Augmenting the Web of Data using Referers

Hannes Mühleisen
Freie Universität Berlin
Networked Information Systems Group
Königin-Luise-Str. 24/26
14195 Berlin, Germany
muehleis@inf.fu-berlin.de

Anja Jentzsch
Freie Universität Berlin
Web-based Systems Group
Garystr. 21
14195 Berlin, Germany
mail@anjajentzsch.de

ABSTRACT

Linked Data relies on one central concept: Typed links connect entities stored within data sets published by different individuals. Manual input and mapping are common techniques to create these links. We propose a novel method, where HTTP Referer information is used to create new links between Linked Data entities stored in different data sets. We evaluate our method using 27.86 million real-world log entries from web servers hosting Linked Data.

1. INTRODUCTION

On the ever growing Web of Data there is not only a relevant overlap of data on the same real world concepts but also a growing number of entities that are related to each other. Since 2007, the number of data sets on the Web of Data doubles every few months[1]. Data providers have to constantly keep up with the growth of the Web of Data and new linking possibilities. We contribute to this development by providing a novel way for Linked Data publishers to find new and suitable link targets.

The Web of Data forms a single global data space for the very reason that its data sources are connected by links. However, as the current state of the Linked Open Data cloud shows, most data sources are not sufficiently interlinked, with over 50% of them only being interlinked with only one or two other data sources[2]. Almost two thirds of the data sources do not link back to all the data sources they are linked from. This leads to a weakly interlinked and often unidirectional graph of Linked Data which impedes applications relying on link traversal. In addition, for the integration of data duplicate detection and linkage recording are crucial preliminaries. While some fully automatic tools for link discovery do exist [6], most tools generate the links semi-automatically based on user-defined link specifications [11, 12, 10].

In this paper, we propose a novel – fully automatic – approach for back link generation. Here, the "Referer" request header defined by the HTTP protocol specification is used to discover remote documents containing Linked Data entities linking to local entities. Since RDF links between entities are typed, the type of a back link depends on the type of the "forward" link. In order to generate correctly typed back links, the remote document URLs are dereferenced, and the retrieved document analyzed. Documents are searched for the URL of local entities. Matches are then used to determine or create the semantically correct link property to be used for a back link. Using the local and remote entity URLs along with the back link property, new back links can be created and inserted into the data set. We present a fully deterministic algorithm for this back link generation approach.

The remainder of this paper is structured as follows: Section 2 details our approach and algorithm for automatic back link generation for Linked Data sets. Section 3 describes the evaluation of our approach using 27.86 million log entries from web servers hosting Linked Data. Section 4 gives a short overview over related approaches for link generation, and finally Section 5 concludes the findings of this paper and gives pointers to areas of future work.

2. LINK GENERATION APPROACH

According to the Linked Data principles, links between entities contained in different data sets and stored on different servers are an integral part of Linked Data [3]. These links into other data sets are often used to provide background information, or lead to other related entities. Apart from using central databases such as *Sindice*[13], many Linked Data tools and applications are dependent on considerable quantities of links. For example, the *SQUIN* SPARQL query processor uses link traversal to resolve patterns within a query [8].

Within the *Resource Description Format (RDF)* data model, links are directed and have link semantics specified; each link is required to be labeled by a machine-readable URI. Data sets are independent in management and storage, and links between entities are not a part of any meta-level central system, but reside in the data set they were created in. Figure 1 shows this principle: Two entities, `ds1:res1` and `ds2:res2` are linked from `ds2:res2` to `ds1:res1` using the link type `ex:p1` (1). The back link from `ds1:res1` to `ds2:res2` is not required to be present, its link type is also unknown a priori. (2) shows the physical storage of the entities and the link, *Dataset 1* contains the entity `ds1:res1`, and *Dataset 2* contains both the entity `ds2:res2` as well as the link. Should a link-traversal based tool encounter `ds1:res1`, it would have no way of reaching `ds2:res2` without the help of central databases.

[1] `http://lod-cloud.net`
[2] `http://lod-cloud.net/state`

LDOW2011, March 29, 2011, Hyderabad, India.

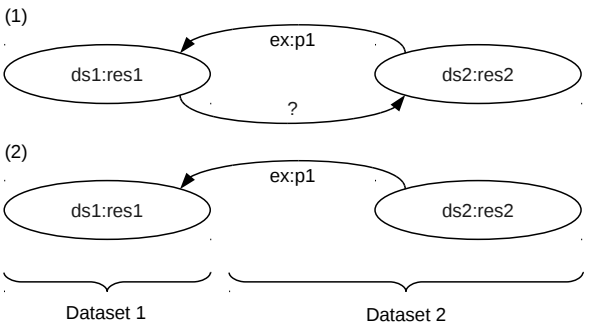

Figure 1: Linked Data Links and Storage Locations

Links between different data sets cannot – so far – be created automatically without complex entity recognition schemes or data structure conventions. Thus, link creation is often based on human interaction, which represents a tedious process and is only practicable between two different data sets at a time. An automatic or supporting process for link generation would be desirable, even if only a subset of possible links can be discovered. In the "classic" WWW, links are often created on the basis of a link exchange; web authors communicate the intent of linking to each other's sites, a process that can be beneficial for both sites and their visitors. The amount of links is kept low as not to distract readers. For Linked Data entities however, a large amount of links to other entities is not disruptive for its usage, as these entities are mainly published for use by computer programs. Hence, as content is machine-readable, link exchange can be performed automatically.

The Linked Data specification defines the *Hypertext Transfer Protocol (HTTP)* as underlying data exchange protocol. Linked Data entities are thus requested and served using this protocol. The HTTP specification defines the *Referer*[3] header field as part of HTTP requests [7]. This field can be set by the user agent program to the URL of the site that it was referred from.

> "The Referer[sic] request-header field allows the client to specify, for the server's benefit, the address (URI) of the resource from which the Request-URI was obtained[...] The Referer request-header allows a server to **generate lists of back links to resources** for interest, logging, optimized caching, etc." [7, sec. 14.36]

The value of the Referer header is commonly added to request log files by standard web servers, for example by the Apache HTTP Server. For human-only web sites, the Referer values are currently mainly analyzed to track visitor sources such as search engine queries. In the case of Linked Data, the highlighted part of the Referer definition is more relevant: If RDF crawlers and user agents would correctly set this field, a program could generate back links to local entities from the web server's log files and increase the overall connectivity of the Linked Data cloud.

If Referer information are to be used to create links between RDF entities, the link property URI has to be determined first, as RDF does not allow untyped links. For very generic cases, the *RDF Schema (RDFS)* specification defines the `rdfs:seeAlso` property, which "indicates a entity that might provide additional information" [4]. However, the Linked Data specification allows the retrieval of remote entities ("dereferencing") in order to gain more information about that entity. The dereferenced remote RDF document can then be processed into RDF statements, possibly yielding the link property that was used to refer to a local entity. Reconsider the situation depicted in Figure 1, if a Referer value of `ds2:res2` is logged for an HTTP request to the server hosting `ds1:res1` as part of *Dataset 1*, an automatic process can retrieve the document describing `ds2:res2` to determine the property value of the link pointing to `ds1:res1`, in this case `ex:p1`.

One of the strengths of RDF is the possibility to describe the vocabularies used to link entities in a machine-readable and dereferenceable way as well. This description can be encoded using either RDFS or the *Web Ontology Language (OWL)* [1]. Using the `owl:inverseOf` property, a property itself can define which property is to be used for back links. For example, the link property `hasChild` could have the inverse link property `hasParent`. Alternatively, vocabularies can specify properties to be symmetric, for example the property `hasFriend` could be defined to be symmetric (assuming a main-stream sociocultural environment). Should a link property neither have an inverse link property, nor be defined to be symmetric, the remote statement linking the local and remote entity can be included into the local data set. Since agents can follow properties regardless of their direction, these links can be useful to them as well.

Figure 2 gives examples for both cases. For both pictures, the dashed elements are new to the local data set. If the inverse property is unknown, the remote statement is included (1). If the inverse property is known – for example by dereferencing the property URL – the correct link property `ex:p2` known to be the `owl:inverseOf` `ex:p1` along with the entity URL of the remote resource `ds2:res2` is included (2).

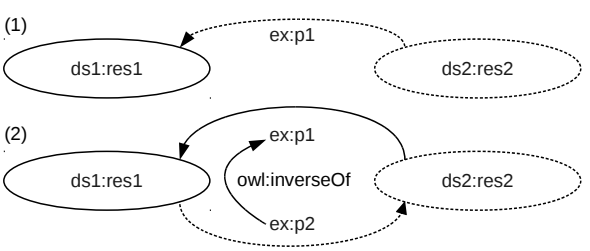

Figure 2: New Back Link Properties

From these prerequisites, the automatic generation or recommendation of back links in the Linked Data context is possible. The following algorithm can be executed fully automatically, and – given Referers are supplied by the user

[3]This spelling is used in this paper to be consistent with the HTTP specification

agents – will generate new and meaningful links between Linked Data entities in different data sets. In the following, RDF statements are encoded as triples in the triple notation $(subject, predicate, object)$. Algorithm 1 details the process of link (and statement) generation: After the document pointed to by the Referer URL has been retrieved, two cases are differentiated: If the response contains RDF statements, they are checked whether the local entity URL occurs as subject or as object. If the local entity occurs as an object, the remote statement is returned. If the local entity occurs as an object in one of the statements, three cases are possible: First, the link property may be symmetric, in this case it is used to create the connecting statement (Line 11). Second, if the inverse property is known, that property is used to create the new statement (Line 14). Third, if neither of both is the case, the remote statement is also returned. For non-RDF-documents, a string search for the URI of the local entity within the remote document is performed, if a match is found, a `rdfs:seeAlso` link is created as well (Line 22), since this link property explicitly allows linking to non-RDF resources [4].

Algorithm 1 Link Generation from Referers

Require: Requested local entity URL u, Referer URL r

```
 1: r_doc ← retrieve(r)
 2: if isRDF(r_doc) then
 3:   statementSet ← parseRdf(r_doc)
 4:   for all statementSet as s do
 5:     if subject(s) == u then
 6:       return s
 7:     end if
 8:     if object(s) == u then
 9:       p ← predicate(s)
10:       if isSymmetric(p) then
11:         return (subject(s), p, u)
12:       end if
13:       if hasInverseProperty(p) then
14:         return (subject(s), inverse(p), u)
15:       end if
16:       n ← createNewLocalUrl()
17:       return s
18:     end if
19:   end for
20: else
21:   if contains(r_doc, u) then
22:     return (u, rdfs:seeAlso, r)
23:   end if
24: end if
```

The statements generated by this algorithm can now be used in a variety of ways. We propose two methods: First, the statements could be handed over for review by another software component or the person responsible for the local data set. Second, an automatic inclusion into the data set is also feasible. In this case, we recommend storing the statements in a separate Named Graph, along with a machine-readable provenance annotation, for example using the Provenance Vocabulary [9].

3. EVALUATION

To answer our research question and validate our algorithm, real log files from web servers hosting Linked Data sets were analyzed. Two sets of log files were made available for the USEWOD 2011 Data Challenge [2]. The first set of files was created on the web server of the *DBpedia* project, the second set on the web server hosting the *Semantic Web Dog Food* project. Both servers used the Apache "combined" log format[4], which is the default setting. Each log entry is represented by one line in the log file. Each log entry is similar to the following sample entry in the "combined" format (line breaks added, not an actual log entry):

```
160.45.170.10 [07/Jan/2010:09:52:45 -0800]
"GET /resource/South_Bend,_Indiana HTTP/1.1"
303 40
"http://en.openei.org/wiki/South_Bend,_Indiana"
"Mozilla/4.0"
```

The format is structured into fields for client IP address, date and time, HTTP request method and URL, status code, bytes transmitted for the response, "Referer" request header field, and user agent (browser). In order to generate new links, two things have to be determined: First, the URL of the local resource that was requested, and second the URL of the remote resource the user agent visited before. This data can be taken from the described log file format.

In total, about 27.86 million log entries were parsed, filtered, and checked for "interesting" Referer entries. Filtering included the removal of log entries without the optional Referer field, local redirects, and log entries with Referer entries pointing to result pages of search engines such as Google, Yahoo, etc.. For all remaining entries, the Referer URL was resolved, and the resulting HTML or RDF document searched for the URL of the local resource identifying a local entity. Requests expressed their preference for RDF document responses using the `Accept` HTTP header. Thus, this operation was defined to have four possible outcomes:

- *Not found* – The local resource was not found in the remote document, neither in plain text nor RDF
- *Text match* – the local resource was found occurring in a plain text or HTML response
- *RDF subject match* – the local resource was found in a remote RDF statement as the subject entry
- *RDF object match* – the local resource was found in a remote RDF statement as the object entry. In the last case, the properties used to link to the local resource were also recorded

For RDF matches (not considering possible links to HTML documents), new statements linking the local and remote resources were generated according to our algorithm. Then, an additional request was performed on the local data set to see whether the local data set already contains this statement. If this was not the case, the new statement could have been added to the data set.

The frequencies of the possible outcomes mentioned above as well as the properties used for object matches can give

[4] `http://httpd.apache.org/docs/current/logs.html#combined`

an indication whether the additional links created using our approach merit the additional effort of analyzing log files for Referer entries.

Table 1 contains the detailed results of our evaluation. For each data set, the raw amount of log entries, the amount of log entries with Referers, the amount of Referer URLs ultimately dereferenced, and the amount of unique dereferencing results are given in the first block. The second block details the frequencies for the different result types as described above. The third block gives the amounts of new statements that could be generated from our results, and the amount of generated statements according to our algorithm that were not yet contained in the respective data set. The quality of the generated statements was evaluated using manual inspection, and no obviously bogus statements were found. It has to be noted that we limited the generation of new statements to RDF matches, since they enable more meaningful back links. Since this analysis included "live" data[5], results may vary for repeated analyses of the same log file set.

	DBpedia	*SWDF*
Log entries	19,770,157	8,092,552
Referer set	1,328,595	533,188
Dereferenced	4,217	20,451
Unique Results	3,255	6,146
Result type		
Not found	2,229	4,821
Text match	431	1,168
Subject match	395	47
Object match	200	110
Statements		
Total	595	157
New	507	136

Table 1: Evaluation Results

The most frequent properties used in object matches are given in Table 2 for the two data sets. Entries with less than ten occurrences are are not included. Both the dereferencing results as well as the statements generated for the respective data sets are available online[6] in order to to support further analysis.

Property URI	*Freq.*
DBpedia	
http://www.w3.org/2002/07/owl#sameAs	95
http://dbpedia.org/ontology/wikiPageRedirects	77
http://rdfs.org/sioc/ns#links_to	21
http://www.rkbexplorer.com[..]#duplicate	3
SWDF	
http://www.w3.org/2002/07/owl#sameAs	42
http://xmlns.com/foaf/0.1/knows	35
http://www.w3.org[..]rdf-schema#seeAlso	16

Table 2: Link Property Usage

[5]Accessible on 2011/03/10

[6]`http://page.mi.fu-berlin.de/muehleis/ldow2011/`

Two main conclusions can be drawn from our evaluation: First, the generation of new links between Linked Data entities is indeed possible using log files, which contain Referer values. Second, the comparably small amount of statements generated shows the failure of Linked Data clients and crawlers to properly set the Referer header.

4. RELATED WORK

Link discovery between data entities across data sets requires linkage recording and duplicate detection techniques. While there is a large amount of related work on these topics in the database community [15, 5] as well as on ontology matching in the knowledge representation community [6], the approaches for Linked Data are still limited at the moment.

The Silk Link Discovery Framework [11] is an identity resolution framework which generates RDF links between data items based on user-provided link specifications which are expressed using the Silk Link Specification Language. Silk is available in different variants, one on them being Silk Server. Silk Server can be used as an identity resolution component within applications that consume Linked Data from the Web. It provides an HTTP API for matching instances from an incoming stream of RDF data.

LIMES [12] is a link discovery framework for the Web of Data. It is available as a web interface as well as standalone tool. It offers string metrics.

LinQuer [10] is a tool for semantic link discovery over relational data, based on string and semantic matching techniques and their combinations. The LinQuer framework rewrites linkage requirement queries into standard SQL queries that can be run over relational data sources. LinQuer is meant to be used together with relational databases to RDF wrappers such as D2R Server or Virtuoso RDF Views.

Raimond et al. [14] propose a link discovery algorithm that takes into account both the similarities of data entities on the Web of Data and of their neighbor entities. The algorithm is implemented within the GNAT tool.

The RKBExplorer sameAs service[7] provides a unified view over different Linked Data sets by managing owl:sameAs links to identify duplicate URIs. The links have to be provided to the system from external sources, which also applies to the related BackLink service.

Most of the current approaches generate links semi-automatically based on user-defined link specifications. This requires data providers to keep up with new linking possibilities and schemata. Furthermore, except for Silk Server and RKBExplorer's sameAs service, data sets to be linked have to be specified manually. This doesn't scale for the growing number of data sets on the Web of Data.

[7]`http://www.rkbexplorer.com/sameAs/`

5. CONCLUSION

Acting on the fourth Linked Data principle, namely the need for cross-dataset links between Linked Data entities, we have identified the Referer request header field defined by the HTTP specification as a possible source for automatic creation of those links. However, the presence of an Referer URL does not prove the presence of an existing link to a local entity. Thus, our approach is based on applying the third Linked Data principle – the possibility of de-referencing arbitrary URLs – on the Referer URL. When retrieving the document identified by the Referer, we were able to ascertain the presence of a link between a remote entity to a local entity along with the link type used. We were then also able to determine the semantically correct back link property and create a new locally stored back link leading from a local entity to a remote entity.

We have evaluated our fully automatic approach using log entries from web servers hosting the DBpedia and Semantic Web Dog Food data sets. In total, 27.86 million log entries were analyzed, and 24,668 Referer URLs were dereferenced, yielding 9,401 distinct results. From these results, we were able to generate 643 new typed links. Our results show the feasibility and practicability of automatic back link generation for Linked Data entities using Referer information in general and web server log files in particular.

From our results, the failure of many Linked Data clients and spider programs to add the Referer header field to their requests was identified to be the main factor limiting the amount of statements generated by our algorithm. We therefore would like to urge developers of Linked Data tools to set the Referer request header to the resource where the URL of the document currently retrieved was found whenever possible.

5.1 Further Work

Since our approach can be used to directly add statements based on information loaded from remote sources, the statements generated are easily susceptible to malicious requests and malicious remote statements. For example, if an attacker would publish RDF data linking a popular DBpedia entity (e.g. `dbpedia:Berlin`) to his advertisement page, and then creating a request to this entity with his document as Referer, the algorithm would automatically create a link from the popular resource to the advertisement page. To overcome this problem, one could evaluate provenance information in order to establish and enforce a required trust level, before new links are created [9].

We would also like to create a generic tool for Linked Data server administrators, which they can use to automatically process their log entries for interesting Referers, generate new back links, and automatically publish these links again in their local data set. Alternatively, the tool could also display the new statements to an administrator for approval.

Acknowledgments

This work has been partially supported by the "DigiPolis" project funded by the German Federal Ministry of Education and Research (BMBF), support code 03WKP07B. The authors would like to thank the reviewers and their colleagues R. Oldakowski and M. Luczak-Rösch for their insights.

6. REFERENCES

[1] Sean Bechhofer, Frank van Harmelen, Jim Hendler, et al. Owl web ontology language reference, 2004.

[2] B. Berendt, L. Hollink, V. Hollink, M. Luczak-Rösch, K. H. Möller, and D. Vallet. USEWOD2011 — 1st international workshop on usage analysis and the web of data. In *20th International World Wide Web Conference (WWW2011), Hyderabad, India*, 2011.

[3] Tim Berners-Lee. Linked data, 2006. http://www.w3.org/DesignIssues/LinkedData.html accessed 2010-08-12.

[4] Dan Brickley, R.V. Guha, and Brian McBride. Rdf vocabulary description language, 02 2004.

[5] Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, and Vassilios S. Verykios. Duplicate record detection: A survey. *IEEE Trans. on Knowl. and Data Eng.*, 19(1):1–16, 2007.

[6] Jérôme Euzenat, Alfio Ferrara, Christian Meilicke, et al. Results of the ontology alignment evaluation initiative 2010. In *Proc. 5th ISWC workshop on ontology matching (OM), Shanghai (CN)*, pages 85–117, 2010.

[7] Fielding, Gettys, Mogul, Frystyk, Masinter, Leach, and Berners-Lee. Hypertext transfer protocol – http/1.1, 1999.

[8] Olaf Hartig and Andreas Langegger. A database perspective on consuming linked data on the web. *Datenbank-Spektrum, Semantic Web Special Issue*, 10 / 2010, 2010.

[9] Olaf Hartig and Jun Zhao. Publishing and consuming provenance metadata on the web of linked data. In Deborah L. McGuinness, James Michaelis, and Luc Moreau, editors, *IPAW*, volume 6378 of *Lecture Notes in Computer Science*, pages 78–90. Springer, 2010.

[10] Oktie Hassanzadeh, Reynold Xin, Renée J. Miller, Anastasios Kementsietsidis, et al. Linkage query writer. *PVLDB*, 2(2):1590–1593, 2009.

[11] Robert Isele, Anja Jentzsch, and Christian Bizer. Silk Server - Adding missing Links while consuming Linked Data. In *1st International Workshop on Consuming Linked Data (COLD 2010), Shanghai*, 2010.

[12] Axel-Cyrille Ngonga Ngomo and Sören Auer. Limes - a time-efficient approach for large-scale link discovery on the web of data, 2011.

[13] Eyal Oren, Renaud Delbru, Michele Catasta, Richard Cyganiak, et al. Sindice.com: a document-oriented lookup index for open linked data. *Int. J. of Metadata and Semantics and Ontologies*, 3:37–52, November 10 2008.

[14] Yves Raimond, Christopher Sutton, and Mark Sandler. Automatic interlinking of music datasets on the semantic web, 2008.

[15] William E. Winkler. Overview of record linkage and current research directions. Technical report, Bureau of the Census, 2006.