A Spatial Web Crawler for Discovering Geo-servers and Semantic Referencing with Spatial Features

Sonal Patil, Shrutilipi Bhattacharjee, and Soumya K. Ghosh

School of Information Technology,
Indian Institute of Technology Kharagpur, India
{sonalspatil,shrutilipi.2007}@gmail.com, skg@iitkgp.ac.in

Abstract. Improvement of technologies in the field of spatial data collection provide a lot of research opportunities in the field of *Geographic Information System*. The geospatial data are often dynamic in nature and available in heterogeneous format. The online spatial data sources are one of the key avenues for publishing and retrieving the geo-spatial data. Efficient discovery of these data sources through Internet, retrieval and analysis of useful information, is one of major challenges in this field. The paper proposes a framework for discovering the geo-spatial data sources using a spatial web crawler. This will facilitate processing of spatial queries involving distributed heterogeneous data repositories. This is being done with the help of the *Web Feature Service* (WFS) standard specification provided by *Open Geospatial Consortium* (OGC). Geo-spatial information is retrieved further for semantic annotations of the data sources using ontology. The semantic information is stored in the form of feature_type repositories as the area of interest lies around the geographic features provided by the geo-servers. The performance study analyzes the accuracy of discovery and semantic annotation of geo-servers for better understanding the framework.

Keywords: Geospatial Data, Data discovery, Spatial Web Crawler, Semantic Indexing, Ontology.

1 Introduction

The exponential growth of geo-spatial data offers enormous research scope to deliver accurate and efficient approaches for discovering and analyzing the spatial information through Internet. In 2005, article [1] reported that *NASA*'s *Earth Observing System Data and Information System* produced over 3 terabytes of data daily. This kind of gigantic amount of data limits the efficient retrieval of suitable geo-spatial data source and getting meaningful knowledge out of it. These limitations also affect the search related spatial attributes, stored as the thematic layers in *geographic information system* (*GIS*) [2].

A web crawler is an automated program or script to retrieve resources from Internet. It is provided with the input set of URLs. From the downloaded pages,

R. Natarajan (Ed.): ICDCIT 2014, LNCS 8337, pp. 68–78, 2014.

all the hyperlinks are extracted from the crawled pages and added to the queue. The process continues till the stopping criterion is met. It also indexes the pages which is used in search engines. Topical crawler is the web crawler which aims to search for a particular topic. The proposed work focusses on developing an efficient topical web crawler for discovering geo-spatial data sources in the Internet and extracting the meaningful information in the form of *feature_types* supported by those geo-servers. It is followed by the semantic indexing of data sources with respect to the stored *feature_types*. This work utilizes the *Web Feature Service* (*WFS*) [3] to get the metadata information related to geographic features provided by any geo-servers. *Open Geospatial Consortium* (*OGC*)[1] has established the *OpenGIS WFS* implementation specification so that the spatial data providers can use them to publish and retrieve their data on Internet. This work is further extended to distinguish spatial data sources as per their offered *feature_types* for efficient indexing and hence better search. *Ontology* [4] [5] is the tool to semantically describe the knowledge base, represent it formally and distinguish between its unique concepts. *Spatial feature ontology* is constructed for annotating geo-server with the proper *feature_types* reference. The *ontology*, used for the experiments, is being populated with the spatial features in the Indian context (refer to Fig. 3) but the coverage of the crawler is global. The results are found to be well balanced between *precision* and *recall* in terms of information retrieval. The overall objectives of this work are as follows,

– Building a spatial web crawler using *WFS* based on *OGC* standard.
– Building a domain *ontology* with spatial *feature_type*.
– Semantic matching using *ontology* and indexing of geo-servers with offered *feature_type* reference.
– Performing experiment with test seed URLs and analyzing the performance of the crawler in terms of accurate semantic annotations.

State of the art reports many research works regarding spatial web crawler. Developing topical web crawler is one of the utmost interesting areas to the researchers in the last few decade. Karkaletsis *et al.* [6] proposed a technique for identifying domain specific web sites and extracting interesting information from the associated web pages. Lopez *et al* [7] measured the performance of the public search APIs. For this, they have tested three main commercial search engines for discovering geographic web services, namely, Bing, Google and Yahoo!. Mukhopadhyay *et al.* [8] proposed a domain specific ontology based search engine for crawling and download the domain specific web pages in WWW. With reference to these works, some topical crawlers has been proposed for spatial web service discovery. Li *et al.* [9] have proposed an effective crawler to discover and update the services using *web map service* by *OGC* utilizing concurrent multi-threading technique. Jiang *et al.* [10] proposed a prototype system of WFS crawler based on the OpenGIS WFS Specification which can discover and update the service content of the WFS servers dynamically. A location-based search engine has been proposed by Ahlers *et al.* [11] which is capable to

[1] `www.opengeospatial.org`

derive spatial context from the unstructured web resources automatically. Their proposed indexer also assigns geo-context to the web-pages for further use.

The paper is organized into four sections. Section 1 gives the overall description of the problem and the proposed solution, followed by the present state of the art regarding the spatial web crawlers and discovery of geo-servers. In section 2, the proposed work is demonstrated with different modules. Performance evaluation is being carried out in the section 3 followed by analysis. Finally, the conclusion is drawn in the section 4.

2 Proposed Framework: WFS Crawler

The proposed spatial web crawler has been developed to focus on geo-spatial features available in the geo-servers. The overall architecture is discussed in this section followed by the implementation specifications and evaluation.

2.1 Crawler Architecture

The crawler developed for discovering data sources based on *WFS* includes *seed set*, *URLQueue*, *extraction module*, *WFS module*, *XML analyzer*, *ontology* and *WFS feature_type repository* to archive relevant WFS geo-servers and related *feature_types*. The proposed architecture of the crawler is illustrated in Fig. 1 with its components.

The initial seed set is the entry section for the crawler. First, all these URLs will be added to the *URLQueue* including the URLs extracted by the crawler. *URLQueue* acts as a buffer to store all the hyperlinks. Once the URL is evaluated to be WFS server, it is not neccessary that it will be linked to the other WFS server. Hence, the next URL from the *URLQueue* will be considered. This process is continuous. The proposed spatial crawler is divided into three main modules i.e. *extraction module*, *WFS module* followed by *analysis and indexing module*. These modules are discussed below.

Extraction Module: This module starts with the URL extracted from the *URLQueue* and reading its page source. Its job is to extract all the hyperlinks present in the page, convert them to absolute URLs, filter the duplicate URLs and then push them to the *URLQueue*. Duplicate URLs are filtered from the *URLQueue*. After completion of cycle of the URL through all the modules, the next URL will be extracted. This process will be continuous.

WFS Module: This module is responsible to check whether server is a WFS server. Extracted URL from *URLQueue* is checked to search for the keywords such as *GetCapability*, *request*, *WFS* etc. If the URL contains all these keywords, it is directly sent to *WFS module* otherwise the *GetCapabalities* request is generated for the extracted URL by appending URL with string
`"services?REQUEST=GetCapabilities&version=1.1.0&service=WFS"`

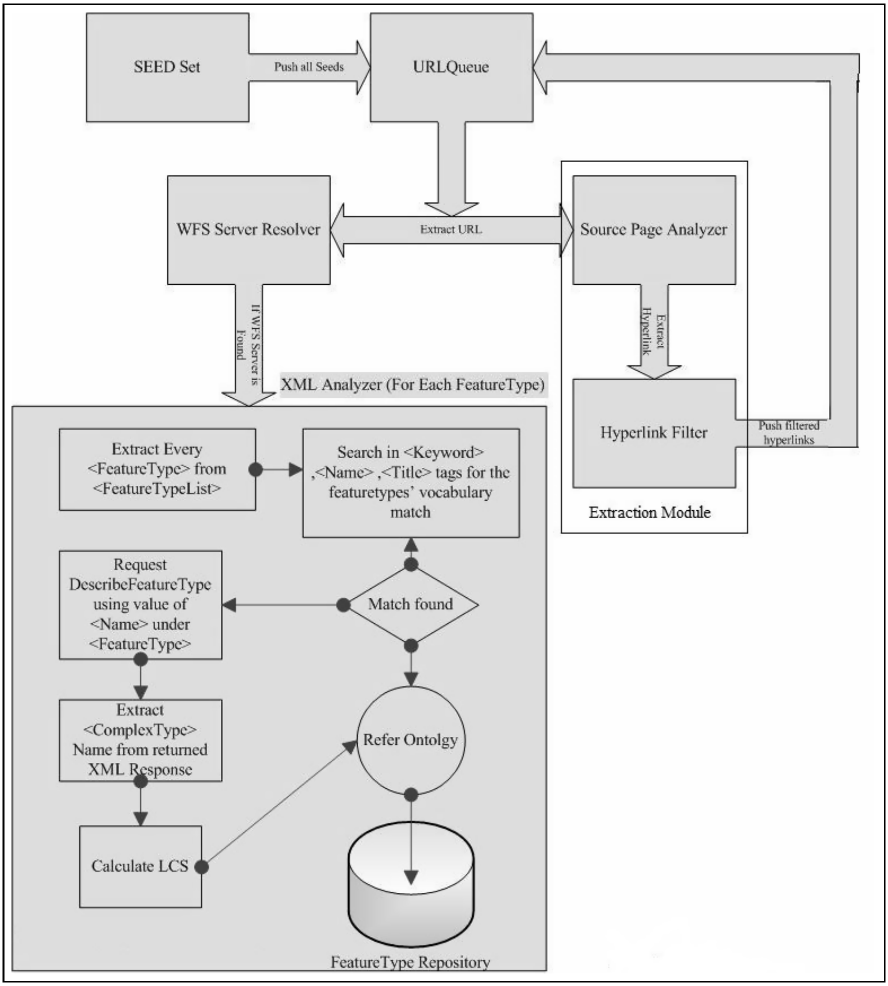

Fig. 1. Proposed Framework for Spatial Web Crawler

and the transformed URL will be sent to *WFS module*. The response of the *GetCapabalities* request is searched to find tags <WFS_Capabilities>. If the tags are present in the XML response, then the URL represents the WFS server. The XML response is sent to the *analysis and indexing module.*

Analysis and Indexing Module: This module analyses the XML response sent by the *WFS module*. There are various tags present in the XML. The <FeatureTypeList> is extracted from the XML and for each of the <FeatureType> tag (under <FeatureTypeList> tag), the <keyword> tag (if present), the <title> and <name> tags are checked to see if any of them contains the words from any of the feature_type's vocabulary which is created for *ontology.* This *ontology* is built by organizing the spatial features, namely, *water-body, building, forest, road* etc. Some standard semantic relations (*such as hyponym, meronym* etc) can be used for building *ontology.* The basic format of <FeatureTypeList> is shown in Fig. 2. If the match is found with any of the *feature_type*, the geo-server reference will be placed into the repository corresponding to that particular *feature_type* as per the semantic matching using *ontology* e.g. if the keyword has word "canal", since "canal" comes under "water-body"(superclass of canal in the *ontology*), corresponding geo-server will be referred under the repository "water-body". If the direct match is not found with any of the keywords, using the <Name> node value of the tag <FeatureType> and "namespace" (if mentioned), *DescribeFeatureType* request is formed by appending string `"?service=WFS&version=1.1.0&request=DescribeFeatureType&typename="+name"` to the original URL. The XML returned by that request is analyzed to know the *feature_type* of that feature using "name" attribute of the <ComplexType> tag. That value is compared with each of the feature_type's vocabulary list and using Longest Common Subsequence (LCS) algorithm [12], maximum length subsequence is determined. This LCS will return the value of the matched *feature_type.* The geo-server reference will be placed into the repository corresponding to that particular *feature_type* as per the semantic matching using *ontology.* This is needed to be checked for each of the <FeatureType> of the <FeatureTypeList>.

2.2 Spatial Crawler Algorithm Description

Algorithm 1 gives the crawler's main algorithm. Initially the *seed set* is taken to be a file having number of seed URLs. *URLQueue* is a *FIFO Queue.* The file is read at start of procedure and each of the seed URLs is pushed to the *URLQueue.* This algorithm consists of two main functions. One of them is with reference to the *extraction module* which is extracting all the hyperlinks and the other refers to the next *WFS Module* to recognize whether web server is WFS geo-server or not. Function *CrawlURL* is crawling the web page and transforming the extracted URLs to the absolute URLs using function *TransformIntoAbsoluteURL*, followed by filtering of duplicate URLs which are already present in the repository, *URLQueue.* Before passing URL to *CheckWFSServer* function, the URL is checked whether it is in *GetCapabilities* Request form or not. If so, we are passing it with the TRUE indicator, otherwise FALSE indicator. In *CheckWFSServer* function, if the indicator is FALSE, the corresponding XML is retrieved by making *GetCapabilities* Request. The XML response is checked to verify whether the server is WFS geo-server by checking the presence of <WFS_Capability> tags. If the server is found to be a WFS geo-server, the XML response is sent to the next module for analysis. *Ontology* is used in the *analysis and indexing module.* It is implemented using *HashMap* which is mapping featuretypes to their parent featuretype. *HashSet* is used to store the vocabulary of all the featuretypes which are taken for this study. *HashSet* and *HashMap* are used to make the searching

```
<WFS_Capabilities>
...
  <FeatureTypeList>
  <Operations>
    <!--operations supported by all FeatureTypes-->
  </Operations>
  <FeatureType>
  <!--information about this FeatureType-->
                    <Name>FeatureType Name</Name>
                    <Title>FeatureType Title</Title>
                    <Abstract>Short Description</Abstract>
                    <Keywords>Keywords</Keywords>
                     ...
                    <Operations>
                    <--operations supported for this FeatureType-->
                    </Operations>
          </FeatureType>
           ...
           ...
           ...
          </FeatureTypeList>
...
</WFS_Capabilities>
```

Fig. 2. GetCapability Response

process faster. *Analyze* function uses the XML response to retrieve the list of *feature_types* provided by that geo-server. For each of the *feature_type*, the *ontology* is addressed to know the parent *feature_type*. The first phase is the direct matching of the values under tags <keyword>, <name> and <title> with each element in *HashSet FeatureTypeVocab* until it finds the match. If it does not find match, it will go to the next phase which makes *DescribeFeatureType* request to extract the *feature_type*. Longest Common Subsequence (LCS) algorithm is used further to find the LCS with each word from *HashSet FeatureTypeVocab* and the extracted *feature_type* name of the <FeatureType> being considered. One word from *FeatureTypeVocab* is selected which has length above LCS length threshold and has maximum length of LCS among all. This is being implemented by using function *ApplyLCS*. That match will be searched in *HashMap* to find the parent *feature_type*. After finding the parent *feature_type* in either of the phase, the geo-server URL is saved into the corresponding parent *feature_type* repository.

3 Performance Evaluation

The performance of the proposed framework is evaluated on the basis of three metrics namely precision, recall and F1-measure [13]. The efficiency of the crawler is measured by analyzing the relevant number of geo-servers found for particular *feature_type* and total number of expected or existing web servers supporting that corresponding *feature_type*.

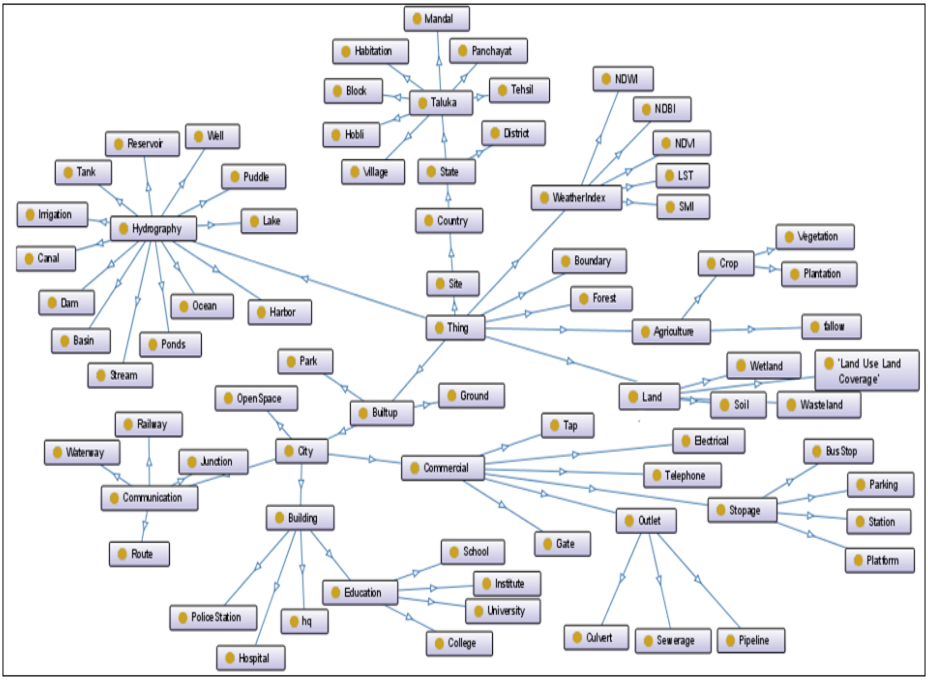

Fig. 3. FeatureType Ontology

For each *feature_type*, these three metrics are calculated, followed by average precision, average recall and average F1-measure for all the feature types. These are calculated based on the semantic annotations of geoservers using different LCS threshold, for the function ApplyLCS in Algorithm 1. *Ontology* displayed in Fig. 3 is used for our study.

The following seed URLs are taken for this case study. Some of the GIS websites are being used as seed to optimize the processing of web pages. Some of the results returned by the general search engines such as Google, for the keywords like "getcapabilities", "wfs", "request", "geoserver" are taken to be the other seeds.

- `https://www.google.co.in/search?q=wfs+getcapabilities+request+geoserver`
- `http://www.bing.com/search?q=wfs+getcapabilities+request+geoserver`
- `http://www.gise.cse.iitb.ac.in`
- `http://bhuvan5.nrsc.gov.in`
- `http://203.110.240.68:8888/iitkgp-wms`

Some of the geoservers discovered by the above seed URLs are referred for the performance evaluation. *GetCapabilities* request for these geoservers are listed below. First four geoservers are considered mainly for the experiments which are having mass data.

Algorithm 1. Spatial Crawler Algorithm

```
Input: SeedSet-Seed URLs, MaxURLs-Stopping Criterion
Result: List of WFS Servers and Indexed Documents
i = 0;
foreach Seed in SeedSet do
    PushQueue(URLQueue,Seed);
    i++;
end
while i ≠ MaxURLs do
    WFSflag = false;
    URL = PopQueue(URLQueue);
    CrawlURL(URL);
    if URL contains "WFS" AND URL contains "request" AND URL contains
    "GetCapabilities" then
        WFSflag = true;
    end
    CheckWFSServer(URL,WFSflag);
    i++;
end

    CrawlURL(URL)
    Input: URL
    Result: Hyperlinks extracted after crawling Webpage
    WebPage = ReadPageSource(URL);
    URLList = ExtractHyperLinks(WebPage);
    foreach U in URLList do
        Trans_URL = TransformIntoAbsoluteURL(U);
        if Trans_URL NOT IN URLQueue then
            PushQueue(URLQueue,U);
        end
    end

    CheckWFSServer(URL, WFSFlag)
    Input: URL, WFSFlag
    Result: Server is WFS geo-server or not AND XMLResponse of GetCapability
            Request
    if WFSFlag ≡ TRUE then
        XMLResponse = ReadXML(URL);
    else
        XMLResponse = SendGetCapabilityRequest(URL);
    end
    if XMLResponse CONTAINS WFS_Capabilities tag then
        Analyze(XMLResponse,URL);
    end

    Analyze(XMLResponse, URL, FeatureTypeVocab, FeatureOntology)
    Input: XMLResponse, URL, FeatureTypeVocab, FeatureOntology
    Result: Indexed Documents by FeatureTypes
    initialization;
    foreach <FeatureType> in <FeatureTypeList> do
        Match = NULL; MatchFlag = FALSE;
        foreach F in FeatureTypeVocab do
            if F MATCHES WITH Value of <Keyword> OR <Title> OR <Name>
            then
                Match = ExtractParentClass(F, FeatureOntology);
                MatchFlag = TRUE;
                Break;
            end
        end
        if MatchFlag ≡ FALSE then
            NameVal = GetNameTagValue(FeatureType);
            XML = GetDescribeFeatureTypeRequest(URL, NameVal);
            TypeNameVal = ExtractTypeName(XML);
            Match = ApplyLCS(TypeNameVal);
        end
        if Match ≠ NULL then
            PutIntoRepository(URL, Match);
        end
    end
```

```
ApplyLCS(FeatureTypeName, FeatureTypeVocab, FeatureOntology)
Input: FeatureTypeName, FeatureTypeVocab, FeatureOntology
Result: Maximum Matching Subsequence from Vocabulary with the
        FeatureTypeName
Match = NULL; MaxLength = 0;
foreach F in FeatureTypeVocab do
    LCS = FindLCS(F, FeatureTypeName);
    if Length(LCS) > MaxLength AND Length(LCS) > LCS_Threshold then
        Match = F;
        MaxLength = Length(LCS);
    end
end
if Match ≠ NULL then
    ParentMatch = ExtractParentClass(Match, FeatureOntology);
    PutIntoRepository(URL, ParentMatch);
end
```

- http://giswebservices.massgis.state.ma.us/geoserver/wms/services?REQUEST=GetCapabilities&version=1.1.0&service=WFS
- http://www.gise.cse.iitb.ac.in/geoserver/wfs/services?REQUEST=GetCapabilities&version=1.1.0&service=WFS
- http://bhuvan5.nrsc.gov.in/bhuvan/ows?service=wfs&version=1.1.0&request=GetCapabilities
- http://203.110.240.68:8888/iitkgp-wms/services?REQUEST=GetCapabilities&version=1.1.0&service=WFS
- http://www2.dmsolutions.ca/cgi-bin/mswfs_gmap?SERVICE=WFS&VERSION=1.0.0&REQUEST=getcapabilities
- http://mapserver.ngdc.noaa.gov/cgi-bin/Sample_Index?request=getcapabilities&service=wfs&version=1.1.0
- http://www.bsc-eoc.org/cgi-bin/bsc_ows.asp?version=1.1.1&service=WFS&request=GetCapabilities

The *feature_types* present with these geo-servers are analyzed using the layer-preview facility present with those WFS services. The relevant *feature_types* that are actually supported by these geo-servers are collected manually for evaluation. Performance is measured based on various LCS threshold. If LCS is found to be above threshold length, it will be considered for further processing. Then the resultant *feature_type* is checked in the *ontology* shown in Fig. 3 and the corresponding geo-server is placed in that corresponding *feature_type's* superclass's repository. The precision model considered for each of the *feature_type* is as follows,

$$precision = \frac{(Number\_of\_relevant\_geoservers\_found)}{(Total\_Number\_of\_geoservers\_found)} * 100\% \tag{1}$$

Average precision is calculated by taking average of precision of all the *feature_types*. It is observed that the precision has increased with increase in LCS length threshold. As the length of LCS is increasing, it tends to be a direct comparison between the two strings. Hence the occurrence of irrelevant results will get decreased. The recall is being calculated as follows,

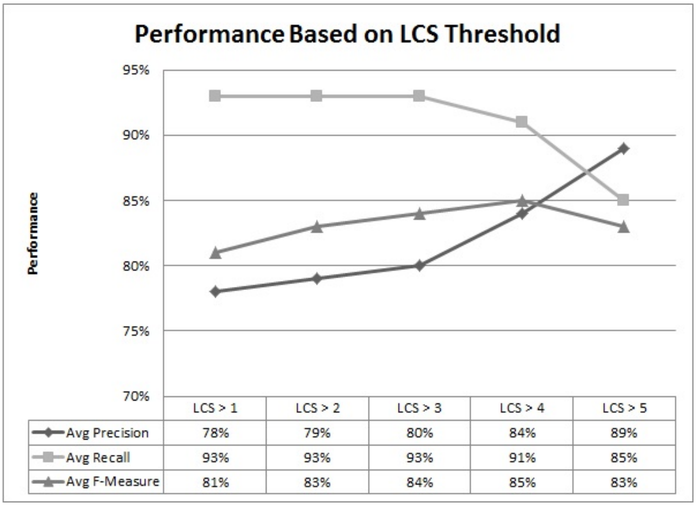

Fig. 4. Performance Evaluation of the Proposed Crawler based on LCS Threshold

$$recall = \frac{Number\_of\_relevant\_geoservers\_found\_in\_search}{Total\_Number\_of\_existing\_relevant\_geoservers} * 100\% \quad (2)$$

Average recall is calculated by taking average of all the *feature_types'* recall. It is observed that the recall is the maximum for the LCS threshold greater than 3. It is increasing from LCS length threshold greater than 1 up-to 3 and then it will start decrementing. With increment of LCS length, chances of finding word matches with vocabulary *FeatureTypeVocab* will get reduced resulting decrement in recall. Based on precision and recall, F1-measure i.e., balanced F-score of precision and recall, is calculated as follows,

$$F1 = 2 * \frac{(precision * recall)}{(precision + recall)} \quad (3)$$

Similarly, average F1-measure is calculated by taking average of F1-measure of all the *feature_types*. As shown in Fig. 4, it is observed that F1-measure is also increasing till LCS length threshold greater than 4 and then it starts decreasing. Since the result is following a trend of decremented recall beyond LCS length threshold greater than 4, the experiment is not carried out further. Again, for most of the featuretype words present in the vocabulary, average length is not more than length 5. Hence, it will increase precision beyond this threshold, but recall will definitely decrease.

4 Conclusion

Geospatial data are often voluminous, dynamic and heterogeneous in nature. The discovery of spatial data repositories over the Internet and information retrieval from these heterogeneous datasets are crucial for resolving spatial queries. This paper presents a spatial web crawler based on WFS (Web Feature Service) specification by OGC, which analyzes the spatial queries and searches for suitable geo-servers for query resolution. It also indexes the geo-servers with respect to relevant feature types to annotate the semantic reference with them. This work can be extended further to extract the location information, geometric properties of spatial features and retrieving the relevant spatial data from the repositories for further processing.

References

1. Aeronautics, U.S.N., Administration, S.: Earth System Science Data Resources: Tapping Into a Wealth of Data, Information, and Services. National Aeronautics and Space Administration (2005)
2. Star, J., Estes, J.: Geographic information systems. Prentice-Hall, Englewood Cliffs (1990)
3. Vretanos, P.: Web feature service implementation specification. Open Geospatial Consortium Specification, 04–094 (2005)
4. Gruber, T.R.: Toward principles for the design of ontologies used for knowledge sharing. International Journal of Human-computer Studies 43(5), 907–928 (1995)
5. Bhattacharjee, S., Prasad, R.R., Dwivedi, A., Dasgupta, A., Ghosh, S.K.: Ontology based framework for semantic resolution of geospatial query. In: 2012 12th International Conference on Intelligent Systems Design and Applications (ISDA), pp. 437–442. IEEE (2012)
6. Stamatakis, K., Karkaletsis, V., Paliouras, G., Horlock, J., Grover, C., Curran, J.R., Dingare, S.: Domain-specific web site identification: the crossmarc focused web crawler. In: Proceedings of the 2nd International Workshop on Web Document Analysis (WDA 2003), Edinburgh, UK (2003)
7. 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 Information Review 35(6), 909–927 (2011)
8. Mukhopadhyay, D., Biswas, A., Sinha, S.: A new approach to design domain specific ontology based web crawler. In: 10th International Conference on Information Technology (ICIT 2007), pp. 289–291. IEEE (2007)
9. 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. International Journal of Geographical Information Science 24(8), 1127–1147 (2010)
10. Jiang, J., Yang, C.J., Ren, Y.C.: A spatial information crawler for opengis wfs. In: Sixth International Conference on Advanced Optical Materials and Devices, International Society for Optics and Photonics, pp. 71432C–71432C (2008)
11. Ahlers, D., Boll, S.: Location-based web search. The Geospatial Web: How Geo-browsers. Social Software and the Web 2.0 are Shaping the Network Society, 55–66 (2007)
12. Kondrak, G.: *N*-gram similarity and distance. In: Consens, M.P., Navarro, G. (eds.) SPIRE 2005. LNCS, vol. 3772, pp. 115–126. Springer, Heidelberg (2005)
13. Makhoul, J., Kubala, F., Schwartz, R., Weischedel, R., et al.: Performance measures for information extraction. In: Proceedings of DARPA Broadcast News Workshop, pp. 249–252 (1999)