Information Retrieval with AJAX:

Searching Digital Libraries

Literature Review

Marc Bowes

May 5, 2009

ABSTRACT

Searching is paramount to extracting previously stored information from Digital Library systems.

A lot of research has been done on information retrieval in the past, with the grand vision only now coming into fruition. While these fruits now form the basis of billion dollar companies, there has always been an emphasis on on-line retrieval.

This paper will investigate a completely off-line Digital Library system built on AJAX technology, specifically how to search information given the limitations of AJAX. Emphasis will be placed on the integration of existing technology and how it can be used to implement information retrieval using AJAX.

1 INTRODUCTION

Digital Libraries describe the conceptual space whereby a computer system makes it possible to store and retrieve a collection of digital objects [1, 17, 18, 19].

While a lot of research has gone into Digital Libraries, most Digital Library solutions follow a model where information is retrieved across a network [1, 17, 18, 19]. Historically, this has been necessary in order to overcome limitations in hardware, specifically storage [1]. Modern technology has not only overcome these limitations, but has done so in a way which is affordable.

These advances allow a Digital Library system to run on a single machine. This paper will look at the applications of existing information retrieval techniques to an AJAX-based Digital Library solution. The Digital Library system will be distributed on some sort of medium (for example, a DVD) and should “just work”, without needing to install any software on the target machine. A DVD reader and a Web browser with Javascript support should be good enough. This approach is feasible, as demonstrated by the Bleek and Lloyd Collection prototype [15, 16].

2 AJAX

1. Background

AJAX, or Asynchronous Javascript and XML, is a technology which allows one to add programming logic to an HTML page [2, 5]. AJAX describes the set of technologies and programming approaches to making the Web more dynamic. Traditionally, AJAX is used to increase responsiveness and usability on a Web page, by only updating portions of a page at a time [2]. However, AJAX can be used to build an application without any need for external connectivity, that is, a desktop-style application in a Web browser.

AJAX allows for the dynamism and richness typically reserved for desktop applications [2, 3, 4, 5]. This is due to the programmable layer provided by Javascript. This layer allows for manipulation of the otherwise static HTML. Javascript uses XMLHttpRequest (XHR), a de facto standard developed by Microsoft for transferring data [2].

In regular HTTP communication, a client sends requests to a Web server. This server then returns HTML, which is rendered in the client's browser. This model is inefficient when the returned page is largely similar to the previous page [2]. AJAX uses asynchronous techniques to improve the user's experience – reloading only aspects of the page which change. While the asynchronous requests are almost always over a network, this is not a limitation.

The AJAX-based Digital Library system will make use of AJAX techniques to provide desktop software reliant only on a browser with Javascript support.

2. Approaches

Javascript is limited in the connections it can make, including access to the computer it is being run on. These limitations are for security purposes [6, 7, 8].

Javascript limitations prohibit reading or writing to disk. This is a formidable obstacle, and not easily overcome. There are three primary approaches to solving this problem.

2.1 ActiveX

Internet Explorer provides ActiveXObject, which allows file access. However, while this is a viable workaround, it would limit the Digital Library system to Internet Explorer only. This is highly undesirable, as Internet Explorer runs on Microsoft Windows only, which requires licensing. This will impose limitations on who can use the Digital Library.

The upside of this route is that the code is very simple – sample code is widely available on the Internet [10], including Microsoft's own documentation [9]. If one had control over the target market, using ActiveX could be the best solution.

2.2 Mozilla extensions

On the opposite side of the fence, Mozilla provides extensions to their Web browser, Firefox. These extensions make it possible to compile additional functionality into the browser's Javascript [21].

While these extensions make it possible to read or write to files, this approach suffers from similar drawbacks to the ActiveX approach by limiting the code to Firefox.

While it is possible to write in support for both ActiveX and Mozilla extensions, this is not a good solution. There are more browsers to cater for, such as Apple's Safari browser, or the popular Opera browser.

2.3 Applet

This approach involves embedding a Java Applet in the Web page. The Javascript then connects to the Applet as if the Applet was any other Web server (see Figure 1). The Applet has access to the client's computer.

Provided the Applet is on the same disk as the data, the Applet (usually) requires little to no authorization. In the case of the Digital Library system, this means that the Applet would have to reside on the same media as the HTML pages and data repository [11].

Javascript is able to access the Applet, and its public methods, as if it were any other object in the DOM. This approach means that the Javascript code remains clean, while keeping cross-browser support.

The restrictions on domain could prove problematic should the repository ever be split across media. In this case, code in the Applet would need to be electronically signed.

More information on this topic is available in the SUN developer guide [12].

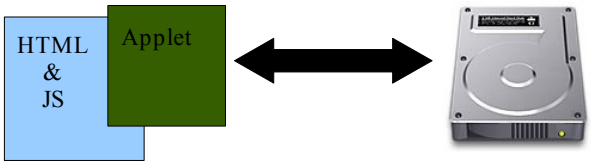

Figure 1: Javascript in the HTML proxies through (embedded) Java Applet to read from and write to disk.

3 INFORMATION RETRIEVAL

3.1 Background

Information retrieval is not a new topic in Computer Science. Since Vannevar Bush's publication in 1945, significant research has been done in the field [1]. Research has typically focused on how to search ever increasing volumes of information, as well as how to improve the quality of the yielded results.

Initially, information retrieval techniques were limited by the available hardware in the 1960s, where storage space limited searching to title, author, journal and keywords [1]. As hardware improved, systems were able to support full-text searches, where the full text of the article is indexed [1]. Beyond full-text searching lies semantic searching, where emphasis is placed on concepts conveyed by the query, rather than the occurrence of the exact words.

The existing research is sufficient for the needs of the AJAX-based Digital Library system. This paper will investigate the application of existing techniques.

3.2 Full-text searching

Searching the full text of a document is an expensive. operation. It is an exhaustive search, as the full text (title, author, keywords, abstract) in all documents must be scanned in order to report results.

To lessen the cost of full-text searching, a technique known as indexing is used [1, 14, 15, 16]. A so-called “inverted index” is created, where words are mapped to articles [1]. This inverted index makes lookup

operations cheap: one only needs to apply boolean operators (such as AND, OR) to the list of returned article pointers.

Inverted indices can be extended with proximity operators, which allow queries to narrow the scope of the returned results by enforcing that certain terms appear near each other in the text. This is achieved by storing the offset of the word along with the article pointer [1].

3.3 Preprocessing

In order for the above-mentioned technique to work, the inverted index needs to be generated. Index generation has been extensively covered by research into information retrieval [1, 14, 15, 16].

The preprocessor needs to traverse the digital collection, inverting each document so that words are mapped to their occurrences in articles [1, 14].

Certain words, such as 'and' or 'of', are considered too noisy to be indexed, and are omitted from the procedure [1, 14].

Words are shrunk to a canonical form, so that matches can be performed against the base word, rather than the specific version – for example, 'retriev' rather than 'retrieve' or 'retrieving' [1].

A document index need also be created. This is a mapping of document identifiers to the actual resources [14].

3.4 Using the index

The search page will read from the index, perform the necessary operations, and generate a list of results (see Figure 2). The generated results will provide summary information (such as title and author) as well as links to the actual digital objects.

This page will need to use a workaround to read from the indeces, as mentioned in section 2: AJAX. The search is entirely off-line: it is only displaying a subset of pregenerated content.

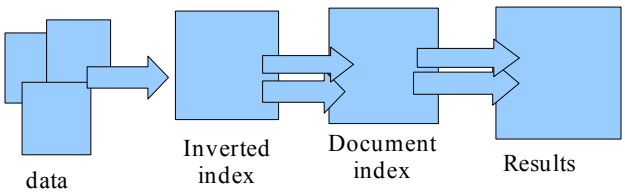

Figure 2: Data is inverted so that words map to document pointers. Points are looked up in the document index, which is used to generate human readable results.

3.5 Search in unusual systems

In traditional information retrieval, a search query is built and then results are retrieved. The user is then able to browse the results or search again [2]. With the advent of Web 2.0 has come the notion of dynamic searching, a hybrid of searching and browsing [2]. The principal is that the system provide suggestions while the query is being built.

Facebook (http://facebook.com), a social networking site, allows one to search one's friends. While entering a name, a drop down list appears with partial matches to the query. For example, typing in “John” might yield a list containing “John Smith” or “Andrew Johnson”. This behavior is achieved using AJAX to request results based on the current input. Facebook use the user's history to rank the results, so that friends who are visited more often are ranked higher. This technique allows quick matching of queries to results, usually without leaving the page.

It is also possible to embed all search results in the search page using Javascript [8]. This allows for rapid negotiation of results as no network traffic is required, making it suitable in cases where latency is an issue. However, this technique is not suitable for handling lots of data due to memory restrictions.

A natural overlap of the two approaches is to only embed only frequently referenced friends in Javascript variables, making the search appear very responsive. This is useful in scenarios where latency may be high and memory cannot hold all the results. A great example of this scenario is mobile devices such as cellphones. MXit (http://mxit.com), a mobile chat client, use an adaption of this technique by only listing frequently referenced online contacts (the rest of the contact list is sent later) [25]. This is useful in cases where a user may have a thousand contacts, which could dramatically slow down the login process.

4 SIMILAR SYSTEMS

4.1 Greenstone

Greenstone is a Digital Library system which makes it possible to export a digital collection for off-line viewing [19]. The resulting system is static, but needs a Web server in order to run. Greenstone collections can be updated, but the rebuilding process can take a day or two for large collections [19].

Greenstone supports full-text searching and metadata browsing. The system is extensible, meaning it can

support a wide variety of metadata types, and plugins can be written to handle new types (conversion to Dublin Core) [19].

4.2 EPrints and DSpace

EPrints is an open-source package for building open access repositories. It is written in Perl and runs under most operating systems, although the Windows version is still under development [23].

DSpace is an open-source package for managing a wide variety of digital objects. It is written in Java and JSP, using Oracle or PostgreSQL for storage [24].

Both systems provide similar functionality (browsing and searching digital collections) and there is much contention as to which system is better [22]. However, neither system provides functionality to export the collection: they are exclusively on-line solutions.

5 CONCLUSIONS

This paper has looked at the applications of existing technology towards a completely off-line Digital Library solution. Software requirements are minimal, and modern operating systems come standard with the required tools.

The combination of a Web browser and Javascript provides an almost perfect environment to build a desktop application. However, security concerns limit what Javascript has access to. As a workaround, it is suggested than an in-page Applet is built. This Applet will act as a proxy: it will do the disk access on behalf of the page (AJAX) code.

Information retrieval techniques are already well established and suit the needs of the proposed off-line AJAX Digital Library. A preprocessor is inevitable, as an index needs to be generated in order to search the digital collection. The preprocessor will generate an inverted index: mapping words to the documents in which they appear. Logical operators such as AND/OR are the applied to the query to narrow the scope of the lookup.

6 REFERENCES

1. Schatz, B.R., 1997, *Information Retrieval in Digital Libraries: Bringing Search to the Net* , Science, 275:237-334, January 1997
2. Wusteman, J., O'hIceadha, P., 2006, *Using Ajax to Empower Dynamic Searching*, Information Technology and Libraries, 25(2):57-64, June 2006
3. Coombs, K.A., 2007, *Building a Library Web Site on the Pillars of Web 2.0*, Computers in Libraries, 27(1), January 2007
4. Miller, P., 2005, *Web 2.0: Building the New Library*, Ariadne, 45, October 2005
5. O'Reilly, T., 2005, *What Is Web 2.0.* Available: http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html, September 2005
6. Anupam, V., Mayer, A., 1997, *Security of Web Browser Scripting Languages: Vulnerabilities, Attacks, and Remedies*, Proceedings of the 7th USENIX Security Symposium, January 1998
7. Garrett, J.J., February 18 2005, *Ajax: A New Approach to Web Applications.* Available: http://adaptivepath.com/publications/essays/archives/000385.php, February 2005
8. Mikkonen, T., Taivalsaari, A., October 2007, *Using JavaScript as a Real Programming Language* , SMLI TR-2007-168, October 2007
9. ANON, 2007, *ActiveX Data Objects (ADO) Frequently Asked Questions.* Available: http://support.microsoft.com/kb/183606, March 2007
10. ANON, 2005, *Using JavaScript to read a client-side file*. Available: http://timstall.dotnetdevelopersjournal.com/using_javascript_to_read_a_clientside_file.htm, April 2005
11. ANON, date unknown, *The Javascript FAQ*. Available: http://www.linuxtopia.org/online_books/javascript_guides/javascript_faq/reading2.htm
12. ANON, date unknown, *JavaScript to Java Communication (Scripting)*, http://java.sun.com/j2se/1.4.2/docs/guide/plugin/developer_guide/js_java.html
13. Baeza-Yates, R., Ribeiro-Neto, B., February 1999, *Modern Information Retrieval*, SMLI TR-2007-168, February 1999
14. Witten, I.H., Moffat, A., Bell, T.C., 1999, *Managing Gigabytes*, San Fransisco:Morgan Kaufmann Publishers, Inc.
15. Suleman, H., *in-Browser Digital Library Services*, ECDL 2007
16. Suleman, H., *Digital Libraries without Databases*, ECDL 2007
17. Lagoze, C., Krafft, D., Payette, S., Jesuroga, S., November 2005, *What is a Digital Library Anymore, Anyway?*, D-Lib Magazine, 11(5), November 2005
18. ANON, 1995, *Digital Libraries*, Communications of the ACM 38(4), April 1995
19. Witten, I.H., McNab, R.J., Boddie, S.J., Bainbridge, D., 2000, *Greenstone: A Comprehensive Open-Source Digital Library Software System*, University of Waikato, 2000
20. ANON, date unknown, *Dynamic HTML and XML: The XMLHttpRequest Object. Available*: http://developer.apple.com/internet/webcontent/xmlhttpreq.html
21. ANON, date unknown, *How to read and write files in JavaScript*. Available: http://www.c-point.com/JavaScript/articles/file_access_with_JavaScript.htm

22. Kim, J., 2005, *Finding Documents in a Digital Institutional Repository: DSpace and Eprints*, Proceedings 68th Annual Meeting of the American Society for Information Science and Technology (ASIST), November 2005

23. ANON, date unknown, *EPrints*. Available: http://en.wikipedia.org/wiki/Eprints

24. ANON, date unknown, *DSpace*. Available: http://en.wikipedia.org/wiki/DSpace

25. ANON, 2008, *MXit Open Protocol 5.8.2*, MXit Devzone, March 2009. Available: http://devzone.mxit.com/download/2 (requires account)