International Journal of Geo-Information

MDPI

Article

Discovering Land Cover Web Map Services from the Deep Web with JavaScript Invocation Rules

Dongyang Hou [1,2], Jun Chen [1,2,*] and Hao Wu [2]

[1] School of Environment Science and Spatial Informatics, China University of Mining and Technology, Xuzhou 221116, China; houdongyang1986@cumt.edu.cn

[2] National Geomatics Center of China, 28 Lianhuachi West Road, Beijing 100830, China; wuhao@nsdi.gov.cn

* Correspondence: chenjun@nsdi.gov.cn; Tel.: +86-10-6388-1088

Academic Editors: Jamal Jokar Arsanjani and Wolfgang Kainz

Received: 25 January 2016; Accepted: 23 June 2016; Published: 30 June 2016

Abstract: Automatic discovery of isolated land cover web map services (LCWMSs) can potentially help in sharing land cover data. Currently, various search engine-based and crawler-based approaches have been developed for finding services dispersed throughout the surface web. In fact, with the prevalence of geospatial web applications, a considerable number of LCWMSs are hidden in JavaScript code, which belongs to the deep web. However, discovering LCWMSs from JavaScript code remains an open challenge. This paper aims to solve this challenge by proposing a focused deep web crawler for finding more LCWMSs from deep web JavaScript code and the surface web. First, the names of a group of JavaScript links are abstracted as initial judgements. Through name matching, these judgements are utilized to judge whether or not the fetched webpages contain predefined JavaScript links that may prompt JavaScript code to invoke WMSs. Secondly, some JavaScript invocation functions and URL formats for WMS are summarized as JavaScript invocation rules from prior knowledge of how WMSs are employed and coded in JavaScript. These invocation rules are used to identify the JavaScript code for extracting candidate WMSs through rule matching. The above two operations are incorporated into a traditional focused crawling strategy situated between the tasks of fetching webpages and parsing webpages. Thirdly, LCWMSs are selected by matching services with a set of land cover keywords. Moreover, a search engine for LCWMSs is implemented that uses the focused deep web crawler to retrieve and integrate the LCWMSs it discovers. In the first experiment, eight online geospatial web applications serve as seed URLs (Uniform Resource Locators) and crawling scopes; the proposed crawler addresses only the JavaScript code in these eight applications. All 32 available WMSs hidden in JavaScript code were found using the proposed crawler, while not one WMS was discovered through the focused crawler-based approach. This result shows that the proposed crawler has the ability to discover WMSs hidden in JavaScript code. The second experiment uses 4842 seed URLs updated daily. The crawler found a total of 17,874 available WMSs, of which 11,901 were LCWMSs. Our approach discovered a greater number of services than those found using previous approaches. It indicates that the proposed crawler has a large advantage in discovering LCWMSs from the surface web and from JavaScript code. Furthermore, a simple case study demonstrates that the designed LCWMS search engine represents an important step towards realizing land cover information integration for global mapping and monitoring purposes.

Keywords: web map service; land cover; focused crawler; deep web; data integration

1. Introduction

A web map service (WMS) is an international standard protocol for publishing and accessing geo-referenced maps on the web [1–3]. This standard has facilitated the integration, access and value-added applications of geospatial information [1,4,5]. In the past few years, an increasing volume

of land cover data and maps has been made available through WMSs for facilitating on-line open data access [6,7], supporting collaborative data production [8,9], and assisting in crowd-sourcing sampling and validation [10–13]. One example is the WMS-based information service system, which enables the open access and sharing of one of the world's first 30 m Earth land cover maps, called GlobeLand30 (www.globeland30.org) [14]. There are many other land cover web map services (LCWMSs) that provide global, national or local land cover data/maps. Some of these LCWMSs are registered in catalogues (e.g., the Catalogue Service for the Web, CSW) on the basis of Service-Oriented Architecture [15,16]. These LCWMSs can be discovered easily by matching keywords in corresponding catalogues. Some LCWMSs are dispersed in the surface web, which refers to content stored in static pages [17]. These can be accessed directly by visiting the hyperlinks associated with these static pages [18]. However, others are hidden in the deep web, which refers to content hidden in dynamic pages (often behind query interfaces), in script code, and so on [17,19]. In particular, with the rapid adoption of the Open Geospatial Consortium (OGC) standards, such services are increasingly employed by geospatial web applications that use JavaScript code supplemented by third-party JavaScript libraries (e.g., OpenLayers and ArcGIS for JavaScript) [20,21]. This JavaScript code and the libraries are often referenced in the form of JavaScript links, such as "<script src= '../OpenLayers.js'></script>". Such deep web LCWMSs are difficult to discover by simply visiting hyperlinks. For example, the LCWMSs for GlobeLand30 can be discovered only by analysing the JavaScript code exposed by the service system (www.globeland30.org).

Recently, discovery and integration of these dispersed land cover data services have been stimulated by a wide range of development agendas, research programmes and practical applications [10,22–24]. One example is the United Nation's 2030 sustainable development agenda, which critically depends on the availability and utilization of land cover information at global, national and local scales. Unfortunately, no one single land cover data set or service can currently meet the needs of all users. Moreover, many LCWMSs exist as isolated "information islands" and are not well connected [25]; therefore, it is natural to consider linking all the land cover information services scattered around the world to provide a more reliable land cover information service. This issue arose and was discussed at the 9–10 June 2015 international workshop organized by the International Society for Photogrammetry and Remote Sensing (ISPRS) and the Group on Earth Observations (GEO) [22]. It was proposed to design and develop a Collaborative Global Land information service platform (CoGland); however, doing so faces a number of technical challenges [22]. One of these challenges is to automate the discovery and connection of the existing but isolated LCWMSs to form a "one stop" portal for an integrated information service [25].

Automatic discovery of LCWMSs can be realized in either a passive or active manner. The passive method uses a keyword-matching approach to discover services in registered catalogues [15,16,26]. The success of this method depends largely on the willingness of service providers to register their WMSs and make them available [15,16,26]. However, numerous services published on the web are not registered in any catalogues, and thus cannot be found through catalogue searches [2,27]. Various search engines, which use web crawlers to continuously traverse static pages [28–30], have been developed for finding services dispersed in the surface web [31,32]. General-purpose search engines (such as Google and Bing), customized search engines with general crawlers and focused crawlers are the three most commonly used approaches [16,27,31–34]. In essence, however, these active approaches can only find geospatial web services that reside in static pages. Nevertheless, a considerable number of WMSs exist behind query interfaces and are hidden within JavaScript code [16,20,21]. The key challenge here is to detect and understand the deep web query interfaces and JavaScript code that signify WMSs and to extract them. The detection and understanding of query interfaces has received some attention [35]; however, few efforts have been devoted to the detection and understanding of JavaScript code specifically for discovering WMSs. Therefore, discovering WMSs from JavaScript code in geospatial web applications remains an open question [21].

This paper aims to solve this problem. It proposes a focused deep web crawler that can find more LCWMSs from both the deep web's JavaScript code and from the surface web. First, a group of JavaScript link names are abstracted as initial judgements. Through name matching, these judgements are used to judge whether a fetched webpage contains a predefined JavaScript link that may execute other JavaScript code to potentially invoke WMSs. Secondly, some JavaScript invocation functions and URL formats for WMS are summarized as rules from prior knowledge of how WMSs are employed and presented in JavaScript code. Through rule matching, the rules can be used to understand how geospatial web applications invoke WMSs using JavaScript code for extracting candidate WMSs. The above operations are fused into a traditional focused crawling strategy situated between the tasks of fetching webpages and parsing webpages. Thirdly, LCWMSs are selected from the list of extracted candidate WMSs through WMS validation and matching land cover keywords. Finally, a LCWMS search engine (LCWMS-SE) is designed and implemented that uses a rule-based approach to assist users in retrieving the discovered LCWMSs.

The remainder of this paper is organized as follows. Section 2 reviews related work about both the surface and deep geospatial web service discovery as well as the discovery of other deep web resources from JavaScript code. Section 3 outlines the active discovery approach, including a description of the initial judgements, the JavaScript invocation rules and the proposed focused deep web crawler. The design and implementation of the search engine that retrieves the discovered LCWMSs is described in Section 4. Preliminary experiments and analysis are presented in Section 5, and Section 6 concludes the paper.

2. Related Work

A WMS provides three operations: GetCapabilities, GetMap, and GetFeatureInfo to support requests for metadata, static maps and feature information about the corresponding georeferenced maps, respectively [1,16]. The GetCapabilities operation is intended to request service metadata, such as service abstract, reference coordinate system, the format, etc. [16]. The GetMap operation enables users to obtain static maps from multiple servers by submitting parameters that include layer names, bounding boxes, format, styles, etc. [3,16]. The GetFeatureInfo operation requests metadata for the selected features [16]. Because metadata can help users find the service layers they need and determine how best to use them [36], in general, discovering WMSs means finding the URLs of the GetCapabilities operation. Other OGC geospatial services such as the Web Feature Service (WFS) and the Web Coverage Service (WCS) also have GetCapabilities operations. Therefore, discovery methods for those services are analogous to the discovery methods for WMSs.

2.1. Surface Geospatial Web Services Discovery

The most notable research for actively discovering surface geospatial web services can be divided into two types of approaches. The first type of discovery approach utilizes the application programming interfaces (APIs) of general-purpose search engines, employing predefined queries to search the Internet for discovering OGC geospatial web services [27,32,33,37,38]. For example, the Refractions Research OGC Survey [37] used the Google web API with two queries "request = getcapabilities" and "request = capabilities" to extract the results and then used a number of Perl "regular expressions" for a complete capabilities URL in each returned page. Lopez-Pellicer et al. [33] employed Bing, Google and Yahoo! APIs with two sets of 1000 queries to measure the performances of the three search engines in discovering OGC geospatial web services. They finally identified Yahoo! as the best performer. In the last two years, Bone et al. [32] designed a geospatial search engine for discovering multi-format geospatial data using the Google search engine with both user-specified and predefined terms. Kliment et al. [27,39] used the advanced query operators "inurl:" of the Google search engine to discover specific OGC geospatial services stored in Google's indexes. This type of discovery approach is easily implemented, is low cost and can avoid the performance issues of systems that attempt to crawl the entire web [32]. However, a general-purpose

search engine is intended to cover entire topics without considering the characteristics of OGC geospatial web services; thus, with this approach, actual OGC geospatial services will be inundated with a large amount of search results [16]. Moreover, although general-purpose search engines (e.g., Google) have already attempted to incorporate deep web resources hidden behind hypertext markup language (HTML) forms [35], they still ignore some resources hidden in JavaScript code, especially OGC geospatial web services invoked by JavaScript libraries. Furthermore, the public API of general-purpose search engines has some calling limitations imposed through permissions, allowed numbers of call, special policies, and so on. For instance, the API of the Google custom search engine provides only 100 free search queries per day [40].

The second type of discovery approach is to utilize customized search engines. Customized search engines can be categorized into general-purpose and topic-specific search engines [29]. Customized general-purpose search engines are equipped with a web crawler, which can continually collect large numbers of webpages by starting from a series of seed URLs (a list of URLs that the crawler starts with) and without a specific topic [32,41,42]. For example, Sample et al. [43] utilized Google search APIs to generate a set of candidate seed URLs for GIS-related websites and developed a web crawler to discover WMSs. Shen et al. [15] developed a WMS crawler based on the open source software NwebCrawler to support active service evaluation and quality monitoring. Patil et al. [44] proposed a framework for discovering WFSs using a spatial web crawler. Compared to general-purpose search engines, customized general-purpose search engines can easily control the process for discovering geospatial web services and can dramatically scale down the set of URLs to crawl. However, the crawled URLs still go far beyond only geospatial pages that contain geospatial web services [16].

To solve the above problems, topic-specific search engines with focused crawlers were developed. With a series of seed URLs and a given topic, focused crawlers aim to continually crawl as many webpages relevant to the given topic and keep the amount of irrelevant webpages crawled to the minimum [41]. For example, Lopez-Pellicer et al. [26,45] developed a focused crawler for geospatial web services. This crawler utilized Bing, Google and Yahoo! search APIs to generate seed URLs and used best first and shark-search strategies to assign priority to these URLs. Li et al. [16], Wu et al. [46] and Shen et al. [34] also developed focused crawlers based on different URL priority assignment approaches to discover OGC geospatial web services. Because they further scale down the set of URLs that must be crawled, these methods can improve discovery performance in both technical and economic perspectives [16]. However, the above crawlers in the customized search engines parse webpages solely through HTML parsing engines, which have no ability to identify and submit HTML forms and do not address JavaScript code. Therefore, these customized search engines are not able to discover WMSs in the deep web.

2.2. Deep Geospatial Web Services Discovery

The deep web refers to data hidden behind query interfaces, scripted code and other types of objects [17,47]. From this perspective, the underlying geospatial web services registered in catalogues are opaque because they are hidden behind catalogue query interfaces [47,48]; as such, geospatial web services registered in catalogues form part of the Deep Web. In general, searching services from known catalogues is implemented by keyword matching approaches [16]. However, a number of catalogues are unknown to the public. Therefore, discovering services from these unknown catalogues depends on successfully detecting and understanding the query interfaces of these unknown catalogues [47,48]. In the past few years, several works have been performed to address the query interface detection and understanding problem. For example, Dong et al. [49] proposed a novel deep web query interface matching approach based on evidence theory and task assignment. These works can be used for detecting and understanding the query interfaces of unknown catalogues, but their original purposes were not focused on discovering geospatial web services from the deep web.

Recently, some efforts have been made to discover geospatial web services from the deep web. For example, Lopez-Pellicer et al. [20,21] analysed the characteristics of the deep web geospatial content in

detail. They pointed out that disconnected content, opaque content, ignored content, dynamic content, real-time content, contextual content, scripted content and restricted content are all part of the deep web geospatial content. Furthermore, they summarized six heuristics on the characteristics of the deep web content in the form of texts. Then, they designed an advanced geospatial crawler with plugin architecture and introduced several extension plugins for implementing some of the heuristics to discover part of geospatial contents hidden in the deep web. Finally, they indicated that discovering services in geospatial web applications based on JavaScript libraries (e.g., OpenLayers) is still one of the open questions that require further research. This was one motivation behind this study's effort to develop a new active discovery approach. Therefore, our contribution lies in the development of a method to discover WMSs in geospatial web applications through focused crawling by inspecting the JavaScript code using a rule-matching approach.

2.3. Use of JavaScript Code in Deep Web Resources Discovery

Various resources hidden in JavaScript code comprise part of the deep web. Discovery of these resources has been the subject of research in the past few years. For example, Bhushan and Kumar [50] developed a deep web crawler by utilizing some regular expressions (Match/Replace) to extract clear-text links hidden in JavaScript code. Hou et al. [42] summarized matching rules formalized by a regular expression to extract the geographical coordinates of built-up areas from JavaScript code. Some resources dynamically generated by JavaScript code can be extracted using interpreters (i.e., the JavaScript parsing engine and Webkit). For instance, Hammer et al. [51] developed a WebKit-based web crawler to extract online user discussions of news. In geospatial web applications, WMSs are rendered as maps for visualization using geospatial JavaScript libraries; hence, the WMSs are hidden in JavaScript code. In general, these JavaScript libraries have concrete functions used to invoke WMSs. Learning from the discovery experiences of other deep web resources, this paper summarizes the invocation rules to extract candidate WMSs from JavaScript code.

3. Methodology

The previous discussions and analyses conclude that numerous LCWMSs exist in the JavaScript code of geospatial web applications and must be discovered. Discovering these LCWMSs requires addressing two challenges: detecting JavaScript code and understanding what that code does. Figure 1 represents the conceptual framework for discovering LCWMSs hidden in JavaScript code and the surface web. Compared with other focused crawler-based discovery methods, the proposed approach involves three major tasks. The first task is detecting predefined JavaScript links presented in the fetched webpages. This task is responsible for judging whether other JavaScript code potentially employs WMSs by matching the code against predefined judgements, which are composed of some JavaScript link names. The second task involves understanding the detected JavaScript code. It is the responsibility of this task to extract and validate candidate WMSs from JavaScript code through rule matching. Achieving this goal depends largely on JavaScript invocation rules, which are composed of WMS functions and their URL formats. The third task is to select available LCWMSs from the extracted candidate WMSs using a land cover keyword-matching approach. In Figure 1, the tasks in the dashed box are similar to other focused crawler-based discovery methods. More details are specified in the following subsections.

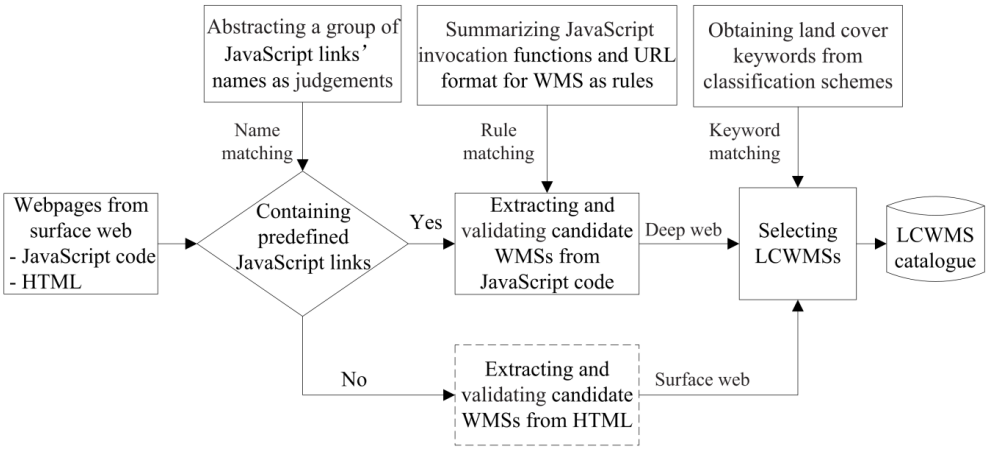

Figure 1. The conceptual framework for discovering LCWMSs.

3.1. Detection of JavaScript Links Using Judgements

Most geospatial web applications use third-party JavaScript libraries to invoke WMSs for map visualization. The four best-known JavaScript libraries are OpenLayers [52], ArcGIS API for JavaScript [53], Leaflet [54] and Mapbox.js [55]. For example, the GlobeLand30 [9,14] and CropScape [6] geospatial web applications were developed using OpenLayers. It is standard syntax to enclose a reference to a JavaScript library in a "<script>" tag when it is used to develop a web application with HTML [56], as shown in Table 1. The "src" property value of the "<script>" tag is a JavaScript link to specific external JavaScript code or library. Therefore, a webpage that refers to any of the four WMS-related JavaScript libraries is a reasonable candidate for potentially containing any WMSs. Based on this finding, this paper summarizes judgements to determine whether the fetched webpages have predefined JavaScript links that execute JavaScript code known to be related to WMSs.

Table 1. Reference formats of the four WMS-related JavaScript libraries.

WMS-Related JavaScript Libraries	Examples of Reference Formats
OpenLayers 2.x	<script src= ' ../OpenLayers.js'></script> <script src= ' ../OpenLayers.debug.js'></script>
OpenLayers 3.x	<script src= ' ../ol.js'></script> <script src= ' ../ol-debug.js'></script>
ArcGIS API for JavaScript	<script src= ' http://js.arcgis.com/3.14></script> <script src= '../arcgis_js_api/library/3.9/arcgis_compact'></script>
Leaflet	<script src="http://cdn.leafletjs.com/leaflet/v0.7.7/leaflet.js"></script>
Mapbox.js	<script src='https://api.mapbox.com/mapbox.js/v2.2.3/mapbox.js'></script>

The judgements are composed of some JavaScript link names that are predetermined based on the reference formats of OpenLayers, ArcGIS API for JavaScript, Leaflet and Mapbox.js as they appear in webpages, as shown in Table 1. These judgements include "openlayers," "ol.js," "ol-debug.js," "arcgis," "leaflet," and "mapbox." The judgements are performed by matching names with the JavaScript links in fetched webpages. When a JavaScript link in fetched webpages contains one of the predefined names, it indicates that other JavaScript code in the webpages may employ WMSs. Therefore, the other JavaScript code will be addressed by the JavaScript invocation rules described in this paper.

Moreover, two additional measures are adopted to further avoid traversing all JavaScript links in a geospatial web application. The first is to include some WMS-related keywords extracted from the naming schemes of many actual JavaScript links known to launch WMSs. These keywords include "map," "initial," "wms," "layer," "conus," "capabilities," "demo," "query," and "content." Only when

the name of a JavaScript link contains at least one of the above WMS-related keywords will the JavaScript link itself be addressed by two subsequent JavaScript invocation rules. The second adopted limiting measure is to summarize some keywords not related to WMSs from the naming schemes of numerous well-known JavaScript libraries, such as JQuery, ExtJS, Proj4js and AngularJS. The keywords mainly consist of "jquery," "ext-," "proj4js," "angularjs,""openlayers," "ol.js," "ol-debug.js," "leaflet," "mapbox" and so on. When the name of a JavaScript link contains one of the above WMS-unrelated keywords, the JavaScript link will be ignored.

3.2. Understanding of JavaScript Code Using Invocation Rules

Only a discovery approach that understands how geospatial web applications invoke WMSs by JavaScript libraries will be able to discover the WMSs in these applications [57]. Therefore, two JavaScript invocation rules are summarized based on development knowledge of geospatial web applications about WMSs to understand what such JavaScript code does.

As shown in Table 2, the first JavaScript invocation rule is derived from the OpenLayers [52], ArcGIS API for JavaScript [53], Leaflet [54] and Mapbox.js [55], all of which are JavaScript libraries that provide support for OGC's web mapping standards [58]. The listed total of seven common functions are often directly used to invoke WMSs in a large number of geospatial web applications. Therefore, to make the discovery method understand how geospatial web applications invoke WMSs, the seven functions collectively compose the first JavaScript invocation rule, which is formalized by the regular expression shown in Figure 2. The rule obtains JavaScript code fragments containing the URLS of candidate WMSs using a simple rule-matching approach. For example, the JavaScript code fragment "OpenLayers.Layer.WMS.Untiled ("Rivers," "http://129.174.131.7/cgi/wms_conuswater.cgi", ...)" in the CropScape geospatial web application [6] can be extracted by simply matching the first rule.

```
OpenLayers\s*\.\s*Layer\s*\.\s*WMS\s*\([\w\W\n]*?\) |
OpenLayers\s*\.\s*Layer\s*\.\s*WMS\s*\.\s*Untiled\s*\([\w\W\n]*?\) |
OpenLayers\s*\.\s*Layer\s*\.\s*WMS\s*\.\s*Post\s*\([\w\W\n]*?\) |
ol\s*\.\s*source\s*\.\s*ImageWMS\s* ([\w\W\n]*?\) | ol\s*\.\s*source\s*\.\s*TileWMS\s* ([\w\W\n]*?\) |
WMSLayer\s*\([\w\W\n]*?\) | L\s*\.\s*tileLayer\s*\.\s*wms\s*\([\w\W\n]*?\)
```

Figure 2. A regular expression representing the first rule.

Table 2. Invocation functions and the formalization of the first rule.

JavaScript Libraries	Invocation Functions	Parameter Description
OpenLayers 2.x	OpenLayers.Layer.WMS (name, url, params, options)	Parameter "url" represents a URL for a WMS
	OpenLayers.Layer.WMS.Untiled (name, url, params, options)	
	OpenLayers.Layer.WMS.Post (name, url, params, options)	
OpenLayers 3.x	ol.source.ImageWMS ({url:url, . . . })	The value in the key-value pair"url:url" represents a URL for a WMS
	ol.source.TileWMS ({url:url, . . . })	
ArcGIS API for JavaScript	WMSLayer (url, options)	Parameter "url" represents a URL for a WMS
Leaflet	L.tileLayer.wms (url, options)	
Mapbox.js	L.tileLayer.wms (url, options)	

The seven common functions are also deeply encapsulated in other functions that simply invoke WMSs. For example, in the GeoNetwork site [59], the function "OpenLayers.Layer.WMS" is encapsulated as the function "createWMSLayer," which takes only an array parameter as an argument to specify the base URLs of the WMS. It is impossible to exhaustively list all the encapsulated

functions, because different geospatial web applications may adopt different encapsulated function names. Therefore, to make the discovery method understand how encapsulated functions invoke WMSs, the second JavaScript invocation rule is composed of the base URL formats of the WMSs and is also formalized by a regular expression, as shown in Figure 3. The regular expressions for the two rules are compiled by following the rules of the C# language. Through a simple rule-matching approach, the second rule is used to extract http or https URLs that cannot be extracted by the first rule from JavaScript code. For example, no URL can be extracted by the first rule from the GeoNetwork site [59]; however, a URL can be extracted by matching the second rule from the JavaScript code fragment "GeoNetwork.WMSList.push ("Demo Cubewerx (WMS)-2," "http://demo.cubewerx.com/demo/cubeserv/cubeserv.cgi")" in the GeoNetwork site [59].

```
(http|https)\://[a-zA-Z0-9\-\.]+\.[a-zA-Z]{2,3}(:[a-zA-Z0-9]*)?/?
([a-zA-Z0-9\-\._\?\,\'/\\\+&$%\$#\=~])*
```

Figure 3. A regular expression of the second rule.

The two JavaScript invocation rules should be applied in a specific sequence, as shown in Figure 4. Only when Javascript code referenced by a fetched webpage is identified as the potential source of a WMS will the first rule be executed. Then, only if the first rule does not yield any URLs containing candidate WMSs will the second rule be executed. This sequence is necessary because the second rule is more general than the first rule; therefore, it acts to complement the first rule to capture all the URLs in the identified JavaScript code.

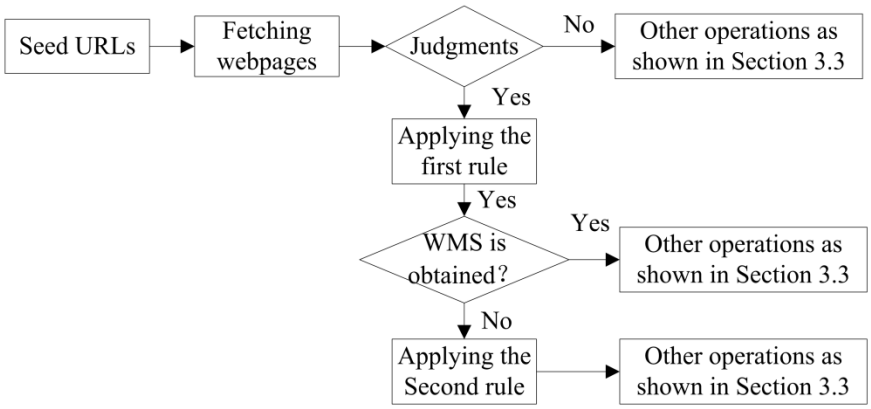

Figure 4. Instructions of the two JavaScript invocation rules.

Figure 5 presents the pseudocode to illustrate how to use the two JavaScript invocation rules. Steps 1–45 use the first JavaScript invocation rule to extract the URLs of potential WMSs. Steps 8–15 obtain a string for the URL of a potential WMS by splitting the matched string between the former two commas because the second argument represents the base URL for a WMS when calling functions in version 2 of the OpenLayers API (OpenLayers 2.x). Steps 16–21 also obtain a string for the URL of a potential WMS by splitting the matched string between the string "url:" and the first comma because the value of the key "url" represents the base URL for a WMS when calling functions in version 3 of the OpenLayers API (OpenLayers 3.x). Similarly, a string for the URL of a potential WMS is obtained by splitting the matched string between the first left parentheses and the first commas in Steps 22–27 because the first parameter represents the base URL for a WMS in ArcGIS API for JavaScript functions, Leaflet, and Mapbox.js. Steps 28–31 indicate that the extracted parameter is the URL of a potential WMS if it matches a URL format. When the extracted parameter is a JavaScript variable, a new function that potentially returns the URL of the WMS will be generated and executed by the Jurassic JavaScript

parsing engine by calling the CallGlobalFunction function [60], as shown in Steps 32–43. In these steps, the syntax for the new function is composed of the JavaScript variable and a return statement, as shown in Step 38. Steps 47–53 illustrate how to apply the second JavaScript invocation rule to extract the URLs of potential WMSs.

```
Input: jsCode, fR, sR: string variables for JavaScript code, first rule and second rule.
Output: URL_queue: a string array variable for potential WMSs' URLs
BEGIN
    M_JS: a string array variable for the matched JavaScript code
1   M_JS <= Matches(fR)
2   If ( m_JS != null) // the matched JavaScript code is not empty
3   {
4       For(int i=0;i< mJS.length;i++)
5       {
6           P_URL: a local string variable for a potential WMS's URL
7           B_num, l_num: local integer variables
8           If (m_JS [i].Contains("Layer.WMS"))
9           {
10              B_num <= m_JS [i].IndexOf(",") // Get the location of the first comma
11              // Get the second comma's location
12              L_num <= m_JS [i]. IndexOf (",", b_num +1)
13              //Get the substring of m_JS [i] between the former two commas
14              P_URL <= m_JS [i].SubString(b_num +1, l_num - b_num -1)
15          }
16          Else If (m_JS [i].Contains("ImageWMS") || m_JS [i].Contains("TileWMS"))
17          {
18              B_num <= m_JS [i].IndexOf("url:")
19              L_num <= m_JS [i]. IndexOf (",", b_num +1)
20              P_URL <= m_JS [i].SubString(b_num +4, l_num - b_num -4)
21          }
22          Else If  (m_JS [i].Contains("WMSLayer") || m_JS [i].Contains("tileLayer.wms"))
23          {
24              B_num <= m_JS [i].IndexOf("(")
25              L_num <= m_JS [i]. IndexOf (",", b_num +1)
26              P_URL <= m_JS [i].SubString(b_num +1, l_num - b_num -1)
27          }
28          If (p_URL.Contains("http://") || p_URL.Contains("https://"))
29          {
30              URL_queue.Add(p_URL)
31          }
32          Else  // in case that p_URL is a JavaScript variable name
33          {
34              S_pURL: a local string variable for JavaScript statement containing potential
35  URL
36              s_pURL <= Get the substring of jsCode between and containing the string "var
37  P_URL =" and its neighbouring semicolon
38              Generate a JavaScript function of "function GetValue() {  the value of s_pURL
39  Return p_URL }"
40              P_URL <= Execute the JavaScript function GetValue using JavaScript paring
41  Engine Jurassic
42              URL_queue.Add(p_URL)
43          }
44      }
45  }
46  Else
47  {
48          M_JS <= Matches(sR)
49          For(int j=0;j< m_JS.length;j++)
50          {
51              URL_queue.Add(m_JS [i])
52          }
53  }
END
```

Figure 5. Pseudocode of the use of the JavaScript invocation rules.

3.3. Discovery of Land Cover Web Map Services (LCWMSs) with a Focused Deep Web Crawler

A focused deep web crawler is proposed to discover LCWMSs from JavaScript code and the surface web. The focused deep web crawler mainly relies on the summarized judgements, JavaScript invocation rules and a JavaScript parsing engine for handling the JavaScript code. Moreover, as

mentioned in Section 2.1, a focused crawler is able to efficiently traverse the web to find WMSs in the surface web. Therefore, the proposed crawler uses the focused crawler to traverse the web for finding WMSs in the surface web. In the proposed crawler, the judgements serve as a bridge between the JavaScript invocation rules and the focused crawler.

Figure 6 illustrates the framework for the focused deep web crawler. The proposed crawler starts with a series of seed URLs. Next, it begins to fetch the webpages corresponding to the seed URLs or to crawled URLs. Then, the proposed crawler executes one of two branches according to certain judgements. When the judgements are fulfilled, the first branch executes. Otherwise, the second branch executes. The first branch mainly analyses any JavaScript code in the page with the Jurassic JavaScript parsing engine and the two JavaScript invocation rules, according to the pseudocode detailed above. It aims to obtain candidate WMS URLs from JavaScript code, while the second branch is intended to obtain candidate WMS URLs from the surface web. The second branch starts by parsing the HTML of webpages to extract their titles and contents. Then, a relevance calculation is executed using the traditional vector space model and the cosine formula. In this step, the vector space model is an algebraic model that represents webpages and the given topic as two vectors of keywords. The cosine function is a measure of similarity between each webpage vector and the topical vector by comparing the deviation of angles between the two vectors. This calculation is responsible for measuring the degree of similarity between the extracted webpages and the given topical keywords. If the relevance value is smaller than a given threshold, the webpage is discarded. Otherwise, when the relevance value is equal to or greater than the given threshold, the URLs in the webpage are extracted and a priority score for URLs will be assigned based on the texts in URLs, their parent webpages and anchor texts. Furthermore, to more precisely target candidate WMS URLs, any extracted URLs that end with common file extensions such as ".pdf," ".tif," ".js," ".doc," ".zip" and so on are discarded.

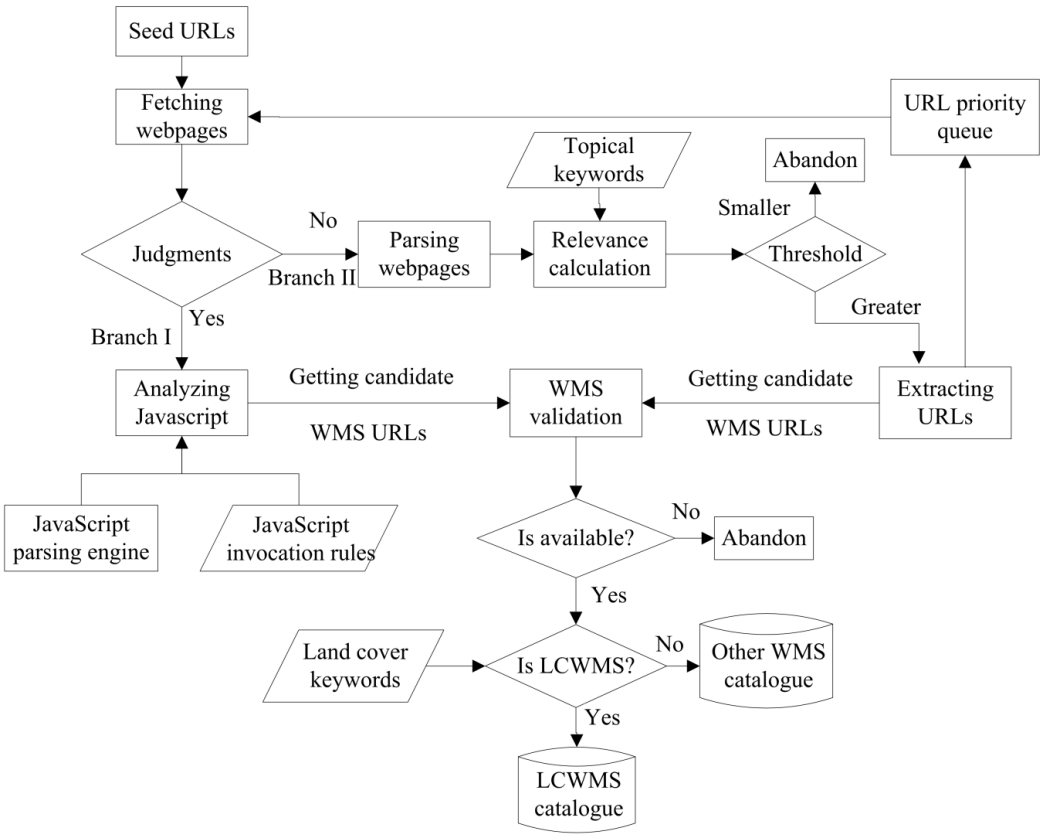

Figure 6. The framework of the focused deep web crawler for active discovery of LCWMSs.

After the candidate WMS URLs are obtained from these two branches, they are submitted to the component that performs WMS validation. The WMS validation component submits a GetCapabilities request to check whether the potential WMS URL corresponds to an available WMS. When the URL is not a valid WMS, it is discarded. When the URL is an available WMS, its capability file will be parsed to obtain metadata, such as the service name, service abstract, service keywords, bounding boxes, layer names, layer titles, layer abstracts and so on. Then, the available WMSs will be classified to identify the LCWMSs through matching land cover keywords. When the WMS metadata of service/layer names, titles, abstracts and keywords contain one of the land cover keywords, the WMS is classified as a LCWMS and stored in a LCWMS catalogue. Otherwise, the WMS is stored into a separate WMS catalogue. Finally, URLs in the URL priority queue will continue to be submitted for fetching webpages until the URL priority queue is empty or other conditions are fulfilled.

A total of 97 land cover keywords in English were collected from nine well-known classification schemes, as shown in Table 3. The nine well-known classification schemes include GlobeLand30 [9], International Geosphere-Biosphere programme (IGBP) [61], University of Maryland (UMD) [62], Global Land Cover 2000 (GLC 2000) [63], GlobCover [64], land-use monitoring using remote sensing from the Chinese Academy of Sciences [65], National Land Cover Database (NLCD) [66], Earth Observation for Sustainable Development of Forests (EOSD) [67] and the Dynamic Land Cover Dataset (DLCD) [68].

Table 3. Land cover keywords.

Representative Keywords	Land Cover Keywords	Number
Land cover	land cover, land use, landcover	3
Cropland	cropland, wheat, corn, rice, greenhouse, pasture, paddy, cultivated, hay, agricultural land, olive, vineyard, saccharum, farmland	14
Forest	forest, broadleaf, needleleaf, tree, woodland, evergreen	6
Grassland	grassland, savannas, sedge, graminoid, hummock, grass, meadow, herb, tussock	9
Shrubland	shrubland, scrub, dwarf, bush, brush	5
Wetland	wetland, marsh, bog, moor, intertidal, mangrove, mudflat, salina, beachland	9
Water	water, lake, reservoir, pond, river, ocean, sea, flood	8
Tundra	tundra, lichens, moss, cryptogam, bryoids	5
Impervious area	impervious area, urban, built-up, artificial surfaces, road, railway, airport, landfill, port, mining, settlement	11
Barren land	barren land, nudation, salt, alkali, sand, gravel, rock, biological crust, barren, bare ground, open space, unconsolidated shore, rubble, bedrock, bare soil, gobi, desert, bareland	18
Snow	snow, ice, glacier, colluvium	4
Vegetation	vegetation, herbaceous, plant, chenopod, forb	5

4. LCWMS-SE Implementation

4.1. LCWMS-SE Architecture

A working prototype of the LCWMS-SE was implemented based on the Microsoft NET Framework 3.5. The LCWMS-SE can automatically find LCWMSs from both the surface and deep webs. Moreover,

it enables users to retrieve their required services among the discovered LCWMSs. The LCWMS-SE is available, along with documentation, at [69]. The URL is temporary, but users will be able to track the progress of LCWMS discovery from the GlobeLand30 service system in future.

The LCWMS-SE comprises a focused deep web crawler, an indexing module, a retrieval module and a user query interface, as shown in Figure 7. The focused deep web crawler and indexing module are a desktop application based on C# WinForms. The information retrieval module and user query interface are a web application based on ASP.Net.

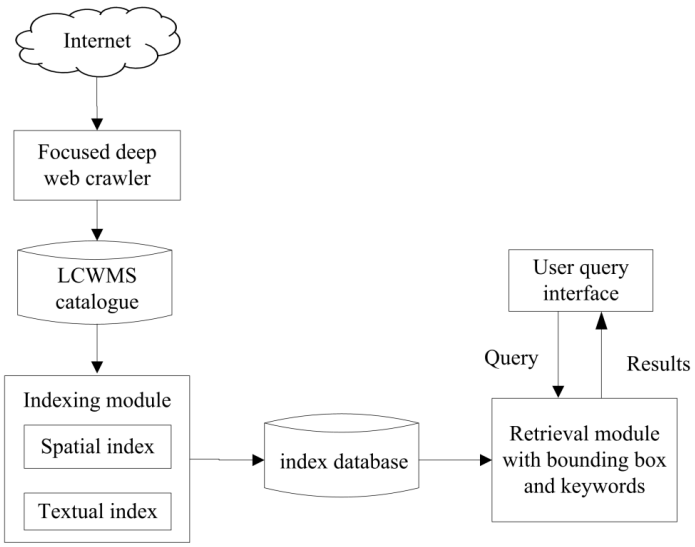

Figure 7. The LCWMS-SE architecture showing modules and linkages.

The focused deep web crawler is responsible for fetching and discovering LCWMSs from the internet as discussed in Section 3.3. The discovered LCWMSs are stored in an LCWMS catalogue. The indexing module is responsible for indexing the discovered LCWMSs in spatial and textual forms to generate an index database. The retrieval module, which allows users to enter a bounding box and keywords, provides search and rank functionalities. The user query interface allows users to submit queries, which are addressed through stop word filtering, synonym expansion and so on. Finally, the user query interface submits these queries to the retrieval module and presents ranked results to the users.

4.2. Indexing and Retrieving Modules

A bounding box is a mandatory parameter for users, who need to execute GetMap operations to obtain their required maps [70]. These users usually know the bounding boxes in advance when searching for their desired LCWMSs. When the bounding box is provided as a query parameter, the returned LCWMSs will better conform to the users' requirements. Therefore, the working prototype of the LCWMS-SE takes a user-specified bounding box into account, treating it as as a major query parameter.

Because one goal of the LCWMS-SE is to retrieve LCWMSs with respect to keywords and the user-specified bounding box, both textual and spatial information need to be indexed. The textual information was indexed as an inverted file structure using Lucene.Net 3.0.3 [71]. The indexed textual information contains the service name, service title, service abstract, service keywords, service URL, layer names, layer titles and layer abstracts. The spatial information was indexed using the BboxStrategy in Lucene.Net Contrib Spatial.NTS 3.0.3 [72]. The BboxStrategy is a spatial strategy for indexing and searching bounding boxes. In the BboxStrategy, the bounding box is stored as four numeric fields that represent the minimum and maximum X and Y coordinates, respectively. In the

search engine, only LCWMSs that have at least one bounding box using the EPSG:4326 (European Petroleum Survey Group) geographic coordinate reference system are spatially indexed.

On the basis of these textual and spatial indexes, this search engine implements three search modes. The first mode is based on a keyword-matching approach. The mode will search only for specified keywords in the title, name, abstract, and keywords at both the service level and the map layer level. The second search mode uses the bounding box-based method. This mode returns only LCWMS whose spatial extent has the specified spatial relationships with the specified bounding box. These spatial relationships refer to "contain," "contained" and "overlap." The third mode is a combination of the first two modes. When users select this query mode, the search engine returns only LCWMSs that contain the specified keywords and whose spatial extent has the given spatial relationships with the specified bounding box.

4.3. User Query Interface

The user query interface aims to both convey the users' requirements to the system and display the system's responses to the user. In other words, it is a bridge between the users and the system [73]. The user query interface of the search engine is composed of seven parts, including a text input field for entering query terms, an interactive graphical interface, a bounding box interface, a retrieval mode interface, a relationships interface, a search button and a textual list as shown in Figure 8.

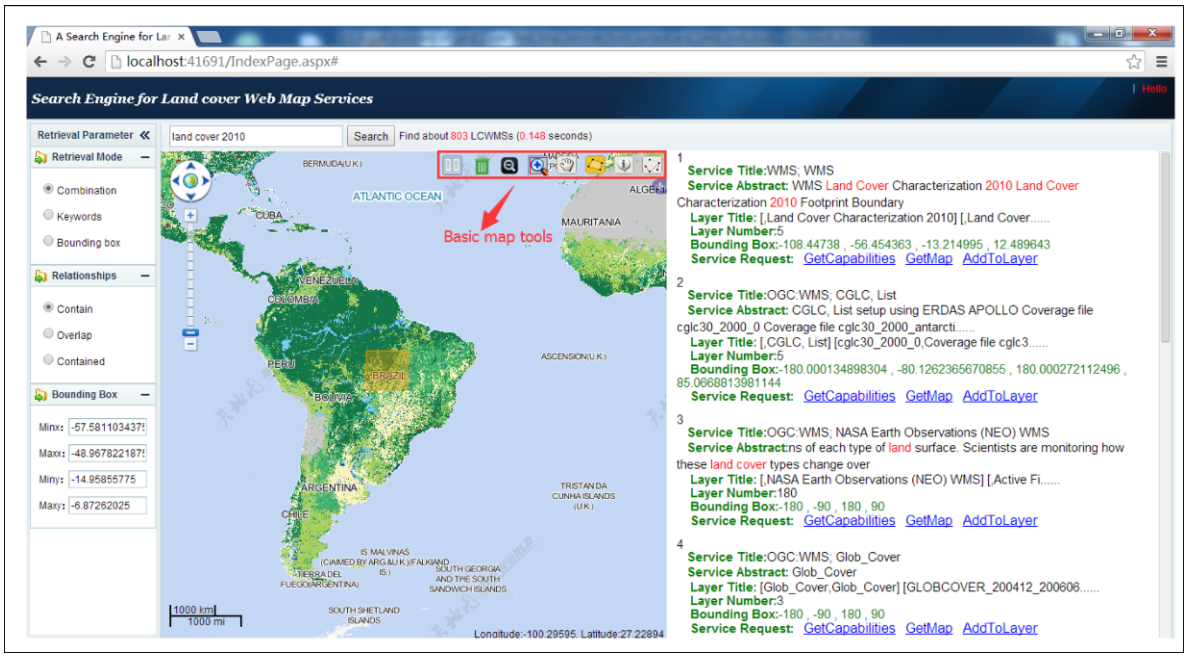

Figure 8. The user query interface to display land cover web map services.

Specifically, a text input for a query term enables users to specify what their desired LCWMSs should be about in the form of a list of keywords. Meanwhile, the bounding box interface and interactive graphical interface allow users to specify the area to which their desired LCWMSs should cover in the form of coordinates or a map. The interactive graphical interface was implemented using the OpenLayers API [52] and includes some basic map tools, such as pan, zoom, rectangle selection, etc. Its base map is the GlobeLand30 dataset [14] for the year 2010. The retrieval mode interface allows users to interact with the search engine in any of the three different modes described in Section 4.2. The relationships interface enables users to select among three different spatial relationships to restrict the search results in spatial forms. After entering keywords and/or the bounding box, assigning the retrieval mode and spatial relationship, the user clicks the search button and relevant matching

LCWMSs will be displayed in the text list. Every result contains the service title, service abstract, layer titles, layer number and bounding box. Service requests such as GetCapabilities, GetMap and AddToLayer are also provided, if the LCWMS has a bounding box that uses the EPSG:4326 geographic coordinate reference system. The AddToLayer request allows users to integrate the returned LCWMSs into the interactive graphical interface.

5. Experiments and Analysis

This section compares the performance of the proposed focused deep web crawler to a focused crawler-based approach, evaluates the enumerations of the LCWMSs that these crawlers could locate on the web, and demonstrates the LCWMS-SE's abilities concerning the retrieval and integration of discovered LCWMSs. The experiments are carried out using an Intel Pentium 4 CPU, running at 3.20 GHZ, with 1 GB of RAM, and 6 M of bandwidth.

5.1. Experiment 1: Discovering WMSs from JavaScript Code

This section presents an experiment carried out on 15 March 2015 that demonstrates the ability of the proposed focused deep web crawler to discover WMSs hidden in JavaScript code compared to a focused crawler-based approach. Table 4 lists the eight seed URLs used in the first experiment. The eight seed URLs correspond to eight online geospatial web applications that invoke a total of 43 unique WMSs. The 43 unique WMSs are all hidden in JavaScript code; of these, only 32 WMSs are available. In this experiment, the crawling scope, which refers to where the crawler can traverse the internet and retrieve webpages by hyperlinks, is restricted to only these eight online geospatial web applications.

Table 4. The eight seed URLs used for the first experiment.

	Name	URL	Number WMSs	literature
1	OneGeology Portal	http://portal.onegeology.org/	7	Duffy et al. [74]
2	CropScape	http://nassgeodata.gmu.edu/CropScape/	7	Han et al. [6]
3	GLC30 information service	http://www.globallandcover.com/	8	Chen et al. [9,14]
4	eHabitat WPS	http://ehabitat-wps.jrc.ec.europa.eu/ehabitat/	9	Dubois et al. [75]
5	GeoNetwork	http://www.fao.org/geonetwork/srv/en/main.home	5	Robinson et al. [76]
6	NIWA Ocean Survey	http://www.os2020.org.nz/project-map-sam/	4	Wood [77]
7	Visor xeográfico	https://sitegal.xunta.es/sitegal/ev/GeographicalViewer	1	Porta et al. [78]
8	ZOO Project	http://www.zoo-project.org/site/ZooWebSite/Demo/SpatialTools#ZOOspatialtoolsdemo	2	Fenoy et al. [79]

The proposed focused deep web crawler discovered all 32 available WMSs, while the focused crawler-based approach did not discover any WMSs. This indicates that the proposed focused deep web crawler has the ability to discover WMSs hidden in JavaScript code. Its success largely depends on the judgements, the JavaScript invocation rules and the JavaScript parsing engine discussed earlier in this paper. Using these, the focused deep web crawler knows how geospatial web applications invoke WMSs. In contrast, the focused crawler-based approach uses only an HTML parsing engine and, therefore, has no ability to investigate WMSs hidden in JavaScript code; consequently, it cannot discover the WMSs hidden in these URLs.

Section 2.3 discussed some studies that use rules to extract URLs from JavaScript code concerning the general domains. In addition, the open source crawler Heritrix has developed an extended module (ExtractorJS) to extract URLs from JavaScript code. To compare the efficiency between the existing crawlers with a JavaScript module and our approach, a supplementary experiment was conducted on 25 April 2016. The baseline for comparison is the Heritrix ExtractorJS module. The Seed URLs are the URLs numbered 2, 3, 5, 6 and 7 in Table 4 (the remaining three URLs are invalid). The crawling scope

is restricted to only these five online geospatial web applications. Both approaches discovered all the available WMSs hidden in JavaScript code. However, the approach using ExtractorJS extracted and validated 3266 URLs to discover those available WMSs, while our approach extracted and validated only 58 URLs to discover available WMSs, indicating that our approach has a higher efficiency than the approach with ExtractorJS. This is because the criteria, the two additional measures, and the first JavaScript invocation rules let our approach understand how those geospatial web applications invoke WMSs. In contrast, the approach using ExtractorJS utilizes a more general rule than the second JavaScript invocation rules in our approach to extract URLs from JavaScript code.

5.2. Experiment 2: Enumerating LCWMSs

The second experiment was carried out to enumerate the LCWMSs that the proposed crawler discovered on the web. In this experiment, there are two sets of seed URLs. The first set uses the same eight geospatial web URLs as Experiment 1. The second set is obtained by submitting a set of queries to the Bing Search API. These queries include mandatory keywords, land cover keywords and land cover product names. The mandatory keywords are associated with three operations of a WMS and the names of related JavaScript libraries, as shown in Table 5. The land cover keywords are listed in Table 2 in Section 3.2. A total of 31 land cover product names are reported in Table 6. Each Bing query is composed of a mandatory keyword and a land cover keyword or product name. Executing these queries against the Bing Search API resulted in a total of 4834 URLs. Moreover, additional queries are dynamically generated from the fetched webpages using a maximum-frequency method to perform daily updates to the list of seed URLs.

Table 5. Mandatory keywords.

Sources	Mandatory Keywords
Three operations	WMS, getcapabilities, getmap, service, getfeatureinfo
Related JavaScript libraries	JavaScript, OpenLayers, Arcgis, Mapbox, Leaflet

Table 6. Land cover product names.

Sources	Land Cover Product Names
Global	GLCC 2.0 IGBP, ISLSCP II IGBP, Modis Land cover, UMD, GLC2000, GLOBCOVER, GLCNMO, GLCSHARE, GEOWIKI, CCI-LC, GLC250, FROM-GLC, GlobeLand30
Regional	CORINE, LUCAS, GLOBCorine, PELCOM, NALCMS, NALCD, LBA-ECO, MERISAM2009, SERENA, AFRICOVER, NLCD, LCT, LCC, LULC, DLCD, NLUD-C, HLCR, RLC

This experiment ran intermittently from 16 June 2015 to 11 November 2015. A total of 17,874 available WMSs with 11,901 LCWMSs were discovered, as shown in Table 7. Table 7 indicates that the proposed approach discovers a considerably larger number of WMSs than those found by the previous works of Li et al. [16], Lopez-Pellicer et al. [26], Kliment et al. [27] and two well-known websites [80,81]. These previous works did not focus on a particular domain and they all searched for general WMSs. In addition, the discovered WMSs and LCWMSs have a total of 1886 unique WMS hosts, while Bone et al. [32] reported finding only 823 unique WMS hosts. The superior performance of the proposed approach is largely because the proposed approach finds extra WMSs in the deep web based on their JavaScript invocation rules and reduces the influence of topic drift to discover WMSs quickly and accurately by updating the seed URLs regularly.

Table 7. Comparison of the number of web map services (WMSs) found by various methods.

Authors	Year	Total Number of WMSs	Number of Available WMSs
Li et al.	2010	1126	1126
Lopez-Pellicer et al.	2012	6263	6263
Kliment et al.	2015	3694	2683
catalog.data.gov	2015	8585	Unknown
data.gov.uk	2015	2222	Unknown
Ours	2015	20478	17,874 (WMS) 11,901 (LCWMS)

However, as time goes on, the number of WMS naturally increases, which affects the credibility of the comparison. Therefore, the number of services found by Kliment et al. [27] and the two well-known websites [80,81] was tracked again on 27 April 2016. A total of 4233 WMSs with 2922 available WMSs were found in the website [82] provided by Kliment et al. [27]. In addition, a total of 8618 and 4029 WMSs were found by the two well-known websites [80,81], respectively. However, the number of WMSs discovered by our proposed approach is still larger than the number of WMSs found in the above three methods.

Moreover, seed URLs and crawling scope may also affect the credibility of the comparison. In Experiment 2, the seed URLs used in Li et al. [16] and the two well-known websites [80,81] are unknown. Similar to our approach, the seed URLs in Lopez-Pellicer et al. [26] were supplied by submitting a set of queries to the Bing Search API and the Google Web Search API. This way is similar to our approach. Kliment et al. [27] used a Google search engine-based method to discover services. This method has no need of seed URLs, but the Google search engine has more seed URLs than our approach. Furthermore, none of these approaches imposed any limits on the crawling scope. In this experiment, the seed URLs had only a small effect on the final results because all the approaches executed over a long period of time (in fact, some approaches are still running today). Therefore, seed URLs and crawling scope have minimal effect on the credibility of the comparison in this experiment.

Although our approach has discovered more WMSs and LCWMSs than the compared approaches, those approaches are still valid and necessary. Because they are complementary to each other and can be utilized to create a good balance between the costs required for WMS discovery and efficiency. For example, the Google search engine-based method used in Kliment et al. [27], which has low development costs, could be combined with our approach to improve discovery efficiency.

Figure 9 represents the spatial distribution and numbers of the discovered LCWMSs in our approach. The spatial location of a LCWMS can be obtained through its IP (Internet Protocol) address. The most important locations of LCWMSs (where 5 or more were found) are labelled in Figure 8. The numerators and denominators of these labels represent the numbers of LCWMSs and WMSs, respectively.

The LCWMSs discovered above can be subdivided into real LCWMSs and generalized LCWMSs based on their original data types. Real LCWMSs are published directly from land cover data and maps, while generalized LCWMSs are released from geographic data and maps related to land cover. To estimate the number of real LCWMSs in the discovered LCWMSs, a split-and-random sampling strategy is carried out. First, the number of LCWMSs containing data about each class except vegetation in Table 3 is calculated through matching the land cover keywords of each class in a prioritized sequence. For example, a LCWMS is classified as cropland if its metadata contains one of the cropland keywords in Table 3. Then, ten random LCWMSs for each class are selected as samples to manually determine whether they are real LCWMSs. This process was carried out five times (only one time for the shrubland and tundra classes). Table 8 represents the number of LCWMSs containing data about each class and the average ratio of real LCWMSs.

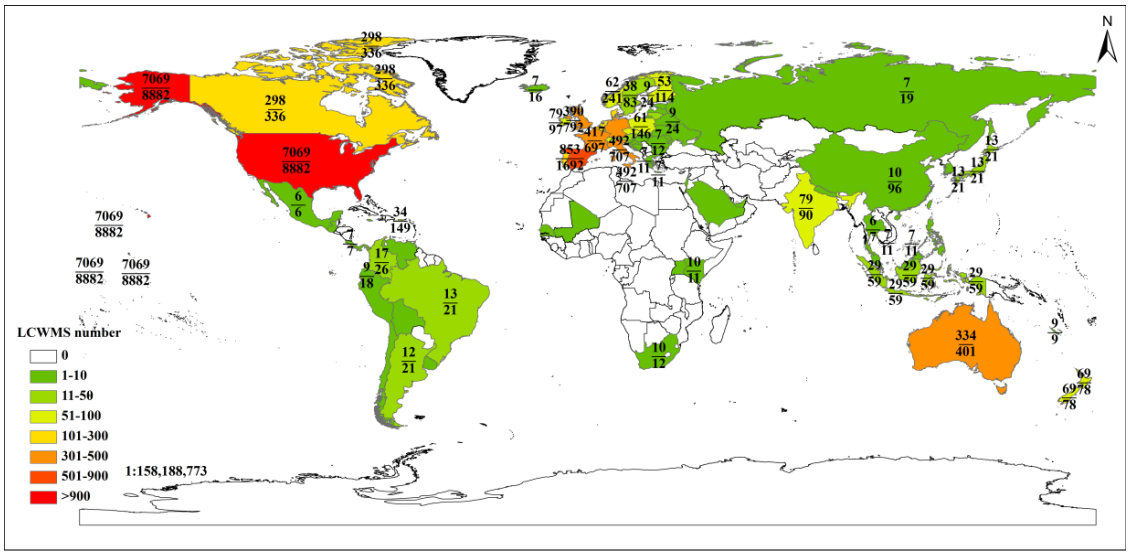

Figure 9. Spatial distribution and numbers of the discovered LCWMSs.

Table 8. Number of LCWMSs about each class and the average ratio of real LCWMSs.

Representative Keywords	Number of LCWMSs	Ratio of Real LCWMSs
Land cover	789	90%
Cropland	354	50%
Forest	484	60%
Grassland	119	80%
Shrubland	16	50%
Wetland	198	60%
Water	3471	40%
Tundra	8	37.5%
Impervious area	1430	50%
Barren land	482	30%
Snow	4074	40%

It can be observed from Table 8 that the ratio of real LCWMSs in the discovered LCWMSs vary with different classes. Up to 90% of the real LCWMSs contain data about the land cover class. These real LCWMSs include the US Geological Survey's NLCD, land cover data of South America for the year 2010, GlobeLand30 data, and so on. However, only approximately 30% of the real LCWMSs contain data about the barren land class. This is because many LCWMSs in this class are published from geological survey data, whose metadata often contains one or more of the barren land keywords in Table 3.

5.3. Case Study for Retrieval and Integration

Additionally, a case study is presented to demonstrate the retrieval and integration of the discovered LCWMSs in the LCWMS-SE. Assume that a user wants to search for new land cover datasets around Brazil to compare them with the GlobeLand30 dataset for the year 2010. Accordingly, to match the query's intent, the keywords "land cover 2010" and a bounding box around Brazil should be submitted to the designed search engine. As shown in Figure 10, a total of 803 LCWMSs were found among the discovered LCWMSs. The query takes approximately 0.148 s. The bounding boxes of all the returned LCWMSs contain the specified bounding box. Figure 11 represents the integration between the land cover data from the GlobeLand30-2010 dataset and that of South America for the year 2010

with 30 m resolution after a user clicked the "AddToLayer" link (red arrow). This demonstrates that the designed search engine has the ability to retrieve and integrate LCWMSs.

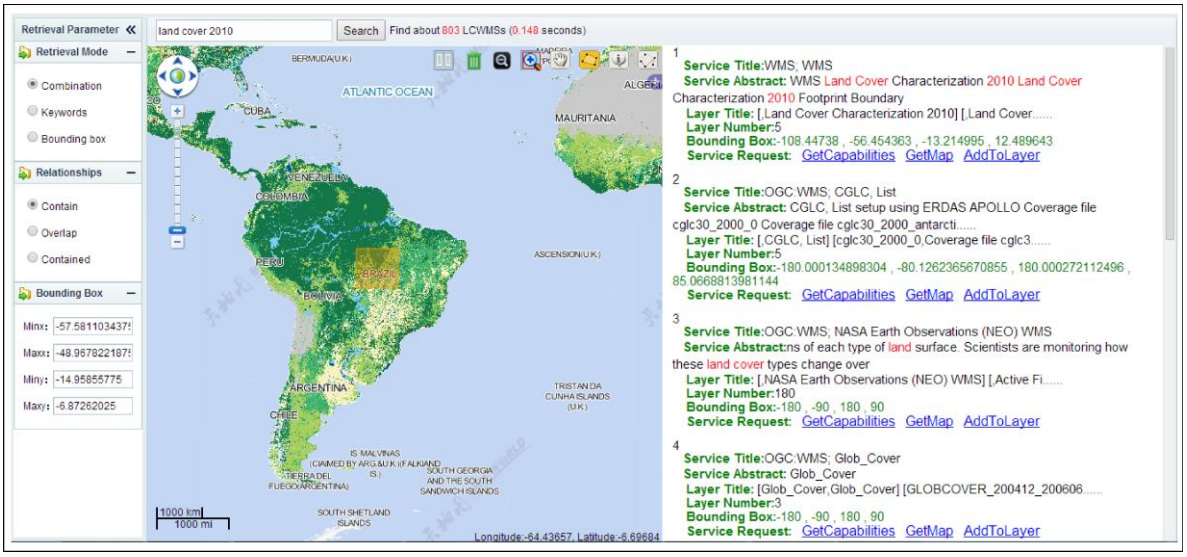

Figure 10. The user interface that displays LCWMSs for the case study.

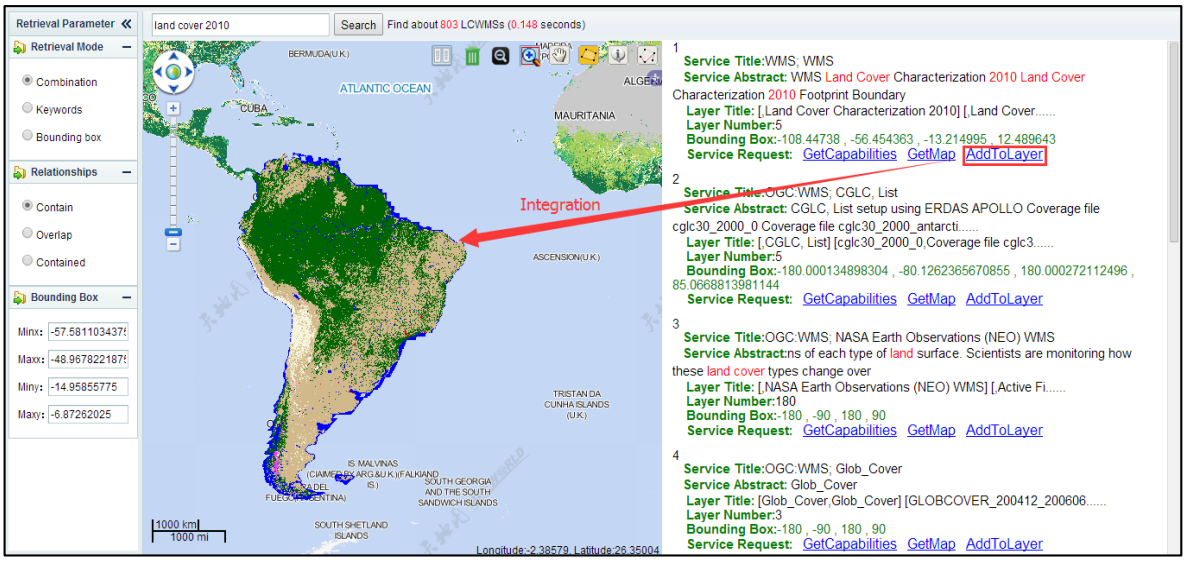

Figure 11. Interface displaying the integration of land cover web map services.

6. Conclusions

Automatic discovery and connections to isolated land cover web map services (LCWMSs) help to potentially share land cover data, which can support the United Nations's 2030 sustainable development agenda and conform to the goals of CoGLand. Previous active discovery approaches have been aimed primarily at finding LCWMSs located in the surface web and behind query interfaces in the deep web. However, very few efforts have been devoted to discovering LCWMSs from deep web JavaScript code, due to the challenges involved in web map service (WMS)-related JavaScript code detection and understanding. In this paper, a focused deep web crawler was proposed to solve these problems and discover additional LCWMSs. First, some judgements and two JavaScript invocation rules were created to detect and understand WMS-related JavaScript code for extracting candidate WMSs. These are composed of some names of predefined JavaScript links and regular

expressions that match the invocation functions and URL formats of WMSs, respectively. Then, identified candidates are incorporated into a traditional focused crawling strategy, situated between the tasks of fetching webpages and parsing webpages. Thirdly, LCWMSs are selected by matching with land cover keywords. In addition, a LCWMS search engine was implemented based on the focused deep web crawler to assist users in retrieving and integrating the discovered LCWMSs.

Experiments showed that the proposed focused deep web crawler has the ability to discover LCWMSs hidden in JavaScript code. By searching the worldwide web, the proposed crawler found a total of 17,874 available WMSs, of which 11,901 were LCWMS. The results of a case study for retrieving land cover datasets around Brazil indicate that the designed LCWMS search engine represents an important step towards realizing land cover information integration for global mapping and monitoring purposes.

Despite the advantages of the proposed crawler, much work remains to improve the effectiveness of discovering LCWMSs. First, the proposed crawler considers only one script type (JavaScript). However, other script types (i.e., ActionScript) can also invoke LCWMSs. In the future, additional rules will be summarized to help the proposed crawler discover LCWMSs invoked from other scripting languages. Secondly, some currently available LCWMSs may become unavailable in the future. Future work will include a monitoring mechanism to monitor and assess the quality of the discovered LCWMSs. Third, the proposed crawler utilized the cosine function to measure topical relevance and to assign priorities to URLs. Moreover, it adopted a keyword-matching approach to classify each WMS. In the future, machine-learning approaches such as support vector machines will be applied to measure topical relevance, assign URL priorities and classify WMSs.

Acknowledgments: This work has been funded by the National Science Foundation of China (Project #41231172 and #41301412). The authors would like to thank the journal editor and reviewers for their constructive comments, which helped improve the paper.

Author Contributions: Dongyang Hou conceived the active discovery approach, performed the experiments, and drafted the manuscript. Jun Chen and Hao Wu made substantial contributions to methodological development and preparation of the manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

WMS	Web Map Service
LCWMS	Land Cover Web Map Service
URL	Uniform Resource Locator
CSW	Catalogue Service for Web
OGC	Open Geospatial Consortium
ISPRS	Photogrammetry and Remote Sensing
GEO	Group on Earth Observations
CoGland	Collaborative Global Land information service platform
LCWMS-SE	LCWMSs Search Engine
WFS	Web Feature Service
WCS	Web Coverage Service
HTML	Hypertext Markup Language

References

1. Zhang, C.; Li, W. The roles of web feature and web map services in real-time geospatial data sharing for time-critical applications. *Cartogr. Geogr. Inf. Sci.* **2005**, *32*, 269–283. [CrossRef]
2. Florczyk, A.J.; Nogueras-Iso, J.; Zarazaga-Soria, F.J.; Béjar, R. Identifying orthoimages in Web Map Services. *Comput. Geosci.* **2012**, *47*, 130–142. [CrossRef]
3. Li, W.; Song, M.; Zhou, B.; Cao, K.; Gao, S. Performance improvement techniques for geospatial web services in a cyberinfrastructure environment—A case study with a disaster management portal. *Comput. Environ. Urban Syst.* **2015**, *54*, 314–325. [CrossRef]

4. Plesa, L. The design, implementation and operation of the JPL OnEarth WMS Server. In *Geospatial Services and Applications for the Internet*, 1st ed.; Sample, J.T., Shaw, K., Tu, S., Abdelguerfi, M., Eds.; Springer: New York, NY, USA, 2008; Volume 5, pp. 93–109.
5. Hu, C.; Zhao, Y.; Li, J.; Ma, D.; Li, X. Geospatial web service for remote sensing data visualization. In Proceedings of the 2011 IEEE International Conference on Advanced Information Networking and Applications (AINA), Biopolis, Singapore, 22–25 March 2011; pp. 594–601.
6. Han, W.; Yang, Z.; Di, L.; Mueller, R. CropScape: A web service based application for exploring and disseminating US conterminous geospatial cropland data products for decision support. *Comput. Electr. Agric.* **2012**, *84*, 111–123. [CrossRef]
7. Stepinski, T.F.; Netzel, P.; Jasiewicz, J. LandEx—A GeoWeb tool for query and retrieval of spatial patterns in land cover datasets. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2014**, *7*, 257–266. [CrossRef]
8. Han, G.; Chen, J.; He, C.; Li, S.; Wu, H.; Liao, A.; Peng, S. A web-based system for supporting global land cover data production. *ISPRS J. Photogramm. Remote Sens.* **2015**, *103*, 66–80. [CrossRef]
9. Chen, J.; Chen, J.; Liao, A.; Cao, X.; Chen, L.; Chen, X.; He, C.; Han, G.; Peng, S.; Lu, M.; et al. Global land cover mapping at 30 m resolution: A POK-based operational approach. *ISPRS J. Photogramm. Remote Sens.* **2015**, *103*, 7–27. [CrossRef]
10. See, L.; Perger, C.; Hofer, M.; Weichselbaumb, J.; Dresel, C.; Fritz, S. Laco-Wiki: An open access online portal for land cover validation. *ISPRS Ann. Photogramme. Remote Sens. Spat. Inf. Sci.* **2015**, *1*, 167–171. [CrossRef]
11. Fritz, S.; McCallum, I.; Schill, C.; Perger, C.; Grillmayer, R.; Achard, F.; Kraxner, F.; Obersteiner, M. Geo-Wiki.Org: The use of crowdsourcing to improve global land cover. *Remote Sens.* **2009**, *1*, 345–354. [CrossRef]
12. Fritz, S.; McCallum, I.; Schill, C.; Perger, C.; See, L.; Schepaschenko, D.; van der Velde, M.; Kraxner, F.; Obersteiner, M. Geo-Wiki: An online platform for improving global land cover. *Environ. Model. Softw.* **2012**, *31*, 110–123. [CrossRef]
13. Bastin, L.; Buchanan, G.; Beresford, A.; Pekel, J.F.; Dubois, G. Open-source mapping and services for Web-based land-cover validation. *Ecol. Inf.* **2013**, *14*, 9–16. [CrossRef]
14. Chen, J.; Ban, Y.; Li, S. China: Open access to earth land-cover map. *Nature* **2014**, *514*, 434–434.
15. Shen, S.; Zhang, T.; Wu, H.; Liu, Z. A catalogue service for internet GIServices supporting active service evaluation and real-time quality monitoring. *Trans. GIS* **2012**, *16*, 745–761. [CrossRef]
16. Li, W.; Yang, C.; Yang, C. An active crawler for discovering geospatial web services and their distribution pattern—A case study of OGC Web Map Service. *Int. J. Geogr. Inf. Sci.* **2010**, *24*, 1127–1147. [CrossRef]
17. Piccinini, H.; Casanova, M.; Leme, L.P.; Furtado, A. Publishing deep web geographic data. *GeoInformatica* **2014**, *18*, 769–792. [CrossRef]
18. Dixit, A.; Bhatia, K.K.; Yadav, J. Design and implementation of domain based semantic hidden Web crawler. *Int. J. Innov. Adv. Comput. Sci.* **2015**, *4*, 73–84.
19. Manvi, M.; Dixit, A.; Bhatia, K.K. Design of an ontology based adaptive crawler for hidden Web. In Proceedings of the International Conference on Communication Systems and Network Technologies (CSNT), Gwalior, India, 6–8 April 2013; IEEE: Washington, DC, USA, 2013; pp. 659–663.
20. Lopez-Pellicer, F.J.; Béjar, R.; Soria, F.J.Z. Crawling invisible geospatial endpoints. In *Providing Semantic Links to the Invisible Geospatial Web*, 1st ed.; Universidad de Zaragoza: Zaragoza, Spain, 2012; pp. 20–71.
21. Lopez-Pellicer, F.J. Semantic Linkage of the Invisible Geospatial Web. Ph.D. Thesis, Universidad de Zaragoza, Zaragoza, Spain, 2011.
22. ISPRS Report on the International Workshop on Supporting Future Earth with Global Geo-Information. Available online: http://www.isprs.org/news/newsletter/2015--03/53_Report_on_Beijing_Workshop.pdf (accessed on 1 November 2015).
23. Sustainable Development Knowledge Platform. Available online: https://sustainabledevelopment.un.org/sdgsproposal (accessed on 13 January 2016).
24. Latham, J.; Cumani, R.; Rosati, I.; Bloise, M. Global Land Cover SHARE (GLC-SHARE) Database Beta-Release, Version 1.0. Available online: http://www.fao.org/uploads/media/glc-share-doc.pdf (accessed on 7 January 2016).
25. Chen, J.; Wu, H.; Li, S.; Chen, F.; Han, G. Services oriented dynamic computing for land cover big data. *J. Geomat. Sci. Technol.* **2013**, *30*, 369–374.

26. Lopez-Pellicer, F.J.; Rentería-Agualimpia, W.; Nogueras-Iso, J.; Zarazaga-Soria, F.J.; Muro-Medrano, P.R. Towards an active directory of geospatial web services. In Proceedings of the International AGILE'2012 Conference, Avignon, France, 24–27 April 2012; pp. 63–79.
27. Kliment, T.; Kliment, M.; Tuchyňa, M. Making more OGC services available on the Web discoverable for the SDI Community. In Proceedings of the SGEM 2015 GeoConference on Informatics, Geoinformatics and Remote Sensing, Albena, Bulgaria, 16–25 June 2015.
28. Hou, D.; Wu, H.; Chen, J.; Li, R. A focused crawler for borderlands situation information with geographical properties of place names. *Sustainability* **2014**, *6*, 6529–6552. [CrossRef]
29. Almpanidis, G.; Kotropoulos, C.; Pitas, I. Combining text and link analysis for focused crawling—An application for vertical search engines. *Inf. Syst.* **2007**, *32*, 886–908. [CrossRef]
30. Wilkas, L.R.; Villarruel, A. An introduction to search engines. *J. Spec. Pediatr. Nurs.* **2001**, *6*, 149–151. [CrossRef]
31. Lopez-Pellicer, F.J.; Béjar, R.; Florczyk, A.J.; Muro-Medrano, P.R.; Zarazaga-Soria, F.J. A review of the implementation of OGC Web Services across Europe. *Int. J. Spat. Data Infrastruct. Res.* **2011**, *6*, 168–186.
32. Bone, C.; Ager, A.; Bunzel, K.; Tierney, L. A geospatial search engine for discovering multi-format geospatial data across the web. *Int. J. Dig. Earth* **2016**, *9*, 47–62. [CrossRef]
33. Lopez-Pellicer, F.J.; Florczyk, A.J.; Béjar, R.; Muro-Medrano, P.R.; Zarazaga-Soria, F.J. Discovering geographic web services in search engines. *Online Inf. Rev.* **2011**, *35*, 909–927.
34. Shen, P.; Gui, Z.; You, L.; Hu, K.; Wu, H. A topic crawler for discovering geospatial Web services. *J. Geo-Inf. Sci.* **2015**, *17*, 185–190.
35. Madhavan, J.; Ko, D.; Kot, Ł.; Ganapathy, V.; Rasmussen, A.; Halevy, A. Google's deep web crawl. In Proceedings of the VLDB Endowment, Auckland, New Zealand, 23–28 August 2008; pp. 1241–1252.
36. Mehdi, S.A.; Ali, M.; Nima, G.; Zahra, R.; Reyhaneh, S.; Peyman, B. How to implement a governmental open source geoportal. *J. Geogr. Inf. Syst.* **2014**, *6*, 275–285. [CrossRef]
37. Refractions Research: OGC Services Survey Article. Available online: http://www.refractions.net/expertise/whitepapers/ogcsurvey/ogcsurvey/ (accessed on 13 September 2015).
38. Shulman, L.; Ioup, E.; SAMPLE, J.; Shaw, K.; Abdelguerfi, M. Automated web service discovery. In Proceedings of the DASFAA2007 International Workshop on Scalable Web Information Integration and Service (SWIIS2007), Bangkok, Thailand, 9–12 April 2007; pp. 56–64.
39. Project Overview | Borderless Geospatial Web. Available online: http://bolegweb.geof.unizg.hr/site8/overview (accessed on 13 September 2015).
40. Custom Search JSON/Atom API | Custom Search | Google Developers. Available online: https://developers.google.com/custom-search/json-api/v1/overview (accessed on 13 September 2015).
41. Batsakis, S.; Petrakis, E.G.; Milios, E. Improving the performance of focused web crawlers. *Data Knowl. Eng.* **2009**, *68*, 1001–1013. [CrossRef]
42. Hou, D.; Chen, J.; Wu, H.; Li, S.; Chen, F.; Zhang, W. Active collection of land cover sample data from geo-tagged web texts. *Remote Sens.* **2015**, *7*, 5805–5827. [CrossRef]
43. Sample, J.T.; Ladner, R.; Shulman, L.; Ioup, E.; Petry, F.; Warner, E.; Shaw, K.; McCreedy, F.P. Enhancing the US Navy's GIDB portal with web services. *IEEE Inter. Comput.* **2006**, *10*, 53–60. [CrossRef]
44. Patil, S.; Bhattacharjee, S.; Ghosh, S.K. A spatial web crawler for discovering geo-servers and semantic referencing with spatial features. In Proceedings of the 10th International Conference on ICDCIT 2014, Bhubaneswar, India, 6–9 February 2014; pp. 68–78.
45. Lopez-Pellicer, F.J.; RenteríA-Agualimpia, W.; Béjar, R.; Muro-Medrano, P.R.; Zarazaga-Soria, F.J. Availability of the OGC geoprocessing standard: March 2011 reality check. *Comput. Geosci.* **2012**, *47*, 13–19. [CrossRef]
46. Wu, H.; Liao, A.; He, C.; Hou, D. Topic-relevance based crawler for geographic information web services. *Geogr. Geo-Inf. Sci.* **2012**, *28*, 27–30.
47. Reid, J.; Waites, W.; Butchart, B. An infrastructure for publishing geospatial metadata as open linked metadata. In Proceedings of the AGILE 2012 International Conference on Geographic Information Science, Avignon, France, 24–27 April 2012; pp. 99–104.
48. Lopez-Pellicer, F.J.; Florczyk, A.J.; Nogueras-Iso, J.; Muro-Medrano, P.R.; Zarazaga-Soria, F.J. Exposing CSW catalogues as linked data. In *Geospatial Thinking*, 1st ed.; Painho, M., Santos, M.Y., Pundt, H., Eds.; Springer: Berlin, Germany, 2010; pp. 183–200.

49. Dong, Y.; Li, Q.; Ding, Y.; Peng, Z. ETTA-IM: A deep web query interface matching approach based on evidence theory and task assignment. *Exp. Syst. Appl.* **2011**, *38*, 10218–10228. [CrossRef]
50. Bhushan, B.; Kumar, N. Surfacing deep web behind scripting languages. *Int. J. Comput. Sci. Netw. Secur.* **2012**, *12*, 55–58.
51. Hammer, H.; Bratterud, A.; Fagernes, S. Crawling JavaScript websites using WebKit—With application to analysis of hate speech in online discussions. In Proceedings of the Norsk Informatikkonferanse (NIK), Stavanger, Norway, 18–20 November 2013; pp. 25–36.
52. OpenLayers: Free Maps for the Web. Available online: http://www.openlayers.org/ (accessed on 10 May 2014).
53. Esri. ArcGIS API for JavaScript. Available online: https://developers.arcgis.com/javascript/ (accessed on 10 September 2015).
54. Leaflet—A JavaScript Library for Interactive Maps. Available online: http://leafletjs.com/ (accessed on 10 September 2015).
55. v2.2.3 JavaScript Library: All|Mapbox. Available online: https://www.mapbox.\unhbox\voidb@x\hbox{com/mapbox.js/api/v2.2.3/} (accessed on 10 September 2015).
56. Negrino, T.; Smith, D. Getting acquainted with JavaScript. In *JavaScript and Ajax for the Web: Visual QuickStart Guide*, 1st ed.; Peachpit Press: Berkeley, CA, USA, 2008; pp. 2–26.
57. Schrader-Patton, C.; Ager, A.; Bunzel, K. Geobrowser deployment in the USDA forest service: A case study. In Proceedings of the 1st International Conference and Exhibition on Computing for Geospatial Research & Application, Washington, DC, USA, 21–23 June 2010.
58. Parr, D.A.; Scholz, M. Building a low-cost geographic website for collecting citizen science contributions. *Pap. Appl. Geogr.* **2015**, *1*, 205–211. [CrossRef]
59. GeoNetwork—The Portal to Spatial Data and Information. Available online: http://www.fao.org/geonetwork/srv/en/main.home (accessed on 10 September 2015).
60. Jurassic—A Javascript Compiler for .NET—Home. Available online: http://jurassic.codeplex.com/ (accessed on 10 September 2015).
61. Loveland, T.; Reed, B.; Brown, J.; Ohlen, D.; Zhu, Z.; Yang, L.; Merchant, J. Development of a global land cover characteristics database and IGBP DISCover from 1 km AVHRR data. *Int. J. Remote Sens.* **2000**, *21*, 1303–1330. [CrossRef]
62. Hansen, M.; DeFries, R.; Townshend, J.R.; Sohlberg, R. Global land cover classification at 1 km spatial resolution using a classification tree approach. *Int. J. Remote Sens.* **2000**, *21*, 1331–1364. [CrossRef]
63. Bartholomé, E.; Belward, A. GLC2000: A new approach to global land cover mapping from Earth observation data. *Int. J. Remote Sens.* **2005**, *26*, 1959–1977. [CrossRef]
64. Bicheron, P.; Defourny, P.; Brockmann, C.; Schouten, L.; Vancutsem, C.; Huc, M.; Bontemps, S.; Leroy, M.; Achard, F.; Herold, M. Globcover: Products Description and Validation Report. Available online: http://due.esrin.esa.int/files/GLOBCOVER_Products_Description_Validation_Report_I2.1.pdf (accessed on 10 September 2015).
65. Data Center for Resources and Environmental Sciences, Chinese Academy of Sciences. Available online: http://www.resdc.cn/data.aspx?DATAID=99 (accessed on 10 May 2015).
66. Vogelmann, J.E.; Howard, S.M.; Yang, L.; Larson, C.R.; Wylie, B.K.; van Driel, N. Completion of the 1990s National Land Cover data set for the conterminous United States from Landsat Thematic Mapper data and ancillary data sources. *Photogramm. Eng. Remote Sens.* **2001**, *67*, 650–662.
67. Wulder, M.; Nelson, T. EOSD Land Cover Classification Legend Report. Available online: http://198.103.48.10/index.html/pub/geobase/official/lcc2000v_csc2000v/doc/EOSD_legend_report-v2.pdf (accessed on 10 May 2015).
68. Lymburner, L.; Tan, P.; Mueller, N.; Thackway, R.; Lewis, A.; Thankappan, M.; Randall, L.; Islam, A.; Senarath, U. The National Dynamic Land Cover Dataset Geoscience Australia Record 2011/31. Available online: http://data.daff.gov.au/data/warehouse/dlcdd9abll078/dlccd9abll0781011a/DLCDv1Overview_1.0.0.pdf (accessed on 10 May 2015).
69. A Search Engine for Land Cover Web Map Services. Available online: http://www.geookay.com:5588/ (accessed on 26 June 2016).

70. Xin, Q.; Min, S.; Chen, X.; Jing, L.; Kai, L.; Jizhe, X.; Qunying, H.; Chaowei, Y.; Bambacus, M.; Yan, X.; et al. A spatial web service client based on Microsoft Bing Maps. In Proceedings of the 2011 19th International Conference on Geoinformatics, Piscataway, NJ, USA, 24–26 June 2011.
71. NuGet Gallery | Lucene.Net 3.0.3. Available online: https://www.nuget.org/packages/Lucene.Net/ (accessed on 10 September 2015).
72. NuGet Gallery | Lucene.Net Contrib Spatial.NTS 3.0.3. Available online: https://www.nuget.org/packages/Lucene.Net.Contrib.Spatial.NTS (accessed on 10 September 2015).
73. Purves, R.S.; Clough, P.; Jones, C.B.; Arampatzis, A.; Bucher, B.; Finch, D.; Fu, G.; Joho, H.; Syed, A.K.; Vaid, S. The design and implementation of SPIRIT: A spatially aware search engine for information retrieval on the Internet. *Int. J. Geogr. Inf. Sci.* **2007**, *21*, 717–745. [CrossRef]
74. Duffy, T.; Tellez-Arenas, A. OneGeology Web Services and Portal as a Global Geological SDI-Latest Standards and Technology. Available online: http://meetingorganizer.copernicus.org/EGU2014/EGU2014--6603.pdf (accessed on 10 September 2015).
75. Dubois, G.; Schulz, M.; Skøien, J.; Bastin, L.; Peedell, S. eHabitat, a multi-purpose Web Processing Service for ecological modeling. *Environ. Model. Softw.* **2013**, *41*, 123–133. [CrossRef]
76. Robinson, T.P.; Franceschini, G.; Wint, W. The Food and Agriculture Organization's gridded livestock of the world. *Vet. Ital.* **2007**, *43*, 745–751. [PubMed]
77. Wood, B.A. A New Zealand case study. *OSGeo J.* **2014**, *13*, 2–12.
78. Porta, J.; Parapar, J.; García, P.; Fernández, G.; Touriño, J.; Doallo, R.; Ónega, F.; Santé, I.; Díaz, P.; Miranda, D.; et al. Web-GIS tool for the management of rural land markets. *Earth Sci. Inf.* **2013**, *6*, 209–226. [CrossRef]
79. Fenoy, G.; Bozon, N.; Raghavan, V. ZOO-Project: The open WPS platform. *Appl. Geomat.* **2013**, *5*, 19–24. [CrossRef]
80. Search for a Dataset—Data.gov. Available online: http://catalog.data.gov/dataset?q=WMS&res_format=WMS&sort=score+desc%2C+name+asc (accessed on 31 October 2015).
81. Datasets. Available online: https://data.gov.uk/data/search?q=WMS&res_format=WMS (accessed on 31 October 2015).
82. University of Zagreb Computing Centre. Products | Borederless Geospatial Web. Available online: http://bolegweb.geof.unizg.hr/site8/products (accessed on 27 April 2016).