IV JORNADAS DE SIG LIBRE

OpenSearch-geo: The simple standard for geographic web search engines

O. Fonts[(1),] *J. Huerta*[(1),] *L. Díaz*[(1),] *C. Granell*[(1)]

(1) Institute of New Imaging Technologies, Geographic Information Research Group, Universitat Jaume I, Castelló, Spain. {fonts, huerta, laura.diaz, carlos.granell}@ uji.es

ABSTRACT

When publishing information on the web, one expects it to reach all the people that could be interested in. This is mainly achieved with general purpose indexing and search engines like Google which is the most used today. In the particular case of geographic information (GI) domain, exposing content to mainstream search engines is a complex task that needs specific actions.

In many occasions it is convenient to provide a web site with a specially tailored search engine. Such is the case for on-line dictionaries (wikipedia, wordreference), stores (amazon, ebay), and generally all those holding thematic databases. Due to proliferation of these engines, A9.com proposed a standard interface called OpenSearch, used by modern web browsers to manage custom search engines.

Geographic information can also benefit from the use of specific search engines. We can distinguish between two main approaches in GI retrieval information efforts: Classical OGC standardization on one hand (CSW, WFS filters), which are very complex for the mainstream user, and on the other hand the neogeographer's approach, usually in the form of specific APIs lacking a common query interface and standard geographic formats.

A draft 'geo' extension for OpenSearch has been proposed. It adds geographic filtering for queries and recommends a set of simple standard response geographic formats, such as KML, Atom and GeoRSS. This proposal enables standardization while keeping simplicity, thus covering a wide range of use cases, in both OGC and the neogeography paradigms.

In this article we will analyze the OpenSearch geo extension in detail and its use cases, demonstrating its applicability to both the SDI and the geoweb. Open source implementations will be presented as well.

Keywords: ***Geospatial search engines, OpenSearch, Geographic web services, geoweb.***

INTRODUCTION

Web content findability is generally addressed by mainstream search engines, such as Google. Sites like Wikiloc[1] demonstrate that geographic content can be effectively exposed to Google's geoindex, and can be retrieved through a general web search, just like any other non geographic content. However, the data publisher has to manually adapt its content and publish it in the way google wants it, and has little control over how and when the data will be indexed or ranked [1]. It might even be never indexed.

So relying on external geoindexation engines can be valuable, but it in many cases data publishers will want to setup their own search engines, addressing spatial and full text indexation, that return optimally ranked result sets.

Geodata search standardization efforts in OGC include Catalogue Services for the Web query syntax (Common Query Language at catalog level) [2] and WFS filters (at feature level) [3]. These syntaxes are suitable for fine grained selection and provide high levels of expressiveness. But they assume previous knowledge of query syntax, data schema and its semantics, restricting its potential to advanced users already familiar with the queried dataset nature. These engines might search into a catalog for metadata retrieval, or directly into the data for feature selection based on its attributes. In any case, for the results to be useful, they should link as directly as possible to the data they represent [4].

To target the widest audience (*mass market* in OGC vocabulary), a radically simple interface must be provided [4], consisting of one unique text box to search across all the data. Specialization of the search engine can take benefit from data nature awareness to optimize the result set. For example, finding the queried text in 'dc:title' would rank higher than finding it in 'dc:abstract'.

Besides from OGC standards services, there is other rapidly growing geographic (or geotagged) content in the web, which we will call neogeography, often ignoring OGC standardization efforts because of their complexity and specialized audience. Many of such web services offer their own API to query against their georeferenced datasets, such as geonames[2], flickr[3] or twitter[4]. Each API having its own query syntax and response formats, search clients cannot be generic.

There is a need for a search interface suitable for both OGC and neogeograpy web services that provides standarization while keeping simplicity.

Such a standard was proposed by A9.com and is called OpenSearch [5]. OpenSearch:

- Describes search engine capabilities in a structured, machine-readable way.
- Provides a simple set of allowed request parameters.
- Provides a simple set of response formats.
- Is extensible.

[1] http://www.wikiloc.com
[2] http://www.geonames.org/export/ws-overview.html
[3] http://www.flickr.com/services/api/
[4] http://apiwiki.twitter.com

One of its extensions is the OpenSearch-geo draft, proposed by Andrew Turner [6]. While geo-search services continue to grow in number, there are yet few of them implementing this standard, and there is a lack of client implementations too. One of the aims of this work is to analyze existing opensearch-geo implementations, check their interoperability, and promote new ones.

Pedro Gonçalves et al. are working in an OpenSearch extension for OGC Catalogue Services [7], now in draft status, that extends and adapts the original A.Turner’s proposal. The second aim of this paper is to address interoperability issues in these proposals, identifying:

- Coherence between both drafts, proposing a merged solution,
- Degrees of freedom that could lead to non-interoperable services, specially with respect to response formats and its content tags,
- Core (mandatory) functionality vs. recommended (optional) functionality, keeping in mind that expressiveness is a plus, but simplicity is a must,
- Backwards compatibility with simple OpenSearch (non geo-aware) clients.

In the following section the OpenSearch specification is briefly introduced, describing its parameters, response formats and use in web browsers. The next section discusses the geo extension thoroughly, based on the two draft specification efforts available: Description document, parameter names and formats, error handling, encoding, and popular geographic response formats are analyzed. A third section is devoted to existing service and client implementations. Finally, some conclusions are drawn.

OPENSEARCH SPECIFICATION

In this section the OpenSearch standard [5] is briefly described, introducing its autodiscovery mechanism, description document, request parameters and response formats. Its use in web browsers is also introduced.

OpenSearch was conceived by A9.com, an Amazon.com company, as a mechanism to trigger a distributed search over a collection of web sites and merge the results in an unique result set (search results syndication).

The entry point to a search engine is its description document. This document contains, along with other metadata, an URL template for each accepted response format. Each template indicates the mandatory and optional query parameters, as well as the syntax to build the query as an URL. The query is performed as an HTTP GET.

There is only one mandatory parameter, {searchTerms}, which are the keywords to be searched for. Other optional parameters are used for paginated results, preferred language, and request and response encoding. The original parameters list can be extended incorporating namespaces to the description document, which makes possible the ‘geo’ extension.

The response formats can be syndication formats as RSS 2.0 [8] or Atom 1.0 [9], but others may be used without restriction. In XML-based responses, the ‘opensearch’ namespace[5] is used to add pagination information and a reference to the originator query, and the ‘atom’ namespace[6] is used to add a reference to the search description

[5] http://a9.com/-/spec/opensearch/1.1/
[6] http://www.w3.org/2005/Atom

document. This last feature enables search engine autodiscovery from a response document. The autodiscovery mechanism can be used also in an HTML 4 document by means of a link tag.

Probably the most popular use case of OpenSearch is in modern web browsers like Firefox 2+ [10], and Internet Explorer 7+ [11]. When an HTML page with an OpenSearch autodiscovery tag is loaded, the web browser offers the possibility to add a custom search engine to its “search bar”.

When creating advanced OpenSearch services, such as geo-enabled ones, it is almost obliged to keep compatibility with browser’s functionality, as it is the most popular and convenient way to exploit them. This means including, at least, an HTML response format, and reviewing main browser’s recommendations ([10, 11]). A JSON suggestion (“as-you-type”) response format [12, 13] is also recommended when technically possible, as to be useful it requires low response times (<100 ms).

OPENSEARCH GEO EXTENSION

The OpenSearch geo extension adds new parameters and suggests a collection of simple geographic response formats. For the sake of simplicity, all coordinates in this standard are expressed as geodetic WGS84 latitude and longitude (EPSG:4326).

There are two OpenSearch-geo draft proposals: The original from A. Turner [6] and a proposed OGC extension for CSW [7]. The former is in ‘draft’ status, but constitutes the main public accessible reference. The later is in discussion status inside OGC, so it can suffer substantial changes and is intended for discussion only (non normative). The Geo extension is analyzed in this section, both proposals are compared, and an outline on how to merge them is suggested.

Description document

The geo extension adds a new namespace[7] to the description document that allows for specific parameters in the url templates.

For services restricted to a specific geographic domain, we suggest to add an optional bbox tag to the description document. This could prevent clients to perform queries out of service’s geographic domain, and makes the own service geoindexable. The geoRSS-simple syntax [15] could be used (note that geoRSS coordinate pairs are latitude-longitude ordered, inversely from {geo:bbox?} parameter) .

```
<?xml version="1.0" encoding="UTF-8"?>
<OpenSearchDescription
  xmlns="http://a9.com/-/spec/opensearch/1.1/"
  xmlns:geo="http://a9.com/-/opensearch/extensions/geo/1.0/"
  xmlns:georss="http://www.georss.org/georss"
  <ShortName>Local geostuff</ShortName>
  <!-- Other service metadata -->

  <georss:box>42 -71 43 -69</georss:box>

  <Url type="application/vnd.google-earth.kml+xml"
template="http://example.com/search.kml?nom={searchTerms}&amp;box={geo:bbox}"/>
  <Url type="application/atom+xml"
template="http://example.com/search.atom?nom={searchTerms}&amp;box={geo:bbox}"/>

 </OpenSearchDescription>
```

[7] http://a9.com/-/opensearch/extensions/geo/1.0/

Request

Discussion on request parameters, error handling and request encoding follows.

Parameters

To keep compatibility with non geo extension aware clients, all geo parameters should be optional for them. Services claiming to be “OpenSearch geo” (that is, including the ‘geo’ namespace in the description document) should implement at least the {geo:box?} parameter (see Table 1). This is stated in [7] and could be translated to [6]. Implementing a box filter over a geographic dataset is simple and provides a fundamental functionality.

Table 1: Compared OpenSearch Geo parameters in both drafts

Parameter name in OpenSearch.org [6]	Parameter name in OGC [7]	Definition and Format
{geo:box?}	{geo:box?}	Four comma separated geodetic coordinates (WGS84) describing a rectangular filter, in the form “west,south,east,north” (as in WMS BBOX [14]).
{geo:polygon?}	{geo:geometry?}	Polygon as in geoRSS-simple [15]; Well-Known Text geometry [16].

{geo:lat?}
{geo:lon?}
{geo:radius?}
	{geo:lat?}
	{geo:lon?}
	{geo:radius?}
		A circle described as a center (lat, lon) and a radius (eters).

{geo:locationString?}	{geo:name?}	A place name (text).
	{geo:relation?}	One of “overlaps”, “contains”, “disjoint”.
	{geo:uid?}	Unique identifier.

Parameter name incoherences between both drafts should be resolved. This is the case for {geo:polygon?} vs. {geo:geometry?}, and {geo:locationString?} vs. {geo:name?}. As [6] has been publicly available for a time, we recommend keeping the original {geo:polygon?} and {geo:locationString?} parameter names.

{geo:polygon?} and {geo:geometry?} parameters serve similar purposes, but the format is defined differently in both proposals. {geo:polygon?} [6] is a comma-separated coordinate pair list (in lat, lon order) describing a 2D simple polygon external ring in clockwise order:

$$\mathtt{lat_1,lon_1,lat_2,lon_2,lat_3,lon_3,[\ldots],lat_1,lon_1}$$

A polygon has at least three different points, plus the last one being the same as the first, closing the loop. Note that this is the same polygon serialization used in geoRSS-simple profile [15].

The alternative {geo:geometry?} [7] parameter content relies on Well-Known Text standard [16]. Well Known Text syntax provides for point, linestring, polygon, triangle, polyhedralsurface, tin, multipoint, multilinestring, multipolygon and geometrycollection geometry types, either in 2D or 3D, and optionally with linear referencing (measured). Such expressiveness comes at a price, and this extra complexity is not providing a substantial gain. So keeping the original {geo:polygon?} parameter as defined in [6] is proposed.

As both standards stay, {geo:lat?}, {geo:lon?} and {geo:radius?} should be used together to describe a circle[8].

{geo:box?}, {geo:polygon?}, and the ‘circle’ ({geo:lat}, {geo:lon} and {geo:radius}) triplet are geometric filters. They are mutually exclusive. That is, only one of them can be used in a given query. If a query contains more than one geometric filter, search engine behavior is undefined.

{geo:relation?} value can be one of “overlaps”, “contains”, “disjoint”[9]. These keywords are a subset of Common Query Language “geoop names” ([2] p. 14). Search engines should ignore it if not accompanied by a geometric filter.

{geo:locationString?} is a text field indicating a place name to search into. Its behavior is rather unpredictable, and will depend mostly on how the search engine deals with it. {geo:uid?} is an unique identifier of the record in the repository context [7]. Both functionalities could be assimilated in the general {searchTerms} textbox.

Thus, the geo parameters could be reduced, to ‘box’, ‘polygon’, ‘circle’ and ‘relation’.

Error handling

Many restrictions in parameter format and combinations have been stated. The OpenSearch geo specification should identify the potential derived errors in a client request. For example, bbox’s xmin greater than xmax, invalid polygon syntax, lat and lon stated without a radius, invalid relation name, more than one geometric filter, etc.

OpenSearch has no hard rule about how to communicate errors to the client. However, the “developer how to” suggests returning a well-formatted response with an item describing the error in a human readable way[10]. We suggest using the HTTP 400 “Bad Request” client error status code [17], with an error description as payload.

Url and character encoding

Not being an OpenSearch specific problem, character and url encoding is a common source of problems. When constructing a request from its template, all parameter values should be url (percent) encoded. According to RFC 3986 [18], the universal characters (international alphabets) should be first byte encoded in UTF-8, then the result percent-encoded. The ECMAScript [19] function encodeURIComponent is a convenient way of performing this within a web browser environment.

So it is recommended for new OpenSearch services to include UTF-8 InputEncoding as default, as already stated in the description document specification [5]. Older services will accept ISO-8859-1 as InputEncoding, but this practice is discouraged in accordance to RFC 3986 [18].

[8] Radius is expressed in meters, and lat, lon in degrees. In the lat-lon space, the degrees-per-meter factor is different for each axis, and depends on latitude. As a rough approximation (for small radii and considering the Earth spherical), consider degrees-per-meter in the north-south axis as 0.000009. And for the east-west axis as 0.000009/cos(lat). So the ‘circle’ in Earth’s surface is approximately transformed to an ellipse in the lat-lon space, with its major axis in the east-west axis, and more excentric at higher latitudes. For a ‘circle’ centered at the poles, the transformation is a box spanning all latitudes wide.

[9] “overlaps” means matching all the resources partially or totally inside the geometric filter. “contains” only selects the resources totally inside. “disjoint” matches the resources totally outside (not(overlaps)).

[10] http://www.opensearch.org/Documentation/Developer_how_to_guide#How_to_indicate_errors

The comma (,) used in {geo:box?} or {geo:polygon} to separate coordinate values is often not replaced by its percent-encoded equivalent %2C, and the space character, which has a percent-encoded value of %20, is usually encoded as a plus sign '+'. OpenSearch services should consider these variants for compatibility with all client implementations.

Response formats

The available response formats are identified by its MIME types in the service description document. OpenSearch does not set any mandatory format, nor limits its number. The most widely used include RSS 2.0 (application/rss+xml), Atom 1.0 (application/atom+xml), and HTML (text/html) or XHTML ('application/xhtml+xml). Atom and RSS are suitable for syndication, and HTML/XHTML for human-readable in-browser visualization. JSON suggestions[11] format is also very popular.

Any OpenSearch geo service should include geographic content in its responses. In order for clients and services to interoperate, the server should offer geographic content encoded in a predictable way. It should be mandatory for 'geo' services offering RSS or Atom to encode geographic content in GeoRSS [15], and those offering XHTML responses, to use the 'geo' microformat [21]. The same way, JSON responses should encode geographic content using GeoJSON [22] (except for JSON suggestions).

Discussion on recommended geo formats follows.

Geo microformat in XHTML

This format provides a machine-readable way to describe a coordinate pair (thus, only a point), using existing HTML tags, so HTML parsers can identify geographic tagging in conventional web pages. It consists of a root tag with class "geo" with two child tags, with "latitude" and "longitude" classes. For example:

```
<div class="geo">This result is located at:
 <span class="latitude">37.386013</span> lat,
 <span class="longitude">-122.082932</span> lon.
</div>
```

For example, the Minimap Sidebar[12] extension for Firefox can detect the locations so described and conveniently display them over a map.

GeoRSS in RSS and Atom

RSS and Atom responses should use GeoRSS to encode geographic content. Using non standard geographic serialization leads to generic clients not knowing how to parse it, thus breaking interoperability. GeoRSS comes in two serializations or profiles:

- **GeoRSS-simple** serialization is designed to be maximally concise. The representations available (point, box, circle, line and polygon) require only a single tag to be described.
- **GeoRSS-GML** is a simple GML 3.1.1 profile to represent the same elements (Point, Envelope, CircleByCenterPoint, LineString and Polygon) more verbosely. GML adds multiple Coordinate Reference Systems support, which won't be used in OpenSearch.

[11] http://www.opensearch.org/Specifications/OpenSearch/Extensions/Suggestions/1.1
[12] http://minimap.spatialviews.com/

The GeoRSS specification includes an XSD schema and a GML profile that can be used for format validation. It defines "<georss:where> as the tag that signals geographic content – either in GeoRSS Simple or GML." [15]. In the practice, this tag is not always used (in fact, some examples from the own specification don't use it). So parsers should be aware of the *de-facto* optionality of this tag, and detect geographic content by the presence of 'georss' namespace[13] content, be it inside a <georss:where> tag or not.

Finally, note that OpenSearch draft extension for CSW [7] sets Atom as a mandatory format.

KML

We strongly recommend to implement KML as a response format in OpenSearch geo services, as it is well defined (by both Google and OGC specifications) and easy to implement (well known, widely used in web, with many libraries available), and it provides unique features for visualization and interaction (styling contents, network links, 3D capabilities, etc.), thus being the suitable format for rendering over an interactive map or globe.

There are two ways to describe an item (or <Placemark>) in KML [1]. One is through a CDATA element inside the <Description> tag, containing a fragment of HTML tagged text. This description is suitable for visualization of human-readable content, where some multimedia elements (images or videos) can also be embedded. Google Maps and Google Earth, for instance, show this HTML description inside the pop-up bubble associated with the Placemark. Another way to describe item attributes is through the <ExtendedData> element, which allows for structured content through the use of a predefined schema (see section 9.2 in [20] for further details).

In some use cases it can be useful to maintain results attribute's structure. For example, for carrying a Dublin Core metadata set. But clients intended for final users may not parse and display <ExtendedData> elements, so the use of an HTML alternative representation under <Description> is recommended.

GeoJSON

OpenSearch geo services delivering JSON responses (application/json mime type) should use GeoJSON, for the same reasons that RSS and Atom should use GeoRSS: It is a standard that enables service-independent parsing of geographic content.

OpenSearch specific information contained in responses is addressed in XML-based documents by means of opensearch[14] and atom[15] namespaces. JSON has not an associated schema. This lack of schema has led to each service adopting its own serialization format[16], what forces clients to develop specific parsers.

Fortunately GeoJSON fixes the structure to be used, but only for geographic content [12]. It is not defined how to describe other OpenSearch response intrinsic elements in JSON, as alternate links, paging information, the originating query, or the autodiscovery mechanism.

The following example, inspired in the Atom format, is proposed as a generic structure for OpenSearch JSON responses:

[13] http://www.georss.org/georss
[14] http://a9.com/-/spec/opensearch/1.1/
[15] http://www.w3.org/2005/Atom
[16] http://www.opensearch.org/Community/JSON_Formats

```
{ "opensearch": {
    "totalResults": 4230000, "startIndex": 21, "itemsPerPage": 10,
    "Query": {
      "role": "request", "searchTerms": "New York History",
      "startPage": 3, "geo": { "box": "-74.0667,40.69418,-73.9116,40.7722" }
    }
  },
  "links": [
    { "rel": "alternate",
      "type": "text/html",
      "href": "http://example.com/New+York+History?pw=3&bbox=-
74.0667,40.69418,-73.9116,40.7722" },
    { "rel": "alternate",
      "type": "application/atom+xml",
      "href": "http://example.com/New+York+History?pw=3&bbox=-
74.0667,40.69418,-73.9116,40.7722&format=atom" },
    { "rel": "self",
      "type": "application/json",
      "href": "http://example.com/New+York+History?pw=3&bbox=-
74.0667,40.69418,-73.9116,40.7722&format=json" },
    { "rel": "prev",
      "type": "application/json",
      "href": "http://example.com/New+York+History?pw=2&bbox=-
74.0667,40.69418,-73.9116,40.7722&format=json" },
    { "rel": "next",
      "type": "application/json",
      "href": "http://example.com/New+York+History?pw=4&bbox=-
74.0667,40.69418,-73.9116,40.7722&format=json" },
    { "rel": "first",
      "type": "application/json",
      "href": "http://example.com/New+York+History?pw=1&bbox=-
74.0667,40.69418,-73.9116,40.7722&format=json" },
    { "rel": "last",
      "type": "application/json",
      "href": "http://example.com/New+York+History?pw=42299&bbox=-
74.0667,40.69418,-73.9116,40.7722&format=json" },
    { "rel": "search",
      "type": "application/opensearchdescription+xml",
      "href": "http://example.com/opensearchdescription.xml" }
  ],
  "bbox": [-74.0667, 40.69418, -73.9116,  40.7722],
  "results": {
    "type": "FeatureCollection",
    "features": [
      { "type": "Feature",
        "geometry": {
          "type": "LineString",
          "coordinates": [
            [-73.9972, 40.73763], [-73.99167, 40.73519],
            [-73.99035, 40.737015], [-73.98914, 40.73643],
            [-73.990431, 40.734640], [-73.991504, 40.731617]
          ]
        },
        "properties": {
          "title": "New York History",
          "description": "... Union Square.NYC - A virtual tour ...",
          "links": [ { "href":
"http://www.columbia.edu/cu/lweb/eguids/amerihist/nyc.html" } ]
        }
      }
    ]
  }
}
```

IMPLEMENTATIONS

Existing search engines

Table 2 summarizes some on line OpenSearch geo engines at the time of writing.

Table 2: Existing OpenSearch geo engines, parameters and response formats

Name	Geo parameters	Response formats
Geocommons[17]	box	HTML, Atom, KML, JSON.
Terradue[18]	box, uid	RDF/DCLite4G (but declared as XHTML), HTML (not working), Metalink, GML (but declared as XHTML).
GeoNetwork (FAO)[19]	box (not working)	RSS (but declared as HTML)

Geocommons service georreferences results by means of a bounding box, described as a four vertex polygon in KML, and using the <georss:box> simple tag in Atom (this last not working at the time of writing, having all coordinates the zero -“0”- value). JSON response items have a “bbox” property, but don’t follow the GeoJSON format. Atom provides alternate and edit links for all entries. HTML does not use the geo microformat. Item descriptions are plain text.

Terradue service returns a RDF/XML format (see [7], annex E.3) containing DCLite4G[20] metadata description for each item. Georeferencing is achieved in <dct:spatial> tag, serialized in WKT [16] form. HTML format is not working (HTTP error code 404, ‘not supported’). GML Earth Observation Profile is another supported format.

GeoNetwork opensource is a standards based catalog application supporting “geo OpenSearch” since version 2.1.0[21]. There are many running instances, the main one being FAO’s[22]. ‘geo’ box parameter is ignored. It returns GeoRSS results, encoding bbox in GeoRSS-gml format. Item description is HTML inside a CDATA block. In order to improve OpenSearch geo support, we have submitted a patch to GeoNetwork’s code[23], that enables {geo:box?}, {geo:locationString?} and {geo:geometry?} request parameters, and adds HTML as response format. These improvements will presumably be available in next (2.5.0) version.

In conclusion, there are still few OpenSearch geo clients, each one with its own peculiarities, that could gain interoperability if guidelines stated in previous section were adopted.

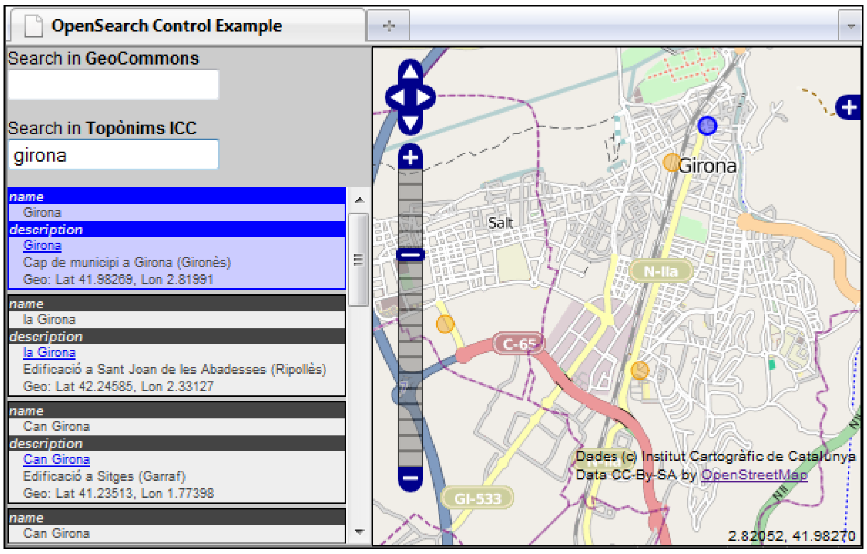
Proposed OpenLayers client

As far as we know, there are no OpenSearch geo clients that can be used as a reference implementation for testing purposes. Thus we are developing an OpenLayers[24] client[25] to easily add an OpenSearch geo control to web mapping applications:

```
new OpenLayers.Control.OpenSearch( {
       div: searchControlDiv,
       description: "http://example.com/opensearchdescription.xml"
    });
```

[17] http://core.geocommons.com/opensearch.xml
[18] http://catalogue.terradue.com/genesi/envisat_meris/mer_rr__1p/description/
[19] http://www.fao.org/geonetwork/srv/en/portal.opensearch
[20] http://wiki.osgeo.org/wiki/Geodata_Metadata_Requirements#Dublin_Core_lite_for_Geo_.28DClite4G.29
[21] http://geonetwork-opensource.org/software/geonetwork_opensource/releases/2.1.0
[22] http://www.fao.org/geonetwork/
[23] http://trac.osgeo.org/geonetwork/ticket/190
[24] http://www.openlayers.org
[25] http://trac.openlayers.org/ticket/2453

This client parses OpenSearch description documents, seeking for supported geo parameters and formats, automatically constructs a search box and selects a response format from those supported by OpenLayers (GeoRSS, Atom, KML or GeoJSON).


OpenSearch Control Example

Search in GeoCommons

Search in Topònims ICC

girona

name
Girona
description
Girona
Cap de municipi a Girona (Gironès)
Geo: Lat 41.98269, Lon 2.81991

name
la Girona
description
la Girona
Edificació a Sant Joan de les Abadesses (Ripollès)
Geo: Lat 42.24585, Lon 2.33127

name
Can Girona
description
Can Girona
Edificació a Sitges (Garraf)
Geo: Lat 41.23513, Lon 1.77398

name
Can Girona

Girona
Salt
N-IIa
C-65
GI-533
Dades (c) Institut Cartogràfic de Catalunya
Data CC-By-SA by OpenStreetMap
2.82052, 41.98270

Fig. 1. OpenSearch Control in OpenLayers

CONCLUSIONS

The simple OpenSearch-geo interface has great potential for geoweb's discoverability & usability, suitable for both highly specialized catalogs and simple geotagged content. There are few implementations whose particularities in response formats make them not always compatible with a generic client. There are two draft proposals, with the risk of incoherence between them and increasing complexity to the interface.

In this article we have discussed:

- Parameters described on both drafts, and a common proposal focused on simplicity,
- Guidelines on standard geographic content encoding in a variety of response formats,
- Actual OpenSearch geo search engines,
- Support for OpenSearch geo in GeoNetwork and OpenLayers.

These contributions are intended for discussion, in the hope that they can help bringing maturity to the proposal, and promote the proliferation of interoperable implementations.

Future work includes promoting these ideas to the specification proposals, increasing tools for potential implementers, further GeoNetwork service and OpenLayers client development to fully support the specification, and analysis on how other existing geoweb applications and standards could profit from OpenSearch geo.

ACKNOWLEDGEMENTS

We want to thank the support from Jo Walsh (Open Knowledge Foundation), Jose Garcia and Francois Prunayre (GeoNetwork developers), and Papeschi Fabrizio (Institute of Methodologies for Environmental Analysis, CNR).

This work has been partially supported by the European project EuroGEOSS and by the CENIT España Virtual project through the Instituto Geográfico Nacional (IGN).

REFERENCES

[1] ABARGUES, C. (2009) “Discovery and retrieval of Geographic data using Google.” *Master of Science in Geospatial Technologies;TGEO0011*. http://run.unl.pt/handle/10362/2536
[2] NEBERT, D. et al. (2007) “OpenGIS® Catalogue Services Specification. Version 2.0.2.” *Open Geospatial Consortium Inc.* Ref. OGC 07-006r1
[3] VRETANOS, P. (2005) “Web Feature Service Implementation Specification.” *Open Geospatial Consortium Inc.* Ref. OGC 04-094.
[4] WALSH, J. (2007) “On Spatial Data Search.” *Terradue White Paper.* Ref. T2-Research-07-003-OnSearch.
[5] CLINTON, D. et al. “OpenSearch 1.1 Specification (draft 4).” *Opensearch.org*
[6] TURNER, A. “OpenSearch Geo Extension 1.0 (draft 1).” *Opensearch.org*
[7] GONÇALVES, P. (editor) (2010) “OpenGIS® OpenSearch Geospatial Extensions Draft Implementation Standard. Version 0.0.2.” *Open Geospatial Consortium Inc.* Ref. OGC 09-084r3.
[8] “RSS 2.0 Specification. Version 2.0.11” (2009) *RSS Advisory Board*. http://www.rssboard.org/rss-specification.
[9] NOTTINGHAM M., SAYRE, R. (editors) (2005) “The Atom Syndication Format”. *IETF.* RFC 4287.
[10] “Creating OpenSearch plugins for Firefox” *Mozzila Developer Center.* https://developer.mozilla.org/en/Creating_OpenSearch_plugins_for_Firefox [last visited Feb 2010].
[11] “Search provider extensibility in Internet Exporer” *Microsoft Developer Network*. http://msdn.microsoft.com/en-us/library/cc848862%28VS.85%29.aspx#spe_addprov [last visited Feb 2010].
[12] “JavaScript Object Notation (JSON)” http://www.json.org/
[13] CLINTON, D. (editor) “OpenSearch Suggestions Extensions 1.1 (draft 1)”. *Opensearch.org*
[14] BEAUJARDIERE, J. (editor) (2006) “OpenGis® Web Map Server Implementation Specification. Version 1.3.0.” *Open Geospatial Consortium Inc.* Ref. OGC 06-042. Section 7.3.3. GetMap Request parameters (pp. 33-37).
[15] “GeoRSS”. http://www.georss.org [last visited Feb 2010].
[16] HERRING, J. R. (editor) (2006) “OpenGis® Implementation Specification for Geographic information – Simple feature access – Part 1: Common architecture. Version 1.2.0” *Open Geospatial Consortium Inc.* Ref. OGC 06-103r3. Chapter 7: Well-known Text Representation for Geometry (pp.53-63).
[17] BERNERS-LEE, T. et al. (1999) “Hypertext Transfer Protocol – HTTP/1.1” *IETF.* RFC 2616.
[18] BERNERS-LEE, T. et al. (2005) “Uniform Resource Identifier (URI): Generic Syntax” *IETF*. RFC 3986. Chapter 2. Characters.
[19] (2009) “ECMAScript Language Specification. 5th edition.” *ECMA International.* ECMA-262. 15.1.3. URI Handling Function Properties.
[20] WILSON, T. (editor) (2008) “OGC® KML. Version 2.2.0”. *Open Geospatial Consortium Inc.* Ref. OGC 07-147r2.
[21] “Geo microformat draft specification. http://microformats.org/wiki/geo [last visited Feb 2010].
[22] BUTLER, H., DALY, M., DOYLE, A., GILLIES, S., SCHAUB, T., SCHMIDT, C. (2008) “The GeoJSON Format Specification. Rev. 1.0.”. http://geojson.org/geojson-spec.html [last visited Feb 2010].