Performing Object Consolidation on the Semantic Web Data Graph*

Aidan Hogan
National University of Ireland, Galway
Digital Enterprise Research Institute
Galway, Ireland
aidan.hogan@deri.org

Andreas Harth
National University of Ireland, Galway
Digital Enterprise Research Institute
Galway, Ireland
andreas.harth@deri.org

Stefan Decker
National University of Ireland, Galway
Digital Enterprise Research Institute
Galway, Ireland
stefan.decker@deri.org

ABSTRACT

An important aspect of Semantic Web technologies is the issue of identity and uniquely identifying resources, which is essential for integrating data across sources. Currently, there is poor agreement on the use of common URIs for the same instances across sources and as a result a naively integrated dataset might miss associations between resources. To solve the problem, we present a method for performing large-scale object consolidation to merge identifiers of equivalent instances occurring across data sources; the algorithm is based on the analysis of defined inverse functional properties: properties which have values unique to an instance. We apply our object consolidation algorithm to a dataset of over 72M instances collected from more than 3M Web documents, and offer evaluation and discussion on the resulting integrated graph.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing

General Terms

Algorithms, Design

Keywords

object consolidation, semantic web, entity-centric, smushing

1. INTRODUCTION

A number of prominent ontologies have emerged on the Semantic Web as the de facto choice for describing data in specific domains. Examples are Friend of a Friend[1] (FOAF) for describing people, Semantically-Interlinked Online Communities[2] (SIOC) for specifying online communities, and Simple Knowledge Organisation System[3] (SKOS) for encoding classification schemes and thesauri. Classes and properties in these specifications are identified via common identifiers in the form of URIs[4] which are consistently used across sources. On the instance level, however, there is still little agreement between sources on the use of common URIs to identify specific entities, such as people, forums, or categories. In fact, since the assignment of URIs to instances is optional within the Resource Description Framework (RDF), many entities are described anonymously without use of a URI. Where agreement on URIs for resources cannot be reached, multiple URIs may exist for the one resource.

The problem of object consolidation has received significant attention in the database community under the names of record linkage, instance fusion, and duplicate identification. Due to the lack of formal specification for determining equivalences, these approaches are mostly concerned with probabilistic methods. In more formal approaches, properties unique to an entity can be used to determine its identity; examples are personal public services numbers or email addresses which can be used to identify people. On the Semantic Web, these "inverse functional properties" are specified in Web Ontology Language (OWL) descriptions. For example, from the FOAF specification one can determine that `foaf:mbox` (a person's email address) is a property unique to a person; hence, two instances sharing the same value for `foaf:mbox` properties must be the same real-world entity.

There is much more agreement on values that are established in real-world areas or widely used applications such as email addresses or instant messaging usernames than on seemingly artificial URIs as the identifying factor of entities. Thus, equivalence of identifiers based on inverse functional properties has to be taken into account when merging RDF graphs, since infrequent reuse of instance identifiers across sources leads to problematic data integration: the total knowledge contribution for an entity may be fragmented over multiple instances. One desired outcome of the Semantic Web effort is a massive data graph spanning the Web, and agreement on identifiers is crucial to achieving a well-connected graph where associations between entities are recorded correctly. Fusing identifiers is especially important for entity-centric applications relying on RDF data,

*This work has been supported by Science Foundation Ireland (SFI/02/CE1/I131)

[1] `http://www.foaf-project.org/`

[2] `http://www.sioc-project.org/`

[3] `http://www.w3.org/2004/02/skos/`

[4] `http://www.ietf.org/rfc/rfc2396.txt`

ACM 0-89791-88-6/97/05.

such as search and query engines, interactive browsing tools and data mining systems.

There are a number of challenges related to performing object consolidation on Semantic Web data:

- The Web is massive in size, therefore methods operating on considerable fractions of the Web require scalable algorithms. In particular, we require an algorithm that scales object consolidation to the Web.
- Data gathered from the Web has been created by a large number of people and exhibits a lot of variance in terms of quality and completeness. Therefore, we require our algorithm to be robust in the face of potentially problematic data.
- We wish to reduce the number of URIs denoting the same entity to a single canonical URI. We require a method to make such a decision through various analyses on the use of the candidate idenitifiers in the data.

In this paper, we describe an object consolidation algorithm which analyses inverse functional properties and is used to identify and merge equivalent instances in an RDF dataset with more than 400M statements obtained from over 3M sources. Our approach consists of the following steps: firstly, we determine a list of inverse functional properties from ontologies that are used to describe instances in the dataset. Secondly, we determine the instances that are equal based on the values of the inverse functional properties. Thirdly, we store the transitive closure of the equivalences in a data structure for efficient lookup. Finally, we determine a canonical URI for each equivalence chain, scan the dataset and rewrite identifiers.

The contributions of this paper are as follows:

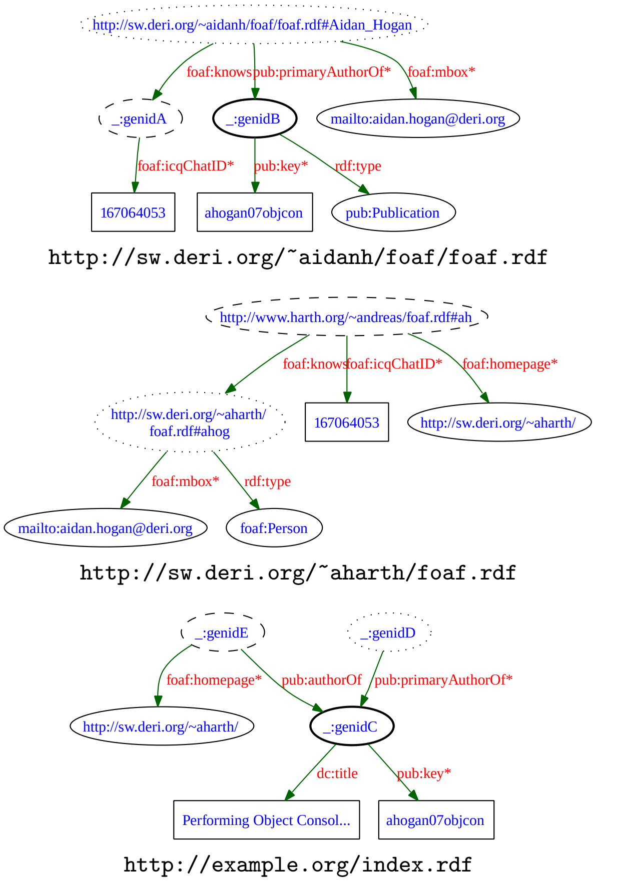
- We describe a method to perform object consolidation on a large Web dataset, and determine multiple alternatives to selecting a canonical URI from a list of choices.
- We provide an empirical analysis on the RDF data currently available online, and measure the effects of object consolidation on the dataset.

The structure of the paper is as such: Section 2 introduces a running example used in the paper. Section 3 provides necessary definitions. Section 4 describes our object consolidation algorithm for identifying and merging equivalent instances. Section 5 provides a before/after characterisation of our large Semantic Web dataset accompanied by a runtime performance evaluation of the algorithm. Section 6 discusses related work, and Section 7 concludes.

2. RUNNING EXAMPLE

To motivate and illustrate the problem, we introduce in the following some example data that will be referred to throughout the text. Note that for brevity we use local prefixes to abbreviate schema URIs for RDF, OWL, Friend of a Friend, and Dublin Core vocabularies, i.e., please interpret `foaf:Person` as the URL `<http://xmlns.com/foaf/0.1/Person>`. In addition, we introduce a new namespace `pub`[5]. Also, please note that we may abbreviate the concept `owl:InverseFunctionalProperty` to `owl:IFP`.

The example graphs in Figure 1 show partial RDF descriptions for this paper and its two authors. The three data graphs are obtained from three distinct sources, namely `http://sw.deri.org/~aidanh/foaf/foaf.rdf`, `http://sw.deri.org/~aharth/foaf.rdf` and also `http://example.org/index.rdf`. Instances with different identifiers but referencing the same real-world entity are denoted in dotted, dashed, and bold line patterns; instances representing the same entity are denoted using the same pattern.



`http://sw.deri.org/~aidanh/foaf/foaf.rdf`

`http://sw.deri.org/~aharth/foaf.rdf`

`http://example.org/index.rdf`

Figure 1: Three sample data graphs from three different sources. Inverse functional properties are depicted with an asterisk (*). Identifiers denoting the same entity are denoted by a common line style.

In addition to the instance data graph, we assume that we have knowledge about which properties are declared as inverse functional in their respective ontology specification document. For the example, our supposition is that we know of the following inverse functional properties: `pub:key` relates a document to a unique key; `foaf:mbox`, `foaf:homepage` and `foaf:icqChatID` relate a person to their personal email address, ICQ username, and homepage; and finally, the property `pub:primaryAuthorOf` relates a primary author to the respective paper. Please observe that Figure 1 adds highlighting (line style and asterisk *) for illustration purposes only; information about equivalences and schema cannot be found in the original instance data.

Given the instance information together with information obtained from the ontology specification, we can deduce a number of identifier equalities. Table 1 shows the inferred

[5] `http://sw.deri.org/2006/11/research/publications.rdfs#`

http://www.harth.org/~andreas/foaf.rdf#ah = genidA = genidE
http://sw.deri.org/~aidanh/...#Aidan_Hogan = http://sw.deri.org/~aharth/foaf.rdf#ahog = genidD
genidB = genidC

Table 1: Identifiers determined as being equal based on matching `owl:IFP` values.

equalities.

Ideally, we would like to fuse the identifiers of the equal instances to arrive at a graph where associations between real world entities are expressed correctly. Figure 2 shows the output of applying our object consolidation algorithm to the data in Figure 1.

The rest of the paper is concerned with describing an algorithm which starts with a large-scale instance graph of data as exemplified in Figure 1, deduces equalities and manages them efficiently, replaces equivalent identifiers with a canonical one and arrives at a consolidated data graph as illustrated in Figure 2.

3. PREREQUISITES

In the following section, we briefly describe the data indexing we employ but firstly, we introduce the data format used for representing the data graph.

We assume the reader is familiar with the basic notions of RDF [10]. We use an extension of the RDF N-Triples format for the description of our data. Namely, we extend the classical triple model (subject, predicate, object) with context. We use context to encode the URL of the data source from hence a triple originated. Whereas individual statements in RDF N-Triples are called triples, we call triples with context, "quadruples" or "quads". We call the data format used "N-Quads". Context is used as a means of tracking the provenance of data, which is an important aspect in evaluating object consolidation and how data is integrated from different sources. Context can also be analysed in the decision of which canonical URI to use, as discussed in Section 4.4. What follows is a formal definition of the concepts of a triple and quadruple.

Definition 3.1. *(RDF Triple) Given a set of URI references $\mathcal{R}$, a set of blank nodes IDs $\mathcal{B}$, and a set of literals $\mathcal{L}$, a triple (s, p, o) $\in (\mathcal{R} \cup \mathcal{B}) \times \mathcal{R} \times (\mathcal{R} \cup \mathcal{B} \cup \mathcal{L})$ is called an RDF triple.*

Definition 3.2. *(Quadruple) A pair (t, c) with a triple t and c $\in (\mathcal{R})$ is called a quadruple or quad (s, p, o, c). For a quad q, $s(q)$ denotes the subject of the quad, $p(q)$ denotes the predicate, $o(q)$ denotes the object and $c(q)$ denotes the context.*

For the purposes of indexing, we store quadruples or quads in sorted, blocked, compressed files on-disk. Lookups can be carried out as index scans or via binary search. For the purpose of our object consolidation algorithm, we re-configure quads from their natural ordering of subject, predicate, object, context (SPOC) to predicate, object, context, subject (POCS). We create an on-disk POCS index over these re-ordered quads and sort them respective to the POCS order. The POCS index created for the purpose of this work is distributed according to a hash function over the predicate of each quad. Further details of the distributed index are outside the scope of this paper; the index can be abstracted as an on-disk data structure containing a list of sorted quads in POCS ordering.

4. ALGORITHM

The following section discusses the operation of the object consolidation algorithm used to merge equivalent instances. The algorithm is run pre-query-time and the results are materialised in the index. Therefore, we avoid possible issues of the object consolidation algorithm affecting query response times. Also, we can guarantee complete consolidation through multiple iterations which could be too expensive for a query-time algorithm.

We describe a once-off object consolidation algorithm which operates on the data present in an index.

Specifically, in this section we describe

- our method for achieving a list of inverse functional properties
- our algorithm for finding equivalences
- our data structure for storing equivalences in memory
- the process of picking the identifiers to consolidate to
- our method of rewriting the index (materialising equivalences)
- the possible iterative nature of the algorithm

4.1 Obtaining Ontologies

Object consolidation for RDF data involves analysis of inverse functional properties and their values. Properties are defined as being inverse functional in their respective ontologies. Thus, prior to object consolidation, to achieve a list of properties which are inverse functional, we must acquire the available ontologies which describe properties in the dataset.

To achieve the ontology data applicable to the instance data we have, we assume that the location of the ontology describing a property or class (if available) is consistent with the namespace URI of that property or class. To obtain the list of namespaces, we perform an index scan and for every property (resource in the predicate position) or class (resource in the object position of a triple/quad with predicate `rdf:type`) encountered we trim after the last hash or slash to achieve that concept's namespace URI. For instance, we expect the ontology information for predicate `<http://xmlns.com/foaf/0.1/mbox>` at the URL `http://xmlns.com/foaf/0.1/`. During the index scan we preserve a unique list of namespaces URIs, and upon completion of the scan, we attempt to retrieve the data from these URLs, using HTTP accept headers set to `application/rdf+xml`.

Having retrieved the ontologies applicable to the dataset, we now parse and scan these ontologies for properties which are defined with value `owl:IFP` for `rdfs:subPropertyOf` or `rdf:type`. All properties defined as such are inverse functional and the full list of these properties' URIs is written to a file.

With a list of IFPs in hand, we can now begin detecting equivalence of instances present in the dataset.

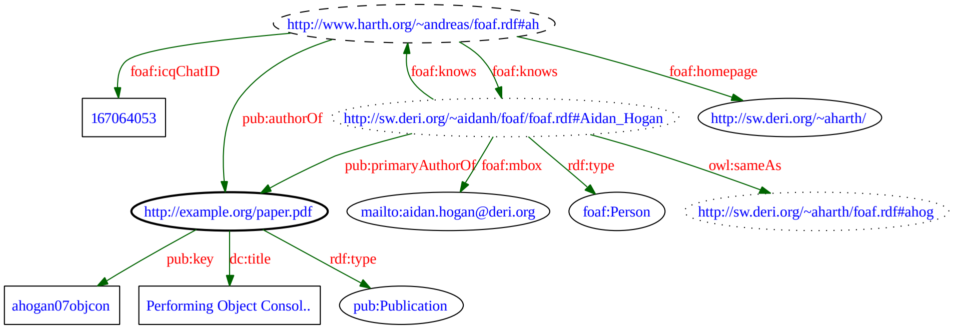

http://www.harth.org/~andreas/foaf.rdf#ah
foaf:icqChatID
foaf:knows
foaf:knows
foaf:homepage
167064053
pub:authorOf
http://sw.deri.org/~aidanh/foaf/foaf.rdf#Aidan_Hogan
http://sw.deri.org/~aharth/
pub:primaryAuthorOf
foaf:mbox
rdf:type
owl:sameAs
http://example.org/paper.pdf
mailto:aidan.hogan@deri.org
foaf:Person
http://sw.deri.org/~aharth/foaf.rdf#ahog
pub:key
dc:title
rdf:type
ahogan07objcon
Performing Object Consol..
pub:Publication

Figure 2: Output of the object consolidation algorithm applied to the three graphs from Figure 1.

4.2 Detecting Equivalence

We begin with the formal definition of an instance and instance equivalence, and then discuss in detail our method of detecting equivalence between instances.

Definition 4.1. *(Instance) Given our dataset a set of quads $\mathcal{Q}$, an instance I with identifier $i \in (\mathcal{R} \cup \mathcal{B})$ has a set of quads $\mathcal{I} \subset \mathcal{Q}$ and optionally a set of in-linking quads $\mathcal{J} \subset \mathcal{Q}$ where for each quad $q \in \mathcal{I}$, $s(q) = i$ and for each quad $q \in \mathcal{J}$, $o(q) = i$.*

Definition 4.2. *(Instance Equivalence) Given an instance I_0 and an instance I_1, and a set of inverse function properties $\mathcal{IFP} \subset \mathcal{R}$, instance I_0 and I_1 are equivalent if there exists a quad $q_0 \in \mathcal{I}_0$ and a quad $q_1 \in \mathcal{I}_1$ such that $p(q_0) = p(q_1) \in \mathcal{IFP}$ and $o(q_0) = o(q_1)$. Instance equivalence is reflexive, symmetric and transitive.*

We use the term instance equivalence to refer to the case whereby multiple instances with differing identifiers describe the same resource or entity. Table **??** shows the equivalent instances from Figure 1.

To find instances matching the criteria for equivalence specified in Definition 4.2 we perform a sequential index scan of the POCS index. Since the POCS index is sorted respective of predicate, then object, then context, then subject; the POCS index contains groups of quads with the same predicate and object values. Thus we require minimal overhead to find quads q_0 and q_1 where $p(q_0) = p(q_1)$ and $o(q_0) = o(q_1)$. With the additional constraint of $q_0 \& q_1 \in \mathcal{IFP}$ we can say that $s(q_0)$ and $s(q_1)$ are different identifiers for equivalent instances.

Algorithm 1 summarises the process.

Algorithm 1 find equivs($POCS$, $\mathcal{IFP}$)

```
for quads $q_k$ in $POCS$ do
  if $p(q_{k-1}) = p(q_k)$ and $p(q_k) \in \mathcal{IFP}$ then
    if $o(q_{k-1}) = o(q_k)$ then
      store equivs($s(q_{k-1})$,$s(q_k)$)
    end if
  end if
end for
return $eq.list$
```

Now that we can detect equivalence, we require a data structure for the algorithm to store these equivalences efficiently.

4.3 Storing Equivalence Data

We design and implement an in-memory data structure to maintain listings of equivalent instances. Thus we use a list of lists to store the URIs of instances which meet the constraints for equivalence; this data structure can be thought of as a list of rows of variable length with each row containing instance identifiers. The identifiers in each row are for equivalent instances. Thus we implement the reflexive, symmetric and transitive properties of instance equivalence.

We also use a hashtable to implement an inverted index which maps identifiers in the equivalent instance list to the rows in which they are found. Thus, given an instance identifier i, we can find the row i appears in and from this we can find all of the identifiers of the equivalent instances of i. An identifier may only appear in one row at a time, specifically to satisfy transitivity. Equivalent identifiers are added in pairs. If one identifier is already present in a row of the list (as dictated by the inverted index) the other identifier is added to that row. If both are already present in different rows, both rows are merged. If neither is present a new row is created. We call this data structure an equivalence list or table. Algorithm 2 outlines the process of adding two equivalents to the equivalence list.

Algorithm 2 store equivs(A,B)

```
$r_A$ = find $eq.list$ row A is in
$r_B$ = find $eq.list$ row B is in
if !$r_A$ and !$r_B$ then
  new row $r_{AB}$
else if $r_A$ and $r_B$ then
  merge $r_A$ and $r_B$
else if $r_A$ then
  add B to $r_A$
else if $r_B$ then
  add A to $r_B$
end if
return
```

The equivalence list is filled as the sequential scan is performed on the POCS index and equivalences are detected. The worst case memory requirements for the equivalence list would occur for an index where each instance is equivalent to at least one other instance. In this scenario, for an index with N instances, the equivalence list would have to store a list of lists with N entries and a hashtable with N buckets.

4.4 Merging Instances

Having filled the equivalence list by scanning the POCS index, we must now begin the task of merging instances to one consolidated instance. We begin by defining the process of merging instances.

Definition 4.3. *(Merging Instances) To merge an instance I_0 with identifier i_0 and an instance I_1 with identifier i_1, we replace identifier i_1 with i_0. Thus, $\mathcal{I}_0 = \mathcal{I}_0 \cup \mathcal{I}_1$ and $\mathcal{J}_0 = \mathcal{J}_0 \cup \mathcal{J}_1$.*

We wish to merge instances under a consolidated identifier. We call this consolidated or canonical identifier the pivot identifier or pivot element. There must exist a pivot element for each row in the equivalence list; we see the identifiers in the row as candidates. There are no formal guidelines for such selection of pivot elements for RDF data and there are multiple alternatives to selecting a pivot identifier from an equivalence chain or row.

Decisively, we can choose URIs over blank node identifiers as URIs can be recycled for future extensions of the resource description. However, if all of the candidates are blank node IDs, we must decide whether or not to create and assign a URI for the consolidated instance. Currently, we do not create a URI and instead pick a blank node pivot ID from the candidates.

We are faced with numerous choices if there are multiple URI candidates. The choice of URI makes no difference to a machine interpretation of the consolidated instance. However, humans may desire the instance URI to be somehow informative; being perhaps a web resource with information linked to the instance or being human interpretable with some keywords contained within the string, etc. Also, it would be courteous to choose URIs which people assign to themselves or their property. Choosing suitable URIs in this respect encourages better future URI agreement.

Thus, in choosing between URIs, there are multiple possible approaches (or combination of approaches).

1. **Random or lexical ordering:** a URI is chosen from the candidates at random or alphabetically. This is the simplest method to implement.

2. **Most agreed upon across data sources:** a URI is chosen which appears in the most distinct data sources and so is the most agreed upon.

3. **Count of occurrence:** the URI with the highest degree is chosen (i.e., the URI appearing in the largest number of statements).

4. **Links analysis:** a URI is chosen according to a links analysis technique. ReConRank[8] could be used as it is suited to RDF ranking and incorporates context; thus it would also take into account the agreement across data sources. However, links analysis would be quite expensive to implement for large datasets.

We currently choose method 3 to pick pivot identifiers. If the occurrence count still yields multiple candidates (i.e., multiple identifiers occur the same number of times), we pick one arbitrarily by alphabetical order. We perform the occurrence count of all the identifiers in the equivalence list together by means of another sequential scan of the index. Once we have acquired the count, we can pick the pivot elements. The restriction of selecting URIs over blank node identifiers supersedes the restriction of selecting those with more occurrences. Algorithm 3 outlines the process.

Algorithm 3 pick pivots(*count*, *eq.list*)

```
for row r ∈ $eq.list$ do
  temp = null
  for entry e ∈ r do
    if temp is null then
      temp = e
    else if $temp \in \mathcal{R}$ & $e \in \mathcal{B}$ then
      continue
    else if $temp \in \mathcal{B}$ & $e \in \mathcal{R}$ then
      temp = e
    else if $count_e > count_{temp}$ then
      temp = e
    end if
  end for
  $pivot_r$ = temp
end for
return $pivot$
```

In the example data depicted in Figure 1, `http://sw.deri.org/~aidanh/foaf/foaf.rdf#Aidan_Hogan` is picked as the pivot element before `_:genidE` which is a blank node ID, and picked before `http://sw.deri.org/~aharth/foaf.rdf#ahog` which occurs in less statements (2 vs. 3).

We now can merge the instances in each row under the row's pivot element. For each quad q in the POCS index, if $s(q)$ is found to appear in row a of the equivalence list, $s(q)$ is rewritten as the pivot element of row a – unless of course, the pivot element happens to be $s(q)$ itself. The same applies for $o(q)$. We also store the equivalences we found through object consolidation as owl:sameAs triples, so that references to the old identifiers are maintained in the index.

One pass of the object consolidation process is now complete. The process of rewriting the index with the consolidated identifiers is summarised in Algorithm 4.

Algorithm 4 rewrite index(*POCS*, *pivot*, *eq.list*)

```
rerun = false
for quads q ∈ $POCS$ do
  if o(q) ∈ $eq.list$ then
    o(q) = $pivot_{o(q)}$
    if p(q) ∈ $\mathcal{IFP}$ then
      rerun = true
    end if
  end if
  if s(q) ∈ $eq.list$ then
    s(q) = $pivot_{s(q)}$
  end if
  write quad q to $newPOCS$
end for
for row r ∈ eq. list do
  for entry e ∈ r do
    write $pivot_r$ owl:sameAs e to $newPOCS$
    write e owl:sameAs $pivot_r$ to $newPOCS$
  end for
end for
return rerun
```

4.5 Multiple Iterations

It is quite possible that multiple iterations of the object consolidation process are required to be run. Specifically, if $o(q)$ gets rewritten and $p(q) \in \mathcal{IFP}$, another iteration may be required. Algorithm 4 returns true if another iteration is required.

To illustrate, in Figure 1, the data requires two iterations of object consolidation for completeness. In the first iteration, instances `_:genidB` and `_:genidC` are consolidated through value `ahogan07objcon` for inverse functional property `pub:key`. In the rewrite stage, `_:genidC` is replaced with `_genidB`. Now that instances `http://sw.deri.org/~aidanh/foaf/foaf.rdf#Aidan_Hogan` and `_:genidE` share value `_:genidB` for IFP `pub:primaryAuthorOf`, they are consolidated on the second pass.

A summary of the entire object consolidation process is given by Algorithm 5

Algorithm 5 objcon($POCS$)

```
scan $POCS$ for namespaces
crawl namespace URLs
retrieve $\mathcal{IFP}$ from data
done = false
while !done do
  $eq.list$ = find equivs($POCS$,$\mathcal{IFP}$)
  for entry e in $eq.list$ do
    get count($POCS$)
  end for
  pivots = pick pivots($counts$,$eq.list$)
  done = rewrite index($POCS$,$pivot$,$eq.list$)
end while
```

5. EVALUATION AND DISCUSSION

The following section evaluates the object consolidation algorithm described in Section 4. We begin by discussing our process of acquiring the evaluation dataset and then discuss various properties of the dataset. We highlight various issues we uncovered whilst performing object consolidation on our dataset. Finally, we give insights into the results achieved by performing object consolidation on the evaluation dataset.

5.1 Data Acquisition

For the purpose of evaluation, we retrieve the entire web of RDF data. To do so, we employ MultiCrawler [7] which follows `rdfs:seeAlso` links from data source to data source. MultiCrawler is a distributed architecture for crawling Web data and converting such data to RDF. For the purpose of this work, we only retrieve data which is natively RDF.

5.2 Dataset Characteristics

We now present analysis on the data we have acquired and converted to N-Quads for various properties relevant to data integration across sources and identification of resources.

The data consists of 472,602,998 unique quads describing 72,274,051 instances (unique subjects) equating to an average of 6.539 statements per instance. Of these instances 60,772,168 (84%) are not assigned a URI (anonymous node) whilst 11,501,883 (16%) are provided a URI.

In our measurement of the degree of linkage in the graph, we are only interested in links between instances. Thus, we discard any links with edge `rdf:type` which links instances to their class. We found that there are 107,448,668 inlinks in the graph; this translates to an average in-degree of 1.48.

The data is taken from 3,119,953 contexts. Table 2 is a list of the top 10 hosts which provide data giving their total contribution in quads. They are all social networking sites which export their user profile data to FOAF. This is reflected in the fact that of the 72,274,051 instances, 56,594,815 (78%) are directly of type `foaf:Person` and in second place, 2,918,312 (4%) are of type `foaf:Document`. Of the `foaf:Person` instances, 56,581,590 (99.9%) had no URI identifier whereas for `foaf:Document`, all but 14 of the 2,918,312 were provided URIs. `foaf:Person` instances are real-world entities which can create difficultly in getting agreement on a URI, whereas with `foaf:Document` which mainly describes web pages, no such problem exists.

Host	Statements	%
livejournal.com	438,198,473	92.7%
tribe.net	15,037,190	3.1%
deadjournal.com	7,743,418	1.6%
greatestjournal.com	4,983,473	1.1%
vox.com	3,274,714	0.7%
affymetrix.com	723,028	0.2%
opera.com	408,166	0.09%
rossia.org	307,874	0.07%
klab.lv	258,128	0.05%
typepad.com	232,578	0.05%

Table 2: List of top 10 RDF data providing hosts and the number of quads they contribute.

Figure 3 depicts a graph of the distribution of the number of contexts instance URIs appear in. For example, approximately 10 million URIs are only used in one source of data, around 0.5 million URIs are agreed upon over two sources, etc. We disregard the context distribution for any resources which are of type `rdf:Property`, `rdfs:Class` or `owl:Class` as these are schema concepts. The highest agreement over sources on a URI is that of `http://www.livejournal.com/directory.bml?opt_sort=ut&s_loc=1&loc_cn=US` used by LiveJournal[6] – a weblog hosting site which exports user profiles as FOAF files – to denote US as the country of residence of a user; the URI is used in 1.5M contexts. Generally, URIs used by LiveJournal to denote countries and general interests such as `music` or `arts` are well agreed upon; although they are only agreed upon within LiveJournal data, this data constitutes the bulk of the Semantic Web graph.

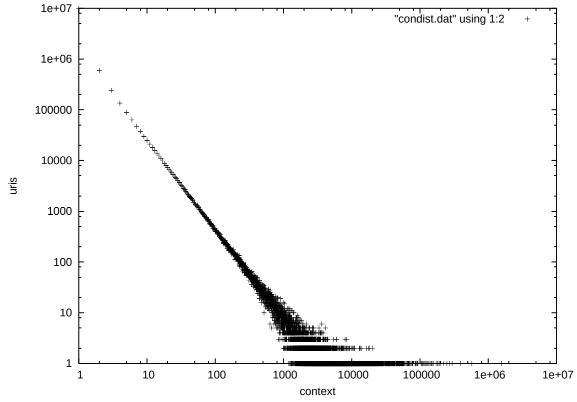


Figure 3: Number of URIs against the number of contexts they appear in (log/log scale)

We now present evaluation for performing our object consolidation algorithm on this dataset.

5.3 Results

The following section presents some insights into the op-

[6] `http://www.livejournal.com`

eration of object consolidation algorithm over the data described in the previous subsection. Firstly we discuss various issues we uncovered whilst performing object consolidation on our dataset. We then present some statistics compiled while running the algorithm.

One major issue we discovered involved `foaf:weblog`, which is defined as being an `owl:IFP` in the FOAF ontology. Thus its semantics mandates that the property uniquely defines the `foaf:Person` instance for which it is described. However, this is not the case for sites such as LiveJournal which use the property to define communal weblogs which multiple people share. By interpreting `foaf:weblog` as an inverse functional property, we incorrectly consolidate members of the same communal weblogs together. Thus we omit `foaf:weblog` from the list of inverse functional properties we utilise.

Another obstacle we encountered involved invalid values being defined for properties relating to instant messaging usernames. For example, many users of Livejournal had entered common invalid values such as ”`ask`”, ”`none`” or simply ”`?`” for various instant messaging usernames which they either did not have, or wished not to reveal. This issue led to erroneous consolidation of 85,803 entities (0.12% of total, 2.9% of those consolidated) to one consolidated entity; the 85,803 entities were linked through a web of shared invalid values. A possible solution to this problem would involve the creation of a manual black-list of values which the algorithm ignores or possibly the provision of a more strict definition of the datatypes of values of datatype properties which can be cross-checked against values observed. Thus, literals with spaces, invalid characters or literals not meeting length requirements for various properties would be ignored during object consolidation.

Overall, object consolidation saw 2,443,939 instances merged to 401,385 pivot elements. Of the pivot identifiers, 400,892 were blank nodes IDs which were merging 2,425,175 instances (6.05:1), 493 were URIs merging 18,764 instances (38.06:1). Besides the invalid 85,803 length equivalence chain, the longest chain found was of length 32,390 and contained equivalent instances referring to a user of Vox[7] (a weblog hosting site) named ”Team Vox”, which had `foaf:knows` inlinks from all other users on the site. Figure 4 shows the distribution of equivalence row lengths.

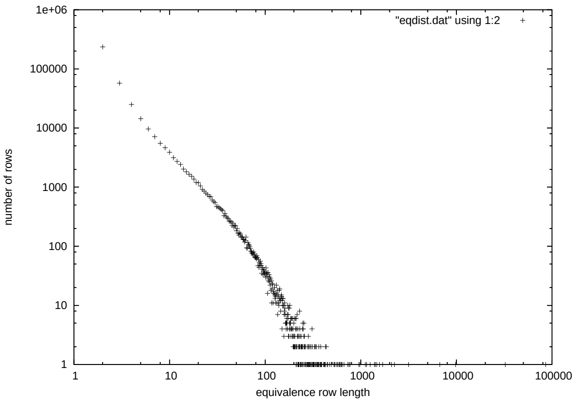


Figure 4: Equivalence row distribution (log/log scale)

Table 3 shows the list of IFPs and the amount of atomic equivalences found through them (where non-zero).

Inverse Functional Property	Equivalences
http://xmlns.com/foaf/0.1/mbox_sha1sum	1927168
http://xmlns.com/foaf/0.1/homepage	1397528
http://xmlns.com/foaf/0.1/aimChatID	54868
http://xmlns.com/foaf/0.1/msnChatID	18154
http://xmlns.com/foaf/0.1/jabberID	9634
http://xmlns.com/foaf/0.1/icqChatID	9530
http://xmlns.com/foaf/0.1/yahooChatID	6392
http://xmlns.com/foaf/0.1/mbox	2856
http://usefulinc.com/ns/doap#homepage	5

Table 3: List of IFPs which result in successful consolidation and the number of *atomic* equivalences they find.

The consolidated dataset contains 70,231,472 instances, a reduction of the number present in the dataset before object consolidation (since object consolidation has merged many instances). Overall, the average degree in the datagraph improves slightly to 1.53 (+0.05). 54,552,267 of the instances in the data are of type `foaf:Person`, a decrease of 2.04M; almost all of the object consolidation was performed on foaf:Person instances.

5.4 Functional Properties

It is interesting to note here that `owl:FunctionalProperties` may also be used for object consolidation. An instance can only have one value for a functional property. Therefore, if an instance is seemingly attributed multiple values for a functional property, these values may be considered equivalent. An example is `foaf:primaryTopic`; an instance can only have one primary topic.

We performed some evaluation on object consolidation which used an algorithm similar to that described in Section 4 using a SPOC ordered index and analysing functional properties; we found that very few instances were merged. We observed six equivalences found through `foaf:primaryTopic` and one through `wot:identity`[8]. Thus we only mention our functional property analysis here and omit it from the main text.

5.5 Schema Mapping

This paper deals specifically with consolidation of data on the instance level. We make no special efforts to adapt the algorithm for consolidation of similar concepts for different schemas. Where inverse functional properties are shared by the same concept under different URIs, consolidation will be performed. However, inverse functional properties are not normally defined for schema concepts. Instead, equivalences are often explicitly defined between concepts through use of properties such as `owl:sameAs`, `owl:equivalentClass` and `owl:equivalentProperty`. Thus, schema mapping can be performed by populating the equivalence list structure through analysis of these properties, picking the pivot identifier as described in Section 4.3 and additionally rewriting the predicate elements in the index as well as the subject and object elements.

6. RELATED WORK

Record linkage has a long history, starting from seminal work by Newcombe et al. in 1959 [12]. Chen et al. use

[7] http://vox.com

[8] `http://xmlns.com/wot/0.1/`

a graph-based data model for representing data, and use inter-object relationships to detect clusters of similar items. Michalowski et al. [11] utilise secondary sources to determine equivalences. A recent survey summarising many of duplicate record detection methods can be found in [4].

The SemTag effort as described in [3] assigns identifiers to Web pages. The authors describe a simple algorithm to disambiguate entities and provide a large-scale effort at tagging and assigning topic URIs to web pages. In contrast, we utilise `owl:InverseFunctionalProperty` information obtained from ontologies to determining equivalences.

The TAP project [6] tries to overcome the problem of two applications using different names for the same concept by a process called "Semantic Negotiation". Applications can bootstrap from a small agreed vocabulary to a shared understanding in a larger domain. The identifiers used in TAP are centrally created, whereas in our approach, URIs denoting concepts stem from a large number of sources on the Web. However, it would still be interesting to combine the Semantic Negotiation approach with our dataset to help applications and data creators decide on URIs for instances.

Bouquet et al. [1] motivate the problem of (re)using common identifiers as one of the pillars of the Semantic Web, and provide a framework and fuzzy matching algorithms to fuse identifiers. We perform object consolidation on a larger scale and avoid use of probabilistic methods.

The idea of performing object consolidation on FOAF data from the Web based on values of inverse functional properties has been coined "smushing"[9]. Brickley discusses in [2] implementation strategies for merging identifiers in RDF based on RDF reasoning engines. However, the technique has not yet been applied to large datasets. Our approach handles data from a large number of sources obtained from the Web; thus to cater for potentially long equality chains derived from many sources, efficient data structures for storing an equality table are required. In addition, we provide detailed statistics on the current usage of data on the Semantic Web.

The Florid system [5] is an F-logic inspired reasoning system that maintains an equality relation data structure to be able to deal with ground equalities. In the Hyperion project [9], the issue of relating identifiers amongst peers is solved using so-called "mapping tables".

Finally, Park and Durusau [13] use the notion of subject-centric merging of ontologies based on a Topic Map approach to denote a fine-grained approach to subject identity.

7. CONCLUSION

We have presented a method to consolidate identifiers from Web data to improve accuracy of applications operating on the dataset. The method includes fetching of ontology information, incremental build-up of an inverted equality table, selecting a canonical identifier, rewriting the dataset and maintaining the footprint of object consolidation through use of `owl:sameAs` statements added to the index.

The experiments we conducted show that the algorithms scale to large datasets and thus are applicable for Web data. Furthermore, we provided an account of the current data situation on the Semantic Web.

[9] `http://lists.w3.org/Archives/Public/www-rdf-interest/2000Dec/0191.html`

8. REFERENCES

[1] P. Bouquet, H. Stoermer, M. Mancioppi, and D. Giacomuzzi. Okkam: Towards a solution to the "identity crisis" on the semantic web. In *Proceedings of SWAP 2006, the 3rd Italian Semantic Web Workshop*, volume 201 of *CEUR Workshop Proceedings*, December 2006.

[2] D. Brickley. Rdfweb notebook: aggregation strategies, January 2002. http://rdfweb.org/2001/01/design/smush.html.

[3] S. Dill, N. Eiron, D. Gibson, D. Gruhl, R. Guha, A. Jhingran, T. Kanungo, S. Rajagopalan, A. Tomkins, J. A. Tomlin, and J. Y. Zien. Semtag and seeker: bootstrapping the semantic web via automated semantic annotation. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 178–186, New York, NY, USA, 2003. ACM Press.

[4] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 19(1):1–16, 2007.

[5] J. Frohn, R. Himmeröder, P.-T. Kandzia, G. Lausen, and C. Schlepphorst. Florid: A prototype for f-logic. In W. A. Gray and P.-Å. Larson, editors, *Proceedings of the Thirteenth International Conference on Data Engineering, April 7-11, 1997 Birmingham U.K*, page 583. IEEE Computer Society, 1997.

[6] R. V. Guha and R. McCool. Tap: a semantic web platform. *Computer Networks*, 42(5):557–577, 2003.

[7] A. Harth, J. Umbrich, and S. Decker. Multicrawler: A pipelined architecture for crawling and indexing semantic web data. In *Proceedings of the 5th International Semantic Web Conference*, pages 258–271, November 2006.

[8] A. Hogan, A. Harth, and S. Decker. Reconrank: A scalable ranking method for semantic web data with context. In *2nd Workshop on Scalable Semantic Web Knowledge Base Systems*, 2006.

[9] A. Kementsietsidis, M. Arenas, and R. J. Miller. Managing data mappings in the hyperion project. In *Proceedings of the 19th International Conference on Data Engineering (ICDE)*, pages 732–734, 2003.

[10] F. Manola and E. Miller. RDF Primer. W3C Recommendation, Feb. 2004. http://www.w3.org/TR/rdf-primer/.

[11] M. Michalowski, S. Thakkar, and C. A. Knoblock. Exploiting secondary sources for automatic object consolidation. In *Proceeding of 2003 KDD Workshop on Data Cleaning, Record Linkage, and Object Consolidation*, 2003.

[12] H. B. Newcombe, J. M. Kennedy, S. J. Axford, and A. P. James. Automatic linkage of vital records: Computers can be used to extract "follow-up" statistics of families from files of routine records. *Science*, 130:954–959, October 1959.

[13] J. Park and P. Durusau. Towards subject-centric merging of ontologies. In *Proceedings of the 9th International Protege Conference*, July 2006.