Putting Hybrid Cultural Data on the Semantic Web

Kate Byrne
School of Informatics, University of Edinburgh
k.byrne@ed.ac.uk

Abstract

A prerequisite for joining the rapidly growing Semantic Web is to expose data as RDF triples. In the cultural heritage world the data in question is very often a mixture of structured database fields and associated textual documents. Transforming relational database (RDB) content to RDF is not altogether straightforward and the issues are examined as a preliminary to the much more difficult step of augmenting the RDB content by extracting structured RDF triples directly from natural language text, using a specially designed *txt2rdf* process. This opens the way to a true integration of the hybrid data so common in heritage management. Finally we lead up to experimental results showing structured queries (using SPARQL) that cannot be answered from the RDB material alone, but which are satisfied against the augmented graph. In this domain there are potentially vast amounts of textual material available for linking to structured records, so the future possibilities of the techniques described are exciting.

Key words: Semantic Web, cultural heritage, NLP, RDF, txt2rdf, hybrid data, semantic querying

1 Introduction

Cultural archive managers are becoming increasingly interested in the potential of the Semantic Web [Berners-Lee et al., 2001, Feigenbaum et al., 2007] to both open up and interconnect their rich data repositories. The Semantic Web, or Web of Data, is steadily growing as more and more information is exposed in the necessary format. Notable examples from the cultural world include CultureSampo [Hyvönen et al., 2007] in Finland, the MultimediaN E-Culture demonstrator [Schreiber et al., 2006], and the experimental "semantic search" prototype[1] within the Europeana project.

The basic principle of the Semantic Web is that if all of its data is held in the same, simple, standardised format – as RDF triples forming a directed

[1] http://eculture.cs.vu.nl/europeana/session/search

graph[2] – then interconnecting distributed datasets becomes dramatically easier. A Semantic Web "resource" is a node or arc in the graph, and can be thought of as the smallest unit of data in the system. Each resource is identified by a URI (such as a web address, beginning "http://..."). If my data shares at least one resource with your data, and both are open to the Web, our datasets are immediately linked and a query can be run over their combination. No further steps are needed. There is huge potential for building connections between archive collections and making them accessible to queries from the Web.

The catch, such as it is, is that transforming data into RDF is not as straightforward as one might hope[3]. In the cultural field, as in other domains such as business and finance, structured data is still typically managed in relational database (RDB) format. The process of converting a typical heritage database to RDF – the RDB2RDF step as it is usually called – is looked at in Section 2 below. The data used comes from the National Monument Record of Scotland (NMRS) maintained by RCAHMS (The Royal Commission on the Ancient and Historical Monuments of Scotland[4]). Because scalability was a central goal, the entire database was used: around 270,000 site records with about 1 million associated archive items.

Despite its historical popularity, the relational database is by no means ideal for managing cultural data. As well as fixed fields that can be mapped straightforwardly into a relational design, cultural data abounds with notes, free text documents, associated archive items and so on, which are awkward to represent in RDB tables. Web-based interfaces for public querying usually have to be restricted to just a small subset of the actual RDB schema, and the rest of the data, typically including all the free text, is inaccessible to queries (though of course it can be included in the results). By contrast with the Semantic Web connection principles just explained, linking separately curated RDB data – museum finds, site-based archives, library catalogues and so forth – is technically difficult.

An RDF triple is designed to state a single fact or "relation", such as "*Skara_Brae–hasLocation–Orkney*", and the RDB2RDF process translates data from database tables into lists of such statements. In principle there is nothing to prevent the factual content of *any* data being expressed in exactly the same way. A key objective in my work is to extract "fact" triples automatically from natural language text, by means of an NLP (Natural Language Processing) "pipeline" of processes I call *txt2rdf*. An overview of the procedure is given in Section 3. Combining this with the RDB conversion process gives us the potential of a truly integrated dataset for semantic searching.

In the course of this work a large RDF dataset was built, containing around

[2]RDF, Resource Description Framework, is a W3C Recommendation as specified in Klyne and Carroll [2004]. An RDF graph is a network of interconnecting triples of the form *subjectNode–propertyArc–objectNode* (often called SPO triples). Triples become connected whenever the object of one is the subject of another.

[3]The transformation may be actual (the data is rewritten in RDF format) or virtual (the data is not changed but is made to appear as if in RDF). The principles in each case are identical and, for the purposes of this paper, the differences are immaterial.

[4]`http://www.rcahms.gov.uk/`

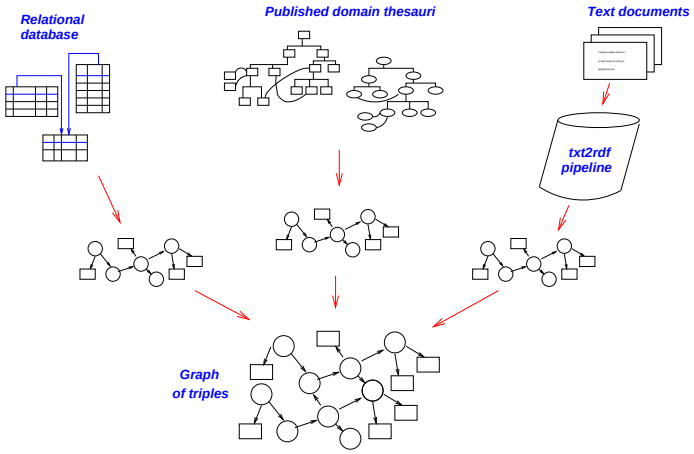

Figure 1: An overview of the *tether* system. See Fig. 4 for the txt2rdf pipeline.

22 million triples derived from the NMRS data and associated text documents. So far only a comparatively small amount of free text has been processed but there is enough to test the proposition that augmenting the basic RDB data with statements derived from text will enable new queries to be answered. Section 4 describes some experiments on this theme, using the SPARQL[5] query language (the W3C recommended standard for querying RDF).

Although the work described is at the prototype stage so far, there is clear potential for combining hybrid data using the techniques described. Section 5 discusses the implications for developing more powerful Web-based query mechanisms, that will further expand public access to our cultural heritage worldwide.

2 Converting Structured Data to RDF

The experiments described in this paper were part of a larger programme of work on populating the Semantic Web by combining RDB data with text relations and with published domain ontologies [Byrne, 2009]. Figure 1 shows the overall layout of the system, which was named *Tether*. This section concentrates on the RDB2RDF process, shown on the left side of the figure, and on RDF shema design considerations.

Automatic conversion software is available for translating RDB data into RDF but there are drawbacks to the basic procedure. These are discussed in Byrne [2008] and some of the key points are summarised here. Schema design for RDF graphs is not the same as for relational databases and it is worth inserting a manual design step to reap benefits later. In order to produce a compact and flexible structure one must design a schema intended for RDF, rather than attempt to translate a relational data model unchanged. For cultural

[5] `http://www.w3.org/TR/rdf-sparql-query/`

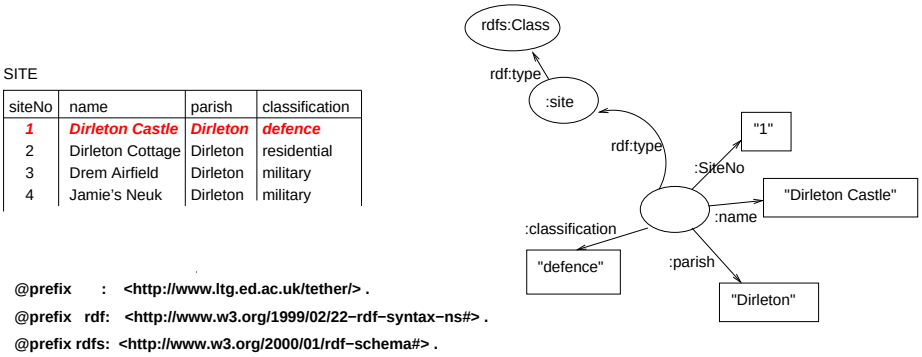

Figure 2: Translation of a relational table tuple to RDF, at its simplest.

archives choosing to publish their material on the Semantic Web, there is a real opportunity to convert to a schema that will promote integration with other similar collections. A simple cultural data schema is proposed in Byrne [2008], based around the "Who? What? Where? When?" or "People, Places, Things and Events" models often used in heritage data management.

2.1 The RDB2RDF Process

The basic RDB2RDF conversion process is straightforward: take each RDB table in turn and "walk" across each of its rows, column by column. At each cell thus encountered we form a new RDF triple, whose property arc comes from the RDB column name, and whose object node contains the cell value. Figure 2 illustrates the procedure using a fragment based on the RCAHMS database `SITE` table.

Each database tuple (or table row) becomes a cluster of triples grouped around a node whose type corresponds to the table name. Hence this conversion is sometimes called "Table to Class" transformation. There are now quite a number of automatic conversion tools that work on these general principles[6] but as yet there are no definitive guidelines on best practice in RDB2RDF conversion. The clearest exposition of the basic principles is still Berners-Lee [2006], a design note based on one originally written in 1998.[7] The W3C set up an Incubator group which reported in early 2009, with a "state of the art" paper [Sahoo et al., 2009] and a recommendation (now implemented) that a Working Group be formed to standardise a mapping language. The proper way

[6] Examples include D2RQ [Bizer and Cyganiak, 2007], Dartgrid [Wu et al., 2006], Dan Connolly's `dbview` program (`http://dig.csail.mit.edu/2006/dbview/dbview.py`), R_2O [Barrasa et al., 2004] and DB2OWL [Cullot et al., 2007]. They have differing degrees of scope for customisation by the user. SquirrelRDF and R2D2 (a sister project of D2RQ) construct a virtual graph "view" of a relational database, and D2RQ and Virtuoso [Virtuoso, 2006] give the user the choice of an instantiated graph or a virtual one.

[7] The version referred to here dates from 2006.

to generate the URI labels has produced a mountain of literature in its own right, and there is plenty of guidance available (see Sauermann and Cyganiak [2007], Miles et al. [2008]) on how to create and serve what Berners-Lee dubbed "cool" URIs.

2.2 Design Issues

The mechanical procedure just described is beguiling in its simplicity but a number of drawbacks become clear when one uses it in practice. See Byrne [2009, Chap. 5] for a full discussion but let us look briefly at the main points. If we translate RDB cell values to literals (as in Fig. 2, where they are strings) then we prevent them forming onwards connections, because a literal cannot be the subject of an RDF triple. Using a bnode (or "blank node", a special RDF node without a label, as illustrated in the figure) for something as vital as the central node of the cluster will make SPARQL queries unnecessarily cumbersome – much better to move the RDB primary key into the central spot, or generate a new surrogate if the RDB doesn't provide something suitable (for instance if the primary key is a concatenation of columns).

A perhaps more contentious point concerns the "column as predicate" translation that is part of the standard RDB2RDF process, where each column name from the relational database becomes a predicate or property arc in the graph. This leads to an enormous number of different predicates in the RDF schema, which in turn will make SPARQL querying complex. I argue that it is better to simplify the set of properties quite radically, reducing them to the minimum needed to express the core relationships of the cultural domain: where a site or object is, what kind of thing it is, who are the agents involved with it and what are the relevant time periods. This is the same kind of argument as motivates the adoption of schema standards like Dublin Core.[8]

Much of the detailed structure can be transferred from the property arcs into the class hierarchy. For example, a simple *hasLocation* property may point at a node for the value "Dirleton". If we need to know whether "Dirleton" is the name of a parish or of a district – and often, we will not – we can examine its node type. (Thus we can do without properties like *hasParish* and *hasDistrict*.) Going a step further and embedding relevant parts of the type hierarchy in the URI[9] may obviate the need even for this.

Figure 3 illustrates these points, using the same small sample of data as Fig. 2. The bnode has been replaced by a URI based on the RDB key, the literals are now resources that can have properties of their own, and the database column names have migrated into the RDFS[10] class hierarchy.

The standard translation procedures (at least at the time of this work) reproduce metadata about the RDB origin which may ultimately be completely irrelevant and, worse, they tend to duplicate this provenance information in the URI labels of the property arcs and the resource nodes. Apart from being

[8] `http://dublincore.org/`

[9] This is in accordance with REST principles, defined in Fielding [2000].

[10] RDFS is the RDF Schema language, see `http://www.w3.org/TR/rdf-schema/`.

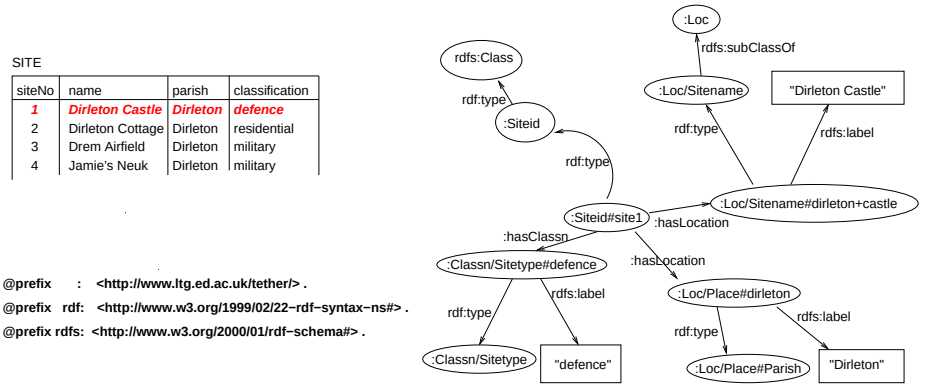

Figure 3: Part of the *ether* design.

more cumbersome than necessary, this approach carries the risk of obscuring connections latent in the data, and creating multiple nodes that could in fact be merged because they represent the same thing. For example, if the resource URI for the Dirleton parish value includes metadata about the originating RDB row, then every row will have a separate "Dirleton" node, which then has to be linked to the canonical resource. There seems no possible advantage in this.

2.3 A Minimalist Schema

The conversion of an archive to RDF for publication through the Semantic Web is an opportunity to standardise the design and hence maximise interoperability with other cultural datasets. Using the RCAHMS dataset as a test-bed, I found that a very simple schema design expressed the core relationships adequately and made querying straightforward. (As a cross-check, a sample of records from the archaeology collections of the National Museum of Scotland was translated into the schema designed around RCAHMS data, and found to fit sufficiently well for meaningful querying across the combined data.) For full management of administrative details, the relational database is probably still the best option, but the point of the Semantic Web is to facilitate interoperability, and simplicity of design is surely the key.

The upper ontology used in *tether* has only 15 classes, with some 45 further classes below that are specific to the RCAHMS collection. The set of predicates is even smaller, as explained above. One reason was to make it easy to ask the most common questions, about "Who? What? Where? When?", at the cost if necessary of blurring details. As is discussed in Section 4, even with such a simple framework there are times when detailed schema knowledge is needed. The ideal would be that software agents could generate sensible queries automatically, based on a user's expressed intention (for example by parsing a request in natural language), without the user requiring *any* schema knowledge

beyond the "common sense" world knowledge that archaeological sites have locations and so on.

The other reason for keeping the schema as small as possible was to enable the RDB data to be integrated with that derived from text relations, as explained in the next section. Extracting text relations using machine learning methods (as here) boils down to a categorisation problem – and the more categories one has, the harder it is. It may well be that in the future, with virtually limitless training data available, automatic methods can subdivide information with a subtlety matching that of the average database analyst. But for now the target data types need to be genuinely distinct from one another, and that inevitably means collapsing the fine distinctions between, for example, districts and parishes.

3 Adding Text Relations – *txt2rdf*

The *txt2rdf* pipeline is a sequence of NLP procedures in which the results of each step are passed into the next step. This pipeline starts with plain text documents as input and the final output is a graph of RDF triples. Since each document is associated with an NMRS site record the text triples can be very easily integrated with the RDB-derived triples, by tying each extracted text relation to a unique site ID from the database. There will be other resource nodes that are common to both the RDB and textual data, such as placenames. The same pipeline can process documents from any source, but without the site ID links one is relying on such shared nodes for linkage.

As illustrated in Figure 4, the pipeline starts with some standard NLP preprocessing steps: splitting the text into "tokens" (which correspond approximately with words), finding the sentence and paragraph breaks, and then doing shallow parsing to annotate each token with a tag indicating its part of speech (POS). Once these basic preparatory steps have been done the key procedures can be carried out: Named Entity Recognition (NER) followed by Relation Extraction (RE). The final operation is to transform the relations into RDF triples and anchor them to individual sites from the RDB data.

3.1 Measured NER and RE Performance

The NER step uses a machine learning approach to detect and categorise NEs (named entities). A model of the features that characterise a typical NE is built and then candidate terms from the text are examined, based on their features, and each is tagged as either an NE of a particular class or as a non-entity.[11] The process for finding binary relations between pairs of NEs is similar. Candidate pairs of NEs from a set of test data are classified as being related or not, based on how their characteristics compare with the model built over training data. The model is built using a list of features chosen to characterise whether a

[11] A "candidate term" here is a sequence of up to six adjacent tokens. The details of the procedure are explained in Byrne [2007].

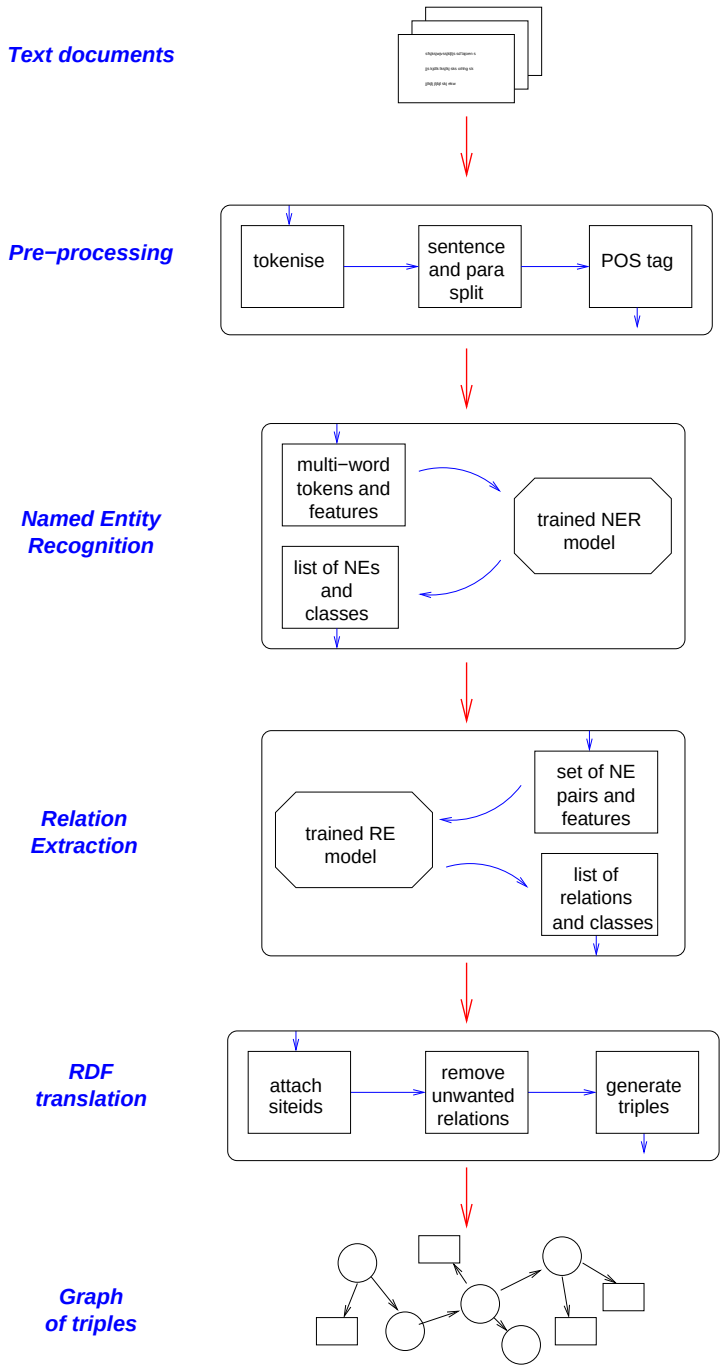

Figure 4: The text to RDF pipeline

relation exists in a particular context; for example, if the two NEs are from the same sentence the main verb of that sentence is an important feature.

Where a relation is detected it is assigned to one of the relation types (again, based on the features of its context), which include *partOf*, *sameAs*, *seeAlso*, *instanceOf*, and a set of event relations: *eventAgent*, *eventAgentRole*, *eventPatient*, *eventDate* and *eventPlace*.[12]

The NER and RE steps were evaluated against a hand-annotated "gold standard", measuring precision and recall and calculating an F-score from them.[13]

Table 1 summarises the results for the NER step, and shows the 11 separate NE types the system was designed to find. Similar kinds of NE classes are grouped together – for example, ADDRESS, PLACE and SITENAME all contain locational information. A less granular system (lumping these three together, say) would be likely to achieve higher scores. Some of the classes (such as ROLE and PERIOD) are very sparsely populated and their results should be treated with caution.

The EVENT class is unorthodox in NER terms because the majority of its instances are verb phrases rather than nouns. At least in the RCAHMS text, events in a site's history are very commonly mentioned in verb phrases, such as "was excavated", "surveyed", "visited" and so on (though of course there are also nominal references, like "excavation by..." and "site visit at..."). Nevertheless, the classifier was gratifyingly successful at modelling the EVENT class, as the results show.

The large SITETYPE and much smaller ARTEFACT classes are of particular importance in this domain. These NE types are for classification terms describing archaeological sites, historic buildings, and object types. It is through these terms that the data can be "grounded" by linking it to published taxonomies for the heritage domain. The principal thesauri used by RCAHMS are for Monument and Object types, and these were converted to RDF using the SKOS framework[14] and following the principles outlined in Section 2. The thesauri are based on MIDAS Heritage [Lee, 2007], the UK historic environment data standard, maintained by English Heritage. In turn, a mapping[15] is available from MIDAS to the CIDOC-CRM [Crofts et al., 2003]. By this means, resources in the *tether* RDF graph can be positioned in the wider Semantic Web of published data. The classification terminology was considered the most important element to concentrate on for this dataset, but similar grounding against published ontologies could be done for other resource classes, such as the GeoNames[16] gazetteer of worldwide placenames.

[12] In a later step, explained in Section 3.2, the detected relations are linked to their parent site – given by the document ID – with an extra relation that is generated in the "RDF translation" step of the pipeline.

[13] Precision is the fraction of all output results that were correct, and recall is the proportion of the entire correct population that was found. The F-score (sometimes "F1 score" to distinguish it from variants) is the harmonic mean of precision (P) and recall (R): $2PR/(P+R)$.

[14] Simple Knowledge Organisation System, `http://www.w3.org/2004/02/skos/`.

[15] See `http://cidoc.ics.forth.gr/docs/MIDAS_mapping_notes.doc` and `http://cidoc.ics.forth.gr/docs/midas_map.xls`.

[16] `http://www.geonames.org/`

	Precision %	**Recall %**	**F-score %**	**Count**
ADDRESS	82.40	81.61	82.00	3,458
PLACE	95.00	66.80	78.44	2,503
SITENAME	64.55	61.20	62.83	2,712
DATE	95.12	82.08	88.12	3,519
PERIOD	84.02	45.54	59.07	400
EVENT	94.98	63.66	76.22	3,176
ORG	99.39	89.66	94.27	2,730
PERSNAME	96.71	74.82	84.37	2,318
ROLE	98.00	54.44	70.00	90
SITETYPE	85.24	52.39	64.89	5,668
ARTEFACT	75.83	18.06	29.17	879
Average	88.02	67.75	76.57	(27,453)

Table 1: Summary of NER results

Table 2 gives the RE results, over known or "gold" NEs. This is a measure of the RE step in isolation, when the classifier is looking at pairs of terms whose NE status or otherwise is known with certainty. An indication of the size of each relation set is given in the table and, as with NE classes, the scores for the smaller sets will be unreliable. The *eventAgentRole* relation set is vanishingly small and should probably be discounted altogether. (The average score would of course be higher if it were discounted.)

In a real situation the RE step has to be run over the results of the NER step, which will mean that some NEs presented to the RE classifier are spurious and some genuine NEs are missing, because of inevitable inaccuracies at the NER stage. Results for the combined steps were calculated over a number of small sets of data and the average scores were found to be much as one would expect by taking the product of the component stages. The average F-score for the entire pipeline is 57.5%, with an average precision score of 73.4%.

Both the NER and RE systems were deliberately designed to favour precision as, for this application, I considered it more important to extract correct statements from the text than to find all the possible ones. New facts can always be added to the Semantic Web structure later, but we need to take all reasonable measures to avoid adding incorrect statements.

There is room for improvement in these results, through better feature design – though the law of diminishing returns notoriously applies to probabilistic modelling. The aim was to build a system capable of handling very large data volumes and there has to be a trade-off in terms of how much feature extraction work is realistic. For example, the option of full linguistic parsing was rejected because of the processing overheads involved. A more promising route to improving overall performance is to add new modules to the pipeline, to deal with term normalisation and co-reference resolution, and to handle negation. This is

Relation	Precision	Recall	F-score	Found
eventAgent	98.42	98.70	98.56	3,794
eventAgentRole	69.23	30.00	41.86	13
eventDate	98.75	98.68	98.71	3,189
eventPatient	87.77	84.61	86.16	1,553
eventPlace	83.58	72.70	77.76	341
hasLocation	83.26	83.00	83.13	5,085
hasPeriod	83.69	73.86	78.47	233
instanceOf	52.00	31.52	39.25	100
partOf	78.87	51.38	62.22	568
sameAs	68.69	44.55	54.05	6,934
seeAlso	50.00	19.68	28.24	122
Average	83.41	69.27	75.68	21,932

Table 2: Summary of RE results

discussed further in Section 4 below.

3.2 Transforming Text Relations to RDF

The formal experimental results have just been shown, but a simple concrete example may now be helpful. Suppose the following text snippet occurs in the text document related to site3402:

> "Bea Mill dates from the 19th century."

If the *txt2rdf* pipeline performs accurately, it will identify "Bea Mill" as a SITENAME and "19th century" as a PERIOD, and find that there is a *hasPeriod* relation between them. (In this example the sentence's verb is a key feature for both detecting and categorising the relation, though *hasPeriod* is a statistically good guess whenever one has found a relation where the object class is PERIOD. It should be noted that this sentence is a very easy example for the model!)

The procedure for mapping this relation to RDF involves generating a second relation, to tie the extracted one to its parent site. We need a connection between the site and the subject of the extracted triple, which is "Bea Mill". As this is a SITENAME, which is essentially a spatial designator, a *hasLocation* predicate is used to connect "site3402" to "Bea Mill". (In general, the NE class of the target is used to determine the appropriate relation label for this added triple.) The next step is to generate suitable URIs to identify each item that is to become an RDF resource. Finally, schema relations are added, to indicate the types of each resource node so that they can find their correct place in the hierarchy of RDFS classes, and to give them human-readable labels.

Figure 5 shows in graphical form the complete set of triples produced from our original text snippet. It should be noted that the type and label relations

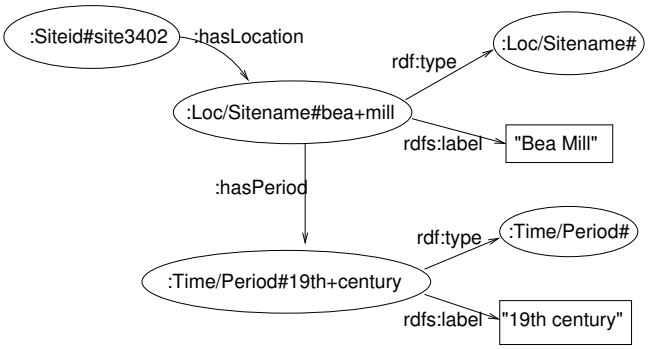

Figure 5: Translation of an extracted text relation to RDF graph form

are only needed once for each unique resource node. Common nodes like "19th century" are typed and labelled just once in the graph. When a new text relation requires this node the relevant schema triples will already be in place and are not added again.

4 Query Experiments

Only a small percentage[17] of the available text documents could be processed through the *txt2rdf* pipeline, because of time constraints. However, around a million triples were added to the RDB-derived graph, and this is enough for meaningful experimentation. Two sets of experiments were carried out, firstly to assess RDF as a viable competitor to standard relational database (RDB) systems in terms of basic query performance, and secondly to examine whether augmenting the database with relations derived from text actually enhances query power.

4.1 Comparing SPARQL with SQL

A series of experiments was run to verify that SPARQL queries over the RDF graph derived from the relational database produce exactly the same results as SQL queries over the original RDB data. From the RCAHMS database, a set of 27 tables were converted to RDF, forming a coherent data set covering about 270,000 site records with around one million archive items and 52,000 bibliographic references. Physically the data was held in Oracle and in MySQL.

The corresponding *tether* graph contained 21 million triples, created as described in Section 2, and held in AllegroGraph[18] (from Franz Inc.) and Jena[19]

[17] Just under 10%, or around 20,000, of the available site text documents were processed.
[18] `http://agraph.franz.com/allegrograph/`
[19] `http://jena.sourceforge.net/`

(from Hewlett Packard). These two systems were chosen from the many available because they are leading examples and each has strengths in different areas. For instance, AllegroGraph is fast, both for data loading and querying, whereas Jena has a more transparent architecture so one can examine exactly how queries are processed.

Clearly one cannot compare SQL and SPARQL over a dataset exhaustively, as there is no limit to the possible queries. But there are standard *kinds* of query that are very commonly run against archive collections of this kind, and these are what the experiments concentrated on. Three very common types of query were tried, shown below with an example of each:

1. **Sitetype by location:** "Show me a list of the churches in Shetland."
2. **Including archive material:** "Summarise the archive material for churches in Shetland."
3. **Restrict by archive type:** "How much of the Shetland archive has been digitally scanned?"

In each case SPARQL and SQL statements were written to extract the data and the results compared. For comparison, here is the SQL for the first query:

```
select m.numlink siteid, m.nmrsname sitename
  from rcmain m inner join rccouncil c on m.council = c.council
     inner join rcclassification cl on m.numlink = cl.numlink
     inner join rc_thesaurus_terms t on cl.the_te_uid = t.the_te_uid
  where c.couname = 'SHETLAND ISLANDS'
  and t.term = 'CHURCH'
  order by m.nmrsname;
```

And this is the equivalent SPARQL query:[20]

```
select distinct ?siteid ?sitename
where {
  ?siteid :hasLocation place:shetland .
  ?siteid :hasClassn sitetype:church .
  ?siteid :hasLocation ?name .
  ?name rdf:type sitename: .
  ?name rdfs:label ?sitename .
}
order by ?sitename
```

It is worth remembering that SQL was originally intended as a "near natural" language. Of the two, the SPARQL seems easier to follow at a glance, though it probably depends on how familiar one is with each. (And there are many ways of writing any given SQL query.)

[20] Throughout this paper I use the standard way of abbreviating RDF URIs, using ":" prefixes. The details of the prefixes are omitted as they would clutter the examples needlessly. A full list of them, and the whole of the *tether* schema, can be found in Byrne [2009, Appendix A].

As expected, the SPARQL and SQL queries produced identical results in each case. Where they differ significantly is in performance time: the SQL queries are typically sub-second, but the SPARQL ones are *much* slower. In AllegroGraph the response times vary between 0.9 seconds and just over 60 seconds (on a 64-bit machine) for repeated runs of a range of typical queries from the three categories above. In Jena typical times were over 7 minutes (though a new version has since appeared, which may be faster).

In the case of the third query one can only frame the query – in either language – if one knows enough detail of the schema to interrogate the prefix of the archive number, which is "SC" for all scanned items. In the *tether* schema this is modelled as *:Desc/Arcdesc#Prefix* – not the kind of thing that could be guessed by a query generator. The dream of a query agent being able to fire SPARQL queries anywhere in the world, without detailed prior knowledge of the data, seems elusive.

4.2 New Queries Made Possible by Text Relations

We now move on to the second set of trials, where the aim was to see if the extension with text relations makes new information available to the user.

An elementary test demonstrates immediately that the text relations provide structured information that cannot be found from the RDB data, for the simple reason that the RDB schema does not include data fields for it. One of the statement types extracted from text shows the organisations that named individuals belong to – this is done by finding ORGANISATION and PERSNAME entities and looking for relationships between them. This type of information is not recorded in the structured RDB fields but may be mentioned as an aside in the text notes. The following SPARQL query produces a list of people and their associated organisations.[21] The "label" patterns are not strictly necessary but they make the results more readable for humans.[22]

```
select ?person ?organisation
where {
    ?pers         :hasLocation    ?org .
    ?pers         rdf:type        persname: .
    ?pers         rdfs:label      ?person .
    ?org          rdf:type        org: .
    ?org          rdfs:label      ?organisation .
}
```

Of course, the output list is not 100% accurate because the *txt2rdf* pipeline is imperfect. Examination of the results also showed that some relations, whilst "correct" in the sense that they were accurately extracted from text notes, is

[21] The query is reproduced rather than its results out of a, possibly excessive, concern about publishing personal data.

[22] A key objective of the Semantic Web is that humans will have much less need to be involved in reading intermediate results, as these will all be machine-processable.

actually out of date (because individuals have moved on) – clearly the system can only ever be as good as the source data it is based on, and dealing with facts that are true only in a limited time period is still an open issue for the Semantic Web. Nevertheless, this simple trial showed that a structured query could be run to find information that was buried in database free text notes but never collated into fixed fields. Clearly one possible application of this technology is as a means of populating RDB fields when curating new types of information becomes necessary. For RCAHMS, the extraction of event-related data is just such a case. It is encouraging that, as the results in Section 3.1 show, performance on EVENT entities and *event* relations is especially good.

In another experiment, a SPARQL query was used to answer the request **"Show me what was found at places in Shetland, when and by whom"**. This is an example of where the RDB fields can give only a very partial answer, but the augmented graph has the information needed. In effect, the RDB structure is used to narrow down the hit list using the location parameter (Shetland) and the rest of the result comes from the text relations. As just mentioned, "find" events are amongst those that RCAHMS is now starting to record formally but which were not held in database fields in the past. This query cannot be framed against the relational database.

Table 3 shows a sample of the results of this query. It illustrates the variation in formats for dates, personal names and object types, when mentioned in free text. There is a need for much more normalisation than has so far been attempted, and this is something that will be attempted in future work if possible.

Once again it should be noted that, from the measured precision and recall scores, we are not expecting complete accuracy. Some of the entries in the table are obviously incorrect, such as the last site167 date and the second site979 object. In the first case, the error of attributing the finding of a Viking Grave to the Viking period is an incorrect relationship. The graph should (and maybe does in another triple) relate this period to the object, not the finding event. Regarding the second case, entity classification errors will have knock-on effects for relation extraction, which may be the cause of "SCALLOWAY" (a place on Shetland) being listed as one of J W Cursiter's finds, along with a polished serpentine knife, at site979 – the place name may have been wrongly identified as an object. It may instead be a relation classification error – classifying the relationship between the find event and the place as *hasPatient* instead of *hasLocation* but this seems less probable as the domains and ranges of the various predicates follow a clear pattern and it was found in trials that RE errors are more often due to the classifier failing to detect a relation than to misclassifying correctly found ones.

Several approaches to improving the performance of the *txt2rdf* pipeline are possible. The machine learning scores can probably be raised by working on the characterisation of NEs and relations through the feature selection process, but the penalties attached to this approach were noted at the end of Section 3.1.

Some very basic normalisation is performed on the extracted NE strings but much more could be done. Ideally, a full co-reference resolution module

site	object	date	agent
:site506	“spindle whorls”	“1948”	“NMAS”
:site506	“bowl”	“1948”	“NMAS”
:site506	“bead”	“1948”	“NMAS”
:site1385	“human bones”	“1833”	
:site510	“human remains”	“1858”	
:site245	“urns”	“1878”	
:site245	“urns”	“1903”	
:site245	“urns”	“1837”	
:site1441	“coins”	“1933”	“W C Carson”
:site1441	“coins”	“1924”	“W C Carson”
:site126	“comb”	“1960”	“National Museum of...”
:site126	“comb”	“1960”	“T Cluness”
:site1006	“human remains”	“1878”	
:site745	“bead”	“1862”	
:site745	“perforated whetstone”	“1862”	
:site997	“rotary quern”	“1933”	
:site997	“urns”	“1933”	
:site167	“GRAVE”	“1866”	
:site167	“Viking Grave”	“1866”	
:site167	“Viking Grave”	“Viking”	
:site538	“hammer-stones”	“1946”	“RCAHMS”
:site225	“hammer-stone”	“1946”	“RCAHMS”
:site225	“sinker”	“1946”	“RCAHMS”
:site766	“pot sherds”	“modern”	
:site681	“Cist”	“A.D. 1877”	“OS”
:site979	“polished serpentine knife”	“1885”	“J W Cursiter”
:site979	“SCALLOWAY”	“1885”	“J W Cursiter”
:site415	“axe”	“1933”	“National Museum of...”
:site415	“axe”	“1933”	“NMAS”
:site1102	“polished stone knives”	“28th May 1968”	“Peter Moar”
:site1102	“polished stone knives”	“28th May 1968”	“Henderson”
:site1102	“stone adze”	“May 1946”	“Peter Moar”
:site1102	“stone adze”	“May 1946”	“Lerwick Museum”
:site1102	“polished stone knives”	“May 1946”	“Moar”
...	...	...	...

Table 3: Sample results for “finds at Shetland sites” query

should be built into the pipeline, so that different text strings referring to the same thing can be unified. This module should also resolve pronoun anaphora, so that all the correctly extracted but completely useless relations of the form "`it` *hasLocation* [a PLACE]" can be assimilated (instead of being discarded as at present). A more ambitious, but clearly useful, project would be to deal with negation, which is not catered for so far. In one of the query experiments, asking for sites where human bones were found, the results included a site whose text notes described several apparent burial sites but included the phrase "no human remains were found". This is a notoriously difficult issue in NLP that traditionally is dealt with by full parsing methods, rather than the deliberately "shallow" processes that were used in *tether* for scalability reasons.

For all its shortcomings, the *txt2rdf* system does the basic job it was designed to do: it shows the potential of the NLP approach for automatically extracting structured data from free text on a large scale.

4.3 Limitations Imposed by the Schema

The schema for RDF triples derived from free text must, of necessity, be fairly simple, because very detailed distinctions cannot be captured accurately enough by computational methods. It doesn't automatically follow that one should use the same schema for the associated RDB data, but this is makes it easier to run simple queries across the entire dataset. One could include the finer detail available from the RDB data by turning RDB attributes into RDFS subproperties of the less granular text-derived properties, but I chose not to introduce this extra complexity over the predicates. (Instead, the *tether* design uses a hierarchy of **classes** to capture the important distinctions, as explained in Section 2.2.) There is a trade-off between query simplicity and power: more queries can be answered if the schema is more detailed, but it becomes harder to design the query statements.

Even though the aim in *tether* was for a minimal schema structure, the "scanned archive" example used above (in Section 4.1) shows that there is already a barrier to independent query agents addressing the graph without prior knowledge. Contrast this with existing web searching through Google, where one expects the query term to be sent across all available data with no advance preparation. The Semantic Web gives us the advantage (over Google) of being able to search for a term in context (for example, "tell me about Dunbar, but only where it's a site location, not a person's name") but, as well as picking the right term, we have to know in advance how to characterise the context (by specifying the type of node to search for).

Several questions arise: Is it possible to publish, and maintain, schema information for the whole of the growing mass of Linked Data? If not, can agents discover the schema for themselves? Is it essential to have a schema at all?

Manually maintaining some master catalogue of up to date schemas for every data collection across the planet seems highly problematic. If the maintenance is not by human intervention then we are already into the second question, of automatic schema discovery based on approaching some new data island (where

the "island" is perhaps a named graph – see Carroll et al. [2005] for a definition) and interrogating it. When RDFS or OWL underlie the graph it is easy enough to find the basic framework; for example, the following query would list all the classes in *tether*.

```
select ?class
where {
  ?class rdf:type rdfs:Class .
}
```

With a bit more effort one could extract the hierarchy of classes and properties, with domains and ranges. Yet even if a query agent were able to do all this, we are still a long way from making the semantics of the structure accessible, so that the agent can deduce which fragments of the structure it needs to address. One response (as has been proposed here) is to minimise the need for schema maintenance, by having a simple, generic structure that can cover all cultural domain data at a high level. The downside is that detailed knowledge remains beyond the reach of universal query agents, because it will always require local structure that in turn necessitates local schema knowledge.

This brings me to my third question, because an alternative response is to wonder whether we should consider managing without a schema altogether. Clay Shirky, for example, suggests that attempts to impose fixed categorisations on data are simply misguided [Shirky, 2006]. Following the "50 million Frenchmen can't be wrong" principle, he argues for social tagging, or folksonomies, instead of ontologies designed by experts. As a user of machine learning tools I believe that with a sufficient quantity of representative data one can build a categorising tool – which is a similar kind of process. However, the sample does have to be representative. Depending on the topic, public opinion can be very unreliable, and very volatile. A combined approach may be the answer, where statistical frequency can inform expert category design and, conversely, a pre-defined schema can be used to marshall disorganised masses of data.

One can simulate queries where there is no schema or where one has minimal schema knowledge using the DESCRIBE function in SPARQL, which simply returns all nodes connected to a specified node, irrespective of what the connection properties are. Some limited exploration of such querying was done, using query terms to try to pick particular nodes and then gathering their local context for post-processing. This is something to be explored in future work. It seems likely that full knowledge of the schema will always be difficult to realise in practical implementations, and approaches that exploit what knowledge the query agent can acquire whilst also giving scope for free exploration, seem worth pursuing.

A final point that is worth mentioning in the context of schema limitations is that there is, as yet, no way of ranking triples in an RDF graph. In other fields (such as neural networks), weighted graphs – where each arc has a weight of a particular value – have been extensively studied. Particularly where one is adding triples derived by probabilistic methods, as in my *txt2rdf* process, there are clear advantages in being able to assign weights to the links, which could be derived straightforwardly from the confidence factor returned by the

statistical classifier. Similarly, if one were to combine folksonomy structures with more formal RDF ontologies, as suggested above, a mechanism for weighting the strength of links would be valuable, based on the number of people who independently made that link. This is an area it would be very interesting to explore further.

5 Conclusions

This paper has given an overview of a lengthy programme of experimental work on practical exploitation of the Semantic Web in the cultural domain. Although the basic techniques are as generic as possible, the particular nature of cultural heritage material inevitably affects the design of component software.

We have seen how structured data, derived from relational databases, can be transformed to RDF graphs and thus become part of the Semantic Web. There is scope for grounding the data against published ontologies, a growing number of which are now being translated to RDF. Even if the ontology required is not yet in RDF format, the transformation of disciplined and highly structured data of this kind is fairly straightforward. In the cases described here, the SKOS framework was used as a basis.

However, it can be argued that the imposition of pre-defined taxonomies restricts the freedom of the new web, and alternative models for schema design were discussed. Future avenues to explore include methods of exploring graphs given only partial schema knowledge and the possibility of assigning weights to graph arcs to enable ranking of facts according to confidence in their correctness.

The central message of the paper is about augmenting the structured data with "facts" automatically extracted from free text. A pipeline, *txt2rdf*, was described which takes in plain text at one end and pumps out RDF triples at the other. These can then be easily combined with related Semantic Web data. The process is by no means perfect but the routes to improve it are very clear. Even as it stands, the pipeline has been shown to produce an integrated RDF graph structure that can answer queries for information that was impossible to retrieve previously. Taken together with the fact that the interconnection of datasets is a fundamental aspect of the Semantic Web, there is potential for powerful and flexible public access systems to cultural archives everywhere.

Acknowledgements

I am most grateful to the reviewers and editors of JoDI (*Journal of Digital Information*) for their helpful and insightful comments during the preparation of this paper; also to ESRC (the UK Economic and Social Research Council) who funded the work.

References

Jesús Barrasa, Óscar Corcho, and Asunción Gómez-Pérez. R_2O, an extensible and semantically based database-to-ontology mapping language. In *Proceedings of the 2nd Workshop on Semantic Web and Databases*, Toronto, Canada, August 2004.

Tim Berners-Lee. Relational Databases on the Semantic Web. Internet note, 2006. URL `http://www.w3.org/DesignIssues/RDB-RDF.html`. v 1.22 2006/02/01 (originally published September 1998).

Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web. *Scientific American*, 284:34–43, 2001.

Christian Bizer and Richard Cyganiak. D2RQ - Lessons Learned. In *Proceedings of the W3C Workshop on RDF Access to Relational Databases*, Cambridge, MA, USA, October 2007. W3C. URL `http://www.w3.org/2007/03/RdfRDB/papers/d2rq-positionpaper/`. Position paper.

Kate Byrne. *Populating the Semantic Web—Combining Text and Relational Databases as RDF Graphs*. PhD thesis, University of Edinburgh, 2009. URL `http://homepages.inf.ed.ac.uk/kbyrne3/research.html#papers`.

Kate Byrne. Having Triplets – Holding Cultural Data as RDF. In Martha Larson, Kate Fernie, Johan Oomen, and Juan Manuel Cigarran, editors, *Proceedings of the ECDL 2008 Workshop on Information Access to Cultural Heritage*, Aarhus, Denmark, September 2008. URL `http://ilps.science.uva.nl/IACH2008/`. ISBN 978-90-813489-1-1.

Kate Byrne. Nested Named Entity Recognition in Historical Archive Text. In *Proceedings of the IEEE International Conference on Semantic Computing (ICSC2007)*, Irvine, California, September 2007. doi: http://doi.ieeecomputersociety.org/10.1109/ICSC.2007.107. URL `http://csdl2.computer.org/persagen/DLAbsToc.jsp?resourcePath=/dl/proceedings/&toc=comp/proceedings/icsc/2007/2997/00/2997toc.xml`.

Jeremy J. Carroll, Christian Bizer, Pat Hayes, and Patrick Stickler. Named Graphs, Provenance and Trust. In *Proceedings of the 14th International World Wide Web Conference (WWW2005)*, pages 613–622, Chiba, Japan, May 2005. IW3C2. URL `http://www2005.org/cdrom/contents.htm`.

Nick Crofts, Martin Doerr, and Tony Gill. The CIDOC conceptual reference model: A standard for communicating cultural contents. *Cultivate Interactive*, (Issue 9), Febrary 2003. URL `http://www.cultivate-int.org/issue9/chios/`.

Nadine Cullot, Raji Ghawi, and Kokou Yétongnon. DB2OWL: A Tool for Automatic Database-to-Ontology Mapping. In *Proceedings of 15th Italian Symposium on Advanced Database Systems (SEBD 2007)*, pages 491–494, , 17–20 2007., Torre Canne, Italy, June 2007.

Lee Feigenbaum, Ivan Herman, Tonya Hongsermeier, Eric Neumann, and Susie Stephens. The Semantic Web in Action. *Scientific American*, 297: 90–97, December 2007. URL `http://thefigtrees.net/lee/sw/sciam/semantic-web-in-action`.

Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. Doctoral dissertation, University of California, Irvine, 2000. URL `http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm`.

Eero Hyvönen, Tuukka Ruotsalo, Thomas Häggström, Mirva Salminen, Miikka Junnila, Mikko Virkkilä, Mikko Haaramo, Eetu Mäkelä, Tomi Kauppinen, and Kim Viljanen. CultureSampo–Finnish Culture on the Semantic Web: The Vision and First Results. In K. Robering, editor, *Information Technology for the Virtual Museum*, pages 25–36, Berlin, November 2007. LIT Verlag. URL `http://www.seco.tkk.fi/publications/`.

Graham Klyne and Jeremy J. Carroll, editors. *Resource Description Framework (RDF): Concepts and Abstract Syntax*. W3C, 10 February 2004. URL `http://www.w3.org/TR/rdf-concepts/`.

Edmund Lee, editor. *MIDAS Heritage — a data standard for the historic environment*. Forum for Information Standards in Heritage (FISH), 2007. URL `http://www.midas-heritage.org.uk/`.

Alistair Miles, Thomas Baker, and Ralph Swick, editors. *Best Practice Recipes for Publishing RDF Vocabularies*. W3C, 23 January 2008. URL `http://www.w3.org/TR/2008/WD-swbp-vocab-pub-20080123/`.

Satya S. Sahoo, Wolfgang Halb, Sebastian Hellmann, Kingsley Idehen, Ted Thibodeau Jr, Sren Auer, Juan Sequeda, and Ahmed Ezzat. A Survey of Current Approaches for Mapping of Relational Databases to RDF. Report from Incubator Group 2009-01-31, W3C, January 2009. URL `http://www.w3.org/2005/Incubator/rdb2rdf/RDB2RDF_SurveyReport.pdf`.

Leo Sauermann and Richard Cyganiak, editors. *Cool URIs for the Semantic Web*. W3C, 17 December 2007. URL `http://www.w3.org/TR/2007/WD-cooluris-20071217/`. Contributors: Danny Ayers and Max Völkel.

Guus Schreiber, Alia Amin, Mark van Assem, Victor de Boer, Lynda Hardman, Michiel Hildebrand, Laura Hollink, Zhisheng Huang, Janneke van Kersen, Marco de Niet, Borys Omelayenko, Jacco van Ossenbruggen, Ronny Siebes, Jos Taekema, Jan Wielemaker, and Bob Wielinga. MultimediaN E-Culture Demonstrator. In *Proceedings of the International Semantic Web Conference (ISWC 2006)*, volume 4273, pages 951–958, Athens, Georgia, November 2006. LNCS. doi: 10.1007/11926078_70. URL `http://www.springerlink.com/content/358uq570630127p3/?p=29569a0a8d9b46ce98c27240b467b5f9&pi=3`.

Clay Shirky. Ontology is Overrated: Categories, Links and Tags. Internet, 2006. URL http://www.shirky.com/writings/ontology_overrated.html.

Virtuoso. Virtuoso: A Superplatform for Merger Driven Data Integration and Business Process Services. Internet White Paper, 2006. URL http://virtuoso.openlinksw.com/overview/.

Zhaohui Wu, Huajun Chen, Heng Wang, Yimin Wang, Yuxin Mao, Jinmin Tang, and Cunyin Zhou. Dartgrid: a Semantic Web Toolkit for Integrating Heterogeneous Relational Databases. In *Semantic Web Challenge at 4th International Semantic Web Conference (ISWC 2006)*, Athens, USA, November 2006. URL http://www.aifb.uni-karlsruhe.de/WBS/ywa/publications/wu06TCM_ISWC06.pdf.