Four Heuristics to Guide Structured Content Crawling

Jürgen Umbrich and Andreas Harth and Aidan Hogan and Stefan Decker

National University of Ireland, Galway

Digital Enterprise Research Institute

{firstname.lastname@deri.org}

Abstract

Search engines focusing on particular media types face difficulties in discovering suitable URIs on the Web. Since the engines are only interested in a small fraction of the Web, a crawler should use heuristics to concentrate on that fraction. To devise such a heuristic, we postulate four hypotheses based on RFCs and W3C recommendations to find cues for certain content types. Tests on a corpus of 22m files (793GB content size) containing 630m URIs show that for the content types text, image, and application, the recommendations are mostly being followed, while results for audio and video are much less consistent. Our findings and recommendations can be implemented as heuristics for efficient discovery of structured content on the Web on top of existing crawlers.

1 Introduction

While established search engines focus on hypertext documents, in recent years a number of specialised search engines have emerged that collect and integrate information from files of particular media types. Seeqpod[1] and Blinkx[2] offer search over audio and video files, Google Scholar[3] and CiteSeer[4] are digital libraries of printable documents, Technorati[5] provides real-time access to news-feeds, and Seekda[6] offers search capabilities for web services. A new generation of search engines with powerful query functionality are emerging such as SWSE[7], Swoogle[8], Watson[9], Fal-

consearch[10] and Sindice[11], which concentrate on semantic web data. Common to all these specialised search engines is that they rely only on documents of specialised media types.

A common issue for targeted search engines is how to discover files of a certain media type on the Web [12]. Most of the search engines provide users with a form or an API to encourage submissions of URIs. In addition they use general Web search engines, like Google, MSN or Yahoo, to harvest URIs using APIs and query constructs like `filetype` or `originurlextension`, but public APIs are restricted by result size and invocation frequency Furthermore, the query constructs only enable searching for a specified file extension instead of querying for the media type of the URIs. Since these methods do not provide enough URIs, all the search engines must invest extra effort traversing the Web in order to find additional sources.

A naïve approach is to use a breadth-first crawler and fetch the connected Web, starting from a seed set of URIs. Search engines in 2005 indexed approximately 11.5 billion documents [8]. In February 2008, the indexed Web was estimated to consist of 45 billion documents[12]. Given that web-scale crawlers can fetch in the order of thousands of pages per second [2], downloading the entire connected Web would take years and require large amounts of CPU power, network bandwidth and storage space. Also, published studies showed that over 90% of the documents on the Web are hypertext files [14] [7]. Therefore, a search engine for a particular media type is focused only on a small subset of the entire Web. Ideally a specialised crawler would download only the relevant portion of the Web and would save time and resources compared to the naïve approach.

Crawling for HTML documents of a particular topic of interest is related to the problem addressed by focused crawling strategies. Focused crawling, as described in [11], [4] or [3], uses decision rules based on content analysis, link structure and anchor text to keep the crawler focused on a

---

[1]http://www.seeqpod.com/

[2]http://www.blinkx.com/

[3]http://scholar.google.com/

[4]http://citeseer.ist.psu.edu/

[5]http://technorati.com/

[6]http://seekda.com/

[7]http://swse.org/

[8]http://swoogle.umbc.edu/

[9]http://watson.kmi.open.ac.uk/WatsonWUI/

[10]http://www.falconsearch.com/

[11]http://sindice.com/

[12]http://www.worldwidewebsize.com/

IEEE computer society

specific topic, such as *"cycling"* or *"HIV"*, to make better use of crawling resources. In contrast, a web crawler for media type targeted search engines is focused on the document formats (such as audio and video) instead of the topic covered by the documents (e.g. "HIV"). The challenge we address is how to develop a theory of focused crawling for particular media types.

Our main idea is to use information encoded in URI's and in the document structure to uncover cues leading to files with certain media types that we can incorporate into the crawling process. Specifically, our contributions are as follows:

- We conjecture four hypotheses that are instrumental in building a crawler focused on gathering files of a particular media type, based on the file extension (**H1**) and path tokens (**H2**) of a URI, the response header field `Content-type` (**H3**) and the position of extracted URIs from HTML documents (**H4**) (Section 2).

- We test the hypotheses, based on a data representative set of 22 million pages with a total file size of 793.36 GB and present as well comprehensive statistics about data size, response code distribution, and the different media types. (Section 3).

- We discuss the result of the hypotheses and give recommendations, based on precision and recall measures, that can improve the discovery of certain media types on the Web.(Section 4).

To summarise our findings: Crawlers targeting on downloading files of a special media type should inspect file extensions and header fields for the content types `text`, `application` and `image`; for the content type s `audio` and `video` crawlers should look at the document tree position. The paper ends with an overview of related work in Section 5 and finally concludes in Section 6.

2 How to Support the Crawling Process

In the following section we present our four hypotheses, based on thoughts and rationale regarding crawling focused on downloading files of a specific media type. The core idea is to take cues from the information available to a crawler during the crawling process such as URI, header, and positional information in the document tree. The goal is to avoid fetching documents of undesirable media types. We explain our thoughts based on an example URI[13] referring to a video from the W3C Video on the Web Workshop in 2007. Our assumption is that by exploiting the identifiers a crawler can target discovery of files of a certain media type.

---

[13]http://www.w3.org/2007/08/video/slides/KidsHealth/Angio.avi

Firstly, we introduce the definition of media types. Media types are registered with IANA[14] and they consist of two parts, the content type and the subtype. Furthermore, most of the registered media types have recommended file extension(s). For our running example the media type is `video/x-msvideo`, with content type `video` and subtype `x-msvideo`. Please note that media types that have a subtype starting with `x-` are not registered with the IANA.

2.1 Crawling Process

A typical web crawler consists of a URI queue, a downloading component, a storage or indexing component and a link extraction component. Figure 1 shows the four basic tasks in the crawling process loop. The basic crawling tasks are: 1) Poll a URI from the queue and 2) establish a connection to the server. Therefore, the client sends a HTTP request and receives a HTTP response. Both messages can contain information about the requested or sent media type. The next crawling task is 3) to download the content of the URI and in the last step 4) to extract new URIs from the downloaded files and add them to the URI queue.

Our focus is to discover content type identifying cues before the file content is downloaded, therefore we omit the crawling task 3) in the the following sections.

2.2 Poll URI

In the first crawling task the available information for a crawler are encoded in the URI. Every network-retrievable document has a Uniform Resource Identifier, called URI, as defined in RFC2396[15]. A URI consists of five components.

$$[PROTOCOL]://[HOST]:[PORT]/[PATH][FILE]$$

The interesting parts, which can contain information about the media type behind the URI, are the *PATH* and *FILE* components: they identify the location of the file on the server. The *PROTOCOL* component specifies only the access protocol for the file and the *HOST* and *PORT* components are the address of the server hosting the file; these components are not dependant on the media type of the content. The file extension is a widely used identifier for the media type behind a URI: web servers often support automatical mappings for static files to `Content-type` response headers using file extensions. Also, applications typically use file extensions to associate files to application programs. Our URI example has the the *FILE* component `Angio.avi` with file extension `avi`, which is mapped to the media type `video/x-msvideo`. The mapped media type matches with the real media type of our example. Consequently, this leads us to our first hypothesis:

---

[14]http://www.iana.org/assignments/mime-types/
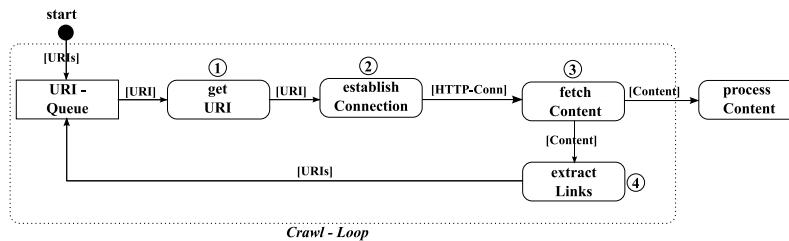[15]http://www.ietf.org/rfc/rfc2396.txt

Figure 1. Basic tasks in the crawling process pipeline. Adapted from [15]

H1: The file extension of the *FILE* part of a URI identifies the media type of the file.

The other interesting component of a URI is the *PATH* component, which can be part of a hierarchical folder structure. Users organise their content in a folder structure, like storing images in a `image` directory or videos in a `video` subfolder, like in our running example. In this case, special subfolders can be used as an indicator for the media types of the documents in this folder. Thus, this leads us to our second hypothesis:

H2: The *PATH* component of an URI identifies the media type of the file.

2.3 Connecting to the Server

The next task in the crawling process is to establish a connection to the server by sending a request and reading the response. Referring to the HTTP/1.1 specification[16] web servers should transmit the `Content-type` and the `Content-encoding` fields in the response. Sometimes, the mapping from file extension to media type is not configured on the server side: In this case, the server uses the default media type `application/octet-stream` or omits the `Content-type` header. For our example URI, the response contains a `Content-type` header field with the value `video/x-msvideo`, which matches the real media type of the sample URI. Hence, these thoughts leads us to our third hypothesis:

H3: The `Content-type` header field identifies the media type of the file.

2.4 Extract new URIs

It is necessary to extract new URIs from the downloaded documents to feed the crawling process loop. For this study, we extract only links from HTML documents. Omitting the textual content of the documents, the available information for a crawler are the position of HTML elements containing

links to other files. The HTML 4.0 specification[17] identifies numerous HTML elements which authors can use to refer to other documents. Beside using navigational links, HTML documents can contain links to meta information, like references to Cascading Style Sheets (CSS)[18] or RSS feeds in the `<LINK>` element or links to embedded audio or video files, like in `<OBJECT>` and `<PARAM>` elements. Different HTML elements are used to link to files of certain media types; accordingly, this leads us to our fourth hypothesis:

H4: The position of a link in an HTML document identifies the media type of the link target.

Please note, hypotheses **H1** and **H3** rely on RFC specifications; however, to the best of our knowledge, no analysis exists of how many systems adhere to the specifications. In Section 3.2 we present the results of testing the hypotheses.

3 Setup and Evaluation

In this section we describe, firstly, our corpus consisting of 22m URIs, the challenge of extracting links and detecting the real media type of a file once downloaded and, secondly, we describe the evaluation of our four hypotheses and present the results.

3.1 Experimental Setup

We use URIs from the Open Directory Project (ODP)[19] and the English Wikipedia[20] as seed sets. Both sites are human edited and contain collections to external resources and are suitable entry points to relevant files on the Web. We extracted 3.8m external links from ODP[21] and 6.4m external links from Wikipedia[22] in December 2007. The entire corpus contains 22.2M documents, 10.2M derived from the crawler in round one and 12M in round two; these consist

---

[16]http://www.w3.org/Protocols/rfc2616/rfc2616.html

[17]http://www.w3.org/TR/WD-html40-970708/htmlweb.html
[18]http://www.w3.org/Style/CSS/
[19]http://www.dmoz.org/
[20]http://en.wikipedia.org/
[21]http://rdf.dmoz.org/rdf/content.rdf.u8.gz
[22]http://download.wikimedia.org/enwiki/

Metric	Values	
Content Size	793.35 GB	
Mean	44.13 KB	
Variance	2.63 KB	
URIs	22,184,518	(100.00%)
200 Response Code	19,030,988	(85.78%)
404 Response Code	1,280,608	(5.77%)
Other Response Codes	1,872,922	(8.44%)
Verified Media Types	17,230,945	
Extracted Links	637,664,510	
Average Unique Links/Page	33.5	

Table 1. Metrics about the Corpus

only of links from the ODP seed set. Table 1 shows general metrics about the corpus. The top five top level domains are: .com (47.92%), .org (10.78%), .net (5.30%), .uk (4.92%) and .de (4.71%).

Comparing our corpus to previously published bigger data sets (815m documents, in 2002 [14], and 75m URIs, in 2005 [10]) we can find similar metrics and statistics for the response code, media type (Table 2) and the top level domain distribution. Thus, we assume that our corpus is a valid and comparable data set for our hypotheses test. To download the contents of the seed set URIs we use the crawler framework MultiCrawler [9] and extracted all links from hypertext documents to test hypothesis four (**H4**).

We use the latest version of the UNIX tool `file` (Version 4.23) to verify the media type of the downloaded files. This tool is a standard Unix program for determining the media type of data contained in a file. Deriving the type of a file includes three different tests that are performed in the following order: checking filesystem metadata, checking the magic number of a file, and checking various text characteristics. A disadvantage of the Unix tool `file` is that it sometimes pools different media types to one single media type, e.g. all different MS Office formats (Word, Excel, PowerPoint) are listed under the media type `application/msword`. Table 2 lists the top ten media types as determined by `file`.

3.2 Evaluation

In this section we present the results of the test of our four hypotheses. To show a higher level of abstraction, we aggregate the different media types into the content type level and omit the subtypes; this mitigates the shortcomings of the tool `file`. Table 3 lists the eight different content types and their occurrences in our corpus. The detailed results are available online[23]. Please note, detailed

Media Type	Count	Percentage
text/html	14,788,347	85.82%
application/x-gzip	1,088,940	6.32%
text/xml	470,487	2.73%
text/plain	323,217	1.88%
application/pdf	200,058	1.16%
image/jpeg	135,520	0.79%
text/x-c++	72,044	0.42%
image/gif	23,597	0.14%
application/x-empty	23,381	0.14%
text/x-c	16,842	0.10%
Total	17,142,433	99.49%

Table 2. Top ten media types detected by the Unix tool `file`

i	Content Type (c_i)	Count ($|F(c_i)|$)
1	text	15,677,707
2	application	1,365,397
3	image	166,496
4	audio	17,501
5	video	2,749
6	message	1,044
7	model	51
8	multipart	0

Table 3. Number of content types detected by the Unix tool `file`

results for hypothesis two (path tokens) and hypothesis four (link position) are omitted due to space limitations. We use precision and recall measures to quantify the correctness for each of our hypothesis. Table 4 lists the symbols used in verifying the hypotheses.

3.2.1 H1: File Extension

Several steps are involved in verifying the extent to which the file extension of a URI identifies the media type of a file. Firstly, we create a mapping between the file extensions and the media types. We start out with a list of common media types[24] and the corresponding file extensions and synchronise the list with the registered media types at IANA. Secondly, we derive the media type based on file extension, which results in $E(c_i)$. Thirdly, we take the intersection between $F(c_i)$ and $E(c_i)$ which denotes the true positives TP_{c_i}. Table 5 contains the precision and recall values pertaining to H1. For computing precision and recall we use: $TP_{c_i} = F(c_i) \cap E(c_i)$, $FN_{c_i} = F(c_i) \setminus TP_{c_i}$

Sets	Description
C	Content types (see Table 3)
T	HTML element positions
$F(c_i)$	URIs of content type c_i, determined by `file`
$E(c_i)$	URIs of content type c_i, determined by file ext.
$P(p_i)$	URIs containing path token p_i
$H(c_i)$	URIs with `Content-type` header field c_i
$L_{t_i}(c_i)$	URIs from HTML link t_i of content type c_i
TP_{c_i}	URIs of content type c_i classified as c_i,
FP_{c_i}	URIs of content type c_j classified as c_i, $j \neq i$
FN_{c_i}	URIs of content type c_i classified as c_j, $j \neq i$.

Table 4. Symbols and their description to test the hypotheses

c_i	$E(c_i)$	TP_{c_i}	PR_{c_i}	RC_{c_i}
`text`	5.869.786	4.593.962	78.26%	29.30%
`appl.`	290.784	226.695	77.96%	16.60%
`image`	192.326	162.553	84.52%	97.63%
`audio`	28.291	15.967	56.44%	91.23%
`video`	9.858	2.039	20.68%	74.17%
`message`	179	46	25.70%	4.41%
`model`	63	51	80.95%	100.00%
`multi.`	1288	0	0.00%	-

Table 5. Results for H1: File Extension

and $FP_{c_i} = E(c_i) \setminus TP_{c_i}$. We can see in Table 5 that we derive precision values greater than 50% for the content types `text`, `application`, `image`, `model` and `audio`, which indicates that the file extension, if available, gives a reasonably good estimate of the real content type. For the content types `video` and `message`, on the other hand, the file extension is a less reliable indicator for the real content type.

3.2.2 H2: Path Tokens

To check if the *PATH* component of a URI identifies the media type we first split the *PATH* component into single path tokens which results in $P(p_i)$. For computing precision and recall we use: $TP_{c_i}(p_i) = F(c_i) \cap P(p_i)$, $FN_{c_i}(p_i) = F(c_i) \setminus TP_{c_i}(p_i)$ and $FP_{c_i}(p_i) = P(p_i) \setminus TP_{c_i}(p_i)$. Using path tokens to identify content type results generally in high precision values: 29 out of 35 path tokens have a precision value of more than 99%. However, recall is low: the recall for all the path tokens is less than 40%, for the majority of the tokens the recall value is less then 10%. Using the path token to identify content type yields accurate predictions, but, we receive only a small subset of the available URIs.

c_i	$H(c_i)$	TP_{c_i}	PR_{c_i}	RC_{c_i}
`text`	11,897,311	10,903,239	91.64%	69.55%
`appl.`	156,766	131.136	83.65%	9.60%
`image`	102,051	99,159	97.17%	59.56%
`audio`	19,206	9,145	47.62%	52.25%
`video`	5,194	1,866	35.93%	67.88%
`message`	130	41	31.54%	3.93%
`model`	15	14	93.33%	27.45%
`multi.`	82	0	0.00%	-

Table 6. Results for H3: `Content-type` header field

3.2.3 H3: `Content-type` header field

To check if web servers send correct `Content-type` header fields we compare the `Content-type` header field $H(c_i)$ with the real content type $F(c_i)$. For computing precision and recall we use: $TP_{c_i} = F(c_i) \cap H(c_i)$, $FN_{c_i} = F(c_i) \setminus TP_{c_i}$ and $FP_{c_i} = H(c_i) \setminus TP_{c_i}$. Table 6 shows the results of the analysis. The results are similar to H1. Precision values for the content types `text`, `application`, `image`, `model` and `audio` are reasonably high, which means that the header information, if available, gives a reasonably good estimate for the real content type. Please observe that not all HTTP responses included a `Content-type` header, which can be seen from the recall. For the content types `audio`, `video` and `multipart` there are more header fields than the detected real media types, which might be due to the web servers sending incorrect `Content-type` header fields.

3.2.4 H4: Document Tree Position

To prove H4, we extracted over 630m URIs from the corpus. Given our restricted resources, we chose not to download the files and use `file` to determine the real media type. Instead, we used the outcome of H1 and the mapping file to estimate the content type of the linked URIs, which yielded media type estimates for 377m URIs, more than half of the extracted links. To check if the position of a link is an indicator for the media type we mapped the links to their content type, which is $L_T(c_i)$. For computing precision and recall we use: $TP_{c_i}(T) = E(c_i) \cap L_T(c_i)$, $FN_{c_i} = F(c_i) \setminus TP_{c_i}(T)$ and $FP_{c_i} = L_T(C) \setminus TP_{c_i}(T)$.

We observed a correlation between the position of the link in the HTML document and the content type. The relevant HTML element for the content type `application` are the `<LINK>`, `<OBJECT>` and `<PARAM>` elements (precision value over 45%). Files of content type `image` can be extracted from `<IMG>` and `<INPUT>` elements with a precision of more than 99.5%. Documents of content

types `audio` and `video` are linked using the `<OBJECT>` element with a precision of 18% and the `<PARAM>` element with a precision of 5%. The low precision values for video has two possible causes: either the use of H1 skews the results of the file type detection, or JavaScript code obfuscation methods are used at the content provider side to deter crawlers from harvesting video links. Please observe that the highest recall for content type `image` is at the `<IMG>` element, for content type `application` is at the `<LINK>` element, and the rest of the content types show the highest recall in the `<A>` element.

4 Discussion and Recommendation

In this section we discuss the result for our hypotheses and give recommendations for a web crawler attempting to fetch only documents of a certain media type.

The test of hypothesis **H1** shows that the file extension of a URI is only correctly used in less than 85% of the cases, with the highest precision occurring for the content type `image` (84.52%). This implies that more than 15% of the URIs have a wrong file extention, and especially for the content type `video` over 79% of the file extensions are incorrect. For the test of hypothesis **H3** we can observe similar effects. The response header fields `Content-type` only partially match the real determined content types. Again, the content type `image` has the highest precision with 97.17%. Compared to **H1**, the response header `Content-type` fields match more often with the real content type. Furthermore, we can see the path tokens can be used as accurate identifiers for the content types behind a URI, as the precision measure shows in hypothesis test **H2**. Moreover, the recall for the content type `application` is in general less than 10% for all tests, except for links in the HTML elements `html/head/link` (45%), `html/body//object` (49%) and `html/body//param` (71%).

Another observable fact is that for hypothesis tests **H1**, **H3** and **H4** the content types `audio`, `video` and `message` showed the lowest precisions. We gave an explanation for each of the hypothesis tests, but another reasonable explanation could be that the Unix tool `file` might not include the latest video and audio formats. This issue surfaced when we manually verified the content types. For some randomly selected video and audio files from our corpus, the media type returned by the Unix tool `file` were either empty results or the media type `application/octet-stream`. Future work is to check the up-to-dateness of the Unix tool `file` and the dependant libraries.

c_i	Precision (>75%)	Recall(>75%)
`text`	H1, H2 ,H3, H4	H4
`application`	H1, H2, H3	-
`image`	H1, H2, H3, H4	H1, H4
`audio`	H2	H1, H4
`video`	H2	H4
`message`	H2	-
`model`	H1, H3	H1, H2, H4
`multipart`	H2	H2, H4

Table 7. Summary of results for the four hypotheses

4.1 Recommendation

Our recommendations are predicated by the goals of the crawler and dependant on the targeted content type and the available information. The goal of the crawling process is to download only files of a certain content type, or in other words achieving high precision during the crawl. Our recommendations are the following, and based on the assumption that the required precision is higher than 75%:

For the content types `text`, `application`, `image` and `model` we recommend to use, if available, the file extension and the `Content-type` header field as identifiers. Furthermore, the position of the link HTML element is a very good identifier for the content types `text` and `image`. The path tokens can be used as an identifier for all of the eight content types.
Table 7 shows a summary based on the hypotheses of our recommendations.

5 Related Work

To the best of our knowledge, this is the first study about discovering hints identifying media types of files behind URIs.

Most of the papers concerned with web content analysis contain statistics about the distribution of content size, top-k media types, top level domains, and HTTP status codes. The data set presented in [14] consists of 815m documents while [10] analysed a corpus of 75m URIs. [7] characterises the community web of the people in Portugal based on 3.2m pages in 2005. [13] analyses the accessibility of information indexed by search engines and presents estimates such as number of online servers on the Web and the number of files and links per domain. In contrast to presenting general statistics about the data sets used, our work examines identifiers for media types and the correctness of the recommended indicators for media types, such as file exten-

sion or header `Content-type` field. Surveys focusing on content metrics exist for media types `application/xml` or `application/rdf+xml`; e.g. [6] analysed the RDF Web in 2002, [16] published a survey of the OWL ontology landscape in 2006, [5] analysed RDF and OWL documents based on 1.7m sources in 2006, and [1] studied the usage of web services in 2006. Again, these studies do not present information about identifiers that can be used to determine the content type of a given URI.

6   Conclusion

We analysed a representative subset of the Web consisting of 22m files and 630m URIs. Comparing our statistics to previously published analysis (from 2002 [14], from 2005 [7]), we conclude that the Web has not significantly changed over the last six years in terms of response code, top level domain, and media type distribution.

We addressed the problem of discovering URIs of particular content types by postulating and testing four hypotheses. The result of the evaluation establishes that for the content types `text`, `image`, and `application`, the recommendations from RFCs (file extension and header content type) and W3C (link position in HTML) are almost being followed, while results for `audio` and `video` are much less consistent. In addition, single path tokens of a URI give a cue to the content type of the file behind the URI. Furthermore, we gave recommendations to assist a crawler traversing the Web for certain media types. Our findings and recommendations can support web crawler developers in implementing heuristics for efficient discovery of structured content on the Web and can help media type targeted search engines to solve the problem of gathering files of a certain media type from the Web.

Acknowledgements

References

[1] D. Bachlechner, K. Siorpaes, H. Lausen, and D. Fensel. Web service discovery - a reality check. In *Proceedings of the 1st Workshop: SemWiki2006 - From Wiki to Semantics, co-located ESWC*, Budva, Montenegro, June 2006.

[2] S. Chakrabarti. *Mining the Web: Discovering Knowledge from Hypertext Data*. Morgan-Kauffman, 2003.

[3] S. Chakrabarti, M. van den Berg, and B. Dom. Focused crawling: A new approach to topic-specific web resource discovery. *Computer Networks*, 31(11-16):1623–1640, 1999.

[4] M. Diligenti, F. Coetzee, S. Lawrence, C. L. Giles, and M. Gori. Focused crawling using context graphs. In *Proceedings of the 26th International Conference on Very Large Databases (VLDB 2000)*, pages 527–534, 2000.

[5] L. Ding and T. Finin. Characterizing the semantic web on the web. In *Proceedings of the 5th International Semantic Web Conference*, Athens, Georgia, November 2006.

[6] A. Eberhart. Survey of rdf data on the web. Technical report, International University in Germany, 2002.

[7] D. Gomes and M. J. Silva. Characterizing a national community web. *ACM Transactions on Internet Technology*, 5(3):508–531, 2005.

[8] A. Gulli and A. Signorini. The indexable web is more than 11.5 billion pages. In *WWW '05: Posters of the 14th international conference on World Wide Web*, pages 902–903, New York, NY, USA, 2005. ACM.

[9] A. Harth, J. Umbrich, and S. Decker. Multicrawler: A pipelined architecture for crawling and indexing semantic web data. In *International Semantic Web Conference*, pages 258–271, 2006.

[10] A. Heydon and M. Najork. Mercator: A scalable, extensible web crawler. *World Wide Web*, 2(4):219–229, 1978.

[11] M. Jamali, H. Sayyadi, B. B. Hariri, and H. Abolhassani. A method for focused crawling using combination of link structure and content similarity. In *Web Intelligence*, pages 753–756. IEEE Computer Society, 2006.

[12] H. Lausen and T. Haselwanter. Finding web services. In *1st European Semantic Technology Conference*, volume 2007. ESTC, June 2007.

[13] S. Lawrence and C. L. Giles. Accessibility of information on the web. *Intelligence*, 11(1):32–39, 2000.

[14] M. Najork and A. Heydon. High-performance web crawling. In *Handbook of massive data sets*, pages 25–45, Norwell, MA, USA, 2002. Kluwer Academic Publishers.

[15] G. Pant, P. Srinivasan, and F. Menczer. Crawling the web, 2004.

[16] T. D. Wang, B. Parsia, and J. Hendler. A survey of the web ontology landscape. In *Proceedings of the 5th International Semantic Web Conference*, Athens, Georgia, November 2006.