Partnership
The Canadian Journal of Library and Information Practice and Research

vol. 10, no. 1 (2015)

White Hat Search Engine Optimization (SEO): Structured Web Data for Libraries

Dan Scott
Assistant Librarian
Laurentian University
dscott@laurentian.ca

Abstract

“White hat” search engine optimization refers to the practice of publishing web pages that are useful to humans, while enabling search engines and web applications to better understand the structure and content of your website. This article teaches you to add structured data to your website so that search engines can more easily connect patrons to your library locations, hours, and contact information. A web page for a branch of the Greater Sudbury Public Library retrieved in January 2015 is used as the basis for examples that progressively enhance the page with structured data. Finally, some of the advantages structured data enables beyond search engine optimization are explored.

Keywords

search engine optimization; structured data; schema.org; RDFa

Introduction

Search engine optimization (SEO) has acquired an unsavoury reputation through its association with spammers who want to sell you snake oil that will help your website rise in the ranks of search engine relevancy. Web rings, spam comments on blogs, and other underhanded tricks have cast aspersions on the perfectly valid pursuit of trying to make your content and services discoverable by people who might benefit from it. The term hackers was once a term associated with those who were simply interested in how technology works and how to make it work better, until it was spoiled by association with unethical actors who wanted to steal content or make lives miserable. Now, however, the developer community has reclaimed the term by differentiating “white hat” hackers vs. “black hat” hackers. So, too, are web developers reclaiming the term SEO by differentiating "white hat" SEO vs. "black hat" SEO.

In this article, I introduce you to some of the basic properties of a website that you need to look at from a white hat SEO perspective to make it easier for your prospective audience to connect to your library and its resources. This article uses examples drawn from a real library website. I encourage you to compare those examples with your own site as the article progresses.

SEO Prerequisites

Before addressing the addition of "structured data" to websites, there are a few prerequisites for improving website discoverability that you should first consider.

robots.txt

A "robot" is a computer program that crawls through websites to extract content and to process it for some purpose. Search engines like Google and Bing use robots to harvest content from sites so that they can serve up links to your pages, images, and library resources in search results; other robots might be interested only in product information so that they can tell you the cheapest place to buy something online. Even though they are computer programs, however, robots are easily confused.

For example, if a robot finds a search results page on your catalogue that lists thousands of results for the term "fish" based on a search result URL like `http://catalogue.example.com/results?term=fish`, it will dutifully crawl through all of the returned records. Each record may have a URL like `http://catalogue.example.com/record/1?term=fish` that maintains the original search term "fish". If the robot subsequently finds a search results page for the term "fishing" based the search result URL `http://catalogue.example.com/results?term=fishing` that, due to stemming, returns the same records as the term "fish", it will dutifully crawl through all of the records again because the URL `http://catalogue.example.com/record/1?term=fishing` differs.

Humans can discern that the record details page for each result will display the same record, and that (if anything) the list of related search results might change on that page without affecting the substantive content on the page. Robots, in contrast, can find it extremely difficult to predict what content these links will return without human intervention. Providing guidance for robots is one of the roles of a webmaster, and while most search engines offer administrators control through "webmaster tools" interfaces, it is cumbersome to repeat those steps for every search engine of interest. By instead publishing a single `robots.txt` file based on the IETF draft (Koster), you can prevent all search engines from crawling search results links for your websites.

The `robots.txt` file must appear at the root of your website; for example, `http://catalogue.example.com/robots.txt` or `http://example.com/robots.txt`. A very simple `robots.txt` file that blocks crawling of search results in this example looks like:

```
User-agent: *
Disallow: /results/
```

- The `User-agent` line identifies to which robot the subsequent rules apply. The asterisk ("`*`") is a wildcard operator signifying that all robots should pay attention to the rules following this line.

- The `Disallow` line tells the robots not to crawl any URLs that start with `"/results/"`.

It is worthwhile to check your websites. All too often, sometimes due to misunderstandings about how robots work, or sometimes due to software that simply can't handle the traffic that robots generate, a `robots.txt` file says:

```
User-agent: *
Disallow: *
```

That tells every robot to entirely ignore your site. Librarians typically want search engines to expose users to their libraries' resources and services. Denying all content to all robots was a relatively common practice when early search engines requested pages faster than web servers were able to deliver, and occasionally caused web servers to crash or become unresponsive to other users. Modern search engines, however, dynamically adapt their crawl rate to ensure web servers remain responsive, because they want to connect users to the best resources on the web.

One other note: `robots.txt` files only affect robots that adhere to the `robots.txt` standard. Search engines like Google, Bing, and Yandex implement the standard, so if your site is constantly being brought down by too much traffic, setting a highly restrictive policy in a `robots.txt` file is likely to only block good robots while the bad robots continue to overload your site with traffic.

Now that you have told the robots what they should not pay attention to on your site, you need to point them at the pages to which they should pay attention.

Sitemaps

By "sitemaps", this article does not refer to website sitemaps that are meant as an aid to humans trying to understand the structure of the website; those kinds of sitemaps tend to involve hierarchy and are most helpful when website search is not useful. Instead, a sitemap for search engines is simply a list of every URL in your website with content in which a search engine would be interested. In a library catalogue, for example, this would be a list of every publicly accessible record; on your main website, it is quite likely every public-facing page.

The sitemaps format is XML and has not changed significantly since it was introduced by Google in 2005 (Sullivan). Every major search engine has subsequently adopted the format. The simplest possible sitemap repeats `<url><loc>...</loc></url>`, listing every page in a given website:

```
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
   <url>
      <loc>http://example.com/about</loc>
   </url>
   <url>
```

```
      <loc>http://example.com/hours</loc>
   </url>
</urlset>
```

To list more than 50,000 URLs, you should break up the sitemap into separate files of 50,000 URLs each. You can then create a sitemap index file that lists each of the constituent sitemap files:

```
<sitemapindex
xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
   <sitemap>
      <loc>http://example.com/sitemap1.xml</loc>
   </sitemap>
   <sitemap>
      <loc>http://example.com/sitemap2.xml</loc>
   </sitemap>
</sitemapindex>
```

Catalogues at many libraries contain over a million items. Creating a sitemap for all of that content, and keeping it synchronized with all of the corresponding additions, deletions, and changes would be an unmanageable amount of manual labour for humans. Machines, however, can easily generate a new sitemap on a regular basis, and many of the content management systems that websites run on either have the ability built-in, or have plug-ins available, that will generate sitemaps. Library systems such as Evergreen (Evergreen Documentation Interest Group) and VuFind ("Search Engine Optimization") recognize the importance of offering automated support for generating sitemaps.

To tell robots the last time a given page was changed, and thus prevent unnecessary traffic to your website, each <url> element can include a <lastmod> element that specifies the last time that URL was modified. Instead of visiting each page on your website, a robot simply needs to read your sitemap and then crawl only the pages that have changed since the last time it visited your site. This ensures that the robots have access to the freshest content on your site—and the sitemap structure remains reasonably simple:

```
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
   <url>
      <loc>http://example.com/about</loc>
      <lastmod>2015-01-29</lastmod>
   </url>
   <url>
      <loc>http://example.com/hours</loc>
      <lastmod>2015-01-25</lastmod>
   </url>
</urlset>
```

Unlike the `robots.txt` file, there is no canonical URL for a sitemap, which means that search engines need to be told where to find your sitemap. While you can set the location of your sitemap through each search engine's "webmaster tools" interface, you can avoid the redundancy of maintaining this information manually at every search engine by adding a `Sitemap:` line to your `robots.txt` file such as:

```
Sitemap: https://example.com/sitemapindex.xml
```

Most search engines will find and use that the next time they refresh their knowledge of your `robots.txt` file.

Structured Data

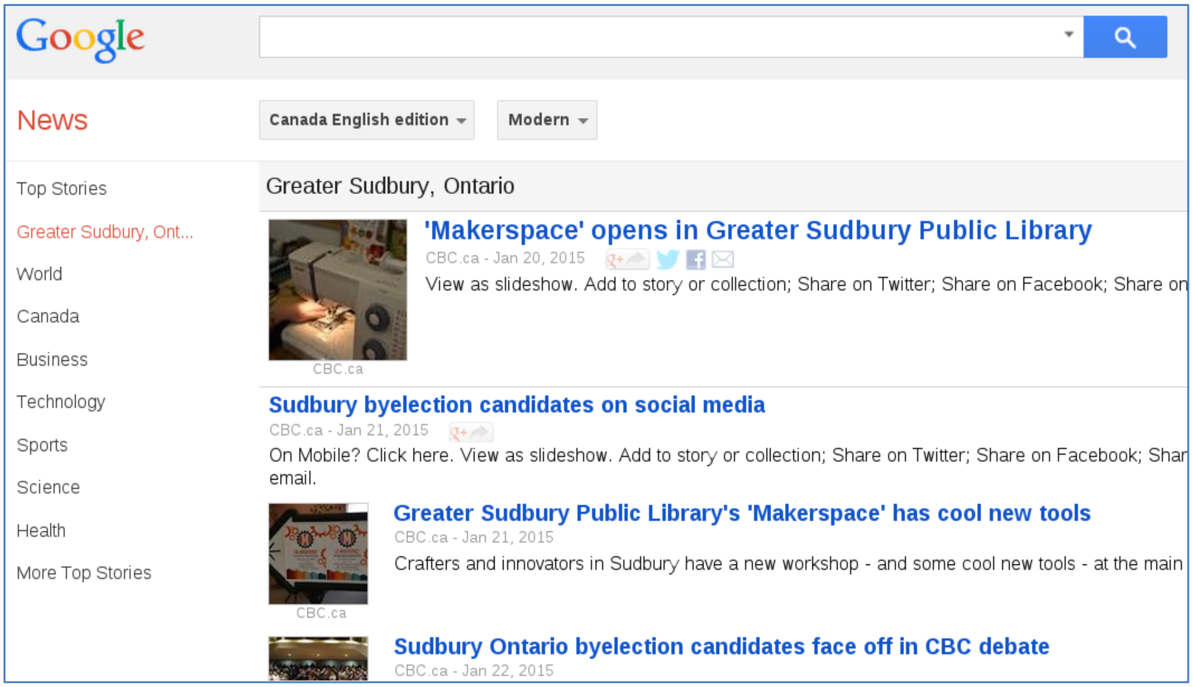
Now that you have deployed a `robots.txt` file to prevent the search engines from wasting their time on pages that have no relevant content, and generated a sitemap to point search engines at the content that they should be spending their time on, you can begin improving the ability of the machines to interpret the content that you are serving up to them. Structured data enables machines to derive information from web pages with more accuracy than heuristics. There is some overlap with core principles of information design and designing your HTML for accessibility, for example:

- Use meaningful HTML elements where possible, such as <`h1`> and <`h2`> heading elements, rather than forcing all content into meaningless <`div`> and <`span`> elements (Gibson and Schwerdtfeger 202). Arrange headings hierarchically: the <`h1`> element should be the title of the page; <`h2`> should be used to break out separate sections of the page into major topics; then <`h3`> should subdivide the content of those major topics. If you feel the need to use a lot of bold style to draw attention to parts of your text, consider restructuring your content instead.

- Don't use an image instead of text. For example, don't create a PNG or JPEG image to use as a button that says "Enter the contest"; even if you add `alt="Enter the contest"`, that's less approachable than simply using "Enter the contest" as the actual text of the <`button`> element.

If your library is located in Ontario and has more than 50 staff members, your website must currently be compliant with Web Content Accessibility Group (WCAG) 2.0 level A ("Make your website accessible"). No matter what jurisdiction your library falls in, however, you should provide your users with an accessible website; Cynthia Ng argues that accessible information design enables your service to be more "findable, accessible, usable, shareable, efficient, and collaborative" ("Making Web Services Accessible With Universal Design").

Even the cleanest, most highly structured HTML does not offer much detailed information to machines: it is still largely a bag of words to them. Machines may be able to interpret the titles of various sections from heading elements, and determine the relative importance of text based on the font size and colour, but it is very challenging for them to decide what picture on a news page is appropriate for associating with the news event that is the focus of the article. Google News, for example, often provides amusing examples of machines guessing incorrectly at the correct picture, headline, or description of what should be relatively uniformly structured news content.



Greater Sudbury, Ontario

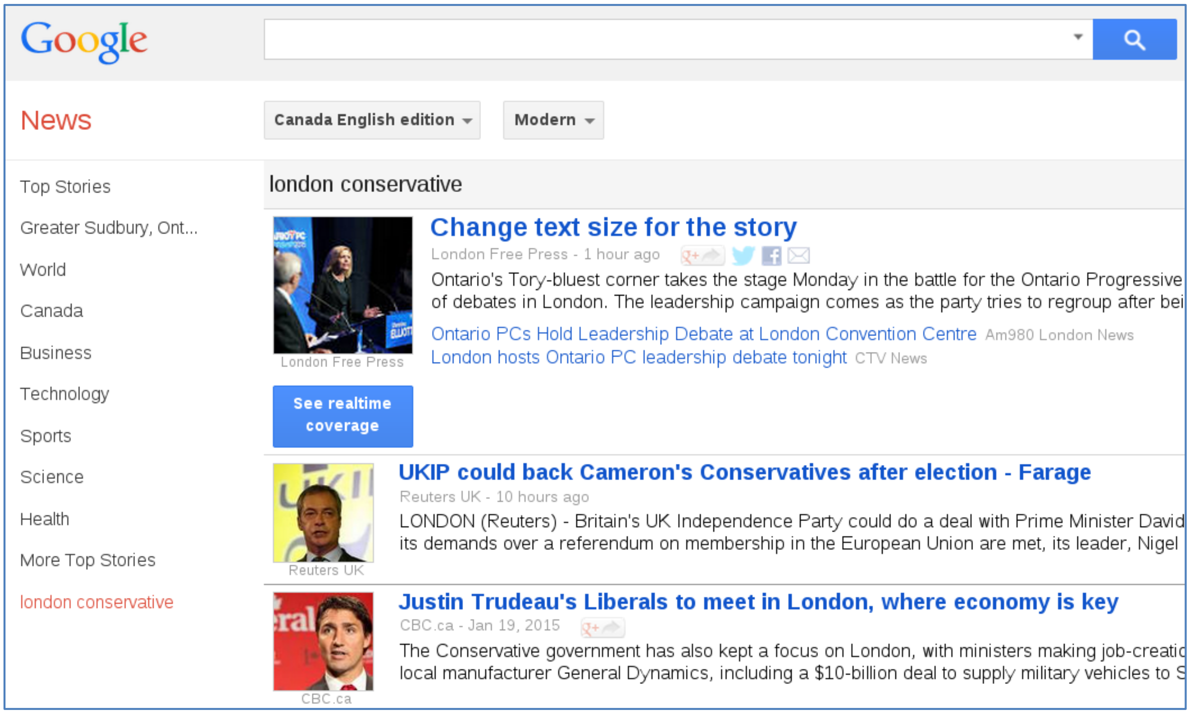
'Makerspace' opens in Greater Sudbury Public Library
CBC.ca - Jan 20, 2015
View as slideshow. Add to story or collection; Share on Twitter; Share on Facebook; Share on

Sudbury byelection candidates on social media
CBC.ca - Jan 21, 2015
On Mobile? Click here. View as slideshow. Add to story or collection; Share on Twitter; Share on Facebook; Shar email.

Greater Sudbury Public Library's 'Makerspace' has cool new tools
CBC.ca - Jan 21, 2015
Crafters and innovators in Sudbury have a new workshop - and some cool new tools - at the main

Sudbury Ontario byelection candidates face off in CBC debate
CBC.ca - Jan 22, 2015

Figure 1. Social media sharing options mistaken for story summary.



london conservative

Change text size for the story
London Free Press - 1 hour ago
Ontario's Tory-bluest corner takes the stage Monday in the battle for the Ontario Progressive of debates in London. The leadership campaign comes as the party tries to regroup after bei
Ontario PCs Hold Leadership Debate at London Convention Centre Am980 London News
London hosts Ontario PC leadership debate tonight CTV News

UKIP could back Cameron's Conservatives after election - Farage
Reuters UK - 10 hours ago
LONDON (Reuters) - Britain's UK Independence Party could do a deal with Prime Minister David its demands over a referendum on membership in the European Union are met, its leader, Nigel

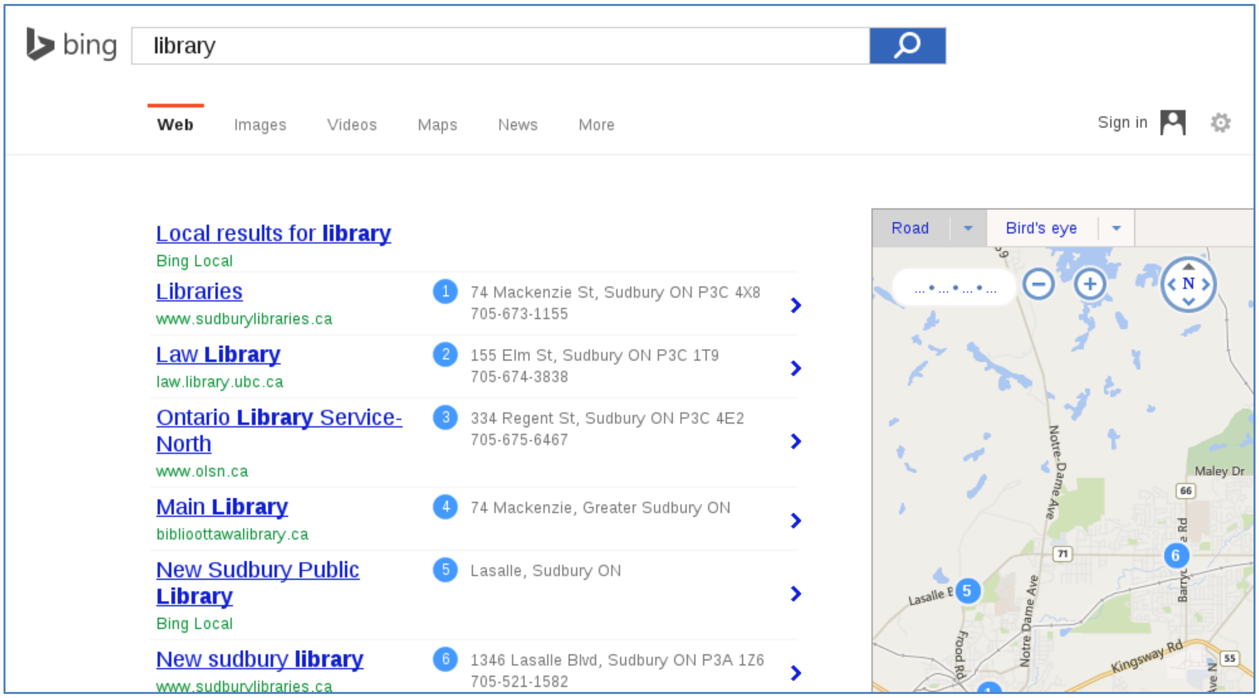
Justin Trudeau's Liberals to meet in London, where economy is key
CBC.ca - Jan 19, 2015
The Conservative government has also kept a focus on London, with ministers making job-creati local manufacturer General Dynamics, including a $10-billion deal to supply military vehicles to S

Figure 2. `Change text size for the story` mistaken for the title of the story.

Library Presence in Search Results

A recurring thread of discussion in the library world is how to make the contents of catalogues show up in search engines so that regular users could be pointed directly at library resources in the course of their general searches. Towards that end, I have done some work with Evergreen, Koha, and VuFind on making those library systems express bibliographic information as linked data in the format that search engines expect (Scott).

If someone in the general vicinity of your library searches for "library" on their phone or laptop, you want your library to show up in the search results. While providing structured data in your library web pages makes that a possibility, a less technically demanding and more common approach has been to manually populate the social media sites that feed the search engines' data sources. For Google, that means setting up a Google+ page for each branch of your library to maintain data such as addresses, hours, contact information, and some photos. Bing, by comparison, relies on third-party social media data such as Foursquare; local experience in early 2015 suggests that it is not particularly trustworthy information.



Local results for library
Bing Local

Libraries
www.sudburylibraries.ca
1 74 Mackenzie St, Sudbury ON P3C 4X8
705-673-1155

Law Library
law.library.ubc.ca
2 155 Elm St, Sudbury ON P3C 1T9
705-674-3838

Ontario Library Service-North
www.olsn.ca
3 334 Regent St, Sudbury ON P3C 4E2
705-675-6467

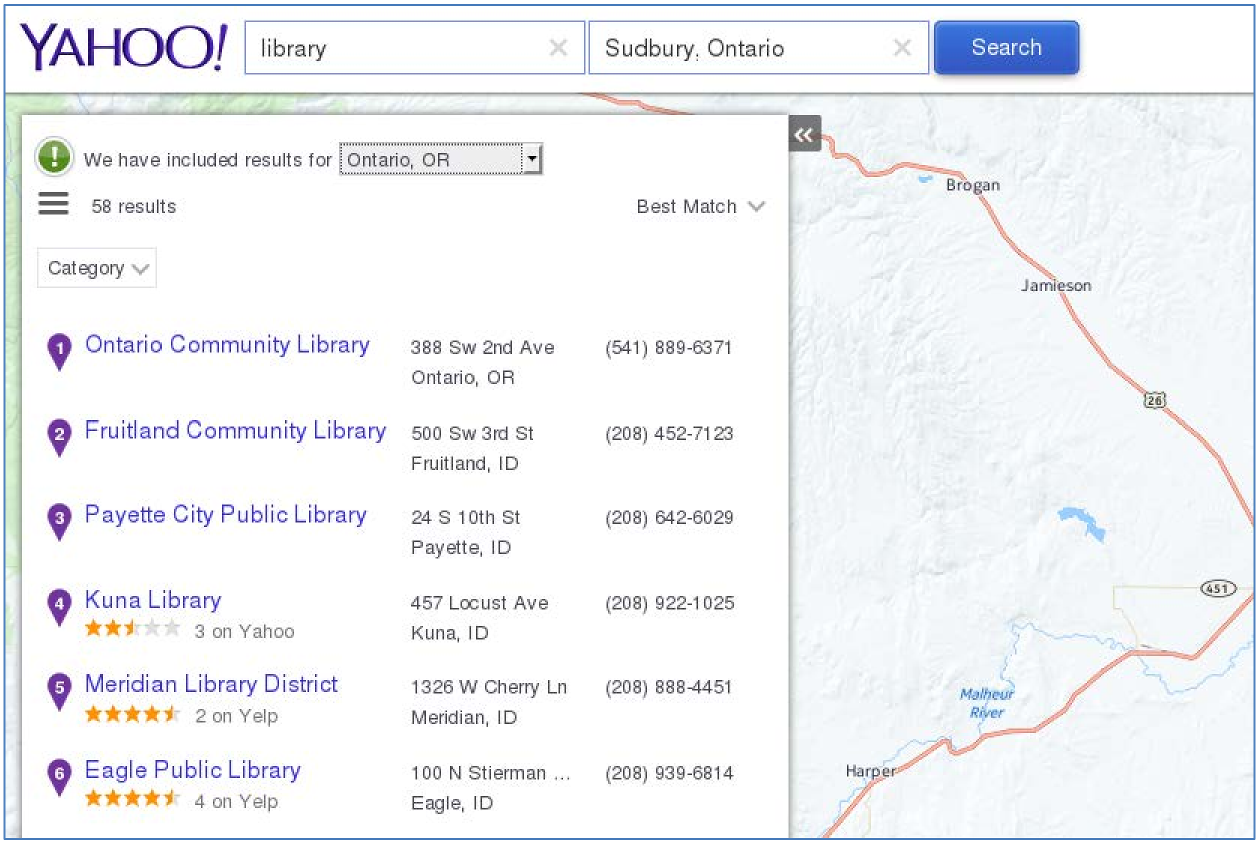
Main Library
biblioottawalibrary.ca
4 74 Mackenzie, Greater Sudbury ON

New Sudbury Public Library
Bing Local
5 Lasalle, Sudbury ON

New sudbury library
www.sudburylibraries.ca
6 1346 Lasalle Blvd, Sudbury ON P3A 1Z6
705-521-1582

Figure 3. Bing "library" search showing some local libraries with incorrect URLs.

Yahoo's local search was even worse in early 2015. Depending on your entry point, in Canada it either directs you to the Canadian Yellow Pages site, where few libraries are found, including none in my community of over 100,000 people; or, when the search is submitted via Firefox's search widget, has no knowledge of Canada at all.



We have included results for Ontario, OR

58 results

1 Ontario Community Library	388 Sw 2nd Ave Ontario, OR	(541) 889-6371
2 Fruitland Community Library	500 Sw 3rd St Fruitland, ID	(208) 452-7123
3 Payette City Public Library	24 S 10th St Payette, ID	(208) 642-6029
4 Kuna Library ★★½☆☆ 3 on Yahoo	457 Locust Ave Kuna, ID	(208) 922-1025
5 Meridian Library District ★★★★½ 2 on Yelp	1326 W Cherry Ln Meridian, ID	(208) 888-4451
6 Eagle Public Library ★★★★½ 4 on Yelp	100 N Stierman ... Eagle, ID	(208) 939-6814

Figure 4. Yahoo local "library" search for Sudbury, Ontario showing closest results as Ontario, Oregon.

Another potential site for finding information about libraries is the OCLC WorldCat Registry, which claims to be the "Authoritative single source for institutional metadata". Unfortunately, the entry for the Greater Sudbury Public Library (GSPL) contains almost no general information—just a name and an address for the main branch—and has no specific information about the 13 branches. The entry for the J.N. Desmarais Library at Laurentian University contains more information because I updated it while writing this article to point at the new catalogue our library had migrated to 5 years ago, but there is still no branch information, and based on the contact information one can discern the conflict between the needs of our ILL department and the general needs of the library. Even metropolitan public library systems like the Toronto Public Library only have a handful of entries, leaving out most of the 99 branches. If a search engine paid for the commercial license needed to use the WorldCat Registry data and populate its local search results for libraries, it would receive an incomplete and non-authoritative dataset.

Maintaining presences across the various social media sites and data partners upon which the search engines rely would help your library appear in local searches. But keeping even relatively static data such as operating hours up to date can be cumbersome across multiple social media sites. Structured data offers a better way forward.

schema.org

Recognizing that the world would benefit from a metadata vocabulary that could be embedded inside web pages to provide more meaning to the machines that digest those web pages, some of the major search engines—Google, Yahoo, Bing, and Yandex—collaborated in 2011 to bring forward such a vocabulary, called “schema.org" (Guha). Using this vocabulary, you can unambiguously identify sections of your library home page that describe your library's name, address, phone number, email address, opening hours, upcoming events, etc. Rather than relying on third parties to provide this data, search engines or other automated processes would, when they come across this kind of markup through their normal web crawling and indexing processes, have much more authoritative data on which to build their services.

A Word about Dublin Core

In the mid-to-late 90's, webmasters commonly used `<meta>` tags incorporating Dublin Core elements like keywords, title, and date in the `<head>` section of web pages to provide machine readable metadata (Wiebel). Unfortunately, to increase the ranking and relevancy of web pages, black hat SEOs used those metadata elements to misrepresent the pages through techniques such as including salacious terms with no connection to the actual content of the web page, and falsifying dates to represent the content as being newer than its actual publication date (Gyongyi and Garcia-Molina). This approach initially worked, but when search engines realized how their results were being manipulated, they began ignoring metadata in `<meta>` tags altogether.

When the search engines introduced schema.org in 2011, they wanted to prevent the manipulation of relevancy and ranking by markup that has no relation to the human readable content of the page. Therefore, schema.org initially required all markup to be inline in text that was visible on the page.

A Word about Open Graph Protocol

Facebook launched the Open Graph Protocol (OGP) standard for integrating structured data in web pages in 2010. The documentation states "We've based the initial version of [OGP] on RDFa which means that you'll place additional tags in the “of your web page" ("The Open Graph protocol"). In practice, this follows the Dublin Core approach of adding `<meta>` tags to the `<head>` section of a web page, and thus would seem to be susceptible to the same relevancy manipulation techniques. However, twenty years later, search engines have much more sophisticated algorithms for detecting the misleading use of metadata. Facebook recommends that developers "mark up [their] website with Open Graph tags to take control over how [their] content appears on Facebook" ("Open Graph markup"). With Facebook reporting almost 1 billion daily active users by March 31, 2015 ("Facebook Reports First Quarter 2015 Results"), many developers find that recommendation hard to ignore.

OGP is a useful approach for providing more structured information about a given web page, but as it is limited to the `<head>` section of a web page, OGP is primarily useful for summarizing web pages for social media sharing (a title, description, and image). It is not well-suited for publishing structured data about distinct objects that appear within the content of the page, such as hours of operation and descriptions of events; thus, this article focuses on the use of schema.org to satisfy these requirements in library web pages.

Back to schema.org

Looking at a real library branch web page helps identify some of the major elements that libraries will want to mark up using schema.org. This article uses an example based on the home page for the South End branch of the GSPL. The page offers many pieces of data that would be useful to search engines or other tools, including:

- Library name
- Address
- Home page
- Phone number
- Fax number
- Hours of operation
- Description of available services
- Picture
- Related branches

To start applying the schema.org vocabulary to this library page, you need to identify the central object of the page: the library itself. The schema.org "Search" box enables you to find that there is, indeed, a `Library` type. In schema.org, as with many vocabularies, types (that is, classes of objects that have properties to describe them) start with an uppercase letter, while properties (that is, the possible attributes of objects) start with a lowercase letter.

The documentation for types in schema.org list the most specific properties for a given type at the top, and then, as you move down, you see the more generic properties that that given type inherits from its more general parents. In the case of `Library`, it inherits properties from `LocalBusiness`, which in turn inherits more general properties from `Organization`, which in turn inherits the most general properties from the top-level `Thing` type.

The first step is to tell the machines that you are going to describe a Library on this HTML page. To do so, you can use microdata, RDFa, or JSON-LD. Microdata and RDFa are competing but reasonably equivalent standards for adding simple properties to HTML to identify markup in context, while JSON-LD is JavaScript markup that appears in its own <script> tag. This article uses RDFa, as it is a W3C standard and, along with microdata, is still most broadly recognized by search engines.

On Scope

Put very simply, the scope of a given HTML element extends from its start tag to its closing tag, including all of the content inside of it. One convention is to use the `<body>` element to identify the main thing that we're marking up on an HTML page.

With RDFa, you can identify the default vocabulary in the external scope using the `@vocab` attribute; and then after that, all types and properties are automatically appended to the vocabulary URL. You use the `@typeof` attribute to declare what type you are marking up. So your body tag should now look like:

```
<body vocab="http://schema.org/" typeof="Library">
...
</body>
```

Now you need to identify the name of the library, in this case, "South End Library". Looking at all of the properties for `Library`, you will find `name` is a property defined at the most generic level of the vocabulary as part of `Thing`. The `name` property is used to identify every name-like thing in schema.org, including the titles of books and movies. In well-structured HTML, the main title of the page is contained within an `<h1>` element, and indeed, the example follows that pattern. To declare a property of a type in RDFa, use the `@property` attribute.

```
<body vocab="http://schema.org/" typeof="Library">
...
  <h1 property="name">South End
Library&nbsp;<em>&nbsp;</em></h1>
...
</body>
```

Aside

As you start adding structured markup to your HTML, you will often find problems in the markup that you would not normally see in your browser. In this example, `<em>&nbsp;<em>` is appended to the end of the library name. Use the opportunity to clean up your HTML in general.

Picture This

People often share links via social media, and the social media service does its best to find an image to associate with the link. Unfortunately, in the absence of any machine-readable metadata, the service occasionally picks a random image from the latest news or event post that has nothing to do with the core content on the page. You can help those services choose an appropriate image from a web page by declaring the `image` property, which also is defined generically on the `Thing` type, for the object you're describing.

In this case, there is an image of the South End Library sign defined by an `<img>` element. If an RDFa parser finds a `@src` or `@href` attribute on the same element as a `@property` attribute, it uses the value of the `@src` or `@href` attribute. Otherwise, the RDFa parser uses the text content of the element as its value. In this case, there is a `@src` attribute on the `<img>` element, so you can simply add the image property to the element:

```
<body vocab="http://schema.org/" typeof="Library">
...
  <h1 property="name">South End Library</h1>
  <p><img property="image"
      src="/en/aboutus/resources/Hours_South.jpg"
      alt="Exterior view of the South End Library"></p>
...
</body>
```

With the addition of these few attributes to the HTML, the web page for this library branch now defines a type, name, and image in a machine-readable manner.

Testing Your Work: Structured Markup Tools

Before going any further, consider testing your work as you progressively enhance the web page. A number of testing tools will report on what structured data is contained in a given chunk of HTML. A few of my favourites are:

- RDFa play - visualizes RDFa markup
- Structured Data Linter - a strict parser that warns about misuse of schema.org and markup
- Structured Data Testing Tool - Google's testing tool

Each time you adjust your web content to apply more structured data, or to incorporate design or content changes to your pages, you should test your markup using one or more of these tools to ensure that you are publishing the structured data you expect.

Adding a Logo and a URL for the Library

When you run the current example through the structured markup tools, the Google tool complains that the `Library` type is missing the `logo` and `url` properties. This warning tells you that, while there are dozens of possible properties that could be applied to the `Library` type, Google will generally try to use those two specific properties to enhance the display of search results. The `logo` property is inherited from both the `Organization` and `Place` types, and there is, in fact, a logo identified on the web page. In this case, it comes before the name of the library, but the order of elements on a page does not matter to machines.

```
<body vocab="http://schema.org/" typeof="Library">
...
  <img property="logo" alt="Company Logo"
   src="/en/images/structure/company_logo.png">
  ...
  <h1 property="name">South End Library</h1>
  <p><img property="image"
      alt="Exterior view of the South End Library"
      src="/en/aboutus/resources/Hours_South.jpg"></p>
...
</body>
```

To define the url property—meant to provide the definitive link for whatever thing is being described—for the library, you can simply link to the current page. This example includes a link to the current page as part of the navigation that lists all of the libraries in GSPL, so reuse that:

```
<body vocab="http://schema.org/" typeof="Library">
...
  <img property="logo" alt="Company Logo"
   src="/en/images/structure/company_logo.png">
  ...
  <a property="url" href="/en/aboutus/southendlibrary.asp"
   title="Branch Hours & Contacts::South End Library"
   class="current">South End Library</a>
  ...
  <h1 property="name">South End Library</h1>
  <p><img property="image"
      alt="Exterior view of the South End Library"
      src="/en/aboutus/resources/Hours_South.jpg"></p>
...
</body>
```

Addressing the Problem of Locating the Library

One of the goals of this exercise is to help search engines like Google, Yahoo and Bing figure out where in the world your libraries are located. Fortunately, an address is defined on the page, and schema.org defines an address property for Place and Organization parent types, so you can mark that up:

```
<body vocab="http://schema.org/" typeof="Library">
    <p property="address">1991 Regent Street <br>
    Sudbury, ON P3E 5V3<br>
    Phone: (705) 688-3950<br>
    Fax: (705) 522-7788 </p>
</body>
```

But there is a problem which you can quickly identify using one of the RDFa testing tools: the scope of the <p> element that contains the address also contains the phone and fax numbers. You want to separate those out. A common approach is to wrap the target content in a new <span> element on which we can define the property:

```
<body vocab="http://schema.org/" typeof="Library">
    <p><span property="address">1991 Regent Street <br>
    Sudbury, ON P3E 5V3</span><br>
    Phone: (705) 688-3950<br>
    Fax: (705) 522-7788 </p>
</body>
```

Now you can use the same approach to identify the phone and fax numbers using the properties that you find defined for `Place` and `Organization`.

```
<body vocab="http://schema.org/" typeof="Library">
    <p><span property="address">1991 Regent Street <br>
    Sudbury, ON P3E 5V3</span><br>
    Phone: <span property="telephone">(705) 688-3950</span><br>
    Fax: <span property="faxNumber">(705) 522-7788</span></p>
</body>
```

When you check the documentation for `address`, you should notice that the expected value is `PostalAddress`. All of the properties that we've applied until now have expected either Text or URL types as their values. While search engines expect humans to get things wrong in creating schema.org markup and will do the best they can if they're just given simple text values for properties, they can do a much better job of interpreting the web page if they're given more structured data.

You can identify `PostalAddress` as a schema.org type because it starts with a capital letter. To communicate to the machines that you're embedding a `PostalAddress` type in the markup, you have to add another `@typeof` attribute to the `address` property:

```
<body vocab="http://schema.org/" typeof="Library">
    <p><span property="address" typeof="PostalAddress">
    1991 Regent Street <br>
    Sudbury, ON P3E 5V3</span><br>
    Phone: <span property="telephone">(705) 688-3950</span><br>
    Fax: <span property="faxNumber">(705) 522-7788</span></p>
</body>
```

Then you can define the `streetAddress`, `addressLocality`, `addressRegion`, and `postalCode` properties by adding more <span> tags:

```
<body vocab="http://schema.org/" typeof="Library">
    <p><span property="address" typeof="PostalAddress">
      <span property="streetAddress">1991 Regent
```

```
Street</span><br>
      <span property="addressLocality">Sudbury</span>,
      <span property="addressRegion">ON</span>
      <span property="postalCode">P3E 5V3</span>
    </span><br>
    Phone: <span property="telephone">(705) 688-3950</span><br>
    Fax: <span property="faxNumber">(705) 522-7788</span></p>
</body>
```

We're starting to get into some serious markup here! As you add more properties to your HTML, you might consider using CSS instead of hard-coded `<br />` elements to control the display and offer more flexibility for different browsing experiences, such as mobile web browsers and screen readers.

Hours of Operation

Along with knowing where your library is, your users want to know when it is open. Opening hours information is so important for the schema.org search engines that the vocabulary offers two different ways to declare it. This article uses the slightly more complex, but at the same time easier to deploy, `openingHoursSpecification` property and corresponding `OpeningHoursSpecification` type to identify the opening hours for the library in the example:

```
<body vocab="http://schema.org/" typeof="Library">
<tr>
  <td>
    Sunday<strong>**</strong></td>
  <td>12 PM -4 PM</td>
</tr>
</body>
```

Aside

For accessibility purposes, the table header should be in a `<thead>` and use `<th>`. The "`**`" should link to the footnoted content so that a screen reader can help a user understand why that Sunday has been flagged.

```
<body vocab="http://schema.org/" typeof="Library">
<tr property="openingHoursSpecification"

    typeof="OpeningHoursSpecification">
  <td property="dayOfWeek"

      href="http://purl.org/goodrelations/v1#Sunday">
    Sunday<strong>**</strong></td>
  <td><time property="opens" datetime="12:00:00">12 PM</time>
    - <time property="closes" datetime="16:00:00">4 PM</time>
  </td>
</tr>
</body>
```

This markup has introduced several new concepts. First, rather than using a simple text value for the day, the example added an `@href` attribute that points at a specific value from the enumeration for the `DayOfWeek` type expected by the `dayOfWeek` property. Second, the `opens` and `closes` properties used specific time formats as required by the `Time` type supplied via the `@datetime` attribute in a `<time>` tag. While this is not strictly inline markup, it is still close to the content in context, and this approach is required when the human-readable content itself does not meet the standard format demanded by the property.

Related Branches

The `LocalBusiness` type from which `Library` inherits offers the `branchOf` property, which enables web pages to identify a parent organization. Given a link to the parent organization, search engines can then identify all other branches that link to this parent organization and thereby determine the overall shape of a group of libraries. The simplest approach is to use the overall home page for the complete set of libraries as the link to the parent organization, as is expressed in the navigation:

```
<body vocab="http://schema.org/" typeof="Library">
...
  <a property="branchOf" href="/en/index.asp">
    <img property="logo" alt="Company Logo"
     src="/en/images/structure/company_logo.png">
  </a>
...
</body>
```

Remember, putting a `@property` on an <a> tag automatically takes the value of the `@href` attribute.

Events

Libraries want the events they hold to be a success. Making the data about the events as machine-readable as possible so that search engines, websites and apps that aggregate events from across different organizations in your geographic area, and even your own website so that it can repurpose the event information in a news feed, will help promote your events.

Unfortunately, GSPL uses an Innovative Interfaces, Inc. events calendar which breaks the very first prerequisite this article identified for making content visible to machines: its `robots.txt` file does not allow access to the `/iii/` directory in which events are advertised. It also uses session-based URLs. For this example, use the event information under the assumption that your library can use a calendar system that is friendly to the web and interested in making their information openly available.

Here is the core information from one event:

> Family - Story Time @ South End Library
>
> Description: Develop your child's appreciation of language, rhythm and imagination through storytelling, puppetry, songs, finger plays and rhymes.
>
> Date: Thursday, January 15, 2015
>
> Time: 10:30 AM - 11:30 AM
>
> Location: South Branch
>
> Program Type: Family
>
> Series: Thursdays, 08-01-2015 to 29-01-2015, 10:30AM - 11:30AM
>
> Public Note: Free! Children under 12 must be accompanied by a parent or guardian at all times while in the library.
>
> 30 Seats Remaining

The schema.org site defines `Event` as a type, and one of its subtypes is the more specific `ChildrensEvent` type, which seems appropriate for this event. Mark up the basic name and `description` properties, using the `description` property multiple times:

```
<div vocab="http://schema.org/" typeof="ChildrensEvent">
  <div property="name">Family - Story Time @ South End
Library</div>
  <div>Description: <span property="description">Develop your
child's appreciation of language, rhythm and imagination through
storytelling, puppetry, songs, finger plays and
rhymes.</span></div>
  <div>Date: Thursday, January 15, 2015</div>
  <div>Time: 10:30 AM - 11:30 AM</div>
  <div>Location: South Branch</div>
  <div>Program Type: Family</div>
  <div property="description">Public Note: Free! Children under
12 must be accompanied by a parent or guardian at all times
while in the library.</div>
  <div property="description">30 Seats Remaining</div>
</div>
```

As the South Branch is both the location and the organizer of the event, you can use both values in a single `@property` attribute by separating them with a space:

```
<div vocab="http://schema.org/" typeof="ChildrensEvent">
  <div property="name">Family - Story Time @ South End
Library</div>
  <div>Description: <span property="description">Develop your
child's appreciation of language, rhythm and imagination through
storytelling, puppetry, songs, finger plays and
rhymes.</span></div>
  <div>Date: Thursday - January 15 2015</div>
  <div>Time: 10:30 AM - 11:30 AM</div>
  <div>Location: <a property="location organizer"
href="/southendlibrary.asp">South Branch</a></div>
  <div>Program Type: Family</div>
  <div property="description">Public Note: Free! Children under
12 must be accompanied by a parent or guardian at all times
while in the library.</div>
  <div property="description">30 Seats Remaining</div>
</div>
```

Finally, you need to provide the start and end dates and times for your event. As with opening hours, you can use the `<time>` element with the `@datetime` property, but this time the date is included in the value of the property:

```
<div vocab="http://schema.org/" typeof="ChildrensEvent">
  <div property="name">Family - Story Time @ South End
Library</div>
  <div>Description: <span property="description">Develop your
child's appreciation of language, rhythm and imagination through
storytelling, puppetry, songs, finger plays and
rhymes.</span></div>
  <div>Date: Thursday - January 15 2015</div>
  <div>Time:
    <time property="startDate" datetime="2015-01-15T10:30">10:30
AM</time>
    - <time property="endDate" datetime="2015-01-15T11:30">11:30
AM</time></div>
  <div>Location: <a property="location organizer"
href="/southendlibrary.asp">South Branch</a></div>
  <div>Program Type: Family</div>
  <div property="description">Public Note: Free! Children under
12 must be accompanied by a parent or guardian at all times
while in the library.</div>
  <div property="description">30 Seats Remaining</div>
</div>
```

To narrow the scope of the event a little further, you can add the `typicalAgeRange` property using a `<meta>` element and the `@content` attribute to suggest that it is appropriate for ages 2 through 5.

```
<div vocab="http://schema.org/" typeof="ChildrensEvent">
  <div property="name">Family - Story Time @ South End
Library</div>
  <div>Description: <span property="description">Develop your
child's appreciation of language, rhythm and imagination through
storytelling, puppetry, songs, finger plays and
rhymes.</span></div>
  <div>Date: Thursday - January 15 2015</div>
  <div>Time:
    <time property="startDate" datetime="2015-02-15T10:30">10:30
AM</time>
    - <time property="endDate" datetime="2015-02-15T11:30">11:30
AM</time></div>
  <div>Location: <a property="location organizer"
href="/southendlibrary.asp">South Branch</a></div>
  <div>Program Type: <meta property="typicalAgeRange"
content="2-5">Family</div>
  <div property="description">Public Note: Free! Children under
12 must be accompanied by a parent or guardian at all times
while in the library.</div>
  <div property="description">30 Seats Remaining</div>
</div>
```

Beyond Search Engines

While this article has focused on optimizing web pages for the consumption of search engines, making your library information publicly available in a well-structured format offers other advantages. You may be better positioned to build a local search service that blends your website location and event data with bibliographic and archival object descriptions from catalogues and digital archives applications. Your community members will more easily incorporate your library information and events into their own websites and mobile applications.

Beyond your community, publishing institutional data using a common vocabulary like schema.org opens up the possibility of provincial, national, and international library associations building a truly authoritative registry of libraries by regularly crawling member library sites and aggregating the information, rather than relying on occasional manual updates. If more libraries adopted schema.org markup for their web pages, an effort like "Canada's Libraries and Archives" (Antoniuk et al) could, rather than being a snapshot of library and archive data at one point in time, become a continually updated resource that incorporates contact information, opening hours, events, and other rich information.

Conclusion

This article has provided guidelines for ensuring that your library web pages can be efficiently indexed by search engines through the use of `robots.txt` and sitemap files. I demonstrated how libraries currently fare poorly in search engines' "local search" tools, at least without updating location and opening hours metadata in various social media sites, and posited that incorporating the machine-readable schema.org vocabulary into library web pages offers a potential centralized solution. Towards that end, this article took the home page for a sample library branch and followed a few simple, concrete steps to augment the pages with schema.org types and properties. I also demonstrated how to publish schema.org markup for events. Finally, I have provided several examples of how incorporating schema.org markup into library web pages not only optimizes those pages for search engines, but builds a firmer foundation for mobile and web applications.

Works Cited

Antoniuk, Jeffery et al. "Canada's Libraries and Archives." *RSC Expert Panel on The Status and Future of Canada's Libraries and Archives*. Canadian Writing Research Collaboratory, June 2014. Web. 15 Jan. 2015. <http://cwrc.ca/rsc-src/>.

Evergreen Documentation Interest Group. "Sitemap generator." *Evergreen Documentation*. evergreen-ils.org, 2014. Web. 24 Jan. 2015. <http://docs.evergreen-ils.org/2.8/_sitemap_generator_2.html>.

"Facebook Reports First Quarter 2015 Results." *Investor Relations*. investor.fb.com, 22 Apr. 2015. Web. 24 Apr. 2015. <http://investor.fb.com/releasedetail.cfm?ReleaseID=908022>.

Gibson, Becky, and Richard Schwerdtfeger. "DHTML Accessibility: Solving the JavaScript Accessibility Problem." *Proceedings of the 7th international ACM SIGACCESS conference on Computers and accessibility*. ACM, 2005. 202-203 PDF file.

Guha, Ramanathan. "Introducing schema. org: Search Engines Come Together for a Richer Web." *Google Official Blog* (2011). Web. 15 Jan. 2015. <http://googleblog.blogspot.ca/2011/06/introducing-schemaorg-search-engines.html>.

Gyongyi, Zoltan, and Hector Garcia-Molina. "Web spam taxonomy." *First international workshop on adversarial information retrieval on the web (AIRWeb 2005).* 2005.

Koster, Martijn. "A Method for Web Robots Control". *The Web Robots Pages*. robotstxt.org, 4 Dec. 1996. Web. 24 Jan. 2015. <http://www.robotstxt.org/norobots-rfc.txt>.

Ng, Cynthia. "Making web services accessible with universal design". *Learning LibTech*. cynng.wordpress.com, 22 Jan. 2015. Web. 24 Jan. 2015. <https://cynng.wordpress.com/2015/01/22/making-web-services-accessible-with-universal-design/>.

Ontario. Ministry of Economic Development, Employment and Infrastructure. "Make your website accessible". *Making Ontario Accessible*. Queen's Printer for Ontario, 2015. Web. 24 Jan. 2015. <http://www.mcss.gov.on.ca/en/mcss/programs/accessibility/info_sheets/info_comm/website.aspx>.

"Open Graph Markup". *Facebook Developers*. developers.facebook.com, 2015. Web. 24 Apr. 2015. <https://developers.facebook.com/docs/sharing/webmasters#markup>.

Scott, Dan. "Seeding Structured Data by Default via Open Source Library Systems." *The Semantic Web: Trends and Challenges*. Springer International Publishing, 2014. 659-674. Digital file.

"Search Engine Optimization". *VuFind Wiki*. vufind.org, 13 Jun. 2014. Web. 24 Jan. 2015. <https://vufind.org/wiki/search_engine_optimization>.

Sullivan, Danny. "New Google Sitemaps Web Page Feed Program". *Search Engine Watch*. Incisive Interactive Marketing LLC, 2 Jun. 2005. Web. 24 Jan. 2015. <http://searchenginewatch.com/sew/news/2061916/new-google-sitemaps-web-page-feed-program>.

"The Open Graph protocol". *ogp.me*, 20 Oct. 2014. Web. 24 Jan. 2015. <http://ogp.me>.

Weibel, Stuart. "A proposed convention for embedding metadata in HTML." *W3C workshop on distributed indexing and searching*. 1996. Web. 24 Jan. 2015. <http://www.w3.org/Search/9605-Indexing-Workshop/ReportOutcomes/S6Group2.html>.