Tethering Cultural Data with RDF

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

Abstract

This paper describes a research project leading to the development of a data querying application based on the Jena RDF store. The starting point is a collection of relational databases holding cultural heritage material from the National Collections of Scotland. This source data is a mixture of fixed fields and free text, supported by background material such as domain thesauri. The aim is to translate all the relevant material into RDF triples, as a step towards building a query application aimed at non-expert users who do not know how the data is structured and are ignorant of the specialist domain terminology needed to write good queries. The paper describes how the translation to RDF addresses both of these problems. Two core tasks are involved: extraction of two-place relations from free text using Natural Language Processing (NLP) techniques, and automatic assembly of all the relevant data into a graph database. An interactive query interface is proposed, in which a tailored summary based on the user's initial query is generated, and the user is then invited to refine the query based on this information.

1 Introduction

The programme of work this paper describes was inspired by the need to improve public access to the collections held by cultural heritage organisations in Scotland. These collections describe historic sites, buildings and objects, and comprise a mixture of structured data, text, and graphical material such as photographs, maps, plans and so forth. Although digitising the entire collections will take many years, we are now at the stage where all the text-based data is on computers along with a growing body of accompanying graphical archive. Naturally there is much interest in improving access to the material over the Internet, but although there are many query interfaces available, they are not ideal: to get the best out of them the user needs prior understanding of the domain terminology and a grasp of how the collections are structured. Although much of the useful information is in text fields, query access to free text ranges from awkward to impossible. Another problem is that the material has been collected over decades or even centuries, and in the past was aimed at specialists. Transforming it into something accessible to the general public is a difficult and potentially very expensive task. The aim of the project described here is to

explore how Natural Language Processing (NLP) techniques and ideas from the Semantic Web can bring us closer to the goal of making the information easily available to interested members of the general public, who are now the core audience. Three of the National Collection bodies have contributed copies of their data for the project: RCAHMS,[1] NMS,[2] and NLS.[3] It was important to have a selection of datasets because an additional project goal is to facilitate cross-collection querying so that, for example, a user who is interested in a particular archaeological site from the RCAHMS database could be offered information on objects found at it, from the NMS collections.

One of the key problems is to find a way of representing the information expressed in the free text notes so that it can be easily queried alongside fixed-field data. This will be dealt with as a relation extraction task, where a relation is defined as a two-place predicate or `(subject, predicate, object)` triple, such as `(V G Childe, excavated, Skara Brae)`.[4] The fixed-field and thesaurus data is similarly transformed into triples, which are then held in a triple store repository. The data transformation tasks are described in Section 2.

Once all the relevant information has been translated into the simple, standardised format of an RDF graph, it becomes possible to create a query mechanism that can guide the user through the data by exploring potentially useful links and summarising across subgraphs. Data presentation is another key aspect of accessibility — it may be a non-trivial exercise to produce clear and readable information for the user. The use of RDF suggests the possibility of using natural language generation to convert RDF statements into sentences. Section 3 deals with these application issues. Finally, Section 4 gives a brief review and conclusion.

At the time of writing, the project is in the early stages of transforming the fixed fields and thesauri. This document outlines the detailed planning that has been done for the whole programme of work. The system has been named *Tether*.[5]

2 Translation to RDF

As explained above, all relevant source information is to be translated into a single format of binary predicates. The obvious physical implementation of this is as RDF triples, and the resulting new database will be a collection of RDF graphs. There are interesting issues surrounding data maintenance and updating but they will not be covered in this paper, where the assumption is that the data is read-only. There are many triple stores now available; Jena[6] was chosen as the

[1] The Royal Commission on the Ancient and Historical Monuments of Scotland, `http://www.rcahms.gov.uk/`.

[2] National Museums of Scotland, `http://www.nms.ac.uk/`.

[3] The National Library of Scotland, `http://www.nls.uk/`.

[4] The physical implementation will be in RDF, so the subject and predicate will in fact be URIs.

[5] "Tether" is a dialect word for "three", used in the north of England for counting sheep.

[6] `http://jena.sourceforge.net/`

implementation platform after a survey of a number of alternatives.[7]

The project data consists of approximately 250,000 site records and texts from RCAHMS with 750,000 associated archive item records, plus 114,000 archaeology database records with associated texts from NMS, and 20,000 records and texts from NLS covering historical collections including books, broadsheets, photographs and maps. Experiments so far suggest this will translate to something in the order of 20 million triples. It would certainly be possible to reduce the volume, but part of the project's aim is to assess the viability of the whole approach with real-world data and realistic database sizes. In choosing a triple store the link to persistent database storage (in this case in MySQL) was an important criterion. It might be possible to hold the entire *Tether* database in memory, but typically this will not be the case for real data collections, and arguably the more interesting Semantic Web applications are those capable of identifying relevant subsets of distributed stored information efficiently and loading them into memory for detailed processing.

The word "ontology" is sometimes used to refer to the type of structure being described, but here the term "graph database" will be used, reserving "ontology" for more formal systems with rulesets, over which it is possible to run logic processors and reasoners. At this stage of the project it is not clear how feasible or indeed useful it will be to attempt, for example, consistency checking across the structure.

There are three elements to the construction of the triples database, dealing respectively with database fields, thesauri and free text.

2.1 RDBMS to RDF

At its simplest, translating a relational database to RDF triples is a straightforward process, easily automated. Every table or relation in the database can be translated into a set of triples in the form `(relationID, attribute, value)`, where `relationID` indicates the parent table, `attribute` is an attribute or column name, and `value` is the value held in that column. Thus a ten-column database table would generate ten triples per row of data. See Berners-Lee (2006) for a discussion of the issues, particularly of how to generate the URIs.

For this project some minor variations to the basic procedure are proposed, firstly to "pre-join" or denormalise the tables where appropriate, and secondly to introduce some manual schema design by grouping similar attributes (predicates in RDF) together. In addition, empty fields are omitted, and concatenated database keys are replaced by new surrogate keys.[8]

Figure 1 illustrates the translation procedure for some simplified extracts from RCAHMS database tables. (In this diagram and the others, simple labels are shown instead of URIs, for readability.)

[7] The key considerations were the use of persistent storage, the design of the triples tables and the likely development and maintenance of the system.

[8] The SITE-ARC table in Fig. 1 is based on a real RCAHMS database table, which uses a concatenation of `siteNo` and `arcNo` as its primary key. This is awkward in RDF, where we want a single identifier for the resource node, so an extra column (`siteArcNo`) was added as a preliminary step.

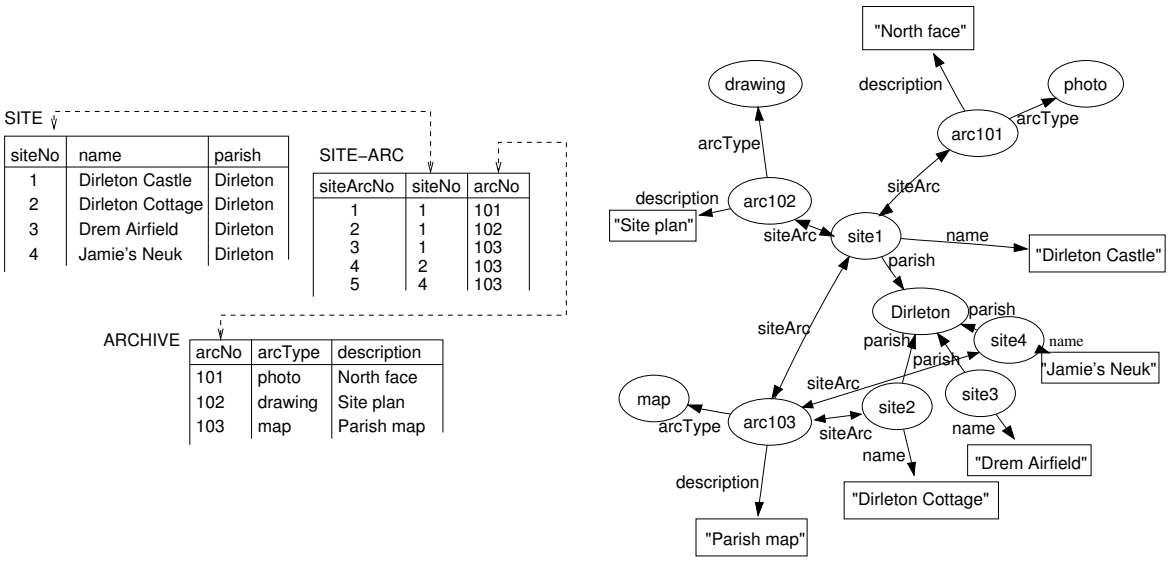

Figure 1: The table fragments are translated by creating one triple for each column of each row. For the subject of each triple an ID string (`site` or `arc`) is appended to the primary key field. The method takes advantage of the fact that the intermediary table between SITE and ARCHIVE has no local attributes beyond the necessary foreign keys.

Note that where there is an intermediary table resolving a many-to-many relationship between two database tables (SITE and ARCHIVE in this case), there is an opportunity for compression, which Figure 1 illustrates. In a relational database the intermediary must contain a foreign key for each of its parent tables, but in a graph representation this duplication can be avoided if the intermediary has no attributes of its own (i.e. if it contains only keys), by recognising that fact and implementing a special two-way link[9] between the relevant primary keys. This saves one triple per row of the intermediary table (around 750,000 triples in *Tether*). In practice, because of the way the data was loaded, one would probably know which way round the links point, but strictly the bi-directionality should be allowed for at query time. For example, a SPARQL query over the graph in Figure 1, to list any archive items associated with sites, could be expressed as:

```
SELECT ?sitename ?arcitem ?arcdesc
WHERE {
  ?site  name  ?sitename .
  OPTIONAL {
    {{?site siteArc ?arc} UNION {?arc siteArc ?site}}
    ?arc  arcType  ?arcitem .
    ?arc  description  ?arcdesc . }
  }
```

[9]Since all the properties are potentially reversible the directionality is not particularly significant, except as regards the semantics of the statements (see Section 3.2). This link is described as "special" because it can validly appear as `(siteX, siteArc, arcY)` or `(arcY, siteArc, siteX)`, which is not true in general for *Tether*.

The `UNION` clause ensures that all `siteArc` links are found, whichever way round they point. This query will return the following results:

```
 SITENAME            ARCITEM       ARCDESC
"Dirleton Castle"    photo        "North face"
"Dirleton Castle"    drawing      "Site plan"
"Dirleton Castle"    map          "Parish map"
"Dirleton Cottage"   map          "Parish map"
"Drem Airfield"
"Jamie's Neuk"       map          "Parish map"
```

In much the same way, the translation procedure can eliminate an intermediate node where the database uses codes for values taken from a picklist, by pre-joining the relevant tables. For example, `region` is held on site records as a code, pointing at a lookup table of region names. Instead of holding a triple such as `(site1, region, "01")` and then having to connect "01" to the region it refers to, we can de-reference the link and produce the triple `(site1, region, Borders)`. In practice it will probably be necessary to hold a literal for Borders region and also give it a URI as a resource that can be the subject of other statements, so there will be a number of related triples. The high number of triples required makes it even more desirable to eliminate any that are *not* actually needed.

In order to make the summarisation step (described in Section 3.1) tractable, it will (sadly) be necessary to introduce some manual schema design, rather than using an entirely automatic procedure. As they stand, each of the source databases has hundreds of separate fields, which translate into properties or predicates in RDF — the arcs on the graph. The intention is to simplify the graph by grouping similar predicates together, leaving the details of the relationships between them to the class or schema level as `rdf:type` and `rdfs:subClassOf` properties. So, for example, a new "location" predicate might include database attributes like "parish", "county", "grid reference", "postal address", and so on. Figure 2 shows an example.

2.2 Thesauri to RDF

There are many, many specialised thesauri for the cultural heritage domain.[10] Crofts et al. (2003) provide an overview of an attempt by CIDOC (Comité International pour la Documentation) to harmonise efforts, under the *Conceptual Reference Model*. Most of the thesauri are hierarchical, expressing subtype relations between classes. The contents of a representative set of these

[10]To name just a few: TMT (Thesaurus of Monument Types, maintained by English Heritage), FISH (Forum on Information Standards in Heritage, for the UK and Ireland, maintained by the MDA (which formerly stood for "Museum Documentation Association")), SPECTRUM (UK Documentation Standard for museums, maintained by MDA), AAT (Art and Architecture Thesaurus, one of the Getty vocabularies), ULAN (Union List of Artist Names, from the Getty), TGN (Thesaurus of Geographic Names, the third of the Getty set), TGM (Thesaurus for Graphic Materials, for image indexing, from the Library of Congress), LCSH (Library of Congress Subject Headings, much used in the library world).

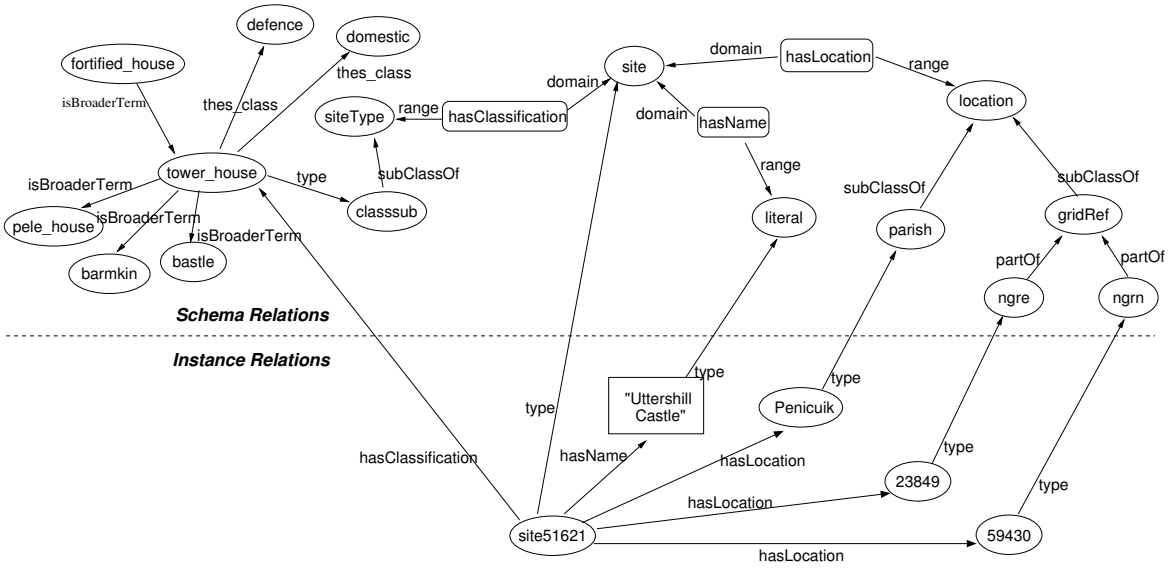

Figure 2: If the graph schema were hand-designed instead of automatically generated, complexity could be moved into the schema, with simpler instance relations.

standard thesauri will be translated into RDF triples. The W3C SKOS[11] (Simple Knowledge Organisation System), which is designed for handling thesauri, may provide the implementation framework. Figure 3 shows an example of how such an RDF graph might appear. The subset shown is the context of the two terms "cairn" and "chambered cairn", from TMT (Thesaurus of Monument Types).[12]

2.3 Relation Extraction from Free Text

The final and most interesting step is to populate the graph database with information derived from free text in the database records. This is an active research field and there are many possible approaches. See Riloff and Lorenzen (1999); Schutz and Buitelaar (2005); Huang et al. (2004) and Yangarber and Grishman (2001) for a contrasting range of examples. Most methods use some combination of machine learning tools and hand-crafted rules. For this project methods that can cope efficiently with large data volumes will be preferred, even if this means some sacrifice of thoroughness.

In outline, the steps that will be followed are:

1. Package the text as individual documents, linked to their parent database records. Then carry out standard NLP preprocessing steps such as tokenisation, part of speech tagging

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

[12] Duplicate links reflect redundancy in the source data, and will be automatically eliminated on loading into the triple store (which is a *set* of statements).

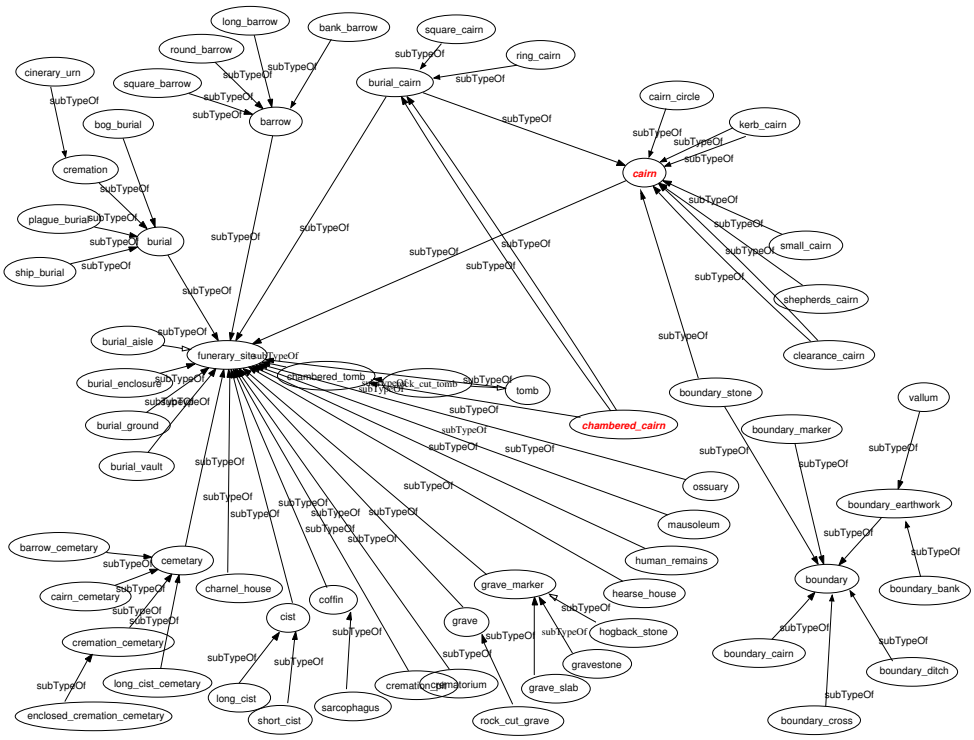

Figure 3: Graph generated for terms “cairn” and “chambered cairn”.

and chunking (i.e. identifying verb phrases, noun phrases, etc.)

2. Perform named entity recognition and classification. This is another standard NLP procedure, identifying references to significant “entities” such as people, places, site classifications and so on. The recognition step finds the strings (such as “Charles Rennie Mackintosh”) and the classification step assigns a class label (“person” in this case, or “architect” if the classification is more granular).

3. Deal, as well as possible, with the problems of co-reference resolution (the same entity appearing in multiple surface forms, such as “C R Mackintosh”, “Rennie Mackintosh”), and anaphora (e.g. pronouns will be replaced by their referents). Translate each entity into a canonical form with a URI, as they will become resources in the RDF graph.

4. Identify candidate predicates by doing frequency analysis on verb phrases. Cluster the candidates into synonym groups using a word-to-word distance measure. (There are several approaches to the word distance problem, often using WordNet (Miller et al., 1990). Cilibrasi and Vitanyi (2004) gives an interesting alternative, the “Normalised Google distance”.) If necessary, prune the set of synonym groups to a manageable number. They will become RDF properties.

5. Build triples from the sentences, where the predicate is one of the set just described and

the subject or object is a named entity, or both are. If there is only one NE, use the noun phrase or prepositional phrase required by the predicate.

The individual steps of this procedure have been tested with subsets of the data, to check their basic viability. Carrying out the whole process and evaluating it accurately will form a significant portion of the overall project.

3 Application Design

Once the RDF database has been fully populated, a query application will be built over it. Two significant aspects of the design are discussed in this section.

3.1 Querying Graph Data

There are any number of languages available for querying RDF data, and SPARQL[13] is the preferred choice here. Most of the Semantic Web tools are restricted (so far, at least) to simple subgraph matching and do not support graph searching algorithms, as is discussed by Angles and Gutierrez (2005) and Stuckenschmidt (2005). This seems curious, as one might have assumed that the benefit of translation to graph format was to allow functions like finding shortest paths between nodes, comparing the degree of nodes, and doing k-neighbourhood queries. For this project it seemed preferable to work within established or emerging standards, so some functions must be dispensed with. However, data repositories like Jena, which use a RDBMS back-end store, have to perform a SPARQL-to-SQL translation to send queries to the stored graph; so the possibility of using SQL directly against the triples tables is open. (Degree of node queries, for example, are straightforward in SQL, and k-neighbourhood queries can be done if k is fixed.)

To illustrate the proposed query procedure, consider a practical example such as a user query for "burial customs", against the RCAHMS data. This is a perfectly reasonable query from a non-specialist, but is too general to produce particularly useful results in practice. Querying CANMORE[14] for `Site Type = "burial"` (the nearest approximation to the query that's possible) produces 2,588 hits in either alphabetic or geographical order, with no subdivision between, say, 20th Century or Iron Age burial grounds. The *Tether* system will try to provide more guidance and context, by breaking down the mass of results into categories. The steps would be:

1. Check the query terms against the graph database to find preferred terms, related terms and subtype terms. For this example, we would get preferred terms like "burial", "funerary site", "burial enclosure", "burial cairn"; non-preferred terms including "boat burial", "Viking burial" and "burial chamber"; and narrower terms like "bog burial", "cremation", "plague burial", and "chambered cairn".

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

[14] The current RCAHMS online query facility, available at `http://www.rcahms.gov.uk/`.

2. The term "custom" is not likely to be found in a thesaurus (it does not occur in the Thesaurus of Monument Types), so it would pass on as an unprocessed term. The likeliest place for a match is within parts of the graph database derived from the free text. If no match is found the term would be discarded;[15] otherwise the process skips the next step, finds the "site" nodes related to each match found, and carries on from Step 4.

3. Establish that "burial" is a "site classification". Find all the nodes of type "site" with "classification" edges linking to one of the terms in the list built at Step 1.

4. Collect the *other* properties of these site nodes, e.g. location, period, associated people or events and so forth. Count the numbers within each broad group of properties (possible in SQL).

5. Amalgamate threads returned by each query term and present the user with a summary of the top groups, that might be of the form:

Period:	Iron Age, 350	Mediaeval, 640	Modern, 820	Others, 550
Location:	Strathclyde, 280	Lothian, 560	Orkney, 730	Others, 990
Has archive?:	Yes, 1500	No, 1100		

   ...etc. (These figures are notional.)

6. Allow the user to choose extra search criteria from these groups, e.g. Modern sites in the Lothians with archive material, and combine these criteria with the original query to produce the final result.

The differences between this and other query interfaces that allow combinations of search criteria are firstly that the criteria are specifically tailored for each query and guaranteed to produce relevant results for it, and secondly that no expertise is assumed on the part of the user, who only has to pick from criteria offered, not specify them.

3.2 Presentation of Results

One of the difficulties for organisations that want to make their material available to a wider audience is that it may not be in a suitable form for presentation to the lay reader. A text written for professional archaeologists may be almost unintelligible to, say, a school-child studying Scottish history. Ideally one would like to be able to present information on the same topics in different ways to different types of user, and perhaps even in the reader's native language. Natural language generation techniques make these realistic goals. The *M-PIRO* project (Isard et al., 2003) produced a system for generating descriptions of museum objects in different languages and for different levels of user. The system has been successfully adapted to handle a sample of RCAHMS data, as a proof-of-concept demonstrator. In principle at least, the small hand-built ontology used in the demonstrator could be replaced with a large RDF database. Thus there is

[15] One possibility is to use WordNet to find synonyms for terms that have no match in the graph database.

the possibility of producing descriptive text output, tailored for the individual user, from *Tether*. This would not be practicable if the data were not held as RDF.

4 Conclusion

This paper has given an overview of a research project that is currently underway. Although the primary goals are practical ones — concerned with real data accessibility issues — the aim is also to explore Semantic Web and NLP techniques as generic tools, and assess their applicability to cultural heritage information. One of the contentions being tested is that cultural data is, at present, generally held in a format that is not ideal for it, and translating it into a different layout will make it more accessible, particularly to the non-expert user who knows neither the data structure nor the jargon of the field.

The results will be evaluated against standard metrics for recall and precision, and in comparison to what is available in other existing systems. To be counted successful, the system should have performance comparable with existing SQL-based applications, increased retrieval power, and be simpler to use.

References

Renzo Angles and Claudio Gutierrez. Querying RDF data from a graph database perspective. In *2nd. European Semantic Web Conference (ESWC2005)*, volume 3532, pages 346–360, Heraklion, Greece, May 2005. Lecture Notes in Computer Science. URL `http://www.dcc.uchile.cl/~cgutierr/papers/`.

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).

Rudi Cilibrasi and Paul M. B. Vitanyi. Automatic meaning discovery using Google. Internet, 2004. URL `http://arxiv.org/abs/cs.CL/0412098`.

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/`.

Minline Huang, Xiaoyan Zhu, Yu Hao, Donald G. Payan, Kunbin Qu, and Ming Li. Discovering patterns to extract protein-protein interactions from full texts. *Bioinformatics*, 20(18):3604–3612, 2004. URL `http://bioinformatics.oxfordjournals.org/cgi/content/short/bth451v1`.

Amy Isard, Jon Oberlander, Colin Matheson, and Ion Androutsopoulos. Speaking the users' languages. *IEE Intelligent Systems*, 18(1):40–45, January/February 2003. URL `http://www.computer.org/intelligent/`.

George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J. Miller. Introduction to WordNet: an on-line lexical database. *International Journal of Lexicography*, 3(4):235–244, 1990. URL ftp://ftp.cogsci.princeton.edu/pub/wordnet/5papers.ps. Revised August 1993.

Ellen Riloff and Jeffrey Lorenzen. Extraction-based text categorization: Generating domain-specific role relationships automatically. In Tomek Strzalkowski, editor, *Natural Language Information Retrieval*, chapter 7, pages 167–196. Kluwer Academic, 1999. strz99.

Alexander Schutz and Paul Buitelaar. *RelExt:* a tool for relation extraction from text in ontology extension. In *Proceedings of the 4th International Semantic Web Conference (ISWC)*, Galway, Ireland, November 2005. URL http://www.dfki.de/~paulb/pub.html.

Heiner Stuckenschmidt. Towards an RDF query language - comments on an emerging standard. SIG SEMIS (Semantic Web and Information Systems), June 2005. URL http://www.sigsemis.org/columns/rdf/Querying/document_view.

Roman Yangarber and Ralph Grishman. Machine learning of extraction patterns from unannotated corpora: Position statement. In *Proceedings of Workshop on Machine Learning for Information Extraction*, pages 76–83, Berlin, 2001.