Evaluator 1

ESIP Community Ontology Repository

The ESIP Community Ontology Repository (COR) is a custom-built software application from previous projects repurposed here as it has been for other projects such as X-DOMES. COR consists of a web-based UI, an API, used mostly for managing ontologies and building sets of terms, and a publicly-accessible SPARQL endpoint that powers all search-related capabilities across the repository. Search features are accessible through the web-based UI directly powered by the SPARQL endpoint with result sets displaying lists of triples. The layout of the UI is sleek and simple, making it very easy to navigate to intended functionality. However, since most of the UI functionality is driven by the SPARQL endpoint, some features suffer from performance issues based on the size of the ontology. The API and its Swagger documentation and services are incredibly intuitive and provide easy ways to interact with most of the portals features. My overall impression of COR is that it was intended to be a tool for managing ontologies and list of keywords and it does a great job of providing these services through the UI and API. It's my view that it doesn't seem well- suited to address the functionality required by the use cases in this report mainly due to its recommendation that these functions be served by the SPARQL endpoint.

Use Case #1: Use of Semantics within Search Engines

	The ontology portal has an API via which the ACME system can submit the term to be matched.	**The ontology portal can semantically match terms received as input to terms in the ontologies stored there.**	**The ontology portal can return a set of matching terms to the requesting application.**
COR	YES, but requires knowledge of semantics and SPARQL	YES	YES, but requires knowledge of semantics and SPARQL

The COR repository does not provide term search through its Swagger API, but does provide a SPARQL endpoint at which users can write queries to perform a similar function. As the use case is documented, it requires that a user not know anything about semantics to perform a term search across the repository. In this case, COR doesn't provide functionality to meet these requirements. COR does provide some documentation (https://mmisw.org/orrdoc/query/) with links to learn about SPARQL, but we noticed the link to example SPARQL queries did not resolve at the time of writing this report.

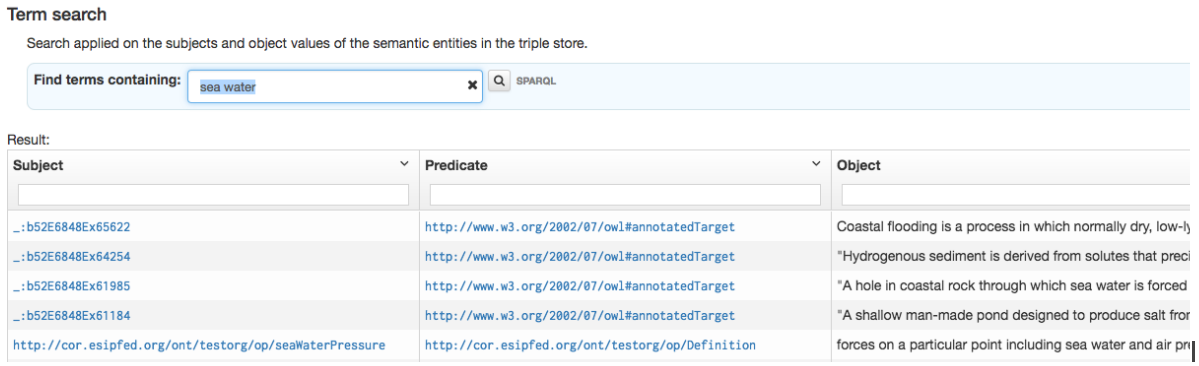
(https://marinemetadata.org/community/teams/vocdev/orrioos). However, it's my opinion, that for a novice to get something useful with a term query in SPARQL, this might be a difficult understanding the nuances of what to include in the query, how to structure it, and filter out any noise. Conversely, having the option to write a SPARQL query allows an application trying to improve search results the ability, in one query, to retrieve information about a term match's parents, children, and other relationships inside the repository. In this case, a knowledge of RDF triples and semantics would provide the potential to meet the real-world requirements for search engine term lookup use case.

I would suggest that COR provide an example SPARQL query for how someone might discover a matching term, information about the ontology it belongs to, and the classes parent, child relationships and mappings. I note that this example might be already available at https://marinemetadata.org/community/teams/vocdev/orrioos, but I was unable to resolve the URL for this report.

Use Case #2: Browsing a Portal for a Relevant Ontology

	The ontology portal provides the capability of searching across all of the ontologies it stores.	**There is a user interface and/or api that accepts a search term as input and returns appropriate results.**	**There are links among related concepts within an ontology.**
COR	YES	YES	YES

COR has a nice interface for filtering down ontologies by the some of the options set when adding an ontology to the repository. Then, when clicking an ontology, you can view the ontology's metadata fields and then in a section labeled data, the ontology is represented in RDF triples. In this form, the triples can be filtered by free-text for each triple element - subject, predicate, and object. Another view of the ontology is provided via LODE - the Live OWL Documentation Environment - an ontology documentation service that lets you browse classes and properties seeing the relationships in a clickable UI. COR's SPARQL endpoint drives the term search which is a nice feature for seeing all the related entries to a query. However, for someone not used to RDF triples, the result set may be a little confusing to interpret. In the picture below, I've searched for 'sea water' and the first results are blank nodes.



Term search

Search applied on the subjects and object values of the semantic entities in the triple store.

Find terms containing: sea water

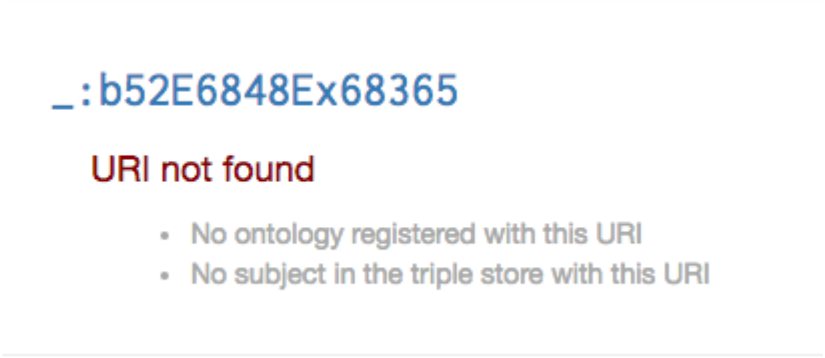
Result:

Subject	Predicate	Object
_:b52E6848Ex65622	http://www.w3.org/2002/07/owl#annotatedTarget	Coastal flooding is a process in which normally dry, low-l
_:b52E6848Ex64254	http://www.w3.org/2002/07/owl#annotatedTarget	"Hydrogenous sediment is derived from solutes that preci
_:b52E6848Ex61985	http://www.w3.org/2002/07/owl#annotatedTarget	"A hole in coastal rock through which sea water is forced
_:b52E6848Ex61184	http://www.w3.org/2002/07/owl#annotatedTarget	"A shallow man-made pond designed to produce salt fro
http://cor.esipfed.org/ont/testorg/op/seaWaterPressure	http://cor.esipfed.org/ont/testorg/op/Definition	forces on a particular point including sea water and air pr

think this may be a little confusing for new users and frustrating to experienced users who will want to write a better SPARQL query to expand out the blank nodes.

Here are a couple suggestions that might improve the user experience:

1. When clicking a blank node (http://cor.esipfed.org/ont/?uri=_:b52E6848Ex68365), the UI says it's not found. This isn't surprising for those who have worked with blank nodes into SPARQL endpoints, but inexperienced users might not understand why. One way to handle this would be to rewrite the SPARQL query that backs a URL like this to display at least some information as to where the triple came from.



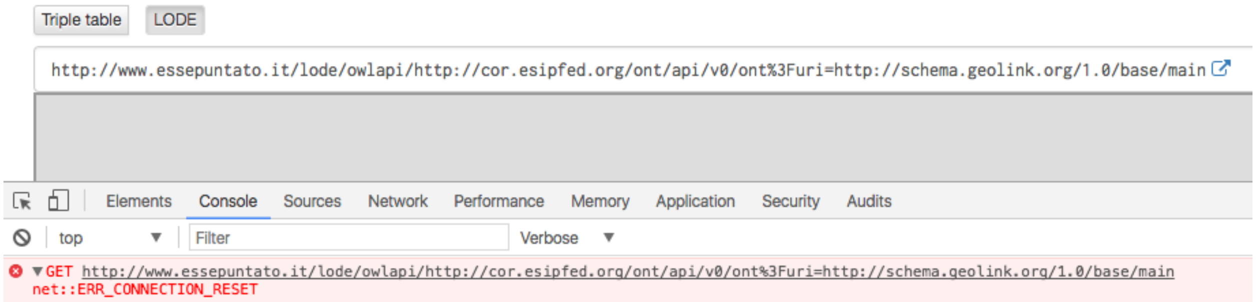
_:b52E6848Ex68365

URI not found

- No ontology registered with this URI
- No subject in the triple store with this URI

2. It was difficult to track down which ontology a set of returned triples came from when executing a term search. From Looking at the graph structure of the SPARQL endpoint, it looks like each ontology is loaded into a separate graph, and it might be helpful to add the graph IRI to the default display results to provide context for the triples.

3. When using LODE viewer on large ontologies, sometimes the service times out as it is dependent on the LODE server run at http://www.essepuntato.it/lode/owlapi/.

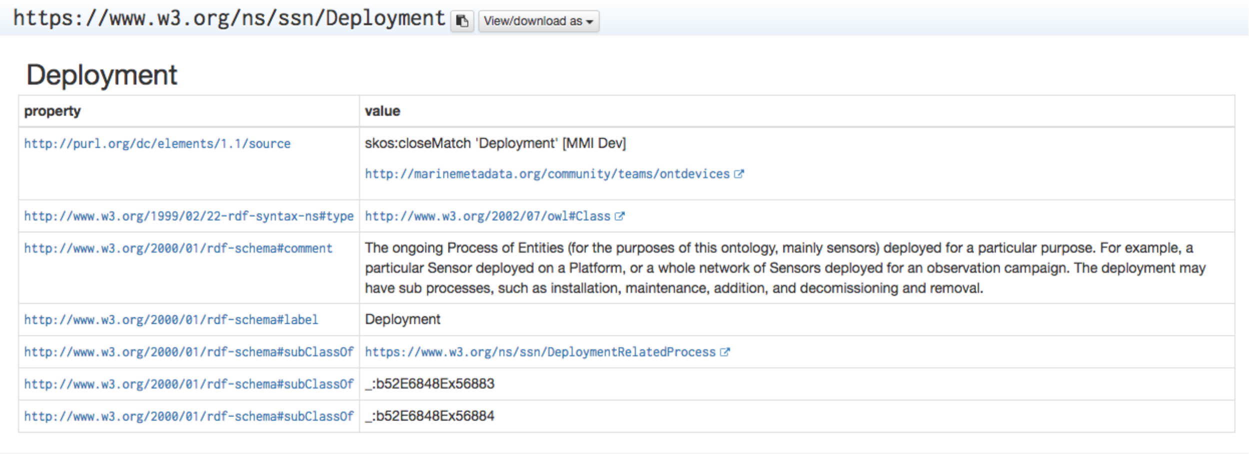


```
http://www.essepuntato.it/lode/owlapi/http://cor.esipfed.org/ont/api/v0/ont%3Furi=http://schema.geolink.org/1.0/base/main
```

```
▼GET http://www.essepuntato.it/lode/owlapi/http://cor.esipfed.org/ont/api/v0/ont%3Furi=http://schema.geolink.org/1.0/base/main
net::ERR_CONNECTION_RESET
```

This service, in the past, has been slow to respond with large ontologies, but, the code is open source on Github (https://github.com/essepuntato/LODE) and can be run locally so that it can be performance tuned and doesn't compete with other services taxing the www.essepuntato.it server.

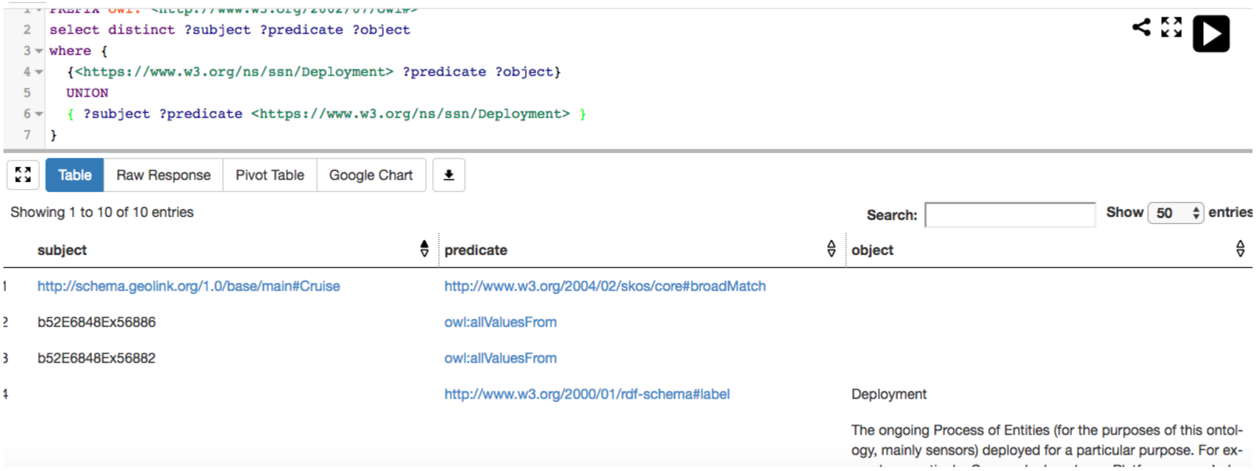
4. When a class is found from a term search, by clicking the class, you see the triples where the class is the subject of those triples. To help understand what might be related to this class, it would also be helpful to return triples where the class is also in the object part of the triple. For example, the Deployment class <https://www.w3.org/ns/ssn/Deployment> in the SSN ontology returns 7 triples.



Deployment

property	value
http://purl.org/dc/elements/1.1/source	skos:closeMatch 'Deployment' [MMI Dev] http://marinemetadata.org/community/teams/ontdevices
http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#Class
http://www.w3.org/2000/01/rdf-schema#comment	The ongoing Process of Entities (for the purposes of this ontology, mainly sensors) deployed for a particular purpose. For example, a particular Sensor deployed on a Platform, or a whole network of Sensors deployed for an observation campaign. The deployment may have sub processes, such as installation, maintenance, addition, and decomissioning and removal.
http://www.w3.org/2000/01/rdf-schema#label	Deployment
http://www.w3.org/2000/01/rdf-schema#subClassOf	https://www.w3.org/ns/ssn/DeploymentRelatedProcess
http://www.w3.org/2000/01/rdf-schema#subClassOf	_:b52E6848Ex56883
http://www.w3.org/2000/01/rdf-schema#subClassOf	_:b52E6848Ex56884

By using a UNION query, you can find 3 more triples about the Deployment class which helps a user visualize some of the mappings that can be done between classes in COR which is a really nice feature.



```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
select distinct ?subject ?predicate ?object
where {
  {<https://www.w3.org/ns/ssn/Deployment> ?predicate ?object}
  UNION
  { ?subject ?predicate <https://www.w3.org/ns/ssn/Deployment> }
}
```

Showing 1 to 10 of 10 entries

Search:

Show 50 entries

	subject	predicate	object
1	http://schema.geolink.org/1.0/base/main#Cruise	http://www.w3.org/2004/02/skos/core#broadMatch	
2	b52E6848Ex56886	owl:allValuesFrom	
3	b52E6848Ex56882	owl:allValuesFrom	
4		http://www.w3.org/2000/01/rdf-schema#label	Deployment
			The ongoing Process of Entities (for the purposes of this ontology, mainly sensors) deployed for a particular purpose. For ex-

Recommendation for both Repositories

One emerging idea in the semantics community has demonstrated to be effective for evaluating whether an ontology meets a specific set of needs. Research has shown that when an ontology provides a set of competency questions that it seeks to answer, it is easier for potential adopters to understand the concepts within the ontology preventing misunderstandings and misuse. It would be great if both repositories could add a place for ontology managers to provide the competency questions the ontology seeks to address. For more information on competency questions for ontologies, see the following papers:

Ren, Y., Parvizi, A., Mellish, C., Pan, J. Z., van Deemter, K., & Stevens, R. (2014). Towards Competency Question-Driven Ontology Authoring. Lecture Notes in Computer Science. Springer International Publishing. https://doi.org/10.1007/978-3-319-07443-6_50

Bezerra, C., Freitas, F., & Santana, F. (2013, November). Evaluating Ontologies with Competency Questions. 2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT). IEEE. https://doi.org/10.1109/wi-iat.2013.199

Use Case #3: Annotating Text

	The ontology portal includes an annotation tool.	**The annotation tool has a UI and/or API that enables users to access the annotation tool.**	**The annotation tool is able to accept text as input either by uploading a document or by entering text directly.**	**The annotation tool is able to identify terms in the text that match ontology concepts.**
COR	NO	NO	NO	NO

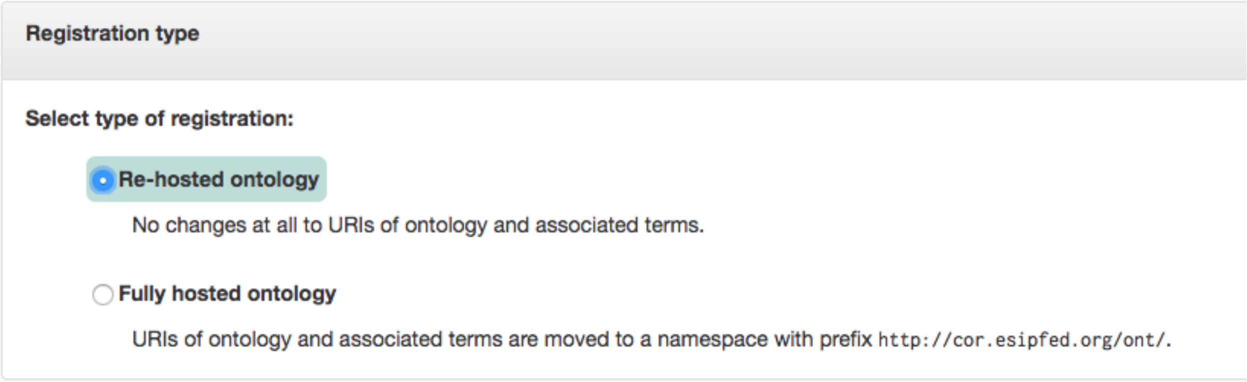
	The annotation tool is able to display the extracted terms along with the concepts/ontologies to which they could be mapped.	**The annotation tool is able to accept input from users accepting or rejecting suggested matches.**	**The annotation tool is able to mark up a text document with appropriate hyperlinks.**
COR	NO	NO	NO

The COR doesn't provide an annotation service and could not be evaluated on such basis. However, a user could write a SPARQL query returning just Classes in the repository that contain a word or word grouping. The Class itself would be the annotation to the text, but since this isn't explicitly provided by COR I hesitate to suggest that this capability is something all users, novices and experts could use and get the same result. Apart from implementing this as a service, an example SPARQL query would help users understand how this might be done.

Use Case #4: Editing, Extending and Releasing New Versions of an Existing Ontology

	There is a user authentication system.	A UI and/or API that enables users to upload ontology files (in a variety of formats?)	A UI that allows users to view an existing ontology.	A UI that allows users to edit an existing ontology.	A version control system.
COR	YES	YES	YES	YES	YES

COR's ontology adding, editing, and versioning are incredibly powerful, and host of useful options. First, ontologies can be added through either the UI or the Swagger API. From the UI, you are walked through steps of adding an ontology via URL or local file upload, and then have the nice option to decide if the ontology should be fully-hosted from the repository.



Registration type

Select type of registration:

● **Re-hosted ontology**

No changes at all to URIs of ontology and associated terms.

○ **Fully hosted ontology**

URIs of ontology and associated terms are moved to a namespace with prefix `http://cor.esipfed.org/ont/`.

This feature is great for users without the capability of hosting themselves with all the Linked Data services that are recommended for an ontology online - URIs for all elements that resolve and content negotiate along with human readable representations such as HTML. As these tools exist for reuse, offering this as a service from the COR makes it very attractive for users. Another great feature of COR is the ability to build a new vocabulary by creating terms and definitions in the UI without having to know all of the inner workings of semantics and SKOS or OWL. Creating new mappings between classes was easy and intuitive, providing a way to easily set the predicate for mappings.

In executing this use case, one issue came up related to ontology access control and the SPARQL endpoint. When adding an ontology, one of the fields that can be set is whether the ontology should be visible to the public or just by the ontology owner. When setting an ontology as only visible by the owner, the ontology still gets loaded into the SPARQL endpoint, and was available to the public for query. It looks like the ontology metadata is not available through the UI, but it seems misleading to have it loaded and queryable from the SPARQL endpoint. I hesitate to recommend this change be implemented because the SPARQL endpoint is foundational to the other aspects of the repository such as term searching. It's unclear how to effectively implement this and keep the SPARQL endpoint as the main delivery mechanism for other services. Or, maybe these other services are not available, even to the private ontology owner, until the ontology becomes public? This is a difficult problem to tackle, but users should be aware that their ontologies are exactly private.

Evaluator 2

Summary:

The ESIP Community Ontology Repository (ESIP COR) is a deployment of the Marine Metadata Initiative's Ontology Registry and Repository (MMI-ORR). The MMI-ORR is a web application through which you can create, update, access, and map ontologies and their terms. The COR software that underlies the COR portal has been significantly upgraded and is a beta release.

In general, the ESIP COR portal has a user interface that would prove daunting for non-developer and/or semantic technologists, but appears to have many more resources available to document how the portal can be used. The documentation is by no means easy to navigate, but does appear to be fairly complete albeit confusing in its apparent alpha status. In terms of longer term sustainability, there does appear to be funding to support further development by a currently funded project, but it is unclear how long that funding would last. Also, it's unclear whether there would be funding to support the much needed help with documentation and user interface improvement.

1. Use Case #1: Use of Semantics within Search Engine

a. Ability to complete workflow tasks

The web interface (http://cor.esipfed.org/ont/#/st/sensor) provides a straightforward term search interface (*sensor* used as test term). Results are presented in a list of RDF triples. The page clearly states the target of the term search (*Search applied on the subjects and object values of the semantic entities in the triple store*). The SPARQl "label" to the right of the search-term entry box and search button reveals the SPARQL query that is being applied. This ability to display of the SPARQL query was missed during the initial term-search test (see results of this well down in the API discussion).

The API landing page (http://cor.esipfed.org/ontapi/) provides access to a variety of capabilities with very clever use of **Try it out** buttons to demonstrate the functionality and underlying code associated with each flavor of API action. Under ontology, one finds a button labeled *Gets information about registered ontologies and terms*. There one finds a ***turi*** input box labeled *Use this parameter to exclusively make a "term request"*. Putting the term *sensor* in the box and clicking the **Try it out** button returns an empty result set.

At the very top of the API page is the statement: *The main ORR documentation is located at:* http://mmisw.org/orrdoc/ … where one finds the ORR User Manual. A left-hand column menu link takes you to *Querying via REST API*. There one is directed to *Querying via SPARQL API* for queries at the term level. At that location a further redirection leads to https://marinemetadata.org/community/teams/vocdev/orrioos where one sees various SPARQL examples offering term-based queries. The page includes that statement *You can exercise this functionality using the form at* http://mmisw.org/ont/sparql.html ... which responds with { "error" :

"invalid format=html"}. If one has RDF/SPARQL tools in hand and a fondness for regex, API queries can be undertaken at this stage against the endpoint: http://cor.esipfed.org/sparql.

If one is in search of more specific assistance, near the top of the API page is a URL for *let us know if you have questions or suggestions* (which takes you here: https://github.com/mmisw/mmiorr-docs/issues). Not clear why the link doesn't go here: https://github.com/mmisw/orr-portal/issues where 24 open and 82 closed issues regarding ORR-PORTAL are found. After some tinkering and noodling with Google searches, it was possible to send a query to the site development staff who responded with prompt courtesy. Having missed the SPARQl "label" to the right of the search-term entry box and search button in the web interface, it was not immediately clear that term search required SPARQL query manipulations. In retrospect, such should have been obvious, but was not during first-pass testing. The development staff's message allowed that: *Being aware that the SPARQL mechanism can be not as user-friendly in some scenarios, we are planning on adding some "shortcuts" through the main ORR API interface for typical operations like the "term search" visible in the portal*. and in a subsequent note: *I have advanced an implementation of a simplified interface for basic semantic queries through the main ORR endpoint*,

After all was said and done, a search of https://stackoverflow.com/ located an RDF/SQL wrapper module that supported applying the web interface's underlying SPARQL query against the repository's endpoint with success. That query being:

```
select distinct ?subject ?predicate ?object
where {
 ?subject ?predicate ?object.
 filter (regex(str(?subject), "sensor[^/#]*$", "i")
   || regex(str(?object), "sensor", "i"))
}
order by ?subject
```

Usability

- The web interface returns several hundred triples in which the object contains the term *sensor.*
- The API was a more complex undertaking, partly because first-pass testing of the web interface missed the essential fact that the underlying queries were actually SPARQL queries.

Quality and clarity of documentation

The web interface was straightforward to use. The API documentation takes a very interesting approach with useful examples and underlying code demonstrated by the ***Try it out*** buttons. Vagaries of using the API were noted above

Ontology ingest capability

Not tested … not part of the Workflow & underlying Requirements

Portal access

Access to the web interface was straightforward with vagaries as noted above related to using API.

Quality and usability of the portal's display.

The web interface worked as expected. The API's output could be used by a developer to create a useful web interface.

Search-capabilities

Both the web interface and the API do what they were designed to do.

Errors

No errors were encountered with the web interface.   Working through implementation of a query via the API requires the normal interactions with error handling found in the chosen programming environment. Did not encounter errors coming back the API implementation from the portal's endpoint other than the expected "invalid query format" when working out the nits and bits of any given interaction.

API support

The documentation pages were quite useful as noted above, notwithstanding the confusion about what constituted a "term" search and the somewhat convoluted path leading to examples of term-based SPARQL queries. As noted, a request for assistance evoked a very useful reply with prompt courtesy … and the follow-up pointed to improved "simplified" term-search functionality.

Maintainability

The ESIP-COR portal demonstrates the innovative nature of a "new" implementation. For example, a fresh approach to API documentation with demonstrations of both functionality and the underlying code. And on the other side of the coin, a very lean history of use by community members with widely varying levels of technical expertise. Probably most importantly, sustainability of support over the long term once project funds are expended comes into question.

Overall impressions

- The term-search web interface does what the use case required with a rather raw level of response (i.e., a long list of triples from the RDF store without any interpretative information).
- The same is true of the API, assuming a given developer has SPARQL tools and RDF experience in hand. Both web and API interfaces appear to rely on projects and funding aimed specifically at this set of capabilities for access to the underlying ontologies. One

worries about long-term support for such an environment, interesting and inventive as it is in many ways.

2. Use Case #2: Browsing a Portal for a Relevant Ontology

Ability to complete workflow tasks:

All tasks could be completed.

Usability:

- This portal seems quite usable for the scenario described in this use case in terms of the relative functionality of the basic search functions.
- The initial screen listing (presumably as what is seen is not labeled) all of the ontologies contained in the repository provides a useful and necessary context initially and when interpreting the search results.
- The filters that can be applied to the listing of ontologies are also very useful both initially and for search result analysis, although an explanation or definition of the terms would be helpful (see Quality & Clarity of documentation discussion below).
- The display of the search results is less helpful for this use case as it doesn't provide a very accessible mechanism for seeing why a particular ontology was selected from the search done. To interpret the results and choose to look at a particular ontology, this user would have to know to look at the predicate values in addition to the object text as the latter does not provide a context for where the terms appear (e.g., does not display the KWIC – key word(s) in context).
- There is no way to easily see from the search result view what class or classes either match or relate to the terms entered; nor are there ways to visualize where the classes or other properties sit in relation to others, such as seeing them in a tree structure, or a hierarchy. As a results, the usability of the searching functionality is diminished for this portal.

Quality & clarity of documentation:

- Given the seemingly straightforward workflow tasks associated with this use case, the documentation that would be useful is fairly clear in that there is reasonable explanation of the difference in creating a term search and a SPARQL search once you click to each option. You would need to click to those options rather than seeing any kind of hover help or other way of getting to more information from the first screen (After the welcome screen). Presumably, the user in this case would not necessarily know the advantages / disadvantages of each option in terms of the returned results or potential for a more precise search (which would be helpful, in an ideal world). As a result, clicking through to the Term Search page seems the natural next step.
- Once on the search page, the user in this case, would probably not understand the explanation for the difference between the basic search and the keyword search, and so may not necessarily know which to choose, nor why there are differences in the result

sets (i.e., a search for “sea water” in each brings back results from the basic search, but nothing at all from the keyword search. This user might well ask ????). The explanation for the keyword search is probably too semantic-geeky for this user, and even the basic search explanation is not expressed in ways that this user may fully understand. I suspect the result would be to just use the basic search.

- Some explanation of what to do or what is seen from the search results is not present or linkable, and would be very helpful for this user. Some examples of a search and ways to interpret the search result would be another helpful strategy for documentation.
- On the initial browse page, some explanation or definitions of the categories used for filtering or searching would be helpful as either hover help or some other quick way of understanding the meaning of the terms “resource type”, “ontology type”, and “global filter”, for example.

Ontology Ingest capability: n.a.

Portal access: Gaining portal access was quite straightforward.

Quality & usability of the portal’s display:

- Overall, the quality of this portal’s display was very good although more immediately accessible help for definitions and next steps could be very helpful for the new and returning user (i.e., by using “hover” help text for definitions, examples within the fields where input is possible that could be overwritten, or a single click link to information such as to the specific place in the “Getting Started” documentation.
- The visual display and the navigation of the portal’s pages is probably overly complex for the user in this case as the language and link structure for help seems much more geared to the developer, sysadmin or semantic geek. It does seem possible to get questions answered, but the user in this case would probably give up long before answering questions about what they were seeing in a search result set and why.

Search capabilities:

- The basic search seemed to be a straightforward full text search, and this seemed to work fairly well as the search results that were checked had the terms used somewhere in the metadata.   There didn’t seem to be a great deal of precision operating since there were no limiting parameters applied, but the results seemed adequate for this kind of search.
- The keyword search did not return any results at all when the same term (sea water) was submitted; nor did a list of other terms such as “sea-water” and “sea water return any results, so this did not appear to be working as what one would normally consider a keyword search.

Errors:

- There were not always responses to inputs, e.g., a keyword search returned nothing at all, so not clear if there was an error or what the error would be.
- There did seem to be avenues for finding out more about the various options for search and other functionality, but the navigation to the documentation would be difficult for anyone but a persistent developer or semantic-geek to find.

API support: Didn't use the API for this use case, but in reading the documentation on the Introduction and Getting Started pages, the link to the "Contact us" did not go anywhere, so some other path would have had to be found.

Maintainability:

See API Support above and Recommendation section below.

Overall impression: This use case was satisfied technically, but in order to be a very efficient service for this user, it would probably require more semantic technology knowledge on the part of the user described in this use case, or more obvious on-screen documentation and user help / support than is currently present.

3. Use Case #3: Annotating Text

Ability to complete workflow tasks:

There did not seem a way to use this portal for this use case unless the "text" file would be in "table" format (meaning a CSV file, presumably?).

Usability:

Quality & clarity of documentation:

Ontology Ingest capability:

Portal access:

Quality & usability of the portal's display:

Search capabilities:

Errors:

API support:

Maintainability:

Overall impression: This use case is not applicable to this portal.

4. Use Case #4: Editing, Extending and Releasing New Versions of An Existing Ontology

Ability to complete workflow tasks:

- This portal seemed to work fairly well for the 4th use case although steps 14 and 15 of the workflow instructions could not be completed. Otherwise, all steps were relatively straightforward.

Usability:

- The requirements for the use case were all met.
- There could, conceivably, be quite a bit of confusion on the part of the user with respect to the differences between an ontology and a vocabulary. The one example vocabulary in the portal includes metadata that has a resource type of “parameter” and asks for an “ontology creator”. The data is shown in a table format, but is not viewable / downloadable as a CSV or other table format.

Quality & clarity of documentation:

- The documentation for the tasks associated with this use are confusing. What is the difference between a “vocabulary” and an “ontology”? The explanation at: http://mmisw.org/orrdoc/vocab/import/ refers to vocabularies, but the screen insert here, and the actual screens of the functional pages talk about creating and importing “ontologies”.
- The distinction between a vocabulary and an ontology needs to be more clearly spelled out with explanations and examples wherever there is discussion about the associated tasks in the MMI ORR documentation.

Ontology Ingest capability:

- Seemed fairly straightforward.
- Is there no way to delete an ontology that is being tested? Could not find a way to delete.

Portal access: No problems accessing the portal.

Quality & usability of the portal’s display:

- The display for this portal is not designed for the faint of heart from a semantic technology point of view. The main screen (after the landing page from the “Access the COR” choice) is complex, but seems inclusive of most tasks that the user in this case would want to accomplish.
- The use of hover-over help language on the http://cor.esipfed.org/ont/#/ page would certainly be helpful for the first user in the scenario, and very much advisable for those users who would want to approach their tasks of editing vocabularies and updating ontologies using a more user-friendly interface.
- It was very helpful to immediately see the versions of the ontologies visible although the ISO format for reading them was difficult to see on the screen with all of the descriptive categories present on the screen (e.g., URI, Name, etc.). Perhaps narrowing the author field in order to see more / all of the version field values would be more helpful.

Search capabilities: n.a.

Errors:

- As there are definitely opportunities for error-making as part of the workflow for this use case, it is very helpful to have an option for owner-only views of what has been done with the option at any time of making public the submissions of ontologies (and presumably, vocabularies although that was not done as part of the exploration of this use case).

- The Contact Us screen did work on the browse page which is helpful.

API support: n.a.

Maintainability:

See Recommendation section below.

Overall impression:

- This use case seemed to be the most appropriate and successful for this portal by comparison to the other use cases.
- The documentation for using the portal is fairly extensive, on target, for the most part given the tasks associated with this use case.
- Navigation between sub or overall sections of the documentation is complicated, however, and one could easily get lost if exploring possible approaches to tasks, or investigating puzzling responses to terms, tasks, activities.

Evaluator 3

Summary of COR

COR is an adaptation of an ontology registry and repository originally developed through the Marine Metadata Initiative. The documentation has been partially converted, through many traces of the MMI origins remain. The MMI ORR was an innovative product, probably a bit ahead of the needs of the target community when it was developed, and therefore did not have a large impact. Importantly it does not have any significant user or developer community around it. Nevertheless, the deployment instructions in the documentation appear to be quite straightforward, and give a clue to the underlying architecture, which appears to largely re-use standard components which are mostly well maintained.

COR displays triples rather exuberantly, which is reassuring to an RDF tyro but will be confusing to users who have only used higher level interfaces and notations, and likely to be daunting to users new to ontologies. The UI is much more modern, with little presentation ‘chart junk’ (see Tufte), but also more austere. The use of an external service (LODE) to render user-friendly documentation is sensible, and a similar approach might be used for graphical visualization - e.g. based on VOWL. There is a faceted browse, and search functionality is very simple (though effective) - based on string comparisons of all subjects and objects in triples (as shown in the SPARQL displayed through the mouse-over). The technical basis of COR in RDF is transparent, and it does not support any ontology formalizations other than RDFS/OWL. This is unlikely to be restricting in practice since most ontologies of interest to the ESIP community are either developed in OWL or can easily be converted from other representations. The UI has a few minor annoyances - the main one being that the list of triples auto-scrolls when moused over. Probably could be easily fixed. However, human-friendly rendering and graphics are limited/non-existent.

The COR API is documented using OpenAPI/Swagger, which is pleasing, though it has some limitations. In particular where some parameters can only be used in certain combinations (e.g. predicate + subject | object, containing + in) this is only explained in the text description. Also, as well as the form for ‘trying it out’ some example queries would be helpful.

Some of the interface functionality appears to depend on OMV metadata, harvested from within a submission. However, this requirement is not very obvious during submission or in the documentation. (By comparison, LOV throws a warning if the expected VOAF, VANN and DC metadata is not detected early in the submission process, allowing you to go back and fix it before completing submission.) Perhaps the OMV dependence is historic, and now obsolete?

Detailed comments arising from use-case evaluation

Use of semantics within search engines

The search capability in both repositories is limited to partial string text matching. Thus neither appears to address the specific scenario described in the use case, which is to deliver a _ranked_ list of results. Where the use case is to ‘semantically match’ terms, then this is actually limited to matching strings in either the resource URI or within the value of one of its labels.

COR returns a set of RDF triples, showing all kinds of RDF resources (including individuals). There are more search options, including finding resources related to another resource using a specified predicate. However, it does not appear to be possible to combine text-search and predicate related in a single query. The COR API documentation is based on the Swagger standard. However, the documentation should clarify which query parameters can be used in combination.

Browsing for a relevant ontology

Ontology browsing is effective in both repositories.

In practice the use case specified in the instructions is a variation on the searching use-case already discussed. COR manages all RDF types.

Annotating text

This specific function is not supported by COR.

Editing/extending/releasing new versions

Uploading using COR was straightforward. However, there are only two options for display of the ontology content: a list of triples, and a HTML rendering using an external tool (LODE). There is no tree-based class hierarchy, for example.

Editing/extending ontologies in both platforms is supported by uploading a replacement for the whole ontology. COR also supports adding a new term, but neither platform allows changing resources (terms) in place.

“Releasing” a version of an ontology is supported in both portals by changing the viewing restrictions/visibility from private/owner to public. Both repositories also have a separate ‘status’ flag - though the values for these are different and do not appear to correspond to any external enumeration of values.

Appendix: Response

Table 1

Line	Original text	Our note	Response Type
42	does not provide term search through its Swagger API	This feature was recently added (https://github.com/mmisw/orr-portal/issues/112).	Update
47	we noticed the link to example SPARQL queries did not resolve at the time of writing this report…I note that this example might be already available at https://marinemetadata.org/community/teams/vocdev/orrioos, but I was unable to resolve the URL for this report.	There was an extended window during the review period when the MMI web site was down. The site was successfully accessed by another reviewer at another time, and is available now and for the foreseeable future.	Update
67	you can view the ontology's metadata fields and then in a section labeled data, the ontology is represented in RDF triples	Yes, this is correct for the case of "uploaded" ontologies. The interface is different and much more user friendly in the case of so-called "ORR vocabularies" and "ORR mappings."	Clarification
148	ontologies can be added through either the UI or the Swagger API	To be clear, the API used for general repository operations is the ORR API; it provides Create-Replace-Update-Delete capabilities for ontologies, users, organizations, and triple store maintenance. This API is defined using the OpenAPI (aka Swagger) specification; and its documentation page is powered by the Swagger-UI tool.	Clarification
163	When setting an ontology as only visible by the owner, the ontology still gets loaded into the SPARQL endpoint, and was available to the public for query.…This is a difficult problem to tackle, but users should be aware that their ontologies are exactly private.	The fact that the ontologies are not truly private is documented at http://mmisw.org/orrdoc/faq/#how-can-i-view-ontologies-in-the-repository. (We agree this documentation needs to be more visible.)	Clarification
201	There one finds a turi input box labeled Use this parameter to exclusively make a "term request". Putting the term sensor in the box and clicking the Try it out button returns an empty result set.	This is the correct result for a 'term request' (the name of this parameter is now *tiri*, due to recent upgrades). This query expects the complete IRI for the term (as opposed to a term search).	Clarification
207	At that location a further redirection leads to https://marinemetadata.org/community/teams/vocdev/orrioos where one sees various SPARQL examples offering term-based queries.	Work is in progress to make the documentation at http://mmisw.org/orrdoc/ much more self-contained, in particular regarding the SPARQL topic and examples.	Update
304	nor are there ways to visualize where the classes or other properties sit in relation to others, such as seeing them in a tree structure, or a hierarchy	Full-fledge display/visualization of ontologies has traditionally been out of scope in the ORR effort.	Information

309	Given the seemingly straightforward workflow tasks associated with this use case, the documentation that would be useful is fairly clear in that there is reasonable explanation of the difference in creating a term search and a SPARQL search once you click to each option. You would need to click to those options rather than seeing any kind of hover help or other way of getting to more information from the first screen (After the welcome screen). Presumably, the user in this case would not necessarily know the advantages / disadvantages of each option in terms of the returned results or potential for a more precise search (which would be helpful, in an ideal world). As a result, clicking through to the Term Search page seems the natural next step	If this is an issue, can the issue be clarified? Perhaps the user is suggesting that the hover help documentation be added?	Request for Info
341	The visual display and the navigation of the portal's pages is probably overly complex for the user in this case as the language and link structure for help seems much more geared to the developer, sysadmin or semantic geek.	Does this observation also refer to the main ontology list page?	Request for Info
362	API support: Didn't use the API for this use case	This seems to be suggesting there is no REST API documentation, but there is a "REST API" documentation section of the help at http://mmisw.org/orrdoc/api/.	Clarification
393	There could, conceivably, be quite a bit of confusion on the part of the user with respect to the differences between an ontology and a vocabulary.	Documentation has recently been updated (http://mmisw.org/orrdoc/semweb/) to indicate that, in general, both terms can be used interchangeably. And, within the context of the ORR system itself, to indicate that "vocabulary" is used to refer to the ontologies created with the "ORR vocabulary tool."	Update
409	Is there no way to delete an ontology that is being tested? Could not find a way to delete.	The option for "ontology unregistration" is only available for users with the "admin" role.	Information
428	it is very helpful to have an option for owner-only views of what has been done with the option at any time of making public the submissions of ontologies	We are not sure if this is an affirmation or a request. The COR does have views of owner-only submissions, and the ability to make private submissions public.	Request for Info
459	The use of an external service (LODE) to render user-friendly documentation is sensible, and a similar approach might be used for graphical visualization - e.g. based on VOWL.	Both LODE and VOWL have preliminarily been incorporated via "iframes" (pointing to the external services). (VOWL was recently disabled due to technical issues.) Ideally these services should be deployed on servers that the particular ORR instance (e.g, the COR) team can manage.	Information
476	Some of the interface functionality appears to depend on OMV metadata, harvested from within a submission.	The metadata can be harvested if it exists within a submission, but most often comes from user-entered metadata (in the metadata interface), which is then encoded using the OMV standard. (This enables users to revise their metadata either through the metadata interface, or in the ontology itself if it is downloaded, modified, then re-submitted.)	Clarification

480	Perhaps the OMV dependence is historic, and now obsolete?	It is historic, but not obsolete, as it still works and supports a dual workflow. Also, to date there has been no single standard to serve as a replacement, though one is anticipated shortly. Of course, we agree more metadata vocabularies could be supported, but the support of an existing standard, albeit dated, should not be considered a weakness.	Clarification
484	The search capability in both repositories is limited to partial string text matching. Thus neither appears to address the specific scenario described in the use case, which is to deliver a _ranked_ list of results. Where the use case is to 'semantically match' terms, then this is actually limited to matching strings in either the resource URI or within the value of one of its labels.	As stated the first sentence is not correct for COR, which offers the SPARQL endpoint for enabling more complicated search. The evaluator may mean to describe the limitations of the UI search interface, but examples are given for using SPARQL to perform semantic matching.	Clarification
508	neither platform allows changing resources (terms) in place.	COR allows editing content in place for vocabularies created with the integrated "vocabulary tool" (via the "Create Vocabulary" button). We refer to these vocabularies as "ORR vocabularies". A similar editing capability is available for mappings created with the "mapping tool" (via the "Create Mapping" button).	Clarification
512	"Releasing" a version of an ontology is supported in both portals by changing the viewing restrictions/visibility from private/owner to public.	If an ontology is already public, a new version is released as soon as it is submitted.	Clarification
513	Both repositories also have a separate 'status' flag - though the values for these are different and do not appear to correspond to any external enumeration of values.	The possible status values are those from https://www.w3.org/2003/06/sw-vocab-status/note.html plus one or two additions. See also https://github.com/mmisw/orr-portal/issues/13#issuecomment-231433996	Clarification