Re-using Cool URIs: Entity Reconciliation Against LOD Hubs

Fadi Maali
Digital Enterprise Research Institute, NUI Galway, Ireland
fadi.maali@deri.org

Richard Cyganiak
Digital Enterprise Research Institute, NUI Galway, Ireland
richard@cyganiak.de

Vassilios Peristeras
European Commission
vassilios.peristeras@ec.europa.eu

ABSTRACT

We observe that "LOD hubs" are emerging. They provide well-managed reference identifiers that attract a large share of the incoming links on the Web of Data and play a crucial role in data integration within communities of interest. But connecting to such hubs as part of the Linked Data publishing process is still a difficult task. In this paper, we explore several approaches to the implementation of *reconciliation services* that allow third-party publishers to link their data to LOD hubs as part of the data publishing process. We evaluate four approaches using the OAEI Instance Matching Benchmark, and describe their implementation in an extension to the popular data workbench application Google Refine.

Keywords

Linked Data, Google Refine, reconciliation, LOD, URI, entity matching, SPARQL, link generation, Silk

1. INTRODUCTION

Linked Data is a set of standards and practices for publishing structured data on the Web [5]. It can be seen as an approach to data integration at web-scale. It provides uniform data access, uniform syntax, and a uniform data model, but does not address the issues of heterogeneous schemas, duplicate records, and uncertain information quality. On top of an uniform base layer, existing and new research can be applied, on large amounts of real-world data, to yield new insights into the challenges of data integration in heterogeneous, low-coordination environments.

One of these challenges is interlinking and identifier re-use. The Linked Data Principles [2] require URIs to be HTTP-based and dereferenceable. Other sources recommend characteristics such as stability, permanence and readability [4, 20, 18][1]. Minting and maintaining "cool URIs" is thus not a simple task, but can compel other publishers to link to them. This linking across dataset boundaries turns the Web of Linked Data from a collection of data silos into a global data space [5].

[1]http://patterns.dataincubator.org/book/hierarchical-uris.html and http://patterns.dataincubator.org/book/natural-keys.html

We observe that hubs are emerging on the Web of Linked Data. This is visible in the popular LOD Cloud diagram[2]. A number of datasets, including DBpedia[3], Geonames[4], BBC Music[5], Library of Congress Subject Headings[6] and Ordnance Survey Linked Data[7], are attracting a large share of the inlinks. This is likely because they provide comprehensive collections of well-managed identifiers, often from an authoritative source. Providing "URI sets" for entities managed by the state is a cornerstone of the UK's national Linked Data effort [18].

We argue that these hubs are likely to emerge in any domain or community of sufficient interest, and that they are important as providers of reference identifiers that allow integration between any two datasets in their respective domain. We further argue that the overall ecosystem will work best if the hubs support data publishers in linking to them, for example by providing services that make this easier at lower cost.

We are interested in better understanding what such services could look like, and how they can be integrated into RDF conversion tools for relational databases [3, 1], XML [9] and spreadsheets [15, 12].

One important constraint on approaches to such interlinking is that they must stand a chance of actual adoption. This constraint is hard to approach scientifically. The following is probably safe to say: Services that use existing standards, protocols, and implementations, where possible, stand better chances of seeing adoption.

In this paper, we identify four such approaches, assess their suitability and discuss their limitations and associated costs. We describe their implementation in a popular data workbench application, Google Refine, and we report the results of a quantitative evaluation of the approaches against a benchmark dataset.

2. RECONCILIATION

The problem of identifying multiple representations of the same real-world object is known under many different terms: record linkage, duplicate detection, object identification, entity consolidation, co-reference detection, and so on. In the

[2]http://richard.cyganiak.de/2007/10/lod/
[3]http://dbpedia.org/About
[4]http://www.geonames.org/ontology/
[5]http://www.bbc.co.uk/music
[6]http://id.loc.gov/authorities/
[7]http://data.ordnancesurvey.co.uk/

Table 1: Example record

City	*State*	*Country*
Cambridge	Massachusetts	United States

Semantic Web field, it is usually known as instance matching and refers to identifying equivalent resources in two RDF datasets. Throughout this paper, we will use the term *reconciliation*, which we inherit from the environment that motivated this work (Google Refine) and stresses the asymmetric nature of the problem, with a dataset being linked against a well-known set of reference identifiers.

As an example, table 1 shows a record representing the city of Cambridge in Massachusetts. Our goal is to find the corresponding URI in DBpedia. Trying to reconcile it based on label comparison only might give a large number of heterogeneous results including University of Cambridge (dbpedia:University_of_Cambridge)[8] and Cambridge Bay (dbpedia:Cambridge_Bay). Adding a restriction to accept only results that are cities helps narrowing the results down but Cambridge, Ontario(dbpedia:Cambridge,_Ontario) and Cambridge, Maryland (dbpedia:Cambridge,_Maryland) will still be present in the results. Including additional properties, such as limiting the results to only those cities located in Massachusetts, helps achieving the desired result.

In principle, reconciliation is simple: compare each pair of objects using a similarity measure and apply a threshold. If a pair is more similar than the given threshold it is declared a match [7]. Given the strictness of such binary decisions and to cope better with the inherit ambiguity involved, we consider a reconciliation service response to be a ranked list of potential matching resources. Optionally, when the service is confident about a result it can mark it as an "exact match" to enable automatic reconciliation, otherwise user intervention is needed to select from the candidates (or refuse them all).

Reconciliation services must be effective and efficient [7]. Effectiveness refers to the quality of reconciliation results usually measured in terms of precision and recall while efficiency refers to the performance usually measured in terms of reconciliation time.

3. RELATED WORK

To the best of our knowledge, no RDF translator provides direct support for reusing existing URIs, or linking to them, as part of the RDF conversion process. Instead, users commonly depend on one of the following methods:

- Build URIs programmatically: if the reference dataset uses patterned URIs based on some natural keys, such as ISBNs or post codes, then these URIs can be rebuilt as part of the translation process. This requires the pattern to be known and the keys to be part of the source dataset. For example, the RDF Book Mashup[9] uses ISBN numbers in its URIs. Book Mashup URIs can be easily generated while translating any dataset containing ISBN values.

[8]with the assumption that dbpedia is the prefix for http://dbpedia.org/resource/

[9]http://sites.wiwiss.fu-berlin.de/suhl/bizer/bookmashup /index.html

- Lookup services: via querying a service for a list of URIs matching some keywords. The lookup service can be a general semantic search engine (e.g. Sindice[10]) or specific to a particular dataset (e.g. lookup services for DBpedia[11] and Geonames[12]). Some of these services provide APIs for programmatic access, but these APIs differ between services and therefore are not integrated into RDF generation tools.

- Custom code: A service for matching records in the source dataset to the corresponding URIs in a reference dataset can be built. Such services handle a specific domain and a specific use case and are of limited reusability.

Instead of re-using URIs, an alternative approach, which is most commonly used, is to mint new URIs, and then interlink them with other datasets as a second step. Movies in LinkedMDB[13] were interlinked to DBpedia and other datasets based on comparing titles. A number of approximate string matching techniques were evaluated to choose the best performing for LinkedMDB [13]. In [21], FOAF profiles were linked based on comparing `foaf:mbox_sha1sum` as it is an inverse functional property (IFP). These two examples represent work that is tailored to a specific domain and utilizes domain knowledge.

Providing domain-independent support for RDF interlinking is a challenging problem that has also attracted the attention of the Semantic Web research community. The main emphasis of the integration research has been put on matching ontological schemata [10]. Some of these tools can be applied to the task of instance matching. Broadly speaking, the focus has been on research on algorithms, with little attention to the question of interfaces for accessing matching services from client applications.

Some approaches use machine learning for RDF instance matching. For example, [22] employs supervised machine learning to reconcile FOAF profiles. These approaches still require the existence of training data or encoding some domain specific knowledge to train a classifier.

In [11], an architecture and service for storing, managing and publishing co-reference statements between entities is presented. It is orthogonal to the question of how the statements are generated. It supports applications that use co-reference information, not applications that seek to generate co-reference information. The authors have a different take on the dynamics of linking on the Web of Data. Rather than a few hubs that are linked to by many secondary datasets, they see links as hosted in services that are independent from the datasets themselves.

[16] introduces "enhancement operations" to enrich RDF data resulting from a CSV-to-RDF translator. One of these enhancement operations is "object sameAs linking" which adds `owl:sameAs` links to resources listed in a "linking file". Link discovery is limited as it is based on simple string matching. The linking files need to be prepared and the strings to be considered in comparison should be values of `dc:identifier`. Consequently, this linking approach cannot be applied against all RDF datasets.

[10]http://sindice.com/

[11]http://lookup.dbpedia.org/

[12]http://www.geonames.org/export/geonames-search.html

[13]http://www.linkedmdb.org/

Silk – Link Discovery Framework [6], is a tool for finding relationships between entities within different data sources. Silk incorporates a number of similarity measures that can be flexibly aggregated to decide on link assertions. Silk is one of the tools we consider as a reconciliation service implementation.

In the next section, we discuss a number of domain-independent services that can be used for reconciliation. We identify limitations involved and the cost associated with them.

4. APPROACHES

We consider various approaches to searching, querying and matching on RDF data that provide a well-defined API for remote access, as well as a readily available implementation that can be deployed on arbitrary RDF datasets. In the following we discuss how to employ a number of these approaches to perform reconciliation. Reconciliation requests might contain any combination of label, type and related properties constraints.

4.1 SPARQL

SPARQL [19] is a W3C Recommendation language and protocol for querying RDF data. Providing a reconciliation service using the standard SPARQL assures that any standard-compliant SPARQL endpoint can be used as a reconciliation target. SPARQL allows building queries that retrieve resources based on their labels, types and their relations. Thus, reconciliation requests can be directly translated into equivalent SPARQL queries. Nonetheless, there are two main limitations:

- No approximate matching: The only support for non-identical string comparison in SPARQL is through regular expressions. Regular expressions are very limited as a general string comparison utility, and most current SPARQL implementations show poor performance when evaluating regular expressions.
- No ranking: When querying for matching resources, SPARQL filters datasets in a set-based manner. An item is either in the result or not. Results are not ranked.

4.2 SPARQL with full-text search

Some SPARQL vendors provide full-text search extensions to SPARQL. This includes Virtuoso[14], LARQ[15] and LuceneSail[16]. Hybrid queries that combine SPARQL and full-text can provide better results than using regular expressions as they utilize matching algorithms based on IR techniques and support scoring and ranking the results.

There is no standardized syntax to express full-text search queries in SPARQL. The upcoming update of the standard, SPARQL 1.1, will not address this either.

Depending on the unit of indexing, full-text search extensions may not be able to adequately handle literals indirectly associated with resources (see Fig. 1). Search indices only consider literals directly related to resources, as extending the considered environment to include further literals in the graph adds much noise to the results. Performance of different SPARQL full-text search implementations was compared in [17].

[14]http://docs.openlinksw.com/virtuoso/sparqlextensions.html
[15]http://jena.sourceforge.net/ARQ/lucene-arq.html
[16]http://dev.nepomuk.semanticdesktop.org/wiki/LuceneSail

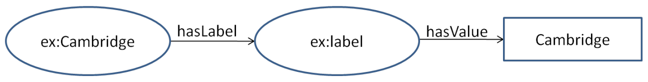


Figure 1: Indirectly related literals

4.3 Silk Server

Silk Server [14] provides a RESTful interface to interlink an RDF stream of input data against some reference dataset. It is based on the Silk Framework [6] and uses the Silk Link Specification Language (Silk-LSL) to describe the interlinking conditions. Silk-LSL is a declarative language that enables describing heuristics to be used to interlink RDF data. Silk represents the state of the art in RDF interlinking and incorporates a number of similarity measures that can be flexibly aggregated to decide on link assertions. By incorporating path expressions, Silk-LSL enables considering resource neighborhoods in the RDF graph (e.g. the case shown in Fig. 1).

Silk Server is mainly designed to be used as an identity resolution component within Linked Data applications. It is not designed as an "open" reconciliation service. Using it in that way has two problems:

- It assumes RDF input.
- It assumes that the structure of the input data is known in advance. The interlinking specification describes both the reference dataset and the input one. This tightly couples the two datasets and restricts the applicability of Silk Server as a general reconciliation service.

Both limitations can be addressed by wrapping requests into RDF according to some specific structure that also must be used in the interlinking specification used by the server (an example is provided in the next section). This is still an inflexible solution.

Unlike the SPARQL approaches, Silk Server requires that the service operator defines a link specification. This is a reasonable cost, as the operator, whom we assume to be the publisher of a hub dataset, is likely interested in providing services that maximize the dataset's usefulness.

4.4 Semantic Web Search Engines

Semantic search engines like Sindice hold crawled copies of large amounts of Web data, and offer search and query services over it. As such, they can be employed as the basis for reconciliation services.

These engines can be used to retrieve a ranked list of resource URIs matching some keywords. Further structural restrictions on the results can be included depending on the search engine's proprietary query language. It may be possible to restrict search by site or type. Each engine defines its own query language and API, which is usually limited in expressivity compared to SPARQL.

Semantic search engines are particularly useful when it is not clear what reference dataset can be used as a reconciliation target, or when the data is distributed across the web. An example is reconciling a list of Semantic Web researchers to their corresponding URIs used in their FOAF files that are distributed over the web. The results can be noisy as the search engine usually indexes large amounts of very heterogeneous data.

5. SCENARIO: GOOGLE REFINE

This section describes our motivating scenario.

Google Refine is a workbench for understanding and manipulating tabular data (e.g. CSV, TSV and Excel). It provides a set of tools that work together to help users understand their data, clean it, transform it and eventually export it in a required format. Google Refine is a Java web application available as an open source project. Unlike the common case of web applications, it is meant to be running locally on user's machine to allow handling sensitive and private data and at the same time reap the benefits of users' familiarity with interacting through web browsers.

At the core of Google Refine is its faceted browsing capabilities which help users navigate the data, understand it and select a subset to apply operations upon. Google Refine provides a set of operations scriptable using an expression language similar to JavaScript in a syntax known as Google Refine Expression Language (GREL). It also integrates a clustering engine. The rich features and capabilities of Google Refine make it an appealing option for translating tabular data into RDF.

5.1 RDF export

Google Refine does not support direct RDF export. In [8] we describe an extension that provides RDF export capabilities[17]. The export functionality is based on describing the shape of the desired RDF through a skeleton detailing what resources and literals to include in the RDF graph, what relations to set between them and what URIs to use for resources. The skeleton design is supported through a GUI shown in Figure 2. The exporter iterates through the project rows and evaluates GREL expressions in the skeleton based on cells' content to produce a subgraph corresponding to each row. The final RDF graph is the result of merging all the row subgraphs. In the designed skeleton, URIs are described as GREL expressions. The generated RDF is thus limited to URIs that can be built via simple expressions from the original data.

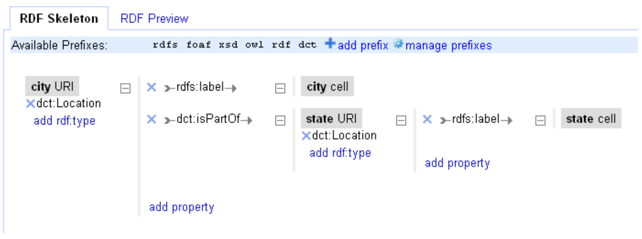


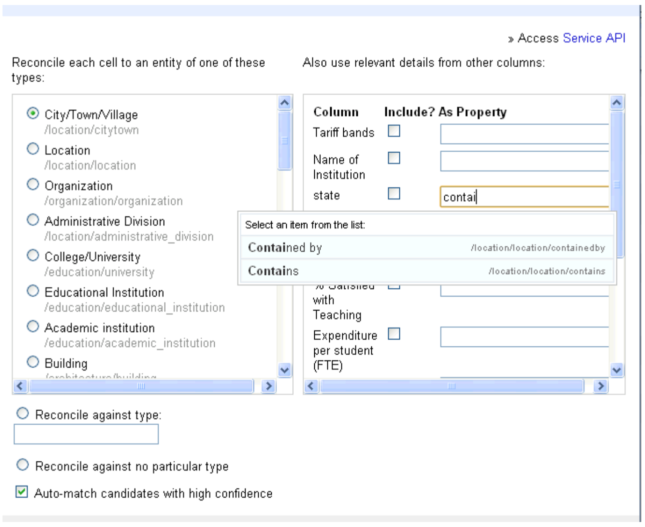
Figure 2: Simple example RDF skeleton

5.2 Freebase Reconciliation

Google Refine can reconcile values in a specific column to entities in Freebase. Freebase is a large collection of entity descriptions, curated by employees and a community of volunteers. Reconciliation to Freebase is a very useful operation as it helps mapping ambiguous textual values to precisely identified Freebase entities.

Upon reconciliation request, Google Refine starts by invoking the Freebase reconciliation service with a sample set of data values. It uses the result to guess a type for the values in the corresponding column. The list of guessed types are presented to the user who can select a specific type or continue without choosing any. The user can also choose to include additional properties in the request to help enhancing the precision of the reconciliation process. Additional properties need to be clearly identified to the reconciliation service i.e. via IDs understandable by the service. To help the user in that a reconciliation service can support autocomplete for properties search. Figure 3 shows a screenshot of the reconciliation interface resulting from reconciling a set of city names against Freebase. In Figure 3 we see that a set of types are suggested with City/Town/Village (with ID `/location/citytown`) at the top of the list. The right part of the figure shows property autocompletion in action. Proceeding with reconciliation as shown in the figure means that the set of values will be reconciled against Freebase for entities of type `/location/citytown` taken into account that the city is **`contained by`** a location matching the corresponding content of the state column in the data.

After receiving the response, the top three matching candidates for each value are presented to the user. the user can then choose to accept one of them or refuse them all. To better inform the user decision, a resource preview is available per candidate where basic information about the candidate is provided. Additionally, a numeric facet is built based on the scores of results provided by the service, allowing the user to find an acceptable threshold for the score and mass-accept or reject certain results. Figure 4 shows a screenshot where a preview for the candidate labeled "Cambridge" is presented. Results marked as a "exact match" will be automatically accepted by Google Refine without the need of user intervention.


» Access Service API

Reconcile each cell to an entity of one of these types:

[✓ City/Town/Village /location/citytown]
Location /location/location
Organization /organization/organization
Administrative Division /location/administrative_division
College/University /education/university
Educational Institution /education/educational_institution
Academic institution /education/academic_institution
Building

Also use relevant details from other columns:

Column	Include?	As Property
Tariff bands	☐	
Name of Institution	☐	
state	☐	contai
% Satisfied with Teaching	☐	
Expenditure per student (FTE)	☐	

Select an item from the list:
Contained by /location/location/containedby
Contains /location/location/contains

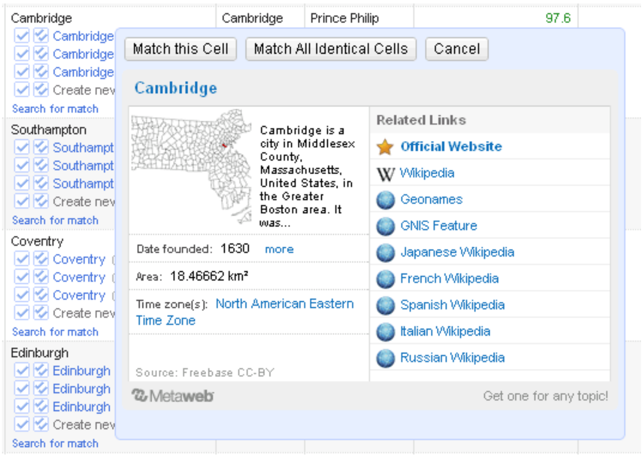
Reconcile against type:

Reconcile against no particular type

[✓ Auto-match candidates with high confidence]

Figure 3: Reconciliation interface of Google Refine

[17]available at http://lab.linkeddata.deri.ie/2010/grefine-rdf-extension/ (reconciliation extension described here is packaged in the same extension since version 0.5)



Cambridge is a city in Middlesex County, Massachusetts, United States, in the Greater Boston area. It was...

Date founded: 1630 more

Area: 18.46662 km²

Time zone(s): North American Eastern Time Zone

Source: Freebase CC-BY

Figure 4: Reconciliation candidates presentation and preview popup

Reconciling against Freebase is useful in many situations, but not all datasets are ready and suitable to be loaded into Freebase. Google Refine defines and supports a standard API to reconcile against other data sources. The standard API is an HTTP-based JSON-represented RESTful interface. Any data source that implements the defined API becomes available for reconciliation from Google Refine. Third party reconciliation services have started to appear[18].

In the RDF world, a reconciliation service over Talis Platform stores has been built[19] and used for reconciling against UK government data[20]. This depends on a Talis-specific search API[21] and does not work against other SPARQL endpoints.

Utilizing some of the already existing services and tools to query RDF would provide a general and low-cost solution for reconciling from RDF-producing applications such as Google Refine, and eliminate the need to build a service from scratch each time an RDF dataset is to be reconciled against.

6. EVALUATION

We compared the approaches described in Section 4 by evaluating them using the Instance Matching Benchmark (IM@OAEI2010)[22] which is a track of the Ontology Alignment Evaluation Initiative 2010 (OAEI2010)[23]. The benchmark provides various datasets with reference alignment to compare results to. From IM@OAEI2010 datasets, we chose the Data Interlinking track (DI) as it uses datasets which are part of the LOD cloud such as DBpedia, DailyMed[24] and Sider[25].

All the implementations described below accept requests consisting of a label, an optional type and optional list of related properties. Each related property is described as a property URI and a value. Services respond with a ranked list of matching resources along with their types and matching scores.

6.1 Implementations

In the following we describe in some detail the reconciliation service implementations.

SPARQL.

The service simply translates the request into a SPARQL query. Labels are compared using case-insensitive regular expressions, a standard SPARQL feature. By default `rdfs:label` is used for the comparison, but a service can be configured to use other properties (also more than one property can be used). Type constraint and related properties are directly translated into triple patterns used in the SPARQL query. As SPARQL query results are not ranked, we rank and score the result based on edit distance between the label of the matching resource and the request label. We used Virtuoso as the store implementation, but the choice of store has no effect on precision and recall as the results are fully determined by the SPARQL language specification.

SPARQL with full-text search.

The reconciliation query is translated into a hybrid SPARQL query (i.e. a SPARQL query with the additional constraint for text search). As each vendor uses their own syntax for the full-text search, we provide multiple implementations (currently supporting Virtuoso and LARQ). LARQ is based on Lucene[26]. We configured it to add the option of performing a similarity-based matching using N-gram. N-gram-based indices cope better with misspellings and other typographical mismatches that can sometimes be found in the datasets to be reconciled. Both LARQ and Virtuoso can return a full-text match score as part of the SPARQL query result. We use these scores to rank results.

Silk Server.

Silk Server needs to be configured so that it knows the shape of the input RDF it expects. Listings 1, 2 and 3 show how Silk Server is configured for the example presented in Section 2. Listing 1 contains a snippet of the reference dataset, Listing 2 contains a snippet of the Silk Server interlinking specification while Listing 3 shows a reconciliation request wrapped as RDF. In the three listings it is assumed that identical prefixes stand for the same namespaces. The tightly coupling of the reconciliation request (wrapped as RDF) and the interlinking specification can be seen by considering lines 3, 14 and 18 in the interlinking specification (Listing 2) that precisely describe how the input data should be structured (reflected respectively in lines 1, 2 and 3 in Listing 3). It is reasonable to have interlinking specification coupled with the reference dataset, but restricting the shape of the input data assumes that any client using Silk Server is aware of how the server interlinking is exactly defined.

[18]http://opencorporates.com/reconcile
[19]http://github.com/ldodds/pho-reconcile
[20]http://ldodds.com/gridworks/
[21]http://n2.talis.com/wiki/Contentbox
[22]http://www.instancematching.org/oaei/imei2010.html
[23]http://oaei.ontologymatching.org/2010/
[24]http://www4.wiwiss.fu-berlin.de/dailymed/
[25]http://www4.wiwiss.fu-berlin.de/sider/
[26]http://lucene.apache.org/java/docs/index.html

Listing 1: reference dataset snippet

```
ex:Cambridge rdf:type ex:City;
    rdfs:label "Cambridge";
    ex:parent ex:Massachusetts
    .

ex:Massachusetts rdf:type ex:State;
    rdfs:label "Massachusetts"
    .
```

Listing 2: Interlinking Specification snippet

```
<SourceDataset dataSource="q" var="a">
  <RestrictTo>
    ?a rdf:type ex:ReconciliationQuery .
  </RestrictTo>
</SourceDataset>
<TargetDataset dataSource="geos" var="b">
  <RestrictTo>
    ?a rdf:type ex:City .
  </RestrictTo>
</TargetDataset>
<LinkCondition>
  <Aggregate type="average">
    <Compare metric="jaroWinkler">
      <Input path="?a/rdfs:label" />
      <Input path="?b/rdfs:label" />
    </Compare>
    <Compare metric="levenshtein">
      <Input path="?a/g:location"/>
      <Input
       path="?b/ex:parent/rdfs:label"/>
    </Compare>
  </Aggregate>
</LinkCondition>
```

Listing 3: Reconciliation query wrapped as RDF

```
_:1 rdf:type ex:ReconciliationQuery;
    rdfs:label "Cambridge";
    g:location "Massachusetts"
    .
```

Sindice search API.

The Sindice search API[27] allows searching for keywords. Type constraints can also be pushed to Sindice via the search API. The result is a list of document URLs containing matching RDF data, and not a list of the actual matching resources. Thus, the documents listed in the search result must be retrieved and then examined to find the URI of the resource that caused the document to match. The documents are retrieved from the Sindice cache API[28]. LARQ (SPARQL with fulltext search) queries are then used to find the matching resource. Additional required related properties constraints are taken into account in the LARQ queries and not pushed to the Sindice API as they adversely affect the ranking according to Sindice API documentation.

6.2 Results

For the evaluation, the DBpedia and Sider datasets were treated as reference datasets to be reconciled against, and the DailyMed dump provided in the benchmark is treated as the input dataset for which appropriate reference URIs are sought. The `dailymed:name` property was used as the reconciliation label.

For DBpedia, the Virtuoso-based public SPARQL endpoint[29] was used, and tested both with standard SPARQL and with Virtuoso full-text extension. Sindice was used with a domain restriction to `dbpedia.org`. Figure 5 shows the effectiveness (precision and recall) and efficiency (reconciliation time) of reconciling the DailyMed resources against DBpedia. Reconciliation without type restriction turned out to be not feasible for SPARQL and Silk. For SPARQL, searching all of DBpedia based on a regular expression shows unacceptable response time. Silk Server, if used without type restriction, attempts to download labels for all DBpedia resources via SPARQL queries on startup, taking an unacceptable amount of time.

For Sider, the dump provided in the benchmark was loaded into a local ARQ instance. It was tested in three configurations: plain SPARQL, full-text LARQ with a default Lucene index, and full-text LARQ with an N-gram index. We didn't evaluate Sindice on the Sider dataset, because the Sider benchmark dump differs considerably from the version of Sider deployed on the Web and indexed in Sindice. Figure 6 shows the results of reconciliation against the Sider RDF dump file.

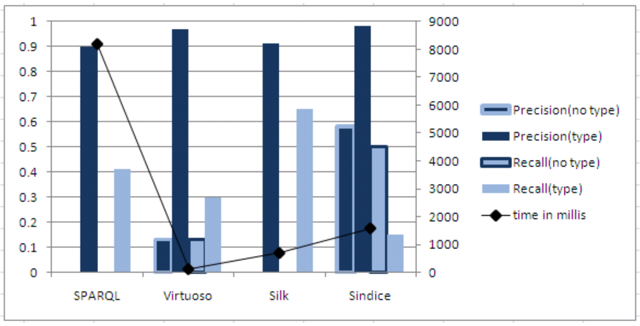


Figure 5: Services performance against DBpedia (performance time corresponds to the right axis)

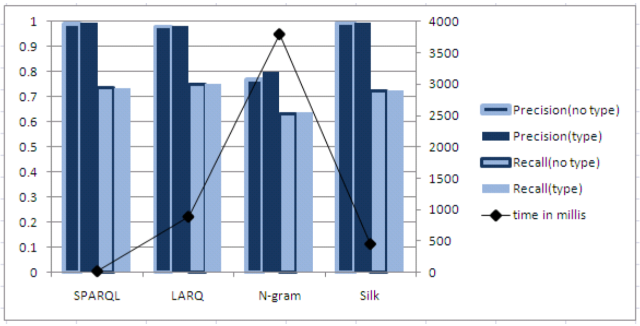


Figure 6: Services performance against Sider RDF dump file (performance time corresponds to the right axis)

[27]http://sindice.com/developers/api
[28]http://sindice.com/developers/cacheapi
[29]http://dbpedia.org/sparql

6.3 Discussion

Examining the figures we notice:

- In general Silk Server shows the best recall values.
- The relatively good results shown for standard SPARQL is mainly due to the nature of the DailyMed dataset, as most resources are drugs with precise official names. It is worth mentioning that using `rdfs:label` instead of `dailymed:name` for reconciliation requests with SPARQL approach results in empty results for all resources.
- N-gram handles typographical differences better than the default Lucene index (LARQ). However with drug names it adversely affects both precision and recall.
- Adding type restriction significantly enhances precision for datasets that cover a wide range of entities, like DBpedia. The decrease in recall resulting from adding type restriction for Sindice is mainly caused by the lack of type information in documents returned by Sindice. Most of these documents just redirect to other resources (each contains a single triple with `dbpedia-owl:redirect` predicate). This is a limitation in Sindice's data organization.

Type guessing is performed when reconciling with no type specified. The guessing is based on examining type information in the results of randomly selected ten requests. Results show that the correct type was always among the top five types suggested.

Generally, the results show the possibility to reconcile using the different approaches against an existing SPARQL endpoint and an RDF dump file with results comparable to those reported for the IM@OAEI2010[30] [23]. Results vary depending on the approach and the involved datasets. In the case of SPARQL, the approach is clearly limited by the interface rather than the application. SPARQL with regular expressions is not well-suited for the task of reconciliation.

It should be pointed out that the good performance of Silk Server comes at the cost of running the server and configuring it according to the reference dataset.

7. OUTLOOK AND CONCLUSION

We assembled the implementations described in the previous section into an extension for Google Refine (Figure 7). It enables turning a SPARQL endpoint or RDF file into a standard Google Refine reconciliation service. These services also support type and property autocomplete and resource preview. Furthermore, reconciliation can be performed against Sindice and Silk Server. All the RDF reconciliation services support type restrictions and taking related values into account. Sindice reconciliation can also be restricted to a specific domain name and the service also helps by trying to guess appropriate domains in a similar way that type guessing is performed.

Referring to Table 1, the City column can now be reconciled against DBpedia Virtuoso endpoint[31] for example. Reconciliation can be restricted to the type `http://dbpedia.org/ontology/City` and to take into consideration the State column (via the property

[30]http://www.instancematching.org/oaei/imei2010/di.html
[31]http://dbpedia.org/sparql

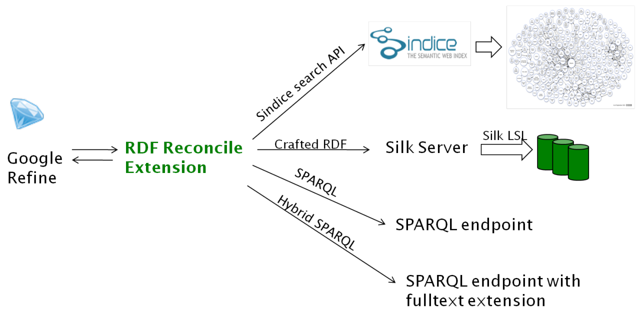


Figure 7: RDF Reconcile extension for Google Refine

`http://dbpedia.org/property/subdivisionName`). After the reconciliation process, the URIs can be used for the City node in the skeleton shown in Figure 2, overcoming the limitation mentioned in Section 5.1. It is worth mentioning that the only input needed by the user to reconcile against DBpedia is the endpoint URL. Nothing is required on the reference data side, as DBpedia supports full-text SPARQL.

Even when the data is not exported as RDF, removing the ambiguity of textual labels is an added value. Furthermore, reconciled columns can be used to join datasets and enrich them by importing additional details from the reference dataset.

With the Web of Data moving beyond the bootstrapping phase, more emphasis needs to be put on the quality and usability of the published data. In this paper we discussed enabling the reuse of existing URIs while publishing RDF data and showed an implementation integrated in a CSV-to-RDF translator. Reuse of existing URIs can help assuring good quality URIs and a better interlinked Web of Data.

8. REFERENCES

[1] S. Auer, S. Dietzold, J. Lehmann, S. Hellmann, and D. Aumueller. Triplify - Light-Weight Linked Data Publication from Relational Databases. In *18th International World Wide Web Conference*, pages 621–621, April 2009.
[2] T. Berners-Lee. Linked Data. World wide web design issues, July 2006.
[3] C. Bizer and R. Cyganiak. D2R Server - Publishing Relational Databases on the Semantic Web. Poster at the 5th International Semantic Web Conference (ISWC2006), 2006.
[4] C. Bizer, R. Cyganiak, and T. Heath. How to Publish Linked Data on the Web. Web page, 2007. Revised 2008.
[5] C. Bizer, T. Heath, and T. Berners-Lee. Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 2009.
[6] C. Bizer, J. Volz, G. Kobilarov, and M. Gaedke. Silk - A Link Discovery Framework for the Web of Data. In *18th International World Wide Web Conference*, April 2009.
[7] J. Bleiholder and F. Naumann. Data Fusion. *ACM Comput. Surv.*, 41(1), 2008.
[8] R. Cyganiak, F. Maali, and V. Peristeras. Self-service Linked Government Data with dcat and Gridworks. In *Proceedings of the 6th International Conference on*

Semantic Systems, I-SEMANTICS '10, pages 37:1–37:3, New York, NY, USA, 2010. ACM.

[9] D. V. Deursen, C. Poppe, G. Martens, E. Mannens, and R. V. d. Walle. XML to RDF Conversion: A Generic Approach. In *Proceedings of the 2008 International Conference on Automated solutions for Cross Media Content and Multi-channel Distribution*, pages 138–144, Washington, DC, USA, 2008. IEEE Computer Society.

[10] J. Euzenat and P. Shvaiko. *Ontology Matching*. Springer-Verlag, Heidelberg (DE), 2007.

[11] H. Glaser, A. Jaffri, and I. Millard. Managing Co-reference on the Semantic Web. In *WWW2009 Workshop: Linked Data on the Web (LDOW2009)*, April 2009.

[12] L. Han, T. Finin, C. Parr, J. Sachs, and A. Joshi. RDF123: From Spreadsheets to RDF. In A. Sheth, S. Staab, M. Dean, M. Paolucci, D. Maynard, T. Finin, and K. Thirunarayan, editors, *The Semantic Web - ISWC 2008*, volume 5318 of *Lecture Notes in Computer Science*, pages 451–466. Springer Berlin / Heidelberg, 2008.

[13] O. Hassanzadeh and M. P. Consens. Linked Movie Data Base. In *Proceedings of the WWW2009 workshop on Linked Data on the Web (LDOW2009)*, 2009.

[14] R. Isele, A. Jentzsch, and C. Bizer. Silk Server - Adding Missing Links while Consuming Linked Data. In *1st International Workshop on Consuming Linked Data (COLD 2010), Shanghai*, 2010.

[15] A. Langegger and W. WoB. XLWrap - Querying and Integrating Arbitrary Spreadsheets with SPARQL. In A. Bernstein, D. Karger, T. Heath, L. Feigenbaum, D. Maynard, E. Motta, and K. Thirunarayan, editors, *The Semantic Web - ISWC 2009*, volume 5823 of *Lecture Notes in Computer Science*, pages 359–374. Springer Berlin / Heidelberg, 2009. 10.1007/978-3-642-04930-9_23.

[16] T. Lebo and G. T. Williams. Converting governmental datasets into linked data. In *Proceedings of the 6th International Conference on Semantic Systems*, I-SEMANTICS '10, pages 38:1–38:3, New York, NY, USA, 2010. ACM.

[17] E. Minack, W. Siberski, and W. Nejdl. Benchmarking Fulltext Search Performance of RDF Stores. In *Proceedings of the 6th European Semantic Web Conference on The Semantic Web: Research and Applications*, ESWC 2009 Heraklion, pages 81–95, Berlin, Heidelberg, 2009. Springer-Verlag.

[18] C. Offices. Designing URI Sets for the UK Public Sector. A report from the Public Sector Information Domain of the CTO Council's cross-Government Enterprise Architecture, October 2009.

[19] E. Prud'hommeaux and A. Seaborne. SPARQL Query Language for RDF. W3C Recommendation, January 2008. `http://www.w3.org/TR/rdf-sparql-query/`.

[20] L. Sauermann and R. Cyganiak. Cool URIs for the Semantic Web. World Wide Web Consortium, Note NOTE-cooluris-20081203, December 2008.

[21] L. Shi, D. Berrueta, S. Fernández, L. Polo, S. Fernández, and A. Asturias. Smushing RDF Instances: Are Alice and Bob the Same Open Source Developer? In *ISWC2008 workshop on Personal Identification and Collaborations: Knowledge Mediation and Extraction (PICKME 2008)*, 2008.

[22] J. Sleeman and T. Finin. A Machine Learning Approach to Linking FOAF Instances. In *Proceedings of the AAAI Spring Symposium on Linked Data Meets Artificial Intelligence*. AAAI Press, January 2010.

[23] X. Zhang, Q. Zhong, F. Shi, J. Li, and J. Tang. Rimom results for oaei 2009. In *OM'09*, 2009.