Preserving Linked Data on the Semantic Web by the application of Link Integrity techniques from Hypermedia

Rob Vesse, Wendy Hall, Leslie Carr
Intelligence, Agents & Multimedia Group
School of Electronics & Computer Science
University of Southampton
Southampton
SO17 1BJ
{rav08r,wh,lac}@ecs.soton.ac.uk

ABSTRACT

As the Web of Linked Data expands it will become increasingly important to preserve data and links such that the data remains useful. In this work we present a method for locating linked data to preserve which functions even when the URI the user wishes to preserve does not resolve (i.e. is broken/not RDF) and an application for monitoring and preserving the data. This work is based upon the principle of adapting ideas from hypermedia link integrity in order to apply them to the Semantic Web.

General Terms

Experimentation, Reliability, Design

Keywords

Semantic Web, Linked Data, Link Integrity, Preservation

1. INTRODUCTION

The Web of Linked Data is characterised by the interlinking between disparate heterogeneous data sources and the fact that the links between the data sources are one of the primary mechanisms for navigating through this data space. Since links are essential to the Web of Linked Data we believe that it is important to have mechanisms in place to maintain link integrity. The aim of link integrity is to ensure that a link works correctly in that traversing the link takes you to a resource and that as far as possible the resource is the one intended by the provider of the link. On a larger scale, link integrity deals with the overall integrity of interlinked datasets such as documents within a Content Management System (CMS) or the linked data sets available on the Semantic Web. Therefore link integrity is one way of ensuring data integrity within the overall system which in our use case is linked datasets.

Link integrity is an existing and well known problem from hypermedia where there were two problems to be dealt with - dangling links and the editing problem. Dangling links are the most well known problem and are regularly experienced by users on the Web as they find themselves presented with an HTTP error as the link they followed pointed to a resource which cannot be retrieved. The editing problem refers to the situation in which the content at the end of the link is changed so that it is no longer what the creator of the link intended to link to. Both these issues affect the Semantic Web since on the Semantic Web everything is interlinked data, therefore data is immediately susceptible to dangling links. The editing problem becomes much more problematic on the Semantic Web since anyone can make a statement about anything so the meaning of things on the Semantic Web is subject to semantic drift[1].

Due to the interlinking between data on the Semantic Web we show that it is possible to exploit the data model such that links themselves can be used to recover missing data in the event of a dangling link being encountered. This provides for a means to retrieve the data that was/may have been at a given URI even if that URI is no longer resolvable. Using this approach for locating data about a URI we are able to preserve and monitor data about a URI from multiple sources and to recover data about URIs that are no longer functioning as described in Section 3.1.

The Semantic Web also introduces two additional problems in link integrity specific to linked data. The first of these is URI Identity & Meaning - what does a URI mean and does this meaning actually matter to the applications that use it and the data that contains it - which is very much an open research debate and beyond the scope of our current work. The second is the co-reference problem which refers to the situation in which some 'thing' we wish to make statements about has multiple URIs that could be used for it. In this work we utilise existing work in this area of research as part of our algorithm for preserving linked data.

Section 2 of this paper covers the related work in link integrity for hypermedia and the Semantic Web which we use ideas from in Section 3 to design and develop our algorithm and the AAT software. Section 5 outlines our plans for future research in this area and we conclude in Section 6 by discussing the potential benefits of link integrity for the Semantic Web.

2. RELATED WORK

Link integrity in hypermedia first received serious attention in the late 1980s and early 1990s primarily from researchers in the open hypermedia community. Systems like Microcosm [11] and HyperG [17] were among the first to consider the issue in depth, Davis's thesis [9] and Kappe's

[1]See the discussion on the W3C Semantic Web Mailing List for an example of this - `http://lists.w3.org/Archives/Public/semantic-web/2009May/0315.html`

WWW2010, April 26-30, 2010, Raleigh, North Carolina.
.

1995 paper [16] provide examples of link integrity in open hypermedia. The widespread growth of the World Wide Web [5] in the mid-1990s led to some new research but as search engines became commonplace towards the end of the decade research interest dwindled. It was perceived that users did not care sufficiently to warrant research into the problem as they could locate missing resources effectively using search engines, in addition the scale of the Web by that time was simply too vast for many proposed solutions to handle. Davis's survey [10] provides a good overview of the state of this research as of the end of the 1990s. Another reason for the decline in research was that the fact that links could fail was one of the reasons the Web was able to expand as fast as it did since it didn't matter if links failed and produced the familiar HTTP 404 error; users were able to publish content without worrying about whether their links to external content were valid.

Ashman's 2000 paper [4] which discusses link integrity with particular reference to electronic document archives provides both a useful survey of existing work and describes a key motivation for ongoing research. As more document collections were translated into digital forms and placed onto intranets people once again started to be concerned about link integrity. Users wanted assurances that links into the document archives would work consistently and ideally links out of the archives would work correctly as well since it may not be possible to alter the archived documents without invalidating the integrity of the archive.

In this vein Veiga and Ferreira [24, 25] discuss the possibility of turning the Web into an effective knowledge repository by use of replication and versioning. Their work follows on from earlier work such as Moreau & Gray's [19] which proposed limited use of replication and versioning but had significant reliance on author and user involvement in the process. In Veiga & Ferreira's work there is no requirement for author involvement in the process, only the end user need use a browser plugin to indicate the content they wish to replicate and preserve. Their results showed that the user could preserve the sections of the Web they were interested in with no perceivable performance impact - on average there was only a 12ms increase in retrieval time for resources. In Section 3 we discuss using an approach of this kind for the Semantic Web.

Phelps & Wilensky introduced the concept of lexical signatures for Web pages in their Robust Hyperlinks paper [21]. They compute the lexical signature of a page and append it to all links to that page so that in the event of the link failing a browser plugin can use the signature to relocate the page using a search engine. The obvious flaw in their work was that it required rewriting all the links on the Web but Harrison & Nelson later showed that these signatures need only be computed Just-in-Time (JIT) when a link fails [12]. In their Opal system the signatures can be computed JIT by retrieving cached copies of the pages from a search engine cache, computing the signature and then using search engines to relocate the page. As discussed in Section 3.1 a JIT style approach can be effectively used to recover linked data about a URI.

2.1 Semantic Web Research

Unlike the traditional Web it is not possible for semantic search engines like Sindice [22] and Falcons [8] to fulfil the same role as document search engines because the users in the Semantic Web domain are typically client applications rather than humans. When a human encounters a dead link they usually navigate to a search engine and enter an appropriate search phrase to find alternative sources of information. For a client application encountering a dead link they will typically have no concept or how/where to find alternative sources of information and URIs for linked data are not always ideal for searching upon compared to textual search for documents. It should be noted that as with the existing Web if the Web of Linked Data is to undergo a massive expansion in the same way things must be allowed to fail but this does not mean we shouldn't attempt to mitigate the problem as far as possible.

In terms of the Semantic Web there has been research into the versioning and synchronisation of RDF data which is relevant to aspects of our work such as Tummarello et al's RDFSync [23] which is an algorithm for efficiently synchronising changes in RDF between multiple machines. This shows that change detection in RDF is non-trivial due to the inherent data isomorphism caused by the use of blank nodes but also shows that it can be achieved in an efficient manner. More recent research from Papavassiliou et al [20] has shown that using information about very basic changes in the RDF - such as that provided by systems like RDFSync or All About That (see Section 3.2) - can be used to build applications which provide useful information to end users. In the case of Papavassiliou's et al's paper they built a system which furnished users with high level descriptions of how RDFS vocabularies have changed in order to aid users in working with such vocabularies. In addition there are systems like the Talis Platform[2] which is a Semantic Web store that implements a versioning mechanism whereby updates can be made via a Changeset protocol [1]. As part of this protocol they utilise a useful lightweight vocabulary for publishing changes in RDF data as RDF which as will be discussed in Section 3.2.3 we reuse in our own system.

Regarding Semantic Web specific link integrity problems the research has largely focused on the co-reference problem. Since there are many organisations publishing similar data semantically (bibliographic databases being a prime example) there are frequently many URIs for a single entity such as an author. Co-reference research aims to develop ways to efficiently and accurately determine URI equivalences and refactor the data or republish this information to help other Semantic Web applications. There are several competing philosophies ranging from the Okkam approach described by Bouquet et al [7] which advocates universally agreed URIs for each entity to the Co-reference Resolution Service (CRS) approach of Jaffri et al [15] which determines co-referent URIs and republishes the information in dedicated triple stores. The CRS approach taken by the ReSIST project[3] within the RKB Explorer[4] application has potential for use in link integrity as the information provided by a CRS could be utilised in a JIT fashion as in Harrison & Nelson's work and we demonstrate how this can be done in Sections 3 and 4.

In terms of link maintenance for the Semantic Web there has been some research in the form of the Silk framework by Volz et al [28] which is a framework for computing links between different datasets. Their approach allows users to stipulate arbitrarily complex matching criteria to do entity matching between datasets, the links produced from this can

[2] `http://www.talis.com/platform`
[3] `http://www.resist-noe.org/`
[4] `http://www.rkbexplorer.com`

then be published via a CRS style service or added to the relevant datasets. As proposed in their later paper [27] this can be used as part of a link maintenance strategy, the possibility of combining this with our approach is discussed in Section 5. In a similar vein Haslhofer and Popitsch's DSNotify system [14] can monitor linked resources and inform the application when links are no longer valid using feature based similarity metrics like the Silk framework.

3. METHOD

As we have discussed it is not realistic to maintain link integrity in a pre-emptive way since such solutions have been consistently shown not to scale to Web scale in previous work. Therefore the focus must be on recovery in the event of failure and preservation to guard against the loss of data which is considered interesting/useful to end users. As the amount of data in the Web of Linked Data starts to expand massively - particularly with linked data being adopted by an increasing number of major organisations - we expect that as with the early document web there'll be an increasing amount of content published by both big companies and individuals. Just like the document web this explosion of content will most likely include much content that is poorly maintained and will lead to increasing numbers of broken links. We have two connected goals in this work 1) to provide a means to retrieve resource descriptions in the form of linked data about a URI even when the the URI is non-functional and 2) to provide the means for an end user to preserve and version these descriptions. To attempt to solve this problem we present an expansion algorithm for retrieving Linked Data about a URI even if that URI itself has failed in Section 3.1 and a preservation system built using this algorithm in Section 3.2.

3.1 Expansion Algorithm

Since the goal of this work is to preserve linked data it was deemed essential that as far as possible we leverage existing linked data technologies and services in order to effect this preservation. To this end we designed a relatively simple algorithm which uses simple crawling techniques which are directed by a user definable expansion profile (see Definition 1). Our aim with this algorithm is to provide resource descriptions of a URI regardless of whether the URI itself is dereferenceable.

Even in the case where a URI is used only as an identifier in the description of another resource and is not itself dereferenceable it is likely that we can still retrieve some data about it. The fact that a URI is minted only as an identifier and that the person/organisation minting the URI does not provide the means to dereference the URI does not affect our ability to find data about it assuming that the identifier is used elsewhere i.e. it is reused as part of linked data.

Definition 1. An expansion profile is a Vocabulary of Interlinked Datasets (VoID) description of a set of datasets and linksets that should be used to locate linked data about the URI of interest. The VoID description may be optionally annotated with additional properties which affect the behaviour of the algorithm.

Drawing on ideas described in Alexander et al's Vocabulary of Interlinked Datasets (VoID) [3] about the way it can be used to direct crawlers we decided to use VoID as the primary means of expressing an expansion profile. We introduce a couple of additional predicates since we require the means to allow end users to specify some basic characteristics of how the algorithm should behave and there is a type of service we need to express which is not contained in the VoID ontology. VoID has concepts of Datasets and Linksets, the former represent a set of data which may have SPARQL endpoint(s) and/or URI lookup endpoint(s) while the latter represent the types of interlinkings between datasets. What VoID does not have a means to express is the location of a service provided by a dataset which allows an application to retrieve URIs which are considered equivalent to a given URI - this we term a URI discovery endpoint (see Definition 2). A discovery endpoint differs from a lookup endpoint in that the latter is expected to return everything the dataset knows about the given URI as opposed to only returning equivalent URIs. Examples of existing discovery endpoints on the Semantic Web include RKBExplorer's CRSes [15] and sameAs.org[5]. Another key difference between a lookup and discovery endpoint is that links discovered from a discovery endpoint are considered to be on the same level of the crawl for the purposes of the algorithm i.e. they do not have increased depth relative to the URI that discovery is performed upon. By this we mean that the execution of the algorithm results in performing a breadth-first depth-limited linked data crawl starting from a given URI - in this tree structure a discovery endpoint introduces sibling nodes for a URI while a lookup endpoint introduces child nodes for a URI.

Our other extensions to VoID allow individual datasets/linksets to be marked as ignored (the algorithm will not use them) and for the user to define to what depth the algorithm should crawl to (defaults to 1). These extensions are defined as part of the AAT schema detailed in Section 3.2.1.

Definition 2. A URI discovery endpoint is an endpoint that when passed a URI returns a Graph containing equivalent URIs of the input URI typically in the form or `owl:sameAs` links.

As already stated the actual algorithm is a simple crawler which uses the input expansion profile as a guide to which potential sources of linked data it should use to try and find data about the URI of interest - this procedure is detailed in Algorithm 1. Note that the algorithm does not terminate in the event of an error retrieving data from a particular URI/endpoint and simply continues, by doing this it is still possible to retrieve some data even if the starting URI does not return a valid response. The algorithm will continue and issue queries about the URI to the various endpoints described in the given expansion profile so unless the URI refers to a document that had very poor linkages or was not indexed by the semantic search services used some RDF will be returned. This approach has similarities to the JIT style approach of Harrison & Nelson [12] in that there doesn't need to be any foreknowledge of the URIs you wish to recover data about when you discover they are broken since by utilising the caches and lookup services of relevant datasets it is still possible to recover data about the URI.

The basic behaviour of the algorithm is only to follow `owl:sameAs` and `rdfs:seeAlso` links but the end user can specify that any predicate be treated as a link to follow by the specifying an appropriate VoID linkset in their expansion profile.

[5] `http://www.sameas.org`

Algorithm 1 Expansion Algorithm

```
Require: URI, Expansion Profile
ToExpand as a set of pairs of URIs and Depths
while ToExpand ≠ ∅ do
    Remove first pair from ToExpand
    if Graph with URI is already in the Dataset then
        Continue
    if Depth >Max Depth then
        Continue
    Retrieve the Graph at the URI
    Add the Graph to the Dataset
    for all Triples in Graph do
        if Triple is a Link then
            Add a new pair to ToExpand
    for all Datasets in Expansion Profile do
        if Dataset has a SPARQL Endpoint then
            Issue a DESCRIBE for the URI against the Endpoint
            Add resulting Graph to the Dataset
            Process the Graph for additional Links
        if Dataset has a Lookup Endpoint then
            Issue a Lookup for the URI against the Endpoint
            Add resulting Graph to the Dataset
            Process the Graph for additional Links
        if Dataset has a Discovery Endpoint then
            Issue a Discovery for the URI against the Endpoint
            for all Equivalent URIs do
                Add a new pair to ToExpand
return Dataset
```

There are already some existing systems which work in a similar way to our algorithm such as the sponger middle ware using in Virtuoso [2]. The main difference between our algorithm and algorithms such as those in the Virtuoso sponger is that our algorithm is only interested in linked data and it does not infer/create any additional data. Unlike the Virtuoso sponger it does not attempt to turn non-linked data into RDF and it does not do any inference over the data it returns, it is designed only to find and return (in the form of an RDF dataset) linked data about the URI of interest. Yet as expansion profiles may reference any datasets and associated endpoints they wish there is no reason why a user could not direct our algorithm to utilise a service like URIBurner[6] which uses the Virtuoso sponger in order to get the benefits of the additional inferred data.

3.1.1 Default Profile

Since the end user of such an algorithm may not always know where to look for linked data about the URI they are interested in the algorithm has a default expansion profile which is used in the case when no profile is specified. This profile uses 3 data sources which are in our opinion important hubs of the Web of Linked Data:

- DBPedia[7] - The DBPedia SPARQL endpoint is used to lookup URIs
- Sindice[8] Cache - The Sindice Cache API[9] allows the retrieval of Sindice's cached copy of the RDF from a URI.
- SameAs.org[10] - SameAs.org provides a URI discovery endpoint (see Section 3.1 and Definition 2) which can be used to find URIs which are equivalent to a given URI

The default profile[11] has a max expansion depth of 1 which means it only considers URIs which are immediate neighbours of the starting URI.

In the case where the end user does know which linked data sources will have useful information about the URI they can specify their own expansion profile which is used instead of the default profile. In this case the algorithm will use the datasets and linksets they define in the profile to discover linked data about the URI of interest, for example if attempting to recover data about a person it may be useful to follow `foaf:knows` links.

3.2 Preservation

The preservation approach taken is to allow the end user to monitor and preserve a set of linked data that they are interested in. The data is preserved not at the data source but rather at a local level on the users server with the user able to republish this data as they desire. This is in line with the ideas of Veiga & Ferreira [25] in that the end user specifies the parts of the Web they want to preserve and then the software takes care of this. The data must be preserved in such a way that the original data can be efficiently extracted from it and sufficient information to provide versioning over the data is kept.

In the Semantic Web domain the objects of interest are URIs we propose that a profile of a URI be preserved (see Definition 3). Since the data being processed is RDF it is logically divided into triples which can be preserved and monitored individually. It is deemed necessary to store information pertaining to the temporality and provenance of each triple - when it was first seen, last updated, source URI(s) and whether it has changed or been retracted/deleted from the RDF.

Definition 3. A URIs profile is the transformed and annotated form of the linked data retrievable about a given URI such that the temporality and provenance of the triples contained therein are inferable from the profile

In terms of user interface the system should allow a user to view a profile both in the stored form and in its original form. The system must monitor the original data source over time updating the profiles as necessary such that it can provide a report of changes in the data to the user. Since a URI profile will contain versioning information the interface should allow a user to view a particular version of the profile.

3.2.1 Schema

As the first stage of implementation an RDF Schema for All About That[12] (AAT) is defined which embodies classes and properties which allow the description and annotation of triples in such a way that the required information as discussed in the preceding proposal can be stored for each

[6] `http://www.uriburner`
[7] `http://dbpedia.org`
[8] `http://www.sindice.com`
[9] `http://www.sindice.com/developers/cacheapi`
[10] `http://www.sameas.org`
[11] `http://www.dotnetrdf.org/expander/defaultProfile`
[12] This schema is available at `http://www.dotnetrdf.org/AllAboutThat/`

triple. The schema defines a class for representing profiles called `aat:Profile` and uses the `rdf:Statement` class to represent triples. `rdf:Statement` is used as the basis of triple storage as it makes it possible for non-AAT aware tools to extract the original triples from the profile easily. A number of properties are defined which store meta data about the profile itself such as created & updated date, source URI and a locally unique identifier for the profile. Similar properties are defined for triples which allow the first and last asserted dates, source URI and change status of a triple to be indicated. A key distinction in the schema is between `aat:profileSource` and `aat:source`, despite storing equivalent data two predicates are created since the former expresses the URI which is the starting point for the profile while the latter expresses all the URIs at which a given triple is asserted.

While there were alternative schemas and vocabularies available that could have potentially been used to store the required data the motivation behind designing our own schema was to provide a lightweight schema that attached all data to a single subject for ease of processing. Alternatives such as the Provenance Vocabulary by Hartig & Zhao [13] are far more expressive but they potentially require introducing multiple intermediate blank nodes which would significantly complicate the processing needed to implement many of the core features of AAT. Similarly the Open Provenance Model as described by Moreau et al [18] is highly expressive but like the Hartig & Zhao's vocabulary the RDF serialization is overly complex for use in AAT. As discussed in Section 5 there is no reason why the data contained in AAT could not be exposed in other provenance vocabularies but for AATs processing and storage a lightweight vocabulary is preferable.

The use of reification was chosen over the use of named graphs primarily due to the need to make annotations at the level of individual triples rather than at the graph level, usage is motivated by the fact that the mechanism provides a clear and obvious schema for encoding a triple and adding additional annotations to it. While reification may significantly increase the size of the data being stored initially over time this balances out compared to named graphs where it is necessary to either store many copies of the same graph or store multiple named graphs which represent a series of deltas to the original data. The other difficulty inherent in the named graphs approach is that the annotations typically would then be held separately in other named graphs which adds to the complexity of the data processing. Nevertheless named graphs are used within AAT since each profile naturally forms a named graph and AAT generates several related named graphs about each profile detailing change history and changesets as described in Section 3.2.3.

3.2.2 Profile Creation & Update

To create a URIs profile linked data about the URI is first retrieved using the expansion algorithm presented in Section 3.1; then using the AAT schema each triple can be transformed into a set of triples which represent an annotation of the original triple. For each triple in the original RDF a blank node is created which is then used as the subject of a set of triples which represent the required information about the original triple. Figure 1 shows an example triple and Figure 2 shows it transformed into the AAT form. A URIs profile consists of a set of transformed triples where each profile is a named graph in the underlying store.

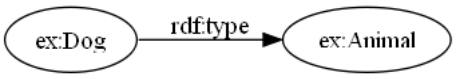

Figure 1: Original Triple

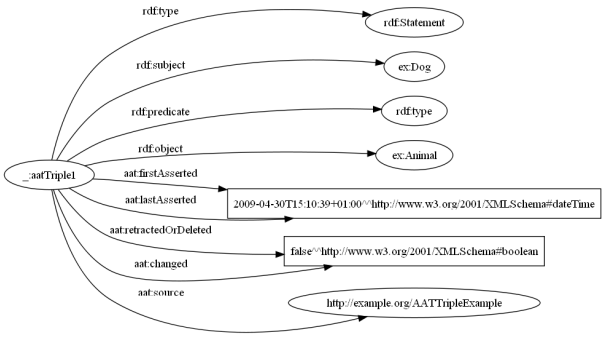

Figure 2: Triple transformed to AAT Annotated Form

Since the user needs to both browse the data they are preserving as well as potentially republish it, a Web based interface was designed as the primary interaction mechanism. The interface allows users to explore the data by first selecting a profile to view and then allowing them to view profile contents, export, versions and change reports. A user may also use the interface to add new URIs they wish to monitor to the system and to initiate updates to profiles (see Definition 4). Following linked data best practices [6] and to provide the ability for the user to republish their preserved data multiple dereferenceable URIs for each profile are created and accessible through the Web interface. These allow the retrieval of the profile contents which consists of all the triples ever retrieved from the profile URI in the transformed form, the export of the profile (see Definition 5) and various meta graphs about a profile e.g. change history, changesets. This means that the profile of a URI has a URI and thus can itself be profiled if it was desired.

Definition 4. An update of a profile occurs when AAT using the Expansion algorithm to retrieve RDF about the given URI. The triples contained are compared with the triples currently in the profile and the profile updated accordingly

Definition 5. The export of a profile is the recreation of the RDF in its original form based upon the current contents of the profile. An export represents the RDF as it was last seen by AAT

3.2.3 Change Reporting

A key feature of AAT is the ability to generate change reports about how the RDF at the profiled URI has changed over time. To do this a number of relatively simple computations over the annotated triples can be made based primarily on the first and last asserted dates of the triples. In creating change reports four different types of changes in the RDF are looked for (see Definitions 6-9). A distinction is made between missing knowledge and retracted or deleted knowledge as it may be possible for triples to be perceived to be temporarily non-present in the RDF. For example in the event of a transient network issue making some/all of the

relevant URIs unretrievable the updated date for the profile will still be updated leaving all the triples in the profile to appear missing. The length of time we require Triples to be missing before we consider them to be deleted is currently set to 7 days for our monitoring of the BBC dataset described in Section 4.2.1, this time period is a domain specific parameter that can be adjusted depending on the data that is being monitored.

Definition 6. New knowledge is any triple that is new to the RDF at the profiled URI

Definition 7. Changed knowledge is any triple where the object of the triple has changed. Only triples where the predicate has a cardinality of 1 can be considered to change

Definition 8. Missing knowledge is any triple no longer found in the RDF at the profiled URI but which was recently seen in the RDF

Definition 9. Retracted or deleted knowledge is any triple no longer found in the RDF at the profiled URI which has not been seen for a reasonable length of time

In regards to the concept of changed knowledge consider some arbitrary predicates `ex:one` and `ex:many` which have cardinalities of 1 and unrestricted respectively. Since `ex:one` has a cardinality of 1 it can be said whenever the object of that triple has changed it is changed knowledge. Yet it cannot be said for `ex:many` triples as the predicate has unrestricted cardinality, therefore each triple using this predicate must be treated as a unique entity i.e. one instance of a triple using this predicate cannot be considered to replace another. In the examples the fact that $< A >$ was related to $< C >$ via the predicate `ex:many` in Example 1 and now is instead related to $< E >$ in Example 2 doesn't mean they are related to $< E >$ instead of $< C >$, it just means they no longer consider themselves related to $< C >$. The fact that they are related to $< E >$ is new knowledge while the fact they related to $< C >$ is missing/deleted knowledge, but if the value of the `ex:one` relationship had changed then that would be considered changed knowledge.

Example 1 Original Graph

```
<A> ex:one <B> .
<A> ex:many <C> .
<A> ex:many <D> .
```

Example 2 Modified Graph

```
<A> ex:one <B> .
<A> ex:many <D> .
<A> ex:many <E> .
```

When a change report is computed is it itself serialized into an RDF Graph using the Talis Changeset ontology [1] which is stored as a named graph in the underlying store and republished via the web interface. Each Changeset generated links back to the previous Changeset (if one exists) such that a end user/client application consuming the data can follow the history of changes, a special URI which retrieves the most recent Changeset is provided such that users have a starting point for this. Separate to Changesets a named graph containing a history for each profile is also stored which links to all the relevant Changesets for a profile.

4. RESULTS

4.1 Expansion

To test the expansion algorithm we took a small sample or URIs which included the URIs of the authors, places associated with the authors and TV programmes from the BBC (since we use the BBC programmes dataset for our preservation tests as described in Section 4.2). The results shown in Table 1 show that the amount of linked data that can be obtained using the default expansion profile described in Section 3.1.1 varies depending on the URI being profiled. Expanding the URI of a person potentially produces a large number of small graphs particularly if that person is a well published academic since many bibliographic databases are exposed as linked data and provide small amounts of data about people. As can be seen URIs for places return varying amounts of data which depends on the size and relative importance of the place. Conversely expanding the URIs of BBC programmes using the default profile produces very little linked data, we suspect that this is due to the type of data and the fact the linking it uses it mostly based on the BBCs ontologies. As outlined in Section 5 we plan to conduct experiments in the future to asses the efficacy of the algorithm on various types of data and using domain specific expansion profiles.

One of the benefits of the algorithm is that as can be seen in the results in Table 1 the algorithm is trivially parallel. Increasing the number of threads used to process the discovered URIs shows a significant reduction in the time taken to retrieve the linked data. Experiments were conducted with higher number of threads but 8 threads was found to be optimal since beyond 8 threads erratic behaviour is observed due to two factors: 1. underlying limitations of the HTTP API used in terms of stable concurrent connections and 2. high volumes of concurrent access to a single site look like DoS attacks and lead to temporary bans on accessing those sites. Differences in the number of triples and graphs returned for URIs can be attributed to a couple of factors. In the case of the London URI where the difference is dramatic - over 200,000 triples difference - this is because with a smaller number of threads connections seem more likely to time out though we are unsure why this is. In the other cases many of the graphs were from the same domain name and the API used to retrieve the RDF had a bug regarding connection management for multiple concurrent connections to the same domain which caused connections to fail unexpectedly which is why a reduction in the amount of data is observed as the number of threads increased.

4.2 Preservation

4.2.1 BBC Programmes

In order to test AAT properly it was used to monitor a subset of the BBC Programmes[13] dataset which is a large and constantly changing linked data set which allowed for both the testing of the scalability of AAT and for the verification that it's change detection algorithms worked as

[13] `http://www.bbc.co.uk/programmes/developers`

Table 1: Sample Expansion Algorithm Results

URI	Total Graphs	Total Triples	Retrieval Time (seconds)	Thread Used
Rob Vesse	4	115	13.8	1
http://id.ecs.soton.ac.uk/person/11471	4	115	1.8	2
	4	115	1.8	4
	4	115	2.1	8
Wendy Hall	691	4,068	786.3	1
http://id.ecs.soton.ac.uk/person/1650	692	4,070	383.8	2
	692	4,070	375.9	4
	692	4,070	359.6	8
Les Carr	368	2,694	438.9	1
http://id.ecs.soton.ac.uk/person/60	279	2,516	109.5	2
	238	2,434	75.9	4
	204	2,366	64.8	8
Ilkeston	6	444	19.1	1
http://dbpedia.org/resource/Ilkeston	5	393	13.5	2
	5	416	9.6	4
	5	393	5.3	8
Southampton	24	3,735	57.2	1
http://dbpedia.org/resource/Southampton	23	3,497	43.8	2
	23	3,497	27.3	4
	23	3,497	55.3	8
Nottingham	17	4,154	41.4	1
http://dbpedia.org/resource/Nottingham	16	4,048	39.5	2
	16	4,048	27.4	4
	16	4,048	25.9	8
London	13	53,886	142.4	1
http://dbpedia.org/resource/	13	53,870	211.9	2
	13	53,870	149.8	4
	14	280,424	385.8	8
Eastenders	2	612	1.8	1
http://www.bbc.co.uk/programmes/b006m86d	2	612	0.7	2
	2	612	0.6	4
	2	612	0.7	8
Panorama	2	174	1.4	1
http://www.bbc.co.uk/programmes/b006t14n	2	174	0.9	2
	2	174	0.7	4
	2	174	0.6	8

Table 2: BBC Programmes preserved dataset size over 1 week

Date and Time	Number of Changed Profiles	Average Changes per Profile	Max. Changes	Min. Changes
12/2/2010	163	43	311	1
13/2/2010	105	2	25	1
14/2/2010	90	2	25	1
15/2/2010	87	2	25	1
16/2/2010	90	2	25	1
17/2/2010	87	2	25	1
18/2/2010	86	2	25	1

Figure 3: BBC Programmes Demonstration Application built on top of data from AAT

intended. The subset used was all the brands (i.e. programmes) associated with the service BBC1 (the BBCs main TV channel) since this includes many brands which change regularly such as soaps and news broadcasts. Table 2 demonstrates the average number of changes detected over just a short period.

As can be seen in Table 2 you can see that the BBC update their dataset on a daily basis, the initial high number of changes is due to starting from a base dataset that was a couple of months old due to architectural changes made to AAT to support the use of the expansion algorithm and improve the efficiency of the system. The average number of changes being 2 is due to the fact that the typical update we see the BBC make to their data is that they add a triple describing a newly broadcast episode of a programme and update the value of the `dc:modified` triple. The apparently high number of 25 for the maximum changes is due to one of the program URIs failing to resolve resulting in the contents of that profile being considered to missing so the change report for each day reports those triples as removed. The relatively high number of profiles changing each day is due to the fact that as already stated many of the programmes associated with BBC 1 are broadcast daily such as soaps and news bulletins and that the BBC publish data about programmes several days before the programmes are actually broadcast.

To demonstrate the reuse of the data being harvested we created a demonstration application which is a simple web based faceted browser which lets users browse through information about recently shown BBC shows. Facets can be used to filter by Genre and Channel and the user can view detailed information about both programmes and the individual episodes. This application was presented as part of an earlier prototype of AAT described in [26] and shown in Figure 3. Like previous work by Papavassiliou et al [20] it shows that simple information about basic triple level changes in RDF (additions, deletions etc) can be reprocessed into useful applications for end users.

4.2.2 *Architecture & Scalability*

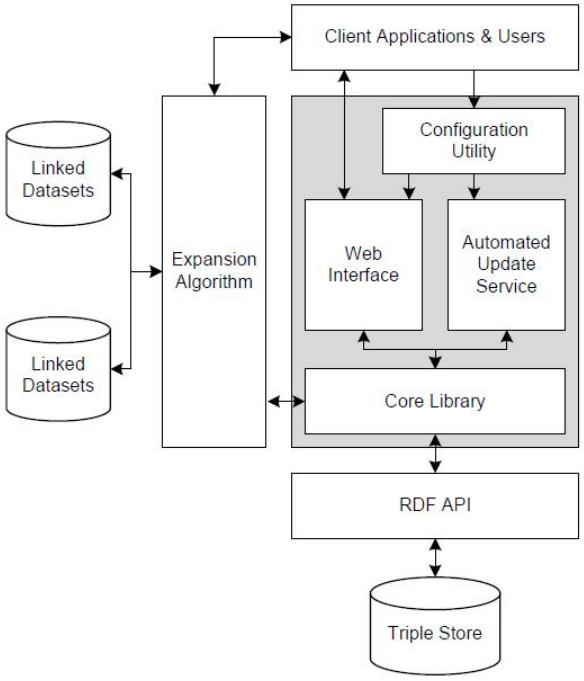

Figure 4: All About That Architecture

AATs architecture is constructed as shown in Figure 4, and as can be seen it is decomposed into several components which then rely on some external standalone components: an RDF API and the expansion algorithm. AAT is theoretically agnostic of its underlying storage though in practise differences in implementation between triple stores mean only certain stores are currently viable for use as the backing store. In the early prototyping stage a RDBMS based store was used which was sufficient for initial prototyping but not scalable for real world testing so then the usage of production grade triple stores was adopted. Initially it was intended to use the open source release of Virtuoso[14] as the backing store but it was found that Virtuoso didn't correctly preserve boolean typed literals which created issues in the internal processing of data within AAT. 4store[15] was then used briefly but it was found that it was unable to handle the heavy volume of parallel read/writes which AAT uses during its data processing due to 4store's concurrency model. Currently AAT runs again AllegroGraph[16] since it has demonstrated in testing the ability to handle the high volumes of read/writes necessary for using AAT on the large dataset described in the preceding section.

In terms of general scalability the majority of algorithms in AAT need to run on a single thread for each profile but it is trivial to process multiple profiles in parallel and this is the approach taken currently. Since work can be divided over multiple threads it will also be possible to significantly increase the scalability by dividing the work over a cluster of machines which would allow much larger datasets to be monitored efficiently.

[14] `http://www.openlinksw.com/virtuoso`
[15] `http://4store.org`
[16] `http://www.franz.com/agraph/allegrograph/`

5. FUTURE WORK

There are a number of things that could be done to improve the expansion algorithm outlined in Section 3.1 with regards to both making it more intelligent in how it retrieves linked data and in conducting a detailed analyses of the data returned. Manual inspection of the data shows that it does appear to be relevant to the URI of interest but it is proposed that a full IR analysis of this is conducted in order to statistically confirm this initial assessment. Additionally as was seen in Table 1 some types of URIs produced very little linked data using the default expansion profile, a broader analysis using domain specific profiles is necessary to ascertain whether those URIs have low levels of interlinking or if the interlinkings just use domain specific links rather than the generic `owl:sameAs` and `rdfs:seeAlso` links that are followed by default.

In terms of improving the intelligence of the algorithm at the moment it submits every URI to every SPARQL, lookup and discovery endpoint described in the expansion profile, it would improve the speed of the algorithm if it could use some decision making as to which endpoints a given URI should be submitted. Conversely though there is the possibility that this would impact the effectiveness of the algorithm so it would be necessary to conduct experiments to determine whether there is a trade off between speed and accuracy. It is also worth considering that searching on URIs is not the only viable mechanism for finding additional linked data about a URI of interest. Using terms extracted from the RDF such as the objects of `rdfs:label` or `dc:title` triples would provide a way to augment URI based lookup with term/text based search results from semantic search engines. There are already frameworks like Silk [28] which can be used to do this and it would be useful to integrate the Silk framework with the expansion algorithm.

One limitation inherent in AAT is that currently is does not do any kind of special handling of blank nodes which means that if data contains blank nodes AAT will continuously think it has encountered new knowledge when most likely it has not. For the data we have worked with so far this is generally not an issue since the linked data community tends to avoid blank nodes but if we are to provide for preserving all kinds of RDF effectively then we need to handle blank nodes properly. Solving this problem may involve doing some sub-graph matching and isomorphism to see if the sections of the graph that contain blank nodes can be mapped to the previously seen sections of the graph as in Tummarello et al's RDFSync [23]. The blank nodes themselves could either be left as-is or they could be translated to URIs as done by systems like the Talis[17] platform.

Given that this work was inspired by traditional link integrity techniques from hypermedia it is interesting to note that it has the potential to be applied back to the document web since there is increasing cross-over between the document and data web primarily due to the increasing uptake of RDFa. As increasing numbers of documents embed structured data using RDFa it will become possible to preserve and monitor the structured information embedded in ordinary web pages in the same way as can be done with linked data now, therefore we envisage this as having applications in automated monitoring and maintenance of document based websites.

As mentioned in Section 3.2.1 a lightweight schema is used by AAT to annotate and store the data but there are alternative vocabularies that could have been used such as the provenance ontology [13] and the open provenance model [18]. It would be a fairly easy and potentially useful enhancement to map the AAT schema to these vocabularies such that the data could be retrieved in the desired form by users/client applications designed to work with those formats.

[17] `http://www.talis.com/platform`

6. CONCLUSION

In this work we have introduced a simple but powerful expansion algorithm which can be used to retrieve linked data about a URI even when that URI is not resolvable. This provides an important tool for preserving data in the Semantic Web and recovering from data loss and shows that in the Semantic Web links themselves can be exploited as a means to recover from broken links. As we have outlined in Sections 4.1 and 5 there is a need to conduct a detailed analysis of the algorithm to asses it's efficacy for a wider variety of URIs and using domain specific expansion profiles. Depending on the results of this analysis the algorithm may need to be further refined to improve both it's speed and accuracy.

We have also presented the All About That (AAT) system which allows users to monitor and preserve linked data they are interested in using the expansion algorithm as the primary retrieval method for deciding which linked data to preserve based on a starting URI. As we demonstrated in Section 4.2.1 we envisage the usage of such a system as a base on which to build rich Semantic Web applications that can take and present the changing data in interesting and useful ways to end users. It also fulfils a role in the overall goal of our research which is to provide a suite of algorithms and systems which can be used to manage both data and link integrity on the Semantic Web.

As has been discussed in Section 5 there are some limitations in the current versions of our algorithm and the AAT system which we intend to investigate and address in the future. It is clear that there is still a significant amount of work to be done to create a comprehensive set of tools such that they can be applied to as wide a variety of data on the Semantic Web as is possible and experiences of past research in link integrity for the document Web tells us that there will be no perfect solution.

Despite this it is our belief that as the Semantic Web grows data and link integrity will be increasingly important issues to users as their applications come to rely upon linked data. There is a need to have systems in place such that data can be preserved and accessed even if the original sources are gone or unavailable. This has already been seen with the release of services like the Sindice Cache API[18] which is used as one of the data sources in the default expansion profile (see Section 3.1.1). Additionally with rising adoption of RDFa embedded inside documents on the web systems like this become applicable for the preservation of the structured data embedded in the document based Web as discussed in Section 5.

[18] `http://www.sindice.com/developers/cacheapi`

7. REFERENCES

[1] Changeset protocol, 2007. `http://n2.talis.comn/wiki/Changeset_Protocol`.

[2] Virtuoso sponger. Technical report, OpenLink Software, 2009. http://virtuoso.openlinksw.com/Whitepapers/html/VirtSpongerWhitePaper.html.

[3] K. Alexander, R. Cyganiak, M. Hausenblas, and J. Zhao. Describing linked datasets: On the design and usage of void, the 'vocabularly of interlinked datasets'. In *Proceedings of the Linked Data on the Web Workshop (LDOW2009), Madrid, Spain*, April 2009. http://ceur-ws.org/Vol-538/ldow2009_paper20.pdf.

[4] H. Ashman. Electronic document addressing: dealing with change. *ACM Comput. Surv.*, 32(3):201–212, 2000.

[5] T. Berners-Lee, R. Cailliau, A. Luotonen, H. F. Nielsen, and A. Secret. The world-wide web. *Commun. ACM*, 37(8):76–82, 1994.

[6] C. Bizer, R. Cyganiak, and T. Heath. How to publish linked data on the web, 2007. http://sites.wiwiss.fu-berlin.de/suhl/bizer/pub/LinkedDataTutorial.

[7] P. Bouquet, H. Stoermer, and B. Bazzanella. An entity name system (ens) for the semantic web. In *5th European Semantic Web Conference, ESWC 2008*, volume 5021, page 258. Springer, 2008.

[8] G. Cheng, W. Ge, and Y. Qu. Falcons: searching and browsing entities on the semantic web. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 1101–1102, New York, NY, USA, 2008. ACM.

[9] H. Davis. *Data Integrity Problems in an Open Hypermedia Link Service*. PhD thesis, University of Southampton, November 1995. http://eprints.ecs.soton.ac.uk/6597/.

[10] H. C. Davis. Hypertext link integrity. *ACM Comput. Surv.*, page 28, 1999.

[11] A. M. Fountain, W. Hall, I. Heath, and H. C. Davis. Microcosm: an open model for hypermedia with dynamic linking. In *Hypertext: concepts, systems and applications*, pages 298–311, New York, NY, USA, 1992. Cambridge University Press.

[12] T. L. Harrison and M. L. Nelson. Just-in-time recovery of missing web pages. In *HYPERTEXT '06: Proceedings of the seventeenth conference on Hypertext and hypermedia*, pages 145–156, New York, NY, USA, 2006. ACM.

[13] O. Hartif and J. Zhao. Guide to the provenance vocabularly, 2009. http://sourceforge.net/apps/mediawiki/trdf/index.php?title=Provenance_Vocabulary.

[14] B. Haslhofer and N. Popitsch. DSNotify–Detecting and Fixing Broken Links in Linked Data Sets. In *Proceedings of 8th International Workshop on Web Semantics*, 2009.

[15] A. Jaffri, H. Glaser, and I. Millard. Managing uri synonymity to enable consistent reference on the semantic web. In *IRSW2008 - Identity and Reference on the Semantic Web 2008*, 2008.

[16] F. Kappe. A scalable architecture for maintaining referential integrity in distributed information systems. *Journal of Universal Computer Science*, 1(2):84–104, 1995. http://www.jucs.org/jucs_1_2/a_scalable_architecture_for.

[17] F. Kappe, K. Andrews, J. Faschingbauer, M. Gaisbauer, M. Pichler, and J. Schipflinger. *Hyper-G: A new tool for distributed hypermedia*. Institutes for Information Processing Graz, 1994.

[18] L. Moreau, J. Freire, J. Futrelle, R. McGrath, J. Myers, and P. Paulson. The open provenance model. December 2007. http://eprints.ecs.soton.ac.uk/14979/.

[19] L. Moreau and N. Gray. A Community of Agents Maintaining Links in the World Wide Web (Preliminary Report). In *The Third International Conference and Exhibition on The Practical Application of Intelligent Agents and Multi-Agents*, pages 221–235, London, UK, Mar. 1998. http://www.ecs.soton.ac.uk/~lavm/papers/gcWWW.ps.gz.

[20] V. Papavassiliou, G. Flouris, I. Fundulaki, D. Kotzinos, and V. Christophides. On Detecting High-Level Changes in RDF/S KBs. In *The Semantic Web: 9th International Semantic Web Conference (ISWC2009)*, pages 473–488. Springer, 2009.

[21] T. A. Phelps and R. Wilensky. Robust hyperlinks: Cheap, everywhere, now. In *Digital Documents: Systems and Principles*, pages 514–549. Springer, 2004.

[22] G. Tummarello, R. Delbru, and E. Oren. Sindice. com: Weaving the Open Linked Data. In *The Semantic Web: 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC*, pages 552–565. Springer, 2008.

[23] G. Tummarello, C. Morbidoni, R. Bachmann-Gm "ur, and O. Erling. RDFSync: efficient remote synchronization of RDF models. In *The Semantic Web: 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007+ ASWC 2007, Busan, Korea, November 11-15, 2007, Proceedings*, pages 537–551. Springer, 2007.

[24] L. Veiga and P. Ferreira. Repweb: replicated web with referential integrity. In *SAC '03: Proceedings of the 2003 ACM symposium on Applied computing*, pages 1206–1211, New York, NY, USA, 2003. ACM.

[25] L. Veiga and P. Ferreira. Turning the web into an effective knowledge repository. *ICEIS 2004: Software Agents and Internet Computing*, 14(17), 2004.

[26] R. Vesse, W. Hall, and L. Carr. All about that - a uri profiling tool for monitoring and preserving linked data. In *ISWC 2009*, August 2009. http://eprints.ecs.soton.ac.uk/17815.

[27] J. Volz, C. Bizer, M. Gaedke, and G. Kobilarov. Discovering and maintaining links on the web of data. In A. Bernstein, D. R. Karger, T. Heath, L. Feigenbaum, D. Maynard, E. Motta, and K. Thirunarayan, editors, *The Semantic Web - ISWC 2009*, volume 5823 of *Lecture Notes in Computer Science*, pages 650–665. Springer, 2009.

[28] J. Volz, C. Bizer, M. Gaedke, and G. Kobilarov. Silk–a link discovery framework for the web of data. In *2nd Linked Data on the Web Workshop (LDOW2009)*, 2009.