Using WYSIWYM to Create an Open-ended Interface for the Semantic Grid

F. Hielkema, C. Mellish & P. Edwards
Department of Computing Science
University of Aberdeen
Aberdeen
AB24 3UE
{fhielkem, cmellish, pedwards}@csd.abdn.ac.uk

1 Introduction

Central to the vision of the Semantic Grid is the adoption of metadata and ontologies to describe resources, to promote and enhance collaboration (De Roure et al., 2005). This raises the question of how such metadata comes into existence. Ideally the users should create it themselves, which raises the issue of how a scientist should create RDF. In our work in the area of e-social science[1], we aim to support social scientists in their research, using (Semantic) Grid technologies. For this, a tool is needed that facilitates easy creation of RDF by non-experts, to enable researchers to deposit and describe their own data. We believe that, for social scientists, natural language is the best medium to use, as the way they conduct their research and the structure of their documents and data indicate that they are more oriented towards text than graphics.

We originally envisaged such tools as being driven by an underlying ontology. However, from the start users expressed a fear of 'being trapped in the ontology', due to the contested nature of many social science concepts (Edwards et al., 2006). We therefore aim to maximise the users' freedom, keeping the tools open-ended by supporting dynamic evolution of metadata and integrating ontologies with folksonomies (Guy and Tonkin, 2006). A folksonomy is a social classification process where users can annotate their resources with keywords or tags, which are not restricted in any way. In some folksonomies, e.g. Flickr[2], users can use other users' tags, so that a set of frequent tags emerges. Using a folksonomy, we could suggest feasible tags to influence user-behaviour, without restricting the user to a pre-defined set of concepts.

Natural language applications are often domain specific and not very flexible. This makes the open-endedness we need a great challenge. Existing elicitation approaches, such as using Controlled Languages, restrict in great measure what the user can and cannot say. We believe that to achieve the desired open-endedness and flexibility, the best approach is not based on natural language processing, as it is as yet beyond the state of the art to reliably parse all user utterances, but based on natural language generation. In WYSIWYM (Power et al., 1998), the user can specify information by editing a feedback text that is generated by the system, based on a semantic representation of the information that the user has specified already. This NLG-approach, we believe, can give us both the flexibility we need and fluent language output. The expressivity of the language need not be restricted as it is generated by the system, and does not need to be parsed; and if we enable the user to modify the underlying data structure while using the tool, we have the desired open-endedness.

Figure 1 shows a feedback text (generated by the current system) for a scenario in which a social scientist is depositing data that forms part of a study into rural accessibility. Existing options for depositing such data (e.g. the UK Data Archive[3]) are found to be too restrictive by some social scientists. We therefore think there is scope for a tool that allows scientists to describe their data themselves, in a way they see fit.

In the next sections we will first describe some related work in NLG and the Semantic Grid community, then describe the design and implementation of our metadata-elicitation tool. We will discuss possible methods for keeping the tool open-ended and unrestrictive, and how folksonomies may be a part of the solution, and conclude with a description of remaining issues and plans for the future.

2 Related Work

Existing Semantic Grid tools that avoid the need to write RDF are often graphical (Handschuh et al., 2001). Natural language approaches include GINO (Bernstein and Kaufmann, 2006), an ontology editor with an approach reminiscent of NL-Menus (Tennant et al., 1983), and Controlled languages, e.g. PENG-D (Schwitter and Tilbrook, 2004).

Natural language approaches tend to restrict expressivity to ensure that every entry can be parsed,

[1] http://www.policygrid.org/
[2] http://www.flickr.com/
[3] http://www.data-archive.ac.uk/

Bibliography
Access to *'APAT'* is public. It was deposited on 9 March 2007.
It was deposited by *John Farrington*. John Farrington's email address is j.farrington@abdn.ac.uk. He is an employee of *the University of Aberdeen*.
John Farrington is the principal investigator of 'APAT'.

Methodology
'APAT's' observation units were individuals and focus groups.

Domain
'APAT' supports *'Settlements, Services and Access'*. John Farrington, *Jon Shaw*, *Matthew Leedal* and *Margaret Maclean* are the authors of 'Settlements, Services and Access.'

Figure 1: Example of a WYSIWYM description

limiting the language and often making it stilted, so that there is a small learning curve before the user knows which structures are allowed. Tools that generate natural language from ontologies (though not for elicitation purposes) include Wilcock (2003) and ONTOSUM (Bontcheva, 2005). Wilcock uses templates, achieving portability but paying a price in expressivity and accuracy. ONTOSUM assumes the ontologies contain labels with the appropriate lexicalisation of their resources, and that their part-of-speech tags can be easily derived.

In order to maintain full expressivity and to shorten the learning curve, we have elected to use WYSIWYM (What You See Is What You Meant) (Power et al., 1998). This is a natural language generation approach where the system generates a feedback text for the user that is based on a semantic representation. This representation is edited directly by the user by manipulating the feedback text. Figure 1 shows a feedback text generated by our system.

As the text is generated by the system and does not have to be parsed, we do not have to restrict what can be said, so the language retains its expressivity and the user does not need to learn what is acceptable input. The system is guided by an underlying datastructure, in our case a lightweight ontology. While the original WYSIWYM could only be ported to new domains by having an expert create a new lexicon, we wish to allow the user to extend the ontology while using the tool (i.e. while describing a resource). This, provided we have a NLG-component robust enough to deal with this, will ensure the desired open-endedness.

In the next section we describe the ongoing implementation of a WYSIWYM-tool for metadata elicitation from users unfamiliar with RDF. In section 4 we discuss ways to keep the tool open-ended.

3 Design and Implementation

We are building a tool that elicits metadata from the user in the form of RDF triples, i.e. statements of the form: 'subject - predicate - object'. The tool presents the users with a text containing an expansion point (anchor) for each object that is mentioned, which has a menu with possible properties associated with that object. These objects and properties are defined by an underlying OWL-Lite ontology[4]. The ontology we use for development is based on the UK Data Archive. This lightweight ontology is only a seed; users can extend or replace it (see section 4). We intend to ensure that other OWL-Lite ontologies can be substituted. Such ontologies should be well-formed, be clear about which objects are permitted in the domain and range of properties, and for the benefit of the generated text should have clear object and property names (e.g. HasAuthor), as these names are used for generation with only some minor adjustments (such as adding determiners and removing capitals).

The current system consists of five components: the semantic graph, the ontology reader, the RDF-creator, the natural language generator (text planner and surface realiser) and the interface.

The **interface** shows the feedback text with anchors indicating expansion points, which contain menus with types of information that can be added. Google Web Toolkit[5] was used to create the prototype interface.

The **semantic graph** stores the information the user is adding. Initially a generic graph is created, so an initial feedback text can be produced; the graph is updated each time the user adds information.

The **ontology reader** creates a model of a given OWL-Lite ontology, which is consulted throughout the building of the semantic graph and extended with all new properties or objects that the user adds. The ontology specifies the range and domain of the properties; i.e. the properties in each anchor menu, and the (type(s) of) resource that can be selected or added as the range of a selected menu item.

The semantic graph is translated to a list of RDF triples by the **RDF-creator**. These triples are stored, with the relevant resource(s), in a shared repository of social science resources on the Semantic Grid. The RDF-creator will in the future also store any changes the user made to the ontology.

The **natural language generator** maps the semantic graph to (HTML) text that contains anchors. It consists of a text planner and a surface realiser:

- Text Planner

  This component creates paragraph boundaries and headers. Each property in the semantic graph is mapped to a dependency tree (Mel'cuk, 1988). Before this mapping, the properties are grouped firstly for their source node (so all information about an object is grouped), secondly

[4] http://www.csd.abdn.ac.uk/research/policygrid/ontologies/UKDA/UKDA.owl
[5] http://code.google.com/webtoolkit/

for their label and thirdly for their target. Properties with identical source and label are marked for aggregation in the surface realiser.

- Surface Realiser

  The surface realiser maps the dependency trees to text using the SimpleNLG package[6]. To improve the conciseness of the text, it performs (at present) limited aggregation, when a group of properties has the same source and label (e.g. 'x and y are the authors of z'). It also keeps track of which objects are salient, in order to use pronouns. The text in Figure 1 was generated by the system.

The next section outlines our ideas for extending the system to make it open-ended and flexible.

4 Achieving Open-endedness

To avoid trapping the user in an ontology, two strategies present themselves: first, to make the ontology less restrictive by making it open-ended, and second, to do away with ontologies entirely. We are not prepared to do the latter, as ontologies provide a useful framework and are at the core of the Semantic Grid. We will therefore try to make the most of the first strategy in two ways: enabling the user to adapt any ontologies that are used, and where possible using a much less complex structure: the folksonomy.

4.1 Open-endedness in Ontologies

Although our system is driven by an ontology, we have kept this very lightweight (OWL-Lite, using only domain and range of properties and cardinality restrictions), and will give the user the power to adapt this ontology to his/her own needs. Extending an ontology with new classes and properties is no great problem; but those properties then need a suitable natural language representation. In our system this means they need an entry in the lexicon that maps them to a dependency tree (classes merely need a noun phrase).

A straightforward way to obtain such entries is to let a system administrator create them when needed. However, this would cause considerable delays for the user and would look almost as restrictive as not allowing new property creation at all. Instead, we want to enable the system to create these lexicon entries immediately, so the user can use the new property that session. Using the property name that the user provides, the system should generate an appropriate lexicon entry.

Hallett (2006) describes a portable WYSIWYM application that generates its lexicon automatically. It is unclear, however, how such a fully automated approach would work in practice. ONTOSUM (Bontcheva, 2005) depends on the user to provide lexicalisations of ontology resources. Their application generates the lexicon automatically, assigning pos-tag 'noun' to classes and 'verb' to properties; the user can then change the ontology or correct the pos-tags through an iterative process. This approach assumes that the user is willing to, and capable of, editing the ontology and providing pos-tags.

In contrast, we have decided to use a semi-automatic approach, but one that requires little expertise from the user. The system has the linguistic knowledge, and the user knows what the surface realisation should be; together they can generate a new lexicon entry. We are trying to identify common sentence types to mould into templates. The system inserts the root form of the property name into each template, and presents them for the user to choose from. For instance, given the property 'deposit', with domain 'Person' and range 'Document', the system may generate the following:

1. John Farrington deposits APAT
2. The depositor of APAT is John Farrington
3. APAT has the depositor John Farrington
4. APAT's depositor is John Farrington

The user chooses the most suitable representation, e.g. the first option, and then fine-tunes it by manipulating verb tense, actor and root, adding or removing determiners and prepositions, and switching the domain and range. In our example, changing the verb to past tense and passive action results in the surface form: 'APAT was deposited by John Farrington'. Once the user is satisfied with the representation, the corresponding dependency tree is stored in the lexicon and used for realising all future instances of this property.

4.2 The Freedom of Folksonomies

There are two types of property in OWL-Lite: object and datatype properties. The object properties have a class as their range, the datatype properties a data-type, such as 'string' or 'date'. The implementation allows the user complete freedom in specifying the value of a 'string' datatype, using free text to enter them. Unfortunately this means that we have no control over the quality of the entered data; we cannot prevent a user from entering nonsense.

This is unavoidable when the major goal is to afford the user freedom. However, the problem can be alleviated. Spelling mistakes can be prevented by checking all entries against a dictionary - but this can be very frustrating for the user, and a problem when he/she wants to enter new, foreign or subject-specific words. We believe folksonomies are a better solution here. A folksonomy stores which tags have been used with which frequency. In our system, each datatype property has its own folksonomy, as people would specify different values for the property

[6] http://www.csd.abdn.ac.uk/∼ereiter/simplenlg/

'country' than for 'sampling method'. Every time the user selects a datatype property and is prompted to enter a value, the corresponding 'tag cloud' is generated and shown to the user. A tag cloud gives an overview of tags used by other users; their frequency is reflected in the relative font size.

We believe that folksonomies will stimulate the emergence of a community set of tags (Guy and Tonkin, 2006), prompting the user to use the same values as other users, or to adopt a similar style. It should in large part protect the system from mistakes such as spelling errors, and, when queried, increase the likelihood of a search term being associated with more than one resource. The user however retains complete freedom, as he/she does not have to use the folksonomy values but can still use free text; and every entry the user makes is immediately added to the folksonomy. The folksonomy, then, allows us to subtly guide user behaviour, while being completely unrestrictive.

5 Conclusion and Future Work

We have outlined our ongoing development of a open-ended metadata elicitation tool, that allows users to create RDF by editing text. By using natural language generation instead of parsing we ensure that all user input is understood by the system. The system is driven by a lightweight ontology; open-endedness is achieved by enabling the user to extend the ontology where necessary. To enable this, a lexical specification for an appropriate surface realisation is generated by the system using feedback from the user, who chooses the most appropriate realisation and fine-tunes it.

To guide user behaviour without being restrictive, we employ folksonomies, collections of tags with information about how, when, where and how often they have been used. For each datatype property, a tag cloud is generated suggesting appropriate tags to use. When the user enters a new value, this is added to that property's folksonomy.

We intend to use our tool to populate a repository of social science resources. This will help us investigate user requirements on querying. Our aim is to use the same approach, using WYSIWYM with open-ended ontologies and folksonomies, for querying and information presentation. This way the user works with the same interface for each repository-related task, which should greatly improve usability. The folksonomy will be very useful in querying, with the tag cloud showing possible search terms and their frequency in the repository. A search term taken from the folksonomy would guarantee a result.

We believe that WYSIWYM will be as suitable for query formulation as for metadata elicitation. Information presentation in WYSIWYM could be an interactive process, allowing the user to request extra information where needed. Eventually, WYSIWYM will furnish us with an interesting and highly useful set of tools for the Semantic Grid.

References

A. Bernstein and E. Kaufmann. 2006. Gino - a guided input natural language ontology editor. In *International Semantic Web Conference 2006*, pages 144–157.

Kalina Bontcheva. 2005. Generating tailored textual summaries from ontologies. In *ESWC*, pages 531–545.

D. De Roure, N.R. Jennings, and N.R. Shadbolt. 2005. The Semantic Grid: Past, Present and Future. In *Proceedings of the IEEE 93(3)*, pages 669–681.

P. Edwards, J. Aldridge, and K. Clarke. 2006. A Tree Full of Leaves: Description Logic and Data Documentation. In *Proceedings of the Second International Conference on e-Social Science*, Manchester, UK.

M. Guy and E. Tonkin. 2006. Folksonomies: Tidying up Tags? *D-Lib Magazine*, 12(1).

C. Hallett. 2006. Generic Querying of Relational Databases using Natural Language Generation Techniques. In *Proceedings of the Fourth International Natural Language Generation Conference*, pages 88–95, Nottingham, UK.

S. Handschuh, S. Staab, and A. Maedche. 2001. Cream: Creating relational metadata with a component-based, ontology-driven annotation framework. In *K-CAP '01: Proceedings of the 1st international conference on Knowledge capture*, pages 76–83, New York, NY, USA. ACM Press.

I.A. Mel'cuk. 1988. *Dependency Syntax: Theory and Practice*. State University of New York.

R. Power, D. Scott, and R. Evans. 1998. What You See Is What You Meant: Direct Knowledge Editing with Natural Language Feedback. In *Proceedings of the Thirteenth European Conference on Artificial Intelligence*, Brighton, UK.

R. Schwitter and M. Tilbrook. 2004. Controlled Natural Language meets the Semantic Web. In *Proceedings of the Australasian Language Technology Workshop 2004*.

H.R. Tennant, K.M. Ross, R.M. Saenz, C.W.Thompson, and J.R. Miller. 1983. Menu-based Natural Language Understanding. In *Proceedings of the Twenty-first Annual Meetings on Association for Computational Linguistics*, pages 151–158, Cambridge, Massachusetts.

G. Wilcock. 2003. Talking OWLs: Towards an Ontology Verbalizer. In *Human Language Technology for the Semantic Web and Web Services (ISWC'03)*, pages 109–112, Sanibel Island, Florida.