 doi:10.20944/preprints202002.0326.v1

Serverless GEO labels for the semantic sensor web

Anika Graupner[1] and Daniel Nüst[1]

[1]Institute for Geoinformatics, University of Münster, Münster, Germany

Correspondence: Daniel Nüst (daniel.nuest@uni-muenster.de)

Abstract. As the amount of sensor data made available online increases, it becomes more difficult for users to identify useful datasets. Semantic web technologies improve discovery with meaningful ontologies, but the decision of suitability remains with the users. The GEO label provides a visual summary of the standardised metadata to aid users in this process. This work presents novel rules for deriving the information for the GEO label's multiple facets, such as user feedback or quality information, based on the Semantic Sensor Network Ontology and related ontologies. It enhances an existing implementation of the GEO label API to generate labels for resources of the Semantic Sensor Web. The prototype is deployed to serverless cloud infrastructures. We find that serverless GEO label generation is capable of handling two evaluation scenarios for concurrent users and burst generation. More real-world semantic sensor descriptions and an integration into large scale discovery platforms are needed to develop the presented solutions further.

1 Introduction

The amount of sensoric data captured and accessible today is ever increasing, not the least by the numerous sensing deviced of the Internet of Things (IoT), Smart Cities, and newest Earth observation satellites. The Group on Earth Observation's (GEO) Global Earth Observation System of Systems (GEOSS)[1] attempted to make near real-time environmental data and processing available to users in form of a Spatial Data Infrastructure (SDI, cf. introduction in Janowicz et al., 2010) based on OGC Web Services and standards[2]. The complexity of the task and the continued growth of sensor data means this system is, and probably will remain, in the making. To improve user recognition of geospatial datasets, to promote trust in datasets, and to assist users in the discovery of suitable datasets, the GEO label was developed (Lush, 2015) . Lush (2015) described the original GEO label's design, which comprises several *facets* to convey the most relevant information to users, and evalution process. The label as graphical representation for individual dataset in GEOSS allows users to compare complex metadata and objective and subjective quality information to make an informed dataset selection decision (Lush et al., 2013). For integration with geospatial catalogues and applications, the GEO label generation was encapsulated in a RESTful Web API[3], the *GEO label API*. This API creates labels in SVG format (Dahlström et al., 2011) (besides others) for provided metadata documents or for links to online documents. The format enables interactivity, such as popups or links for parts of the image. Nüst and Lush

All URLs in this documente were last checked on February 19, 2020.

[1]https://en.wikipedia.org/wiki/Global_Earth_Observation_System_of_Systems

[2]https://www.opengeospatial.org/docs/is

[3]https://geolabel.net/api.html

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 February 2020 doi:10.20944/preprints202002.0326.v1

(2019) extended the GEO label for the OGC Sensor Web Enablement[4] (SWE) to transfer the label's usefulness to standardised sensor data. The dynamic nature and size of environmental data require a scalable infrastructure for on-the-fly generation of labels. Therefore, we propose to deploy the GEO label API to cloud computing infrastructures.

The Semantic Web aims to provide machine-readable online data[5]. Janowicz et al. (2010) describe how SDIs can be enhanced with a transparent mapping to the Semantic Web and Sheth et al. (2008) introduce the Semantic Sensor Web (SSW), which connects the Semantic Web with the OGC Sensor Web Enablement. The SSW is a framework for enabling interoperability and meaningful data integration, processing, and reasoning. The metadata captured by ontologies in the semantic sensor web provides a promising source of information for a GEO label. In turn, the GEO label has the potential to improve data discovery in the vastness of geospatial sensor datasets, where specific challenges exist, such as no singular inventory or the dynamic structure of sensor webs (Jirka et al., 2009). Therefore, we propose to extend the GEO label for metadata from the SSW.

The **main contributions** of this work are the mapping of metadata from the semantic sensor web to the GEO label facets, a prototypical implementation of this mapping in a GEO label API implementation, and the prototype's evaluation in selected use cases in scalable cloud-based infrastructures.

In the remainder of this work, we first identify suitable sources of information in ontologies of the SSW and related ontologies. Then we evaluate existing GEO label API implementations and different cloud computing providers to identify suitable base software and cloud platforms for a prototypical implementation. The prototype performance is evaluated in selected usage scenarios. See section *Code and data availability* below for information about the software prototype, used test data, and online deployments of the prototype.

2 GEO label for the Semantic Sensor Web

The SSW's core ontology is the *Semantic Sensor Network Ontology* (SSN, Haller et al., 2019). The modular SSN was first evaluated for suitable fields, including the core ontology SOSA (Sensor, Observation, Sample and Actuator) and SSN's aligned modules. Then we extended the search to include ontologies often used in conjuction with SSN. Adding new alignments between SSN and other ontologies lies beyond the scope of this work. Table 1 shows the ontologies, classes, and properties we identified as suitable sources for the GEO label's facets. The used ontologies and prefixes are listed in Table 2.

3 Serverless GEO label generation

3.1 Serverless computing

Serveless computing allows developers to deploy custom service code in a shared infrastructure (Baldini et al., 2017). This infrastructure is maintained in a scalable way by a platform provider. A gateway is used to expose the GEO label generation functionality in a Web API. A serverless deployment of the GEO label API is very suitable because the generation is a relatively

[4]https://www.opengeospatial.org/node/698
[5]https://en.wikipedia.org/wiki/Semantic_Web

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 February 2020 doi:10.20944/preprints202002.0326.v1

small operation, stateless, and atomic. For the proof-of-concept, cost considerations are out of scope. All deployments and evaluations were conducted within the free tier of the following service providers to demonstrate aplicability of the prototype in more than one platform: Google Cloud Run (GCR) and Amazon Web Services (AWS) Lambda[6].

3.2 GEO label API implementation

The GEO label API was implemented in two software projects, one in Java and one in PHP[7]. In this work, the Java-based implementation is used, because PHP is not supported by the serverless computing providers and the PHP project is not maintained anymore. The rendering of the GEO label is based on an SVG template file. Labels are generated using the template file accourding to XPath expressions (Clark and DeRose, 1999), which detect the presence of certain elements in a provided XML document. To use XPath, the RDF graph must be serialised in RDF/XML. Both implementations support a bespoke JSON-based configuration file format, which allows to update the transformations of metadata documents to labels without touching source code changes, and deploying these updates to instances without updating the installation. A new transformation file was created to realise the conceptual mapping described above[8]. The file is activated when the implementation is provided an RDF/XML document (i.e., the XPath `boolean(/*[local-name()='RDF'])` testing the root element evaluates to true). The implementation of hoverover and drilldown features lies beyond the scope of the proof-of-concept implementation. Table 1 shows excerpts of the XPaths realising the conceptual mapping. The test data[9] was created based on the example data for the SSN vocabulary[10], which was converted to RDF/XML using two different online converters for two varying serialisations into RDF/XML[11]. The following example illustrates the reason for the complexity of the XPaths due to different serialisations of triples from an RDF graph in RDF/XML. In the first one, `rdf:resource` attributes are used to define elements at one level, whereas the second one uses the class names as XML elements and nests the objects.

Observation, converted with MyBluemix Converter:

```
<rdf:Description rdf:about="http://example.org/data/iceCore/12#observation">
  <sosa:hasSimpleResult rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">42</sosa:hasSimpleResult>
  <sosa:observedProperty rdf:resource="http://example.org/data/iceCore/12#CO2"/>
  <prov:wasAssociatedWith
rdf:resource="http://example.org/data/Organization/exampleOrganization"/>
  <rdf:type rdf:resource="http://www.w3.org/ns/prov#Activity"/>
  <rdf:type rdf:resource="http://www.w3.org/ns/sosa/Observation"/>
</rdf:Description>

<rdf:Description rdf:about="http://example.org/data/Organization/exampleOrganization">
  <foaf:name>Example Organization</foaf:name>
```

[6] https://cloud.google.com/run/; https://aws.amazon.com/lambda/
[7] https://geolabel.net/implementations.html
[8] See transformation file source at https://github.com/nuest/GEO-label-java/blob/master/server/src/main/resources/transformations/transformerSSNO.json.
[9] https://github.com/nuest/GEO-label-java/tree/master/testdata
[10] https://www.w3.org/TR/vocab-ssn/#examples
[11] Easy RDF Converter (ERC), http://www.easyrdf.org/converter and MyBluemix RDF Validator und Converter (MBC), http://rdfvalidator.mybluemix.net/.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 February 2020 doi:10.20944/preprints202002.0326.v1

```
  <rdf:type rdf:resource="http://www.w3.org/ns/prov#Organization"/>
  <rdf:type rdf:resource="http://www.w3.org/ns/prov#Agent"/>
</rdf:Description>
```

Observation, converted with MyBluemix Converter:

```
<sosa:Observation rdf:about="http://example.org/data/iceCore/12#observation">
  <rdf:type rdf:resource="http://www.w3.org/ns/prov#Activity"/>
  <prov:wasAssociatedWith>
    <prov:Agent rdf:about="http://example.org/data/Organization/exampleOrganization">
      <rdf:type rdf:resource="http://www.w3.org/ns/prov#Organization"/>
      <foaf:name>Example Organization</foaf:name>
    </prov:Agent>
  </prov:wasAssociatedWith>
  <sosa:observedProperty>
    <rdf:Description rdf:about="http://example.org/data/iceCore/12#CO2">
      <ssn:isPropertyOf rdf:resource="http://example.org/data/iceCore/12"/>
    </rdf:Description>
  </sosa:observedProperty>
  <sosa:hasSimpleResult rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">42</sosa:hasSimpleResult>
</sosa:Observation>
```

In GCR, the API can be deployed in a container. This allows to run the whole GEO label API with the existing Java Servlet[12]. AWS Lambda cannot run the whole Java Servlet application. Therefore a subset of the GEO label API were implemented with a bespoke request handling class (see module `lambda` of the source code). This handler exposes the existing internal methods to generate SVGs based on URLs to metadata documents provided by the API caller. The API, i.e., the request parameters and allows HTTP methods, are configured in Amazon API Gateway. Figure 1 shows a GEO label rendered by the prototype implementation developed as part of this work.

3.3 Performance evaluation

Two usage scenarios were evaluated with an Apache JMeter[13] scripted test plan[14]. For all API queries, the the URL of the example RDF serialisation file `MBC_all_factes_available_ip68smartsensor.rdf` hosted on GitHub is passed via the `GET` request query parameter `metadata` to the SVG-geerating endpoint of the respective API deployments. The responses are deemed successful if the HTTP status code is 200 (“OK”) and the content type is `image/svg+xml`.

For both serverless computing providers, the default configurations were used for the evaluations. GCR allows to configure the number of containers, the number of parallel requests handled by one container, and the required minimum response time.

[12] https://en.wikipedia.org/wiki/Java_servlet

[13] http://jmeter.apache.org/

[14] JMeter test plan file `GEO_Label_API.jmx` and the result files are available online at https://github.com/nuest/GEO-label-java/tree/master/misc/JMeterTests.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 February 2020 doi:10.20944/preprints202002.0326.v1

Figure 1. GEO label based on SSW sensor metadata, rendered by the GCR deployment, with all eight facets fulfilled; source URL: https://glbservice-nvrpuhxwyq-ew.a.run.app/glbservice/api/v1/svg?metadata=https://raw.githubusercontent.com/nuest/GEO-label-java/master/server/src/test/resources/ssno/ERC_all_factes_available_ip68smartsensor.rdf.

The GCR deployment used zone `europe-west1` with 256 Mebibyte working memory and 1 CPU, at a concurrency setting of 80. AWS Lambda starts more instances of a Lambda function as needed, limited by a configurable concurrency parameter (value 1000) for the number of running functions in the used region `eu-central-1`. The working memory on AWS set to 1 Gibibyte with the default values for scaling[15].

Scenario A evaluated a geospatial catalogue service with 1000 users, whose browsing of the browser-based catalogue user interfaces results in 1 request per second each. Figures 2 and 3 show the response times during the test execution for GCR and AWS Lambda respectively. Both graphs show a cold start effect, where additional resources are activated over time and reduce the average response times to under 500 milliseconds for both platforms and only few requests taking over 1 second to complete. These results imply that the serverless label generation is suitable for interactive use since the response times below 1 second do not interrupt the user's traing of though (cf. Nielsen, 1993).

Scenario B tested the batch generation of labels, where a single operator of a sensor catalogue wants to generate 100 labels at once. Here the overall time for processing all requests is measured. The operations were repeated 5 times. For GCR, this lead to failures due to the memory limit. The test could be completed with a memory of 1 Gibibyte and 2 CPUs per container instance (data file: `GCR_Scenario_3_V2`). These additional resources could stem from the overhead of the full Java Servlet compared to the more minimal Lambda function handler. With the increased resources, the durations started with 45 seconds in the first run and went down (via 39, 18, 15) to only 3 seconds for the fifth repetition. For AWS Lambda, the processing takes 8 seconds on the first run and falls to 1 second less for the second to fifth repetitions (data file: `AWS_Scenario_3_V2`). These effects are likely achieved by a combination of scaling and the built-in caching of the GEO label API.

[15] https://docs.aws.amazon.com/lambda/latest/dg/scaling.html

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 February 2020 doi:10.20944/preprints202002.0326.v1

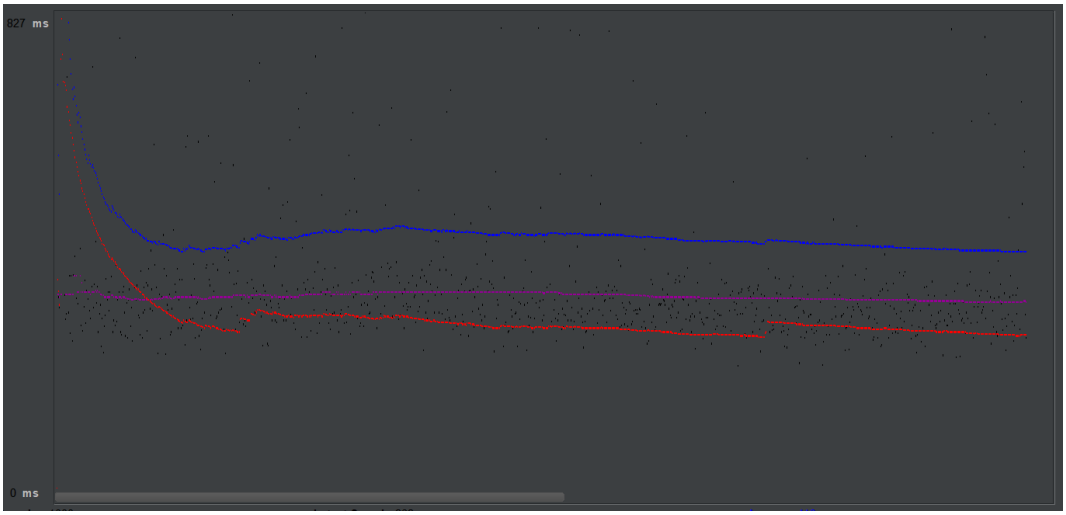

Figure 2. JMeter plot of response time for GCR deployment under scenario A, violett: median, blue: average, red: standard deviation, black: observed value; result data file: `GCR_Scenario_2_V1`.

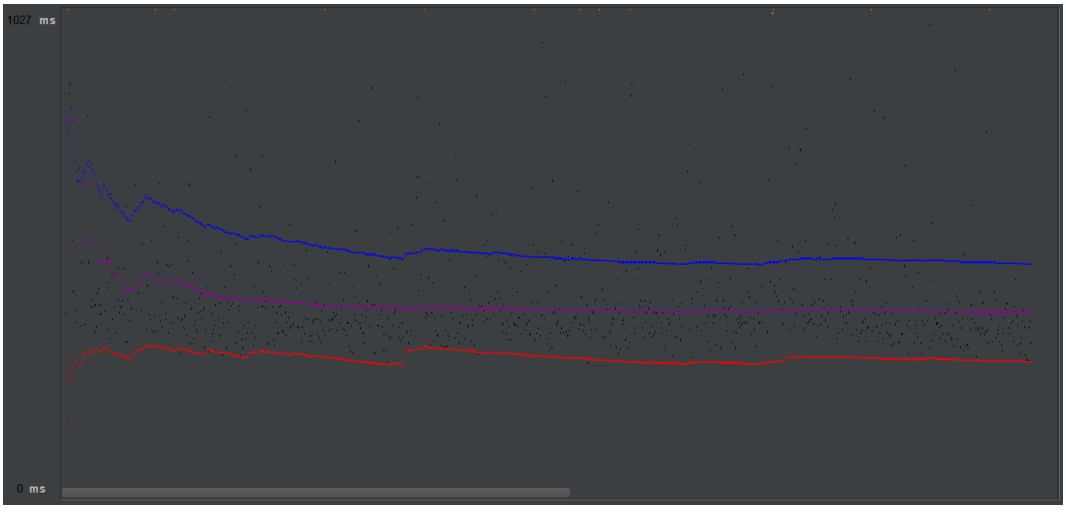

Figure 3. JMeter plot of response time for AWS deployment in scenario A, violett: median, blue: average, red: standard deviation, black: observed value; result data file: `AWS_Scenario_2_V1`.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 February 2020 doi:10.20944/preprints202002.0326.v1

A third test scenario with 1000 parallel request could not be completed by either platform with the maximum available hardware configurations. The error messages (`Connection reset` and `SSL handshake terminated`) hint at the services blocking the large number of parallel requests, so that more powerful, and costly, deployments would be needed. Reducing the number of parallel requests eventually lead to successful scenario execution at 600 requests in 51 seconds for GCR and 300 requests in 9 seconds for AWS Lambda (data file: `GCR_Scenario_4_2_V3` respectively `AWS_Scenario_4_2_V4`).

4 Conclusions

The GEO label's goal to improve data discovery by providing a visual overview of available information in machine-readable metadata has been transferred to the Semantic Sensor Web. Data sources for all facets could be found. Compared to the centrally managed instances and industry-driven OGC standards of the original GEO label, no distinction between metadata given by providers and by (third-party) commenting servers is necessary. The document-based approach using RDF/XML could be built quickly on the existing implementation, but it does not leverage the power of the Semantic Web, e.g., reasoning on dynamically built graphs and aligned ontologies. The complications of relying on serialisations into RDF/XML are already clear for some facets, e.g., producer profile. A different approach based on native semantic web technologies, such as SPARQL (Harris and Seaborne, 2013), could also allow to support the notion of "half-filled" facets, i.e., availability at a higher level, e.g. by measuring the distance between resoures in a graph. The created mapping is limited by the conceptual approach of creation, which would ideally be combined with a survey on commonly used ontologies and classes, but there is a lack of a large body of public sensor metadata in SSNO format. A larger test set could also improve the scoping of the facet data sources, e.g. improve if a comment is actually given about a relevant sensor resource, and not on some minor part of the RDF graph. The serverless platforms proofed suitable for the realistic test scenarios, though naturally the used free tiers have limits. It also becomes clear that the different costs models and configurations make serverless solutions harder to compare. A strictly cost-based comparison of scenarios tuned to deliver similar performance could remedy these challenges. The data could point to an advantage for the reduced implementation of the AWS Lambda functions compared to the full Java Servlet running in containers on GCR, though both showed scaling mechanisms to be effective. Finally, the usefulness of the GEO label remains to be demonstrated in broad deployments with many users and extensive user studies. With the practical solutions for label generation introduced in this work, the actual spreading of labels will require leading organisations to add labels to their widely used geospatial catalogues. In the meantime, a bottom-up approach with client-side label integration could provide the benefits to interested users (cf. Nüst et al., 2019).

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 February 2020 doi:10.20944/preprints202002.0326.v1

Code and data availability. The software and all examples files are available in the GitHub repository https://github.com/nuest/GEO-label-java. The code repository includes details about configuration and deployment of the used serverless compute platforms, and for local execution with a Docker container. This work is based on release `0.3.0`, which is also archived on Zenodo, see Nüst and Graupner (2020). The GEO label API endpoints of the serverless deployments are as follows:

- Google Cloud Run: https://glbservice-nvrpuhxwyq-ew.a.run.app/glbservice/api/v1
- AWS Lambda: https://6x843uryh9.execute-api.eu-central-1.amazonaws.com/glbservice/api/v1

Author contributions. AG created the SSNO mapping, implemented the prototype, deployed the serverless applications, and conducted the performance evaluation. DN conceived the idea, contributed to the prototype, and wrote the article.

Competing interests. The authors declare no competing interests.

Disclaimer. No potential conflict of interest was reported by the authors.

 doi:10.20944/preprints202002.0326.v1

References

Baldini, I., Castro, P., Chang, K., Cheng, P., Fink, S., Ishakian, V., Mitchell, N., Muthusamy, V., Rabbah, R., Slominski, A., and Suter, P.: Serverless Computing: Current Trends and Open Problems, in: Research Advances in Cloud Computing, edited by Chaudhary, S., Somani, G., and Buyya, R., pp. 1–20, Springer, Singapore, https://doi.org/10.1007/978-981-10-5026-8_1, 2017.

Clark, J. and DeRose, S.: XML Path Language (XPath), Version 1.0, W3C Recommendation, http://www.w3.org/TR/xpath, 1999.

Dahlström, E., Dengler, P., Grasso, A., Lilley, C., McCormack, C., Schepers, D., and Watt, J.: Scalable Vector Graphics (SVG) 1.1 (Second Edition), W3C Recommendation, http://www.w3.org/TR/SVG/, 2011.

Haller, A., Janowicz, K., Cox, S. J. D., Lefrançois, M., Taylor, K., Le Phuoc, D., Lieberman, J., García-Castro, R., Atkinson, R., and Stadler, C.: The modular SSN ontology: A joint W3C and OGC standard specifying the semantics of sensors, observations, sampling, and actuation, Semantic Web, 10, 9–32, https://doi.org/10.3233/SW-180320, 2019.

Harris, S. and Seaborne, A.: SPARQL 1.1 Query Language, W3C Recommendation, https://www.w3.org/TR/sparql11-query/, 2013.

Janowicz, K., Schade, S., Bröring, A., Keßler, C., Maué, P., and Stasch, C.: Semantic Enablement for Spatial Data Infrastructures, Transactions in GIS, 14, 111–129, https://doi.org/10.1111/j.1467-9671.2010.01186.x, 2010.

Jirka, S., Bröring, A., and Stasch, C.: Discovery Mechanisms for the Sensor Web, Sensors, 9, 2661–2681, https://doi.org/10.3390/s90402661, 2009.

Lush, V.: Visualisation of quality information for geospatial and remote sensing data: providing the GIS community with the decision support tools for geospatial dataset quality evaluation, Ph.D. thesis, Aston University, https://research.aston.ac.uk/en/studentTheses/visualisation-of-quality-information-for-geospatial-and-remote-se, 2015.

Lush, V., Bastin, L., and Lumsden, J.: Developing a geo label: providing the gis community with quality metadata visualisation tools, Proceedings of the 21st GIS Research UK (GISRUK 3013), Liverpool, UK, pp. 3–5, https://www.geos.ed.ac.uk/~gisteac/proceedingsonline/GISRUK2013/gisruk2013_submission_44.pdf, 2013.

Nielsen, J.: Response Times: The 3 Important Limits, https://www.nngroup.com/articles/response-times-3-important-limits/, 1993.

Nüst, D. and Graupner, A.: nuest/GEO-label-java: Release 0.3.0, https://doi.org/10.5281/zenodo.3673870, 2020.

Nüst, D. and Lush, V.: A GEO label for the Sensor Web, AGILE Short Papers, AGILE, Lisbon, Portugal, https://doi.org/10.31223/osf.io/ka38z, 2019.

Nüst, D., Lohoff, L., Einfeldt, L., Gavish, N., Götza, M., Jaswal, S. T., Khalid, S., Meierkort, L., Mohr, M., Rendel, C., and van Eek, A.: Guerrilla Badges for Reproducible Geospatial Data Science (AGILE 2019 Short Paper), AGILE Short Papers, AGILE, Limassol, Cyprus, https://doi.org/10.31223/osf.io/xtsqh, 2019.

Sheth, A., Henson, C., and Sahoo, S. S.: Semantic Sensor Web, IEEE Internet Computing, 12, 78–83, https://doi.org/10.1109/MIC.2008.87, 2008.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 February 2020 doi:10.20944/preprints202002.0326.v1

Table 1. GEO label facets' data sources in the Semantic Sensor Web

Facet	Availability & hoverover text	XPath
Producer profile	One SSN object (e.g., `sosa:Sensor`) has a responsible `prov:Person` or `prov:Organization` associated using `prov:wasAttributedTo` or `prov:wasAssociatedWith`; hoverover has name(s) using `foaf:name`.	`//*[rdf:Description[rdf:about= //prov:wasAssociatedWith/rdf:resource or rdf:about= //prov:wasAttributedTo/rdf:resource ]/rdf:type[rdf:resource= "http://www.w3.org/ns/prov#Organization" or .../prov#Person"]]` (partial expression)
Producer comments	A comment provided *within* a sensor description is coming from the procuder; hoverover has count and excerpt of comments.	`//*[rdfs:comment]`
Lineage information	At least one usage of `sosa:Procedure`, either directly, or for an SSN object via one of `ssn:implements`, `ssn:implementedBy`, or `sosa:usedProcedure`; hoverover text shows number and type of a procedures' `ssn:Input` and `ssn:Output`	`//*[ssn:implements or ssn:implementedBy or sosa:usedProcedure or sosa:Procedure or rdf:resource= 'http://www.w3.org/ns/sosa/Procedure']`
Standards compliance	If URIs include `w3.org`, at least one ontology is by the W3C as the authority for the Semantic Web.	`//*[contains(*, 'w3.org')]`
Quality information	An `ssn:System` has (a) the property `ssn-system:qualityOfObservation` or (b) at least one property detailing a sensors's capabilities (e.g., accuracy, battery lifetime) identified via the relations `ssn-system:hasSystemProperty`, `ssn-system:hasOperatingProperty`, or `ssn-system:hasSurvivalProperty`; hoverover has names of available properties.	`//*[ssn-system:hasSystemProperty or ssn-system:hasOperatingProperty or ssn-system:hasSurvivalProperty or ssn-system:qualityOfObservation]`
User feedback	A SSN object has a property `duv:hasUserFeedback` connecting it with a `duv:UserFeedback` or `duv:RatingFeedback`; hoverover has counts of feedbacks and an statistics of categorial ratings.	`//*[duv:hasFeedback/duv:UserFeedback[ not(prov:qualifiedAssociation)] or duv:hasFeedback/duv:RatingFeedback ]` (partial expression)
Expert reviews	A user feedback (see above) is qualified with a relation `prov:qualifiedAssociation` decribing the author's role with `prov:hadRole` as an expert, a role that is not specified within PROV-O but can be the literal `expert`; hoverover see *User feedback*	`//*[duv:hasFeedback/duv:UserFeedback/ prov:qualifiedAssociation/ prov:Association/prov:hadRole/prov:Role[ contains(rdf:about, 'expert')] ]` (partial expression)
Citations information	An SSN object, e.g., `sosa:Observation`, has a property `biro:isReferencedBy` pointing to a piece of literature; hoverover has count of references.	`//*[biro:isReferencedBy]`

In column **XPath**: Wrapping `boolean(..)` statements omitted; partial expressions show only one of multiple allowed statements combined with `or`.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 February 2020 doi:10.20944/preprints202002.0326.v1

Table 2. Used ontologies and vocabularies with their prefixes, and namespaces.

Ontology/vocabulary	**Prefix**	**Namespace**
Resource Description Framework	`rdf`	http://www.w3.org/1999/02/22-rdf-syntax-ns#
Resource Description Framework Schema	`rdfs`	http://www.w3.org/2000/01/rdf-schema#
Semantic Sensor Network Ontology	`ssn`	http://www.w3.org/ns/ssn/
Sensor, Observation, Sampling and Actuator	`sosa`	http://www.w3.org/ns/sosa/
System Capabilities Module (of SSN)	`ssn-system`	https://www.w3.org/ns/ssn/systems/
Provenance Interchange Ontology	`prov`	https://www.w3.org/ns/prov#
Friend of a Friend Ontology	`foaf`	http://xmlns.com/foaf/0.1/
Dataset Usage Vocabulary	`duv`	http://www.w3.org/ns/duv#
Data Catalog Vocabulary	`dcat`	https://www.w3.org/ns/dcat#
Bibliographic Reference Ontology	`biro`	http://purl.org/spar/biro/
Web Annotation Ontology	`oa`	http://www.w3.org/ns/oa#