Call for Short Papers: OGRS 2016
http://2016.ogrs-community.org/

Implementing an open source spatio-temporal search platform for Spatial Data Infrastructures

Paolo Corti, Harvard Center for Geographic Analysis
Benjamin Lewis, Harvard Center for Geographic Analysis
Tom Kralidis, Open Source Geospatial Foundation
Ntabathia Jude Mwenda, Harvard Center for Geographic Analysis

Abstract

A Spatial Data Infrastructure (SDI) is a framework of geospatial data, metadata, users and tools intended to provide the most efficient and flexible way to use spatial information. One of the key software components of a SDI is the catalogue service, needed to discover, query and manage the metadata. Catalogue services in a SDI are typically based on the Open Geospatial Consortium (OGC) Catalogue Service for the Web (CSW) standard, that defines common interfaces to access the metadata information.

A search engine is a software system able to perform very fast and reliable search, with features such as full text search, natural language processing, weighted results, fuzzy tolerance results, faceting, hit highlighting and many others.

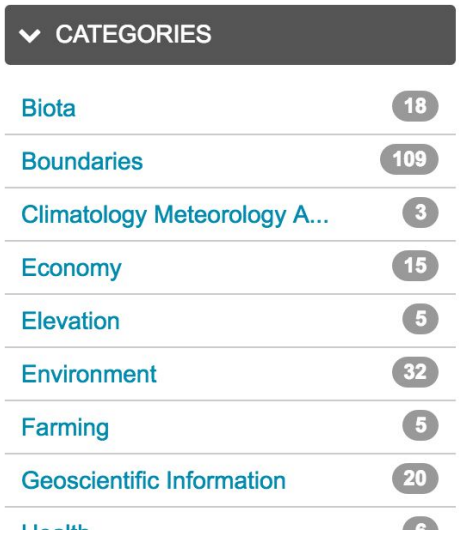
The Centre of Geographic Analysis (CGA) at Harvard University is trying to integrate within its public domain SDI (named WorldMap), the benefits of both worlds (OGC catalogues and search engines).

Harvard Hypermap (HHypermap) is a component that will be part of WorldMap, totally built on an open source stack, implementing an OGC catalogue, based on pycsw, to provide access to metadata in a standard way, and a search engine, based on Solr/Lucene, to provide the advanced search features typically found in search engines.

The need for a search engine in a SDI

Typically the way a search engine works can be split in two distinct phases: indexing and searching. During the indexing phase all of the documents (metadata, in the SDI context) that must be searched are scanned, and a list of search terms (index) is built. For each search term the index keeps track of the identifiers of the documents that contain that search term. During the searching phase only the index is being looked at, and the list of the documents containing the given search term is quickly returned to the client.
This way of working make the search engine extremely fast in outputting results. On top of it a search engine provides a lot of other useful features, improving dramatically the reliability and experience of a user searching for good metadata.
Most notably, the search engine is able to detect the ambiguities of natural languages, thanks to *stop words* (words filtered out during the processing of text), stemming (ability to detect words derived from a common root), synonyms detection, and controlled vocabulary such as thesauri and taxonomies.
It is possible to do phrase searches and proximity searches (search for a phrase containing two different words separated by a specified number of words).
Results are weighted, providing a mean to the end user to access the most desired results. It is possible to search regular expressions, wildcard search and fuzzy search (that can provide results for a given term and its common variations).
It is possible to do boolean queries: a user is able to search results using terms and boolean operators such as AND, OR, NOT.
Hit highlighting is an interesting feature that provides immediate suggestions to the user typing the text to search in metadata.
Another search engine feature that can be extremely useful for searching a metadata catalogue is faceted search. Faceting is the arrangement of search results in categories based on indexed terms. Thanks to this it is possible for example to provide an immediate indication of the number of times that common keywords are contained in different metadata documents.
A typical use case is with metadata categories, keywords and regions: thanks to facets the user interface of the SDI catalogue will display how many different documents contain a given category, keyword or region.

PeerJ Preprints | https://doi.org/10.7287/peerj.preprints.2238v3 | CC BY 4.0 Open Access | rec: 6 Jul 2016, publ: 6 Jul 2016



Search engine can also perform dates and spatial faceting, two features that are extremely useful for browsing geospatial metadata.
Dates faceting will display the number of documents for each given date range. Space faceting will provide the number of layers or features for each given spatial extent. In the following figure it is depicted an heatmap generated by spatial faceting displaying how many layers are in the catalog for each cell in a regular grid.

Furthermore, from a software engineer prospective, typically search engines are highly scalable and replicable, thanks to their shardable architecture.

Integration of a CSW catalogue with a Search Engine

There can be two approaches to integrate a search engine and the CSW catalogue within an SDI:

- combining the best of two worlds
- using the search engine as a backend of the CSW catalogue

HHypermap in this first phase combines the best of the two worlds.
Clients willing to search the catalogue provided by HHypermap in a standard way will be able to run a search using the OGC CSW endpoint.
Clients willing to use the more advanced features, such as faceting or full text features such as synonyms detection, fuzzy tolerance, weighted search, etc. will be able to run a search accessing the restful API provided by the search engine.
Harvard CGA is currently looking in the possibility to enable the search engine itself as a backend to its CSW endpoint. This way it will be possible to access at least a part of the powerful search engine features from within the CSW itself.

Harvard Hypermap: an SDI search engine based on a open source stack

HHypermap is an application that manages OGC web services (such as WMS, WMTS), Esri REST endpoints and other types of map service harvesting, and maintains uptime statistics for service and layers. The aim of HHypermap is to provide a powerful search experience to WorldMap users. WorldMap is an open source mapping platform developed by the CGA to lower barriers for scholars who wish to explore, visualize, edit and publish geospatial information.

HHypermap is totally built on a open source architecture. The web application has been developed on top of the Django web framework based on the Python language. Relational structured information is stored in a PostgreSQL database. Remote services and layers are harvested with Celery, an asynchronous Python task queue, taking advantage of the RabbitMQ message broker. pycsw is a Python library providing the OGC CSW interface. HHypermap can use Solr or ElasticSearch as a search engine, both of which are based on Lucene, a powerful Java-based indexing and search library.

HHypermap source code can be found here: https://github.com/cga-harvard/HHypermap