A Framework for Reference Management in the Semantic Web

Timothy Lewy
School of Electronics and Computer Science
University of Southampton, UK
tml203@ecs.soton.ac.uk

Hugh Glaser
School of Electronics and Computer Science
University of Southampton, UK
hg@ecs.soton.ac.uk

Nigel Shadbolt
School of Electronics and Computer Science
University of Southampton, UK
nrs@ecs.soton.ac.uk

ABSTRACT

Much of the semantic web relies upon open and unhindered interoperability between diverse systems. The successful convergence of multiple ontologies and referencing schemes is key. This is hampered by a lack of any means for managing and communicating co-references. We have therefore developed an ontology and framework for the exploration and resolution of potential co-references, in the semantic web at large, that allow the user to a) discover and record uniquely identifying attributes b) interface candidates with and create pipelines of other systems for reference management c) record identified duplicates in a usable and retrievable manner, and d) provide a consistent reference service for accessing them. This paper describes this ontology and a framework of web services designed to support and utilise it.

Categories and Subject Descriptors

H.3.5 [**Information Systems**]: Online Information Systems - *Web-based services*;; D.2.12 [**Software**]: Software Engineering - *Interoperability*

Keywords

Reference management, Co-reference, Web services

1. INTRODUCTION

The emergence of the semantic web [5] is, in essence, a move from a web of pages designed and published for human consumption, with no intention other than to be viewed by the human eye and parsed by the human brain; to a web of data connected by machine interpretable semantics, that when applied or used in a suitable context produce content or services useful to other semantic systems, agents or end users.

Instead of documents linked by hyperlinks the web becomes entities and resources (people, places, things or concepts) linked by attributes and associations. The knowledge represented in the web is gathered and entities identified by a multitude of persons and processes for many different purposes, from many different sources. It is not uncommon for inconsistencies to occur within and between the data gathered by different processes. Frequently it transpires that some entities are in fact equivalent to one another. For example “Nigel Shadbolt”, co-author of this paper could well be equivalent to a “N. Shadbolt”, author of another paper. Determining this reliably, however, is not an easy task. Entities, or instances are rarely completely specified in a given context and even less frequently specified consistently. Simply performing a naïve comparison of attribute values is, therefore, unlikely to be a resounding success, especially if the values are just string literals.

The problem is inherent in the identification system employed by the semantic web. All entities are assigned a Unique Resource Name (URN), which is a name for the resource appended to the domain from which it was created. It is a URI and appears very similar to the URLs used to locate web pages. This may seem simple "In Semantic Web we not only provide URIs for documents as we have done in the past, but to people, concepts and relationships. ... [B]y giving unique identifiers to the person, the role "writer" and the concept of “research paper" we make very clear who the person is, and the corresponding relation between this person and a particular document.”[9] However, for any real world (large scale) activity, it proves impossible to find a truly unique identifier for any given person, paper or role (i.e. one that would be recognised by any system in any context). Whilst it is possible to create a unique identifier for an entity in a given domain, that identifier would have only local significance to the creator and the creator’s application. Anything attempting to gather data on that resource, from a foreign application, or with reference to another knowledge source would have to resolve it against existing references.

Identifying these equivalent entities is a serious business. Taking persons and names as an example; “Hall W.” is author of a paper. “Wendy Hal” is author of another. “Wendy Hall” is head of this school. All this information has to be reconciled. Names can be overloaded i.e. there could be two entirely different people called Wendy Hall, both of whom might have written research papers. Names are frequently incomplete or inconsistent: “W. Hall”, “N. Shadbolt”, “N. R. Shadbolt, “Hugh Glaser” or “Glaser, H.”. Sometimes they are inaccurate e.g. “Nigel Shadblot” (as opposed to “Nigel Shadbolt”). During 2001, the UK university funding organisation conducted a Research Assessment Exercise (RAE), in which some details on all active researchers were collected and have now been published. The extent of the problem within the UK research community can thus be seen by analysing these RAE 2001 returns. Within the list of researcher names in the institutional submissions (which are recorded as initials and surnames on the HERO website www.hero.ac.uk) 10% of the names lead to clashes between two or more individuals. If the names are restricted to a single initial, the proportion of clashes rises to 17%. Within our own institutional repository, records show that depositors typically give up to six different ways of naming any individual author (due to combinations of full names, initials and names that are incorrectly spelled).

As part of the Advanced Knowledge Technologies project [1] data on UK computer science research was gathered from a variety of sources and combined in a single knowledge base. In merging data from different sources and ontologies, duplicate references arose. Searching the knowledge base for the string “Nigel Shadbolt” reveals some 25 separate references that represent the

same person, none of which are linked. It would be very difficult for any interested person to obtain all the information regarding Nigel Shadbolt from this knowledge base. The problem is also exemplified by the EPrint repository software, part of the Open Archive Initiative [15]. Each repository assigns references to its authors and papers according to a local naming scheme. This is sufficient within a single repository, however co-references (duplicate URNs to a single entity) have to be resolved in order to perform any interesting tasks, for example, gathering every paper by a single author from multiple repositories. This is especially difficult as authors moving from one institution to another can have very different metadata from one repository to the next. It is a crippling problem and effectively isolates semantic repository data to its residing archive.

A few solutions to the URN assignment issue, drawn from other areas of computer science have been suggested and will be outlined in section 2.1 however they prove unsuitable for our use.

The issue of co-reference and equality within the semantic web is crucial. Take, for example, Tim Berners-Lee's semantic web agent [5][17]]. It is given the task to look up a patient's personal information, find their prescribed treatment and then present, to the user, an appointment at an appropriate clinic, at a time when the user is available. There are many different knowledge sources involved here: The patient record, a register of clinics, the clinic's appointment system and the person's scheduler. From the outset the agent will have to do a lot of work to achieve its goal: The patient records might well use a different ontology for describing treatments than the clinic registry, or the clinic appointment system. The three different source ontologies would have to be merged, or at least mapped before the agent can operate between them. This might be in the form of a service available to the agent, or it might be done on the fly [13].

Once mapped, our problem of referential inconsistencies and co-reference resolution is encountered. The patient record system and the clinic registry, whilst possibly using the same class for treatments in their ontologies, may not have used the same URI for identifying the treatment in question. The agent cannot work without resolving this problem.

Only very limited solutions to the co-reference problem have been proposed. A solution is required that works in any situation, with any semantic application; currently the problem lacks even basic formalisation. We have therefore developed an ontology to describe, manage and communicate co-references as they occur, in any domain. This is outlined in section 3. Having established this, we will showcase a range of web services designed to provide the essential functions for resolving and communicating co-references, using our ontology (section 4.).

2. RELATED WORK AND ISSUES

2.1 Proposed Solutions

The problem of co-reference is not new; it has been encountered in fields such as natural language processing and AI. The approach largely taken by the AI community is to enforce that there must only ever be a 1 to 1 relationship between resources and identifiers. This is known as the *unique name assumption* [14]. If one can make this assumption, the problem does not appear. However, we cannot import this to our own uses in the semantic web as it would prove infeasible. A system such as the Digital Object Identifiers (DOI) (www.doi.org), in effect takes the unique name assumption, but it suffers from the problem of a naming authority. It only works to the extent it does, because the assumption is that the owner of a document assigns the DOI. In the semantic web world, anyone and everyone refers to documents, irrespective of whether it has or they know the DOI. Similarly ontologies and datasets are frequently developed by many different people around the world, even within one particular project. The coordination involved in ensuring that all these developers do not create multiple references for a single resource is not practical. From a more global standpoint, it would be virtually impossible to ensure that no resource possessed multiple identifiers within the entire semantic web. If one cannot make the assumption, an alternative would be to enforce it; by introducing naming authorities, similar to those managing Internet domain names. They would distribute and record identifiers, enforcing referential integrity. However a naming authority might have a record of references only to discover that two are actually equivalent or that one encompasses non equivalent entities. New references would have to be checked against virtually every reference in the entire web for equivalence, before authorisation. New resources are created constantly within the semantic web; an authority would stunt this contravening the spirit in which the semantic web was born: that of free, open and unrestricted growth. Furthermore it would not be possible to distribute the authority into sub-authorities as there would be no effective way of delegating references, entities transcend traditional administrative boundaries. Even if it were possible, inevitably equivalent references would be found within or between authorities, creating the need for a resolution system: back to square one.

One interesting technology for resolving duplicate references from a set of candidate duplicates, is the use of communities of practice (CoP)([16]). A community of practice is a *group of people connected by a shared interest in a task, problem, job or practice* [12]. In the context of the semantic web, this can be viewed, for a given person, as the set of entities that indirectly share a sufficient amount of information i.e. the entities that have a number of relations to resources that the given entity is also related to. By obtaining the community of practice for members of sets of potential duplicates, or individual entities, we can derive a measure of similarity from the degree of overlap between CoPs. When this measure is above a threshold level, the sets of duplicates or individuals in question most likely represent the same entity. A tool, ONTOCOPI [3] has been developed for the calculation of CoPs. It has been tested as a component part of a system for co-reference resolution [2]. [2] Proposes a system for eliminating duplicate references that also encompasses ontology population and mapping from multiple, possibly legacy, sources. The framework proposed here is more abstract and can be used for mapping, intra-institution ontology maintenance and inter-institution communication of co-references. A suitable CoP system could well use and be integrated with the framework to provide a higher degree of automation, however alone one would not represent a particularly robust or complete solution.

2.2 Schemas for Co-reference Resolution

RDF [9] does not natively encompass equivalence relations. However most ontology languages that are extensions to RDF, now incorporate, in their schemas, predicates for establishing equivalence between resources. The Ontology Interchange Language (OIL) [6] originally incorporated concepts of equivalence into its schema; this was later incorporated by The Defence Advanced Research Projects Agency (DARPA)'s ontology language DAML+OIL [8]. DAML+OIL has equivalence

predicates, which can assert that two references either do or do not represent the same resource: daml:sameIndividualAs and daml:differentIndividualFrom. The work represented by these languages has evolved into and is now incorporated by the Web Ontology Language (OWL) [11]. The predicate owl:sameAs asserts that two references are logically equivalent to each other and represent the same entity. Similarly owl:differentFrom asserts that two references are different. By using these predicates graphs of co-references can be established and annotated within a knowledge base.

3. BUNDLE CO-REFERENCE ONTOLOGY

The first step towards developing an effective solution to the co-reference issue is to define an ontology that can be effectively used for gathering and handling co-references, or potential co-references. We require a schema that enables co-references to be easily identified, annotated and once recorded to be looked up and returned.

3.1 OWL

The existing OWL schema allows you to assert that two references are equivalent in a 1-1 relationship. This is somewhat insufficient for an effective co-reference system. Firstly the equivalences imply too strong an association. Whilst we are still unsure whether two references are the same we will desire a relation that is less strong. Secondly, for example, although we might wish to represent the knowledge that two different URIs are concerned with the same person, we still may wish to be able to identify related facts against a particular URI, such as associating different addresses with URIs that have come from different institutions. Thirdly, the natural way to establish duplicate references is in sets. OWL only allows 1-1 relationships, forcing any system to work in graphs. Graphs force the user to choose a canonical reference at the start. If multiple references are used as canonical i.e. different references are used as the subject of the equivalence relation, then traversing the graph and finding all references to a given resource becomes inefficient. Thus, a higher cardinality is desirable.

3.2 Bundle Structure

The ontology we have developed uses collections of potential duplicates. Each collection contains a set of duplicates. By using sets, the problem above does not occur. Within a collection, which we are calling a bundle, there may be any number of duplicates and non duplicates. A bundle represents a resource; the duplicates of the bundle are all references that refer to that resource, i.e. saying that an element of a bundle is a duplicate is saying that it refers to the same, or probably refers to the same resource as every other duplicate in that bundle. If it is a non duplicate, then that reference does not refer to the same resource as the duplicates in the bundle. This does not imply anything about what the non duplicates do refer to; just that it is not the same resource that the bundle represents. We found having non duplicates necessary, as it is often takes as much work to ascertain that two references are not the same as it does to ascertain that they are the same. Recording that two references are different entities is frequently as, if not more, important than recording that they are the same. One entity in each bundle may be marked as canonical; this indicates to outsiders the primary reference that they should use. Finally each bundle may have associated with it any number of predicates; this is useful as a reference of how the bundle was constructed. Resources can be conveniently identified as possible candidates using string searches, bundles are capable of recording what predicates were used to identify those candidates.

Bundles are resources of type #Bundle[1], duplicate references are associated using the predicate #duplicate and non duplicates with #notDuplicate. Predicates are associated to the bundle using #hasPredicate, as in Figure 1.

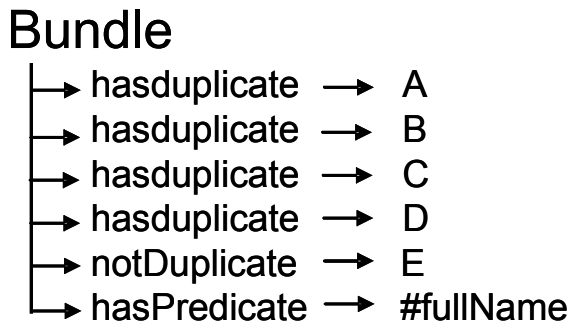

Figure 1. Visualization of a bundle. Duplicates are kept in sets rather than in graphs.

See Figures 3 – 7 for examples of bundles in RDF.

3.3 Utilising Bundles

Conceptually, one explores a knowledge base by some means and constructs bundles for each resource that appears to have co-references. Bundles are an effective means of collecting co-references; duplicates and non duplicates can be added and removed at will as it is essentially a set. This has the added bonus some set calculus can be performed upon it (see section 4.). If two bundles are found to represent the same entity they can simply be merged. They also form a convenient method of communicating references between systems (one can simply pass whole bundles between bundle-literate systems). Once the construction process is complete the whole bundle may simply be asserted into the knowledge base. It then forms part of a consistent reference service (see section 4.4) that can be used to obtain all the references to a resource that exist in the store, as a bundle, given any one of the references to it. This, of course, would therefore include the data on non duplicates and the canonical reference as well: invaluable information to someone wishing to interface with the knowledge base.

Bundles are robust, there are alternatives to asserting them directly: once constructed they can be converted into stronger OWL statements (see section 4.6). Or they can be used to create a type of gazetteer[2] (see section 4.5). Gazetteers are a concept more generally associated with geography. The list of place names against grid references at the back of an atlas is a type of gazetteer. A co-reference gazetteer is actually very similar: it is a list of names (strings) against canonical references to resources in the knowledge base. A string listed in a gazetteer is one that uniquely identifies a certain resource, such as a social security number or a very unique name. It can be used as a form of automatic co-reference resolution; when a new reference is added

1 All partial URIs are part of the AKT ontology and use the name space http://www.aktors.org/ontology/coref#

2 Gazetteer, can be “A geographical index or dictionary” (Oxford English Dictionary, n. 3.).

to a knowledge base strings related to it are checked against entries in the gazetteer. If an entry is found, the new reference refers to a resource that is already present; a bundle can be constructed, or appended to include the new reference.

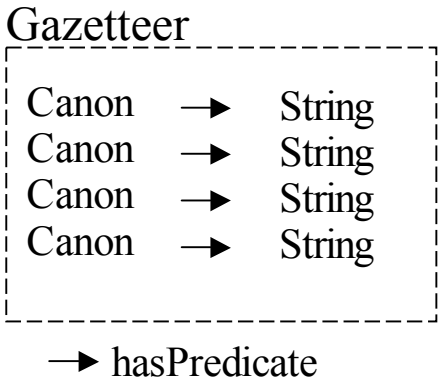

Figure 2. Visualisation of Gazetteer Structure.

Each gazetteer is represented by a model of type #Gazetteer. Within this model are triples of each canonical reference–string pair in the format "Canon, hasString, String". Gazetteer's are predicate dependant, each string only identifies a co-reference if the string is related using the specified predicate. This is necessary because it ensures that, for example, a social security number associated with a person would not falsely identify someone who has the same number, but as their telephone number. There is, therefore, a separate gazetteer for each predicate that can be used to uniquely identify any resources.

4. SERVICE FRAMEWORK

We put our co-reference ontology to use by developing a package of elementary web services designed for manipulating, constructing and operating with bundles, and available at triplestore.aktors.org/~tml203. They provide the essential building blocks for any resolution system. Some are standalone services, whilst some are designed to operate on top of a knowledge base. We used the AKT [1] triplestore, which uses a 3store [7] server and contains a large amount of suitable test data about the AKT IRC. Services which take RDF as input can either be sent RDF directly by HTTP POST or GET or can be pointed to a URL of some RDF. This feature allows the output of one service to be piped to the input of another, by stringing GET requests together. The webservices and ontology together represent the complete framework for resolution and communication.

4.1 Search

The first service, *search*, is quite straightforward: It takes a string and looks for all resources within the knowledge base that have some relation to that string. It then constructs a separate bundle, in RDF, for each result, containing the resource as a duplicate and the predicate that related the resource to the string. Blank nodes are ignored as they cannot be referenced, and so assertions cannot be made regarding them. The theory behind this service is that one enters a string that might lead to possible co-references and the service constructs a bundle for each of the results. The user or an automated system can then start merging those bundles that they believe to represent the same resource.

In our system, searching for "Shadbolt" creates several bundles based on references to various people with the name Shadbolt:

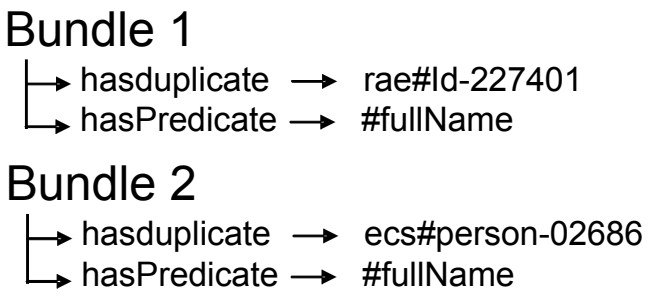

Figure 3. Excerpt from bundles returned by running the search service on the string "Shadbolt"

4.2 Group

As was previously mentioned in section 3.3, using bundles allows some set calculus to be performed. The *group* service performs a union on all the bundles supplied to it, effectively merging all the duplicates, non duplicates and predicates into one bundle. This is useful if all the bundles represent the same resource, it also represents an important building block for larger systems, allowing bundles to be merged automatically. The manual interface, detailed in section 4.8 uses this service to perform bundle merging.

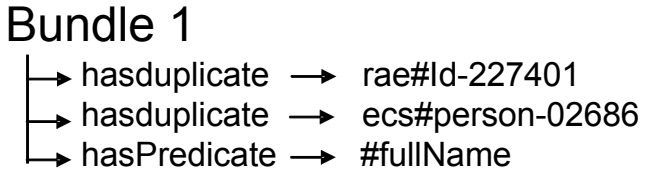

Figure 4. Output of the group service, when given the bundles in Figure as input.

A variant of this service that has been developed is a predicate dependant version. This service, $group_{pred}$, only merges those bundles which have a predicate in common. It does this recursively, so if we consider bundles as sets of predicates, with some undefined number of duplicates:

A = {p1, d1}, B = {p2, d2}, C = {p1, p3, d3}, D = {p2, p3, d4}, E = {p4, d5}

where p1, p2, p3, p4 ∈ Predicates and d1, d2, d3, d4, d5 ⊆ P(duplicates)

The service will perform A1 = A∪C, B1 = B∪D, A2 = A1∪B1 which will leave:

A2 = {p1, p2, p3, d1, d2, d3, d4}

E = {p4, d5}

E remains unmerged as it shared no predicates with any of the other bundles.

It is thus possible to avoid the problem referred to in 3.3 above, where a social security number might be confused with a telephone number, or the university John Hopkins might be confused with an individual of that name.

4.3 Canonical Reference Chooser

In some cases it may be necessary to select canonical references by hand or by use of some form of complex heuristics. However, in many situations the desired canonical reference will either be of

no importance, so long as it is consistent or will always be from a particular ontology. To this end we have produced two services for choosing a canon: $canon_{lex}$ selects a canon for each bundle using a reverse lexicographical ordering of the URIs (the reverse ordering provided more useful results in our tests than forward ordering). The other, $canon_{hier}$, uses a hierarchy of preferred ontologies that is built into the code. It looks for duplicates from its list of preferred ontologies, if one or more is found, the one from the highest point in the hierarchy is chosen. If none are found it uses the lexicographical chooser. The output from each service is shown below, given the first portion RDF as input.

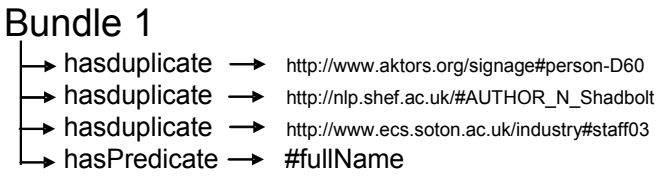

Figure 5. Input to Canonical Choosing Services Examples

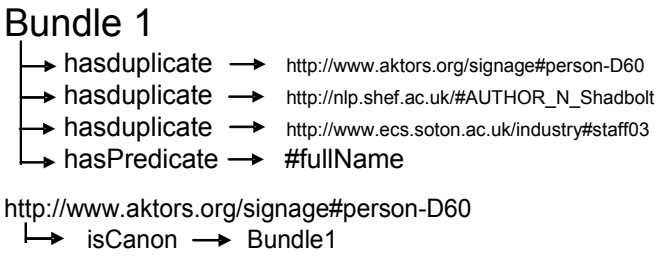

Figure 6. Bundle out put from hierarchical canonical entry chooser, with the AKTor's ontology at the top of the hierarchy.

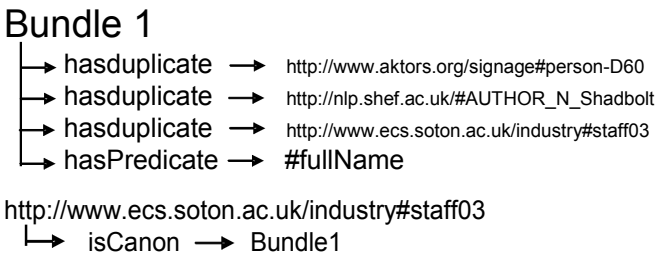

Figure 7. Bundle output from lexicographical canonical entry chooser.

4.4 Consistent Reference Service

Once the bundles have been constructed and asserted into the knowledge base there is the necessity for a service to get them out again. A service we provide is the consistent reference service (CRS). A *CRS* is a service that can be used by anyone to ensure that they are using the correct reference when interfacing with the knowledge base. With the bundle system, this is quite trivial: The service takes a reference to a resource, looks to see if that reference is associated with any bundles and if so, returns all statements regarding that bundle i.e. the bundle, all its contents and the canonical reference. It is a powerful means of communicating co-references; having obtained the bundle, the user or system has all the information they need to know about referencing that resource. They can then manipulate the bundle, add to it, pass it around, assert it into their own knowledge base, etc. However, more normally a system would simply use the CRS to find out what was the approved canonical reference. Furthermore, a CRS can provide a complete solution to co-references within a given domain. If a single CRS were shared by all the knowledge bases within a single domain, for example all EPrint servers or just UK institutional repositories; then it provides a medium for sharing, tracking and communicating co-references for the whole domain. A user would simply have to query the CRS with their reference to the entity they are interested in, then query any other store using the CRS, with the canonical reference and they could find any data regarding that entity in the domain.

If the bundle in Figure 8 were asserted into the knowledge base, it might be retrieved by accessing the CRS with the input "http://nlp.shef.ac.uk/#ARM_AUTHOR_Nigel__Shadbolt".

We envisage systems where cooperating sites use appropriate CRSs to register their own IDs, and then can choose to use the known canonical references to communiate with other sites. Of course, the CRS will change the bundles (at a rate which will depend on the application domain), and so the user of the CRS will need to periodically confirm that it is using the up to date canonical reference to get the best usage.

4.5 Gazetteer

Bundles can be used as a basis on which to build gazetteer entries, as was discussed in section 3.3. It can be the case that a maintainer is confident that any future occurrences of the set of strings under consideration will always be references to the same thing. Consequently, it is useful to record this so that other acquisition tools can assert using an existing URI, rather than compounding the problem by making up one of its own.

There are two parts to the gazetteering system, a service, *gazette*, for generating entries from bundles and a service for generating bundles from entries: essentially one for creating new entries and one for using the existing ones.

The gazetteer entry creator does its best to create a gazetteer entry from bundles that are supplied to it. In each bundle it looks for a label in the knowledge base for each reference and then creates a gazetteer entry, using the canonical entry from the bundle as the canon to each string. It is to be used only when it is certain that all the reference's labels are unique to that resource.

For our triplestore, passing the bundle in Figure 7 will generate the gazetteer shown below.

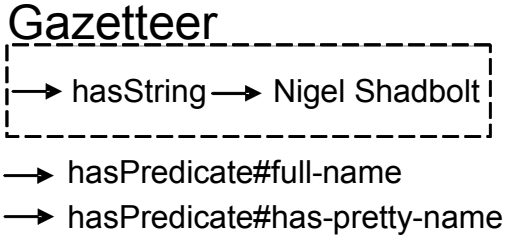

Figure 8. Gazetteer entry generated from RDF in Figure 7.

The second service, *bundle*, looks up gazetteer entries in the triplestore and returns bundles for each canonical reference of all the references that have a relevant string identified with a relevant predicate. Each bundle returned has, associated with it, the canonical reference and the predicates used by the gazetteer entry.

4.6 Bundle – OWL Translator

The Bundle – OWL Translator is for when a maintainer wants to make stronger assertions about bundle contents. When passed a bundle, *to_same_as* converts it into OWL ontology equivalence statements (owl:sameAs for each duplicate and owl:differentFrom for each non duplicate), which produce stronger inferences in a knowledge base. A canonical entry must be present for each bundle passed, as otherwise the translator would not know which reference to use as the subject of the output statements. Given the bundle in Figure 7 as input, this service will produce the output shown in Figure 9.

http://www.ecs.soton.ac.uk/industry#staff03

sameAs

http://www.aktors.org/ontology/signage#person-D60

http://www.ecs.soton.ac.uk/industry#staff03

sameAs

http://nlp.shef.ac.uk/#ARM_AUTHOR_Nigel__Shadbolt

Figure 9. Output of the Bundle – OWL Translator given the RDF in Figure as input.

4.7 Unique name assigner

Whilst bundles are being created they are assigned names unique within the output of the service that is handling them. If a bundle is to be asserted into the triplestore it must have a name unique within the entire knowledge base. Furthermore, it must have a name such that other bundles created at a later date, regarding the same resource, should have the same name. The unique name assigner, *unique*, achieves this by changing the names of all the bundles to names composed from the checksum of the URI of the bundle's canonical reference.

Bundle-034469a5cd8ef4a6742f2fc920f6ea09

- hasduplicate → http://www.aktors.org/signage#person-D60
- hasduplicate → http://nlp.shef.ac.uk/#AUTHOR_N_Shadbolt
- hasduplicate → http://www.ecs.soton.ac.uk/industry#staff03
- hasPredicate → #fullName

http://www.aktors.org/signage#person-D60

- isCanon → Bundle1

Figure 10. Output of the unique name assigner, given the bundle in Figure 6 as input.

5. Manual Interface

The forgoing sections have presented a series of services that can be used by other services and scripts without user intervention. For detailed, accurate work, and for dealing with for example, more common names, we have found the need for a manual system that allows the user to drive the services. To allow this, it must make it easy for the user to explore the information about the entities under consideration, and then easily invoke the appropriate services.

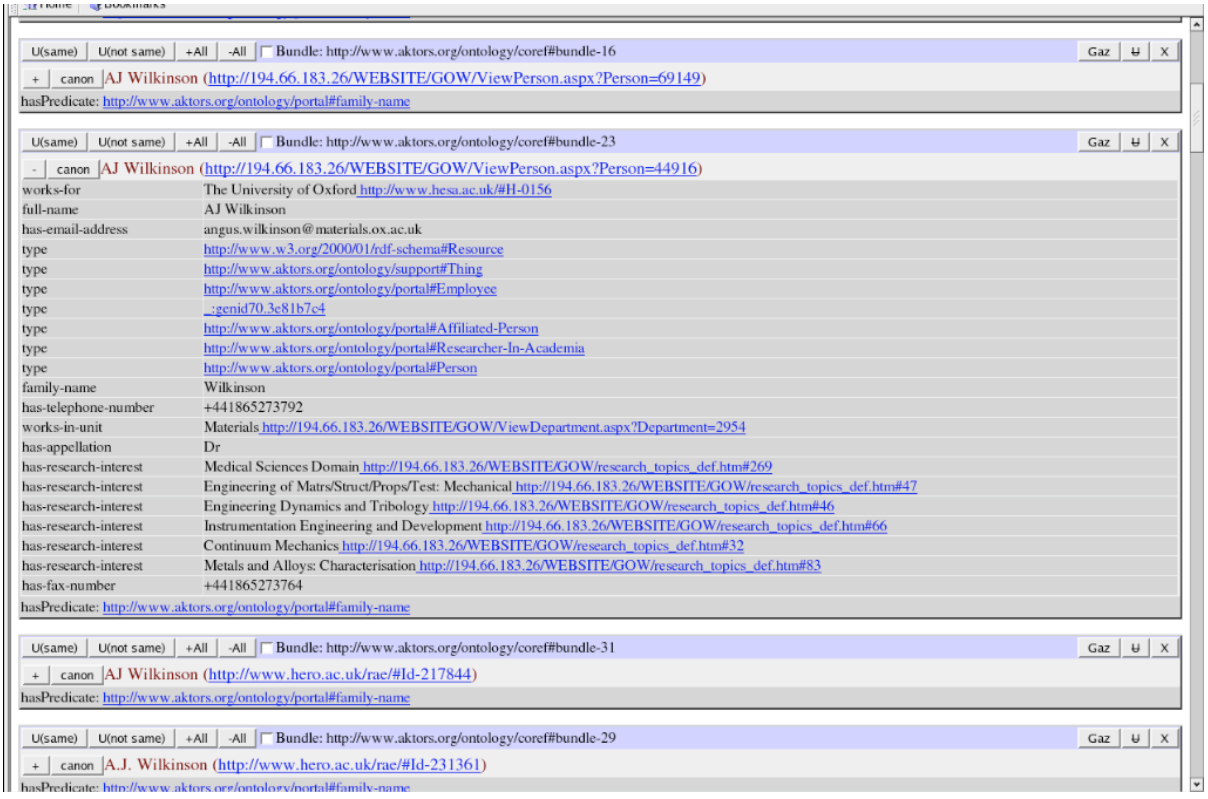

Figure 11. Screen shot of the manual interface, showing several bundles, one of which is displaying all available data.

Figure 12. Screen shot of a single bundle, showing all the operation buttons.

Thus this manual interface (Figure 11, Figure 12) is a key component in the framework. It is a web-based interface that allows the user to visualise bundles, sorted by labels, and perform a number of operations, based on the other services. These include deleting, merging, unmerging, setting non duplicates, setting the canonical reference and creating gazetteer entries. It also has the facility to display all the data associated with any of the references, it performs lookups to find labels for all resources and associations, and provides hyperlinks to all URIs. It provides the user with all the available information in order to be able to make decisions as to whether any bundles represent the same resource. Using this interface the user can perform the entire co-reference resolution process by hand if they so wish.

Typically the user will perform a search for a string (in the figures, we a see a part of the window that has come up in response to a query for "Wilkinson"), which is then presented as a list of bundles, each with one entry. The user now focuses on a particular subset of candidates (those with initials "AJ"). Likely candidates should be close together, as when the user is dealing with names (a common case) the interface orders bundles by initials, while ignoring titles.

By default the interface shows the string, URI, and the predicate, and this can often be sufficient to discard or accept that individual in comparison with the others. Sometimes it is desirable to be able to look at additional information. In this case, it is normal to use the "+" button, which expands the entry to give a visualisation of the related RDF. This has been done for the second "Wilkinson" above.

Having now decided which of the candidates are indeed the same or not, the user checks the bundle box for those entries, and clicks on the "U(same)" or "U(not same)" button to cause them to be merged appropriately.

If the user wishes to go further, then it will be necessary to choose a canonical reference, and this is achieved by simply clicking the "canon" button. With this done, it is possible to construct the RDF for a gazetteer, and this can be achieved by the "Gaz" button.

There are other buttons available to make it as easy to use as possible, such as "~~U~~" to explode a bundle back into separate bundles, and "+All" to expand a number of entries at once.

The interface displays bundles from RDF passed to it either directly or via a URI. The user can bring up the RDF of the manipulated bundles at any point in a new window.

6. CONCLUSIONS

We have described here a set of services that provide a suite for dealing with co-reference problems in an RDF triplestore.

Users can write an end to end script which

- Chooses candidates that may be coreferent;
- Groups them according to predicate or not;
- Chooses a canonical reference against some algorithm;
- Can serve these to other services;
- Constructs a gazetteer for future use;
- Constructs the appropriate owl:SameAs RDF.

Should the user require slightly different components, for example a canonical chooser that used a different algorithm, they can slot it into the structure and still use the other services.

Finally, we have provided a user interface that allows users to interact with the services in a more hands on fashion.

We believe that co-referencing within the semantic web is a growing problem that is only beginning to be appreciated. As the web grows and more, larger, knowledge bases and initiatives appear, the need for an efficient system for managing references will increase. In anticipation of this growing requirement, we have designed and proposed the schema and services outlined in this paper. The use of this system provides a flexible, expandable and readily compatible methodology for coping with inevitable referential inconsistencies.

7. ACKNOWLEDGEMENTS

This work is supported under the Advanced Knowledge Technologies (AKT) Interdisciplinary Research Collaboration (IRC), which is sponsored by the UK Engineering and Physical Sciences Research Council under grant number GR/N15764/01. The AKT IRC comprises the Universities of Aberdeen, Edinburgh, Sheffield, Southampton and the Open University. We also thank Les Carr for his comments and suggestions, and for his involvement in the design of the CRS.

8. REFERENCES

[1] AKT. The akt manifesto. Technical report, 2001. http://www.aktors.org/publications/Manifesto.doc

[2] H. Alani, S. Dasmahapatra, N. Gibbins, H. Glaser, S. Harris, Y. Kalfoglou, K. O'Hara, and N. Shadbolt. Managing Reference: Ensuring Referential Integrity of Ontologies for the Semantic Web. In *13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02),* pages 317-334, Siguenza, Spain, 2002.

[3] H. Alani, K. O'Hara, and N. Shadbolt. ONTOCOPI: Methods and tools for identifying communities of practice. In

Proceedings of the 2002 IFIP World Computer Congress, Montreal, Canada, August 2002.

[4] A. Bagga. Evaluation of coreferences and coreference resolution systems. In *Proceedings of the First Language Resource and Evaluation Conference*, may 1998.

[5] T. Berners-Lee, Hendler J., and O. Lassila. The semantic web. *Scientific American*, may 2001.

[6] D. Fensel, F. van Harmelen, I.Horrocks, G. McGuinness, and P. F. Patel-Schneider. OIL: An ontology infrastructure for the semantic web. IEEE Intelligent Systems, 16(2):38-45, 2001.

[7] S. Harris and N. Gibbins. 3store: Efficient bulk RDF storage. In Proceedings of the 1st International Workshop on Practical and Scalable Semantic Systems (PSSS'03), Sanibel Island, Florida, pages 1-15, 2003.

[8] I. Horrocks. DAML+OIL: A description logic for the semantic web. *IEEE Bull. of the Technical Committee on Data Engineering*, 25(1):4-9, MAR 2002.

[9] M. Koivunen, and E. Miller. W3C Semantic Web Activity. *in proceedings of the Semantic Web Kick-off Seminar*, November 2001.

[10] O. Lassila and R. Swick. Resource Description Framework (RDF) Model and Syntax Specification. W3C recommendation, W3C, feb 1999.

[11] D. McGuinnes, F. van Hermelen. OWL Web Ontology Language Overview. W3C recommendation, W3C, feb 2004.

[12] K. O'Hara, H. Alani, and N. Shadbolt. Identifying Communities of Practice: Analysing Ontologies as Networks to Support Community Recognition. *Proceedings of the World Computer Congress,* 2002.

[13] A. Rahm and A. Bernstein. A survey of approaches to automatic schema matching. *The Very Large Databases Journal,* 10(4):334-350, 2001.

[14] R. Reiter. Equality and domain closure in first order databases. *Journal of the Association of Computing Machinery*, 10(4):334-350, 2001.

[15] Van de Sompel, H. and Lagoze, C. The Santa Fe Convention of the Open Archives Initiative. *D-Lib Magazine*, 6(2), February 2000.

[16] E. Wenger. Communities of Practice: The Key to Knowledge Strategy. Cambridge University Press, 1998.

[17] M. Wooldridge, and N. R. Jennings. Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, vol 10, no. 2, pp. 115-152, 1995.