Managing Broken URLs in Federated Metadata*

Tien-Dung Le and Elena Shulman

European Schoolnet, Rue de Trèves 61, B-1040 Brussels, Belgium.
{dung.le,elena.shulman}@eun.org

Abstract. In a large federation of learning object repositories, learning object locations or learning object URLs in metadata can become out of date when learning objects are moved or deleted. Regular checking of learning object URLs is essential. However, fully checking across a large federation imposes an unsustainable burden on time and resources while negatively impacting networks and repositories. This paper describes a broken URL handling system with a heuristic model that can provide a sustainable solution for federation service managers while enhancing communication and collaboration among federation stakeholders.

Key words: Broken URLs, federated metadata, heuristic model, learning resource exchange

1 Introduction

The Learning Resource Exchange (LRE) is a service that allows European teachers to get access to digital educational content from many different countries and providers (Figure 1). Content providers produce metadata, i.e., machine-readable descriptions of the educational content they want to make available to teachers within and beyond their national learning object repository systems. The LRE provides unified access to Learning Objects (LOs) stored in these different repositories. Each content provider exposes their metadata so that it can be easily accessed by the LRE. The LRE collects metadata from the different content providers and compiles them to produce a digital catalog of learning resources that can be consulted by teachers using the LRE [4]. Along with other information relevant to pedagogical contexts, metadata contains the locations (i.e., Uniform Resource Locators - URLs [1]) where resources can be obtained. Typically, users of the LRE catalog can obtain learning resource of potential interest by following the URL provided with the resource description. If this URL does not point to the expected resource, it is said to be "broken". Although broken URLs can sometimes be the products of transient network or server problems, in most of the cases they are caused by outdated metadata.

* The work presented in this paper is partially supported by the European Community eContent*plus* programme - project ASPECT: Adopting Standards and Specifications for Educational Content (Grant agreement number ECP-2007-EDU-417008). The authors are solely responsible for the content of this paper. It does not represent the opinion of the European Community and the European Community is not responsible for any use that might be made of information contained therein.

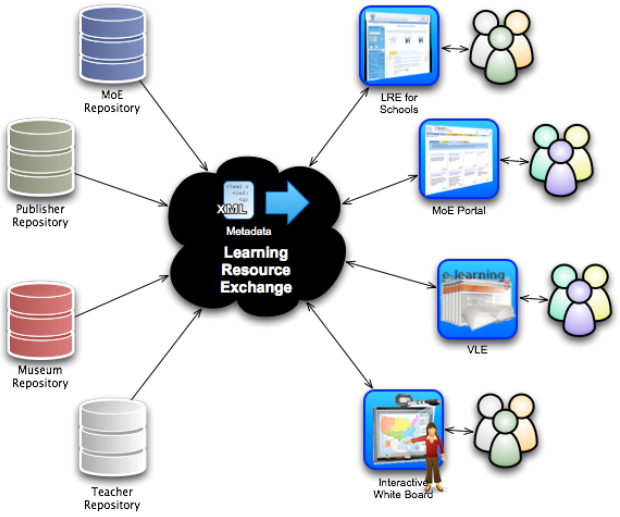

Fig. 1. Learning Resource Exchange.

This paper presents the results of our efforts to develop a solution that will effectively detect broken URLs, automate a system for communicating with content providers and allow greater flexibility for LRE service managers in resolving issues that can negatively impact user experience with the LRE.

Section 2 presents the background to the problem. Section 3 presents the architecture of the broken URLs handling system and details our proposed heuristic algorithm to automatically detect the broken URLs. Section 4 details the guidelines developed to facilitate effective and expeditious communication with content providers and procedures for LRE managers and system responses depending on the status of broken URLs once reported. Section 5 is a discussion including comments on some elements of comparison with related work.

2 Background

Broken URLs are problematic on several levels for a service that manages a large federation of repositories. Unsurprisingly, broken URLs significantly impact user satisfaction with the system. Teachers have reported a loss of confidence in the LREs catalog and perceived quality of the LRE more broadly [6]. From the users point of view, broken URLs are also a source of frustration when teachers attempt to access resources they have previously placed in a favorites folder or when they recommend to colleagues resources that have become suddenly unavailable. As depicted in Figure 1, the LRE federates LO repositories from various origins. LRE content is provided by ministries of education (MoE), commercial and non-profit content providers (Publisher), and cultural heritage organizations (Museums). Some of these repositories might store user-generated

content (Teachers). Because the LRE does not host the objects or control access to objects directly, the LRE must rely on the timely collaboration of content providers depicted in Figure 1 to update their metadata whenever object locations have changed.

Experience has shown that broken URLs can become a significant problem. Part of the solution we propose is based on a heuristic algorithm developed and tested for this purpose. This paper also illustrates the way we have linked a solution for broken URLs detection to support communication and collaboration with content providers.

The LRE has a number of techniques in place intended to ameliorate the issue with metadata quality but none has provided a sustainable solution in a system reliant on coordinated actions in a growing federation. First, to keep the LRE catalog up-to-date, metadata is collected on regular basis (e.g., everyday). This technique ensures that updates to the metadata on the part of content providers are reflected in the main catalog. However, this requires content providers to maintain the accuracy of their own metadata. We have found that this technique has its own set of drawbacks because some providers do not use the metadata they produce for the LRE. Another technique relies on users' scrutiny. Teachers who find broken URLs are invited to report them.

Reported URLs are checked by the system and, if the problem is confirmed, the corresponding entries are removed from the catalog and their content providers receive a request to correct the problem. The main drawback of this technique comes from the fact that it is a highly individualized solution impacting the quality of a few records at a time. Moreover, this technique is fully reliant on the goodwill of the catalog users and does not ultimately shield them from unsatisfactory search and retrieval experience culminating in the discovery of broken URLs.

A third technique consists of systematically checking all the URLs of the catalog on a regular basis. Unfortunately, this solution does not scale. As the number of catalog entries grows, the time necessary to check the catalog fully can be prohibitively extensive in duration. For example, in the LRE a full check of more than 200,000 URLs takes more than two days. To make matters worse, systematically checking all the learning resource URLs available on a system can sometime be considered as "unfriendly" by their administrators.

3 Broken URL Handling System

While allowing the LRE to efficiently detect and make decisions to remove catalog entries with broken URLs from the search, the proposed solution is ultimately intended to support and automate more effective communication with content providers and to facilitate their collaboration in expeditiously updating their metadata. The handling system (Figure 2) and guidelines described in this article accomplish several objectives. They demonstrate the feasibility of a heuristic model in efficiently detecting broken URLs and propose a mechanism to trigger ameliorative actions from providers. They illustrate our efforts at establishing

clear guidelines to support decision making and communication with providers for LRE service managers responsible for metadata quality and user satisfaction. A component to detect and hide broken URLs is only part of the solution. The proposed solution also takes into account and avoids broken URL detection techniques that can appear to be denial of service attacks on partner systems.

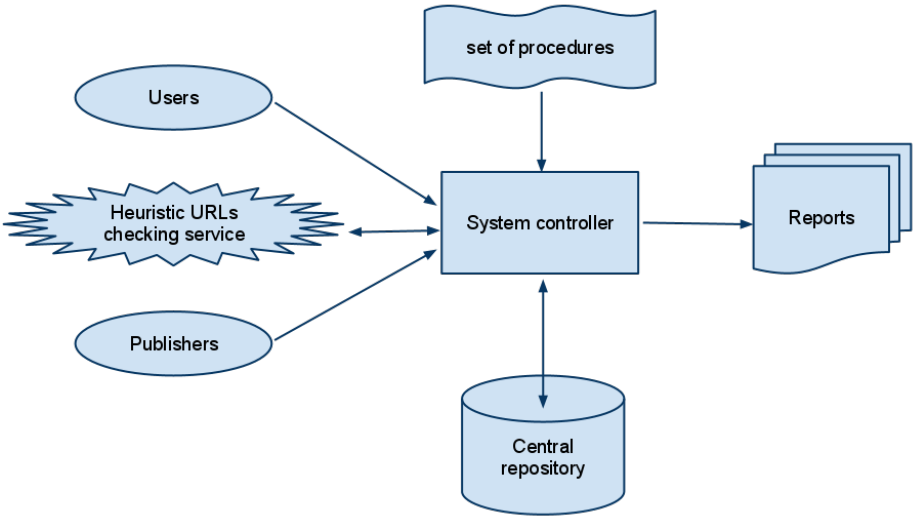

Fig. 2. Broken URLs Handling System.

There are several steps in the detection and handling of broken URLs. The first stage was a full system check for broken URLs to determine the scale of the problem and to create a baseline to test the proposed algorithm. As we discussed, while this kind of check can produce useful data on URLs, it is not feasible to use this technique on a regular basis for reasons already mentioned. The second stage is our proposed long term solution to detect and trigger corrections for bad URLs based on a check of smaller subsets and provide better automated channels for communicating with content providers. Techniques for determining the size and location of the sets to be checked are described below.

As mentioned above, it is necessary to put in place a heuristic checking algorithm to regularly check URLs of the catalog but not to "attack" the systems hosting LOs. Therefore only a subset of URLs should be checked at a time. Our analysis of the LO location distribution shows the URL selection should be based on the LO domain name system (or domain for short). For each domain, only a subset of its URLs should be checked. The first step is to determine which domains should be targeted for checks. To determine if the system stops or continues checking URLs in a domain, a sampling plan is applied after all selected URLs are checked.

3.1 Learning Object Location Distribution

Learning objects in the LRE federation are hosted in one or several domains. Typically, a URL or a LO location refers only to one domain while a domain normally hosts more than one LO. We analyzed the relationship between LOs and domains based on 2 aspects. First we looked at the domain distribution in the catalog, which indicates how many domains have a certain number of LOs. Second we analyzed the distribution of LOs to determine how LOs are distributed among domains.

In June 2010, there were 842 domains in the LRE. Figure 3 shows the domain distribution and Figure 4 shows the LO distribution. A significantly large percentage of domains (60%) hosted just a single learning object. Even more important for the development of our solution was the finding that only 4% of domains hosted more than 1000 LOs and 70% of LOs are located on domains presently hosting over 10,000 LOs. The fact that a small number of domains are hosting 96% of all LOs was instrumental in building a heuristic checking algorithm.

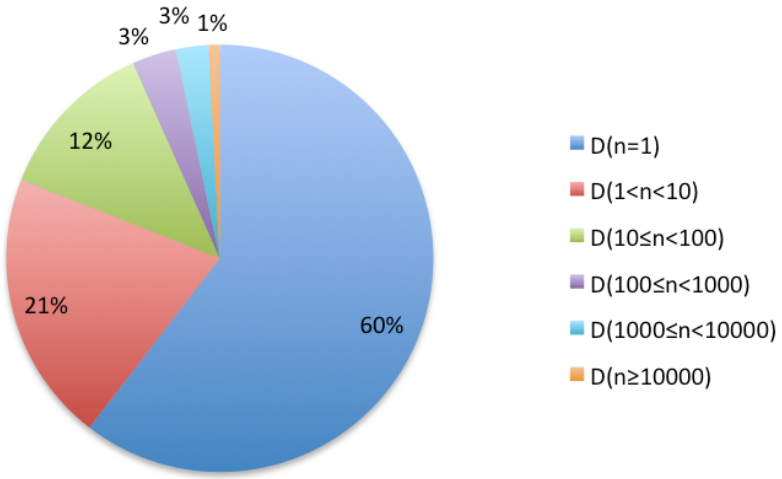

Fig. 3. Domain distribution in the LRE.

3.2 Heuristic Checking Algorithm

Based on the analysis of LO distributions, we propose to check each domain separately. For each domain, two checking steps are applied. First, the system checks all reported-broken URLs, which are reported as broken URLs in the last check, and calculates the confidence of the system on the domain. The second step in the check applies a sampling plan to check all previously good URLs. The sampling plan (Figure 5) is a simplified solution of the sequential sampling plan [7].

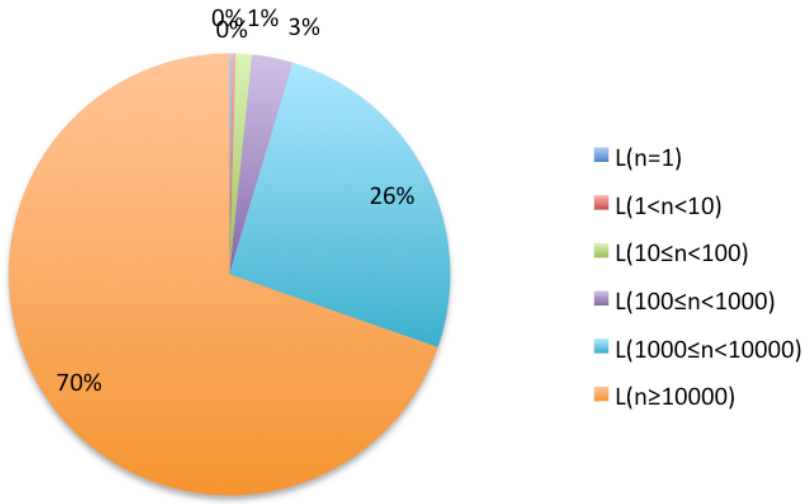

Fig. 4. LO location distribution in the LRE.

Sampling Plan

- Single sampling plan
  One sample of URLs on a domain, such as 10% of URLs, is selected at random. In case there are more than p_2 of selected URLs, such as 90%, are good, all unchecked URLs are considered as good URLs. Otherwise, the whole URLs on the domain are checked.
- Sequential sampling plan
  A set of URLs in a domain, for instance a set of 100 URLs, is selected at a time and after inspection a decision is made to accept (stop checking) or reject (check all URLs) in the domain or select another set in the same domain (continue sampling). In general, this sequential sampling allows for quick decisions, especially when frequency of broken URLs in a domain is particularly high or particularly low.
- Sampling plan in the algorithm
  Because we recognize the difficulties in calculating the slope of accept and reject lines, the checking algorithm uses sequential sampling plan with a simplification - the slope is zero (Figure 5).

Sampling Plan's Parameters

- Pre-defined threshold p_1 presents the low confidence of the checking system on the domain. $p = p_1$ is the reject line in the sampling plan.
- Two pre-defined thresholds p_2^- and p_2^+ present the low-value confidence and the high-value confidence of the checking system on the domain .

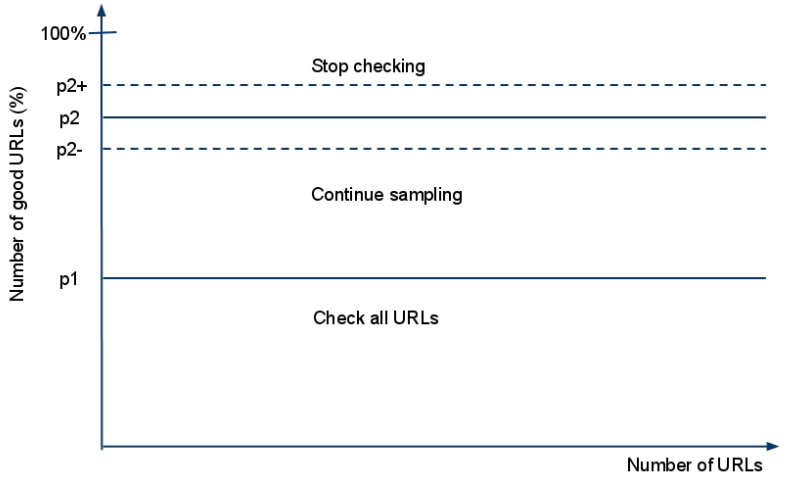

Fig. 5. Sampling plan.

– N_{pb} is the number of broken URLs in the previous check, N_{psb} is the number of stay-broken URLs in the current check, the rate r is calculated as follow

$$r = N_{psb}/N_{pb} \tag{1}$$

– A threshold p_2 presents the confidence of the checking system on the domain. p_2 is calculated as follow

$$p_2 = p_2^- + r * (p_2^+ - p_2^-) \tag{2}$$

$p = p_2$ is the accept line in the sampling plan.

Sampling Plan Procedure

– A group G of URLs is selected at a time for checking. Assume that N_c is the total number of checked URLs in the current check, N_g is the total number of good URLs, p is the percentage of good URLs calculated by

$$p = N_g/N_c \tag{3}$$

– If $p < p_1$ then the system checks all URLs of the set. If $p_2 \leq p$ then the system assumes that all the other URLs on the domain are good and stops checking. Otherwise, the system continues sampling and checking (Figure 5). Algorithm 1 shows the detail of the checking technique.

Algorithm 1 Heuristic checking algorithm

```
1: procedure CHECK
2:     for all domain ∈ domains do
3:         Re-check all broken URLs on domain
4:         Calculate p2 using equation 2
5:         sampling = true
6:         while (sampling and (domain has unchecked URLs)) do
7:             Take a group of URLs for checking
8:             Check this group
9:             Update p using equation 3
10:            if p < p1 then
11:                Check all other URLs
12:                sampling = false
13:            else
14:                if p2 ≤ p then
15:                    sampling = false        ▷ assume that all other URLs are good
16:                end if
17:            end if
18:        end while
19:    end for
20: end procedure
```

3.3 Experiment

To compare our algorithm with the full system check, we first executed a full system check to detect all broken URLs in the LRE, then we executed the heuristic checking algorithm 3 times in simulation mode with the findings from the full system checking results for comparison.

In June 2010, the total number of URLs in the LRE was 236763, among them 45711 (19,31%) URLs were broken[1]. The first run was executed with an assumption that all URLs are good, while the second run and the third run used knowledge from the previous run. The following parameters were used: $G = 100, p_1 = 50\%, p_2^- = 90\%, p_2^+ = 95\%$.

The first run only checked 17,36% of URLs but was able to detect 73,48% of broken URLs. The second and the third checks detected 94,92% and 98,67% of broken URLs. Overall, after 3 runs, the heuristic check algorithm only checked 36,78% of all URLs on average but detected 98,67% of broken URLs.

4 Procedures to Correct Broken URLs

Given that the LRE federates LO repositories from various origins, effective handling broken URLs depends simultaneously on the heuristic checking algorithm that can be deployed to test discrete domains and automation of communication channels with content providers. Therefore, we established a schedule for check

[1] The experiment took place on June 23rd, 2010.

Table 1. Results

Run	Selected URLs (in number and in percentage of the total number of URLs)	Broken URLs (in number and in percentage of the total number of broken URLs)	Rate (number of broken URLs /number of selected URLs)
1	41102 = 17,36%	33587 = 73,48%	81,72%
2	105126 = 44,40%	43388 = 94,92%	41,27%
3	115005 = 48,57%	45101 = 98,67%	39,22%
Overall	710289 = 36,78% (in average)	45101 = 98,67% (max)	54,07% (in average)

sequences and conditions under which LRE service managers could act on the findings of the checking algorithm, initiate the removal of records from search results, set in motion further checks and determine when it was appropriate to return records to the search. Making these conditions explicit allows for better management of metadata quality in the LRE, shields users from negative experiences during search and retrieval and opens opportunities to collaborate with content providers in improving the quality of their own metadata.

The first step in this process begins when a broken link is first found and the LRE system generates an automated notification for providers. System controller coordinates all other components and generates reports. These machine readable reports are set to trigger a fix and can be exposed to providers using RSS, ATOM feeds or downloadable excel sheets depending on the preferences of the providers. After seven days, the LRE system is set to recheck the problem URLs. If the problem persists and there has been no other response from the provider, an automated email message is generated for the LRE system manager notifying them of URL's status, providing relevant details about the record(s) in question and contact information for the provider. Using this data, the LRE manager sends a personal email to the repository manager. The repository manager will be notified that the problem has not been corrected despite an automated report. The guidelines stipulate that after three weeks records with broken URLs will be removed from the search results. At this point we hope to begin a dialog with the repository manager, determine the source of the problem, offer advice or other assistance if needed and provide feedback more generally to repository administrators. In order to mitigate the detrimental effects of broken URL on user experience, we have instituted a time limit for how long records with broken URLs having been identified and reported to providers can remain in the LRE search. One month after the initial discovery of the broken URL, the system will initiate another check of the domain using the algorithm described above. If the problem has not been corrected, records with broken URLs will be removed from search results. This action is accompanied by an automated notification sent to the provider listing items that are no longer in the search results and the reason for their removal. Once records have been removed from the search, a

Fibonacci sequence is implemented to determine if and when the problem URLs are corrected by providers. When the check sequence indicates that the broken URL has been corrected, the records will be returned to the search results. An automated notification to providers, both machine readable and as an email, will be generated to alert the providers of their record(s)' change in status. This ability to return the records to the search takes into account scenarios in which providers are not able to respond expeditiously to reports of broken URLs without unduly impacting the quality of LREs services.

The guidelines described above take into account the need to provide flexibility of responses for LRE service managers and opportunities to improve collaboration between system administrators within the federation.

5 Discussion and Related Works

Although, recognition of the problems and attempts to grapple with the challenges of broken URLs or "link rot" are not new, most of existing approaches to automatically detect broken URLs are based on the relationship between resources or pages [2],[8], [12] which is not applicable in a LO federation where there is almost no link between two learning objects locations.

There are also some methods such as [9], [11], [3] to assure that the links are always available. These methods could be divided into two categories. The first approach involves creating copies and keeping resources in a local repository [9], [11]. This approach assumes that the LOs will not be modified or requires a check of the links on the fly to determine if and when the system needs to switch the links.

The second approach uses a persistent identifier service to resolve the URLs [3]. However, this approach only works well if such service is available for content providers sites, which is not the case in the LRE.

The proposed mechanism to manage broken URLs allows LRE managers to control the quality of LO metadata in a federation. It combines different ways to detect broken URLs and report problems to the content providers. Because the heuristic checking algorithm is able to detect broken URLs without blocking network communication it can be implemented to run in a sequence and schedule intended to provide detection, reports and for corrective actions either on the part of providers or, if necessary, on the part of LRE managers.

6 Conclusion

In this paper, we have presented a framework with a heuristic checking algorithm that allows for the correction of broken URLs in federated metadata. Future work will focus on further enhancements of this framework to detect broken URLs when metadata arrives in the LRE. This future work will involve the development of a filter to remove broken URLs from metadata collected via the Open Archives Initiative Protocol for Metadata Harvesting OAI-PMH [5] or the Simple Publishing Interface SPI [10].

References

1. Berners-Lee, T., Masinter, L., McCahill, M.: Uniform Resource Locators (URL) (RFC 1738). Network Working Group (1994)
2. Ingham, D., Caughey, S., Little, M.: Fixing the "Broken-Link" Problem: The W3Objects Approach. In: Computer Networks and ISDN Systems, 28, pp. 1255-1268 (1996)
3. Kahn, R., Wilensky, R.: A Framework for Distributed Digital Object Services. In: International Journal on Digital Libraries, 6, pp.115-123 (2006)
4. Massart, D.: Towards a Pan-European Learning Resource Exchange Infrastructure. In: Feldman, Y., Kraft, D., Kuflik, T. (eds.) NGITS'2009, LNCS, vol. 5831, pp. 121-132. Springer, Haifa, Israel (2009)
5. Lagoze, C., Sompel, H. V.: The Open Archives Initiative Protocol for Metadata Harvesting http://www.openarchives.org/OAI/openarchivesprotocol.htm (2001)
6. MELT: Final Evaluation Report, D 7.3. http://info.melt-project.eu/shared/data/melt/MELT_D7_3_Final_Evaluation_Report.pdf (2009)
7. NIST/SEMATECH: e-Handbook of Statistical Methods. http://www.itl.nist.gov/div898/handbook (2010)
8. Popitsch, N., Haslhofer, B.: DSNotify: Handling Broken Links in the Web of Data. In: Proceedings of the 19th international WWW conference (WWW2010), NC, USA (2010)
9. Reich, V., Rosenthal, D.: LOCKSS: A Permanent Web Publishing and Access System. D-Lib Magazine, vol. 7 (2001)
10. Ternier, S.: Interim Report: The Simple Publishing Interface Specification. http://ariadne.cs.kuleuven.be/lomi/images/b/ba/CEN_SPI_interim_report.pdf (2009)
11. Veiga, L., Ferreira, P.: RepWeb: Replicated Web With Referential Integrity. In: Proceedings of the 2003 ACM Symposium on Applied Computing, pp. 1206-1211, New York, NY, USA (2003)
12. Wiley, G., Thomas, W.: Improving OpenURL Metadata. Serials Librarian, vol. 56, pp. 282-286 (2009)