The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XLVIII-4/W1-2022
Free and Open Source Software for Geospatial (FOSS4G) 2022 – Academic Track, 22–28 August 2022, Florence, Italy

SEMANTIC QUERYING IN EARTH OBSERVATION DATA CUBES

L. van der Meer[1,*], M. Sudmanns[1], H. Augustin[1], A. Baraldi[2], D. Tiede[1]

[1]Department of Geoinformatics, University of Salzburg, 5020 Salzburg, Austria –
(lucas.vandermeer, martin.sudmanns, hannah.augustin, dirk.tiede)@plus.ac.at
[2]Spatial Services GmbH, 5020 Salzburg, Austria - andrea6311@gmail.com

Commission III, ICWG III/IVb

KEY WORDS: Big Earth data, Earth observation, EO data cubes, Remote sensing, Semantic querying, Spatio-temporal analysis

ABSTRACT:

Earth observation (EO) data cubes have revolutionized the way large volumes of EO data can be stored, accessed, and processed. However, users coming from application domains outside of traditional EO research still face some significant technical barriers when querying an EO data cube with the aim to infer knew knowledge about real world entities and events. They have to interpret EO data in order to give them meaning, which is an ill-posed problem that requires advanced expertise in the field of EO analytics. We propose a semantic querying framework in which users query the EO data cube through an ontology, rather than accessing the data values themselves. The ontology formalizes symbolic representations of real-world entities and events, which are mapped to data values in the EO data cube through a mapping component formulated by an EO expert. This takes away the need for users to be aware of the EO data and how to interpret them, and therefore lowers the technical barriers to extract valuable information from EO data. We implemented a proof-of-concept of our approach as an open-source Python package.

1. INTRODUCTION

Data produced by remotely sensed Earth observation (EO) imagery are an important source of information to monitor and understand Earth processes, and have the potential to play a key role in achieving the Sustainable Development Goals (Kavvada et al., 2020). In the past years we have seen a large increase in the amount of satellites orbiting the Earth, as well as in their technical capabilities. This has lead to a constantly growing pool of EO data at higher temporal, spatial and spectral resolutions than before (Giuliani et al., 2017). Moreover, many of them are freely and openly available to everyone.

To manage the large supply of EO data in an efficient and organized way, recent technological developments have emerged around a central paradigm called the EO data cube (Giuliani et al., 2019). In this approach, the EO data are assigned to a common grid spanning the full spatio-temporal extent of the observations, and allowing efficient coordinate-based access (Lewis et al., 2016). EO data cubes form the backbone of many cloud-based analysis platforms on which users can process large amounts of EO data without having to worry about its size and heterogeneity. This is in contradiction to traditional approaches that include downloading the imagery to ones personal computer and processing it locally (Sudmanns et al., 2020).

Despite its success in abstracting much of the complexity regarding storage, management and access of EO data, users still face some significant barriers when querying an EO data cube. Firstly, they have to be aware of the structure of the EO data cube in order to query it. They have to know what data layers it contains (e.g. the different spectral bands of a satellite image), what they are called, and where to find them.

Secondly, and most importantly, the data that they query are not equal to information. Optical EO data are sampled reflectance values from the land cover material. These values do not have inherent meaning by themselves, i.e. they lack semantics. Translating these values into information about the real world therefore requires interpretation, bridging the so-called sensory and semantic gaps (Arvor et al., 2019). This is not a trivial task, because it involves reconstructing the four-dimensional physical world from two-dimensional imagery containing data that can only describe a limited set of properties of a real-world entity or event, such as its color (Sudmanns et al., 2021a).

Dodging these barriers requires advanced technical expertise in the field of EO analytics. This is a problem especially now that EO data has a growing pool of users that come from application domains outside of traditional EO research (Sudmanns et al., 2020; Wagemann et al., 2021). On top of that there may exist an even bigger pool of potential users that could benefit from using EO data in their application domain, but are prevented from doing so because of technical barriers such as the ones described above. They are generally experts in their application domain, but not experts in the EO data itself. Most of them are used to work with symbolic representations of real-world entities and events, rather than numeric representations of a measured property of them (Arvor et al., 2019).

Hence, we acknowledge the need for an approach that allows to extract information from an EO data cube using symbolic terminology. In this paper, we will propose such an approach, in which users query an EO data cube through an ontology, rather than directly accessing the data values themselves. We term our approach "semantic querying of EO data cubes", referring to approaches with similar goals that exist in the field of relational database management (Chokri, 2007), and building upon earlier work by Tiede et al. (2017).

The paper is structured as follows. In Section 2, we describe the different components of the approach and how they interact with each other, as well as clarify the core concepts. In Section 3, we present the implementation of our approach as an open-source Python package, accompanied with a practical example.

* Corresponding author

In Section 4, we discuss the benefits and limitations of our approach, and give directions for future work. Finally, in Section 5 we conclude our findings.

2. CONCEPTUAL FRAMEWORK

The framework that structures our proposed semantic querying approach consists of multiple components that interact with each other. The overall idea is as follows. A user writes a *recipe* for inference of new knowledge about the real word, in which they refer to real-world entities and events directly by their name. These entities and events are formally conceptualized in an *ontology*. A *mapping* defines how these formalized concepts are represented by the EO data. The EO data themselves are stored inside an *EO data cube*.

As an overarching structure, we define two conceptual domains that span the extent of the framework. The first one is the *semantic domain*. This is the domain of symbolic representations abstracting the real world. The second one is the *image domain*. This is the domain of numeric measurements captured from the real world.

We also define three different roles for actors in the framework. Each of these roles require a different type of expertise. The *application expert* has expert knowledge about a specific field of study (e.g. forestry, agriculture, urban planning) in which EO data can be applied to gain new knowledge about the real world. The *EO expert* has expert knowledge about EO data and how to interpret them. The *software expert* has expert knowledge about the design and deployment of data storage and management systems. The roles don't overlap, and hence, each role can be taken on by a different person without the need for them to also have the expertise of one of the other roles. However, it may be that a single person takes on more than one role. For example, an application expert may also be an EO expert at the same time.

In the following sub-sections we describe each of the components of the framework in more detail, define which of the conceptual domains they fall into, and state which of the roles are acting in it. In the last sub-section, we will shortly discuss how semantically enriching the EO data cube can benefit the structure and efficiency of the framework. The full framework is summarized graphically in Figure 1.

2.1 Ontology

An ontology is an "explicit specification of a conceptualization" (Gruber, 1993). In practice, this means that the ontology formalizes semantic concepts and the relationships between them. Semantic concepts are abstractions of entities and events that exist in the real-world. The ontology should be agreed upon by a community, which may contain both application experts and EO experts.

Semantic concepts can be formalized by specifying their properties. For example: "a mountain lake (concept) has a blue color (property) and a flat topography (property) and a high elevation (property)". Such a formalization can be improved by including relationships between different concepts, in a hierarchical structure that supports the inheritance of properties. For example, the same formalization as above can be achieved by separating three different semantic concepts: "water (concept) has a blue color (property)", "a lake (concept) is water (inherited properties) and has a flat topography (additional property)", and "a mountain lake (concept) is a lake (inherited properties) and has a high elevation (additional property)".

It goes beyond the scope of this paper to provide a detailed explanation of all the different forms of ontologies that exist, and how they can be tailored to usage within the field of EO analytics. For that, we refer to the work of Arvor et al. (2019). The gist is that the role of the ontology in our framework is to specify what the real-world entities and events of interest are and how they are conceptualized, such that they can be queried.

It should be emphasized that in our framework the ontology operates exclusively in the semantic domain. Hence, it does not contain any terms or data belonging to the image domain. That is, an ontology may state that "vegetation has high photosynthetic activity", but not "vegetation has NDVI larger than 0.6".

2.2 EO data cube

The EO data cube is the component of the framework that stores and organizes the EO data. Hence, it falls within the image domain of the framework. Besides the EO data themselves, the data cube may also contain additional data layers that describe certain properties of real-world entities or events. For example, a digital surface model. This is the component where the software experts act. It is their task to construct and manage the infrastructure, or to connect to existing deployments.

There is not a single standard on how an EO data cube should be implemented in practice. Many approaches exist, which we refer to as different *configurations* of an EO data cube. Our semantic querying framework does not put any limitations on what the configuration of an EO data cube should be. To ensure interoperability, it requires two standard elements to be attached to an EO data cube of any configuration.

The first one is the layout. The layout is a repository containing a description of the content of the EO data cube (Sudmanns et al., 2021b). Each distinct data layer in the cube (e.g. the different spectral bands) should be described by a dedicated metadata object in the layout that can be indexed by a unique name assigned to that layer. This metadata object provides information about the values in the layer, but also about where it can be found inside the structure of the cube. The layout may have a nested structure that formalizes a categorization of the data layers. For example, a category "reflectance" could group together different spectral bands of a satellite image.

The second one is the retriever. The retriever is an interface that exposes the content of the EO data cube to the other components in the querying framework. Given a textual reference to a data layer, it is able to retrieve the corresponding data values from the cube, and return them in a three-dimensional, spatio-temporal array. The exact implementation of the retriever depends on the configuration of the EO data cube, but the inputs it accepts and the output it returns does not. Hence, the retriever serves as a standardized API that allows other components of the framework to access the content of the EO data cube independent from its configuration.

2.3 Mapping

The mapping component of the framework forms the connection between the semantic domain and the image domain. This is where the EO experts act. They bring in their knowledge by formulating rules that map semantic concepts formalized in the ontology to data values stored in the EO data cube.

The rules in the mapping should support computational inference on the pixel level. That is, for each spatio-temporal location contained in the EO data cube, they should be able to quantify a relation to the semantic concept. Such a relation may be binary, stating if the concept was observed at the location ("true") or not ("false"). However, relations may also be numeric (e.g. a probability that the concept was observed) or ordinal (e.g. "likely" or "very likely" that the concept was observed).

When formulating the rules of a mapping the EO expert can specify the required data inputs through calls to the retriever of the EO data cube, and thus, does not have to be aware of the configuration of the cube. It is important to emphasize here that there is no limitation to formulate rules as a function of only the data values of the pixel to which they are applied. In order to quantify the relations more accurately, the EO expert may exploit all the values in the EO data cube. For example, they may analyze spatial or temporal patterns around the location of the pixel.

How the rules are formulated in practice depends largely on the approach that the EO expert decides to take. This can be entirely knowledge driven, but also hybrid approaches combining knowledge-driven and data-driven components are possible. For example, a rule may interpret the prediction of a machine learning classifier. Approaches can also differ in how they encode the formulated rules. We refer to all these different approaches as different *configurations* of a mapping.

Our semantic querying framework is flexible, and does not put any limitations on what the configuration of a mapping should be. Instead, it requires a mapping of any configuration to be paired with a translator. The translator is an interface that exposes the rules in the mapping to the other components of the querying framework. Given a textual reference to a semantic concept, it is able to apply the corresponding rules to the data values in the EO data cube, and return for each pixel the quantified relation with the semantic concept. We call the array it returns a *semantic array*, since it contains pixel values that relate directly to semantic concepts. For example, if the rules that map the semantic concept "water" formulate a binary relation, the translation of that concept will be a semantic array containing a "true" value for each pixel at which water was observed, and a "false" value otherwise. The exact implementation of the translator depends on the configuration of the mapping, but the inputs it accepts and the output it returns does not. Hence, the translator serves as a standardized API that allows other components of the framework to evaluate the rules in the mapping independent from its configuration.

2.4 Query recipe

A query recipe is written by an application expert and specifies their instructions for the inference of new knowledge about the real world. In this recipe they reference semantic concepts, as formalized in the ontology, directly by their name. That means that query recipes fall entirely into the semantic domain, and don't contain any image domain terminology.

After referencing a semantic concept, the application expert can specify one or more analytical processes that should be applied to the translation of the concept (i.e. to its semantic array, see section 2.3), in order to obtain their desired result. For example, the application expert may be interested in counting how often a semantic concept was observed at each location in space, during a specified time interval. Or alternatively, obtain a time series that shows for each time an observation was made at how many spatial locations a semantic concept was observed.

In its essence, the query recipe allows the application expert to query the EO data cube through the ontology, without directly accessing the data. That means they don't need to be aware of the structure and content of the EO data cube, nor do they need to have the expert knowledge to interpret the EO data. However, they should be familiar with the content of the ontology.

2.5 Semantically enriching the EO data cube

In our framework, mappings have to span a large gap between the symbolic representations in the ontology and the numeric data in the EO data cube. This gap can be reduced by semantically enriching the EO data cube, meaning that one or more categorical layers are produced that offer at least a first degree of interpretation for each observation in the cube. Such layers can be seen as "semi-symbolic", since they are a first step to connecting EO data with symbolic, semantic classes (Augustin et al., 2019).

Essentially, this enriched structure splits the original role of the mapping into separate phases. In the first phase, the data are mapped to the semi-symbolic variables, which are then stored as additional layer in the EO data cube. In the second phase, the semantic concepts formalized in the ontology are mapped to these semi-symbolic variables. A semi-symbolic layer could be produced by the Satellite Image Automated Mapper (SIAM), which implements a decision tree that assigns each observation a multi-spectral color index (Baraldi et al., 2006), or by the automated deep learning approach of Dynamic World, which produces descriptive land cover and land use labels for each observation (Brown et al., 2022). The EO expert can then refer to these indices and labels in the remaining rules of the mapping, and further customize them to best represent (properties of) semantic concepts. Approaches based on other algorithms are possible as well.

A benefit of the semantic enrichment of the EO data cube is that it increases computational efficiency. Parts of mapping rules that can be easily automated don't have to be evaluated on the fly when executing a query recipe. The benefits are also reflected in the conceptual structure of the framework itself. It is particularly suitable to combine knowledge-driven with data-driven approaches. Furthermore, the remaining rules in the mapping component become simpler, easier to explain, and more stable, especially when the same categorical layers can be calculated for multiple data sources (e.g. both Landsat and Sentinel data, as is the case for SIAM).

3. TECHNICAL IMPLEMENTATION

A proof-of-concept of our semantic querying approach is implemented as an open-source Python package named semantique. It contains functions and classes that allow application experts to formulate their query recipes and execute them within the bounds of a finite spatio-temporal extent, given an instance of an EO data cube and a mapping. It also allows software experts to create Python objects that represent an EO data cube, and EO experts to formulate mappings. In the following sub-sections we describe these different use-cases in more detail. Throughout these sections we regularly refer to a practical example in which we consider an application expert in the

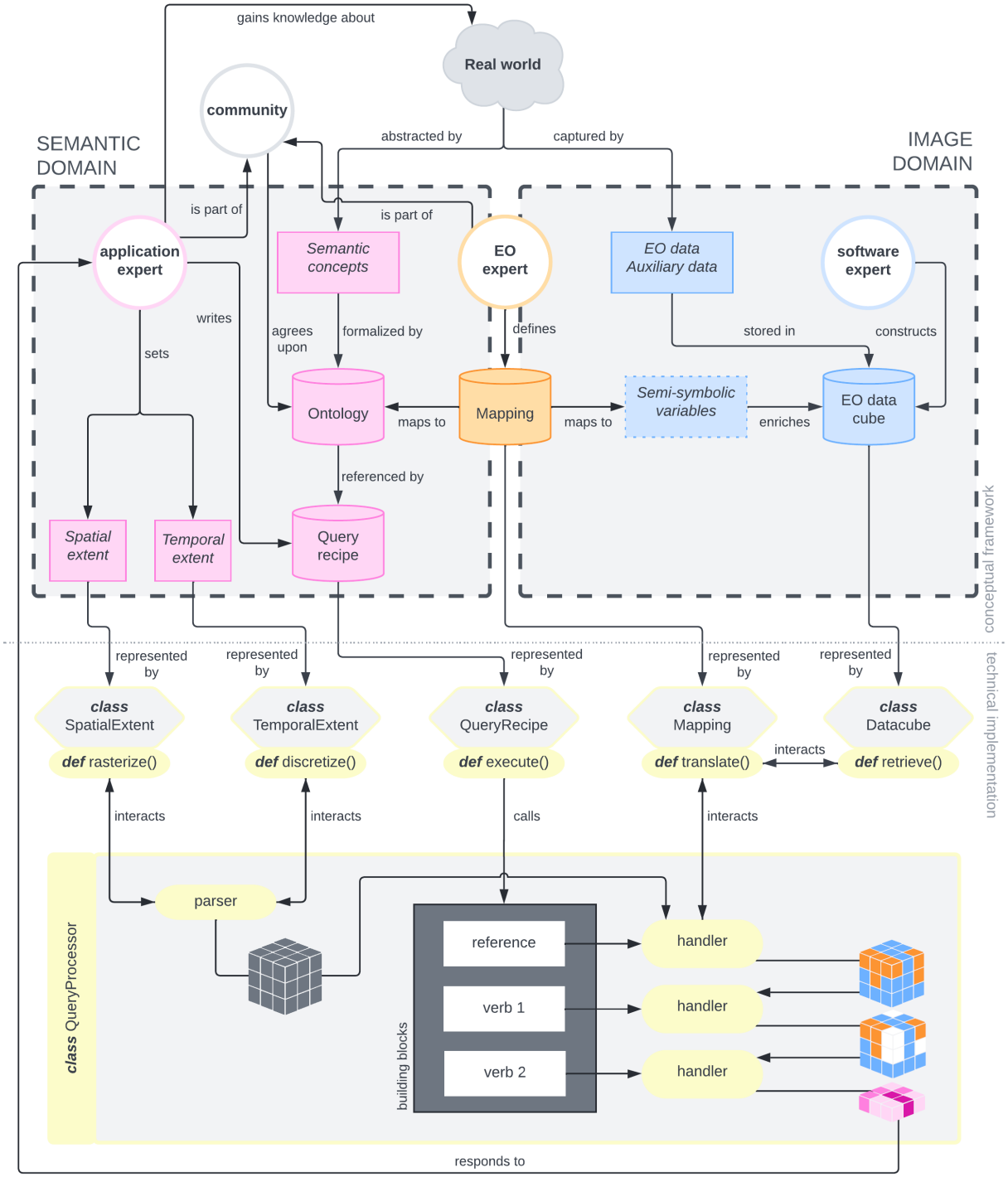

Figure 1. Graphical summary of the conceptual framework and its technical implementation.

field of urban planning who wants to use EO data to monitor urban green and blue spaces in Vienna during the summer of 2021. The example is intentionally oversimplified, in order to keep the focus on clarifying the ideas, and not get lost in the complexity of EO data interpretation.

The source code of semantique is released under the Apache license and available at `https://github.com/ZGIS/semantique`. The code is accompanied by extensive documentation including an API reference and several Jupyter notebooks, to which we refer for more detailed examples and visualizations. The implementation is summarized in Figure 1.

3.1 Writing a query recipe

A query recipe in semantique is represented by an instance of the `QueryRecipe` class. Such an object has the same structure as a Python dictionary, with each element containing the instructions for a single result, indexed by the name of that result. The starting point for the application expert using semantique is to initialize an empty instance of a recipe. The instructions for each desired result can then be added one by one as elements to this object.

The instructions can be intuitively formulated by chaining to-

gether basic building blocks in a with-do structure. The with part of the chain is the textual reference to a semantic concept. Such a reference can be formulated by providing the name of the concept to the `entity()` or `event()` function, depending on what the concept represents (i.e. a real-world entity or event, respectively). The do part specifies one or more analytical processes that should be applied to the array that results from the translation of the referenced concept (i.e. its semantic array, see section 2.3). Each of these processes is a well-defined array operation that performs a single task. It is labeled by an action word that should intuitively describe the operation it performs. Therefore we also call these type of building blocks *verbs*. The currently implemented verbs are listed in Table 1. We emphasize that this is not pretending to be an exhaustive list of all useful array operations, and hence, it might be extended or adapted depending on user feedback. The verbs can be called as methods of the reference, thus forming a chain of processes. Query recipes can be made increasingly complex by nesting multiple processing chains into each other.

Name	Description
Evaluate	Evaluates an expression for each pixel, e.g. an arithmetic operation (add, multiply, ..) or a condition (equals, greater, ..). The operands may include a constant (e.g. multiply by 2) or a second array (e.g. add the pixels of two arrays).
Extract	Extracts the dimension coordinates of a given dimension as a new one-dimensional array.
Filter	Filters values based on a condition. The condition can be evaluated by the Evaluate verb and provided as a binary array. Only the pixels evaluated as "true" are preserved, the others are assigned a nodata value. Besides regular value filters the verb can also be used for spatial filters and temporal filters.
Groupby	Splits the array into distinct subsets that each comprise a group of pixels sharing similar coordinates of a particular dimension, e.g. each group may correspond to a year.
Reduce	Reduces dimensionality by applying a function that returns a single value for each column along a given dimension, e.g. a summary statistic or a boolean.
Shift	Shifts the pixel values along a given dimension with a given offset.
Smooth	Smoothes the pixel values by applying a moving window function along a given dimension.

Table 1. Verbs for semantic arrays.

Instead of a reference to a single semantic concept, a chain may also start with a reference to a collection of multiple semantic concepts. Such collections have dedicated verbs that all in some way combine their semantic arrays back into a single one (see Table 2). Furthermore, verbs dedicated to single arrays (see Table 1) may be called on a collection as well. They will then be applied separately to each member of the collection. In combination with the groupby verb, this allows to model split-apply-combine structures.

Name	Description
Concatenate	Concatenates a collection along a given dimension. This may either be an existing or a new dimension.
Compose	Creates a categorical composition of a collection. This means that a pixel being "true" in the first array of the collection gets a value of 1 in the output, a pixel being "true" in the second array of the collection gets a value of 2, et cetera.
Merge	Merges corresponding pixels in a collection by applying a reducer function.

Table 2. Verbs for collections of semantic arrays.

In our example, we say that the urban planner wants to obtain two different maps: one showing for each location in space how much percent of the time green space was observed, and the other showing that for blue space. However, when calculating these percentages, they don't want to include pixels that where covered by clouds. A code snippet creating the query recipe for these results is shown in Figure 2. The output is an entirely textual object, not containing any data nor any Python code. This makes it easily shareable and storable as a JSON file.

```
import semantique as sq

recipe = sq.QueryRecipe()

recipe["greenspace_map"] = sq.entity("greenspace").\
  filter(sq.entity("cloud").evaluate("not")).\
  reduce("time", "percentage")

recipe["bluespace_map"] = sq.entity("bluespace").\
  filter(sq.entity("cloud").evaluate("not")).\
  reduce("time", "percentage")
```

Figure 2. Code snippet for writing a query recipe.

3.2 Representing an EO data cube

An EO data cube in semantique is always represented by an instance of a sub-class that inherits from the abstract base class `Datacube`. Such an object always has a `retrieve()` method that implements the retriever of the EO data cube. This flexible structure allows software experts to write their own sub-classes tailored to a specific EO data cube configuration they use, with their own implementation of the `retrieve()` method. Any object representing a EO data cube in semantique is initialized by its layout, which is expected to be written by a software expert, distributed with the EO data cube, and formatted as a JSON file. This makes them easily shareable and storable, and easy to load into a Python session as a dictionary. An EO data cube representation in semantique never contains the data values of the cube themselves. Instead, it will have some property that

allows the retriever to access the actual storage location. How this property looks like will depend on the configuration of the EO data cube.

Through the `Opendatacube` sub-class semantique has built-in support for EO data cubes that are deployed using the Open Data Cube software (Open Data Cube, 2022). It contains a retriever that knows how to interact with the Python API of Open Data Cube.

3.3 Constructing a mapping

A mapping in semantique is always represented by an instance of a sub-class that inherits from the abstract base class `Mapping`. Such an object always has a `translate()` method that implements the translator of the mapping. This flexible structure allows EO experts to write their own sub-classes tailored to a specific mapping configuration they use, with their own implementation of the `translate()` method.

Semantique formalizes its own native mapping configuration through the `Semantique` sub-class. It implements a knowledge-driven approach in which the EO expert formulates rules that quantify binary relations between properties of semantic concepts and data values. These property relations are then combined with a logical "and" operator to quantify the relation for the semantic concept as a whole. A `Semantique` instance has the same structure as a nested Python dictionary, in which the first layer of keys refer to the low-level categorization of the semantic concepts into entities and events, the second layer of keys refer to the names of the semantic concepts, and the third layer of keys refer to the names of their properties.

To formulate the rules for each property, the EO expert can use the same building blocks as made available for writing query recipes. However, they can start a processing chain with a textual reference to a data layer in the EO data cube, rather than to a semantic concept. Such a reference can then be directly followed by an evaluate verb that evaluates a condition on the data values of one or more layers. More complex rules can be formulated by utilizing more verbs, and nesting multiple processing chains into each other. The translator of a `Semantique` instance can evaluate the formulated rules while retrieving the required data values from a given EO data cube instance, and merge the resulting binary arrays using a logical "and" operator.

In our example, a mapping is needed that maps the semantic concepts "greenspace", "bluespace" and "cloud" to data values in the EO data cube. We assume a semantically enriched EO data cube that contains SIAM color indices as a categorical layer (see Section 2.5). The EO expert uses this layer to map the "color" property of the semantic concepts, which happens to be the only property the ontology formalized for these concepts. A code snippet creating the described mapping is shown in Figure 3. The output is again an entirely textual object that can be easily shared and stored as a JSON file.

3.4 Executing a query recipe

To execute their formulated query recipe, the application expert can call its `execute()` method. At this point they have to provide a mapping and a representation of an EO data cube. They can initialize a mapping instance of a certain configuration with rules that are shared with them by the EO expert, and a data cube instance of a certain configuration with a layout that is shared with them by the software engineer. The application

```
import semantique as sq
from semantique.mapping import Semantique

mapping = Semantique()
mapping["entity"] = {"greenspace": {}, "bluespace": {},
"cloud": {}}

greenlike = sq.layer("appearance", "colortype").\
  evaluate("in", [1, 2, 3, 4])

bluelike = sq.layer("appearance", "colortype").\
  evaluate("in", [21, 22, 23, 24])

cloudlike = sq.layer("appearance", "colortype").\
  evaluate("in", [25, 27, 28])

mapping["entity"]["greenspace"]["color"] = greenlike
mapping["entity"]["bluespace"]["color"] = greenlike
mapping["entity"]["cloud"]["color"] = cloudlike
```

Figure 3. Code snippet for constructing a mapping.

expert does not have to be aware of the content of these files, nor of the implementations of the respective `translate()` and `retrieve()` methods of the initialized objects.

Before executing the query recipe with respect to the mapping and the EO data cube, the application expert needs to set the spatial and temporal bounds in which the query needs to be evaluated. These bounds need to fall entirely inside the spatio-temporal bounds of the EO data cube. Semantique contains the classes `SpatialExtent` and `TemporalExtent` as representations for spatial and temporal bounds. In our example, the spatial extent is the city border of Vienna, and the temporal extent may be set equal to the interval between 1st of June 2021 and the 1st of September 2021.

When the `execute()` method is called, semantique internally creates a `QueryProcessor` object. This object can be seen as a worker that takes care of all tasks involving query processing, such as parsing, optimizing and executing the query. It first collects and stores the provided components as class properties. Then, it rasterizes the spatial extent and discretizes the temporal extent, and combines them together into a three-dimensional array that serves as a template for retrieved subsets of the EO data cube and semantic arrays. For the execution of the query, the query processor contains specific handler functions as methods, with each of them being able to handle one specific processing task. For example, a handler exists for calling the translator of the mapping in order to translate a semantic concept reference, and other handlers exist for applying a specific verb to the semantic array that was returned by the translator.

As a data structure for the spatio-temporal extent, retrieved subsets of the EO data cube and semantic arrays, the query processor uses `DataArray` objects from the xarray package (Hoyer and Hamman, 2017). It extends these objects by a semantique-specific accessor, that contains for example the implementations of the different verbs as methods. Most of them combine different array processing functions from xarray.

The response of the query processor, and hence, the object that is returned to the application expert, is a dictionary in which each element contains the executed instructions of a single result as a `DataArray` object. The dimensions of these arrays depend on the verbs that where called. Some results might only

have spatial dimensions (i.e. a map), others only a temporal dimension (i.e. a time series), and others might even be dimensionless (i.e. a single value such as a summary statistic).

In our example, both results are two-dimensional maps. They are visualized in Figure 4. Note that especially the blue space map shows clearly how oversimplified mappings based only on spectral signatures of single observations fail to capture all the complexity of the real world. Many shadowed areas are falsely mapped as being water. This underlines that it is of great importance for an accurate mapping to address multiple properties of semantic concepts, utilizing multiple data sources in an convergence-of-evidence approach.

Finally, it is important to emphasize that the ontology itself is not exploited by the query processor at all. It relies fully on the mapping in order to translate a semantic concept reference into a semantic array. Therefore, semantique does not require any specific structure of the ontology. Although we do recommend to take the time to explicitly formalize an ontology, query execution will also work when the ontology is "silently" agreed upon, and only made explicit through its mapping.

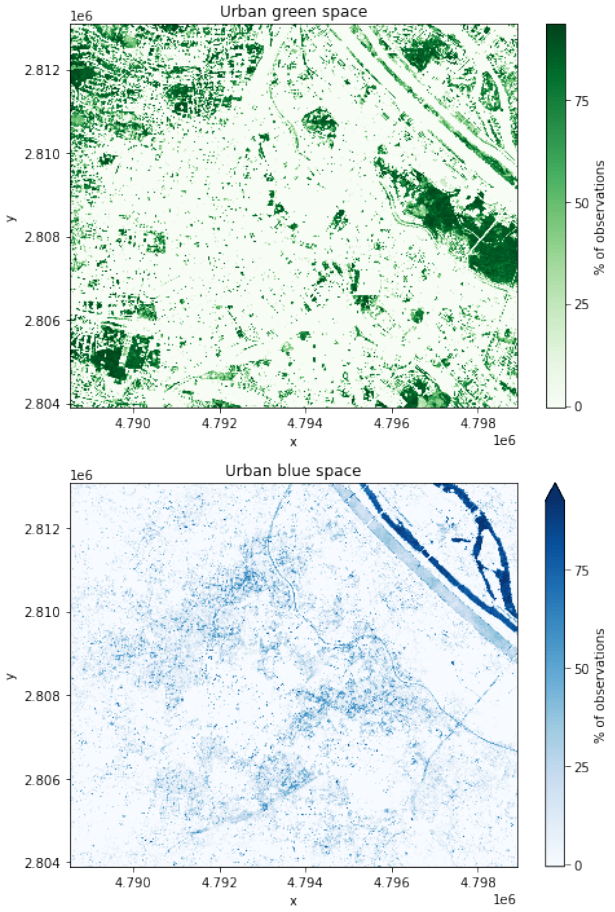

Figure 4. Visualized output of the query execution.

4. DISCUSSION

4.1 Benefits of the approach

The main benefit of using ontology-based approaches in EO analytics should not be measured in terms of a higher accuracy of the outputs, but in structural improvements (Arvor et al., 2019). This is also true for our semantic querying approach. The explicit separation between the semantic domain and the image domain benefits application experts, who can now infer knowledge from EO data and other, auxiliary data sources, without the need to be aware of the technical implications regarding the ill-posed interpretation of these data. Moreover, we believe the structure can also improve existing image processing workflows of EO experts in several ways.

Firstly, it makes the worfklows clearer and easier to manage. Definitions of semantic concepts are now made explicit in a single, separated component, instead of being implicitly represented inside larger query statements or analysis scripts. As such, it improves readability, reduces the need for duplicated code, and is less prone to mistakes in concept definitions. Also, concept definitions can be easily updated, without the need to rewrite the analysis scripts.

Secondly, it benefits interoperability and reusability. In our approach, query recipes reference stable concepts representing the real world. This makes them largely independent from the data in the EO data cube and the rules of the mapping. The same query recipe "count greenspace observations over time" can be used on different data sources and with different mappings. Whenever new data are made available or new interpretation techniques are developed, the query recipe remains stable. Also, both query recipes and mappings can be easily shared, reused and adapted, thus lowering the barriers for the exchange of knowledge between different working groups and different disciplines.

We see it as an additional benefit of our approach that the implemented workflow of chaining together different building blocks into a query recipe can easily be supported by a visual programming interface, taking away the need for an application expert to write code. This is demonstrated already in an operational setting by Sen2Cube.at, a nation-wide semantic data cube infrastructure for Austria, which uses the semantique package in the background (Sudmanns et al., 2021a).

4.2 Limitations and directions for future work

In the current implementation of our approach, application experts do not have to be aware anymore of the EO data and how to interpret them. However, they still need to understand what the structure of a translated semantic concept is (i.e. the semantic array) and how to apply array-specific processes to it. In a "next-level" semantic querying framework, this would not be needed anymore. The application expert can simply ask "How much of Vienna was green space in 2021?". The query processor would then be able to translate this text internally into a query recipe, and infer the spatial and temporal extent. This is more of a point on the horizon where semantic querying of EO data could go, rather than a realistic goal to achieve on short-term. It would allow the integration of EO into linked data and knowledge graphs.

A more graspable conceptual limitation of our approach is that it is still entirely pixel-based. Although it does allow EO experts to formulate mapping rules that look beyond a single pixel, it is not yet well-suited for purely object-based approaches. This makes it harder to formulate rules that map certain properties of real-world entities, such as their shape. A second conceptual limitation is that we did not yet develop a standardized way to quantify uncertainties in the mapping rules that goes beyond simple cloud cover statistics.

The technical side of our implementation also comes with its limitations, which mainly affect the computational efficiency of the query processor. Semantic queries are not yet validated nor optimized before execution. A first step to a more performant query processor could be to support parallel computation and lazy loading through the dask package, which is already well-integrated with xarray. Semantique could also benefit from integration with existing open-source platforms for EO data analysis, such as OpenEO Platform (openEO Platform, 2022), and emerging standards for exposing big EO data, such as STAC (SpatioTemporal Asset Catalog, 2022).

As a final remark, we should emphasize that in order to make the semantic querying approach useful, we should constantly motivate EO communities to explicitly formalize their knowledge into ontologies and associated mappings. In this regard, we encourage bottom-up approaches in which smaller communities (e.g. a research group) formalize their own ontologies and mappings, tailored to their specific applications. In line with the thoughts of Camara (2020) we believe this is a more realistic approach than waiting for a top-down attempt in which the whole EO community manages to capture all different contexts and views in a single ontology and few mappings.

5. CONCLUSIONS

In this paper we presented a new approach for semantic querying of EO data cubes, in which a user infers new knowledge from the data through an ontology, rather than by accessing the data values themselves. We implemented a proof-of-concept implementation of our approach as an open-source Python package, and believe it can be an important contribution to a broader uptake of EO data across various application domains.

ACKNOWLEDGEMENTS

This research was funded by the Austrian Research Promotion Agency (FFG) under the Austrian Space Application Programme (ASAP) within the projects Sen2Cube.at (project no.: 866016), SemantiX (project no.: 878939) and SIMS (project no.: 885365). We thank Steffen Reichel for advice on the technical implementation.

REFERENCES

Arvor, D., Belgiu, M., Falomir, Z., Mougenot, I., Durieux, L., 2019. Ontologies to interpret remote sensing images: why do we need them? *GIScience & Remote Sensing*, 56(6), 911-939.

Augustin, H., Sudmanns, M., Tiede, D., Lang, S., Baraldi, A., 2019. Semantic Earth Observation Data Cubes. *Data*, 4(3).

Baraldi, A., Puzzolo, V., Blonda, P., Bruzzone, L., Tarantino, C., 2006. Automatic Spectral Rule-Based Preliminary Mapping of Calibrated Landsat TM and ETM+ Images. *IEEE Transactions on Geoscience and Remote Sensing*, 44(9), 2563-2586.

Brown, C. F., Brumby, S. P., Guzder-Williams, B., Birch, T., Hyde, S. B., Mazzariello, J., Czerwinski, W., Pasquarella, V. J., Haertel, R., Ilyushchenko, S., Schwehr, K., Weisse, M., Stolle, F., Hanson, C., Guinan, O., Moore, R., Tait, A. M., 2022. Dynamic World, Near real-time global 10m land use land cover mapping. *Scientific Data*, 9(1), 251.

Camara, G., 2020. On the semantics of big Earth observation data for land classification. *Journal of Spatial Information Science*.

Chokri, B. N., 2007. Ontology-based Semantic Query Processing in Database Systems. PhD thesis, Humboldt-Universität zu Berlin.

Giuliani, G., Camara, G., Killough, B., Minchin, S., 2019. Earth Observation Open Science: Enhancing Reproducible Science Using Data Cubes. *Data*, 4(4).

Giuliani, G., Dao, H., De Bono, A., Chatenoux, B., Allenbach, K., De Laborie, P., Rodila, D., Alexandris, N., Peduzzi, P., 2017. Live Monitoring of Earth Surface (LiMES): A framework for monitoring environmental changes from Earth Observations. *Remote Sensing of Environment*, 202, 222-233.

Gruber, T. R., 1993. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2), 199-220.

Hoyer, S., Hamman, J., 2017. Xarray: N-D labeled arrays and datasets in Python. *Journal of Open Research Software*, 5(1), 10.

Kavvada, A., Metternicht, G., Kerblat, F., Mudau, N., Haldorson, M., Laldaparsad, S., Friedl, L., Held, A., Chuvieco, E., 2020. Towards delivering on the Sustainable Development Goals using Earth observations. *Remote Sensing of Environment*, 247, 111930.

Lewis, A., Lymburner, L., Purss, M. B. J., Brooke, B., Evans, B., Ip, A., Dekker, A. G., Irons, J. R., Minchin, S., Mueller, N., Oliver, S., Roberts, D., Ryan, B., Thankappan, M., Woodcock, R., Wyborn, L., 2016. Rapid, high-resolution detection of environmental change over continental scales from satellite data – the Earth Observation Data Cube. *International Journal of Digital Earth*, 9(1), 106-111.

Open Data Cube, 2022. opendatacube.org (10 June, 2022).

openEO Platform, 2022. openeo.cloud (10 June, 2022).

SpatioTemporal Asset Catalog, 2022. stacspec.org (10 June, 2022).

Sudmanns, M., Augustin, H., van der Meer, L., Baraldi, A., Tiede, D., 2021a. The Austrian Semantic EO Data Cube Infrastructure. *Remote Sensing*, 13(23).

Sudmanns, M., Augustin, H., Van Der Meer, L., Werner, C., Baraldi, A., Tiede, D., 2021b. One GUI to rule them all: Accessing multiple semantic EO data cubes in one graphical user interface. *GI_Forum*, 9(1), 53-59.

Sudmanns, M., Tiede, D., Lang, S., Bergstedt, H., Trost, G., Augustin, H., Baraldi, A., Blaschke, T., 2020. Big Earth data: disruptive changes in Earth observation data management and analysis? *International Journal of Digital Earth*, 13(7), 832-850.

Tiede, D., Baraldi, A., Sudmanns, M., Belgiu, M., 2017. Architecture and prototypical implementation of a semantic querying system for big Earth observation image bases. *European Journal of Remote Sensing*, 50(1), 452–463.

Wagemann, J., Siemen, S., Seeger, B., Bendix, J., 2021. Users of open Big Earth data – An analysis of the current state. *Computers & Geosciences*, 157, 104916.