"*. The geospatial relation in the first example question is a topological one ("border"). As in the previous category, other geospatial relations in this set of questions include cardinal or distance (as in the second example question).

D. Asking for features of a given class that are in a geospatial relation with any features of another class, e.g., *"Which churches are near castles?"*. Arguably, this category of questions might not be useful unless one specifies a geographical area of interest; this is done by the next category of questions.

E. Asking for features of a given class that are in a geospatial relation with an unspecified feature of another class, and either one or both, is/are in another geospatial relation with a feature specified explicitly. E.g., *"Which churches are near a castle in Scotland?"* or *"In Greece, which beaches are near villages?"*.

[1] `https://www.grammarly.com/`

[2] `https://quillbot.com/`

[3] For comparison purposes, for each question category, we comment whether the search engines Google and Bing can answer such questions after having tried a few examples.

F. As in categories C, D and E above, plus more thematic and/or geospatial characteristics of the features expected as answers, e.g., *"Which mountains in Scotland have height more than 1000 meters?"*.

G. Questions with quantities and aggregates, e.g., *"What is the total area of lakes in Monaghan?"* or *"How many lakes are there in Monaghan?"*.

H. Questions with superlatives or comparatives, e.g., *"Which is the largest island in Greece?"* or *"Is the largest island in France larger than Crete?"*.

I. Questions with quantities, aggregates, and superlatives/comparatives, e.g., *"Which city in the UK has the most hospitals?"* or *"Is the total size of lakes in Greece larger than lake Loch Lomond in Scotland?"*.

Table 7.1 describes GeoQuestions1089 giving numbers per type of question.

7.2 Comparison to GeoQuestions201

GeoQuestions201 contains mostly simple questions that can be answered with simple queries. For that reason, the state of the art geospatial QA engines are able to answer a significant portion of it correctly, as was shown in [23] and confirmed by our own experience while developing GeoQA3.

GeoQuestions1089 includes numerous complex questions that require both solid natural language understanding and advanced SPARQL features (nested queries, not-exists filters, arithmetic calculations) to be answered. For example: *"How many times bigger is the Republic of Ireland than Northern Ireland?"* or *"What is the population density of the municipality of Thessaloniki?"* or *"How much of the UK is woodland?"* or *"Is Belfast closer to the capital of the Republic of Ireland or the capital of Scotland?"* or *"Which islands don't have any lakes but have forests?"*. Additionally, GeoQuestions1089 is targeted on YAGO2geo, enabling easier comparison of engines that target this KG. Furthermore, because YAGO2geo also includes data about the United States and Greece, new challenges arise that must be dealt with by a good QA engine. For instance, some Greek entities lack English labels, which makes disambiguation more difficult. All in all, GeoQuestions1089 is a more varied and more challenging dataset that uses a much wider array of SPARQL functionality in its queries compared to GeoQuestions201.

7.3 Using GeoQuestions1089 to benchmark Geospatial Question Answering engines

In this section, we use the dataset GeoQuestions1089 to benchmark the QA engines GeoQA2, the one by [23] and finally GeoQA3. We ran the experiments on a machine with the following specifications: Intel Xeon E5-4603 v2 @2.20GHz, 128 Gb DDR3 RAM, 1.6 TB HDD (RAID-5 configuration).

7.3.1 Methodology and metrics

The question answering engine that is being evaluated attempts to generate a query for each natural language question in the dataset. If the generation is successful, the query is then processed by the transpiler that rewrites the query using materialized relations and it is then sent to a geospatial RDF store that executes the query over our knowledge graph.

Table 7.1: GeoQuestions1089 statistics

Category	KG	Count in GeoQuestions$_C$	Combined in GeoQuestions$_C$	Count in GeoQuestions$_W$	Combined in GeoQuestions$_W$
A	YAGO2geo	144	175	14	17
	YAGO2geo + YAGO2	31		3	
B	YAGO2geo	134	139	11	11
	YAGO2geo + YAGO2	5		0	
C	YAGO2geo	155	178	12	14
	YAGO2geo + YAGO2	23		2	
D	YAGO2geo	25	25	0	0
	YAGO2geo + YAGO2	0		0	
E	YAGO2geo	134	135	7	7
	YAGO2geo + YAGO2	1		0	
F	YAGO2geo	21	24	1	2
	YAGO2geo + YAGO2	3		1	
G	YAGO2geo	146	174	8	11
	YAGO2geo + YAGO2	28		3	
H	YAGO2geo	114	142	7	8
	YAGO2geo + YAGO2	28		1	
I	YAGO2geo	22	25	2	2
	YAGO2geo + YAGO2	3		0	
All	YAGO2geo	895	1017	62	72
	YAGO2geo + YAGO2	122		10	

The result is compared to the gold result included in GeoQuestions1089. In the case of GeoQA3, the transpiler is already embedded as a component, so the rewriting process happens inside the engines pipeline. To accept an answer as correct, it must match the gold result exactly. We do not consider partially correct answers (e.g., when computed answers are a proper subset of the ones in the gold set) as correct. Likewise, we do not consider a superset of the answers in the gold set as correct. We chose to not use precision, recall and F-measure because the correct number of returned answers/entities for each query varies greatly, which biases the metric towards certain kinds of questions.

7.3.2 Evaluating GeoQA2

To evaluate GeoQA2 we set up three Strabon [33] endpoints. In the first two we store YAGO2 and YAGO2geo respectively. These endpoints are required by GeoQA2 to generate queries. In the third endpoint, which we use for retrieving the answers to our generated queries, we store YAGO2, YAGO2geo and its materialization.

Tables 7.2 and 7.3 show the results of the evaluation. The column “Generated Queries” gives the percentage of questions for which GeoQA2 was able to generate a query. The column “Correct Answers” gives the percentage of questions for which the query that was generated was able to retrieve the correct set of answers.

We observe that the complexity of the structure of the question affects significantly the performance of the system. For instance, GeoQA2 performed decently in answering rather simple questions (i.e., geospatial relation between two features), while it has difficulties in answering more structurally complex questions (i.e., questions with a combination of superlatives and quantities, questions with more sophisticated syntax or vocabulary). In addition, we see that GeoQA2 is a robust engine, meaning that it loses only a small percentage of its effectiveness when the input questions contain spelling, grammar or syntax mistakes.

Our benchmark showcases three core weaknesses of the GeoQA2 engine. First, a rule-based understanding of natural language, which falls apart for questions outside the specified rules. Second, the inherent difficulty of instance identification, especially for entities that have the same or extremely similar names (e.g., there are multiple places called Athens). Third, the limited array of GeoSPARQL queries that can be constructed using the existing templates, which are not enough to answer many of our more complex questions.

7.3.3 Evaluating the system of Hamzei et al.

The engine of [23] requires two servers, an Apache Solr server, used for placename and place type identification, and an Apache Jena GeoSPARQL Fuseki server for executing the generated queries. Even though a Solr index is provided in the code repository of the engine, it is not suitable for our dataset. [23] use a modified version of YAGO2geo that does not include Greece and includes a number of additional entities from Open Street Map. We create a new Solr index that includes YAGO2 and YAGO2geo. We load the Fuseki endpoint with YAGO2, YAGO2geo, the materialized relations of YAGO2geo and the materialization of the surface area of every polygon in YAGO2geo. The last part is necessary because Fuseki does not have the ability to calculate the surface area of a polygon.

In a similar vein to the evaluation of GeoQA2, the generated queries of the engine are

Table 7.2: Evaluation of GeoQA2 over GeoQuestions$_C$.

Category	Generated Queries	Correct Answers
A	84%	47.42%
B	76.25%	58.99%
C	79.21%	44.38%
D	56%	12%
E	80%	31.85%
F	66.66%	16.66%
G	74.13%	32.18%
H	71.12%	26.05%
I	84%	20%
Total	76.99%	38.54%

Table 7.3: Evaluation of GeoQA2 over GeoQuestions$_W$.

Category	Generated Queries	Correct Answers
A	82%	47.05%
B	81.81%	54.54%
C	85.71%	57.14%
D	50%	33%
E	88%	0.00%
F	36.36%	0.00%
G	50.00%	0.00%
H	100.00%	0.00%
I	50%	50%
Total	72.22%	34.72%

processed by our transpiler before being sent to the Apache Jena Fuseki endpoint whose answer is compared to that included in GeoQuestions1089. To communicate with the Fuseki endpoint we use Apache Jena's own SPARQL-OVER-HTTP scripts to make sure that queries are sent and results are returned correctly. Tables 7.4 and 7.5 show the results of the evaluation.

We make three main observations. First, we see that as questions become more complex, the effectiveness of the engine drops dramatically, as was the case in our evaluation of GeoQA2. The more complex the question, the less likely it is that the query generator is able to construct the proper GeoSPARQL query, with the most extreme example being questions of type I. Second, the system severely underperforms in questions of Category A, which is one of the simpler categories. This is caused by the lack of a dedicated step for named entity disambiguation. For example, if given the input question "*Where is Dublin located?*" the engine of [23] will return the location of every place named "Dublin" in the KG, instead of the location of the capital of the Republic of Ireland. This leads to an explosive increase of returned answers. Moreover, there is no mechanism for ranking the returned answers in accordance to their relevance, so even taking the first 3 answers as candidates doesn't significantly change the picture. Instead of a dedicated disambiguation step, the engine relies on the automatic resolution of disambiguation during query execution, which is an approach that works well for category B questions. In the original evaluation of their system, the authors disregarded toponym disambiguation, but we consider it a core part of question answering. Third, the system can handle spelling, grammar, and syntax mistakes without performance loss.

The main weakness of the engine of [23] is the lack of a dedicated disambiguation step. This leads to answers that contain numerous irrelevant results, i.e., the system is lacking precision. The other significant weakness is the rule-based approach to query generation

Table 7.4: Evaluation of the system of [23] over GeoQuestions$_C$. Because the query generator of the engine was not designed to work with entities that do not have detailed geometries, we also provide statistics for the subset of questions that target YAGO2geo only.

Category	GeoQuestions$_C$		GeoQuestions$_C$ without YAGO2 Questions	
	Generated Queries	Correct Answers	Generated Queries	Correct Answers
Type-A	89.71%	10.85%	88.88%	12.50%
Type-B	95.68%	53.23%	95.52%	55.22%
Type-C	97.75%	30.33%	97.41%	32.90%
Type-D	100%	12%	100%	12%
Type-E	99.25%	7.40%	99.25%	7.46%
Type-F	79.16%	4.10%	76.19%	4.76%
Type-G	98.27%	11.49%	97.94%	13.01%
Type-H	97.18%	7.74%	96.49%	7.89%
Type-I	92%	0%	95%	0%
Total	95.77%	18.97%	95.53%	20.67%

Table 7.5: Evaluation of the system of [23] over GeoQuestions$_W$

Category	GeoQuestions$_W$	
	Generated Queries	Correct Answers
A	88.23%	17.64%
B	100.00%	54.54%
C	100.00%	35.71%
D	100.00%	0.00%
E	87.50%	0.00%
F	90.90%	0.00%
G	100.00%	0.00%
H	100.00%	0.00%
I	100.00%	0.00%
Total	94.44%	19.44%

that is unable to deal with complex queries.

7.3.4 Evaluating GeoQA3

Similarly with the evaluation of GeoQA2, we set up three Strabon [33] endpoints. In the first two we store YAGO2 and YAGO2geo respectively. These endpoints are required by GeoQA3 to generate queries. In the third endpoint, which we use for retrieving the answers to our generated queries, we store YAGO2, YAGO2geo and its materialization.

Tables 7.6 and 7.7 show the results of the evaluation. The column “Generated Queries” gives the percentage of questions for which GeoQA3 was able to generate a query. The column “Correct Answers” gives the percentage of questions for which the query that was generated was able to retrieve the correct set of answers.

We observe that the complexity of the structure of the question affects significantly the performance of the system. For instance, GeoQA3 like GeoQA2 performed decently in answering rather simple questions (i.e., geospatial relation between two features), while it has difficulties in answering more structurally complex questions (i.e., questions with a

Table 7.6: Evaluation of GeoQA3 over GeoQuestions$_C$.

Category	Generated Queries	Correct Answers
A	77.08%	56.00%
B	91.79%	57.50%
C	87.09%	45.46%
D	88.00%	24.0%
E	94.77%	34.80%
F	85.71%	8.33%
G	88.35%	33.90%
H	85.96%	29.50%
I	72.72%	24.00%
Total	87.03%	41.38%

Table 7.7: Evaluation of GeoQA3 over GeoQuestions$_W$.

Category	Generated Queries	Correct Answers
A	64.70%	35.29%
B	100%	18.18%
C	100%	35.71%
D	0.00%	0.00%
E	71.42%	14.28%
F	100.00%	0.00%
G	45.45%	18.18%
H	62.50%	0.25%
I	50.00%	50.00%
Total	77.77%	26.38%

combination of superlatives and quantities, questions with more sophisticated syntax or vocabulary). In addition, GeoQA3 maintains its robustness even when input questions contain spelling, grammar or syntax mistakes.

Our benchmark showcases two core weaknesses of the GeoQA3 engine, which are similar to the first two weaknesses appearing in GeoQA2. First, a rule-based understanding of natural language, which falls apart for questions outside the specified rules. Second, the inherent difficulty of instance identification, especially for entities that have the same or extremely similar names (e.g., there are multiple places called Athens).

7.3.5 Engine Comparison

The results of our evaluation, for engines using templates, show that GeoQA2 significantly outperforms the QA engine of [23] by generating twice the amount of correct queries. The main factor of this performance gap is the existence of a dedicated named entity disambiguation step in GeoQA2 (instance identifier). Other than this main difference, the two engines are similar in a number of ways. Both utilize dependency and constituency parsing to understand the structure of the input question and the relations that exist among its tokens. Likewise, both engines have a rule-based query generator, although the engine of [23] uses a more dynamic of approach of combining smaller templates which allows it to generate queries for a significantly larger portion of the dataset. Considering these similarities, it follows that the engines must share some weaknesses. That is the case, with the inability of either engine to reliably answer complex questions being their most important weakness.

According to the outcomes of our evaluation, GeoQA3 clearly surpasses the QA engine presented by [23], for reasons similar to GeoQA2, and delivers enhancements and a slight

overall performance increase compared to GeoQA2 for sentences that are grammatically and syntactically correct, i.e., GeoQuestions$_C$. Firstly, by employing a dynamic approach to query generation and removing the usage of templates, we achieved a generation technique that successfully produces results even from complex questions, which as shown in Tables 7.6 massively outperforms GeoQA2 in generating executable SPARQL and GeoSPARQL queries. Secondly, from the results it is showcased that correct answers are slightly increased in almost all categories compared to GeoQA2. By utilizing Llama 2 [58] and Stanford's NER [43] deep learning techniques, our geospatial QA engine managed a better and more meaningful natural language understanding, which provide the information required for the generation of a query from a given natural language question. At the same time, a consequence of this improved natural language understanding being a core part of the GeoQA3 pipeline is that the engine produces worse results for sentences that include significant syntactical mistakes, like those in GeoQuestions$_W$, which in turn leads to a lower overall score in this subset of GeoQuestions1089. It is important to note that GeoQA3's ability to answer questions with multiple properties or many concepts and instances is not well represented in the evaluation results. Even though a large number of the questions in GeoQuestions1089 pose significant challenges, the amount of features and properties in them is small. This is a result of the crowdsourcing effort and the instructions given to the students. GeoQA2 is limited to answering questions that have at most two (2) concepts, two (2) instances and one (1) property. GeoQA3 has no such limitation making it able to generate queries for a significantly larger variety of questions.

7.4 Summary

In this chapter, we presented an evaluation of GeoQA3 on the geospatial question answering benchmark GeoQuestions1089 and how its performance compares to our question answering baselines. We showed that GeoQA3 outperforms both engines, although only by 7% in the case of GeoQA2. We presented our interpretation of the results and the limitations of each engine. Nonetheless, we consider these results promising, considering that this is a new approach to question answering that has not been iterated upon. In comparison, GeoQA2 has been worked on for 6 years. Additionally, the chosen benchmark does not capture GeoQA3's ability to answer more varied questions, like questions that contain more concepts or features.

8. CONCLUSIONS & FUTURE DIRECTIONS

In this thesis we have addressed the challenges of providing access to the YAGO2geo KG for non-expert users using natural language QA interfaces. Given the use of geospatial contexts in many practical situations, this challenge is of major importance while adopting QA for wide use.

We presented the geospatial question answering engine GeoQA3, the latest in the family of GeoQA question answering engines and the first to introduce the usage of Large Language Models in its pipeline. GeoQA3 debuts the novel approach of LLM-guided query generation in the space of geospatial question answering, combining heuristics and traditional semantic parsing, with state-of-the-art neural models. GeoQA3 is not a template-based system, meaning that it can answer a wider array of questions, of any length. All in all, our system improves upon the state of the art of geospatial question answering, while also having a high future development ceiling, having surpassed the performance of GeoQA2 which has been iterated upon for a number of years.

GeoQA3 as a basis for the question answering engine EarthQA2 for satellite data archives. EarthQA2 can be used to answer queries such as “Find Sentinel-2 images that cover Crete, have cloud cover less than 10% and have been captured in July 2023”.

For future work, we will continue advancing our approach by using neural models to extract more information out the natural language questions. To that end, we will extend upon our current approach of LLM-guided query generation, by fine-tuning Llama2 on GeoQuestions1089 to provide more detailed and complex information which will guide the query generator through more complex tasks. A prototype implementation of this advancement allows us to generate LIMIT clauses with greater accuracy, as well as identifying properties more reliably. In addition, we are looking into integrating constrained natural languages as intermediate targets for query generation, as done in [36] for SPARQL, which could possibly lead to the creation of an end-to-end neural model for question answering over geospatial knowledge graphs. We will also look at the potential of using LLMs to modify and/or complete queries to combat the issue of error propagation and to handle complex queries that require nesting.

ABBREVIATIONS - ACRONYMS

LLM	Large Language Model
KG	Knowledge Graph
QA	Question Answering
GIS	Geographic Information Systems
NER	Named Entity Recognition
NERD	Named Entity Recognition and Disambiguation
POS	Part-of-Speech
URI	Uniform Resource Identifier
LSTM	Long Short-Term Memory
OSM	OpenStreetMap
GADM	Global Administrative Areas
NBD	National Boundary Dataset
OGC	Open Geospatial Consortium
POI	Point of Interest
AI	Artifical Intelligence
NLP	Natural Language Processing

BIBLIOGRAPHY

[1] Elise Acheson, Stefano De Sabbata, and Ross S. Purves. A quantitative analysis of global gazetteers: Patterns of coverage for common feature types. *Computers, Environment and Urban Systems*, 64, 2017.

[2] Dirk Ahlers. Assessment of the accuracy of GeoNames gazetteer data. In *GIR*, 2013.

[3] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary G. Ives. DBpedia: A nucleus for a web of open data. In Karl Aberer, Key-Sun Choi, Natasha Fridman Noy, Dean Allemang, Kyung-Il Lee, Lyndon J. B. Nixon, Jennifer Golbeck, Peter Mika, Diana Maynard, Riichiro Mizoguchi, Guus Schreiber, and Philippe Cudré-Mauroux, editors, *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007*, volume 4825 of *Lecture Notes in Computer Science*, pages 722–735. Springer, 2007.

[4] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1533–1544. ACL, 2013.

[5] Kurt D. Bollacker, Robert P. Cook, and Patrick Tufts. Freebase: A shared database of structured general human knowledge. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada*, pages 1962–1963. AAAI Press, 2007.

[6] Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. Large-scale simple question answering with memory networks. *CoRR*, abs/1506.02075, 2015.

[7] Andreas Both, Dennis Diefenbach, Kuldeep Singh, Saeedeh Shekarpour, Didier Cherix, and Christoph Lange. Qanary - A methodology for vocabulary-driven open question answering systems. In Harald Sack, Eva Blomqvist, Mathieu d'Aquin, Chiara Ghidini, Simone Paolo Ponzetto, and Christoph Lange, editors, *The Semantic Web. Latest Advances and New Domains - 13th International Conference, ESWC 2016, Heraklion, Crete, Greece, May 29 - June 2, 2016, Proceedings*, volume 9678 of *Lecture Notes in Computer Science*, pages 625–641. Springer, 2016.

[8] Wei Chen. Parameterized spatial SQL translation for geographic question answering. In *2014 IEEE International Conference on Semantic Computing, Newport Beach, CA, USA, June 16-18, 2014*, pages 23–27. IEEE Computer Society, 2014.

[9] Yi-Hui Chen, Eric Jui-Lin Lu, and Ting-An Ou. Intelligent sparql query generation for natural language processing systems. *IEEE Access*, 2021.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019.

[11] Dennis Diefenbach, Vanessa López, Kamal Deep Singh, and Pierre Maret. Core techniques of question answering systems over knowledge bases: a survey. *Knowl. Inf. Syst.*, 55(3):529–569, 2018.

[12] Dennis Diefenbach, Thomas Pellissier Tanon, Kamal Deep Singh, and Pierre Maret. Question Answering Benchmarks for Wikidata. In *ISWC Posters & Demos*, 2017.

[13] Alishiba Dsouza, Nicolas Tempelmeier, Ran Yu, Simon Gottschalk, and Elena Demidova. WorldKG: A world-scale geographic knowledge graph. In *CIKM '21: The 30th ACM International Conference on Information and Knowledge Management, Virtual Event, Queensland, Australia, November 1 - 5, 2021*, pages 4475–4484. ACM, 2021.

[14] Max J. Egenhofer and Robert D. Franzosa. Point set topological relations. *International Journal of Geographical Information Systems*, 5(2):161–174, 1991.

[15] Yu Feng, Linfang Ding, and Guohui Xiao. Geoqamap -geographic question answering with maps leveraging llm and open knowledge base. 09 2023.

[16] Paolo Ferragina and Ugo Scaiella. TAGME: on-the-fly annotation of short text fragments (by wikipedia entities). In Jimmy Huang, Nick Koudas, Gareth J. F. Jones, Xindong Wu, Kevyn Collins-Thompson, and Aijun An, editors, *Proceedings of the 19th ACM Conference on Information and Knowledge Management, CIKM 2010, Toronto, Ontario, Canada, October 26-30, 2010*, pages 1625–1628. ACM, 2010.

[17] Apache Software Foundation. *Apache Jena*, 2023.

[18] Andrew U. Frank. Qualitative spatial reasoning about distances and directions in geographic space. *J. Vis. Lang. Comput.*, 3(4):343–371, 1992.

[19] Georgios N Giatrakos. Extending yago2geo with geospatial information from other countries. 2020.

[20] Rolf Grütter, Ross S. Purves, and Lukas Wotruba. Evaluating topological queries in linked data using DBpedia and GeoNames in switzerland and scotland. *Trans. GIS*, 21(1):114–133, 2017.

[21] Ehsan Hamzei. *Place-related question answering: From questions to relevant answers*. PhD thesis, 2021.

[22] Ehsan Hamzei, Haonan Li, Maria Vasardani, Timothy Baldwin, Stephan Winter, and Martin Tomko. Place questions and human-generated answers: A data analysis approach. In Phaedon C. Kyriakidis, Diofantos G. Hadjimitsis, Dimitrios Skarlatos, and Ali Mansourian, editors, *Geospatial Technologies for Local and Regional Development - Proceedings of the 22nd AGILE Conference on Geographic Information Science, Limassol, Cyprus, June 17-20, 2019*, Lecture Notes in Geoinformation and Cartography, pages 3–19. Springer, 2019.

[23] Ehsan Hamzei, Martin Tomko, and Stephan Winter. Translating place-related questions to GeoSPARQL queries. In *Proceedings of the Web Conference (WWW)*, 2022.

[24] Ehsan Hamzei, Stephan Winter, and Martin Tomko. Place facets: a systematic literature review. *Spatial Cogn. Comput.*, 20(1):33–81, 2020.

[25] Lynette Hirschman and Robert J. Gaizauskas. Natural language question answering: the view from here. *Nat. Lang. Eng.*, 7(4):275–300, 2001.

[26] Johannes Hoffart, Fabian M. Suchanek, Klaus Berberich, and Gerhard Weikum. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artif. Intell.*, 194:28–61, jan 2013.

[27] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutiérrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, Axel-Cyrille Ngonga Ngomo, Axel Polleres, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan Sequeda, Steffen Staab, and Antoine Zimmermann. *Knowledge Graphs*. Synthesis Lectures on Data, Semantics, and Knowledge. Morgan & Claypool Publishers, 2021.

[28] Krzysztof Janowicz, Pascal Hitzler, Wenwen Li, Dean Rehberger, Mark Schildhauer, Rui Zhu, Cogan Shimizu, Colby K. Fisher, Ling Cai, Gengchen Mai, Joseph Zalewski, Lu Zhou, Shirly Stephen, Seila Gonzalez Estrecha, Bryce D. Mecum, Anna Lopez-Carr, Andrew Schroeder, Dave Smith, Dawn J. Wright, Sizhe Wang, Yuanyuan Tian, Zilong Liu, Meilin Shi, Anthony D'Onofrio, Zhining Gu, and Kitty Currier. Know, know where, knowwheregraph: A densely connected, cross-domain knowledge graph and geo-enrichment service stack for applications in environmental intelligence. *AI Mag.*, 43(1):30–39, 2022.

[29] Vidur Joshi, Matthew E. Peters, and Mark Hopkins. Extending a parser to distant domains using a few dozen partially annotated examples. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 1190–1199. Association for Computational Linguistics, 2018.

[30] Nikolaos Karalis, Georgios M. Mandilaras, and Manolis Koubarakis. Extending the YAGO2 knowledge graph with precise geospatial knowledge. In Chiara Ghidini, Olaf Hartig, Maria Maleshkova, Vojtech Svátek, Isabel F. Cruz, Aidan Hogan, Jie Song, Maxime Lefrançois, and Fabien Gandon, editors, *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part II*, volume 11779 of *Lecture Notes in Computer Science*, pages 181–197. Springer, 2019.

[31] Sergios-Anestis Kefalidis, Dharmen Punjani, Eleni Tsalapati, Konstantinos Plas, Mariangela Pollali, Michail Mitsios, Myrto Tsokanaridou, Manolis Koubarakis, and Pierre Maret. Benchmarking geospatial question answering engines using the dataset geoquestions1089. In Terry R. Payne, Valentina Presutti, Guilin Qi, María Poveda-Villalón, Giorgos Stoilos, Laura Hollink, Zoi Kaoudi, Gong Cheng, and Juanzi

Li, editors, *The Semantic Web - ISWC 2023 - 22nd International Semantic Web Conference, Athens, Greece, November 6-10, 2023, Proceedings, Part II*, volume 14266 of *Lecture Notes in Computer Science*, pages 266–284. Springer, 2023.

[32] Manolis Koubarakis and Kostis Kyzirakos. Modeling and querying metadata in the semantic sensor web: The model stRDF and the query language stSPARQL. In *Extended Semantic Web Conference*, pages 425–439. Springer, 2010.

[33] Kostis Kyzirakos, Manos Karpathiotakis, and Manolis Koubarakis. Strabon: A semantic geospatial DBMS. In Philippe Cudré-Mauroux, Jeff Heflin, Evren Sirin, Tania Tudorache, Jérôme Euzenat, Manfred Hauswirth, Josiane Xavier Parreira, Jim Hendler, Guus Schreiber, Abraham Bernstein, and Eva Blomqvist, editors, *The Semantic Web - ISWC 2012 - 11th International Semantic Web Conference, Boston, MA, USA, November 11-15, 2012, Proceedings, Part I*, volume 7649 of *Lecture Notes in Computer Science*, pages 295–311. Springer, 2012.

[34] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. In Kevin Knight, Ani Nenkova, and Owen Rambow, editors, *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 260–270. The Association for Computational Linguistics, 2016.

[35] Jae-Gil Lee and Minseo Kang. Geospatial big data: Challenges and opportunities. *Big Data Res.*, 2(2):74–81, 2015.

[36] Jens Lehmann, Sébastien Ferré, and Sahar Vahdati. Language models as controlled natural language semantic parsers for knowledge graph question answering. In Kobi Gal, Ann Nowé, Grzegorz J. Nalepa, Roy Fairstein, and Roxana Radulescu, editors, *ECAI 2023 - 26th European Conference on Artificial Intelligence, September 30 - October 4, 2023, Kraków, Poland - Including 12th Conference on Prestigious Applications of Intelligent Systems (PAIS 2023)*, volume 372 of *Frontiers in Artificial Intelligence and Applications*, pages 1348–1356. IOS Press, 2023.

[37] Haonan Li, Ehsan Hamzei, Ivan Majic, Hua Hua, Jochen Renz, Martin Tomko, Maria Vasardani, Stephan Winter, and Timothy Baldwin. Neural factoid geospatial question answering. *Journal of Spatial Information Science*, 23:65–90, 2021.

[38] Denis Lukovnikov, Asja Fischer, and Jens Lehmann. Pretrained transformers for simple question answering over knowledge graphs. In *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part I*, volume 11778 of *Lecture Notes in Computer Science*, pages 470–486. Springer, 2019.

[39] Farzaneh Mahdisoltani, Joanna Asia Biega, and Fabian M. Suchanek. Yago3: A knowledge base from multilingual wikipedias. In *Conference on Innovative Data Systems Research*, 2015.

[40] Gengchen Mai, Krzysztof Janowicz, Rui Zhu, Ling Cai, and Ni Lao. Geographic question answering: Challenges, uniqueness, classification, and future directions. *CoRR*, abs/2105.09392, 2021.

[41] Gengchen Mai, Bo Yan, Krzysztof Janowicz, and Rui Zhu. Relaxing unanswerable geographic questions using A spatially explicit knowledge graph embedding model. In Phaedon C. Kyriakidis, Diofantos G. Hadjimitsis, Dimitrios Skarlatos, and Ali Mansourian, editors, *Geospatial Technologies for Local and Regional Development - Proceedings of the 22nd AGILE Conference on Geographic Information Science, Limassol, Cyprus, June 17-20, 2019*, Lecture Notes in Geoinformation and Cartography, pages 21–39. Springer, 2019.

[42] Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

[43] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60, 2014.

[44] Matthew Perry and John Herring. OGC GeoSPARQL - A Geographic Query Language for RDF Data. OGC Implementation Standard OGC 11-052r4, Open Geospatial Consortium, September 2012.

[45] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. MS MARCO: A human generated machine reading comprehension dataset. In Tarek Richard Besold, Antoine Bordes, Artur S. d'Avila Garcez, and Greg Wayne, editors, *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, December 9, 2016*, volume 1773 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2016.

[46] Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan T. McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. Universal dependencies v1: A multilingual treebank collection. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asunción Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation LREC 2016, Portorož, Slovenia, May 23-28, 2016*. European Language Resources Association (ELRA), 2016.

[47] OpenAI. Gpt-4 technical report, 2023.

[48] Dharmen Punjani, Markos Iliakis, Theodoros Stefou, Kuldeep Singh, Andreas Both, Manolis Koubarakis, Iosif Angelidis, Konstantina Bereta, Themis Beris, Dimitris Bilidas, Theofilos Ioannidis, Nikolaos Karalis, Christoph Lange, Despina-Athanasia Pantazi, Christos Papaloukas, and Georgios Stamoulis. Template-based question answering over linked geospatial data. *CoRR*, abs/2007.07060, 2020.

[49] Dharmen Punjani, K Singh, Andreas Both, Manolis Koubarakis, Ioannis Angelidis, Konstantina Bereta, Themis Beris, Dimitris Bilidas, T Ioannidis, Nikolaos Karalis, et al. Template-based question answering over linked geospatial data. In *Proceedings of the 12th Workshop on Geographic Information Retrieval*, pages 1–10, 2018.

[50] Punjani, D. et al. The question answering system geoqa2. In *2nd International Workshop on Geospatial Knowledge Graphs and GeoAI: Methods, Models, and Resources*, 2023.

[51] Spiros Skiadopoulos and Manolis Koubarakis. Composing cardinal direction relations. In Christian S. Jensen, Markus Schneider, Bernhard Seeger, and Vassilis J. Tsotras, editors, *Advances in Spatial and Temporal Databases, 7th International Symposium, SSTD 2001, Redondo Beach, CA, USA, July 12-15, 2001, Proceedings*, volume 2121 of *Lecture Notes in Computer Science*, pages 299–320. Springer, 2001.

[52] Christian Strobl. Dimensionally extended nine-intersection model ("de-9im"). 2008.

[53] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In Carey L. Williamson, Mary Ellen Zurko, Peter F. Patel-Schneider, and Prashant J. Shenoy, editors, *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, pages 697–706. ACM, 2007.

[54] Alon Talmor and Jonathan Berant. The web as a knowledge-base for answering complex questions. In Marilyn A. Walker, Heng Ji, and Amanda Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 641–651. Association for Computational Linguistics, 2018.

[55] Lappoon R. Tang and Raymond J. Mooney. Using multiple clause constructors in inductive logic programming for semantic parsing. In Luc De Raedt and Peter A. Flach, editors, *Machine Learning: EMCL 2001, 12th European Conference on Machine Learning, Freiburg, Germany, September 5-7, 2001, Proceedings*, volume 2167 of *Lecture Notes in Computer Science*, pages 466–477. Springer, 2001.

[56] Thomas Pellissier Tanon, Denny Vrandecic, Sebastian Schaffert, Thomas Steiner, and Lydia Pintscher. From freebase to wikidata: The great migration. In Jacqueline Bourdeau, Jim Hendler, Roger Nkambou, Ian Horrocks, and Ben Y. Zhao, editors, *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11 - 15, 2016*, pages 1419–1428. ACM, 2016.

[57] Thomas Pellissier Tanon, Gerhard Weikum, and Fabian M. Suchanek. Yago 4: A reason-able knowledge base. *The Semantic Web*, 12123:583 – 596, 2020.

[58] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev,

Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288, 2023.

[59] Priyansh Trivedi, Gaurav Maheshwari, Mohnish Dubey, and Jens Lehmann. Lc-quad: A corpus for complex question answering over knowledge graphs. In Claudia d'Amato, Miriam Fernández, Valentina A. M. Tamma, Freddy Lécué, Philippe Cudré-Mauroux, Juan F. Sequeda, Christoph Lange, and Jeff Heflin, editors, *The Semantic Web - ISWC 2017 - 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part II*, volume 10588 of *Lecture Notes in Computer Science*, pages 210–218. Springer, 2017.

[60] Priyansh Trivedi, Gaurav Maheshwari, Mohnish Dubey, and Jens Lehmann. Lc-quad: A corpus for complex question answering over knowledge graphs. In Claudia d'Amato, Miriam Fernández, Valentina A. M. Tamma, Freddy Lécué, Philippe Cudré-Mauroux, Juan F. Sequeda, Christoph Lange, and Jeff Heflin, editors, *The Semantic Web - ISWC 2017 - 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part II*, volume 10588 of *Lecture Notes in Computer Science*, pages 210–218. Springer, 2017.

[61] Ricardo Usbeck, Ria Hari Gusmita, Axel-Cyrille Ngonga Ngomo, and Muhammad Saleem. 9th challenge on question answering over linked data (QALD-9) (invited paper). In Key-Sun Choi, Luis Espinosa Anke, Thierry Declerck, Dagmar Gromann, Jin-Dong Kim, Axel-Cyrille Ngonga Ngomo, Muhammad Saleem, and Ricardo Usbeck, editors, *Joint proceedings of the 4th Workshop on Semantic Deep Learning (SemDeep-4) and NLIWoD4: Natural Language Interfaces for the Web of Data (NLIWOD-4) and 9th Question Answering over Linked Data challenge (QALD-9) co-located with 17th International Semantic Web Conference (ISWC 2018), Monterey, California, United States of America, October 8th - 9th, 2018*, volume 2241 of *CEUR Workshop Proceedings*, pages 58–64. CEUR-WS.org, 2018.

[62] Denny Vrandecic and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85, 2014.

[63] Gerhard Weikum, Xin Luna Dong, Simon Razniewski, and Fabian Suchanek. Machine knowledge: Creation and curation of comprehensive knowledge bases. *Foundations and Trends in Databases*, 10(2-4):108–490, 2021.

[64] David Wood, Markus Lanthaler, and Richard Cyganiak. RDF 1.1 concepts and abstract syntax, February 2014.

[65] H. Xu, E. Hamzei, E. Nyamsuren, H. Kruiger, S. Winter, M. Tomko, and S. Scheider. Extracting interrogative intents and concepts from geo-analytic questions. *AGILE: GIScience Series*, 1:23, 2020.

[66] Wen-tau Yih, Matthew Richardson, Christopher Meek, Ming-Wei Chang, and Jina Suh. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 2: Short Papers*. The Association for Computer Linguistics, 2016.

[67] Eman M. G. Younis, Christopher B. Jones, Vlad Tanasescu, and Alia I. Abdelmoty. Hybrid geo-spatial query methods on the semantic web with a spatially-enhanced index of DBpedia. In Ningchuan Xiao, Mei-Po Kwan, Michael F. Goodchild, and Shashi Shekhar, editors, *Geographic Information Science - 7th International Conference, GIScience 2012, Columbus, OH, USA, September 18-21, 2012. Proceedings*, volume 7478 of *Lecture Notes in Computer Science*, pages 340–353. Springer, 2012.

[68] Shuguang Zhu, Xiang Cheng, and Sen Su. Conversational semantic parsing over tables by decoupling and grouping actions. *Knowledge-Based Systems*, 204:106237, 2020.