Sieve: Linked Data Quality Assessment and Fusion

Pablo N. Mendes, Hannes Mühleisen, Christian Bizer
Web Based Systems Group
Freie Universität Berlin
Berlin, Germany, 14195
first.last@fu-berlin.de

ABSTRACT

The Web of Linked Data grows rapidly and already contains data originating from hundreds of data sources. The quality of data from those sources is very diverse, as values may be out of date, incomplete or incorrect. Moreover, data sources may provide conflicting values for a single real-world object.

In order for Linked Data applications to consume data from this global data space in an integrated fashion, a number of challenges have to be overcome. One of these challenges is to rate and to integrate data based on their quality. However, quality is a very subjective matter, and finding a canonic judgement that is suitable for each and every task is not feasible.

To simplify the task of consuming high-quality data, we present *Sieve*, a framework for flexibly expressing quality assessment methods as well as fusion methods. Sieve is integrated into the Linked Data Integration Framework (LDIF), which handles Data Access, Schema Mapping and Identity Resolution, all crucial preliminaries for quality assessment and fusion.

We demonstrate Sieve in a data integration scenario importing data from the English and Portuguese versions of DBpedia, and discuss how we increase completeness, conciseness and consistency through the use of our framework.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; H.2.5 [**Information Systems**]: Database Management—*Heterogeneous databases*

Keywords

Linked Data, RDF, Data Integration, Data Quality, Data Fusion, Semantic Web

1. INTRODUCTION

The Web of Linked Data has seen an exponential growth over the past five years[1]. From 12 Linked Data sets catalogued in 2007, the Linked Data cloud has grown to almost 300 data sets encompassing approximately 31 billion triples, according to the most recent survey conducted in September 2011 [10].

The information contained in each of these sources often overlaps. In fact, there are approximately 500 million explicit links between data sets [10], where each link indicates that one data set 'talks about' a data item from another data set. Further overlapping information may exist, even though no explicit links have been established yet. For instance, two data sets may use different identifiers (URIs) for the same real world objects (e.g. Bill Clinton has an identifier in the English and the Portuguese DBpedia). Similarly, two different attribute identifiers may be used for equivalent attributes (e.g. both `foaf:name` and `dbprop:name` contain the name 'Bill Clinton'.)

Applications that consume data from the Linked Data cloud are confronted with the challenge of obtaining a homogenized view of this global data space [8]. The Linked Data Integration Framework (LDIF) was created with the objective of supporting users in this task. LDIF is able to conflate multiple identifiers of the same object into a canonical URI (identity resolution), while mapping equivalent attributes and class names into a homogeneous target representation (schema mapping).

As a result of such a data integration process, multiple values for the same attribute may be observed – e.g. originating from multiple sources. For attributes that only admit one value (e.g. total area or population of a city) this represents a conflict for the consumer application to resolve. With the objective of supporting user applications in dealing with such conflicts, we created *Sieve* - Linked Data Quality Assessment and Data Fusion.

Sieve is included as a module in LDIF, and can be customized for user applications programmatically (through an open source Scala API) and through configuration parameters that describe users' task-specific needs. Sieve includes a Quality Assessment module and a Data Fusion module. The Quality Assessment module leverages user-selected metadata as quality indicators to produce quality assessment scores through user-configured scoring functions. The Data Fusion module is able to use quality scores in order to perform user-configurable conflict resolution tasks.

In this paper we demonstrate Sieve through a data integration scenario involving the internationalized editions of DBpedia, which extracts structured data from Wikipedia.

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

LWDM2012 March 26–30, 2012, Berlin, Germany

In this scenario we consume data from the English and Portuguese DBpedia editions and wish to obtain data about Brazilian municipalities that is more complete, concise, consistent and up to date than in their original sources.

In Section 2 we describe the LDIF architecture, placing the newly created Sieve modules in the larger context of data integration. Subsequently, we explain the Quality Assessment (Section 3) and Data Fusion (Section 4) modules in more detail. In Section 5 we describe our demonstration setup and in Section 6 we discuss the results. Finally, in Section 7 we make final comments and point to future directions.

2. LDIF ARCHITECTURE

The Linked Data Integration Framework (LDIF) [13] offers a modular architecture to support a wide range of applications in producing a homogenized view over heterogeneous data originating from diverse sources. The architecture of LDIF is displayed in Figure 1.

Data sets are imported into LDIF through Web Data Access Modules. Currently supported data access modules include the Triple/Quad Dump Import, which loads files encoded in all major RDF formats from the disk or the Web, a Crawler Import which relies on the dereferenceability of URIs to obtain RDF by navigating through the Web of Linked Data, and finally the SPARQL Import, which allows SQL-like queries to import data from SPARQL servers [7] on the Web.

As data is imported, its provenance or lineage is also recorded. Provenance data contains information about the history of data items, for example their origins. For provenance tracking, LDIF relies on the Named Graphs data model [6]. Information is stored as quadruples (quads) in the form <subject, predicate, object, graph>.

```
_:i0 ldif:importId "DBpedia.0" ldif:provenance .
_:i0 ldif:lastUpdate \\
     "2011-09-21T19:01:00-05:00"^^xsd:dateTime ldif:provenance .
_:i0 ldif:hasDatasource "DBpedia" ldif:provenance .
_:i0 ldif:hasImportType "quad" ldif:provenance .
_:i0 ldif:hasOriginalLocation "dbp_dump.nt.bz2" ldif:provenance .
```

Figure 2: Provenance metadata output by LDIF.

The fourth element of the quad is interpreted as a graph identifier, a way to refer to a group of triples. LDIF is agnostic to the provenance model used, allowing users to attach arbitrary provenance metadata to their named graphs. While performing import jobs, LDIF also automatically adds its own provenance metadata that record time of import, URL source, etc. as displayed in Figure 2. Please note that URIs in the figure were shortened and the second line was wrapped due to space restrictions.

Since data from multiple sources may use different vocabularies to describe overlapping information, LDIF includes a Schema Mapping step, which relies on the R2R Framework [2]. R2R alleviates schema heterogeneity by translating source schema element names, structure and values into a user-configured target representation. Through its R2R Mapping Language, the framework supports simple or more complex (1-to-n and n-to-1) transformations. Moreover, it is able to normalize different units of measurement, perform string transformations or manipulate data types and language tags.

Furthermore, since data may also include multiple identi-

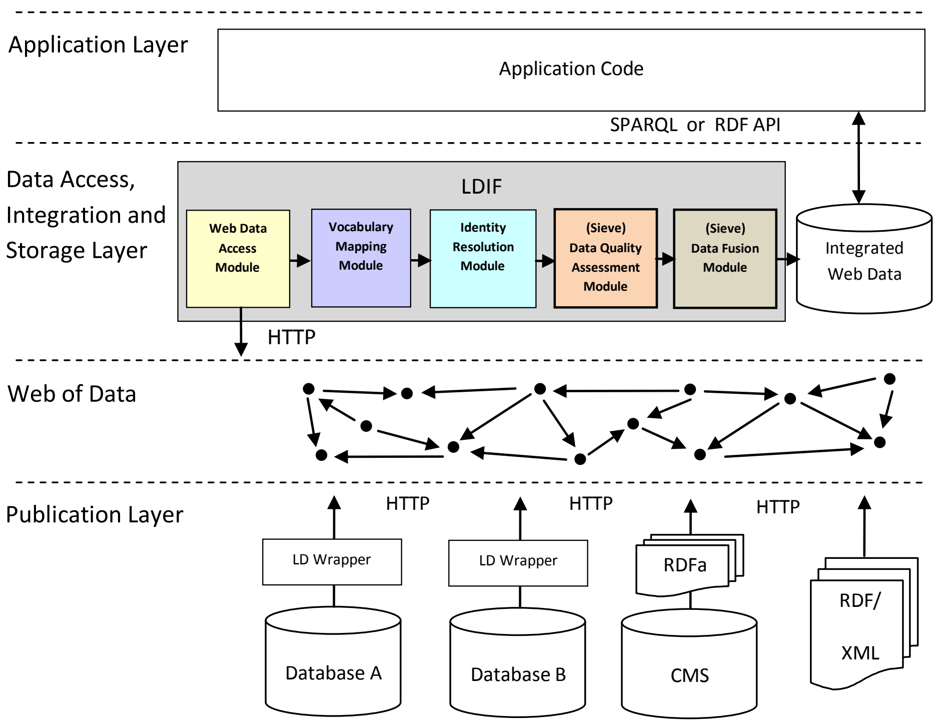

Figure 1: LDIF Architecture

fiers for the same real-world entity, LDIF also supports an Identity Resolution step through Silk [14][9]. The Silk Link Discovery Framework allows flexibility in the identity resolution heuristics through the declarative Silk - Link Specification Language (Silk-LSL). In order to specify the condition which must hold true for two entities to be considered a duplicate, the user may apply different similarity metrics to property values of an entity or related entities. The chosen similarity scores can also be combined and weighted using various similarity aggregation functions. The result of the identity resolution step is a set of equivalence links (`owl:sameAs`) between URIs.

As an additional normalization step, LDIF's URITranslator module can homogenize all URI aliases identified as duplicates by Silk, grouping all properties of those URIs into one canonical target URI. For instance, suppose that we have 3 URIs identified as duplicates, with 2 properties each. The URITranslator will be able to produce one canonical URI with 6 property values. The choice of which URI will represent the cluster of duplicates can be done through a string pattern (URI minting) or through heuristics to choose between the existing URIs. The current implementation uses a 'readability' heuristic that prefers URIs with higher proportion of letters. After replacing the original URIs by a canonical one, URITranslator adds `owl:sameAs` links pointing at the original URIs, which makes it possible for applications to refer back to the original data sources on the Web. If the LDIF input data already contains `owl:sameAs` links, the referenced URIs are also normalized accordingly.

As a result of the aforementioned data integration steps, the schema and object identifiers will be normalized according to user configuration, but the property values originating from different sources may be of low quality, redundant or conflicting. In order to alleviate this problem, we developed Sieve as a LDIF module to perform quality assessment and data fusion. Sieve assumes that data has been normalized through schema mapping and identity resolution. Therefore, as a result of this normalization process,

- if two descriptions refer to the same object in the real world, then these descriptions should have been assigned the same URI,

- if there are two property values for the same real world attribute, then these two properties should have been assigned to a unique URI with two property values.

3. DATA QUALITY

A popular definition for quality is "fitness for use" [11]. Therefore, the interpretation of the quality of some data item depends on who will use this information, and what is the task for which they intend to employ it. While one user may consider the data quality sufficient for a given task, it may not be sufficient for another task or another user. Moreover, quality is commonly perceived as multifaceted, as the "fitness for use" may depend on several dimensions such as accuracy, timeliness, completeness, relevancy, objectivity, believability, understandability, consistency, conciseness, availability, and verifiability [1].

These information quality dimensions are not independent of each other and typically only a subset of the dimensions is relevant in a specific situation. Which quality dimensions are relevant and which levels of quality are required for each dimension is determined by the specific task at hand and the subjective preferences of the information consumer [12].

In Sieve, the quality assessment task is realized through a flexible module, where the user can choose which characteristics of the data indicate higher quality, how this quality is quantified and how should it be stored in the system. This is enabled by a conceptual model composed of assessment metrics, indicators and scoring functions [1].

Assessment Metrics are procedures for measuring an information quality dimension. In our model, each assessment metric relies on a set of quality indicators and calculates an assessment score from these indicators using a scoring function.

A **Data Quality Indicator** is an aspect of a data item or data set that may give an indication to the user of the suitability of the data for some intended use. The types of information which may be used as quality indicators are very diverse. Besides the information to be assessed itself, indicators may stem from meta-information about the circumstances in which information was created, on background information about the information provider, or on ratings provided by the information consumers themselves, other information consumers, or domain experts.

A **Scoring Function** is an assessment of a data quality indicator to be evaluated by the user in the process of deciding on the suitability of the data for some intended use. There may be a choice of several scoring functions for producing a score based on a given indicator. Depending on the quality dimension to be assessed and the chosen quality indicators, scoring functions range from simple comparisons, like "assign true if the quality indicator has a value greater than X", over set functions, like "assign true if the indicator is in the set Y", aggregation functions, like "count or sum up all indicator values", to more complex statistical functions, text-analysis, or network-analysis methods.

An **Aggregate Metric** is a user-specified aggregate assessment metric built out of individual assessment metrics. These aggregations produce new assessment values through the average, sum, max, min or threshold functions applied to a set of assessment metrics. Aggregate assessment metrics are better visualized as trees, where an aggregation function is applied to the leaves and combined up the tree until a single value is obtained. The functions to be applied at each branch are specified by the users.

3.1 Sieve Quality Assessment Module

In Sieve, users have the flexibility of defining relevant indicators and respective scoring functions for their specific quality assessment task. A number of scoring functions are provided with Sieve which can be configured for a specific task through an XML file. Moreover, users can extend the current list of scoring functions and configuration options by implementing a simple programmatic interface that takes in a list of metadata values and outputs a real-valued score.

The complete list of supported scoring functions is available from the Sieve specification website[2]. In this paper we have used:

- TimeCloseness – measures the distance between the input date from the provenance graph to the current date, with more recent data receive scores closer to 1

[2] http://www4.wiwiss.fu-berlin.de/bizer/sieve/

- Preference – assigns decreasing scores to each graph URI provided in the configuration

Through the Sieve Quality Assessment Specification Language, users can define AssessmentMetric elements that use a specific ScoringFunction implementation to generate a score based on a given indicator and parameters provided in the Input element. Multiple AssessmentMetric elements can be composed into a AggregatedMetric element, yielding a score which is an aggregation of the component scores.

Listing 1 shows a quality assessment policy that outputs scores for the dimensions "Recency" and "Reputation". In the Recency dimension, Sieve will use the TimeCloseness function in order to measure the distance between two dates: a date input as an indicator, and the current date. In this case, the configuration used the property lastUpdated as indicator (line 6), and a range parameter (in days) to normalize the scores. The function outputs a score between 0 and 1, where values closer to 1 indicate that the two dates are very close. The computed score will be output as value for the sieve:recency property. Consequently, the more recently updated graphs are ranked higher by this function. In the Reputation dimension, Sieve will take a parameter list (line 11) with a space-separated list of graphs to trust. These graphs will be scored by the function Preference from higher to lower priority in order of appearance in the list parameter. Consequently, in the example at hand, values originating from the Portuguese DBpedia version will take higher priority over those originating from the English version (line 12).

```
1  <Sieve>
2    <QualityAssessment>
3      <AssessmentMetric id="sieve:recency">
4      <ScoringFunction class="TimeCloseness">
5          <Param name="timeSpan" value="7"/>
6          <Input path="?GRAPH/provenance:lastUpdated"/>
7      </ScoringFunction>
8      </AssessmentMetric>
9      <AssessmentMetric id="sieve:reputation">
10       <ScoringFunction class="ScoredList">
11         <Param name="priority"
12             value="http://pt.wikipedia.org http://en.
                   wikipedia.org"/>
13       </ScoringFunction>
14     </AssessmentMetric>
15   </QualityAssessment>
16 </Sieve>
```

Listing 1: Sieve Data Quality Assessment Specification Example

Users can also aggregate multiple scores into a new aggregated metric. In our running example, one could have chosen to aggregate sieve:recency and sieve:reputation into an indicator, say, sieve:believability which may use the function TieBreaker(sieve:recency, sieve:reputation) to rank scores by recency first, and in case two articles have the same recency, assign a higher score to the most reputable source. We refer the reader to a detailed explanation of the quality assessment specification language at our project website[2].

The output of the quality assessment module is a set of quads, where the calculated scores are associated with each graph. An example is shown in Figure 3, where enwiki:Juiz_de_Fora is the URI for a graph grouping all triples extracted from the English Wikipedia page for the Brazilian municipality of Juiz de Fora. These scores represent the user-configured interpretation of quality and can be consumed downstream by applications that will rank, filter or transform data based on their judged quality. For instance, these scores will be used subsequently by the Data Fusion module when deciding how to fuse quads based on the metadata associated with each graph.

```
enwiki:Juiz_de_Fora sieve:recency "0.4" ldif:provenance .
ptwiki:Juiz_de_Fora sieve:recency "0.8" ldif:provenance .
enwiki:Juiz_de_Fora sieve:reputation "0.9" ldif:provenance .
ptwiki:Juiz_de_Fora sieve:reputation "0.45" ldif:provenance .
```

Figure 3: Quality assessment metadata output by Sieve

4. DATA FUSION

In the context of data integration, Data Fusion is defined as the "process of fusing multiple records representing the same real-world object into a single, consistent, and clean representation" [5]. Data Fusion is commonly seen as a third step following schema mapping and identity resolution, as a way to deal with conflicts that either already existed in the original sources or were generated by integrating them. For instance, by mapping two equivalent attributes from different schemata, the system may generate one canonical attribute with two different values. Similarly, the identity resolution step may collapse two object identifiers into one, requiring that applications deal with the multiple attribute values originating from each data source.

Our data fusion framework is inspired by the work of Bleiholder and Naumann [3]. They described a framework for data fusion in the context of relational databases that includes three major categories of conflict handling strategies:

- Conflict-ignoring strategies, which defer conflict resolution to the user. For instance, the strategy PassItOn simply relays conflicts to the user or application consuming integrated data.
- Conflict-avoiding strategies, which apply a unique decision to all data. For instance, the strategy TrustYourFriends prefers data from specific data sources.
- Conflict-resolution strategies, which decide between existing data (e.g. KeepUpToDate, which takes the most recent value), or mediate the creation of a new value from the existing ones (e.g. Average).

In contrast to their framework, we provide a stricter separation of data quality assessment functions and fusion functions. In our framework, the function TrustYourFriends combines two aspects: One aspect of quality assessment: the assignment of higher 'reputation' scores to some sources; and one aspect of data fusion: prefer values with highest scores in a given indicator (in this case, reputation). Similarly, the fusion function KeepUpToDate can be expressed in our framework by preferring values with higher scores in the "Recency" indicator.

In Sieve, fusion functions are basically of two types. Filter Functions (deciding strategies) remove some or all values from the input, according to some quality metric. One example filter function is: keep the value with the highest score for a given metric. Transform Functions (mediating strategies) operate over each value in the input, generating a new list of values built from the initially provided ones. A wide

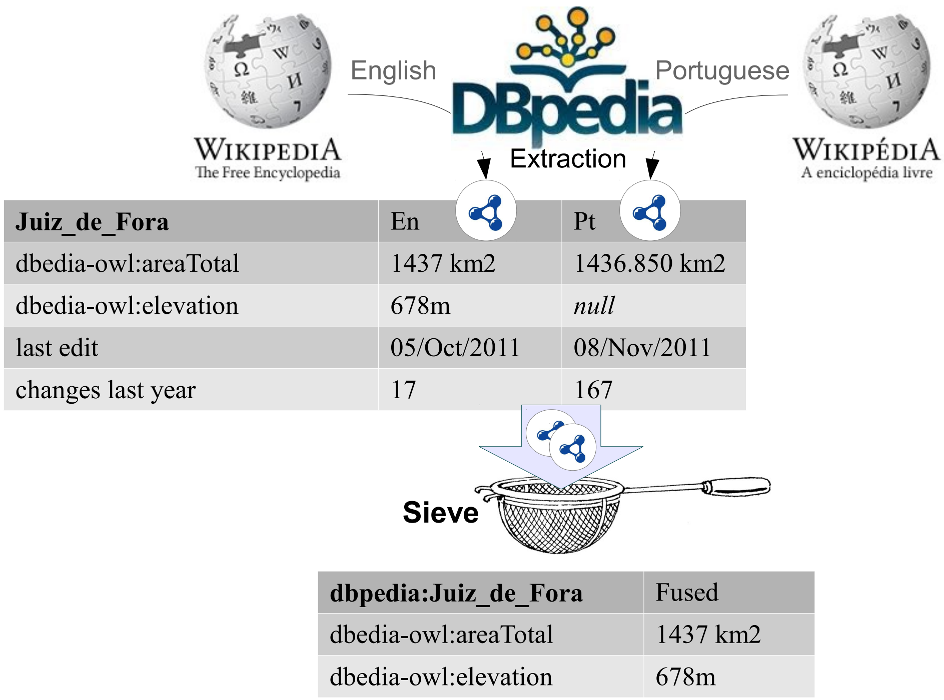

Figure 4: Illustration of the Data Fusion process. Example data originating from the DBpedia Extraction from the English and Portuguese Wikipedia editions are sent through Sieve, generating a cleaner representation.

range of fusion functions have been proposed [4], and are being implemented in Sieve, including:

- Filter – removes all values for which the input quality assessment metric is below a given threshold
- KeepSingleValueByQualityScore – keeps only the value with the highest quality assessment
- Average, Max, Min – takes the average, chooses the maximum, or minimum of all input values for a given numeric property
- First, Last, Random – takes the first, last or the element at some random position for a given property
- PickMostFrequent – selects the value that appears more frequently in the list of conflicting values

The complete list of supported fusion functions is available from the Sieve specification website[2].

4.1 Sieve Data Fusion Module

Similarly to the quality assessment module, the data fusion module can also be configured through XML. The Sieve Data Fusion Specification language takes a property-centric perspective. The user has the flexibility of deciding what action to perform for each property of a class that requires data fusion. Actions range from ignoring conflicts (e.g. PassItOn) to filtering out information (e.g. Filter) based on quality indicators, and can include also value transformations (e.g. Average).

The quality indicators used for deciding on which data fusion operation to perform are provided by the quality assessment module. The data fusion component takes as input a set of metadata quads (containing quality information), a set of entity descriptions containing the properties to be fused and a data fusion specification.

```
1  <Sieve>
2    <Fusion>
3      <Class name="dbpedia:Settlement">
4        <Property name="rdfs:label">
5          <FusionFunction class="PassItOn"/>
6        </Property>
7        <Property name="dbpedia-owl:areaTotal">
8          <FusionFunction class="
               KeepSingleValueByQualityScore"
9                            metric="sieve:reputation"/>
10       </Property>
11       <Property name="dbpedia-owl:populationTotal">
12         <FusionFunction class="
               KeepSingleValueByQualityScore"
13                           metric="sieve:recency"/>
14       </Property>
15     </Class>
16   </Fusion>
17 </Sieve>
```

Listing 2: Sieve Data Fusion Specification Example

Listing 2 shows a data fusion specification that acts on every instance of dbpedia:Settlement and consolidates information for the properties areaTotal and populationTotal from the DBpedia Ontology. Both properties use the fusion function KeepSingleValueByQualityScore, which removes all but the highest ranked value, where the ranking is determined by a quality indicator calculated in the previous step. The fusion for the areaTotal property takes the value with the highest ldif:reputation (line 9), while populationTotal takes the value with the highest ldif:recency (line 13). Values for rdfs:label are simply repeated to the output, as multiple values for this property are often useful for language customiza-

Table 1: Properties and their occurrences for Brazilian municipalities in different DBpedia editions

property	*only en*	*only pt*	*redundant*	*conflicting*
areaTotal	2/5565	3562/5565	27/5565	378/5565
foundingDate	234/5565	58/5565	1/5565	0/5565
populationTotal	5/5565	3552/5565	47/5565	370/5565

tion in client applications.

5. DEMONSTRATION

In order to demonstrate the capabilities of Sieve, we have collected data about municipalities in Brazil from different Wikipedia editions, namely the English- and Portuguese-language Wikipedia editions. According to the Brazilian Institute for Geography and Statistics, there are 5,565 municipalities in Brazil[3]. However, this information may be absent from DBpedia due to incomplete or missing Wikipedia infoboxes, missing mappings in the DBpedia Mapping Wiki or irregularities in the data format that were not resolved by the DBpedia Extraction Framework. Furthermore, a particular Wikipedia edition may be out of date or incorrect. See Figure 4 for the schematics of our demonstration.

In this section we show how such problems can be alleviated by fusing data from multiple sources. We start by describing the data sets employed, and subsequently present the results obtained for (slight modifications of) the specifications presented in Listing 1 and Listing 2.

5.1 Data Sets

The data sets employed in our use case were obtained from the English and Portuguese dumps of DBpedia 3.7 [4]. We collected mapping-based properties, inter-language links, instance types and provenance information, which were then fed as quads into Sieve through LDIF.

Target attributes.

For the sake of simplicity of explanation, we will focus our discussion on a few properties. Namely, we have collected the total population, total area and the founding date of each municipality. Table 4.1 shows the distribution of values that occur only in DBpedia English but not in DBpedia Portuguese (second column: *only en*), and vice-versa (third column: *only pt*). The table also shows properties that occurred in both DBpedia dumps (fourth column: *redundant*) with the same values, and properties that occurred in both data sets, but with different values (fifth column: *conflicting*).

Inter-language Links.

Wikipedia pages offer links to other language editions in the left-hand side menu in each article. We collected those links and represented them as `owl:sameAs` links that were provided as input to the LDIF engine. Through the identity resolution module (Silk) and the URI translation module, these identity links will be used to merge object descriptions into one URI per object.

Instance Types.

Wikipedia editors commonly create so called "list pages", which are used to organize collections of links to pages matching certain criteria. We have used a page listing all Brazilian municipalities[5] to derive an extra source of instance type and country location from the Wikipedia page. We used this set as our target universe, that is, the complete set of URIs to be obtained after integration.

Pre-processing provenance..

The DBpedia Extraction Framework tracks the source article from which each property was extracted. For the purpose of this demonstration, we normalized the provenance information from the extraction framework to associate every property value extracted to its original source page in Wikipedia. For each source page, we collected the last update from the Wikipedia dump, and included it in the LDIF provenance graph.

All of the data collected was serialized in the NQuads format and fed into LDIF for processing, resulting in one integrated NQuads output file. The next section discusses the results of this integration process.

6. RESULTS

Data Integration is commonly applied in order to increase data quality along at least three dimensions: completeness, conciseness and consistency [5].

Completeness.

On the schema level, a data set is complete if it contains all of the attributes needed for a given task. On the data (instance) level, a data set is complete if it contains all of the necessary objects for a given task. Naturally, the completeness of a data set can only be judged in the presence of a task where the ideal set of attributes and objects are known.

In the use case described in this paper, the task required retrieving 3 attributes (areaTotal, foundingDate, populationTotal) for all 5565 objects (Brazilian municipalities). According to Bleiholder and Naumann [5], the *extensional completeness* (data level), can be measured in terms of the proportion of target URIs found in the output (Equation 1), while the *intensional completeness* (schema level) can be measured by the proportion of target properties found in the output (Equation 2).

$$extensional\ completeness = \frac{||uniq.\ obj.\ in\ data\ set||}{||all\ uniq.\ obj.\ in\ universe||} \quad (1)$$

$$intensional\ completeness = \frac{||uniq.\ attr.\ in\ data\ set||}{||all\ uniq.\ attr.\ in\ universe||} \quad (2)$$

[3] `http://www.ibge.gov.br/cidadesat`
[4] `http://dbpedia.org/Downloads37`

[5] `http://pt.wikipedia.org/wiki/Anexo:Lista_de_municípios_do_Brasil`

Table 2: **Impact of the data integration process in quality indicators.**

Property p	**Completeness**(p)			**Conciseness**(p)	**Consistency**(p)
	en	pt	final	gain	gain
`areaTotal`	7.31%	71.28%	71.32%	+10.20%	+9.52%
`foundingDate`	4.22%	1.06%	5.27%	+0.34%	-
`populationTotal`	7.58%	71.32%	71.41%	+10.49%	+9.31%

For the original data sets, the English DBpedia contained 2097/5565 municipalities, which means that the extensional completeness was 37% before the integration process, while DBpedia Portuguese contained 3970/5565 municipalities (extensional completeness of 71%). After integration, 3979/5565 municipalities were found, increasing extensional completeness in 36% and 0.5% for the DBpedia English and DBpedia Portuguese respectively.

In order to provide a more fine-grained analysis of the impact of the integrated data set with regard to the original sources, we have defined another measure of *completeness* that takes into consideration the instantiations of properties. That is, it measures the proportion of objects that contain a value for a given property p, in relation to the universe of objects (Equation 3).

$$Completeness(p) = \frac{||obj.\ with\ property\ p\ in\ data\ set||}{||all\ uniq.\ obj.\ in\ universe||} \quad (3)$$

The $Completeness(p)$ for both English and Portuguese-language DBpedia editions are shown on Table 2 for each property in our use case (`areaTotal`, `foundingDate` and `populationTotal`. The percentages shown represent the completeness of DBpedia English before integration (en), DBpedia Portuguese before integration (pt), and completeness after integration (final). As expected, the integration process increased completeness for both data sets, with particularly high increase (more than 9x) for DBpedia English in the properties `areaTotal` and `populationTotal`. The property `foundingDate` was actually more complete in DBpedia English, and provided an increase of roughly 4% in completeness for DBpedia Portuguese.

Conciseness.

On the schema level, a data set is concise if it does not contain redundant attributes (two equivalent attributes with different names). On the data (instance) level, a data set is concise if it does not contain redundant objects (two equivalent objects with different identifiers). The *extensional conciseness* measures the number of unique objects in relation to the overall number of object representations in the data set [5]. Similarly, *intensional conciseness* measures the number of unique attributes of a dataset in relation to the overall number of attributes in a target schema [5]. The intensional conciseness in our use case was 1, since both datasets used the same schema. The extensional conciseness was also 1, since both DBpedia editions only contained one URI per object. Similarly to the case of completeness, we have defined a finer grained *conciseness* metric for a given property p to measure the proportion of objects that do not contain more than one **identical** value for p (redundant), with regard to the universe of unique property values (Equation 4).

$$Conciseness(p) = \frac{||obj.\ with\ uniq.\ values\ for\ p\ in\ data\ set||}{||all\ uniq.\ obj.\ with\ p\ in\ dataset||} \quad (4)$$

The increase in $Conciseness(p)$ for each of the properties p in our use case is shown on Table 2. The properties `areaTotal` and `populationTotal` were 89.80% and 89.51% concise, respectively, while `foundingDate` was 99.66% concise. The integration process yielded an increase in conciseness of roughly 10% for `areaTotal` and `populationTotal`, with only minor increase for `foundingDate` which was already very concise in the original datasets.

Consistency.

A data set is consistent if it is free of conflicting information. In the context of this paper, the **consistency** of a data set is measured by considering properties with cardinality 1 that contain more than one (distinct) value.

We have defined the *consistency* of a data set for a given property p to measure the proportion of objects that do not contain more than one **distinct** value for p, with regard to the universe of unique property values.

$$Consistency(p) = \frac{||obj.\ without\ conflicts\ for\ p\ in\ data\ set||}{||all\ uniq.\ obj.\ with\ p\ in\ dataset||} \quad (5)$$

The increase in $Consistency(p)$ for each of the properties p in our use case is shown on Table 2. The property `foundingDate` had no conflicts observed in the original data. However, we observed an increase in consistency of roughly 10% for `areaTotal` and `populationTotal` which were 90.48% and 90.69% consistent in the original data.

7. CONCLUSION

We have described *Sieve*, a Linked Data Quality Assessment and Data Fusion module. Sieve is employed by the Linked Data Integration Framework (LDIF) after the Schema Mapping and Identity Resolution steps. Sieve's role is to assess the quality of the integrated data and subsequently decide on which values to keep, discard or transform according to user-configured quality assessment metrics and fusion functions. Sieve is agnostic to provenance and quality vocabularies, allowing users to configure which metadata to read, and which functions to apply via a declarative specification language.

Through a use case that imported data about Brazilian municipalities from international DBpedia editions, we have demonstrated the usage of Sieve in a simple scenario that yielded an integrated data set that was more complete, concise, consistent and up to date than the original sources. The English DBpedia, although considered the most mature of the international DBpedia editions, did not have a particularly high coverage of Brazilian municipalities, and benefited from the higher coverage offered by the Portuguese DBpedia for those particular items. Furthermore, as the population of a city is prone to change, it is important to keep the most recent values. Blending values from multiple DBpedia editions allowed us to include the most recent value among the sources. However, identity resolution and schema mapping introduced multiple values for the same properties (values originating from different sources). The application of data fusion through Sieve allowed us to remove redundant and conflicting values, increasing the conciseness and the consistency of the data set.

Future work includes the development of more quality assessment scoring functions and data fusion functions, as well as performance and scalability experiments.

8. ACKNOWLEDGEMENTS

The authors would like to thank Andreas Schultz, Robert Isele and Anja Jentzsch for their valuable comments on this manuscript. We also thank them, as well as Andrea Matteini and Christian Becker for their work on other components of LDIF.

This work was supported by the EU FP7 grants LOD2 - Creating Knowledge out of Interlinked Data (Grant No. 257943) and PlanetData - A European Network of Excellence on Large-Scale Data Management (Grant No. 257641).

9. REFERENCES

[1] C. Bizer and R. Cyganiak. Quality-driven information filtering using the wiqa policy framework. *Web Semant.*, 7:1–10, January 2009.

[2] C. Bizer and A. Schultz. The R2R Framework : Publishing and discovering mappings on the web. *Work*, page 19, 2010.

[3] J. Bleiholder and F. Naumann. Declarative Data Fusion: Syntax, Semantics, and Implementation. pages 58–73. 2005.

[4] J. Bleiholder and F. Naumann. Conflict handling strategies in an integrated information system. In *Proceedings of the International Workshop on Information Integration on the Web (IIWeb)*, Edinburgh, UK, 0 2006.

[5] J. Bleiholder and F. Naumann. Data fusion. *ACM Comput. Surv.*, 41:1:1–1:41, January 2009.

[6] J. J. Carroll, C. Bizer, P. J. Hayes, and P. Stickler. Named graphs. *J. Web Sem.*, 3(4):247–267, 2005.

[7] K. G. Clark, L. Feigenbaum, and E. Torres. SPARQL Protocol for RDF. January 2008.

[8] T. Heath and C. Bizer. *Linked data: evolving the web into a global data space*. Morgan and Claypool, [San Rafael, Calif.], 2011.

[9] 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.

[10] A. Jentzsch, C. Bizer, and R. Cyganiak. State of the LOD Cloud, September 2011.

[11] J. Juran. *The Quality Control Handbook.* McGraw-Hill, New York, 3rd edition, 1974.

[12] F. Naumann. *Quality-Driven Query Answering for Integrated Information Systems.* Springer, Berlin Heidelberg New York, 2002.

[13] A. Schultz, A. Matteini, R. Isele, C. Bizer, and C. Becker. Ldif : Linked data integration framework. 2011.

[14] J. Volz, C. Bizer, M. Gaedke, and G. Kobilarov. Discovering and maintaining links on the web of data. In *The Semantic Web – ISWC 2009: 8th International Semantic Web Conference, Chantilly, VA, USA*, pages 650–665. 2009.