EARTHQA: A QUESTION ANSWERING ENGINE FOR EARTH OBSERVATION DATA ARCHIVES*

Dharmen Punjani[1], Manolis Koubarakis[2,3]† and Eleni Tsalapati[2]

[1]Hubert Curien Laboratory, Université Jean Monnet, St. Etienne, France
dharmen.punjani@gmail.com
[2]Dept. of Informatics and Telecommunications
National and Kapodistrian University of Athens, Greece
koubarak@di.uoa.gr, etsalapati@gmail.com
[3]LeibnizAILab, L3S
Leibniz Universität Hannover, Germany

ABSTRACT

EarthQA is a question answering engine that accepts questions in natural language (English) that ask for satellite images satisfying certain criteria and returns links to such datasets, that can be then downloaded from the CREODIAS cloud platform. The questions can refer to image metadata (e.g., satellite platform, sensing period, cloud cover etc.) but also to entities from the knowledge graph DBpedia (e.g., Mount Etna or the city of Munich). In this way, the users can ask questions like "Give me Sentinel-2 satellite images that show Mount Etna, have been taken in February 2021 and have cloud cover less than 10%."

***Index Terms*—** question answering, knowledge graphs, satellite data archives

1. INTRODUCTION

When Earth Observation (EO) experts or users query an image archive (e.g., ESA's Copernicus Open Access Hub), they typically use a graphical user interface where they select the geographical area of the image(s) they are interested in and additionally specify some other metadata such as sensing period, satellite platform, cloud cover etc. to retrieve the image(s) that satisfy their criteria. In our work, we are developing the *question answering (QA) engine EarthQA* that takes as input a question expressed in *natural language* (English) that asks for satellite images satisfying certain criteria and returns links to such datasets, that can be then downloaded from the CREODIAS cloud platform.

The version of EarthQA presented in this paper has been developed in the context of the Horizon 2020 project AI4Copernicus[1] and a demo is available publicly[2]. A new version of EarthQA is currently under development in the ESA project "DA4DTE: Demonstrator Precursor Digital Assistant Interface for Digital Twin Earth".

2. THE EARTHQA ENGINE

EarthQA has been developed using the Qanary methodology and the Frankenstein platform [1]. Qanary is a lightweight component-based methodology for the rapid engineering of QA pipelines. Frankenstein is the most recent implementation of the ideas of Qanary. Thus, we take advantage of the

*THIS WORK WAS SUPPORTED BY H2020 PROJECT AI4COPERNICUS (GRANT NO. 101016798), THE HELLENIC FOUNDATION FOR RESEARCH AND INNOVATION PROJECT GEOQA (GRANT NO HFRI-FM17-2351), THE ESA PROJECT DA4DTE (SUBCONTRACT NO. 202320239) AND THE FEDERAL MINISTRY OF EDUCATION AND RESEARCH - GERMANY PROJECT LEIBNIZKILABOR (GRANT NO. 01DD20003).

†Corresponding author.

[1]https://ai4copernicus-project.eu/
[2]http://earthqa.di.uoa.gr/

Frankenstein framework to create QA components which collectively implement the QA pipeline reusing some components from the geospatial QA engine GeoQA [2, 3][3] and adding some more components that are appropriate for questions targeting Earth observation data archives.

To answer user questions, EarthQA essentially queries two interlinked *knowledge graphs*: a knowledge graph encoding metadata of satellite images from the CREODIAS cloud platform and the well-known knowledge graph DBpedia (`https://www.dbpedia.org/`). The Sentinel missions Sentinel-1, Sentinel-2 and Sentinel-3 are covered. The metadata knowledge graph uses the ontology of the SPARQL endpoint of CREODIAS[4], a subset of which can be seen in Figure 1.

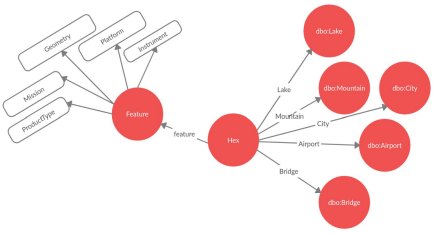

Fig. 1. Part of the ontology used by EarthQA

This ontology contains a class `Feature` which represents satellite products. All the metadata of a product (e.g., mission, instrument or platform) are encoded as properties of the class `Feature`. The class `Feature` also contains the property `Geometry`, which represents the bounding box of the geographical area corresponding to the satellite image of the product. The ontology also contains classes of spatial objects from DBpedia (e.g., `City` or `Mountain`). DBpedia entities (i.e., instances of DBpedia classes) also have geometries; however, these are only latitude and longitude coordinates representing the center of the corresponding areas in geographic space.[5]

[3] `https://github.com/AI-team-UoA/GeoQA2`

[4] `https://sparql.creodias.eu:30035/#/repositories/creodias/overview`

[5] It is clear that for some of the DBpedia classes, e.g., `River` or `City` etc., more complex geometries like lines or polygons would be more appropriate. However, DBpedia does not support them. The more recent geospatial knowledge graph YAGO2geo [4] will be used in future versions of EarthQA to alleviate this situation.

Instances of the class `Feature` (i.e., satellite products) are linked to instances of DBpedia classes (e.g., the city of Munich or Mount Etna) using instances of the class `Hex`. `Hex` is not a conceptual class representing objects of our domain. It is rather an implementation class and its instances are created using the spatial index H3. H3 is a hexagonal hierarchical spatial index developed by Uber[6] which partitions the surface of the globe into a hierarchical collection of hexagons. Every spatial object (e.g., Mount Etna or the geometry of a satellite product) falls into one of these hexagons. Using the class `Hex`, an instance of the class `Feature` (i.e., a satellite product) is interlinked with all DBpedia entities that have geometries that are within the same H3 hexagon as the geometry of the instance. This interlinking of DBpedia entities with satellite mission products represented by the class `Feature` enables the SPARQL endpoint of CREODIAS to answer efficiently queries that refer to both satellite products and DBpedia class instances (e.g., "Give me Sentinel-2 satellite images that show Mount Etna that have been taken in February 2021 and have cloud cover less than 10%").

The architecture of EarthQA is shown in Figure 2.

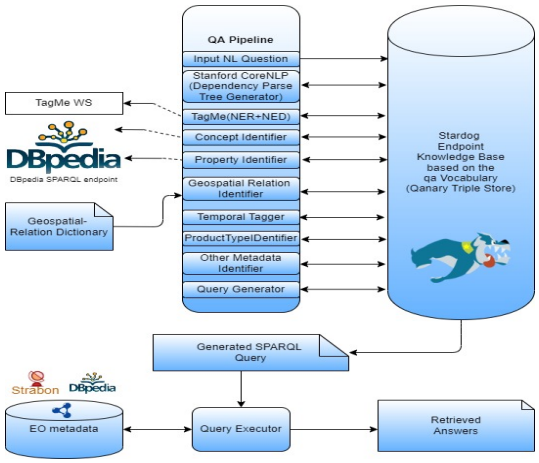

Fig. 2. Architecture of the EarthQA implementation

To get an understanding of how EarthQA works, we will discuss the processing of the ques-

[6] `https://www.uber.com/en-GR/blog/h3/`

tion "Find Sentinel-3A WFR products that covers islands in France with area greater than 8000 square kms, with the data collected in January 2018 and having cloud coverage less than 10%". EarthQA processes this question as follows.

Dependency Parse Tree Generator. This module generates a dependency parse tree of the input question using the StanfordCoreNLP library.

Instance Identifier. This module identifies and maps the entity that is present in the input question to the appropriate resource of the DBpedia ontology. For instance, in the example question, it will identify entity "France" and map it to `dbr:France` in the DBpedia ontology. A sequence of words in the input question might be identified as an instance if it has been tagged as a (proper) noun (POS tags NN, NNS and NNP) during dependency parsing. The mapping to the DBpedia resource is done using the named entity recognition and disambiguation tool TagMeDisambiguate [5].

Concept Identifier. This module identifies and maps the concept (point of interest) that is present in the input question to the appropriate resource of the DBpedia ontology. For instance, in the example question, it will identify the concept "islands" and map it to class `dbo:Island` in the DBpedia ontology. The concepts are identified by the elements of the question that are tagged as nouns (POS tags NN, NNS, NNP and NNPS) during dependency parsing. The mapping to the DBpedia class is done using string matching based on n-grams.

Spatial relation Identifier. This module identifies spatial relations present in the input question and maps them to appropriate properties in DBpedia. For instance, in the example question, it will identify the spatial relation "in" and maps it to `dbo:location` considering class `dbo:Island` and instance `dbr:France` in the DBpedia ontology. Spatial relations are identified based on the POS tags VB, IN, VP, VBP and VBZ generated during dependency parsing.

Property Identifier. This module identifies the attributes of the concept or instance present in the input question and maps them the appropriate properties of DBpedia. For instance, in the input question, it will identify the attribute "area" and map it to property `dbp:areaKm` considering the class `dbo:Island` of the DBpedia ontology. 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.

Temporal Tagger. This module identifies temporal keywords in the input question and annotates them with the appropriate date. For instance, in the example input question it will identify "January 2018" and map it to "2018-01". The module uses the temporal tagger HeidelTime [6].

ProductType Identifier. This module identifies metadata about Mission, Platform and Product type from the input question. It uses list of all the available Mission with its platform and associated product type with mission. N-grams are generated from the question based on the number of words present in the product type. If product type is "Water Full Resolusion" than 3-grams are generated from question and string similarity measures are used to find if question contains product type or not. For instance, in the example question, it will identify "Sentinel-3A WFR" and map it to mission `http://ws.eodias.eu/metadata/mission/Sentinel-3`, platform `http://ws.creodias.eu/metadata/platform/S3A` and product type `http://ws.eodias.eu/metadata/productType/WFR` of the CREODIAS ontology.

Other Metadata Identifier. This module identifies other metadata of the satellite product like cloud coverage, orbit direction, processing level, swath etc. For instance, for the input question, it will identify "cloud coveragre" and map it to the property `creodiasm:cloudCover` in the CREODIAS ontology.

Query Generator. The query generator takes the output from all the previous modules into consideration, and based on that, it generates a SPARQL query corresponding to the input question. The query is generated using handcrafted query templates as in [2]. For the example question, it will generate the SPARQL query shown on the next page (Listing 1).

Query Executor. The query executor runs the SPARQL query generated by the query generator over the geospatial RDF store Strabon where the two knowledge graphs are stored, and returns links to the satellite products requested in the user ques-

tion. The user of EarthQA can then use these links to download the relevant products from the CREODIAS platform.

Listing 1. The Generated SPARQL query for the example question

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX creodiasm: <http://ws.creodias.eu/metadata/attribute#>
select distinct ?title ?geom
where {
 ?hex ?pred ?dbpediaent.
 ?hex <http://ws.creodias.eu/metadata/attribute#feature> ?x .
 SERVICE <https://dbpedia.org/sparql>
    {
        ?dbpediaent a dbo:Island;
            ?p1 dbr:France;
            dbp:areaKm ?property.
        FILTER( ?property > 8000) .
    }
 ?x creodiasm:title ?title .
 ?x creodiasm:geometry ?geom .
 ?x creodiasm:mission <http://ws.creodias.eu/metadata/mission/
      Sentinel-3> .
 ?x creodiasm:platform <http://ws.creodias.eu/metadata/platform/S3A
      >.
 ?x creodiasm:productType <http://ws.creodias.eu/metadata/
      productType/WFR> .
 ?x creodiasm:cloudCover ?cc.
 ?x creodiasm:startDate ?date .
 filter(?cc<10) .
 bind(year(?date) as ?year) .
 bind(month(?date) as ?month) .
 FILTER(?year=2018 && ?month=01) .
}LIMIT 100
```

3. CONCLUSIONS AND FUTURE WORK

Although it can be argued that the current functionality of EarthQA discussed above is equivalent to what is offered by current graphical user interfaces such as the one for the Copernicus Open Access Hub, our long-term vision is to develop an engine that could also be used by *non-experts* interested in EO data. In the extended engine users will be able to ask questions like "I am interested in high resolution satellite imagery, taken during the summer of 2022, over cities in Italy with more than one million residents, that can be used to map green urban areas". In this engine, searching for a satellite image will be as easy as searching for any other piece of information on the Web today. This vision of opening up satellite image archives and making their contents available to expert and non-expert users alike has been the goal of our group since 2010 with work on pioneer projects TELEIOS [7], LEO [8, 9] and ExtremeEarth [10, 11]. Another important direction of our research is to experiment with deep neural networks and large language models in the various stages of the EarthQA pipeline with the hope of making it more effective.

4. REFERENCES

[1] K. Singh et al., "Why reinvent the wheel: Let's build question answering systems together," in *WWW*, 2018.

[2] D. Punjani et al., "Template-based question answering over linked geospatial data," in *Geographic Information Retrieval (GIR 2018) workshop. Collocated with ACM SIGSPATIAL*, 2018.

[3] D. Punjani et al., "Template-based question answering over linked geospatial data," *CoRR*, 2020, Available at `https://arxiv.org/abs/2007.07060`.

[4] N. Karalis, G. M. Mandilaras, and M. Koubarakis, "Extending the YAGO2 knowledge graph with precise geospatial knowledge," in *ISWC*, 2019.

[5] P. Ferragina and U. Scaiella, "TAGME: on-the-fly annotation of short text fragments (by wikipedia entities)," in *CIKM*, 2010.

[6] J. Strötgen and M. Gertz, "HeidelTime: High quality rule-based extraction and normalization of temporal expressions," in *SemEval@ACL*, 2010.

[7] M. Koubarakis, C. Kontoes, and S. Manegold, "Real-time wildfire monitoring using scientific database and linked data technologies," in *EDBT*, 2013.

[8] S. Burgstaller et al., "LEOpatra: A mobile application for smart fertilization based on linked data," in *HAICTA*, 2017.

[9] M. Koubarakis et al., "Managing big, linked, and open Earth observation data: Using the TELEIOS/LEO software stack," *IEEE Geoscience and Remote Sensing Magazine*, vol. 4, no. 3, 2016.

[10] M. Koubarakis et al., "From Copernicus big data to Extreme Earth analytics," in *EDBT*, 2019.

[11] F. Appel et al., "ExtremeEarth: Managing water availability for crops using Earth observation and machine learning," in *EDBT*, 2023.