A next-generation geospatial catalogue: a proof of concept

Sergio Martín Segura
Universidad Zaragoza
Zaragoza, Spain
segura@unizar.es

Francisco J Lopez-Pellicer
Universidad Zaragoza
Zaragoza, Spain
fjlopez@unizar.es

Juan Valiño García
Universidad Zaragoza
Zaragoza, Spain
juanv@unizar.es

F. Javier Zarazaga-Soria
Universidad Zaragoza
Zaragoza, Spain
javy@unizar.es

Abstract

This paper presents an idea for the development of a catalogue system for spatial datasets based on indexing both their metadata and their features. This characteristic is not available in spatial data catalogues in Spatial Data Infrastructures. This catalogue uses features for improving the relevance of responses because metadata records may not convey all the information that users may need for dataset discovery. The underlying search engine guarantees that when the query filters using a bounding box, all the returned datasets should contain features in the queried area and their rank position will depend on the characteristics of the matching features. In a similar way, when the query contains a text query expression, all the returned datasets should contain features related to such query even if their metadata records do not mention. The feasibility of this approach is shown with the development of a proof of concept using open source off-the-shelf technologies like Elasticsearch.
Keywords: search, engine, relevance, rank, elasticsearch.

1 Introduction

There is an implicit mantra from the beginning of the Spatial Data Infrastructures: spatial data should be distributed as a collection of data along with an attached metadata record (Hjelmager et al., 2008; Rajabifard, Kalantari & Binns, 2009). In the context of data publication and data discovery, this mantra assumes that collections of data should be treated as opaque artefacts whose whole description is conveyed in metadata records. The logical consequence of this assumption is data discovery involves the development of data catalogues that only contain metadata records about collections of data. Nearly 25 years ago, Timpf, Raubal & Kuhn (1996) pointed that the then emerging metadata standards, now a firm reality, focus more on what the data producers have to say than on what the data users need to know. Ten years later, works like Larson et al. (2006) Li & Yang (2008), and Macário & Medeiros (2009) revealed that standards-based metadata catalogues did not focus on what the data users need to discover spatial data and, therefore, they should be improved to fulfil this purpose. Recent initiatives such as the OGC-W3C Spatial Data on the Web Working Group address this issue by taking advantage of commercial search engines. Succinctly, they propose to publish a HTML Web-page with search engine focused metadata for each spatial dataset and each spatial thing that is described and wait to see if commercial search engines crawl and index them properly. This could enable non-expert users to use commercial search engines to discover spatial data but expert users will still need spatial catalogues.

How can we improve our spatial catalogue today? We have several ways. For example, a user looking for a specific historical gazetteer may use terms such as “gazetteer” “historical” in its query but also may use multiple pre-existing names that he or she expects to be found in such gazetteer to narrow the search. However, these names that can be found in the gazetteers that fit user needs are noise unless they are explicitly mentioned in metadata records. An improved catalogue system could help by providing an information retrieval process for geospatial catalogue aimed at improving results (Lacasta et al., 2017).

A related problem is the tradition to abstracting the description of the spatial extent of a dataset in metadata records to a simply a bounding box. In some scenarios, the difference between the bounding box and the area of the data may cause to return datasets spatially irrelevant to the intent of the query even when the spatial extent described in the metadata matches the spatial constraint of the query. A quick way to reproduce this scenario in any catalogue is to ask a query with a bounding box within the waters of a gulf (i.e. the intent of the user is “return sea related datasets”) and count the number of land related datasets returned as results: roads, natural reserves, etc. (i.e. the intent “interpreted” by the catalogue is “return land related features“). These datasets probably do not contain sea related features, but their bounding boxes cover parts of waters of the gulf and, therefore, are “fair” results. An improved catalogue system could help by identifying in spatial semantics during the load of metadata records to improve the quality of the responses (Rentería-Agualimpia et al., 2015).

This work presents a different approach to improve catalogues. We assume that the whole description cannot be conveyed by metadata records. Hence, we propose that catalogue indexes should be built mainly from information extracted from the raw data collection. The main contribution of this work is a proof of concept of this approach. The work is organized as follows. First, we introduce the key idea. Next, we describe the off-the-shelf search engine that enables its development. The implementation of the proof of concept is described and some use cases are discussed. Finally, we conclude with some remarks about this proposal and its future.

2 Search and rank strategy

The key idea of our proposal is that, instead of storing just metadata records in the catalogue indexes, we store any information that may be relevant to the discovery task. This information should be extracted from the original metadata record and each feature of the corresponding dataset. The straightforward implementation of this approach is to add the textual information extracted from features to a bag of words and their geometries to a spatial index. With this setup we could perform the following query: return the datasets that contains some features matching the spatial and textual constraints ordered by a relevance rank based on the matched features.

3 Search engine

The strategy above requires a search engine that supports spatial queries and hierarchical queries. Moreover, this new model requires a huge escalation of the information managed by the catalogue system, growing from just a few thousands of datasets to the millions of features that the datasets contain. Modern off-the-shelf search engines such as Elasticsearch provides support for this approach. Elasticsearch is an open source, scalable, field tested, big data-oriented search engine with spatial and parent-child support. Currently, in version 6.X, it supports Geohash Prefix Tree and Quad Prefix Tree spatial index types for heterogeneous shape types at indexing and querying. Companies like GitHub or Cisco are using Elasticsearch to access petabytes of information in just a few milliseconds. Such features met our needs, so we decided to use it as the search engine for the proof of concept.

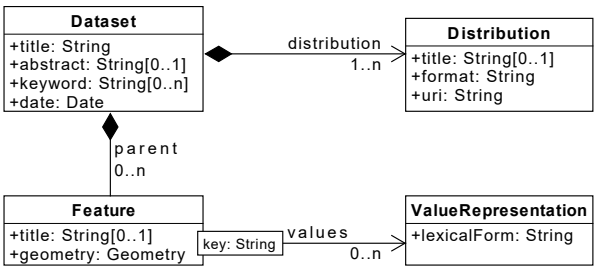
4 The proof of concept

4.1 Information model

The information model of the proof of concept is a minimal model with the essential elements (Figure 1):

- *Dataset.* It represents the whole collection and holds the metadata extracted from the original metadata record.
- *Distribution.* A dataset may have different availability forms. This class represents a format available at an endpoint.
- *Feature.* A dataset contains features, i.e. spatial objects. This class represents indexable textual (*values*) and spatial (*geometry*) information.

Figure 1: Information model.



4.2 Elasticsearch schema

The catalogue data schema had to be designed carefully due to Elasticsearch technical constrains without losing the original model idea. The biggest restriction is that the parent/child relationship requires the parent and the child objects to be stored in the same index and document type in Elasticsearch. The combination of the attributes of the dataset and the feature classes is the document type named root document. In Elasticsearch, the join field works as a pointer to the parent document (join parent) and as a type of document selector (join type). The values of join for datasets and for the features are different as shown in the table (Table 1). Root documents of join type “dataset” encodes data from datasets. Each root document of join type “feature” encodes data from a single feature. Its geometry is stored in the field geometry and the lexical representations of its non-spatial attributes is stored concatenated in a single field named description. The distribution class is encoded as a nested distribution document in root documents of join type “dataset” (Table 2).

Table 1: Root Schema.

Parameter	On Dataset	On features
title	title	title
description	abstract	values
keyword	keyword	-
join parent	-	parent id
join type	“dataset”	“feature”
date	date	-
distribution	distribution []	-
geometry	-	geometry

Table 2: Distribution Schema.

Parameter	On Dataset
title	distribution title
format	distribution format
uri	distribution URI

Figure 2: Ingest process.

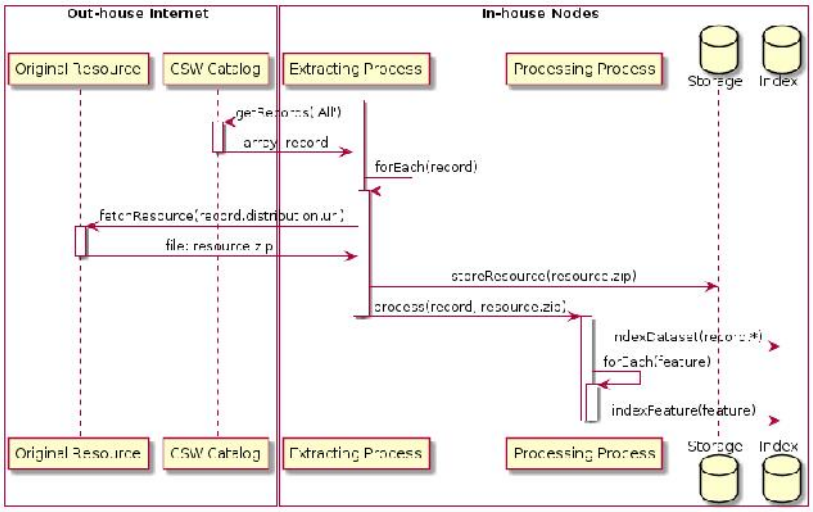
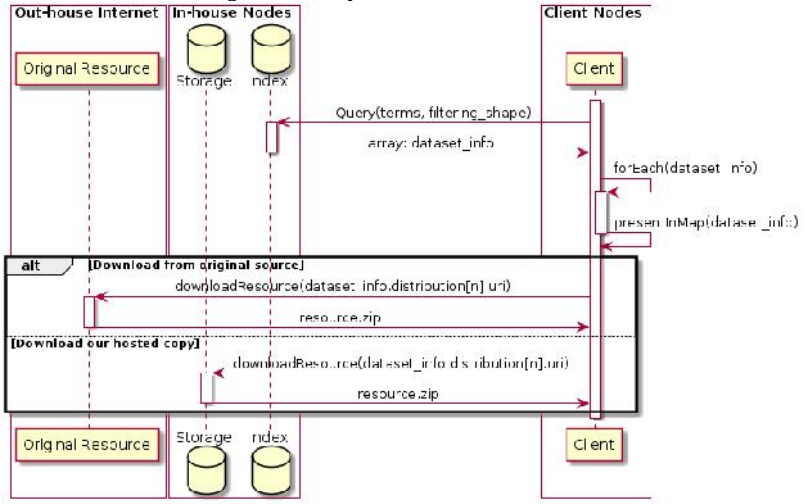


Figure 3: Query and download scenario.



4.3 Ingest process

The ingest process (Figure 2) has two main phases: extraction phase and processing phase. The extraction phase starts with a query to a CSW service to get dataset metadata records. Next, for each record fetched, it downloads the original resources described in the metadata, stores and pass them to the next process. Depending on the file format (GeoJSON, SHP, etc.), the corresponding process, reads that resource downloaded and iterates over all its features transforming them to a common format supported by the index (GeoJSON) and inserting them in th index. The ETL process is implemented in IPython notebooks. To simplify the implementation, we only process datasets available in ESRI Shapefile format, but additional formats could be easily added in future versions as new parallel processes. Each dataset and feature have unique identifiers derived from their properties so the process can be scheduled to update the datasets periodically. The Python libraries used in the ingest process are: OWSLib 0.17.1 to query CSW services; Fiona 1.8.4 to read and process the Shapefiles and Elasticsearch-DSL 6.3.1 to insert data into the index.

4.4 Querying the cataloge

The Query Language of Elasticsearch has an operation called has_child where you can define query that will be applied to the children of an element. The other important operations are match which matches full text queries and geo_shape which filters by GeoJSON area. In our case, we can express this query in pseudocode as:

```
Datasets that {
   has_child {
      Features that {
         match text query
         geo_shape area
      }
   }
   match text query
}
```

This way, the results will be the datasets that better matches the text query, including the texts in their features, if and only if they have at least one feature in this area. The web app will then present them both in a list and over the map, being able to see the individual features that make up the dataset. The

ETL process has inserted references to the original source of the data so the user can download it from there or download the system's hosted version that acts as a mirror the original source fails.

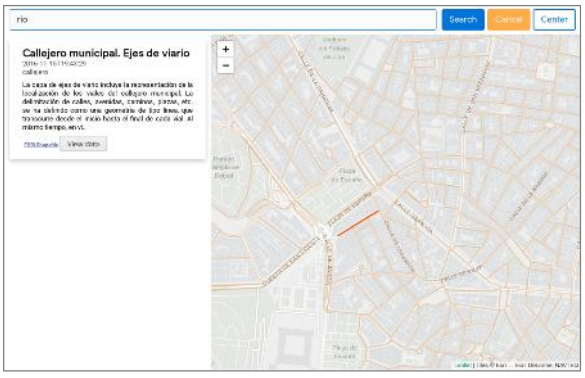
4.5 Web client

On the client side, we need an easy to use search portal with a results list and a map where display the results. It was implemented with the React + Redux framework. The reason to choose React + Redux as the frontend framework was its ability to code independent user interface components and easily compose them into a web application. Apart from the visual style, the most important element is the query that the client is asking to Elasticsearch. The query component was easy to build and was a combination of textual query plus a Leaflet map providing the spatial constraint. The search response component displays the title of the dataset, its description, a list of links to its different distributions where the user can download the dataset or access it if the distribution is a remote download service and the features in the map (Figure 3).

4.6 Testing the system

In order to test the capabilities of our solution, we loaded 42 datasets extracted from the Spanish SDI CSW with more than 85,000 features in a low-end computer with two cores and 8GB of RAM. We performed spatial only queries with the map at different scales to assert that the datasets returned where spatially relevant. In other words: They had features in this area. We also tested searches for something that can be mistaken for a feature different from the intention of the search. In the image (Figure 4) we can see a query performed with the text "río" (river in Spanish) over a neighborhood area in Madrid where there was no watercourse. The results contain a dataset from an address database because the area has a street named "Calle del Río". This example combines the spatial relevance constraint (no hydrology dataset was returned as response because there are no watercourses in the area) and the textual hint based on data (a dataset is discovered because at least one of its features contains the term "río" although its metadata record does not). It is worth to mention that the query latency was under 100ms although speed was not an objective of the tests.

Figure 4: Searching a dataset that contains features about a "río" (river) within a neighborhood of Madrid.



5 Conclusions

The idea that data discovery in SDIs depends on the development of catalogues that only contain metadata records should be reconsidered. This work has presented a proof of concept that shows the feasibility of a catalogue system built from information extracted from metadata and features. A new version is in development with a distributed architecture and multiple formats support aiming a more effective data discovery for users in SDI.

Acknowledgements

This work has been partially supported by the Aragon regional Government (project T59_17R) and the Spanish Government (projects RTC-2016-4790-2 and TIN2017-88002-R).

References

Hjelmager, J., Moellering, H., Cooper, A., Delgado, T., et al. (2008) An initial formal model for spatial data infrastructures. *International Journal of Geographical Information Science*. 22 (11-12), 1295-1309

Lacasta, J., Lopez-Pellicer, F.J., Espejo-García, B., Nogueras-Iso, J., et al. (2017) Aggregation-based information retrieval system for geospatial data catalogs. *International Journal of Geographical Information Science*. 31 (8), 1583–1605.

Larson, J., Olmos Siliceo, M.A., Pereira dos Santos, M., Klien, E., et al. (2006) Are geospatial catalogues reaching their goals? In: *AGILE Conference on Geographic Information Science*. 1 January 2006 Visegrád, Hungary. p.

Li, W. & Yang, C. (2008) A Semantic Search Engine for Spatial Web Portals. In: *IGARSS 2008 - 2008 IEEE International Geoscience and Remote Sensing Symposium*. 2008 IEEE. pp. II-1278-II–1281.

Macário, C.G.N. & Medeiros, C.B. (2009) The geospatial semantic web: Are GIS catalogs prepared for this? In: *Proceedings of the Fifth International Conference on Web Information Systems and Technologies*. [2009 SciTePress - Science and and Technology Publications. pp. 335–340.

Rajabifard, A., Kalantari, M. & Binns, A. (2009) SDI and Metadata Entry and Updating Tools. In: *GSDI 11 World Conference and the 3rd INSPIRE Conference 2009, Rotterdam 15-19 June 2009*. 2009 pp. 121–135.

Rentería-Agualimpia, W., Lopez-Pellicer, F.J., Lacasta, J., Muro-Medrano, P.R., et al. (2015) Identifying geospatial inconsistency of web services metadata using spatial ranking. *Earth Science Informatics*. 8 (2), 427–437.

Timpf, S., Raubal, M. & Kuhn, W. (1996) Experiences with metadata. In: *Proceedings of Symposium on Spatial Data Handling, SDH'96, Advances in GIS Research II*. 1996 pp. 12B31-12B43.