TERRAQ: SPATIOTEMPORAL QUESTION-ANSWERING ON SATELLITE IMAGE ARCHIVES

Sergios-Anestis Kefalidis[1], Konstantinos Plas[1,2], Manolis Koubarakis[1,2]
[1]*Dept. of Informatics and Telecommunications*, *National and Kapodistrian University of Athens*, Athens, Greece
skefalidis@di.uoa.gr, kplas@di.uoa.gr, koubarak@di.uoa.gr
[2]*Archimedes, Athena Research Center*, Greece

***Abstract*—TerraQ is a spatiotemporal question-answering engine for satellite image archives. It is a natural language processing system that is built to process requests for satellite images satisfying certain criteria. The requests can refer to image metadata and entities from a specialized knowledge base (e.g., the Emilia-Romagna region). With it, users can make requests like "Give me a hundred images of rivers near ports in France, with less than 20% snow coverage and more than 10% cloud coverage", thus making Earth Observation data more easily accessible, in-line with the current landscape of digital assistants.**

***Index Terms*—Question-Answering, Knowledge Graph, Geospatial, Temporal, Earth Observation, SPARQL**

I. Introduction

The field of Natural Language Processing is undergoing major advancements caused by the rapid development of Language Models [1–3]. An outcome of this development is the proliferation of digital assistants and natural language interfaces for all manners of systems and knowledge repositories (e.g., Alexa from Amazon, Siri from Apple, ChatGPT from OpenAI, and Claude from Anthropic). The resulting increase in accessibility enables non-technical users to intuitively interact with computer systems and access high-quality information, while also improving efficiency for expert users. In this climate, the task of Knowledge-Graph Question-Answering (QA), also known as Text-to-SPARQL, is as relevant as ever [4, 5].

QA systems take as input queries in natural language and generate semantically equivalent SPARQL queries over a specific knowledge graph (KG). These SPARQL queries are subsequently executed on an RDF store, which in turn returns the answer. The answer can be i) directly presented to the user, ii) integrated into a larger system [6], iii) used as part of a Retrieval-Augmented Generation [7] pipeline, as is the case with digital assistants that rely on knowledge grounding.

In our work, we are developing TerraQ a spatiotemporal QA engine for satellite image archives. User requests can refer to image metadata and geographic entities (e.g., the Loch Ness Lake or the city of Munich) both of which are included in the target knowledge graph. For example, "Show me images of Athens with VV polarization". The goal of our research is to make Earth Observation data archives accessible via natural language, to the benefit of both novice and expert users.

TerraQ belongs to the same family of engines as GeoQA2 [8] and EarthQA [9]. GeoQA2 is a geospatial QA engine, and EarthQA is a satellite-image archive QA engine that reuses some components of GeoQA2 while also adding some additional specialized components. In comparison to these two systems, TerraQ has a number of advantages. First, it does away with template-based query generation. As a result, it is able to answer a wider array of questions and provides better accuracy. Second, TerraQ targets a purpose-built KG with high-quality geospatial information, allowing for more fine-grained results, which was one of the limitations of the original EarthQA paper. Third, unlike EarthQA, the engine does not use specialized components. All thematic information can be integrated into the core engine architecture, making the engine easier to adapt to different domains.

In this paper, we make the following original contributions:

1) We present a specialized knowledge graph that interlinks geospatial information about natural features and administrative divisions with satellite image metadata. Knowledge graph resources are integrated into a hierarchical structure, making it easier to expand with additional geospatial information or thematic knowledge.
2) We develop TerraQ, a spatiotemporal QA engine for image archives. The engine is able to answer simple and complex queries both reliably and quickly in dynamic fashion, while also avoiding the use of query-templates or computationally demanding neural models.

The first version of TerraQ has been developed in the context of the European Space Agency project *DA4DTE: Demonstrator Precursor Digital Assistant Interface for Digital Twin Earth*[1]. The version of TerraQ presented in this paper has been developed in the *FAIR2Adapt*[2] project, and a demo is available publicly at http://terraq.di.uoa.gr/.

This work was partially supported by the Horizon Europe project FAIR2Adapt (GA no. 101188256), the ESA project DA4DTE (subcontract 202320239), and the "NIK. D. XRYSOVERGI" scholarship of the Greek State Scholarship Foundation. This work has also been partially supported by project MIS 5154714 of the National Recovery and Resilience Plan Greece 2.0 funded by the European Union under the NextGenerationEU Program.

[1]http://da4dte.e-geos.earth/
[2]https://fair2adapt-eosc.eu/

II. Knowledge Graph

To provide TerraQ with a powerful geospatial knowledge base, we compiled information from various sources and combined them under a common KG. Our aim was to create a polymorphic database of spatial resources, by integrating natural features and administrative geo-entities under a compact, non-complex ontology, that better facilitates the task of geospatial question answering. The aforementioned facts were collected from the following sources:

- GADM: We collected geospatial features from the Global Administrative Areas (https://gadm.org/) dataset, focusing on features located in Europe. Utilized features from this dataset include countries, cities, regional units, and national administrative divisions.
- Rivers, Points of Interest, and Ports: This dataset is provided by our partners in the DA4DTE project e-GEOS (https://www.e-geos.it/) and includes spatial characteristics for various features within these categories. In addition to spatial data, the dataset also contains comprehensive metadata for each feature.
- Sentinel-1 Images: We incorporated Sentinel-1 satellite image data, including metadata about the images and the satellite's location at the time each image was captured. Links to the images are stored in the knowledge graph, ensuring that they are easily accessible by the Question-Answering engine and external sources. Sentinel-1 images were collected for the years 2020 and 2021.
- Sentinel-2 Images: Similarly to the Sentinel-1 satellite images, we included image links and metadata from Sentinel-2 image collections to enhance the information available in our data model. Sentinel-2 images were collected for the years 2020, 2021 and 2022.
- Sea sectors: A collection of sea sectors covering global water spaces and oceans A total of 101 sea sectors along with their polygons were integrated into the knowledge graph of the Marine Regions [10] data source.

Ontology. We built our ontology on top of well-known and standardized ontologies, namely the YAGO2geo [11] ontology and the GeoSPARQL [12] ontology. The main class of the knowledge graph is named Feature. The Feature class is extended by various subclasses that represent the knowledge provided by the various datasets that we examined previously, namely, rivers, ports, pois (points of interest), Sentinel-1 and Sentinel-2 images and GADM geoentities.

Translation of named location labels to English. While integrating our data, we noticed that many geospatial features were named only in their original languages, with no English labels provided. For instance, the city of Rome was listed only as "Roma" in Italian within the metadata. To provide a consistent framework and to simplify the task of recognizing labels for our engine, we implemented a comprehensive translation pipeline. Using the Mistral-7b LLM [3], we were able to identify and translate a total of 56657 labels from various European languages with high accuracy.

III. The TerraQ Engine

TerraQ consists of a number of components, each of which performs a specific task. Information is propagated from one component to the next. This pipeline is split in four distinct conceptual steps.

First, the WHERE clause is generated by combining basic SPARQL/GeoSPARQL building blocks. This is subsequently passed as additional input to the components responsible for the generation of the SELECT/ASK clause. When both clauses have been constructed, the query generator merges them and makes any necessary additions to construct a complete SPARQL query. In the last step, the query is rewritten to make use of materialized geospatial relations.

The complete architecture of TerraQ is shown in Figure 1. Below, we present the functionality of the system in detail. As our running example we use the request "Show me all images taken in January 2021 with rivers less than 2km away from towns and forests in the Emilia Romagna region, having cloud coverage less than 10%".

Dependency Parse Tree Generator. This module generates a dependency parse tree of the input question using Stanford-CoreNLP [13]. The dependency parse tree is used to identify and store information.

Instance Identifier. This module does named-entity recognition and disambiguation. In the example question, it identifies the entity "Emilia Romagna" and maps it to the resource *yago:Emilia_(region_of_Italy)* in the KG. The mapping to the KG resource happens in two steps. First, WAT [14] links the named entity to a Wikipedia page. Subsequently, the component searches the Knowledge Graph for the resource that best matches the entity returned by WAT. In addition to identifying the instance, this component is responsible for creating the block that will be used in the WHERE clause for the identified instance. The generated block is the following:

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

Concept Identifier. This module identifies and maps concepts present in the input question to the appropriate resource of the KG ontology. For instance, from the example question, it will identify and map the concepts *Image, River, Town, Forest.* The mapping is done using a class label dictionary and string similarity based on n-grams. Additionally, this component is responsible for creating the block that will be used in the WHERE clause for the identified concepts:

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

At the end of the concept identification stage, and after all instances have been identified, we employ a heuristic of consolidation between concepts and instances. Concepts and instances that are not separated by any token are consolidated to reduce the complexity of the generated WHERE-clause and help the query generator produce a correct query. For example, in the question "Where is the Tagus river located?" only the Instance of Tagus is kept and the river concept is consolidated into it.

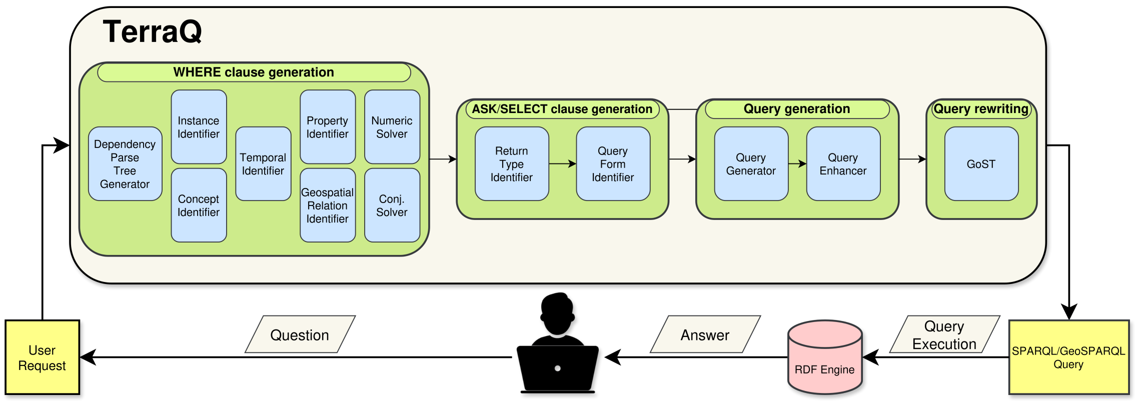

Fig. 1. The conceptual architecture of the TerraQ system

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 the knowledge graph. In the example question, the property "cloud coverage" of the *Image* concept will be identified and mapped to the corresponding property in the KG. For each identified concept, we try to match its properties to the words in the sentence using string similarity on n-grams. Matched properties are identified as candidate properties for this concept. Multiple concepts might have the same candidate property. To resolve this conflict, we introduced a heuristic that selects the syntactically closest concepts as the targets to the properties. This process is similar for instances inside the question.

Again, this component is also responsible for generating the block that will be used in the WHERE clause for the identified properties:

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

In addition, this component uses the dependency parse tree and Part-of-Speech tags to identify words that denote the use of comparatives and superlatives. These are subsequently matched to the appropriate Concept or Property, using a node-distance heuristic on the dependency parse tree.

Spatial relation Identifier. This module identifies spatial relations present in the input question and maps them to appropriate stSPARQL/GeoSPARQL functions. For instance, in the example question, it will identify the spatial relations "in" and "away from" and map them to *geof:sfWithin* and *geof:distance* respectively. Then these relations are mapped to the appropriate previously identified Instances and Concepts by using the following heuristic:

$$\text{distance} = \text{dependency\_parse\_tree\_distance} + \left(\frac{\text{word\_distance}}{100}\right)$$

Again, this component is also responsible for generating the block that will be used in the WHERE clause:

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

and

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

Numeric Solver. This module is responsible for identifying numbers, understanding their use in the input question and enhancing the previously identified elements with additional information. For this purpose we utilize Part-of-Speech tags and the previously described distance heuristic.

In our working example, "less than 2km" is matched to the spatial function of distance and "less than 10%" is matched to the cloud coverage property.

Conjunction Solver. The Conjunction Solver is responsible for handling conjunctions, as those are identified by the dependency parse tree. To that end, it selects all edges of the parse tree tagged as "conj:and". The vertices connected by each of those edges are checked for meaningful conjunctions. Number-to-Property, Number-to-Number and Geospatial-to-Geospatial conjunctions are supported. For Geospatial-to-Geospatial conjunctions additional spatial relations are generated and stored, as if they were created by the Geospatial Relation Identifier, according to the information provided by the vertices. In our example, this is the case with "towns and forests". Number and Property conjunctions function similarly.

Temporal Identifier. This module uses HeidelTime [15] to identify temporal keywords in the input question and annotates them with the appropriate date and/or duration. For instance, in the example input question it will identify "January 2021" and map it to 2021-01.

Return Type Identifier. This module is responsible for identifying the expected form/type of the answer to the question. The supported types are *Name, Coordinates, Number-Property, Number-Count, Image* . For our example, *Image* is the most appropriate return type. For identifying the expected return types, this component leverages the sophisticated language understanding of Llama 2 [2]. We fine-tune our model to output correctly formatted answers. A fallback mechanism that uses heuristics is provided to enable using TerraQ without hardware acceleration (GPU).

Query Form Identifier. This component is responsible for generating the final ASK/SELECT clause, which will be used by the query generator. It takes as input the list of return types generated by the Return Type Identifier and the user request. For each elemnt of the list, we do the following: 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. In the case of *Image*, we insert the appropriate code in the query. To determine the "next" object, we traverse the dependency parse tree.

Query Generator. 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. Information about superlatives, limits and other structures is taken into account in the generation process.

Query Enhancer. The query enhancer is an optional component responsible for modifying the query produced by the Query Generator to fix any mistakes and/or oversights. It is implemented using the Mistral-7b LLM fine-tuned on the dataset GeoQuestions1089 [16]. It serves as a performance-enhancement module that increases the capacity of TerraQ to answer complex questions following the Execution Refinement paradigm [17].

GoST. The GoST transpiler [16] takes the query generated by the Query Generator and rewrites it to use materialized geospatial relations if that is possible. Because geospatial relations like *geof:sfWithin* are computationally expensive we do offline materialization using the tool JedAI-Spatial [18].

IV. Evaluation

To the best of our knowledge, there is no publically available dataset that is suitable for evaluating systems on the task of Text-to-SPARQL for Earth Observation archives. For this reason, we decided to deploy our engine as a pure geospatial QA engine and run an evaluation on the geospatial QA dataset GeoQuestions1089 [16]. Although this did require some tinkering, since GeoQuestions1089 targets the YAGO2geo ontology, the process was straightforward and painless, and we believe that the resulting evaluation is useful for measuring the performance of most dimensions of our engine. Unfortunately, the questions in GeoQuestions1089 do not include temporal information.

To accept an answer as correct, it must match the gold result (included in GeoQuestions1089) exactly. We do not consider partially correct answers as correct. Likewise, for supersets of the answers in the gold set.

The results of our evaluation can be seen in Table I. We benchmark TerraQ with the Query Enhancer disabled, since the model was fine-tuned on the same dataset, which would skew the results. We also compare our engine to GeoQA2 and the engine of Hamzei et al [19].

TABLE I
Evaluation on GeoQuestions1089$_c$ v1.1

Category	GeoQA2 Accuracy	Hamzei Accuracy	TerraQ Accuracy
A	53.52%	28.16%	60.56%
B	62.68%	55.22%	73.13%
C	48.36%	30.06%	47.71%
D	9.09%	4.54%	22.73%
E	24.81%	6.56%	23.36%
F	28.57%	14.28%	38.10%
G	36.30%	12.32%	28.77%
H	23.07%	33.33%	40.17%
I	21.73%	8.69%	26.00%
ALL	40.33%	25.92%	44.36%

We can see that TerraQ outperforms the previous state of the art in most question categories without utilizing templates. All in all, there is 4% uplift in performance over the entire dataset, which is translated to a 10% improvement relative to GeoQA2. The categories with performance regressions are caused by TerraQ's more dynamic nature. TerraQ does not use predefined query templates and employs heuristics for a number of processes as previously described, these heuristics are not performing as well for those particular question categories.

V. Conclusion and Future Work

The success of modern digital assistants has clearly shown that natural language interfaces for computer systems and knowledge repositories can be a great boon for productivity and accessibility. Users nowadays expect to be able to interact with their computers through natural language, reducing the barrier of technical knowledge required for utilizing modern computing capabilities.

This paper presents TerraQ, a spatiotemporal QA system for satellite image archives. Our engine targets a high-quality, purpose-built knowledge graph that contains Sentinel-1 and Sentinel-2 image metadata, as well as geospatial information for administrative divisions and natural features. Requests made in natural language are translated to SPARQL queries, which are subsequently executed by an RDF store. This enables users to request, in natural language, satellite images satisfying a number of complex spatial, temporal and thematic criteria.

Our engine is easily deployable and responsive on commodity hardware, since it does not rely on exceedingly large LLMs. Instead, it utilizes a combination of small-scale LLMs, heuristics and expert knowledge. This development is another step towards our vision of making Earth Observation archives more accessible by both novice and expert users, no matter the available computing capacity.

In the future, we are planning on expanding the capabilities of our system by integrating Visual Question-Answering systems into our Knowledge Graph creation pipeline. This will enable users to express even more complex criteria for image selection, while also maintaining performance.

References

[1] OpenAI, "Gpt-4 technical report," 2024. [Online]. Available: https://arxiv.org/abs/2303.08774

[2] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. Canton-Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P. S. Koura, M. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. E. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, and T. Scialom, "Llama 2: Open foundation and fine-tuned chat models," *CoRR*, vol. abs/2307.09288, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2307.09288

[3] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de Las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M. Lachaux, P. Stock, T. L. Scao, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed, "Mistral 7b," *CoRR*, vol. abs/2310.06825, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2310.06825

[4] D. Diefenbach, V. López, K. D. Singh, and P. Maret, "Core techniques of question answering systems over knowledge bases: a survey," *Knowl. Inf. Syst.*, vol. 55, no. 3, pp. 529–569, 2018. [Online]. Available: https://doi.org/10.1007/s10115-017-1100-y

[5] K. Höffner, S. Walter, E. Marx, R. Usbeck, J. Lehmann, and A. N. Ngomo, "Survey on challenges of question answering in the semantic web," *Semantic Web*, vol. 8, no. 6, pp. 895–920, 2017. [Online]. Available: https://doi.org/10.3233/SW-160247

[6] S. Kefalidis, K. Plas, G. Stamoulis, D. Punjani, P. Maret, and M. Koubarakis, "Developing geospatial web applications using question answering engines, knowledge graphs and linked data tools," in *Web Information Systems Engineering - WISE 2024 PhD Symposium, Demos and Workshops - WEB-for-GOOD 2024, AIWDA 2024, SWIFT-AG 2024, and Demos, Doha, Qatar, December 2-5, 2024, Proceedings*, ser. Lecture Notes in Computer Science, M. Barhamgi, H. Wang, X. Wang, E. Aïmeur, M. Mrissa, B. Chikhaoui, K. Boukadi, R. Grati, and Z. Maamar, Eds., vol. 15463. Springer, 2024, pp. 233–242. [Online]. Available: https://doi.org/10.1007/978-981-96-1483-7_19

[7] P. S. H. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., 2020. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html

[8] S. Kefalidis, D. Punjani, E. Tsalapati, K. Plas, M. Pollali, P. Maret, and M. Koubarakis, "The question answering system geoqa2 and a new benchmark for its evaluation," *Int. J. Appl. Earth Obs. Geoinformation*, vol. 134, p. 104203, 2024. [Online]. Available: https://doi.org/10.1016/j.jag.2024.104203

[9] D. Punjani, M. Koubarakis, and E. Tsalapati, "Earthqa: A question answering engine for earth observation data archives," in *IEEE International Geoscience and Remote Sensing Symposium, IGARSS 2023, Pasadena, CA, USA, July 16-21, 2023*. IEEE, 2023, pp. 1396–1399. [Online]. Available: https://doi.org/10.1109/IGARSS52108.2023.10282475

[10] F. M. Institute, "Global oceans and seas, version 1," 2021. [Online]. Available: https://doi.org/10.14284/542

[11] N. Karalis, G. M. Mandilaras, and M. Koubarakis, "Extending the YAGO2 knowledge graph with precise geospatial knowledge," in *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part II*, ser. Lecture Notes in Computer Science, C. Ghidini, O. Hartig, M. Maleshkova, V. Svátek, I. F. Cruz, A. Hogan, J. Song, M. Lefrançois, and F. Gandon, Eds., vol. 11779. Springer, 2019, pp. 181–197. [Online]. Available: https://doi.org/10.1007/978-3-030-30796-7_12

[12] Matthew Perry and John Herring, "OGC GeoSPARQL - A Geographic Query Language for RDF Data," Open Geospatial Consortium, OGC Implementation Standard OGC 11-052r4, Sep. 2012. [Online]. Available: http://www.opengis.net/doc/IS/geosparql/1.0

[13] C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, and D. McClosky, "The stanford corenlp natural language processing toolkit," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, System Demonstrations*. The Association for Computer Linguistics, 2014, pp. 55–60. [Online]. Available: https://doi.org/10.3115/v1/p14-5010

[14] F. Piccinno and P. Ferragina, "From tagme to WAT: a new entity annotator," in *ERD'14, Proceedings of the First ACM International Workshop on Entity Recognition & Disambiguation, July 11, 2014, Gold Coast, Queensland, Australia*, D. Carmel, M. Chang, E. Gabrilovich, B. P. Hsu, and K. Wang, Eds. ACM, 2014, pp. 55–62. [Online]. Available: https://doi.org/10.1145/2633211.2634350

[15] J. Strötgen and M. Gertz, "Heideltime: High quality rule-based extraction and normalization of temporal expressions," in *Proceedings of the 5th International Workshop on Semantic Evaluation, SemEval@ACL 2010, Uppsala University, Uppsala, Sweden, July 15-16, 2010*, K. Erk and C. Strapparava, Eds. The Association for Computer Linguistics, 2010, pp. 321–324. [Online]. Available: https://aclanthology.org/S10-1071/

[16] S. Kefalidis, D. Punjani, E. Tsalapati, K. Plas, M. Pollali, M. Mitsios, M. Tsokanaridou, M. Koubarakis, and P. Maret, "Benchmarking geospatial question answering engines using the dataset geoquestions1089," in *The Semantic Web - ISWC 2023 - 22nd International Semantic Web Conference, Athens, Greece, November 6-10, 2023, Proceedings, Part II*, ser. Lecture Notes in Computer Science, T. R. Payne, V. Presutti, G. Qi, M. Poveda-Villalón, G. Stoilos, L. Hollink, Z. Kaoudi, G. Cheng, and J. Li, Eds., vol. 14266. Springer, 2023, pp. 266–284. [Online]. Available: https://doi.org/10.1007/978-3-031-47243-5_15

[17] Z. Hong, Z. Yuan, Q. Zhang, H. Chen, J. Dong, F. Huang, and X. Huang, "Next-generation database interfaces: A survey of llm-based text-to-sql," *CoRR*, vol. abs/2406.08426, 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2406.08426

[18] M. Papamichalopoulos, G. Papadakis, G. Mandilaras, M. D. Siampou, N. Mamoulis, and M. Koubarakis, "Three-dimensional geospatial interlinking with jedai-spatial," *CoRR*, vol. abs/2205.01905, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2205.01905

[19] E. Hamzei, M. Tomko, and S. Winter, "Translating place-related questions to geosparql queries," in *WWW '22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022*, F. Laforest, R. Troncy, E. Simperl, D. Agarwal, A. Gionis, I. Herman, and L. Médini, Eds. ACM, 2022, pp. 902–911. [Online]. Available: https://doi.org/10.1145/3485447.3511933