Une démonstration d'un crawler intelligent pour les applications Web

Muhammad Faheem
Pierre Senellart

Institut Mines–Télécom; Télécom ParisTech; CNRS LTCI
Paris, France

We demonstrate here a new approach to Web archival crawling, based on an *application-aware helper* that drives crawls of Web applications according to their types (especially, according to their content management systems). By adapting the crawling strategy to the Web application type, one is able to crawl a given Web application (say, a given forum or blog) with fewer requests than traditional crawling techniques. Additionally, the application-aware helper is able to extract semantic content from the Web pages crawled, which results in a Web archive of richer value to an archive user. In our demonstration scenario, we invite a user to compare application-aware crawling to regular Web crawling on the Web site of their choice, both in terms of efficiency and of experience in browsing and searching the archive.

1 Introduction

The advent of the Web 2.0 in the past decade has had significant impact on the number of Web pages and the amount of user-generated content on the Web : Web users are now billions [9], Web search engine robots such as Google have discovered more than a trillion unique URLs [1], and one of the most popular content-management system for blogs, WordPress, is powering dozens of millions of Web sites [14].

On the other hand, only a small fraction of this Web content can be captured by today's Web crawlers, due to limited bandwidth, extreme ephemerality of some of the content, storage, indexing, or processing costs, or simply the need for selecting high-quality content. This is true for Web search engines : the number of pages indexed by Google in February 2013 is estimated to be around 40 billions [3], to be contrasted to the trillion URLs or so in the frontier. This is all the truer for Web archiving institutions such as Internet Archive [1] and Internet Memory [2] whose mission is to preserve Web content for future generations. Because of limited resources, and of the need of indexing content over time, these can at best host collections with billions of URLs.

A large part of the content on the Web comes from Web sites powered by content management systems (CMSs) incorporating content in a fixed template [6]. This includes in particular a number of Web 2.0 and social Web applications such as blogs, forums, wikis. As we argue in [5], content

1. `http://www.archive.org/`
2. `http://internetmemory.org/`

published on this range of *Web applications* does not only include the ramblings of common Web users but also pieces of information that are newsworthy today or will be valuable to tomorrow's historians. Blogs are used by politicians more and more, both to advertise their political platforms and to listen to citizens' feedback [2] ; Web forums have become a common way for political dissidents to discuss their agenda [10] ; initiatives like the Polymath Project [3] transform blogs into collaborative research whiteboards [11] ; user-contributed wikis such as Wikipedia contain quality information to the level of traditional reference materials [7].

Despite the need for durable archiving of this precious content and limited crawling resources of archiving institutions, current-day archival crawlers crawl the Web in a conceptually very simple manner, irrespectively of the nature of the Web application crawled or the content management system used. This results in redundant Web pages in the archive (as the same logical content is often accessible through different paths in a content-management system), in archives where the template of a page is not distinguished from its content, and in indexing of utility Web pages (user account management, print view of a page, social network widgets) that are not in the scope of the archival task. All together, this forms a considerable waste of crawling resources that could be used to capture other interesting information.

We demonstrate here a new approach to Web archiving that carries out optimized crawling for extraction of the main content of Web applications. We rely on an *application-aware helper* (AAH) that assists the crawling process by identifying crawled Web application types and providing appropriate crawling actions (links to follow, content to extract) accordingly. For supported Web sites, this approach directly targets useful content-rich areas, avoids archive redundancy, and enriches the archive with semantic description of the content. The AAH makes use of a knowledge base of known Web application types, together with algorithms for flexible and adaptive matching of Web applications to these types.

Next, in Section 2, we present the main features and architecture of the application-aware helper. We then present in detail in Section 3 the demonstration scenario that will be proposed to conference attendees.

A companion video for this demonstration paper is available at `http://dbweb.enst.fr/aah`. We refer to our related publication [5] for a detailed presentation of the AAH algorithms, system, and performance.

The present demonstration has also been submitted to CIKM 2013.

2 Architecture

We now present the main components of the AAH for intelligent and adaptive crawling of known and adaptable Web applications. We refer to Figure 1 as an illustration of the architecture of the system, with items of the following enumeration referring to the numbers in the figure.

1. The AAH relies on a knowledge base of Web application types which describes how to crawl a Web site in an intelligent manner. The knowledge specifies precisely how to detect a specific Web application type (CMS) and which crawling actions should be executed to crawl it. The knowledge base also describes the different levels under a Web application type and then, based on this, different crawling actions that should be executed against this specific page level. The knowledge base is described in a custom XML format, well-adapted to the tree structure of a hierarchy of Web applications and page levels. The Web application detection patterns and crawling actions are written in an XPath fragment language.
2. The system loads the Web application type detection patterns from the knowledge base and executes them against a given Web application. If the Web application type is detected, the system runs all possible Web application level detection patterns until a match is found. The

3. `http://polymathprojects.org/`

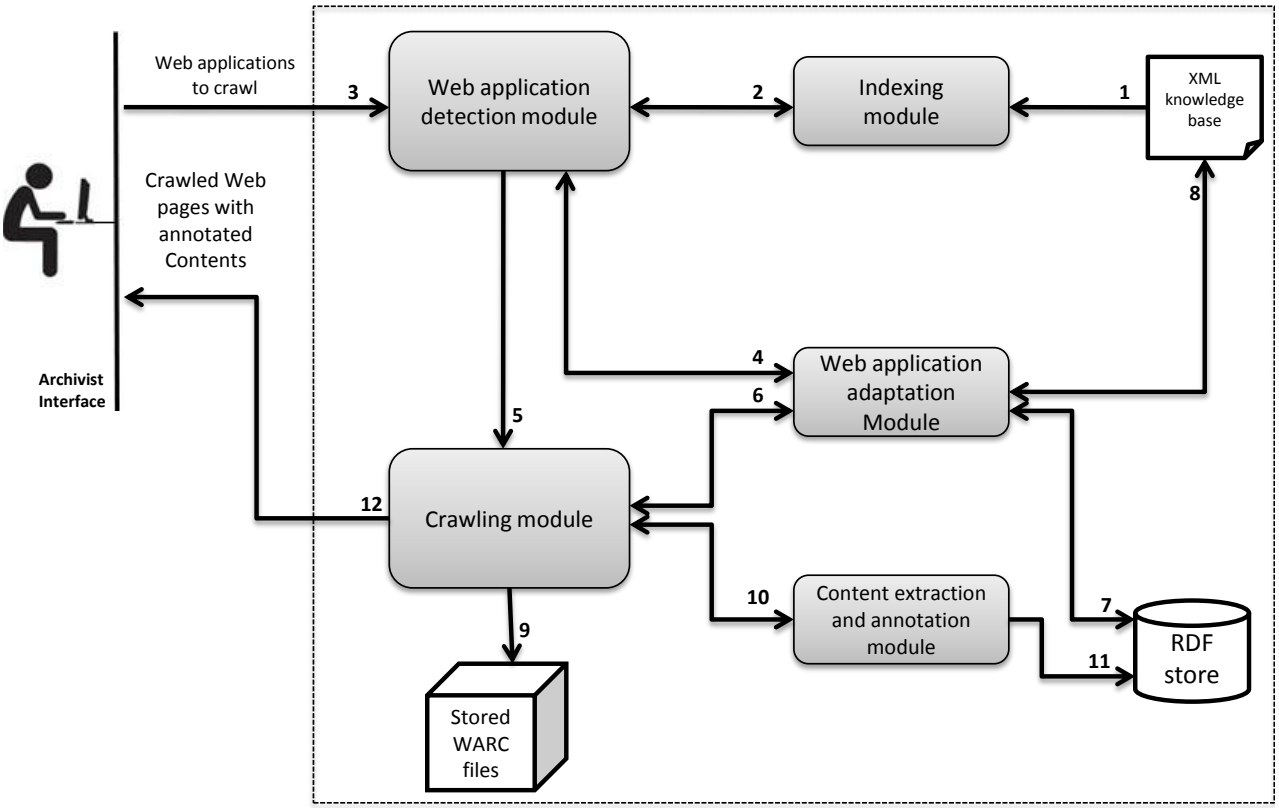
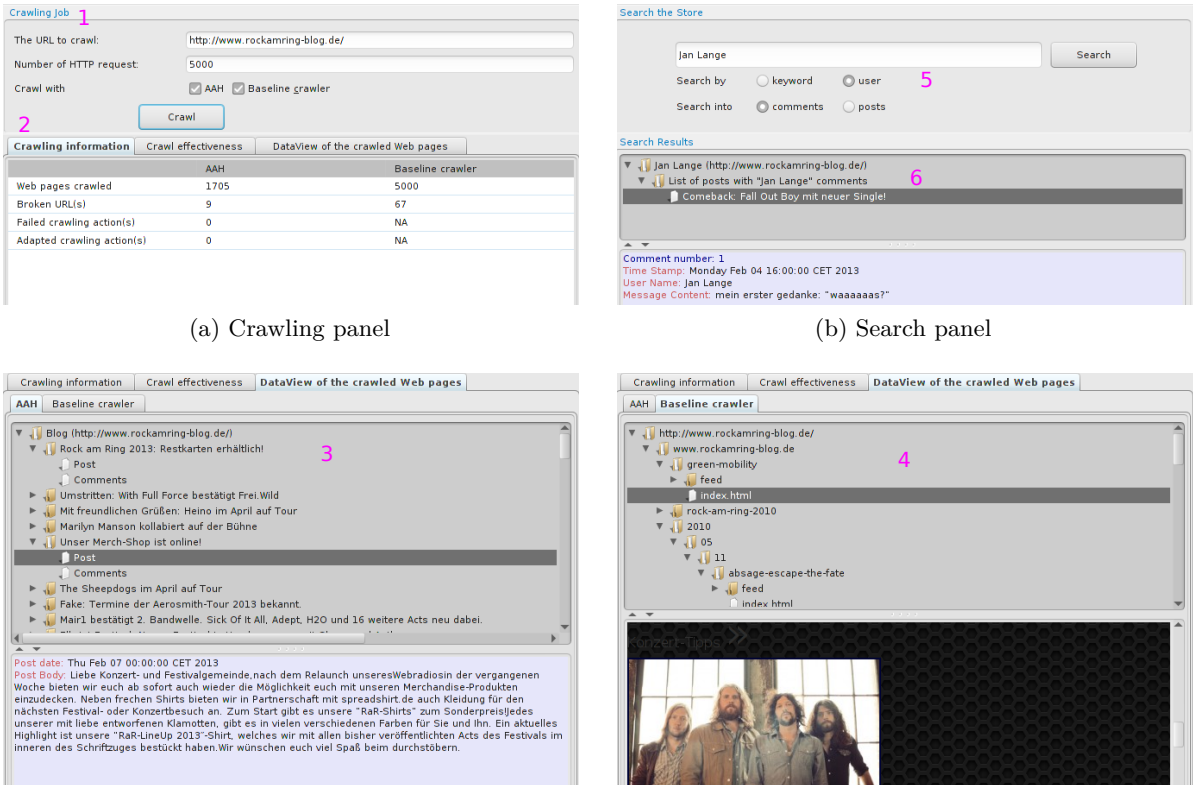


FIGURE 1: Architecture of the AAH

number of detection patterns for detecting Web application type and level will grow with the addition of knowledge about new Web applications. To optimize this detection, we maintain an index of these patterns, that uses a version of the YFilter [4] NFA-based filtering system for XPath expressions adapted to our purposes.

3. Once the system receives a crawling request, it first makes a lookup on the YFilter index to detect the Web application type and level.
4. If the Web application type is not detected, the AAH applies an adaptation strategy to find a relaxed match : we look for approximate matches of detection patterns, thereby handling different versions of a CMS.
5. When the Web application is successfully detected, the AAH loads the corresponding crawling strategy from the knowledge base and crawls the Web application accordingly. This module also implements the basic queue management and resource fetching components.
6. If the system fails to crawl the Web application because of structural changes with respect to the knowledge base, it tries determining the changes and then adapt failed crawling actions. AAH deals with two different cases of adaptation : first, when (part of) a Web application has been crawled before the template change and a recrawl is carried out after that (a common situation in real-world crawl campaigns) ; second, when crawling a new Web application that matches the Web application type detection patterns but for which (some of) the actions are inapplicable. See [5] for details.
7. A Web application that has been crawled before but cannot be recrawled with the same crawling actions is relearned for adaptation by searching for already crawled contents (URLs corresponding to navigation actions, Web objects, etc., stored in an RDF store).
8. In the process of adaptation, the system also automatically maintains the knowledge base with the newly discovered patterns and actions.



(a) Crawling panel

(b) Search panel

(c) AAH archive view

(d) Traditional archive view

FIGURE 2: Graphical user interface of the AAH

9. The crawled Web pages are stored in the form of WARC [8] files, the standard preservation format for Web archiving.
10. Structured content (individual Web objects with their semantic metadata) is extracted from each crawled page, as described in the knowledge base.
11. Structure content is stored in an RDF store, that can be queried by an archive user through SPARQL queries.
12. Crawled Web pages with annotated content are displayed to the archivist through a GUI that we describe in more detail in Section 3.

The Java implementation of the AAH is available in open source from `http://perso.telecom-paristech.fr/~faheem/aah.html`. In addition to being usable in a standalone mode for testing and demonstration purposes, the AAH has also been integrated, in the framework of the ARCOMEM project [4], in the crawl processing chain of both Internet Archive's Heritrix crawler [13] (modified for our purposes) and Internet Memory's [5] proprietary crawler.

3 Demonstration Scenario

We now describe a specific use case where the AAH helps building richer archives with less resource wasted. This is a real-world case where the AAH has been used.

The German broadcasting company SWR is a partner each year of the *Rock am Ring* music festival. SWR archivists are interested in building archives of both official information and public perception

4. `http://www.arcomem.eu/`
5. `http://internetmemory.org/`

of this music festival as displayed on the Web. Many Web sites are in the scope of this archival campaign, however, crawling, storage, analysis resources are limited, and timely archival matters, which requires to limit crawling requests to those that will effectively bring meaningful content to the archive. The AAH has been designed to meet these challenges and ensures that duplicates and templates are avoided, and all useful information has crawled.

In our demonstration scenario, conference attendees will have the possibility of either focusing on this specific Rock am Ring use case, or choose a Web site of their own to crawl.

We now present in detail our demonstration scenario. See the accompanying video for a peek at the GUI (See Figure 2).

1. The user enters in the *crawling* panel the URL of a Web site to crawl with the possibility to limit the number of HTTP requests. The user will be offered to choose any URL she wants ; obviously, interesting results will be produced only for Web application types supported by the AAH (currently, various versions of WordPress, phpBB, and vBulletin, with a robustness to template change thanks to the adaptation module). A few Web sites will be pre-crawled locally to simulate their crawl without having to rely on network delays (especially an issue since the crawler respects per-server crawling delays as per crawling ethics). The user is also given the choice of the specific crawler to run the job : either the AAH crawler, or a baseline non-intelligent crawler, or both. Selecting both allows comparing their performance.

2. Once the crawl is completed (a few seconds when run from a local cache of the Web site, arbitrary longer for a remote crawl), the system shows the number of HTTP requests made by both crawlers, as well as the number of broken links found. For instance, on one specific crawl of the blog `http://www.rockamring-blog.de/`, the AAH has made only 1,705 HTTP requests to crawl the whole given Web application, compared to 5,000 for the baseline crawler with no certainty that all useful content has been retrieved. Comparatively, the AAH also encounters fewer broken links than the baseline crawler. We additionally provide the number of failed and adapted crawling actions. The *crawl effectiveness* panel (watch video) compares crawl performance on a plot of Web site coverage (measured as a number of distinct k-grams [5]) vs number of requests.

3. The system provides the data view of the crawled Web pages with the AAH. The AAH crawls the Web pages in an intelligent manner and extracts useful information. For instance, depending on the Web application, we may show a list of blog posts with crawled content and Web objects, identifying semantic components such as post body, publication date, author, and post comments with their publication date and author. Similar views exist for Web forum content.

4. The data view of the crawled Web pages with the baseline crawler is rather very simple (direct HTML display), as Web pages are crawled blindly, without avoiding spider traps and noisy links.

5. The AAH does not only crawl the Web application in an intelligent manner but also extracts Web objects (e.g., timestamp, comments, author). Crawled Web pages and objects are stored in a large-scale RDF store [12] in the form of RDF triples. The user can run semantic queries on the triple store. For instance, she can look for the posts or comments posted by specific users by specifying their names. The interface also allows searching post bodies or comments with respect to a given keyword.

6. The result of the query are shown in the form of a data view, where, for instance, the list of posts or comments with the given user name are shown.

4 Acknowledgments

This work was funded by the European Union's Seventh Framework Program (FP7/2007–2013) under grant agreement 270239 (ARCOMEM).

Références

[1] J. Alpert and N. Hajaj. We knew the web was big... http://googleblog.blogspot.co.uk/2008/07/we-knew-web-was-big.html, 2008.

[2] S. Coleman. Blogs and the new politics of listening. *The Political Quarterly*, 76(2), 2008.

[3] M. de Kunder. The indexed Web. http://www.worldwidewebsize.com/, 2013.

[4] Y. Diao, M. Altinel, M. J. Franklin, H. Zhang, and P. Fischer. Path sharing and predicate evaluation for high-performance XML filtering. *ACM TODS*, 2003.

[5] M. Faheem and P. Senellart. Intelligent and adaptive crawling of Web applications for Web archiving. In *ICWE*, 2013.

[6] D. Gibson, K. Punera, and A. Tomkins. The volume and evolution of Web page templates. In *WWW*, 2005.

[7] J. Giles. Internet encyclopaedias go head to head. *Nature*, 438, 2005.

[8] ISO. ISO 28500 :2009, Information and documentation – WARC file format.

[9] J. Lynn. Internet users to exceed 2 billion this year. http://www.reuters.com/article/2010/10/19/us-telecoms-internet-idUSTRE69I24720101019, 2010.

[10] J. C. Mulvenon and M. Chase. *You've Got Dissent ! Chinese Dissident Use of the Internet and Beijing's Counter Strategies*. Rand Publishing, 2002.

[11] M. A. Nielsen. *Reinventing Discovery : The New Era of Networked Science*. Princeton University Press, 2011.

[12] N. Papailiou, I. Konstantinou, D. Tsoumakos, and N. Koziris. H2RDF : adaptive query processing on RDF data in the cloud. In *WWW*, 2012.

[13] K. Sigurðsson. Incremental crawling with Heritrix. In *IWAW*, 2005.

[14] WordPress Foundation. Stats. http://en.wordpress.com/stats/, 2013.