Managing Reference: Ensuring Referential Integrity of Ontologies for the Semantic Web

Harith Alani, Srinandan Dasmahapatra, Nicholas Gibbins, Hugh Glaser, Steve Harris, Yannis Kalfoglou, Kieron O'Hara and Nigel Shadbolt *

Intelligence, Agents and Multimedia Group (IAM),
Department of Electronics and Computer Science,
University of Southampton,
Southampton SO17 1BJ,
UK
{ha,sd,nmg,hg,swh,y.kalfoglou,kmo,nrs}@ecs.soton.ac.uk

Abstract. The diversity and distributed nature of the resources available in the semantic web poses significant challenges when these are used to help automatically build an ontology. One persistent and pervasive problem is that of the resolution or elimination of coreference that arises when more than one identifier is used to refer to the same resource. Tackling this problem is crucial for the referential integrity, and subsequently the quality of results, of any ontology-based knowledge service. We have built a coreference management service to be used alongside the population and maintenance of an ontology. An ontology based knowledge service that identifies communities of practice (CoPs) is also used to maintain the heuristics used in the coreference management system. This approach is currently being applied in a large scale experiment harvesting resources from various UK computer science departments with the aim of building a large, generic web-accessible ontology.

1 Introduction

In the context of the Semantic Web ([5]), ontologies are a key technology, providing formalisms in which to express metaknowledge about content and resources. Such formalisms will facilitate the knowledge-based enrichment or annotation of such content and resources. This in turn will allow reasoning over annotations. Reasoning will permit the presentation of content and resources to the users that need them, when they need them. This will facilitate the provision of a range of intelligent Web services ([21]). These services could include, for instance, content-based discovery of knowledge sources, natural language querying, or e-marketplaces.

As these services become increasingly prevalent, ontologies will become more important, and ubiquitous. They will increase in scale, and the issue of *ontology engineering* will become evermore important, as distributed teams (some of

* The names of the authors appear in alphabetical order.

whose members will not be experienced in knowledge representation) begin to put them together. Furthermore, we would expect to see the increasing prevalence of ontology *merging*, where a single ontology is created from the union of two or more ontologies. For example, an interdisciplinary ontology may be created from merging ontologies covering the different disciplines, or a topic-specific ontology may be made usable by merging it with a large general purpose ontology. Finally, such resources carry maintenance costs, particularly in fast evolving domains; the merging of new bodies of information with the old ontology must become more or less routine if ontologies' shelf-lives are to be of acceptable length.

It has been suggested [11] that ontology development lifecycle management policies need to be understood in this context. Thus merging, and other associated tasks such as documenting the rationale for the particular design and structure of an ontology or configuration management, need to be written explicitly into lifecycles. In other words, ontology builders should *start* from the assumption that their work will be merged with that of others and in various ways reused.

However, merging ontologies carries with it obvious risks. The fact that ontologies can be expected to have a distributed etiology implies that the coordination of ontology development will be hard, even with the use of the lifecycle policies suggested in [11] or the benefits of any emerging web/ontology standards. In particular, the uses of terms cannot be expected to be consistent across merged ontologies. Indeed, intra-ontological context can cause reference problems. [20] reports such difficulties in e-commerce ontologies: "systematic treatment was required ... to separate terms like *steamers* under clothing appliances from *steamers* under kitchen appliances." The problem is made more acute in ontologies since the standard techniques for coreference resolution from computational linguistics are deprived of the cues they require to work in the context of ontology structure merging [4].

As part of the Advanced Knowledge Technologies project [2], we have developed an ontology for experimental use, including merging of a number of databases and other resources in the field of computer science. The problem of maintaining referential integrity was an immediate challenge, and the domain has provided us with an opportunity to investigate the issue. A combination of approaches to maintaining referential integrity has been used, in a series of steps starting from general defeasible heuristics for initial disambiguation, through to the application of advanced heuristic services to refine the picture.

We will propose a systematic protocol to assess and flag potential mismatches between semantic intent and encoded content that might affect the reliability of the system. In this paper, we shall focus our attention on the problem of multiple identifiers for the same referent represented in a populated ontology.

We can only assume that our methodology cannot but be circular — we can only rely on the knowledge-based system, in particular the populated ontology itself, to inform us of indicators of referential profligacy. Our procedure will start with the most local features of the representation — the actual realisation of the identifier in terms of characters — and move to more global features of how the classes to which instances belong are related. At each step we shall identify a hypothetical relation that the instances might satisfy and cluster the set of individuals by this relation. We then apply the less local information from the ontology in order to further fine-grain the clusters. At each stage, the grain-size of the referential cohorts can be tuned depending on the nature of the decision processes that such a choice might affect.

Obviously the issue of referential ambiguity and integrity is not a new one, and we review a variety of approaches in section 2. Section 3 discusses our own approach in detail, setting out methods for ontology population (section 3.1), clustering candidate sets of coreferences (section 3.2), and the application of general and more knowledge-based heuristics (section 3.3). The experimental investigation and validation of our approach is discussed in section 4.

2 Work and issues related to referential integrity

2.1 Referential integrity and coreference

The problem of reference has been recognised for centuries — indeed much understanding of classical texts depends on knowing whether two names denote the same person. Thus the great Roman orator, statesman, and man of letters, born in Arpinum, Latium in 143BC is known as Cicero but in much writing he is referred to as Tully. Classical texts are full of references to people who just happen to share the same name and often the same occupation. Thus there are two individuals Marcus Porcius Cato one a great grandson to the other and both of whom were Roman statesmen and writers. Despite conventions arising to distinguish between them 'Cato the Elder' also known as 'Cato the Censor' is still a source of potential referential confusion with 'the Younger Cato of Utica'. These examples have not just fueled an industry in the philosophy of language but illustrate the perennial problem of ensuring that expressions refer to the appropriate objects.

Referential integrity is the database-related practice of ensuring implied relationships are enforced, thereby protecting users from the consequences of their own non-omniscience; we have taken the term, in the context of ontologies, to mean the managing of reference so that the domain is sufficiently understood to allow reasoning about underlying objects to go ahead unhindered by referential confusions.

We have already mentioned the difficulty of different areas in the same ontology providing different contexts for class names to be repeated (steamers). It is obvious that merging ontologies will increase opportunities for clashes as the authors will be ignorant of each others' work and naming conventions. The problem is not restricted to naming; relations will throw up the same confusions. For example, if *Boose* is the author of a book published by Springer in 1999, and *Gaines* is the author of a book published by Springer in 1999, is it the same book?

There are a number of related issues here, but we will focus on the problem of *coreference*. Coreference is the problem that arises when two names refer to the same thing: *IJHCS* and *Int.J.Hum.Comp.Studs*; *Shadbolt*, nigel_shadbolt and *Shadbotl*. Heuristics and memories of previous confusions will be of value, but would need to be very sensitive — for instance, it may be that the system takes *Shadbotl* to be an orthographic variant of *Shadbolt*, whereas it may actually be referring to an obscure Aztec god.

This is a particular problem for the semantic web. Firstly coreference, while not introducing the arguably greater error of conflating two different individuals, will still lead to a lack of efficiency and ability to draw the full set of conclusions if unnoticed; a serious problem if ontologies are to be the workhorses of the Web.

Secondly, coreference must be expected to be endemic on the Semantic Web. Entities (resources) are referred to by Uniform Resource Identifiers (URIs; [28]), human-readable strings. These URIs are names, which denote a set of equivalent instances of a resource – mirrored webpages, for example (this redundancy provides a degree of robustness to a distributed system such as the Semantic Web). Semantic Web resources are not necessarily digital artifacts like web pages which can be retrieved by resolving the URI which names that resource; URIs can also denote resources which are physical entities, such as people or organisations. Languages such as the Resource Description Framework (RDF) ([18] are then used to make statements about the resources denoted by these URIs.

While the semantics of the URIs used as names by the Semantic Web require that identical names must refer to the same entity, they do not require that those names be unique. So URIs should not suffer the Cato problem but will frequently face the Cicero=Tully dilemma. Consequently, two data sources on the Semantic Web may contain statements which refer to the same resource, but by different URIs. As URIs need not denote digital objects, it is not always possible to resolve them and determine if two URIs are coreferent by comparing the objects that they denote. In fact, it is current common practice for Semantic Web agents to treat URIs as opaque symbols and to ignore any retrieval semantics which might otherwise be associated with them (for example, the structure of an `http` URI gives the necessary means for an agent to be able to retrieve the object indicated by that URI).

Because the names used by one agent need not be used by another, it may not be possible to directly combine knowledge from different agents, and so knowledge from one agent must be transformed into a form based on the names used by a different agent. The cost of performing inference across the knowledge of more than one agent must now also include the cost of identifying any coreferent names and subsequently rewriting them in a common form.

Another issue is that of anonymous resources: these are the RDF equivalent of using an existentially quantified variable. Anonymity can be treated as a form of name scoping, in which an agent maintains a local scope for the names it uses in order to avoid collisions with the names used by other agents. When a reasoning service needs to use information from a source which does not externally publish the identifiers it uses, it typically makes the assumption that all existentially quantified variables refer to different entities. In RDF, this is accomplished by assigning a unique *generated identifier* or *genid* to each occurrence of an anonymous resource. In doing so, the RDF processor has transformed a source which has attempted to avoid the issue of name uniqueness (by not using names) into one which is guaranteed to use a different unique name to denote an anonymous resource each time one is mentioned. Thus the web is full of potential referential conundrums.

2.2 Traditional knowledge-based systems

As we have noted, the problems that we have set out above as relating to the semantic web are hardly new. The general response in artificial intelligence, for example in the area of knowledge-based systems, is to use a *unique name assumption* [25], whereby there is a 1-1 correspondence between names and domain objects. This is a simplifying assumption that vastly reduces computational complexity, rendering the issue of coreference trivial.

However, this simplifying assumption ignores the problems of scale that the semantic web and the giant datasets that it is intended to handle will throw up. Even a reasonably-sized KBS would have a small cohort of builders, generally not distributed over space, and so the effort of ensuring the unique name assumption holds is minute compared to any attempt to enforce it over the virtual, geographical and temporal range of the web.

Hence it is unlikely that the AI solution to coreference identification problems, neat though it is, will export. This was reflected in the decision not to introduce the unique name assumption into the DAML+OIL language that is intended to represent ontologies on the web. Instead roundabout locutions such as the `daml:sameIndividualAs` and `daml:equivalentTo` relations have been introduced into the language.

2.3 Databases

Similar problems also crop up in database research, for example with the merging of relational and object-oriented databases, or with varying formats from different vendors. If a table contains two instances of *John Smith*, then the question arises whether these are two John Smiths or one. In an OOD, each Smith (if two there be) would have his own object identity, but in a relational database the problem is generally addressed by adding object identifiers (OIDs) dynamically; generating these can cause difficulties, such as hotspots in the DB caused by repeated calls for OIDs that result in timeouts, and create requirements such as the need to have a flexible enough design not to be dependent on any particular storage medium. Alternatively, large Universally Unique Identifiers (UUIDs) can be created, at some cost of space. Problems such as coreference can be eliminated by using UUIDs behind the scenes to, in effect, reinstate the unique names assumption. However, this device does not remove the need to check that coreference has been eliminated after all. Jagadish and Qian ([15])review a variety of tradeoffs between the approaches taken to these and other referential problems in object-oriented and relational databases.

In database technologies the merging of databases often relies on a mediation layer between the heterogeneous structures. There are different approaches to creating this extra layer. One method is to create a common data model, but for the reasons already reviewed such an approach would be impractical over the global expanse of the semantic web. Another would be to specify matching rules translating directly between source and target; this again is likely to be impractical for very large resources because this type of approach, as noted, in effect leave it up to the user to spot instances of coreference.

Another possibility is the use of query languages and mediators that can answer questions about an information source, generally based on a relational algebra and calculus ([7]). Such languages can either take the form of direct extensions to relational formalisms [9], or define operators similar to functions in Frame Programming Languages ([13]). For example, Abiteboul and Kanellakis ([1]) use OIDs as powerful primitives for the database query language IQL; OIDs are used to represent data structures, to manipulate sets and to express computable queries. Jagadish and Qian ([15]) on the other hand integrate declarative global specifications of constraints, including referential integrity, into an OOD, to attempt to allow both efficiency of representation and localised processing for specific applications.

The difficulty, as ever, is to produce referential integrity without too much of a burden on the user. The problem is arguably more acute in the semantic web world, as ontologies are generally more complex and larger than database schemas. Moreover, database approaches to the problem of ensuring referential integrity when merging or addressing multiple resources still tend to be dominated by syntactic rather than semantic considerations ([12]).

2.4 Resource merging tools and techniques

There has been a good deal of research into explicit tools and techniques for ontology merging. Interesting approaches include, among others, Ontomorph, Chimaera, and PROMPT. OntoMorph ([6]) provides two mechanisms for translating different forms of symbolically represented knowledge: "(a) syntactic rewriting via pattern-directed rewrite rules that allow concise specification of sentence-level transformations based on pattern-matching, and (b) semantic rewriting which modulates syntactic rewriting via (partial) semantic models and logical inference supported by an integrated knowledge representation system." OntoMorph can be viewed as a set of rewrite rules coupled with heuristics and a description logic classifier to check subsumption dependencies. Chimaera ([20]), based on Ontolingua ([10]), offers support to the user by diagnosing areas of the source ontologies where coverage or correctness may be affected by a lack of referential integrity, based on heuristics such as similarities of class names. Currently the only relations reported that Chimaera has supported are subclass/superclass relations, and the set of heuristics is fairly limited, but even so experiments have shown that, despite the restricted palette, it has a wide range of application.

As it turns out, the merging of individuals, the main focus of this paper, is not currently supported, but that is flagged in future work.

PROMPT ([12]) has a broader ambition, to provide the user with support — recall that both OntoMorph and Chimaera are diagnostic tools which leave the decision as to what to do up to the user. The potential size of ontologies entails that reducing the amount of input that the user has to produce, whether creative or routine, is essential, and the AKT approach follows that line (section 3). PROMPT supports a propose-and-revise cycle of actions. It first creates an initial list of matches between the source ontologies, like Chimaera, based on class names. It then suggests a remedial operation (the user can override this or specify his own operation in an editing environment), and then PROMPT can perform the selected/defined operation, execute additional changes consistent with the operation, and determine a new set of conflicts (suggesting more solutions to start the cycle once again).

In each of these cases, we should note that the heuristics in use are not particularly sophisticated, and yet are shown by experiments to be effective ([20], [12]); this indicates that our own set of initial heuristics is likely to make a real difference to large ontologies (section 3.2). Moreover, we also use the ontology's structure itself to determine potential referential problems at a later step (section 3.3). The systems currently in existence actually rely quite heavily on the user's input; it is likely that as ontologies become more ubiquitous more of these operations will have to be automated. The research problem will be improving the trade-off between accuracy and the conservation of human resources.

Our own approach resembles in spirit approaches that facilitate semantic interoperability of heterogeneous resources (e.g. legacy software components) by extracting metadata from the components themselves or user assumptions ([14], [31]). Extracting the information implicit in the resources themselves, and providing explicit models of their mutual constraints, is the key to understanding the interoperability constraints.

3 A stepwise process in identifying coreferences

To tackle the problem of identifying coreferences, we devised a stepwise process which begins by populating the ontology with instances and ends with coreferences detection. In essence, this relies on the simple idea that references are first deemed to be similar enough to be considered co-referent by using different indicators of similarity, and a final decision process to integrate these measures of similarity. The process is illustrated in figure 1. Step 1 on the left-hand side is the ontology population step. This task is (semi-) automatic and described in section 3.1 below. The centre of the figure demonstrates step 2 which produces two sets of clusters of potential instance duplicates[1]. The string-based clusters are created using soft string similarity measures, while the generic clusters are

[1] For the sake of clarity we use the term duplicates when we refer to coreferences in this step as this is more commonly used when dealing with database-like structures.

produced by applying a set of generic heuristics. Step 2 is described in more detail in section 3.2.

The third step in our approach makes use of the notion of *communities of practice* (COPs)where the connections between the instances clustered in previous steps are analysed. The degree of overlap between the connection trends is used as a duplication indicator. More about this step in section 3.3.

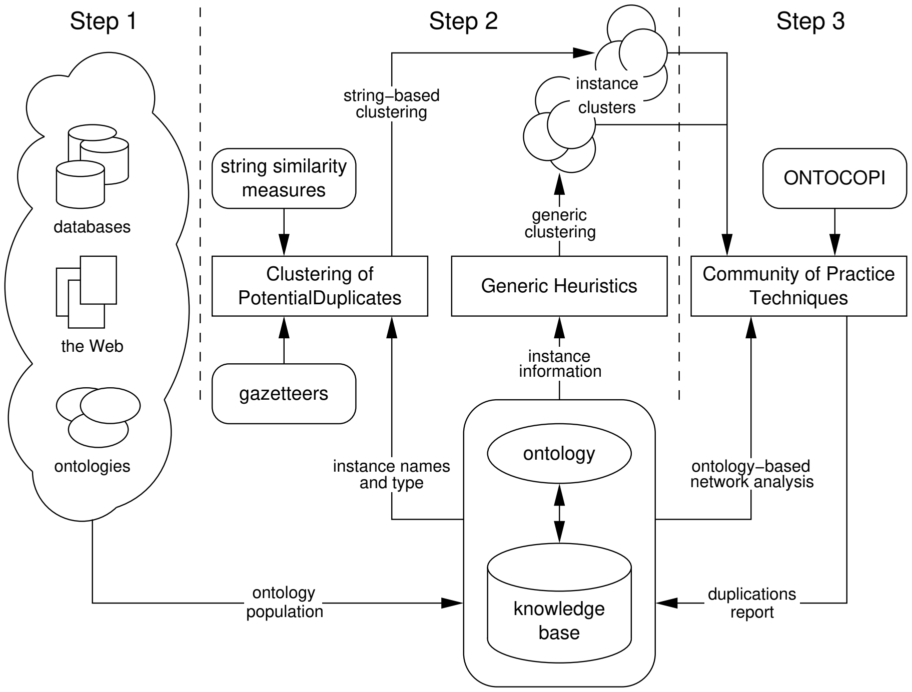


Fig. 1. A stepwise process for identifying coreferences.

3.1 Populating the ontology

In this section we present the procedure for the initial population of the ontology from structured sources such as legacy databases. This process consists of four parts: a) mapping entity types to classes and attributes and relationships to their equivalent properties in the ontology (a discussion of this is beyond the scope of this paper; we encourage the reader to consult [24] for a review of approaches) b) assigning identifiers to strong entities, c) assigning identifiers to weak entities and d) constructing the resulting expressions.

A strong entity is one which possesses a primary key which uniquely identifies it, and so one to which we can assign a unique identifier without introducing coreference problems. We construct such identifiers by concatenating the name of the table which holds the entity set and the values of the attributes that constitute the primary key, and appending them to the identifier that denotes the database. We have no such uniqueness guarantees for weak entities, which do not possess a primary key, and so we generate a different identifier (a genid) for each weak entity that we encounter. We choose this approach over that of referring to weak entities by anonymous means (such as RDF's anonymous resource mechanism) because the later steps in our process need identifiers to be able to refer to coreferents.

Finally, we construct an description for each entity (both strong and weak) in a straightforward way by collecting together their attribute values and relations and rewriting them using corresponding properties from the ontology that were identified in the mapping stage.

3.2 Clustering duplicates and applying heuristics

Before we proceed to describe the clustering methods we use and how they inform our heuristics we briefly discuss approaches that have been common in the Semantic Web community when faced with coreference (or duplicate) identification. There have been two broadly defined tasks: (a) identification of coreferent names, and (b) elimination of such coreference by rewriting statements from one or more sources such that any names which occur in those statements are replaced by the equivalent names used in another source, if such exist. One approach to the problem of identifying coreferences, the Semantic Web community argues, would be to create a centralised service by which data sources can provide a mapping from their names to those used by another source, but this has two key problems: consistency and latency issues involved in the maintenance of such a dictionary, and the inability to establish equivalence of anonymous resources by the simple fact that anonymous resources have no name.

A quite different approach is to use an ontology to guide the design of heuristic rules that will be used to identify coreferences or duplicates. For example, if two data sources declare that two resources with different URIs that represent people have the same social security number, and it is known from the ontology that a social security number can only be used by one person, it can be deduced that the two URIs refer to the same person. The social security number property of a person is an *identifying property* which has a uniqueness constraint such that distinct resources must have different values for that property[2]. A rule that uses an identifying property to infer that two URIs are coreferential is sound because of this uniqueness constraint.

However, this approach requires knowledge of the domain to specify which property is the identifying one. In addition, these might be comparatively rare and searching for them in an unfamiliar ontology is time consuming and error prone. To tackle this, we adopt a more generic view of using the underlying

[2] In DAML terms, identifying properties are subclasses of either `daml:UnambiguousProperty` or `daml:UniqueProperty`.

ontological structures as a basis for defining the heuristic rules. To design these heuristics though, we need to cluster the ontology into sets of potential duplicates. It is to this process that we now turn.

We shall define a set of relations $I_i \subseteq N \times N$ by some procedures indexed by i to induce a clustering of N, *i.e.* we may define a set of subsets of N, wherein names are pairwise related by I_i within each of the subsets. We shall refer to this process as *parallel clustering*, where for each class defined in the ontology, we identify pairs of instances of the class that are similar based on two different criteria. In one we rely on the information encoded in the ontology which we assign to the instances as tuples of attribute-value pairs, thus identifying an instance with the values of all its attributes. In the other we assume that the name given to the instance may be sufficient for its identification[3].

The first clustering method indiscriminately takes all the information in the ontology and for each instance n of a class as input, producing a set of instances of the same class that are related to it, as determined by some procedure indexed by i. Here i could include a thresholding criterion based on the method used for clustering and the degree of slack required of knowledge processes downstream. It is this indifference to the preferential semantics of the attributes that prompts us to call this branch of the parallel clustering "generic" which is reflected in the superscript on I. However, there do exist such generic procedures to be rewritten in terms of rules, like decision trees ([23]), which can be used to ascribe semantics to the clusters that are hidden by our protocol.

We use this clustering method to inform the design of our generic heuristics. For example, a person instance in an underlying academic ontology might also have attributes such as telephone number, postal address, project membership and papers published. These are then used alongside email address mentioned above in the rule for identifying properties and we use the thresholding criterion to identify duplicates. In doing so, we make this procedure (almost) domain-independent as no knowledge of the underlying ontology is required upfront. Domain knowledge is only introduced in the thresholding criterion that the engineer is invited to apply to the generic protocol applied to any ontology.

The second clustering method groups elements using a different relational criterion. In the first instance, this is a simple string-matching criterion, using a dynamic programming algorithm (for example, [29]) to align name strings relative to each other. For instance, we could have two strings n and n' to be related nI_in' by procedure indexed by i which determines how far the strings have to be in terms of the Levenstein string-edit distance (the sum of insertions, deletions and substitutions) [19] [4]. This string-based matching can be supplemented

[3] This assumption is akin to Kripke's arguments for names as rigid designators ([17]), identifying the same referent in all possible worlds. In this case, the possible worlds are distinguished by values of the attributes.

[4] The use of edit-string distances for strings of 4-letter nucleotide alphabets is common in bioinformatics, in which case the intended "semantics" lies in the identification of the corresponding amino-acid sequences.

by a domain-dependent gazetteer for identifying common variants of names of objects.

The intuition behind using this two-fold clustering is the observation that due to legacy issues in populating ontologies, there might be *no overlap* between the attribute values of the same individual represented in the knowledge base.

We now have two sets of clusters each of which can now be explored by methods that heavily draw upon the information in the knowledge base to suggest reasons to the user/engineer as to why identifiers may have a common referent. The task of the steps outlined thus far has been to reduce the computational load of these more computationally intensive processes. These methods though, are not foolproof. Since we treat the underlying ontology as a "black box" we might get an overlap below the threshold criterion defined for identifying duplicates but still the objects are duplicates because the attributes that don't match are contextually equivalent. That is, the attributes do not match and cannot be detected with the clustering methods described here, but the context in which these attributes are defined and the way their objects are connected to each other might reveal potential duplicates. In particular, preliminary experiments with this service has thrown up possible guidelines for ontology population that will carry over much more of the context of knowledge capture to aid this process of determining reference. To further explore this dimension in searching for duplicates we need a mechanism which will allow us to reason about the context in which an object and its attributes are defined, and more crucially, how these are connected. Such a mechanism is described in the following section and is based on the notion of communities of practice.

3.3 Using communities of practice to check for duplicates

In this section, we shall use the relationships registered in the ontology to specify the appropriate relations by which we can either further refine the set of potentially coreferent identifiers, or at least to register the names in the knowledge base with the appropriate relational tags. The procedures adopted can be distinguished from those in section 3.2 in two ways. First, we can look at the classes and relationships that are specific to the domain under consideration and use domain knowledge to structure further analyses. Secondly, we consider the extension of the relationships by composing relationships R, R' between instances represented in the ontology. We compose $n_1 R n$ and $n R' n_2$ to define $n_1 R^*_n n_2$ where R^*_n is a combined relation indexed by the intermediate object n (transitivity is not taken for granted).

ONTOCOPI is an Ontology-based Community of Practice Identifier ([3]), which applies Ontology-based Network Analysis ([3],[22]) to locate the community of practice (CoP) of a given instance. CoPs are informal groups of individuals sharing an interest, task, or a problem ([30]). Here we use ONTOCOPI to produce CoPs for the instances in a given set of potential duplicates $N_k^{(p)s}$ (see 3.2) obtained by the gazetteer and string-based clustering performed on the class of People. For each element from the set of people we define the CoP of person n as the set of people who share a sufficient amount of indirect shared information. This was formulated and described in [22] as a natural number indexed map from the set of people in the ontology to its power set (the set of its subsets, the set of CoPs) obtained by evaluating a graph-traversal algorithm with paths of K edges, each of which is a graphical representation of relations in the ontology.

The degree of overlap between the CoPs can be calculated as a measure of their similarity. If the similarity of two sets is higher than a given threshold (CoP similarity measure) then the two instances in question are regarded as duplicates and are assumed to be the same individual.

The approach of CoP comparison to locate duplication is bound to fail in certain cases. For example, if there are two people with the same surname, and same attributes and similar values for these attributes, that is happen to work in the same environment, on the same projects, and in the same teams, then the two people will have highly overlapping CoPs in which case they will most likely be considered identical. A tactic to help avoid these situations is to assemble sets of heuristics that check for functionally discriminating attributes. In the domain of academic institutions illustrated here one would be if both these names appear as authors on the same paper, which would then set them apart.[5] However, these heuristics might have only limited coverage. For example, in the data set we were considering we did not find any such cases among our 13000 instances. Another more problematic situation is when two people instances are actual duplicates, but are hard to identify by our approach due to the lack of sufficient knowledge about one or both instance, e.g. a person's instance with only a name and title. However, one might say that such instances offer little of value for knowledge services.

The CoPs calculated by ONTOCOPI are generic in the sense that all ontology relationships can be taken into account regardless of their type. This reduces the method to yet another clustering tool when divested of its people and activity-related semantics. Relationships can be selected and weighted as required to put more emphasis on certain relationships while reducing the effect of others. This is desirable if certain relationships are known to lead to coreference identification more than others. For example co-authorship and project membership could be regarded as stronger indications of possible duplication than conference attendance and department membership. The analyses presented here have focused on people as the main class of interest in managing coreference. When viewed as a clustering technique relying on general network analysis, this method is no less applicable in domains where the objects of interest our duplicate instances of projects, papers, journals, etc.

[5] From the mid-eighties, the area of physics called conformal field theory had a number of such instances – A. B. and Al. B. Zamolodchikov and to a less frequent extent V. S. and Vl. S. Dotsenko from the Landau Institute in Moscow and E. and H. Verlinde from Utrecht would have a number of coincident values for these slots.

3.4 Example

We applied the approach described in this paper on a small set of of People instances in our ontology to identify duplicates, using the a threshold of 80% for all measures. The generic clustering technique compared the attribute values of all instances and found no similarities because the selected instances either had very few attribute values to compare, or the values were different, which resulted in very low generic measure values, and therefore no clusters were created.

Instance	ID	Weight
Shadbolt	genid46405	
O'Hara	ECS_02831	10.202
Crow	genid46409	2.0
Shadbolt	ECS_02686	1.825
Löckenhoff	genid46696	1.565
Fensel	genid46692	1.565
Studer	genid46688	1.565
Shadbolt	genid46673	1.565
Shadbolt	genid46651	1.565
Scutt	genid46638	1.565
Motta	genid46621	1.565
Shadbolt	genid46625	1.565
Shadbolt	genid46606	1.565
Stutt	genid46602	1.565
Zdrahal	genid46598	1.565
Motta	genid46594	1.565
Reichgelt	genid46581	1.565
Shadbolt	genid46577	1.565
Hallam	genid46553	1.565
Major	genid46549	1.565

(a)

Instance	ID	Weight
Shadbolt	genid46478	
O'Hara	ECS_02831	10.202
Schreiber	genid46474	2.0
Shadbolt	genid46482	2.0
Shadbolt	ECS_02686	1.825
Löckenhoff	genid46696	1.565
Fensel	genid46692	1.565
Studer	genid46688	1.565
Shadbolt	genid46673	1.565
Shadbolt	genid46651	1.565
Scutt	genid46638	1.565
Motta	genid46621	1.565
Shadbolt	genid46625	1.565
Shadbolt	genid46606	1.565
Stutt	genid46602	1.565
Zdrahal	genid46598	1.565
Motta	genid46594	1.565
Reichgelt	genid46581	1.565
Shadbolt	genid46577	1.565
Hallam	genid46553	1.565

(b)

Instance	ID	Weight
Shadbolt	genid46625	
O'Hara	ECS_02831	10.202
Motta	genid46621	2.0
Shadbolt	ECS_02686	1.825
Löckenhoff	genid46696	1.565
Fensel	genid46692	1.565
Studer	genid46688	1.565
Shadbolt	genid46673	1.565
Shadbolt	genid46651	1.565
Scutt	genid46638	1.565
Shadbolt	genid46606	1.565
Stutt	genid46602	1.565
Zdrahal	genid46598	1.565
Motta	genid46594	1.565
Reichgelt	genid46581	1.565
Shadbolt	genid46577	1.565
Hallam	genid46553	1.565
Major	genid46549	1.565
Shadbolt	genid46536	1.565
Stutt	genid46532	1.565

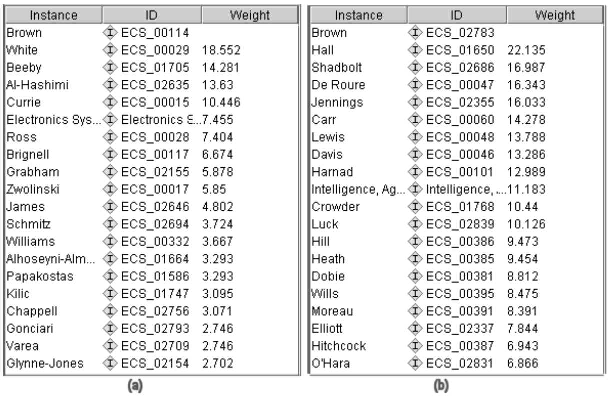
(c)

Fig. 2. ONTOCOPI in action: Communities of Practice for three instances: (a) N.R.Shadbolt (b) NR.Shadbolt (c) N.Shadbolt.

The String Similarity Measure create two small clusters. The first cluster contained the three person instances N.R.Shadbolt, NR.Shadbolt, and N.Shadbolt, with $(11/12) * 100\% = 91.6\%$ and $(10/12) * 100 = 83\%$ string similarity of the second and third string respectively relative to the first[6], while the second cluster contained A.Brown and P.Brown with a string similarity of 85%.

The final stage is to compare the CoPs of the instances in each cluster. The CoPs in figure 2 were computed in ONTOCOPI for the three instances of *Shadbolt* in the first cluster using the automatic relation setting and a link threshold of 4 (i.e. looking for paths of 4 edges or fewer). The left column in the figure shows the name of the instance, the column in the middle displays the system's generated unique ID, and the right column displays the CoP values as calculated by ONTOCOPI. Only the first 20 entries in each CoP will be considered. It can be seen that there are only 3 instances in CoP (a) that are not in (b), and 3 in (b) that are not in (a). The similarity of the two CoPs can be measured as $(34/40) * 100\% = 85\%$, and 90% is the similarity of (a) to (c). Because these COP-similarity values are above or equal to the 80% threshold set earlier, the three *Shadbolt* instances in question will be regarded as duplicates.

[6] String Similarity = 1- the ratio of the Levenstein string distance and the total string length.



Instance	ID	Weight
Brown	ECS_00114	
White	ECS_00029	18.552
Beeby	ECS_01705	14.281
Al-Hashimi	ECS_02635	13.63
Currie	ECS_00015	10.446
Electronics Sys...	Electronics S...	7.455
Ross	ECS_00028	7.404
Brignell	ECS_00117	6.674
Grabham	ECS_02155	5.878
Zwolinski	ECS_00017	5.85
James	ECS_02646	4.802
Schmitz	ECS_02694	3.724
Williams	ECS_00332	3.667
Alhoseyni-Alm...	ECS_01664	3.293
Papakostas	ECS_01586	3.293
Kilic	ECS_01747	3.095
Chappell	ECS_02756	3.071
Gonciari	ECS_02793	2.746
Varea	ECS_02709	2.746
Glynne-Jones	ECS_02154	2.702

(a)

Instance	ID	Weight
Brown	ECS_02783	
Hall	ECS_01650	22.135
Shadbolt	ECS_02686	16.987
De Roure	ECS_00047	16.343
Jennings	ECS_02355	16.033
Carr	ECS_00060	14.278
Lewis	ECS_00048	13.788
Davis	ECS_00046	13.286
Harnad	ECS_00101	12.989
Intelligence, Ag...	Intelligence, ...	11.183
Crowder	ECS_01768	10.44
Luck	ECS_02839	10.126
Hill	ECS_00386	9.473
Heath	ECS_00385	9.454
Dobie	ECS_00381	8.812
Wills	ECS_00395	8.475
Moreau	ECS_00391	8.391
Elliott	ECS_02337	7.844
Hitchcock	ECS_00387	6.943
O'Hara	ECS_02831	6.866

(b)

Fig. 3. ONTOCOPI in action: Communities of Practice for two instances: (a) A.Brown and (b) P.Brown.

If the similarity between CoPs is low, then no duplication will be reported. Figure 3 presents the output of ONTOCOPI for the two *Brown*s in the second cluster. After calculating the CoPs of both instances, no overlap was found, and hence the two Browns will be regarded as names of different people.

4 Experimental Deployment

The work reported in this paper is part of the UK funded Advanced Knowledge Technologies (AKT) Interdisciplinary Research Collaboration (IRC)). This is a six year programme of research begun in October 2000 between five UK Universities - Aberdeen, Edinburgh, Sheffield, the OU and Southampton. One of the goals of the project is to demonstrate the sorts of knowledge intensive technologies that might be deployed on the web to support knowledge management. A number of broad problem opportunity areas have been identified as offering contexts in which to demonstrate these technologies. Unsurprisingly one of these was the use of our own methods to help organise and facilitate the knowledge management practices within our own IRC.

Given the central role our IRC was advocating for ontologies within the semantic web the decision was taken to build a reference ontology for the AKT project. This has evolved over the past 18 months and there is now a sustained effort to populate it with instances from the participating parts of all five of the partner sites. It was agreed that Southampton would be the first site to populate the reference ontology and much of the work reported in this paper arose out of that effort. During the first phase of ontology instantiation something in the order of 18000 instances were included. The primary objects of interest in the AKT ontology include people, publications, projects, research interests meeting events. A phase of automatic coreference management reduced this number by around 6000. The more knowledge intensive phase exploiting communities of practice eliminates another less quantitatively substantial but nevertheless important set of duplicates - those relating to personnel with large publication outputs.

We quickly recognised that there was a need for services that were constantly running to check for changes in the entities held in our various resources - for example our publications database, a database of personnel, and the projects announced and described in particular areas of the Departmental intranet. The question was how far could these methods scale and would some of our services become even more useful if applied to larger instantiated ontologies.

The aim is to populate the AKT reference ontology with instances from each partner site. Since each site has very different resources that hold the information we are interested in we have adopted two main harvesting strategies. The first develops extraction scripts for each site to extract content of the type required for the reference ontology. Often, as might be expected, these are PERL scripts and varieties of relational database query templates. The second approach uses the maintained web sites of each partner as the primary resource. A visual scripting language tool DOME has been developed [27] that is trained on the prototypical page structure of the various partners' web pages. Thus each site has a set of pages that describe or enumerates its staff. Although the structure and format is different for each site it is a relatively straightforward process to train DOME to extract the content and realise it as canonical RDF that maps directly to our ontology.

The daily updates provided by these scripts are already being published on our AKT intranet. The design and implmentation of this referential integrity acrchitecture has allowed us to specify the protocols by which harvesting of data is conducted, which allows a more controlled management of the knowledge repository. We are exploring the scalability of this service.

Our reference ontology can serve not just for the Departments involved in AKT but also the wider UK Computer Science community. There are already bodies in existence that would very much like to have available via a set of annotated web pages the sort of information that our ontology embodies. Examples include current UK computer science faculty, declared areas of research interest, or currently active projects etc. This real time harvesting supplemented by an ontology reference checking service may provide a much better snapshot of actual reality than the *mandraulic* process currently in operation where Departments are asked to submit this same information in a variety of formats for all sorts of purposes. If a community ontology has perceived value one can enter a virtuous spiral where the community begins to use it as *the* reference model to help annotate their own content making subsequent harvesting easier.

The other advantage of the large scale deployment we hope to see and will certainly have for the AKT partners is the provision of a variety of other knowledge services that use these populated ontologies. An obvious example is to extend the COP analysis across various partner sites to see what sorts of inter institutional COPs emerge. This helps us understand the research landscape and the current and shifting patterns of cooperation and intellectual influence [3].

5 Conclusions

We have described a methodology for helping with the management of referential integrity and applied to medium scale ontologies. We advocated a mixture of statistical, string based and AI methods involving human mediation only at the point of key decisions regarding the collapse or otherwise of referential duplicates. Populating ontologies of a reasonable size is required if we are to run many of the knowledge services imagined in the vision being advocated for the Semantic Web.

Automatic methods that harvest content from the web with respect to any reference ontology are bound to be promiscuous, potentially generating thousands of duplicates in only medium sized contexts. It is important that automatic methods of the sort described here perform much of the initial pruning. Thereafter, we advocate the use of knowledge intensive methods to present analyses to help effect decisions about whether to collapse or keep entities distinct.

There are assumptions that some may question in this work. One is the extent to which reference ontologies will be useful at all. In previous work some of the authors have presented a system, APECKS, developed by Tennison [26] where the emphasis was on the collaborative construction of ontologies and in which the expectation was that the class structures and attributes of the ontologies themselves were the objects of debate. APECKS takes a different line to most ontology servers, in that it is designed for use by domain experts, possibly in the absence of a knowledge engineer, and its aim is to foster and support debate about domain ontologies. To that end, it does not enforce ideals of consistency or correctness, and instead allows different conceptualisations of a domain to coexist. Under this view the ontologies would be rather different between users and maintaining different ontologies for different users could be as important as using agreed common reference ontologies across users. However, the issue of populating these idiosyncratic ontologies with instances still arises. Indeed the problem of referential integrity becomes one that literally has to be re-run for each variant.

The content of real value in an ontology that is built and maintained by an organisation or community are objects such as people, projects, products, processes and publications. These objects are typically comprised of many attributes and relations. They may be described from multiple perspectives in different resources on the web. It is important to determine whether these objects are distinct or the same. We think that the methods outlined here are well suited to supporting the increasingly complex detective work of establishing referential identity. We have detailed how we hope to demonstrate the utility of these services in a community wide knowledge harvesting and publication activity mediated via a reference ontology.

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. The views and conclusions contained herein are those of the authors and should not be intepreted as necessarily representing official policies or endorsements, either expressed or implied, of the EPSRC or any other member of the AKT IRC.

References

1. S. Abiteboul and P. Kanellakis. Object identity as a query primitive. In *Proceedings of the International Conference on Management of Data, ACM SIGMOD, Portland, OR, USA*, 1989.
2. AKT. The akt manifesto. Technical report, 2001. http://www.aktors.org/publications/Manifesto.doc
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, HendlerJ., and O. Lassila. The semantic web. *Scientific American*, may 2001.
6. H. Chalupksy. OntoMorph: A Translation System for Symbolic Knowledge. In *Proceedings of the 17th International Conference on Knowledge Representation and Reasoning (KR-2000), Colorado, USA*, April 2000.
7. E.F. Codd. Relational completeness of data base sublanguages. In RustinR., editor, *Database Systems*. Prentice-Hall, 1972.
8. DARPA. DARPA Agent Markup Langugage. Technical report, DARPA, mar 2001.
9. U. Dayal. Queries and views in an object-oriented data model. In R. Hull, R. Morrison, and D. Stemple, editors, *Database Programming Languages: Proceedings of the 2nd International Workshop*. Morgan Kaufmann, 1989.
10. A. Farquhar, R. Fikes, W. Pratt, and J. Rice. The Ontolingua Server: a Tool for Collaborative Ontology Construction. In *proceedings of the 10th Knowledge Acquisition Workshop, KAW'96,Banff,Canada*, November 1996. Also available as KSL-TR-96-26.
11. M. Fernandez-Lopez, A. Gomez-Perez, and M-D. Rojas-Amaya. Ontology's crossed life cycles. In *Proceedings of the 12th International Conference on Knowledge Engineering and Knowledge Management (EKAW'00), Juan-les-Pins, France*, pages 65–79. Springer, 2000.
12. N. Fridman-Noy and M. Musen. PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In *Proceedings of the 17th National Conference on Artificial Intelligence, (AAAI'00), Austin, TX, USA*, July 2000.
13. G. Gardarin, F. Machuca, and P. Pucheral. Ofl: A functional execution model for object query languages. In N.J. Carey and A.S. Schneider, editors, *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, San Jose, California, USA*, pages 59–70. ACM Press, may 1995.
14. S. Heiler, R.J. Miller, and V. Vintrone. Using metadata to address problems of semantic interoperability in large object systems. In *Proceedings of the First IEEE Metadata Conference, Silver Spring, Maryland, USA*, 1996.

15. H. Jagadish and X. Qian. Integrity maintenance in an object-oriented database. In *Proceedings of the 18th International Conference on Very Large Databases, Vancouver, Canada*, pages 469–481, August 1992.
16. G. Kappel and M. Schrefl. Local referential integrity. In *Proceedings of the International Conference on Conceptual Modeling / the Entity Relationship Approach*, pages 41–61, 1992.
17. S.A. Kripke. *Naming and Necessity*. Oxford: BasilBlackwell, 1980.
18. O. Lassila and R. Swick. Resource Description Framework(RDF) Model and Syntax Specification. W3c recommendation, W3C, feb 1999.
19. V.I. Levenstein. Binary codes capable of correcting deletions, insertions and reversals. *Cybernetics Control Theory*, 10:707–710, 1966.
20. D. McGuinness, R. Fikes, J. Rice, and S. Wilder. An Environment for Merging and Testing Large Ontologies. In *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning (KR-2000), Colorado, USA*, April 2000.
21. E. Motta, S. Buckingham-Shum, and J. Domingue. Ontology-driven document enrichment: principles, tools and applications. *International Journal of Human-Computer Studies*, (52):1071–1109, 2000.
22. K. O'Hara, H. Alani, and N. Shadbolt. Identifying Communities of Practice: Analysing Ontologies as Networks to Support Community Recognition. In *Proceedings of the 2002 IFIP World Computer Congress, Montreal, Canada*, August 2002.
23. J.R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.
24. A. Rahm and A. Bernstein. A survey of approaches to automatic schema matching. *The Very Large Databases Journal*, 10(4):334–350, 2001.
25. R. Reiter. Equality and domain closure in first order data bases. *Journal of the Association of Computing Machinery*, 27:235–249, 1980.
26. J. Tennison, K. O'Hara and N. Shadbolt APECKS: Using and Evaluating a Tool for Ontology Construction with Internal and External KA Support. *International Journal of Human-Computer Studies, In Press*
27. T. Leonard and H. Glaser Large scale acquisition and maintenance from the web without source access Workshop 4, Knowledge Markup and Semantic Annotation, K-CAP 2001
28. Sollins,K. and Masinter,L. Functional Requirements for Uniform Resource Names. RFC 1737.
29. R.A. Wagner and M.J. Fischer. The string-to-string correction problem. *Journal of the ACM*, 21:168–173, 1974.
30. E. Wenger. *Communities of Practice: The Key to Knowledge Strategy*. Cambridge University Press, 1998.
31. I-Y. Yao, K-T. Ko, R. Neches, and R. MacGregor. Semantic interoperability scripting and measurements. In *Proceedings of the Working Conference on Complex and Dynamic Systems Architecture, Brisbane, Australia*, 2001.