A Focused Linked Data Crawler based on HTML Link Analysis

Reihaneh Emamdadi, Mohsen Kahani, Fattane Zarrinkalam
Web Technology Lab., Department of Computer Engineering
Ferdowsi University of Mashhad
Mashhad, Iran
reihaneh.emadadi@stu-mail.um.ac.ir, kahani@mail.um.ac.ir, fattane.zarrinkalam@stu-mail.um.ac.ir

***Abstract*—Linked Data can be published as RDF documents or embedded in HTML documents. A linked data crawler is a program that discovers the published linked data from the web by following RDF links. Note that there are RDF documents that are surrounded by HTML documents. Therefore, linked data crawlers require to follow HTML links in addition to RDF links to be able to discover such RDF documents as well as harvest the embedded linked data in HTML documents. However, many HTML documents have not embedded any linked data and not pointed to any RDF documents. So, crawling such HTML documents decreases discovery rate of RDF documents per unit of network bandwidth and wastes computation resources on non-RDF documents. In this paper, a focused linked data crawler is proposed to address this problem. The proposed crawler analyzes and prioritizes HTML links by calculating the possibility that a link will lead to an RDF document. The experimental evaluation shows that the proposed approach is effective in terms of increasing discovery rate of RDF document in comparison with a non-focused linked data crawler.**

Keywords-linked data crawler; focused crawler; RDF link; HTML link; discovery rate

I. INTRODUCTION

Linked Data is envisioned as an open and global data space for exposing, interlinking and integrating machine-readable and structured data on the web. It is based on dereferenceable HTTP URIs to identify resources (such as web document, real-world entities and abstract concepts) and RDF[1] data model for describing the resources in the form of triples, i.e. *subject*, *predicate*, *object*. By using URIs, linked data allows hyperlinks to be set between different data sources. Such hyperlinks are called RDF links.

In the recent years, one of the main consumers of linked data, are linked data search engines [1-6] that discover thousands of data source and provide query processing capabilities over integrated data. They use a crawler to harvest published linked data from the web by following RDF links. Besides, with the launch of Schema.org[2] on 2011, traditional search engines such as Google have started to crawl the linked data embedded in HTML documents by RDFa[3] and microformats[4] .They use the embedded linked data as background knowledge to enhance information retrieval tasks and provide richer and right search results to users.

[1] http://www.w3.org/RDF/
[2] http://schema.org/
[3] http://www.w3.org/TR/xhtml-rdfa-primer/
[4] http://microformats.org/

For ensuring high discovery rate of RDF documents per unit of network bandwidth, a Linked Data crawler should avoid wasted lookups on non-RDF documents [2,3,7]. On the other hand, these crawlers require to crawl HTML documents, because:

- There are websites that have published linked data, however they are not interlinked with existing websites by RDF links. Consequently, ,the linked data crawler must follow HTML links to access them [1,2].
- As the growing of HTML documents marked up by RDFa format, the linked data crawler may not discover such linked data if it avoids HTML documents [3,4].

It is important to note that, many HTML documents have embedded no linked data, or not pointed to any RDF documents. So, the method to encounter HTML documents in linked data crawling, impacts on the crawler performance.

In this paper for addressing mentioned challenges, a linked data crawler is proposed that fetches HTML documents in addition to RDF documents, while analyzing new extracted HTML links to detect two group of links: 1) HTML links that point to an RDF document, and 2) HTML links that point to an HTML document that contains an RDF link or embedded linked data. Then, the crawler assigns higher priority to such links than other HTML links. This can help to harvest linked data published in both RDF and HTML documents and also improve discovery rate of RDF documents.

Similar to focused crawlers [8-10], the proposed crawler uses some heuristics based on link properties, such as anchor text, path of URI (i.e. a series of directories) and etc. for analyzing HTML links. However, in contrast with the focused crawlers which prioritize the links based on degree of their relevance to particular topics, the proposed crawler is focused on the type of documents.

The rest of the paper is organized as follows. Section II gives an overview of related works. In section III, the mechanism and architecture of the proposed crawler is described. Section IV is dedicated to evaluation of the proposed approach. The conclusion and future work are discussed in section V.

II. RELATED WORK

This section discusses related works in the linked data crawling field, as following aspects: HTML crawling approaches, prioritization strategies, and methods for detecting media-type of web documents.

Current linked data crawlers can be divided into three categories based on HTML crawling approach. In the first category, the crawlers do not crawl any HTML documents. Therefore, these crawlers such as SWSE [3], Falcons [5] and Watson [6] search engine crawlers, are not able to discover linked data embedded in HTML documents, and also RDF documents surrounded by HTML documents.

The second category including Sewer [11] search engine crawler, crawl both RDF and HTML documents and follow RDF and HTML links within them. But when crawling, this crawler visits many HTML documents that have not embedded linked data and not pointed to any RDF documents.

In order to control the efficiency of HTML crawling, the third category of linked data crawlers use bounded HTML crawling approach. In other words, these crawlers crawl both RDF and HTML documents but limit crawling space for HTML documents. For example, Sindice search engine crawler [4] crawls only pinged HTML documents by users and extracts embedded linked data and 'href' links with '.rdf' extension within them. Swoogle search engine [1,7] has implemented a bounded HTML crawler in addition to RDF crawler. The bounded HTML crawler crawls only all URIs relative to the given base URI in a limited search space. Also, WebOWL search engine [2] has employed BioCrawler [12] that crawls web documents in dynamic depth manner. In other words, BioCrawler in its current path, increases the depth of crawling by a fixed value when it discovers an RDF document, and decreases the depth if an HTML document is visited.

Some of the linked data crawlers adopt strategies to prioritize new discovered links for improving discovery rate of RDF documents. These prioritization strategies are defined on PLD[5] or URI.

In the PLD based strategies, each PLD has a dedicated queue of URIs and PLDs are prioritized for picking URI from theirs queue. For example, the SWSE search engine crawler [3] picks an URI from an PLD queue if the percentage of returned RDF documents from the PLD is high. This solution helps to reduce the amount of HTTP lookups wasted on non RDF documents and save the computation resources. But, the disadvantage of this approach is that it prioritizes PLDs based on the past observations. In other words, an PLD may contain an URI leading to many RDF documents, but it may have low priority because high percentage of visited documents are non-RDF document.

The URI based strategies use some heuristics to prioritize new URIs in the queue. These heuristics are in-links count for each URI [3], or high priority for URIs submitted manually by users [4].

On the other hand, RDF documents have particular media-types such as RDF/XML, N-Triples, and etc. All of the linked data crawlers considered by us, rely on file extension of an URI and content-type header filed returned by HTTP response, to detect media-type of a target document. But, Umbrich et al. [13,14] discuss that host name, file name and path of an URI in addition to file extension can help to detect media-type of a target document before devoting network bandwidth for downloading it. They also proposed a media-type focused crawler [14] that uses these heuristics to prioritize URIs based on probability that they are of a requested media-type. They evaluated their work for audio, video and image media-types.

[5] A Pay Level Domain (PLD) is a root domain, e.g. "um.ac.ir".

III. The Proposed Approach

Due to the importance of HTML documents in linked data crawling and presence of HTML documents that have not embedded linked data and not pointed to any RDF documents, we proposed a focused linked data crawler that crawls both RDF and HTML documents. In order to control the efficiency of HTML crawling, our crawler analyzes and prioritizes HTML links based on the possibility that a link will lead to an RDF document. For doing this, the proposed prioritization strategy is defined on URI. The crawler uses link properties such as anchor text, path of URI to determine the link priority. To the best of our knowledge and as mentioned in the previous section, there is no prioritization strategies for crawling linked data that use such link properties. In next sections, the proposed approach is described in details.

A. *Web Documents Classification*

We have classified the web documents into the following four classes based on both their content type and the type of links within them:

- **Class R**: RDF documents in different formats including RDF/XML, N3, Turtle, N-Triples, N-quads.
- **Class HR**: HTML documents that contain at least one link pointing to an RDF document. Such links may appear in the HTML document as: i) URIs indicated by "href" attribute that pointed to an RDF document, and ii) URIs extracted from embedded triples in RDFa format that point to an RDF document.
- **Class H**: HTML documents that contain no link pointing to RDF documents.
- **Class N**: other web documents.

Fig. 1 illustrates two modes of a hypothetical web graph with documents as its nodes and hyperlinks as its edges. The node labels in the left and right graph, indicate content type and class of documents respectively. For example, the top first node is classified into class HR because its content type is HTML and its second link points to an RDF document with OWL format.

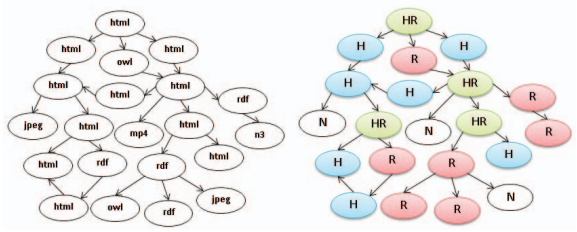

Figure 1. The proposed focused linked data crawler classifies the web documents into four classes (H, HR, R or N) based on both their content type and the type of links within them.

Our main goal is to guide the crawler toward the RDF documents (class R) and the HTML documents pointing to RDF documents (class HR) by analyzing HTML links.

B. The Activity Diagram of the Proposed Crawler

As shown in Fig. 2, the proposed crawler starts crawling with a set of URIs known as seeds. For each URI, the crawler fetches its document from the web. If the content type of document is RDF and its parent is an HTML document, the URI of the parent document is added to S_{HR} set. This set is explained in III.C.3 subsection. If content type of the document is not RDF or HTML, the document is not parsed. Otherwise, the document is parsed and then extracted linked data are stored in a repository for future processing. In next stage, the links within the document are extracted using the following heuristics: i) URIs indicated by hyperlink "href" from HTML parsing, ii) URIs extracted from triples by RDF or RDFa parsing. Then, the extracted links are analyzed, meaning that the crawler predicts the link's target document class before crawling it by, the heuristics introduced in next subsections. The predicted class is R, HR, H or N. The links with the predicted class N is removed because they are unlikely to lead an RDF document. The remained links are prioritize based on their predicted class and added to a priority queue for the next round of crawling.

C. Link Analysis Phase

The link analysis phase is targeted to determine how the links are prioritized in the queue. We adapted the focused crawling approach for analyzing the links.

In general, a focused crawler with a given crawling topic, defines a set of relevant concepts to the topic and a relevancy function. When crawling, the focused crawler estimates degree of relevance of an extracted link or a visited document to the given topic by using defined relevant concepts and relevancy function [8-10].

Instead, the proposed focused linked data crawler estimates degree of relevance of an extracted link to linked data. In other words, the crawler analyzes that whether the target document of an HTML link will belong to class R or class HR. For this, two relevancy functions as $Relevancy_R$ and $Relevancy_{HR}$ are defined.

The algorithm of the link analysis phase is shown as pseudo-code in Fig. 3. Briefly, for each extracted link L, the class of its target document is predicted by using its properties ($\prod_L$) and calling $Relevancy_R$ and $Relevancy_{HR}$ functions. Then, the priority of the link L is determined based on the $predictedClass_L$. The $predictedClass_L$ is R, HR, H or N. The links that are predicted pointing to an RDF document (i.e. $predictedClass_L$=R) are the most important and have highest priority for the next round of crawling. The links are predicted pointing to an HTML document that contains links to RDF documents, are the second important and have second priority. The links with $predictedClass_L$=H is crawled with low priority. The other links have priority equal to "-1" and will be removed. This prioritization strategy can increase the discovery rate of RDF documents if the predicted class of the links are correct. In next sections, the link properties and two relevancy functions $Relevancy_R$ and $Relevancy_{HR}$ for analyzing links, are explained.

1) Link Properties

The proposed crawler uses link properties as parameters for prioritizing the links. The properties of link L is defined as $\prod_L$={$protocol_L$, $host_L$, $path_L$, $filename_L$, $extension_L$, $type_L$, $title_L$, $anchortext_L$, $context_L$}, where:

- $protocol_L$: the transfer protocol indicated in URI of link L
- $host_L$: the host name indicated in URI of link L

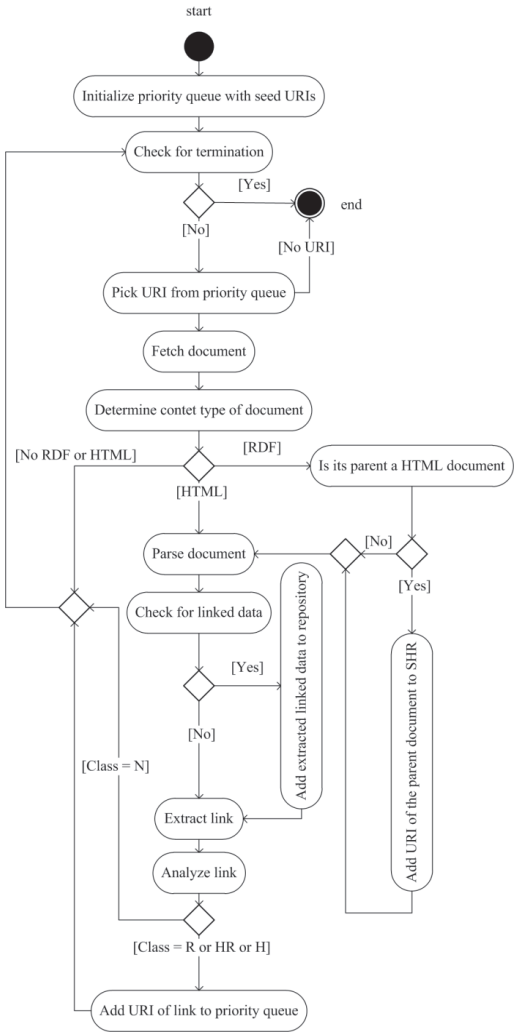

Figure 2. Activity diagram of the proposed crawler

```
1:  For each link L in ExtractedLinksList {
2:      predictedClass_L = Relevancy_R(∏_L,S_R)
3:      if predictedClass_L is H
4:          predictedClass_L =Relevancy_HR(∏_L,S_HR)
5:      if predictedClass_L is R
6:          priority_L = 3
7:      else if predictedClass_L is HR
8:          priority_L = 2
9:      else if predictedClass_L is H
10:         priority_L = 1
11:     else if predictedClass_L is N
12:         priority_L = -1
13: }
```

Figure 3. The Link Analysis Algorithm

- $path_L$: the path indicated in URI of link L. This property is a series of directories separated by character '/'.
- $filename_L$: the file name indicated in URI of link L
- $extension_L$: the file extension indicated in URI of link L
- $type_L$: the type attribute indicated in HTML tag of link L
- $title_L$: the title attribute indicated in HTML tag of link L
- $anchortext_L$: the anchor text of link L
- $context_L$: the surrounding text of link L.

Some of the link properties may be *Null* value. Also $type_L$, $title_L$, $anchortext_L$ and $context_L$ properties are not considered for RDF links because these properties are of specification of HTML format.

2) The $Relevancy_R$ Function

For predicting links pointing to RDF documents (class R) before crawling them, the simplest heuristic is to use the particular file extensions for RDF documents like ".rdf", ".owl" and etc. But sometimes, file extension of URIs has not determined. Also, web documents of different content type may have same file extension. For example, RDF, XML and RSS documents use ".xml" file extension while only RDF documents are targeted for the proposed linked data crawler. So, it needs more heuristics for detecting HTML links with RDF target document. The linked data publishers apply some techniques in their websites to help web crawlers and linked data-aware web browsers to discover RDF data by following HTML links. For example, they may quote the content type of the target document at type attribute, title attribute, anchor text or surrounding text of the link. Also, it is possible to use the particular terms like "rdf", "foaf", "turtle" and etc. as heuristic for detecting RDF documents.

$Relevancy_R$ function is defined as $C = Relevancy_R(\prod_L, S_R)$, where $\prod_L$ is the properties of link L (as mentioned in the previous section), S_R is a set of relevant concepts to the RDF documents and $C \in \{R,H,N\}$ is class of the link L. S_R is union of three sets: the particular content types for RDF documents ($S_{RDFContentTypes}$), the particular file extensions for RDF documents ($S_{RDFFileExtensions}$) and the particular terms for RDF documents ($S_{RDFTerms}$). These sets are constructed in manual and defined as:

$S_R = S_{RDFContentTypes} \cup S_{RDFFileExtensions} \cup S_{RDFTerms}$
$S_{RDFContentTypes}$= { 'application/rdf+xml' , 'text/n3' , … }
$S_{RDFFileExtensions}$ = { '.rdf' , '.owl' , '.nt' , … }
$S_{RdfTerms}$ = { 'rdf' , 'triple' , 'n3' , 'foaf' , … }

$Relevancy_R$ function algorithm contains the following four steps:

1. The link L is predicted as class N (i.e. C=N) if L has non-http protocol, or common file extension that is unlikely to return an RDF document (e.g. ".jpg", ".pdf").
2. The link L is predicted as class R if at least one the following heuristics is satisfied:
   - $type_L$, $title_L$, $path_L$, $filename_L$, $anchortext_L$ or $context_L$ contains at least one of the terms in $S_{RDFTerms}$.
   - $type_L$, $title_L$, $anchortext_L$ or $context_L$ contains at least one of the content types in $S_{RDFContentTypes}$.
   - $extension_L$ is matched to one of the file extensions in $S_{RDFFileExtensions}$.
   - L is an RDF link (i.e. extracted from an RDF document) and $extension_L$ is not determined.
3. If $extension_L$ is a common file extension for HTML documents (e.g. ".html"), or L is HTML link and $extension_L$ is not indicted, Then C=H.
4. Otherwise; C=N.

3) The $Relevancy_{HR}$ Function

After calling $Relevancy_R$ function, the proposed crawler uses the links predicted as class H to identify the HTML links that point to an HTML document that contains at least one link with RDF target document (class HR). In an PLD, URIs of such HTML links are usually similar. This similarity is justified based on behavior of linked data publishers and site managers. For example, suppose an PLD at "http://example.com" contains information about a company's goals, structure, and profiles of staffs. The site manager publishes RDF data for each staff and adds a link to the staff's HTML profile page to aid discovery of data. In home page of this PLD, "staffs" menu points to staffs list. Therefore, these profile pages are considered as class HR based on our classification, and URI of them (such as the following URI_1 and URI_2) are different only in last directory of path property based on the hierarchical structure of the PLD.

URI_1: http://example.com/staffs/staff1
URI_2: http://example.com/staffs/staff2

It is useful to employ this similarity for predicting HTML links as class HR. Let S_{HR} be the set of URIs of HTML documents that point to RDF documents and have been visited by the crawler. Based on the above example, when the crawler visits profile page staff1, URI_1 is added to S_{HR}. Later, if the crawler encounters the link with URI_2, the link is predicted as class HR because URI_2 is similar to URI_1. In general, we introduce a *Similarity Metric* between URIs as follows.

The Similarity Metric. URI_i is similar to URI_j, if URI_i and URI_j are at the same host site but are different in last directory of path property or file name. In other words:

- $host_i = host_j$, $|path_j| = |path_i| = n$, $\forall\ 0 \le k < n-1\ path_i[k] = path_j[k]$, $path_i[n-1] \mathrel{!}= path_j[n-1]$.
- $host_i = host_j$, $path_j = path_i$, $filename_i \neq filename_i$.

where $|path_i|$ is numbers of directories in $path_i$ and $path_i[k]$ is k-th directory in $path_i$ and so on.

Based on this heuristic, the $Relevancy_{HR}$ function is defined as $C = Relevancy_{HR}(\prod_L, S_{HR})$, where $\prod_L$ is the properties of link L, S_{HR} is a set of URIs of HTML documents that contain at least one link with RDF target document and $C \in \{H,HR\}$ is class of the link L. At the

beginning of the crawling process, S_{HR} is empty. As shown in Fig. 2, when an RDF document is visited and its parent is an HTML document, URI of the parent document is added to S_{HR}. So, the proposed crawler learns URIs of the HR class documents.

In the link analysis algorithm (see Fig. 3), after calling $Relevancy_R$ function, for each HTML link L predicted as class H, $Relevancy_{HR}$ function is called and if URI of the link L is similar to one of the URIs in S_{HR}, the link L is predicted as class HR (C=HR) and otherwise, C is remained equal to H.

IV. EXPREIMENTAL EVALUATION

The proposed crawler has been implemented in Java and evaluated through experiments described in next subsections. Briefly, we measured the precision and recall of our crawler and used heuristics for analyzing the links. Also, the discovery rate of RDF documents for the proposed focused Link Data crawler has been compared to a non-focused linked data crawler.

A. Experimental Setup

1) Experiment Environment

The proposed approach should be evaluated on PLDs containing appropriate numbers of both RDF and HTML documents. Up to now, there is no standard PLDs for evaluating the linked data crawlers, Hence, the following four PLDs are chosen manually:

- **dbtune.org:** the published dataset on LOD cloud[6]
- **openlylocal.com:** the published dataset on LOD cloud
- **deri.ie:** the evaluated PLD in [3]
- **rossettiarchive.org:** the evaluated PLD in [7]

The main reasons for choosing these PLDs are: i) including at least 2000 web documents to be fetched, ii) including at least 100 RDF documents, iii) having appropriate speed for response to the crawling request. In our evaluation, crawling process is repeated until 2000 documents from each PLD, are crawled.

2) Evaluation Metrics

Different metrics can be used to measure the efficiency of a focused crawler or a linked data crawler. In this paper, the following metrics are used for evaluating the proposed focused linked data crawler.

Precision. The proposed crawler predicts the class of target document for each link. This prediction may be true or false. The precision of the crawler shows a fraction of crawled web documents with true predicted class and is calculated as $precision_{crawler} = NT_C / N_C$, where N_C is the number of crawled web documents with predicted class C and NT_C is the number of crawled web documents with true predicted class C. Also, for more evaluating the precision of the crawler, the precision of the heuristics introduced in the section III are measured. The precision of heuristic H is calculated as $precision_{Heuristic} = NT_{C,H} / N_{C,H}$, where $N_{C,H}$ is the number of crawled web documents with predicted class C by heuristic H, and $NT_{C,H}$ is the number of crawled web documents that have been predicted correctly as class C by heuristic H.

Recall. In addition to calculate the precision of used heuristics for analyzing the links, the recall of these heuristics is calculate too. The recall of heuristic H is defined as $recall_{Heuristic} = NT_{C,H} / NT_C$, where NT_C is the number of crawled web documents with true predicted class C, and $NT_{C,H}$ is the number of crawled web documents with true predicted class C by heuristic.

Discovery Rate. This metric shows that analyzing and prioritizing HTML links by predicting their class, how much improve the number of discovered RDF documents per unit of network bandwidth. A unit of bandwidth is defined as opening of an HTTP connection by the crawler for visiting a web document. The discovery rate of RDF documents is defined as $discovery\_rate_{crawler} = NT_R / N$, where N is the total number of crawled web documents and NT_R is the number of crawled web documents that have been predicted correctly as class R.

The links are usually organized in the same manner per PLD. For this reason, the mentioned metrics for each PLD are measured separately and then the average is calculated for each metric.

3) Compared Methods

To demonstrate the effect of analyzing HTML links on the performance of a linked data crawler, a non-focused linked data crawler has been implemented for comparison with the proposed crawler. The non-focused linked data crawler crawls both RDF and HTML documents while the new extracted links are not analyzed and added to the queue with the same priority.

B. Experimental Results

The discovery rate of our focused linked data crawler and the non-focused linked data crawler has been compared in Fig. 4. The evaluation shows the proposed approach has improved the discovery rate of RDF documents per unit of network bandwidth.

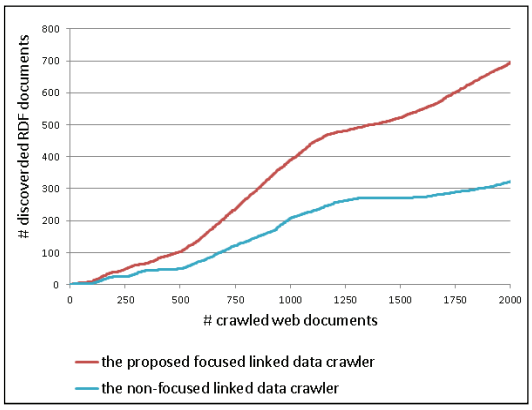

Figure 4. Comparison of the discovery date of RDF documents

TABLE I. shows the average precision of the focused linked data crawler for different PLDs. In order to determine true class of a crawled document based on its content type, we used Apache Tika library[7] (that is a content analysis toolkit). Also, for determining a web

[6] Linked Open Data Cloud: http://lod-cloud.net/

[7] Apache Software Foundation: http://tika.apache.org/

document as class HR, it is necessary that the type of links within the document, has been considered too.

TABLE I. shows that on average, the proposed crawler has the precision above 80% for true predicting the links pointing to RDF documents (class R), or to HTML documents that contain such links (class HR). These true predictions and prioritizing the links based on them, help the crawler to discover more RDF documents (see Fig. 4).

TABLE I. PRECISION OF THE PROPOSED CRAWLER

PLD	$Precision_{crawler}$ for class R	$Precision_{crawler}$ for class HR
dbtune.org	56.96%	90.82%
openlylocal.com	100%	56.57%
deri.ie	75.35%	98.88%
rossettiarchive.org	100%	98.88%
Average	83.08%	86.29%

TABLE II. shows the average precision and recall for each heuristic separately. In this table, the precision value for some heuristics are denoted by "-". This means that these heuristics do not predict anything.

According to TABLE II, all applicable heuristics have precision above 65%. In result, the proposed crawler has high precision for predicting the class of the links.

Among all applicable heuristics, the "$context_L$: $S_{RDFTerms}$" heuristic (i.e. surrounding text of link L contains at least one of the terms in $S_{RDFTerms}$) has lowest precision (66.67%). The reason is that surrounding text of some links contains relevant terms to RDF documents (e.g. 'rdf', 'triple' and etc.), but their target documents are not RDF documents. Note that such links may decrease the precision of the proposed crawler in predicting links as class R. However they are more likely to guide the proposed crawler to RDF documents in next rounds of crawling, compared to other links that do not satisfy this heuristic. Also, there are such conditions for "$anchortext_L$: $S_{RDFTerms}$" and "$path_L$: $S_{RDFTerms}$" heuristics.

On the other hand, the "$path_L$ or $filename_L$: similarity metric" heuristic has high recall (86.34%) for predicting the links as class HR. This shows that URIs of the HTML documents that contain at least one link pointing to an RDF document, have similar path properties per the evaluated PLDs. Consequently, the proposed crawler has predicted most of them correctly.

V. CONCLUSION AND FUTURE WORK

In this paper, we proposed a linked data crawler that utilizes the focused crawling approach to analyze properties of HTML links. For this task, two relevancy functions was defined for predicting the links that are high likely to lead an RDF document. The proposed crawler harvests such links with high priority. Experimental results proved the proposed approach improves the discovery rate of RDF documents. Future works include, doing extensive evaluation on a greater number of crawled web documents and more PLDs, and supporting web documents in different languages using specific language parsers.

TABLE II. PRECISION AND RECALL OF THE HEURISTICS

Heuristic	$Recall_{Heuristic}$	$Precision_{Heuritsic}$
$type_L$: $S_{RDFTerms}$	83.96%	100%
$title_L$: $S_{RDFTerms}$	3.64%	100%
$anchortext_L$: $S_{RDFTerms}$	16%	75%
$context_L$: $S_{RDFTerms}$	15.99%	66.67%
$path_L$: $S_{RDFTerms}$	15.52%	74.42%
$type_L$: $S_{RDFContentTypes}$	83.96%	100%
$title_L$: $S_{RDFContentTypes}$	34.34%	100%
$anchortext_L$: $S_{RDFContentTypes}$	0%	-
$context_L$: $S_{RDFContentTypes}$	0%	-
$extension_L$: $S_{RDFFileExtensions}$	80.89%	99.88%
$path_L$ or $filename_L$:similarity metric	86.34%	72.45%

REFERENCES

[1] L. Ding, T. Finin, A. Joshi, R. Pan, R. S. Cost, Y. Peng, and et al., "Swoogle: a search and metadata engine for the semantic web," In Proceedings of 13th ACM international conference on Information and knowledge management (ACM CIKM'04), ACM New York, NY, pp. 652-659, 2004.

[2] A. Batzios, and P. A. Mitkas, "WebOWL: A Semantic Web search engine development experiment," J. Expert Systems with Applications, vol. 39, pp. 5052–5060, 2012.

[3] A. Hogan, A. Harth, J. Umbrich, S. Kinsella, A. Polleres, and S. Decker, "Searching and Browsing Linked Data with SWSE: the Semantic Web Search Engine," J. Web Semantics, vol. 9, pp. 365-401, 2011.

[4] E. Oren, R. Delbru, M. Catasta, R. Cyganiak, H. Stenzhorn, and G. Tummarello, "Sindice.com: A document -oriented lookup index for open linked data," J. Metadata Semantic and Ontologies, vol. 3, pp. 37-52, 2008.

[5] G. Cheng, and Y. Qu, "Searching Linked Objects with Falcons: Approach, Implementation and Evaluation," J. Semantic Web and Information Systems, vol. 5, pp. 50-71, 2009.

[6] M., Sabou, C. Baldassarre, L. Gridinoc, S. Angeletou, E. Motta, M. d'Aquin, and et al., "WATSON: A Gateway for the Semantic Web," 4th European Semantic Web Conference (ESWC'07), Austria, 2007.

[7] L. Ding, Enhancing Semantic Web Data Access. Ph.D. Thesis, University of Maryland, USA, 2006.

[8] S. Chakrabarti, M. V. D. Berg, and B. Dom, "Focused crawling: a new approach to topic-specific web resource discovery," J. Computer Networks, vol. 31, pp. 1623-1640. 1999.

[9] S. Batsakis, E.G.M. Petrakis, and E. Milios, "Improving the performance of focused web crawlers," J. Data & Knowledge Engineering, vol. 68, pp. 1001-1013, 2009.

[10] D. Hati, and A. Kumar, "UDBFC-An effective focused crawling approach based on URL Distance calculation," 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT'10), pp. 59-63, 2010.

[11] I. Lasek, O. Klimpera, and P. Vojtas, "Scalable Framework for Semantic Web Crawling," 7th Workshop on Intelligent and Knowledge Oriented Technologies, Slovakia, 2012.

[12] A. Batzios, C. Dimou, A. L. Symeonidis, and P. A. Mitkas, "BioCrawler: An intelligent crawler for the Semantic Web," J. Expert Systems with Applications, vol. 35, pp. 524-530, 2007.

[13] J. Umbrich, A. Harth, A. Hogan, and S. Decker, "Four heuristics to guide structured content crawling," 8th IEEE International Conference on Web Engineering (ICWE'08), pp. 196-202, 2008.

[14] J. Umbrich, M. Karnstedt, and A. Harth, "Fast and Scalable Pattern Mining for Media-Type Focused Crawling," Workshop on Knowledge Discovery, Data Mining, and Machine Learning (KDLM'09),2009.