Issues in Monitoring Web Data

Serge Abiteboul

INRIA and Xyleme

Abstract. The web is turning from a collection of static documents to a global source of dynamic knowledge. First, HTML is increasingly complemented by the more structured XML format augmented with mechanisms for providing semantics such as RDF. Also, we believe that the very passive vision of web sites promoted by the current HTTP will soon turn into a much more active one based on pub/sub and web services. Thus, the monitoring of the web is likely to raise new challenges for many years. We consider here some applications involving web monitoring and some issues they raise.

1 Introduction

The web constitutes the largest body of universally accessible information throughout the history of humanity and keeps growing at a healthy pace. This new frontier that was first developed and used by a small number of experts is now turning into a huge distributed information base available to the public at large. However, besides its size, distribution and heterogeneity, the web is difficult to use because in some respect, it is also a huge, evolving junk yard:

- A lot of the information cannot be trusted, e.g., spamming, rumors, etc.
- a lot of the information on the web is stale, e.g., dead web sites or pages no longer maintained.
- even when a site or a page is alive, our vision of it may be out-to-date, e.g., full-text indexes on the web are often not aware of recent changes of pages.

To overcome this difficulty, we need to "monitor" the web. More precisely, web robots need to permanently look for new information on the web, typically by crawling it or by querying databases published on the web. Robots then need to analyze this information, e.g., to evaluate the quality of data that has been found and classify it. Finally, they need to monitor (in the sense of performing some surveillance) the data in order to detect changes.

We first consider the nature of the data one may wish to monitor on the web. We then look at typical applications involving web monitoring. While doing so, we consider technical issues raised by web monitoring. The present paper is motivated by recent works of the author. In particular, it has been motivated by works on the creation of a warehouse of XML data found on the web [11] that lead to the creation of a start-up named Xyleme [20]. The present paper presents a biased vision of a very diverse and active field and is not in any sense a comprehensive survey.

2 The nature of data to monitor

The web was first developed very homogeneously around HTML. In other words, the bulk of the web consisted of hypertext documents. While maturing, the web became increasingly complex. Nowadays, the following information may be found:

– HTML still forms the backbone of the web, but popular document formats such as DOC, PDF, PS are massively present.
– A new web exchange format, namely XML [19], promoted by the w3c [21] and the industry, is radically changing the web. XML marries, in some sense, the document world with the database world [1]. XML can be used to represent documents, but also structured or semistructured data [1].
– A lot of web data is typically retrieved as a result of queries, in particular queries to database systems. Typically, this means that data (query parameters) has to be fed to obtain more data. The term deep web or hidden web [10] are used for such accesses typically via forms, scripts or using a new standard for web services, SOAP [15].
– Finally, multimedia formats for images, sound, video are crucial components of the web that we will not discuss here.

Note that web robots mostly reach the HTML portion of the web although some may index other formats (e.g., PDF for Google [13]) or crawl XML as well (e.g. Xyleme [20]). Databases are typically seen as stop points by crawlers. So, the deep web (larger than the surface web) is still out of the reach of crawlers. However, most people would probably agree that they use more regularly the deep web (e.g., for ordering tickets, or when querying yellow pages) than the surface web. With the web maturing, we believe that this will be increasingly the case.

When considering the nature of data to monitor, one should also mention:

– Public vs. private web: the private web is typically accessible on an Intranet or on the Internet protected by passwords. The main issue here is the control of access rights. From a technical perspective, there is no real difference between private and public data.
– Static vs. dynamic. One can imagine a static page that is kept very fresh by 24-hour operators and similarly a dynamic page that for years will return the same value. The main issues are those already mentioned: the format of the data and whether or not there is a need to provide query parameters.

3 Difference with traditional databases

Although XML is often presented as a new model for data storage with its query language (Xquery [18]), we prefer to view XML here essentially as an exchange format ignoring the nature of the storage (relational, native or other) or the way the data has been produced. Indeed, in this spirit, there is more and more interest in "stream" queries for XML data, see, e.g., [3]. By this we mean that the query refers to data that is being received vs. data that is stored in a database. Note that the problem is intrinsically different. The core of database technology consists in secondary storage management and in taking advantage of indexes. Stream queries are typically applied to main memory data and indexes are not available a-priori. Data management in a web context presents other particularities that we mention next.

First, web data is by nature distributed and heterogeneous. This is in the spirit of distributed databases or more precisely in the spirit of federated databases that stress the independence of data sources that are components of such systems. A main new aspect when considering, for instance, data integration on the web, is the number of sources (tens, hundreds, thousands). Moreover, federated databases typically consider environments where the semantics of data in sources of interest is well understood. This assumption does not scale to the web. We have to be able to discover new sources dynamically and analyze them to understand their semantics on the fly. This clearly complicates monitoring that now has to include a significant artificial intelligence component.

Last but not least, another difference (say, to federated databases) is that the web does not provide for change control on data we are interested in. For HTML pages, it is possible to register with services providing notifications when pages of interest change. This is clearly a very primitive form of change control compared to the sophisticate triggering mechanisms supported by traditional database systems. One can measure the limitations of such approaches when one knows that this is typically implemented by regularly (once a day, a week, a month, more?) reading the page to check whether it has changed and inspecting some hash value of the text of the page.

In the remainder of the paper, we consider applications that deal with web monitoring and mention some technical issues they raise.

4 Electronic commerce

One of the most popular applications concerns comparative shopping. Some system already provide unique entry points to collections of catalogues. The issue here is the integration of information from the various catalogues. One main difficulty is to "wrap" the various sources to extract their information (nature of the product, price, terms of delivery, etc.) Monitoring is another major issue. Information changes at the sources. In some cases, changes may be so rapid (e.g. prices for air travel) that one prefers to use a "mediator" approach. So, instead of extracting the information and monitoring changes, one asks queries each time a user is looking for information. In many cases still, a warehouse approach with data extracted, stored and monitored is used.

Automatic site wrapping and monitoring The wrapping of data sources is mostly performed today semi-automatically from HTML pages. An engineer examines a collection of pages with similar structure and tailors a program to extract information from it. The weakness of this approach is that it has to remain limited to a reasonably small number of sites and page styles. Also, when the pages layout is modified in a site, the wrapper has to be modified. With XML, the situation is somewhat easier and it is possible to use the information contained in tags to automatically understand the semantics of pages and thus wrap them automatically. When one moves to forms or web services, the situation becomes more intricate since one needs to find which data to input in order to obtain information. Intuitively, an automatic system would have to apply a heavy dose of artificial intelligence to understand the semantics of a web site and automatically extract information from it. Now let us assume that the owner of a site is willing to let others use the information. Then the site owner may "publish" the semantics of the site (e.g., in RDF [14]) and the means to obtain information from it using standards such as WSDL [17] and UDDI [16]. Indeed, one may even expect that entire domains will standardize the way data is published on the web, e.g., NewsML for the news industry. This is raising the issue of the semantic web and is mentioned here primarily to stress that an intelligent monitoring of the web should rely on semantics published by web sites, which seems to take us a bit far into the future.

5 Web archiving

The content of the web is an increasingly valuable source of information that is worth being archived. Indeed, there are attempts to archive the entire web (Internet web archive [12]) or portions of it. For instance, we are studying with the Bibliothèque Nationale de France [9] the archiving of the French web [6]. This would be a service to future generations of historians, somewhat like libraries for todays historians. Even the viewpoint that the web is a huge junk yard suggests the need for archiving it when considering the importance of prehistoric junk yards for todays archaeologists. Companies may also want to archive their web (private or public) although the motivation is different here: the "memory" of the enterprise is an important asset. Finally, one may want to archive a web site on some specific topics as part of the service provided by the site (e.g., newspaper websites).

Better web crawlers Because of the lack of change control on the web, one issue in constructing web archives is to decide when to version a particular page. The crawlers have to visit the web and detect changes. They must do so by taking advantage of their network bandwidth. Crawling is in principle a trivial process: start from some page in the "core" of the web and follow links to discover new pages. There are subtleties such as avoiding rapid firing, i.e., requesting too rapidly too many pages from the same web server, an action considered unfriendly. There is also the technical challenge of managing billions of URLs. But most importantly the real problems come from the limitations of network-bandwidth compared to the size of the web.

> Suppose 4 million pages are read per day with a single PC (the rate of Xyleme crawlers for instance); with 25 crawlers, 100 million pages can

be read per day, which already requires a very large network bandwidth. Still, it takes a month to read 3 billion pages. By that time a lot of the pages that have been read are already stale and reaction to changes is too slow. By accessing once every month the pages of, say the New York Time, a lot of important information is missed.

To overcome this problem, one should use the network bandwidth more intelligently. In particular, one should not access all pages with the same frequency. A solution should distinguish between important pages (e.g., pages with high page rank in Google terminology [13] that are read regularly) and less or unimportant pages. A solution should also take into account some estimate of the change frequency of the page (it is not interesting to keep reading an archive) and, last but not least, it should be tailored to the specific goal of the archive.

There are many other issues in web archiving. For instance, web archives have motivated a new research on versioning mechanisms [8]. The data we archive is a rapidly changing graph and this raises complex issues. To illustrate, suppose that we take some first-generation crawler to create the archive. Say the crawler uses some breadth-first traversal of the graph of the web. Then a large time interval may occur between the visit of a page and the visit of the pages it points to. These pages are very likely to have changed or may even have disappeared by the time we access them. This is a very important cause of inconsistencies in the archive. Finally, perhaps the most critical technical challenge for such applications is to be able to also archive data from the deep web.

6 Web surveillance

Web surveillance can be applied in a variety of fields ranging from anti-criminal or anti-terrorist intelligence (e.g., detecting suspicious acquisition of technology) to business intelligence (e.g., discovering potential customers, partners or competitors). Such applications combine web monitoring with data mining and text (or sound or image) understanding. The two main issues from a monitoring viewpoint are (i) to obtain the data (using better web crawlers that were already discussed) and (ii) to filter it (using new query processing technology).

Stream and continuous queries One can distinguish two main approaches:

1. Continuous queries: the same query is evaluated regularly by a query engine and the difference between the answers is analyzed. In other words, we separate data acquisition (crawling and indexing) and query processing; see for instance [5].
2. Stream queries: streams of data are brought from the web (typically from web robots) and these streams of data are filtered; see, e.g., [7].

Typically, stream queries are more appropriate for fast detection of simple patterns on rapidly changing data (e.g., data that is brought in by sensors) whereas continuous queries are more feasible when considering the detection of complex patterns, e.g., patterns involving joins of data from several sources. In this context, we need to develop new optimization techniques for continuous and stream queries. Also, there is a need for a query language to specify such queries since SQL simply misses the points: It does not provide the necessary constructs for querying streams or specifying continuous queries. The query language should allow to specify a number of aspects such as the pattern that one would like to detect, on what data should the query be performed, how often, what should be done with results (should they be posted on the web, sent by email, how often, etc.) A classical query is in some sense very simple: it is a function that returns a result. A continuous or a stream query is more complex in the sense that it creates in some way a contract between the client and the server, it takes time into account in an essential manner, and also some notion of state of the client (e.g., when we compare the last result of a continuous query to the current one).

7 Mobile devices

Mobile devices are certainly becoming more and more popular, e.g. laptops, PDAs, cellular phones, embedded devices, etc. This leads to a vision of lots of data changing rapidly (position of PDA's) and rather volatile (plugging and unplugging laptops). Perhaps a most critical aspect is geography. For instance, a query such as "where is this particular movie showing" should get a different answer depending on the location of the person asking the query. Indeed, the location and the nature of device used are key components of the user's profile that should be taken into account in answering queries.

These mobile devices should be considered, as participating to the web, e.g., by publishing information on it. The rapidly evolving and massive nature of the web is thus amplified. Furthermore, monitoring is particularly critical in a mobile context, e.g., if I want to be alerted when some cultural events happen nearby or whether a friend happens to be in the neighborhood.

To conclude this quick tour of the topic, we consider active web sites.

8 Active web sites

In the previous discussion, we somewhat assumed that web sites were "passive" and that some servers were in charge of the surveillance. This is mostly the situation today. However, one should expect the future web sites to include more and more active features (in the style of active databases) that will facilitate friendly web surveillance. In particular, mobile devices should be viewed as peers willing to share data resources with other peers.

Bringing active database features to the web The most important concept in active databases is that of triggers. One would like web sites to be friendly enough to let us install triggers in, say XML documents they manage; see [4]. Suppose I am interested in the price of a particular camera in the Amazone catalogue. I would like to introduce the trigger (specified in an ad-hoc syntax):

When the price of the Cannon CAN344 changes do email-me

More generally, one would like to have web peers that are willing to cooperate using (continuous) methods and offering services such as change notification. Research in that direction is certainly needed; see Active XML [2] for a proposal in this spirit.

We are still a long way from truely active web sites. However, web services already provide the technology for enabling more active web sites. To illustrate, consider a bibliography database. It is straightforward to implement a web interface for it using, for instance, SOAP services. A service for notifying the addition of new bibliography references can be easily proposed by combining a database trigger with a call to a SOAP service. Similarly, a service for sending (say once a week) the incremental changes to the database may be easily implemented. What is perhaps more critically missing is a global view of the distributed information system this is leading to. In particular, there are challenging semantics and optimization issues.

9 Conclusion

Let us try to abstract a few main ideas from the applications we considered. The high-level vision we sketched is that the web is moving from a collection of static documents to a world of dynamic knowledge. The knowledge may range from implicit (buried in text or other unstructured media), to somewhat explicit via semantic annotations in the form of XML tags attached to data, to very explicit using future standards such as RDF. One should expect to see more and more semantics specified explicitly on the web, because this is the only feasible path if we are to make effective use of this massive amount of data. In this vision of the web as a world-wide knowledge base, an essential aspect is that the knowledge changes in time. Web knowledge has to be monitored to make available a timely version of information.

Acknowledgment We would like to thank all the people involved in the Xyleme project [11] and the Xyleme company (www.xyleme.com). We would also like to thank those who work on Active XML and in particular, Omar Benjelloun, Angela Bonifati, Ioana Manolescu, Tova Milo, and those who discussed with me the French web archive, Gregory Cobena and Julien Massanes. Finally, we thank Victor Vianu for his comments on a first draft of this paper.

References

1. S. Abiteboul, P. Buneman, and D. Suciu. Data on the Web. Morgan Kauffman, 1999.
2. Serge Abiteboul, Omar Benjelloun, Tova Milo. Towards a Flexible Model for Data and Web Services Integration, proc. Internat. Workshop on Foundations of Models and Languages for Data and Objects, 2001.

3. Brian Babcock; Shivnath Babu; Mayur Datar; Rajeev Motwani; Jennifer Widom. Models and Issues in Data Stream Systems, PODS, 2002.
4. Angela Bonifati, Daniele Braga, Alessandro Campi, Stefano Ceri: Active XQuery. ICDE 2002
5. Jianjun Chen, David J. DeWitt, Feng Tian, Yuan Wang: NiagaraCQ: A Scalable Continuous Query System for Internet Databases. SIGMOD Conference 2000: 379-390
6. S. Abiteboul, G. Cobéna, J. Masanes, G. Sedrati, A First Experience in Archiving the French Web, to appear in ECDL, 2002.
7. Benjamin Nguyen, Serge Abiteboul, Grégory Cobena, Mihai Preda, Monitoring XML data on the Web, SIGMOD, 2001.
8. Amelie Marian, Serge Abiteboul, Grégory Cobena, Laurent Mignet, Change-Centric Management of Versions in an XML Warehouse, VLDB, 2001.
9. Julien Masanès, The BnF project for Web archiving, Preserving online content for future generation, ECDL Workshop, 2001.
10. Raghavan, S. and H. Garcia-Molina. Crawling the Hidden Web. in 27th International Conference on Very Large Data Bases. 2001. Rome, Italy.
11. Lucie Xyleme (a nickname for the people who worked on the Xyleme project). A dynamic warehouse for XML Data of the Web. IEEE Data Engineering Bulletin 24(2): 40-47 (2001)
12. The Internet Archive, www.archive.org
13. Google, www.google.com
14. Resource Description Framework, www.w3.org/RDF
15. Simple Object Access Protocol (SOAP) 1.1, http://www.w3.org/TR/SOAP/
16. Universal Description, Discovery and Integration of Businesses for the Web, www.uddi.org
17. Web Service Definition Language, www.w3.org/TR/wsdl
18. XQuery 1.0: An XML Query Language, http://www.w3.org/TR/xquery
19. Extensible Markup Language (XML), www.w3.org/XML
20. Xyleme SA, www.xyleme.com
21. The World Wide Web Consortium, www.w3.org