Pay-as-you-go Data Integration for Linked Data: opportunities, challenges and architectures

Norman W. Paton
School of Computer Science
University of Manchester
Oxford Road, Manchester
M13 9PL, UK
norm@cs.man.ac.uk

Klitos Christodoulou
School of Computer Science
University of Manchester
Oxford Road, Manchester
M13 9PL, UK
christk6@cs.man.ac.uk

Alvaro A.A. Fernandes
School of Computer Science
University of Manchester
Oxford Road, Manchester
M13 9PL, UK
alvaro@cs.man.ac.uk

Bijan Parsia
School of Computer Science
University of Manchester
Oxford Road, Manchester
M13 9PL, UK
bparsia@cs.man.ac.uk

Cornelia Hedeler
School of Computer Science
University of Manchester
Oxford Road, Manchester
M13 9PL, UK
chedeler@cs.man.ac.uk

ABSTRACT

Linked Data (LD) provides principles for publishing data that underpin the development of an emerging web of data. LD follows the web in providing low barriers to entry: publishers can make their data available using a small set of standard technologies, and consumers can search for and browse published data using generic tools. Like the web, consumers frequently consume data in broadly the form in which it was published; this will be satisfactory in some cases, but the diversity of publishers means that the data required to support a task may be stored in many different sources, and described in many different ways. As such, although RDF provides a syntactically homogeneous language for describing data, sources typically manifest a wide range of heterogeneities, in terms of how data on a concept is represented. This paper makes the case that many aspects of both publication and consumption of LD stand to benefit from a pay-as-you-go approach to data integration. Specifically, the paper: (i) identifies a collection of opportunities for applying pay-as-you-go techniques to LD; (ii) describes some preliminary experiences applying a pay-as-you-go data integration system to LD; and (iii) presents some open issues that need to be addressed to enable the full benefits of pay-as-you go integration to be realised.

Categories and Subject Descriptors

C.2.4 [**Distributed Systems**]: Distributed databases; H.3.5 [**Online Information Services**]: Data sharing; H.3.5 [**Online Information Services**]: Web-based services

SWIM 2012, May 20, 2012, Scottsdale, Arizona, USA.

1. INTRODUCTION

Linked Data (LD) seeks to do for data what the web did for documents. In essence, *Linked Data* [4] involves publication of data according to a collection of principles, namely that: (i) all items are identified using URIs; (ii) all URIs are dereferenceable; (iii) when dereferenced a URI leads to useful information represented using RDF; and (iv) links to other URIs are included to allow further navigation. In this setting, publishers make data available following the principles, and users access or process the resulting data using either generic tools or bespoke applications.

The LD activity therefore supports the publication and sharing of data building on technologies developed by the web and semantic web communities. LD follows the web in providing low barriers to entry: publishers can make data available using a small set of standard technologies, and consumers can search for and browse published data using generic tools. This is fine in principle: the low cost of publishing encourages the creation of new resources, indexes constructed by crawlers over these resources allow the data to be discovered, and the well established browser model provides related data by following links. However, in current practice, what users typically obtain when consuming LD is essentially what was published. Thus if the publisher has followed best practice (e.g., by making effective use of standard vocabularies in describing their data, has included links of relevance to the user, and has kept these links up-to-date with respect to available resources), then the resource referenced by a URI may well be useful to both human and software consumers. Thus, while we assume that current tools and techniques can be valuable, we argue that the potential exists for adding significant value to the basic model.

The role of LD as the foundation for an open web of data encourages the creation of numerous diverse publishers: anyone can publish into the web of data. While this creates an opportunity, it also creates challenges for consumers. For example, if a user is interested in data on *Manchester University*, the Sig.ma [38] browser combines results on the university, on Manchester Grammar School, and on a nearby railway station, and in so doing brings together lots of correct, valuable information with a fair amount of misleading or in-

correct data. This is not a criticism of Sig.ma – automatic integration is clearly a challenging business, and forming a clear picture of what is said about Manchester University in the LD cloud is not straightforward. Indeed, providing a coherent and integrated view of data from LD resources retains classical challenges for data integration (e.g. identification and resolution of format inconsistencies in value representation, handling of inconsistent structural representations for related concepts, entity resolution), and poses these on a grand scale. In addition, a collection of issues faced by users of LD has been identified by a study of data sets obtained by crawling a portion of the LD cloud. The study reveals that LD gives rise to various challenges of its own that stand as significant obstacles to the straightforward consumption of LD [25], including dangling references, lack of structured data, incorrect use of RDF constructs, data inconsistent with declared types, inconsistent use of ontologies by different resources, etc.

In the database community, the challenge of providing cost-effective data integration has given rise to ongoing research in pay-as-you-go data integration, proposals for which are often referred to as Dataspaces [18]. The vision is that various of the benefits of manual, resource-intensive data integration could be obtained over time at lower cost, whereby an initial integration is produced automatically that is subsequently refined to support user requirements. We see these strands of activity in *linked data* and *pay-as-you-go data integration* as highly complementary; this paper outlines how they might be brought together to support pay-as-you-go data access and publishing for linked data.

To support pay-as-you-go integration, dataspaces have a lifecycle [21] that involves: *initialisation*, i.e., automatic inference of the relationships between the resources, which typically provides unsatisfactory query results; *usage*, i.e., evaluation of requests over the dataspace, to yield structured and potentially ranked results; and *improvement*, i.e., the collection of feedback, typically in the light of *usage*, which refines the understanding of the integrated resources and their relationships, and thus enables the integration to better correspond to user requirements. The forms of *payment* may be diverse, but typically involve some form of feedback that incrementally refines the initial integration. For example, feedback on the result of a request (specifically to indicate if individual results are true or false positives, or to provide false negatives) can be used to choose between mappings or to derive refined versions of existing mappings that better meet user requirements [2].

This paper discusses the opportunities that result from applying pay-as-you-go data integration principles and techniques to LD, and argues that many different tasks associated with linked data stand to benefit from a pay-as-you-go approach. In so doing, it follows the following structure. In Section 2, we introduce some terminology and techniques associated with classical manual and pay-as-you-go data integration. In Section 3, we discuss how linked data processing, specifically including publishing, searching, querying and browsing, stand to benefit from more integrated representations of data on the web, and how these benefits may be derived in a pay-as-you-go manner. To make things more concrete, in Section 4, we describe some initial results integrating linked data resources using the DSToolkit Dataspace Management System [22]. Building on this experience, we discuss the current state-of-the-art, open issues and challenges in Section 5, before presenting some conclusions in Section 6.

2. DATASPACES

In the database community, the objective of data integration has typically been to provide the illusion that data obtained from multiple distributed data sources stems in fact from a single, well managed resource. In such a setting, a *global schema*[1] is defined that represents the data requirements of the users, and mappings (expressed using a query language) describe how to populate the global schema from the sources. The development of such mappings is challenging in practice because the required data may be distributed across multiple independently developed databases, where the structures and terminologies used to represent the data are inconsistent. The classical data integration lifecycle proceeds as follows:

- *Matching*: to enable the similarities between the available sources and the global schema to be quantified, matchers are run over the schemas (and perhaps samples from the instances). Matchers typically carry out tasks such as string comparisons and dictionary lookups, to identify associations between (assuming relational terminology) tables and attributes in different sources. There are many proposals for matching algorithms and tools for databases and ontologies in the literature (e.g. [31, 16]).

- *Mapping*: as the results of matching algorithms typically contain both false positives and false negatives, schema mapping tools are used, in which a skilled user refines the collection of matchings in use, and creates mappings that describe how to translate data from one representation to another. The mapping tools support users by proposing candidate mappings given collections of matches (e.g. [17]).

The challenges associated with the construction of mappings are reflected in ongoing research into mapping debugging [9] and verification [6]. The labour-intensive nature of such a development process means that classical data integration is best suited for use in high-value integrations that are of manageable size and relatively stable. As LD is associated with numerous, rapidly changing resources, and diverse and unpredictable user requirements, classical approaches to data integration seem inappropriate.

The vision of pay-as-you-go data integration in Dataspaces [18] is that certain of the benefits of classical data integration can be obtained at reduced cost, and that the cost and quality of integrations can be traded-off to reflect specific requirements and constraints. As surveyed by Hedeler *et al.* [21], there are a significant number of proposals for pay-as-you-go techniques or systems, and pay-as-you-go integration is the subject of ongoing investigation. Although there is considerable variety, in terms of scope and methodology, individual proposals can often be characterised by their approaches to:

[1]The name *global schema* is potentially misleading, as there may be many, and there is certainly no requirement that this should subsume the schemas of the participating sources.

- *Initialisation:* Initialisation involves automatic creation of mappings between a global schema and data sources that may be able to be used to populate the former. A global schema may, for example be obtained by unioning source schemas, i.e., there is effectively no initial integration and the user accesses the underlying sources through a uniform interface but in terms of their original models (e.g. [33]), or by merging the schemas from the sources to produce a single model that represents concepts that may appear in multiple sources (e.g. [34]). Where initialisation seeks to provide an integrated representation of data from multiple sources, the associated mappings are likely to be of low quality because they build on imperfect matches between the sources to be intergrated.
- *Improvement:* As the integration from bootstrapping is put to use, feedback is obtained from users and used to revise different aspects of the integration. This feedback may be *explicit*, e.g., on mappings [7] or on query results [2], or *implicit*, e.g., using information from query logs [14]. The feedback can then be used in different ways to revise the integration, e.g., by identifying missing matches [14], selecting between mappings [2, 36] or creating new mappings [2].

While pay-as-you-go data integration techniques can be applied, as above, to insulate users from the diversity of representations used in their domain of interest, we envisage several different application contexts in LD, as discussed in the next session.

3. OPPORTUNITIES

This section discusses how enhanced data integration, brought about in a pay-as-you-go manner, might support both publishers and consumers of LD.

Publication – sharing structure.

Publishing onto the web of data typically includes [20]: (i) identifying existing terminologies that can be used to describe the data to be published, potentially in combination with each other or with some new types; and (ii) translating the data to be published from its current form into the chosen RDF representation.

How might pay-as-you-go data integration techniques help with structure sharing? Given the data to be published, for example from a relational database or an XML document, it would be possible to match this data with resources in the linked data cloud or with ontologies. Automatic mapping generation could then be used to generate RDF representations of the data to be published using identified candidate representations. Such automatically generated mappings are likely to be of low quality, so feedback on these could then be provided by the publisher, for example to indicate where values have been mapped correctly or incorrectly, enabling the generation of revised mappings and thus new examples for inspection. The publisher could also modify the target representation, and request that new mappings be generated to populate the revised version. Thus the automatic generation of matchings and mappings could be used to identify and evaluate candidate ontologies, and feedback following the pay-as-you-go approach could be used to refine default mappings that automate the translation of the existing data into the form for publication.

Publication – linking.

It is one of the principles of LD that published resources should include links to related data [4]. Publication of links from a published resource typically involves [20]: (i) identifying places in the resource representation where there is value to be obtained from external links; (ii) deciding what predicates to use with which to make such links; and (iii) identifying the URLs of suitable link targets.

How might pay-as-you-go data integration techniques help with linking? Given the data to be published, there seems to be value in the creation of queries that can retrieve the URLs that refer out from a published resource. We note that many LD resources contain few external links, suggesting that the manual creation and maintenance of links is imposing a cost on publishers that they may be reluctant to bear. We observe that identifying links between sources has much in common with mapping – in essence, it is necessary to identify resources that contain related data from within which suitable URIs can be identified, along with the conditions that characterise suitable links. Techniques for entity resolution [13], which often have a significant amount in common with matching, can be used to identify candidate link targets for feedback. Publishers or users could be asked to provide feedback on the suitability of the automatically generated links, thus potentially supporting changes to the queries that generated the link targets in ways that improve their precision and recall (e.g. as investigated previously for mappings [2, 36]). There is recent work on inferring linkage rules [26], although opportunities for feedback informing revisions to such rules seem to be largely unexplored.

Searching and Browsing – result alignment.

Searching and browsing are central to the web of documents, and are likely to remain so for the web of data. However, where both tasks take place over LD rather than over documents, there are clearly opportunities for presenting users with representations of the data that are more integrated than those that are published. For example, when search results are displayed in Sig.ma [38], data from multiple sources are brought together in a single report, and pay-as-you-go refinement of such reports is supported because users can provide feedback on the sources and domains that are contributing to a report, for example by accepting or rejecting specific sources.

How might pay-as-you-go data integration techniques help with result alignment? The capabilities provided with Sig.ma for tailoring reports already provide a form of pay-as-you-go result integration. However, there are circumstances in which the default behaviour of a system like Sig.ma will not be close to the required behaviour. For example, assume that we are interested in visiting Brussels; a search using Sig.ma for *Brussels Hotels* produces a report that includes links to hotels in Brussels, but does not directly provide the sort of information a potential visitor seems likely to want about hotels. Instead, the user could format a table that they would like to have populated, for example, including the hotel name, street name, star rating, number of rooms and photo. Populating such a table from multiple sources involves matching the table description with the data from different sources, and developing mappings to populate the table. In pay-as-you-go integration, complete rows in the table, or specific attribute values are likely to be unsatisfactory in the absence of feedback. Feedback of the type already ob-

tained by Sig.ma could be used to refine the matchings used and the mappings generated to better reflect the user's requirements.

Querying – rewriting to different structures.

A web of data described using RDF seems to provide the opportunity to move beyond keyword search, to more precise question answering, using queries. However, composing queries over LD directly is a tricky business. SPARQL queries express their requirements using graph patterns in which predicates must be known to allow precise requests to be constructed. However, in LD, sources that describe data of interest to a user may not only need to be queried using different predicates, but may also have completely different overall graph structures, and may partition the data of interest in different ways. Existing proposals for querying linked data typically (implicitly) assume homogeneous and known representations across source boundaries (e.g. [19]).

How might pay-as-you-go data integration techniques help with querying? Following the goal of data integration from the database community, we assume that authors querying LD should be able to write their queries on the assumption that the data they require is represented using a consistent set of structures and terminologies (whereas in practice it may be physically stored using diverse structures and terminologies). To achieve this, it is necessary to know how data is described across the web of data, and to understand how the different representations relate to each other. In this setting, pay-as-you-go data integration proceeds broadly as described in Section 2, and is illustrated for the DSToolkit dataspace management system in the following section.

4. ARCHITECTURE

There is no single architecture or context for pay-as-you-go data integration; indeed the principles and techniques can be applied in different ways and settings. For example, in Sig.ma [38], pay-as-you-go integration of search results is integrated with a searching and browsing platform. In contrast, the Silk link discovery framework [40] is a free-standing tool that provides a link specification language for describing how to populate and maintain links; such a platform could be extended to take into account different types of feedback. For example, *implicit* feedback might be obtained when the links generated by a link specification were rarely traversed, and if traversed were rarely used for additional navigation. This might suggest that the links were of limited interest or quality, and thus that an alternative link specification should be considered.

In this section we describe how a dataspace management system (DSMS), DSToolkit [22], can be adapted for use with linked data, to support querying over sources with different structures, one of the opportunities discussed in Section 3. As DSToolkit is essentially a library of data integration components, it can be used to support a range of different styles of integration, so the approach followed here should be seen as illustrative rather than prescriptive.

In describing DSToolkit, we use a music case study involving the *Jamendo* and *Magnatune* sources from *dbtune.org*. In the case study, access is provided to the sources by way of a user-supplied global schema over which queries can be written that insulate the user from the differences in the representations used in the sources. The following steps are taken to initialise the dataspace using DSToolkit, as illustrated for the case study in Figure 1.

Structure inference.

The data stored in an RDF source does not conform to any specific structure analogous to the schema of a relational database. However, query processing requires an understanding of how concepts are represented (e.g. for distributed query processing in SPARQL, Quilitz and Leser assume that sources provide structural summaries [30]). To obtain such an understanding, we run a hierarchical agglomerative clustering algorithm [32] over resources in a source, to identify recurring structural patterns. For the case study, Figure 1(a) illustrates the data stored in *Jamendo* and *Magnatune* using Turtle, for which the structure inference process generates the models illustrated in Figure 1(b) as ER diagrams. These structural summaries of the data in the RDF sources are read by DSToolkit, for internal storage using a supermodel – a data model that subsumes the data models of typical sources, and which allows many of the capabilities of a data integration system to be developed in a way that is agnostic to the types of source being integrated [1]. The same supermodel has been used in with RDF by De Virgilio, *et al.* [10, 11], in research into architectures for storing and querying RDF data. In terms of implementation, to integrate RDF sources using DSToolkit, a wrapper has been produced that can import structural summaries, and that can translate queries expressed over the structural summaries into SPARQL.

Matching.

In the case study, the objective is to use a given global schema, which is shown in the middle of Figure 1(c), for querying the underlying sources. Thus we need to make explicit how this global schema relates to the underlying sources. To start this process, we have run a schema matching algorithm (specifically, an n-gram matcher[2]) between the global schema and each of the inferred source schemas, which identifies the matches depicted in Figure 1(c). As is commonly the case with matching, many associations are identified that require selection and grouping to inform mapping generation.

Correspondence inference.

The correspondence inference step uses an evolutionary search over the matches to identify a collection of commonly occurring correspondences between schemas, from the classification of Kim *et al.* [27]. These correspondences include 1-to-1, 1-to-many and many-to-many entity correspondences. Where *many* entity types are identified as participating together in a correspondence, they are considered to be either *horizontally* or *vertically* partitioned; these forms of partitioning effectively construct a new entity type by applying a *union* or *join* operator to their participants, respectively. Figure 1(d) shows the most highly ranked collection of correspondences between the global and the local schemas. In the figure, *DNSC* represents *Different Name for the Same Concept*, and *VP* represents *Vertical Partitioning*. For example, the inferred correspondences consider *Record* in the global schema to be vertically partitioned into *Magnatune.Record* and *Magnatune.Track*; this is because attributes of *Record* in the global schema are spread across both *Magnatune.Record*

[2]We use this type of matcher more for illustrative purposes than because we claim it is particularly suitable in this setting.

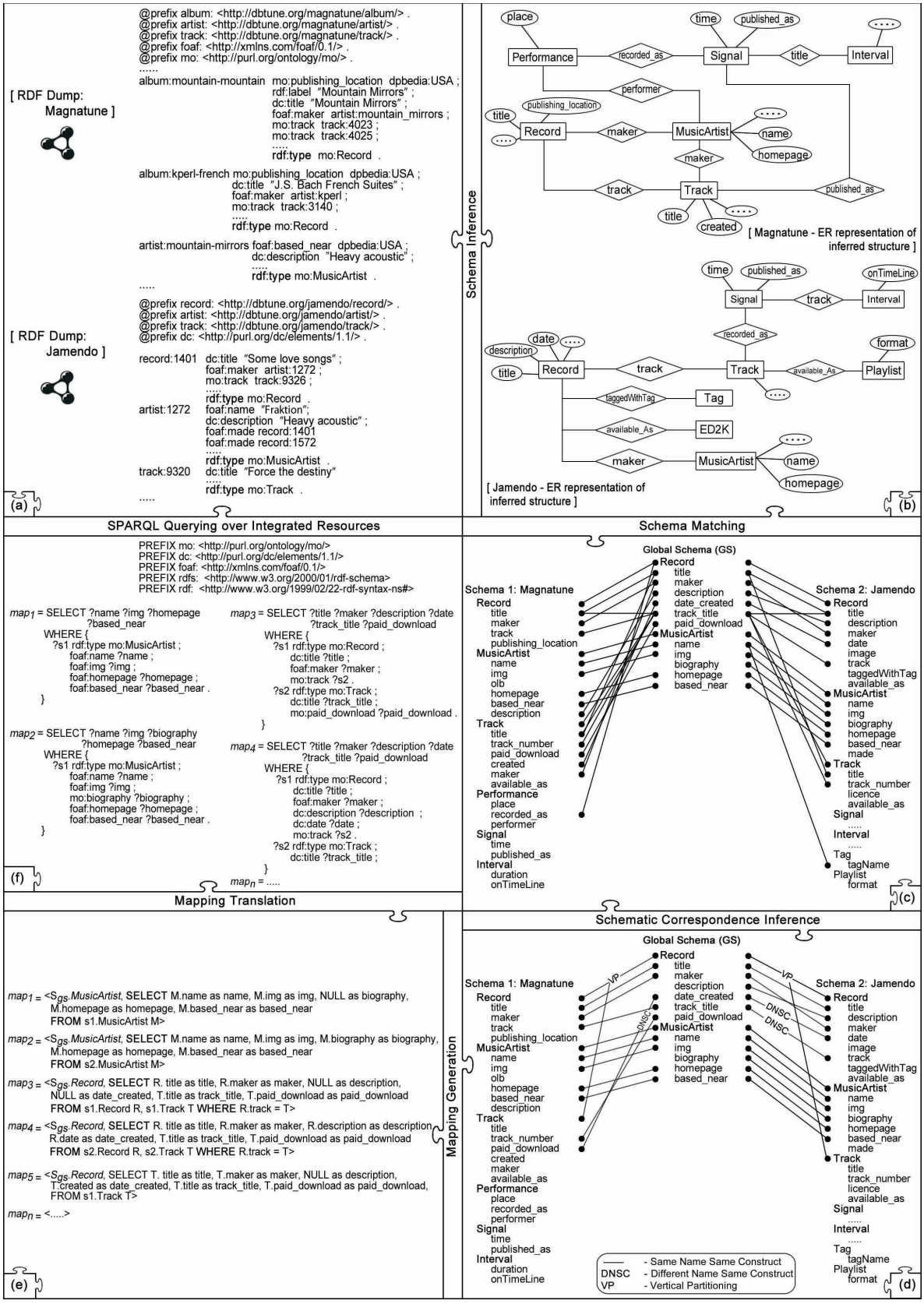

Figure 1: Initialisation for music sources using DSToolkit.

title	maker	date_created	track_title	expected	not expected	mappings
J.S. Bach French Suites	artist:kperl		Suite No 5 in G major BWV 816: Gigue	✓		map_3
J.S. Bach French Suites	artist:kperl		Suite No 2 in C minor BWV 813: Gigue	✓		map_3
Phantoms	artist:hans_christian		The Sacrifice	✓		map_3
Tempest	artist:4013	2007-07-23	A sea Change	✓		map_4
Tempest	artist:4013	2007-07-23	Storm Warning	✓		map_4
Suite No 3 in B minor BWV 814: Anglaise	artist:kperl		Suite No 3 in B minor BWV 814: Anglaise		✓	map_5
Aftermath	artist:hans_christian		Aftermath		✓	map_5

Figure 2: Obtaining feedback on mapping results.

and *Magnatune.Track*.

Mapping generation.

The correspondences inferred in the previous step include sufficient information to allow mappings to be generated from them algorithmically [29]. In DSToolkit, these mappings are expressed internally using SMQL, a query language for the supermodel that non-expert users need not be aware of. Figure 1(e) illustrates some mappings expressed in SMQL, including examples of those generated from both 1-to-1 and 1-to-many correspondences in Figure 1(d). In the mappings, *s1* represents *Magnatune*, *s2* represents *Jamendo* and S_{gs} represents the global schema. Queries expressed over the supermodel in SMQL can be algorithmically rewritten to the query languages of the underlying sources, as described in [23]; the SPARQL generated from the mappings in Figure 1(e) for *Jamendo* and *Magnatune* data in Figure 1(a) is given in Figure 1(f). In terms of implementation, DSToolkit compiles and optimizes SMQL queries that may require access to data from many sources, translates the source-specific subqueries into the source-specific languages for local evaluation, and combines the results using its own query evaluator.

Mapping selection.

Both matching and correspondence inference can give rise to multiple overlapping mappings, which provide different ways of obtaining values in the global schema. Such candidate mappings can be expected to include some mappings that are complementary, some that are alternatives that essentially produce the same data, and many that are incorrect in some way or other. Thus the feedback phase seeks to obtain additional information from users that can guide the improvement of the dataspace. As an example, DSToolkit collects feedback in the form of *true positive*, *false positive* and *false negative* annotations on tuples in the results of queries. These annotations allow estimates to be formed as to the precision and recall of different mappings, which in turn support both *mapping selection* and *mapping refinement* [2]. In *mapping selection*, given a collection of mappings annotated with estimates of precision and recall, it is an optimization problem to identify the set of mappings that will produce the maximum recall for a target precision, or the maximum precision for a target recall. As such, users are able to trade off precision and recall by indicating a target for one, which the system will then seek to meet while maximising the other. Figure 2 illustrates results obtained for a quey over *Record* in the global schema, using the mappings map_3, map_4 and map_5 from Figure 1(e), along with user feedback that indicates which result values were expected and which were not. From these annotations, the precision[3] of map_3 and map_4 can be estimated as *1.0*, and the precision of map_5 can be estimated to be *0*. Thus mapping selection is then able to avoid map_5, which incorrectly populates the *track* and *track_title* with identical values.

This section has described the initialisation and improvement phases of a DSMS, for querying data sources from the LD cloud that describe overlapping types of data in somewhat different ways. Thus this section has presented an approach to addressing one of the opportunities from Section 3, although significant challenges remain, as discussed in the next section.

5. CHALLENGES

In Section 3 we made a case that pay-as-you-go data integration techniques could play an important role supporting recurring requirements for publishing and consuming LD. In Section 4 we provided a concrete example of a DSMS, supporting querying in a way that insulated users from certain heterogeneities in the structures used by sources. However, overall, it is early days for pay-as-you-go data integration in general, and for LD in particular. This section discusses some of the research challenges that need to be addressed if the opportunities from Section 3 are to be fully realised.

- *Source description:* Data integration involves obtaining an understanding of the relationships between data sources, which tends to build on information about source structures. In LD, although good publication practice will lead to recurring use of terminologies, there will inevitably be terminologies with overlapping domains, and different sources will combine them in different ways when representing data. As such, there is a need for research into techniques that infer the structures found in sources in ways that support downstream tasks, such as matching, mapping and query routing. There is ongoing research on summarising linked data sources to support different tasks, such as query routing, which certainly have overlapping requirements with data integration (see Umbrich, *et al.* for a survey [39]).

- *Matching:* There is a considerable body of work on schema and ontology matching [31, 16]. However, the devil is often in the detail for matching strategies, and LD has certain features (such as shared terminologies, a large number of instance stores, global identifiers in the form of URLs, and links across independent sources) that present both challenges and opportunities for matchers. Although there are some results

[3]The precision of a mapping is estimated from user feedback as $precision = \frac{tp}{tp+fp}$, where tp is the number of tuples annotated as *expected* returned by the mapping, and fp is the number of tuples annotated as *unexpected*.

specifically associated with RDF data (e.g. [28]), and RDF matching features in the instance matching component of the Ontology Alignment Evaluation Initiative [15], there still seem to be significant opportunities for exploring matching that addresses the specific features of LD. For example, open issues include the expressiveness of the output produced by matching, where associations often simply denote the fact that some matcher has identified similarity of some form, whereas others have sought to produce richer descriptions of relationships within matching (e.g. [35]).

- *Mapping:* Research into schema mapping has produced a large body of work that seeks to generate, validate and debug mappings that are consistent with known matches and constraints (e.g. [17, 6, 9]). Such results frequently build on properties of the data representations used (e.g. [24]), and there seems be potential for combining such results with insights from graph isomorphisms (e.g. [8]) to identify patterns within and across LD sources for which mappings can be generated automatically in the context of data from matching.
- *Feedback:* For the most part, pay-as-you-go data integration proposals that use feedback collect a single type of feedback that is used for a small number of tasks. As argued in a recent paper [3], in fact there may be a rich collection of different kinds of feedback that not only can be used for a variety of purposes, but also that may be subject to a collection of common operations, e.g. for validation of feedback or clustering of users. In LD, there seems to be considerable potential for user communities to provide feedback that informs the integration of resources in their domains of interest, and there are several examples of the construction and maintenance of data resources using mass collaboration (e.g. Freebase [5]). Mass collaboration can take various forms [12]; for example, contributors might indicate explicitly that two results represent the same real-world object, that a single result conflates two real-world objects, or that specific properties of a result fail to meet their expectations. By contrast, contributors might implicitly indicate their interests through their search terms or the items that they click through. Such feedback could be used to cluster contributors into related groups, and could either be consolidated and applied automatically (e.g., by using different mapping rankings for different groups of contributors), or used to identify potential priorities for curators.

6. CONCLUSIONS

By building on a standardised technical platform, and adopting a number of principles informed by the success of the web, the LD cloud is rapidly developing into a valuable global data resource. However, although this resource benefits from the consistency provided by the use of a standardized platform for describing data, vocabularies and links, the proliferation of publishers and sources means that obtaining an integrated view of the data in an area of interest is often far from straightforward.

This paper has described how techniques from pay-as-you-go data integration may be able to contribute to the growing body of work that seeks to support LD publication and consumption. We are not the first to identify that pay-as-you-go integration might be relevant to linked data. For example, in Hermes [37] keyword queries are translated to SPARQL queries over LD in a setting where there is significant automatic initialisation of structures that support the translation, and in Sig.ma [38], users can provide feedback that refines their experience when browsing. However, our contention in this paper is that the space of opportunities for applying the pay-as-you-go approach is far from fully explored, and that the opportunities for increasing the collection and application of feedback to improve integrations are significant. We have identified several of the opportunities, and described the use of the DSToolkit DSMS to demonstrate how the initialisation and improvement stages of pay-as-you-go data integration can be used to address one of the opportunities, namely query processing over structurally heterogeneous RDF sources.

Acknowledgments

This work has been made possible by funding from the UK Engineering and Physical Sciences Research council, whose support we are pleased to acknowledge.

7. REFERENCES

[1] P. Atzeni, G. Gianforme, and P. Cappellari. A universal metamodel and its dictionary. *T. Large-Scale Data- and Knowledge-Centered Systems*, 1:38–62, 2009.

[2] K. Belhajjame, N. W. Paton, S. M. Embury, A. A. A. Fernandes, and C. Hedeler. Feedback-based annotation, selection and refinement of schema mappings for dataspaces. In *EDBT*, pages 573–584, 2010.

[3] K. Belhajjame, N. W. Paton, A. A. A. Fernandes, C. Hedeler, and S. M. Embury. User feedback as a first class citizen in information integration systems. In *CIDR*, pages 175–183, 2011.

[4] C. Bizer, T. Heath, and T. Berners-Lee. Linked data - the story so far. *Int. J. Semantic Web Inf. Syst.*, 5(3):1–22, 2009.

[5] K. D. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*, pages 1247–1250, 2008.

[6] A. Bonifati, G. Mecca, A. Pappalardo, S. Raunich, and G. Summa. Schema mapping verification: the spicy way. In *EDBT*, pages 85–96, 2008.

[7] H. Cao, Yan Qi, K. S. Candan, and M. L. Sapino. Feedback-driven result ranking and query refinement for exploring semi-structured data collections. In *EDBT*, pages 3–14, 2010.

[8] J. J. Carroll. Matching rdf graphs. In *International Semantic Web Conference*, pages 5–15, 2002.

[9] L. Chiticariu and W. C. Tan. Debugging schema mappings with routes. In *VLDB*, pages 79–90, 2006.

[10] R. De Virgilio, P. Del Nostro, G. Gianforme, and S. Paolozzi. A scalable and extensible framework for query answering over rdf. *World Wide Web*, 14(5-6):599–622, 2011.

[11] R. De Virgilio, G. Orsi, L. Tanca, and R. Torlone. Semantic data markets: a flexible environment for

knowledge management. In *CIKM*, pages 1559–1564, 2011.

[12] A. Doan, R. Ramakrishnan, and A. Y. Halevy. Crowdsourcing systems on the world-wide web. *Commun. ACM*, 54(4):86–96, 2011.

[13] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. *IEEE Trans. Knowl. Data Eng.*, 19(1):1–16, 2007.

[14] H. Elmeleegy, A. K. Elmagarmid, and J. Lee. Leveraging query logs for schema mapping generation in u-map. In *SIGMOD*, pages 121–132, 2011.

[15] J. Euzenat et al. Results of the ontology alignment evaluation initiative 2011. In *Proc. 6th Intl Workshop on Ontology Matching*. CEUR-WS.org, 2011.

[16] J. Euzenat and P. Shvaiko. *Ontology Matching*. Springer, 2007.

[17] L. M. Haas, M. A. Hernández, H. Ho, L. Popa, and M. Roth. Clio grows up: from research prototype to industrial tool. In *SIGMOD Conference*, pages 805–810, 2005.

[18] A. Y. Halevy, M. J. Franklin, and D. Maier. Principles of dataspace systems. In *PODS*, pages 1–9, 2006.

[19] O. Hartig, C. Bizer, and J. C. Freytag. Executing sparql queries over the web of linked data. In *ISWC*, pages 293–309, 2009.

[20] T. Heath and C. Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Morgan & Claypool, 2011.

[21] C. Hedeler, K. Belhajjame, A. A. A. Fernandes, S. M. Embury, and N. W. Paton. Dimensions of dataspaces. In *BNCOD*, pages 55–66. Springer, 2009.

[22] C. Hedeler, K. Belhajjame, L. Mao, C. Gao, I. Arundale, B. F. Loscio, N. W. Paton, A. A. A. Fernandes, and S. M. Embury. Dstoolkit: An architecture for flexible dataspace management. *T. Large-Scale Data- and Knowledge-Centered Systems*, 5:126–157, 2012.

[23] C. Hedeler, K. Belhajjame, N. W. Paton, A. A. A. Fernandes, S. M. Embury, L. Mao, and C. Guo. Pay-as-you-go mapping selection in dataspaces. In *SIGMOD*, pages 1279–1282, 2011.

[24] M. A. Hernández, L. Popa, Y. Velegrakis, R. J. Miller, F. Naumann, and C.-T. Ho. Mapping xml and relational schemas with clio. In *ICDE*, pages 498–499, 2002.

[25] A. Hogan et al. Weaving the pedantic web. In *Linked Data on the Web*. CEUR-WS/Vol-628/, 2010.

[26] R. Isele and C. Bizer. Learning linkage rules using genetic programming. In *Proc. 6th Intl Workshop on Ontology Matching*, volume 814 of *CEUR Workshop Proceedings*, 2011.

[27] W. Kim, I. Choi, S. K. Gala, and M. Scheevel. On resolving schematic heterogeneity in multidatabase systems. *Distributed and Parallel Databases*, 1(3):251–279, 1993.

[28] L. Leme, M. Casanova, K. Breitmsn, and A. L. Furtado. Evaluation of similarity measures and heuristics for simple rdf schema matching. Technical report, PUC, 2008. ISSN: 0103-9741.

[29] Lu Mao, Khalid Belhajjame, Norman W. Paton, and Alvaro A. A. Fernandes. Defining and using schematic correspondences for automatically generating schema mappings. In *CAiSE*, pages 79–93. Springer, 2009.

[30] B. Quilitz and U. Leser. Querying distributed rdf data sources with sparql. In *Proc. 5th ESWC*, pages 524–538. Springer, 2008.

[31] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *VLDB J.*, 10(4):334–350, 2001.

[32] L. Rokach. A survey of clustering algorithms. In *Data Mining and Knowledge Discovery Handbook*, pages 269–298. Springer, 2010.

[33] M. Antonio Vaz Salles, J.-P. Dittrich, S. Kirakos Karakashian, O. René Girard, and L. Blunschi. itrails: Pay-as-you-go information integration in dataspaces. In *VLDB*, pages 663–674, 2007.

[34] A. Das Sarma, X. Dong, and A. Y. Halevy. Bootstrapping pay-as-you-go data integration systems. In *SIGMOD*, pages 861–874, 2008.

[35] P. Shvaiko, F. Giunchiglia, and M. Yatskevich. Semantic matching with s-match. In R. De Virgilio, F. Giunchiglia, and L. Tanca, editors, *Semantic Web Information Management*, pages 183–202. Springer, 2009.

[36] P. P. Talukdar, M. Jacob, M. S. Mehmood, K. Crammer, Z. G. Ives, F. Pereira, and S. Guha. Learning to create data-integrating queries. *PVLDB*, 1(1):785–796, 2008.

[37] T. Tran, H. Wang, and P. Haase. Hermes: Data web search on a pay-as-you-go integration infrastructure. *J. Web Sem.*, 7(3):189–203, 2009.

[38] G. Tummarello, R. Cyganiak, M. Catasta, S. Danielczyk, R. Delbru, and S. Decker. Sig.ma: Live views on the web of data. *J. Web Semantics*, 8(4):355 – 364, 2010.

[39] J. Umbrich, K. Hose, M. Karnstedt, A. Harth, and A. Polleres. Comparing data summaries for processing live queries over linked data. *World Wide Web*, 14(5-6):495–544, 2011.

[40] J. Volz, C. Bizer, M. Gaedke, and G. Kobilarov. Discovering and maintaining links on the web of data. In *ISWC*, pages 650–665, 2009.