Benchmarking Geospatial Question Answering Engines Using the Dataset GeoQuestions1089

Sergios-Anestis Kefalidis[1](✉), Dharmen Punjani[2], Eleni Tsalapati[1], Konstantinos Plas[1], Mariangela Pollali[1], Michail Mitsios[1], Myrto Tsokanaridou[1], Manolis Koubarakis[1], and Pierre Maret[2]

[1] Department of Informatics and Telecommunications, National and Kapodistrian University of Athens, Athens, Greece
{s.kefalidis,etsalapati,mpollali,kplas,cs2200011,mtsokanaridou, koubarak}@di.uoa.gr

[2] Université St. Monnet, St. Etienne, France
dharmen.punjani@gmail.com, pierre.maret@univ-st-etienne.fr

Abstract. We present the dataset GeoQuestions1089 for benchmarking geospatial question answering engines. GeoQuestions1089 is the largest such dataset available presently and it contains 1089 questions, their corresponding GeoSPARQL or SPARQL queries and their answers over the geospatial knowledge graph YAGO2geo. We use GeoQuestions1089 to evaluate the effectiveness and efficiency of geospatial question answering engines GeoQA2 (an extension of GeoQA developed by our group) and the system of Hamzei et al. (2021).

1 Introduction

Users are often interested in posing *geospatial questions* to search engines, question answering (QA) engines and chatbots. Examples of such geospatial questions are: “Which rivers cross London?”, “Is there a Levi’s store in Athens?” and “Which countries border Greece, have the euro as their currency and their population is greater than the population of Greece?”. In this paper, 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 [18,20]. 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, multipolygons etc.) since these are more appropriate for

This work was supported by the first call for H.F.R.I. Research Projects to support faculty members and researchers and the procurement of high-cost research equipment grant (HFRI-FM17-2351). It was also partially supported by the ESA project DA4DTE (subcontract 202320239), the Horizon 2020 project AI4Copernicus (GA No. 101016798) and the Marie Skłodowska-Curie project QuAre (GA No. 101032307).

T. R. Payne et al. (Eds.): ISWC 2023, LNCS 14266, pp. 266–284, 2023.
https://doi.org/10.1007/978-3-031-47243-5_15

modeling the geometries of features such as rivers, roads, countries etc. (as in Wikidata [29], YAGO2geo [14], WorldKG [4] and KnowWhereGraph [12]).

The development of the above geospatial KGs has given rise to *geospatial QA engines* for them. Examples of such systems are the GeoQA engine developed by our group [24,25] and the systems of [1,9,17,28,31]. To evaluate the effectiveness and efficiency of these engines, there is currently only one benchmark: the GeoQuestions201 dataset proposed by our group [25] and used in comparing GeoQA with the systems of [8,9] and [17]. In this paper we go beyond GeoQuestions201 and make the following *original contributions.*

We present the benchmark GeoQuestions1089, which contains 1089 triples of geospatial questions, their answers, and the respective SPARQL/GeoSPARQL queries. GeoQuestions1089 is currently the largest geospatial QA benchmark and it is made freely available to the research community[1]. 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 way, we contribute to a long-term research agenda towards QA systems with geospatial features.

We present the geospatial QA system GeoQA2 which is based on GeoQA [25] and its revised version [24]. GeoQA2 is available as open source[2]. It targets the union of the KG YAGO2 and the geospatial KG YAGO2geo, and improves GeoQA by having been optimized in various ways and being able to answer a greater variety of questions.

Using GeoQuestions1089, we evaluate the effectiveness and efficiency of geospatial QA engines GeoQA2 and the engine of Hamzei et al. [9] and find that although GeoQA2 emerges victorious, mainly because of its disambiguation component, neither engine is able to process complex questions caused by both a limited vocabulary of geospatial relations and a template-based approach to query generation. We stress here that the competitor engine of Hamzei et al. has been designed to target YAGO2geo and therefore cannot answer questions such as "What is the length of the Awali river?" because the entity `yago:Awali_(river)` appears in YAGO2 but not in YAGO2geo meaning that it is lacking detailed geospatial information which is expected by the query generator of the engine.

We show that the pre-computation and materialization of entailed, but not stored explicitly, topological relations between entities in geospatial KGs can lead to substantial savings in geospatial query processing time. We show experimentally that this can speed up question answering for both engines studied.

[1] https://github.com/AI-team-UoA/GeoQuestions1089.
[2] https://github.com/AI-team-UoA/GeoQA2.

2 Related Work

We survey the state of the art in geospatial QA engines and the only dataset that exists for their evaluation (GeoQuestions201). We also introduce the geospatial KG YAGO2geo since it is the only KG of interest to us in this paper.

YAGO2geo was developed by our group in [14]. It is based on the subset of YAGO2 [10] which includes only *geoentities* i.e., entities that have latitude/longitude co-ordinates associated with them (presumably, representing their center). YAGO2geo enriches the geospatial dimension of some of these geoentities with detailed geometries, namely lines, polygons and multi-polygons taken from official administrative datasets (for Greece, the United Kingdom and the Republic of Ireland) and the Global Administrative Areas dataset (GADM,[3]). Hence, YAGO2geo can be used to answer questions that could not be answered by YAGO2 because detailed administrative geospatial knowledge is required (e.g., "Which counties of England are crossed by river Thames?"). Additionally, for natural features such as lakes and rivers, the respective YAGO2 geoentities are enriched with detailed geometries from OpenStreetMap (OSM). Finally, YAGO2geo includes *new* geontities present in the above administrative datasets and OSM that were not present in YAGO2. In 2020, YAGO2geo was further extended with data of administrative divisions of the United States of America from the National Boundary Dataset[4]. YAGO2geo currently contains 703 thousand polygons and 3.8 million lines. YAGO2geo represents geographic knowledge by using the YAGO2 ontology, the GeoSPARQL ontology and the ontologies especially developed by the YAGO2geo developers for each dataset mentioned above.

The first QA engine over a KG has been a system for answering geospatial questions over DBpedia [31]. 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 [25] and its revised version [24]. 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 and the Frankenstein platform [27]. In addition to developing an engine, [25] proposed the dataset GeoQuestions201 for its evaluation. This dataset consists of 201 questions, their answers and the corresponding SPARQL or GeoSPARQL queries. The questions GeoQuestions201 have been categorized by [25] into

3 https://gadm.org/.
4 https://www.usgs.gov/.

seven categories. In this paper we develop a much larger dataset containing a larger variety of questions and we extend the categorization of [25] accordingly.

[17] use deep neural networks (LSTM networks and the large language model BERT) and a template-based approach like the one proposed by GeoQA for generating a GeoSPARQL query corresponding to an input geospatial question. They achieved better results than [25] in producing GeoSPARQL translations of input questions. However, they achieved worse results than [24] 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 [19].

Recently, [1] proposed a system for answering qualitative spatial questions based on deductive spatial reasoning. Initially, the system extracts toponyms and spatial relations from the question text using DeepPavlov [2]. Then, it creates triples based on these outputs and applies the crisp qualitative spatial reasoner of the SparQ toolbox [30]. A limitation of this approach is that it is focused on addressing only three types of spatial questions, two of those (Q-Type2, Q-Type3) are already addressed by GeoQA.

The most recent geospatial QA engine is that of Hamzei et al. [9], which presents an engine that extends the one originally presented in [8]. The system of Hamzei et al. will be discussed in more detail in Sect. 5.

3 The GeoQuestions1089 Dataset

The GeoQuestions1089 dataset consists of two parts, which we will refer to as GeoQuestions$_C$ (1017 questions) and GeoQuestions$_W$ (72 questions) both of which target the union of YAGO2 and YAGO2geo. GeoQuestions$_C$ is the union of the datasets GeoQuestions$_T$ and GeoQuestions$_F$.

To develop GeoQuestions$_T$, we asked each M.Sc. student of the 2020–2021 Knowledge Technologies course of our department to formulate 21 question-query-answers triples targeting YAGO2geo. We asked students to include in their questions one or more features and various kinds of geospatial relations: distance relations (e.g., near, at most 2km from), topological relations (e.g., in, borders, crosses) or cardinal directions (e.g., east of, northeast of). Also, they were asked to have questions for all four countries covered with official data by YAGO2geo: USA, Greece, United Kingdom and Ireland. Finally, one more constraint was that the generated GeoSPARQL queries for three of their questions should be with one, two and three aggregate functions, respectively. In at least one of these three cases, the students were asked to provide a question which can be mapped to an advanced GeoSPARQL expression like a nested query or a not-exists filter. In this way, we wanted to target questions that were more complex than the ones in GeoQuestions201. To obtain the answers, the students were asked to run their GeoSPARQL queries in a YAGO2geo endpoint that we provided. The questions gathered were factoid, simple/complex and, in some cases, with aggregations (e.g., counting), comparatives, or superlatives. The resulting dataset contained 615 questions targeting YAGO2geo.

To develop GeoQuestions$_F$, we asked third-year students of the 2020–2021 AI course in the same department to write 50 questions targeting the subset of OSM and the infoboxes of Wikipedia, imagining scenarios related to traveling or to generating geography questionnaires for students or TV games. The only constraint was that simple but also complex questions should be produced (examples of simple questions from GeoQuestions201 and complex questions from GeoQuestions$_T$ were given). In total, we gathered 9,335 questions. From this set, we randomly chose 1200 questions, for which we hired six M.Sc. students of the same course to clean them and translate them into SPARQL or stSPARQL/GeoSPARQL using YAGO2geo. Because this crowdsourcing effort was less restrictive than that of GeoQuestions$_T$, some questions didn't have answers in YAGO2geo alone. However, they could be answered using the union of YAGO2 and YAGO2geo KGs. After this, the students ran the queries in the YAGO2geo endpoint and stored the answers, when these existed. The resulting dataset contained 402 questions, 280 questions targeting YAGO2geo and 122 questions targeting the union of YAGO2 and YAGO2geo.

The dataset GeoQuestions$_C$ was checked by the authors of this paper. Each question (query) was checked both grammatically and syntactically, using Grammarly[5] and QuillBot[6]. When necessary, and because some queries required exorbitant compute resources to be answered in reasonable time, we rerun the queries against the endpoint using materialized relations (see Sect. 6). 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. In GeoQuestions$_W$, we include the original, incorrect questions with the end goal of benchmarking how capable QA engines are at handling incorrect input.

Extending the categorization of [25], we can see that the questions of dataset GeoQuestions1089 fall under the following categories:[7]

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. Google and Bing both can also answer such questions precisely.
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 2 km from"). In GeoQA2, these questions are answered by querying YAGO2geo using the detailed geometries of features for evaluating the geospatial relation of the question. Google and Bing both cannot answer such factoid questions, but can only return a list of relevant Web

[5] https://www.grammarly.com/.
[6] https://quillbot.com/.
[7] 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.

pages. The recently deployed chat feature of Bing gives more information by saying that "Liverpool ... is located on the eastern side of the Irish Sea".

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). In GeoQA2, these questions can be answered by using the detailed geometries of features from YAGO2geo for evaluating the geospatial relations. Google and Bing can also answer such questions precisely in many but not all cases (e.g., they can answer the first question but not the second).
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?"*. Google and Bing both cannot answer such questions precisely.
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 m?"*. Google and Bing both give links to pages with lists of mountains of Scotland with their height.
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?"*. Google and Bing both can answer precisely the second question but not the first. For the first question both return pages with lists of lakes in Monaghan. The chat component of Bing attempts to answer the first question but fails.
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?"*. Google answers the first question accurately but Bing does not and instead gives a list of links to related pages. The chat component of Bing can answer the first question precisely (Crete). Both engines cannot answer the second question; they only give links to relevant Web pages. The chat component of Bing is able to answer the second question precisely. (Corsica is 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?"*. Google can answer the first question precisely but Bing fails and returns a list of best hospitals in cities of the UK. Both engines cannot answer the second question.

Table 1 describes GeoQuestions1089 giving numbers per type of question.

Table 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	

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 [9] and confirmed by our own experience while developing GeoQA2.

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.

4 The QA Engine GeoQA2

GeoQA2 takes as input a question in English and the union of YAGO2 and YAGO2geo KGs, and produces a set of answers. QA is 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.

The differences between GeoQA [25] and GeoQA2 can be summarized as follows. GeoQA was targeting DBpedia, GADM, and OSM for the United Kingdom and Ireland. GeoQA2 targets the union of YAGO2 and YAGO2geo. Most importantly, GeoQA2 can answer a greater variety of questions, including questions with quantities, aggregates, superlatives and comparatives thanks to the use of constituency parsing and the development of additional templates.

In Fig. 1 we present the GeoQA2 pipeline which contains the following components: dependency parse tree generator, concept identifier, instance identifier, geospatial relation identifier, property identifier and query generator. The functionality of these components will be discussed below using the question "Is the largest island in the United Kingdom larger than Crete by population?".

The *dependency parse tree generator* carries out part-of-speech (POS) tagging and generates a dependency parse tree for the input question using the Stanford CoreNLP toolkit [21].

The *concept identifier* 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

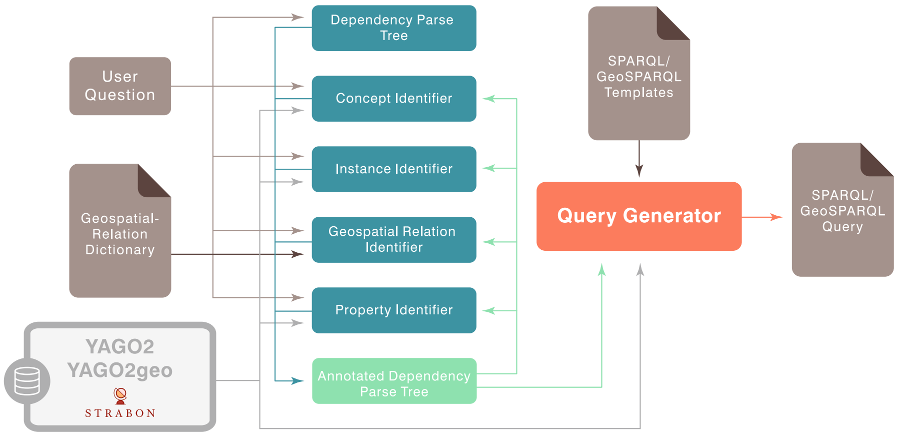

Fig. 1. The conceptual architecture of the GeoQA2 engine

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 *instance identifier* identifies the *features* (*instances*) present in the input question (e.g., "United Kingdom" and "Crete"). The features are identified by the elements of the question that are tagged as proper nouns (POS tags NN, NNS and NNP) during dependency parsing. Then, these elements are mapped to YAGO2geo resources (e.g., `yago:United_Kingdom` and `yago:Crete`) using the TagMeDisambiguate tool [6]. In previous work [24] we tested a set of well-known tools on the task of named entity recognition and disambiguation for geographic entities in the dataset GeoQuestions201. The tool of choice for GeoQA2 is TagMeDisambiguate, since it gave the best results in that study. The instance identifier also queries YAGO2geo to disambiguate the instances that are contained in YAGO2geo, but not in YAGO2.

The *geospatial relation identifier* first identifies the geospatial relations (e.g., "in") in the input question based on the POS tags VB, IN, VP, VBP and VBZ generated during dependency parsing. Then, it maps them to the respective spatial function of the GeoSPARQL or stSPARQL vocabulary (e.g., `geof:within`) according to a mapping between geospatial relations and stSPARQL/GeoSPARQL functions provided by a dictionary.

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`. The attributes in the input question are identified based on the POS tags NN, JJ, NNP and NP generated by the dependency parsing process and the concepts/instances identified by earlier steps.

The *query generator* produces the GeoSPARQL query corresponding to the input question using handcrafted query templates and the annotated parse tree. GeoQA2 has 10 templates while GeoQA [25] had 5 templates. For questions of types G, H and I (see Sect. 3), the query generator also constructs the constituency parse tree of the input question and uses it to modify the templates to support aggregates and superlatives (e.g., "largest").

5 The QA Engine of Hamzei et al.

Like GeoQA2, the engine of Hamzei et al. [9] 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 [9]. The engine uses a four-step workflow consisting of encoding extraction, grammatical parsing, intermediate representation generation and GeoSPARQL query generation. These steps are briefly described below using the question "How many pharmacies are in 200 m 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 [7]. These encoding classes offer a rich representational framework which 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 m radius of" and "in") etc. are mentioned.[8] 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 [13,16] and the large language model BERT [3].

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").

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

$$\begin{aligned}&Count(x) : Place(High\ Street) \wedge Place(Oxford) \wedge Pharmacy(x)\\&\qquad \wedge InRadiusOf(x, HighStreet, 200meter) \wedge In(HighStreet, Oxford)\end{aligned}$$

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. Instead of doing place-name disambiguation, this step relies on string similarity search using an Apache Solr server for identifying instances. The

[8] The conceptual framework of Hamzei et al. [9] 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.

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[9] while a demo is available at[10].

6 Improving the Performance of Geospatial QA Engines

One of the key challenges faced by GeoQA2 and the system of Hamzei et al., but also by any other geospatial query answering system, is the large number of geometric calculations that it has to perform, which often leads to very long response times. For instance, checking whether a geometry is within a large administrative area with complex borders is computationally a very challenging task. Hence, to improve the time performance of the two engines discussed previously, we *pre-computed and materialized* certain relations between entities in the YAGO2geo KG that change infrequently. Materialization is an optimization technique that has been widely used (e.g., see [26] for the case of geospatial KGs).

During the evaluation process of Sect. 7, we observed that topological geospatial relations "within", "crosses", "intersects", "touches", "overlaps", "covers" and "equals" require expensive computations, while "near", "north", "south", "east" and "west" are easily computed. Hence, we decided to materialized the above costly topological relations. This approach is particularly beneficial, since, as shown below, it greatly boosts the performance of computationally demanding queries. To facilitate the evaluation of the QA engines using the materialized relations and to maintain the integrity of the GeoSPARQL queries in GeoQuestions1089, we developed a transpiler to automatically rewrite queries from GeoSPARQL to SPARQL.

One of the major concerns related to materialization is the size of resulting KG, and the overhead that this can cause to its processing. Overall, the materialized version of YAGO2geo had 17,210,176 more triples, which in terms of system memory, amounts to about 3GB and 10.21% increased in total size, but as shown below, it does not affect the performance of the QA system negatively. The time required to calculate the implied geospatial relation was close to 5 d, which can be considered negligible, as it happens offline, and it is being repeated infrequently (only when the KG changes).

The calculation of the implied relations was facilitated by utilizing a distributed implementation of the algorithm GIA.nt [22], implemented in the system DS-JedAI [23][11]. GIA.nt is a holistic geospatial interlinking algorithm that uses the DE-9IM topological model in order to discover all the topological relations between the geometries of two geospatial datasets. It employs a series

[9] https://github.com/hamzeiehsan/Questions-To-GeoSPARQL.
[10] https://tomko.org/demo/.
[11] https://github.com/GiorgosMandi/DS-JedAI.

of efficient techniques such as the dynamic space tiling, the minimum bounding rectangle intersection and the reference point technique to filter out redundant geometry verifications, and, therefore, to significantly reduce the default quadratic complexity of geospatial interlinking.

The geospatial part of YAGO2geo includes the datasets OS, OSI, OSNI, NBD, GAG, GADM and OSM. We used GIA.nt to discover the geospatial relations among the entities within the same dataset as well as all the other aforementioned geospatial datasets (Table 2).

Table 2. Number of materialized relations in YAGO2geo

Dataset	Number of discovered relations
OS	1,242,358
OSI	2,743,769
OSM	5,395,399
OSNI	522,221
GADM	2,773,983
GAG	25,695
NBD	4,506,751

Table 3. Average time performance of queries in YAGO2geo

Question Category	Number of questions	Average time	Average time materialized
B	12	< 1	<0.1
C	10	378	<1
D	13	**27237**	**<2**
E	9	262	<2
F	8	179	<1
G	3	100	<2
H	3	68	<2
I	2	245	<2

Table 4. Time performance in seconds of selected queries in YAGO2geo with and without materialized relations

Id	Question	Category	Execution time simple	Execution time materialized
1	What is the number of parks located within cities?	D	**70129**	**<1**
2	How many nature reserves contain forests?	D	11700	<1
3	Which counties border Donegal county?	C	1747	<1
4	Which is the largest county in the UK?	C	1609	<1
5	Which localities are south of lakes in County Cavan?	E	977	<1
6	Which forests are entirely within an Irish Barony?	E	501	<1
7	Which municipality in Crete region have population over 20000?	F	482	<1

We ran the experiments on a machine with the following specifications: Intel Xeon E5-4603 v2 @2.20 GHz, 128 Gb DDR3 RAM, 1.6 TB hard disk. YAGO2geo and the materialized relations were stored and configured in Strabon [15]. We selected Strabon due to our group's familiarity with its development, but also because it is one of the most efficient centralized geospatial RDF stores [11].

For the time-performance comparison of queries run against YAGO2geo with and without the materialization, we used 60 geospatial questions of GeoQuestions1089 that contain materialized relations. In Table 3 the average time difference to execute the GeoSPARQL and SPARQL queries using materialized relations is presented. The third and fourth column of this table show the average execution times for the queries in the second column. In Table 4 we display some of the queries for which we have significant time improvements. The reason for the slower execution of queries in category D is that they involve comparing complex geometries e.g., the geometries of all lakes and cities in a country. Hence, Strabon performs constly spatial joins involving the geometries that belong to these classes to get the final result. For the simple GeoSPARQL queries of category B, on the other hand, which needs to calculate a spatial relation only between two given geometries, it suffices to identify the triples representing this information, which has already been computed off-line.

7 Evaluation

In this section, we use the dataset GeoQuestions1089 to benchmark the QA engines GeoQA2 and the one by Hamzei et al. [9]. The exact versions of the engines used are available in the repository of GeoQuestions1089. 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).

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 as mentioned in Sect. 6, and it is then sent to a geospatial RDF store that executes the query over our knowledge graph. The result is compared to the gold result included in GeoQuestions1089. 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 F-score because the correct number of returned answers/entities for each query varies greatly, which biases the metric towards certain kinds of questions.

Evaluating GeoQA2. To evaluate GeoQA2 we set up three Strabon 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 5 and 6 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. Finally, the column "Correct Answers*" shows the same percentage computed over the set of questions for which a query was generated.

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.

Table 5. Evaluation of GeoQA2 over GeoQuestions$_C$.

Category	Generated Queries	Correct Answers	Correct Answers*
A	84%	47.42%	56.45%
B	76.25%	58.99%	77.35%
C	79.21%	44.38%	56.02%
D	56%	12%	21.42%
E	80%	31.85%	39.81%
F	66.66%	16.66%	25%
G	74.13%	32.18%	43.41%
H	71.12%	26.05%	36.63%
I	84%	20%	23.80%
Total	76.99%	38.54%	50.06%

Table 6. Evaluation of GeoQA2 over GeoQuestions$_W$.

Category	Generate Questions	Correct Answers	Correct Answers*
A	82%	47.05%	57.14%
B	81.81%	54.54%	66.66%
C	85.71%	57.14%	66.66%
D	50%	33%	66.66%
E	88%	0.00%	0.00%
F	36.36%	0.00%	0%
G	50.00%	0.00%	0.00%
H	100.00%	0.00%	0.00%
I	50%	50%	100.00%
Total	72.22%	34.72%	48.07%

Evaluating the System of Hamzei et al. The engine of Hamzei et al. [9] 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. Hamzei et al. [9] 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.

Table 7. Evaluation of the system of Hamzei et al. [9] 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	Correct Answers*	Generated Queries	Correct Answers	Correct Answers*
Type-A	89.71%	10.85%	12.10%	88.88%	12.50%	14.06%
Type-B	95.68%	53.23%	55.63%	95.52%	55.22%	57.81%
Type-C	97.75%	30.33%	31.03%	97.41%	32.90%	33.77%
Type-D	100%	12%	12.00%	100%	12%	12.00%
Type-E	99.25%	7.40%	7.46%	99.25%	7.46%	7.51%
Type-F	79.16%	4.10%	5%	76.19%	4.76%	6%
Type-G	98.27%	11.49%	11.69%	97.94%	13.01%	13.28%
Type-H	97.18%	7.74%	7.97%	96.49%	7.89%	8.18%
Type-I	92%	0%	0.00%	95%	0%	0.00%
Total	95.77%	18.97%	19.81%	95.53%	20.67%	21.63%

In a similar vein to the evaluation of GeoQA2, the generated queries of the engine are 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 and 8 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 GeoQA. 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 Hamzei et al. [9] 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 [9] is the lack of a dedicated disambiguation step. This leads to answers that contain numerous irrelevant results, i.e.,

Table 8. Evaluation of the system of Hamzei et al. [9] over GeoQuestions$_W$

Category	GeoQuestions$_W$		
	Generated Queries	Correct Answers	Correct Answers*
A	88.23%	17.64%	20.00%
B	100.00%	54.54%	54.54%
C	100.00%	35.71%	35.71%
D	100.00%	0.00%	0.00%
E	87.50%	0.00%	0.00%
F	90.90%	0.00%	0.00%
G	100.00%	0.00%	0.00%
H	100.00%	0.00%	0.00%
I	100.00%	0.00%	0.00%
Total	94.44%	19.44%	20.58%

the system is lacking precision. The other significant weakness is the rule-based approach to query generation that is unable to deal with complex queries.

Engine Comparison. The results of our evaluation show that GeoQA2 significantly outperforms the QA engine of [9] 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 that uses a set of predefined templates that are filled in with instances and concepts to generate the final GeoSPARQL queries, although the engine of [9] 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.

8 Conclusions and Future Work

We presented the dataset GeoQuestions1089 and evaluated the QA engines GeoQA2 and Hamzei et al. [9] using it. We plan to extend the dataset by utilizing semi-automatic techniques as it has been done e.g., in LC-QuAD 2.0 [5]. This will allow us to train geospatial QA engines using deep learning techniques with the hope that they will be more effective than the ones evaluated in this paper.

References

1. Beydokhti, M.K., Duckham, M., Tao, Y., Vasardani, M., Griffin, A.L.: Qualitative spatial reasoning over questions (short paper). In: Ishikawa, T., Fabrikant, S.I., Winter, S. (eds.) Proceedings of the 15th International Conference on Spatial Information Theory, COSIT 2022, 5–9 September 2022, Kobe. LIPIcs, vol. 240, pp. 18:1–18:7. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2022). https://doi.org/10.4230/LIPIcs.COSIT.2022.18
2. Burtsev, M., et al.: Deeppavlov: open-source library for dialogue systems. ACL (4), 122–127 (2018). https://aclanthology.info/papers/P18-4021/p18-4021
3. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: Burstein, J., Doran, C., Solorio, T. (eds.) Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, 2–7 June 2019, Volume 1 (Long and Short Papers). pp. 4171–4186. Association for Computational Linguistics (2019). https://doi.org/10.18653/v1/n19-1423
4. Dsouza, A., Tempelmeier, N., Yu, R., Gottschalk, S., Demidova, E.: WorldKG: A world-scale geographic knowledge graph. In: CIKM '21: The 30th ACM International Conference on Information and Knowledge Management, Virtual Event, Queensland, 1–5 November 2021, pp. 4475–4484. ACM (2021)
5. Dubey, M., Banerjee, D., Abdelkawi, A., Lehmann, J.: LC-QuAD 2.0: a large dataset for complex question answering over wikidata and DBpedia. In: ISWC (2019)
6. Ferragina, P., Scaiella, U.: TAGME: on-the-fly annotation of short text fragments (by wikipedia entities). In: Huang, J.X., Koudas, N., Jones, G.J.F., Wu, X., Collins-Thompson, K., An, A. (eds.) Proceedings of the 19th ACM Conference on Information and Knowledge Management (CIKM 2010), Toronto, 26–30 October 2010, pp. 1625–1628. ACM (2010). https://doi.org/10.1145/1871437.1871689
7. Hamzei, E., Li, H., Vasardani, M., Baldwin, T., Winter, S., Tomko, M.: Place questions and human-generated answers: a data analysis approach. In: Kyriakidis, P., Hadjimitsis, D., Skarlatos, D., Mansourian, A. (eds.) AGILE 2019. LNGC, pp. 3–19. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-14745-7_1
8. Hamzei, E.: Place-related question answering: from questions to relevant answers. Ph.D. thesis (2021)
9. Hamzei, E., Tomko, M., Winter, S.: Translating place-related questions to GeoSPARQL queries. In: Proceedings of the Web Conference (WWW) (2022)
10. Hoffart, J., Suchanek, F.M., Berberich, K., Weikum, G.: Yago2: a spatially and temporally enhanced knowledge base from wikipedia. Artif. Intell. **194**, 28–61 (2013)
11. Ioannidis, T., Garbis, G., Kyzirakos, K., Bereta, K., Koubarakis, M.: Evaluating geospatial RDF stores using the benchmark geographica 2. J. Data Semant. **10**(3–4), 189–228 (2021). https://doi.org/10.1007/s13740-021-00118-x
12. Janowicz, K., et al.: 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). https://doi.org/10.1609/aimag.v43i1.19120
13. Joshi, V., Peters, M.E., Hopkins, M.: Extending a parser to distant domains using a few dozen partially annotated examples. In: Gurevych, I., Miyao, Y. (eds.) Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, 15–20 July 2018, Volume 1: Long Papers, pp. 1190–1199.

Association for Computational Linguistics (2018). https://doi.org/10.18653/v1/P18-1110

14. Karalis, N., Mandilaras, G., Koubarakis, M.: Extending the YAGO2 knowledge graph with precise geospatial knowledge. In: Ghidini, C., et al. (eds.) ISWC 2019. LNCS, vol. 11779, pp. 181–197. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30796-7_12
15. Kyzirakos, K., Karpathiotakis, M., Koubarakis, M.: Strabon: A semantic geospatial DBMS. In: Cudré-Mauroux, P., et al. (eds.) ISWC 2012. LNCS, vol. 7649, pp. 295–311. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-35176-1_19
16. Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., Dyer, C.: Neural architectures for named entity recognition. In: Knight, K., Nenkova, A., Rambow, O. (eds.) NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego, 12–17 June 2016, pp. 260–270. The Association for Computational Linguistics (2016). https://doi.org/10.18653/v1/n16-1030 https://doi.org/10.18653/v1/n16-1030
17. Li, H., et al.: Neural factoid geospatial question answering. J. Spatial Inf. Sci. **23**, 65–90 (2021)
18. Longley, P.A., Goodchild, M.F., Maguire, D.J., Rhind, D.W.: Geographic Information Science and Systems, 4th edn. John Wiley and Sons (2015)
19. Lukovnikov, D., Fischer, A., Lehmann, J.: Pretrained transformers for simple question answering over knowledge graphs. In: Ghidini, C., et al. (eds.) ISWC 2019. LNCS, vol. 11778, pp. 470–486. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30793-6_27
20. Koubarakis, M. (ed.): Geospatial Data Science: A Hands-on Approach Based on Geospatial Technologies. ACM Books (2023)
21. Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., McClosky, D.: The Stanford CoreNLP natural language processing toolkit. In: Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pp. 55–60. Association for Computational Linguistics, Baltimore (2014). https://doi.org/10.3115/v1/P14-5010 https://aclanthology.org/P14-5010
22. Papadakis, G., Mandilaras, G.M., Mamoulis, N., Koubarakis, M.: Progressive, holistic geospatial interlinking. In: Leskovec, J., Grobelnik, M., Najork, M., Tang, J., Zia, L. (eds.) The Web Conference 2021, Virtual Event (WWW 2021)/Ljubljana, 19–23 April 2021, pp. 833–844. ACM/IW3C2 (2021). https://doi.org/10.1145/3442381.3449850
23. Papamichalopoulos, M., Papadakis, G., Mandilaras, G., Siampou, M.D., Mamoulis, N., Koubarakis, M.: Three-dimensional geospatial interlinking with jedai-spatial (2022). https://arxiv.org/pdf/2205.01905.pdf
24. Punjani, D., et al.: Template-based question answering over linked geospatial data. arXiv preprint arXiv:2007.07060 (2020)
25. Punjani, D., et al.: Template-based question answering over linked geospatial data. In: Purves, R.S., Jones, C.B. (eds.) Proceedings of the 12th Workshop on Geographic Information Retrieval, GIR@SIGSPATIAL 2018, Seattle, 6 November 2018, pp. 7:1–7:10. ACM (2018). https://doi.org/10.1145/3281354.3281362
26. Regalia, B.D.: Computational Time and Space Tradeoffs in Geo Knowledge Graphs. Ph.D. thesis, UC Santa Barbara (2020)
27. Singh, K., et al.: Why reinvent the wheel: let's build question answering systems together. In: Champin, P., Gandon, F.L., Lalmas, M., Ipeirotis, P.G. (eds.) Proceedings of the 2018 World Wide Web Conference on World Wide Web (WWW

2018), Lyon, 23–27 April 2018, pp. 1247–1256. ACM (2018). https://doi.org/10.1145/3178876.3186023
28. Stoilos, G., Papasarantopoulos, N., Vougiouklis, P., Bansky, P.: Type linking for query understanding and semantic search. In: Zhang, A., Rangwala, H. (eds.) The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2022), Washington, 14–18 August 2022, pp. 3931–3940. ACM (2022). https://doi.org/10.1145/3534678.3539067
29. Vrandecic, D., Krötzsch, M.: Wikidata: a free collaborative knowledgebase. Commun. ACM **57**(10), 78–85 (2014). https://doi.org/10.1145/2629489
30. Wallgrün, J.O., Frommberger, L., Wolter, D., Dylla, F., Freksa, C.: Qualitative spatial representation and reasoning in the SparQ-toolbox. In: Barkowsky, T., Knauff, M., Ligozat, G., Montello, D.R. (eds.) Spatial Cognition 2006. LNCS (LNAI), vol. 4387, pp. 39–58. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-75666-8_3
31. Younis, E.M.G., Jones, C.B., Tanasescu, V., Abdelmoty, A.I.: Hybrid geo-spatial query methods on the semantic web with a spatially-enhanced index of DBpedia. In: Xiao, N., Kwan, M.-P., Goodchild, M.F., Shekhar, S. (eds.) GIScience 2012. LNCS, vol. 7478, pp. 340–353. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33024-7_25