Flexible Natural Language Access to Community-Driven Metadata

F. Hielkema, P. Edwards, and C. Mellish

Department of Computing Science
University of Aberdeen
{fhielkem, pedwards, cmellish}@csd.abdn.ac.uk

Abstract. A key issue in the Semantic Web is providing easy access to metadata for non-computer scientists. We believe that natural language is the best medium for this, and that the metadata framework should be open-ended in nature. We discuss four challenges: what type of interface to use, how to associate linguistic information with ontologies, how to achieve open-endedness, and how to stimulate collaboration. A metadata elicitation tool is described that uses natural language generation to shape the interface, and integrates dynamic ontologies with folksonomies. Finally, we argue for a new annotation mechanism which associates linguistic information with ontology resources.

1 Introduction

In the PolicyGrid[1] project we are investigating how best to support social science researchers through the use of Semantic Grid [1] technologies. The Semantic Grid is often described as an 'extension of the current Grid in which information and services are given well-defined meaning, better enabling computers and people to work in cooperation'. Semantic Grids thus not only share data and compute resources, but also share and process metadata and knowledge. Our work [2] involves close collaboration between computer scientists and social scientists. These interactions have allowed us to explore a range of issues, including: the extent to which these researchers are comfortable with the Grid as a framework for research practice and collaboration; if ontologies are appropriate (and acceptable) to this community as a way of representing concepts to facilitate their research activities; the utility (or otherwise) of existing metadata frameworks in use by the social sciences; and how best to integrate e-science tools and methods into existing working practices. A key aspect of our work is concerned with support for creation of metadata and access to resources annotated by metadata. We argue that support for these activities by 'normal' (i.e. non computer scientist) users is an important challenge for the entire Semantic Web/Grid research community. Such users are unlikely to understand the finer points of logic formalisms and formal languages, therefore metadata must be presented in an

[1] Funded under the UK Economic and Social Research Council *e-Social Science* programme; grant reference RES-149-25-1027 (http://www.policygrid.org)

accessible format that is easy to understand, allowing users to perform a range of metadata operations (creation, querying, browsing). As most members of the social science community are unfamiliar with complex formalisms such as RDF, this makes them a representative group for the 'non-computer science users' of the Semantic Web/Grid. Within PolicyGrid we are providing a shared Grid repository for social science resources, such as datasets and publications. These resources need an associated description, which should be provided by the user. The first step, therefore, is to provide a tool for metadata elicitation that enables users to create a description of their resources using RDF.

The social science research domain is difficult, if not impossible, to capture in a static ontology. Social scientists argue that concepts are imprecise, contested and mutable [3]: subject to many, conflicting and changing definitions. We therefore need a more dynamic, flexible structure that is open-ended and changes with the users. In this paper we argue that the solution lies in creating 'community-driven ontologies', in two ways: first, enabling users to extend ontologies easily, and second, integrating these dynamic ontologies with folksonomies [4, 5]. 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 can suggest feasible tags to influence user-behaviour, without restricting the user to a pre-defined set of concepts. We argue that such a community-driven approach provides us with the desired open-endedness and avoids the problems associated with trying to model a rich and varied research domain via a static ontology.

Existing Semantic Grid tools that provide access to metadata are often graphical [6, 7]. However, we believe that natural language is more accessible to new users than graphical representations, as we all use language in our daily lives, while graphical readership is an acquired skill [8]. We expect that most of our target community will be unfamiliar with RDF and graphical presentations thereof; for them, a natural language representation would be easier to take in and less susceptible to misinterpretations. We have therefore decided to provide a tool which utilises natural language to make metadata accessible.

We face four challenges if we want to provide access to collaborative metadata with natural language:

1. What type of interface do we use, and what drives it? We believe a natural language generation (NLG) technique, with an underlying ontology to drive it, is the best approach. Natural language generation can of course be used when presenting information for the user to browse, but also for metadata elicitation and query building, by generating a text for the user to edit. This avoids the problems inherent in parsing language. Section 2 discusses these issues and describes the implementation of an NLG-tool for metadata elicitation.
2. To generate language from ontologies, there has to be a lexicon for each domain, which provides linguistic information for its classes and properties.

[2] http://www.flickr.com/

How do we associate this linguistic information with ontologies? How and where do we represent this linguistic information? Our approach, which involves the use of annotation values to link each resource to a lexicon entry, is discussed in Section 3.

3. How do we achieve open-endedness? Enabling the user to extend and create ontologies would be a major step in this direction, but that leaves the issue of who creates the appropriate linguistic information. In our tool we also integrate ontologies with more light-weight data structures, i.e. folksonomies, which do not constrain but advise the user. This is discussed in Section 4.
4. Finally, there is the challenge of stimulating fair and safe collaboration within and between communities, e.g. through collective tagging. We have yet to address this challenge in our work, but our ideas in regard to this point are outlined in Section 5.

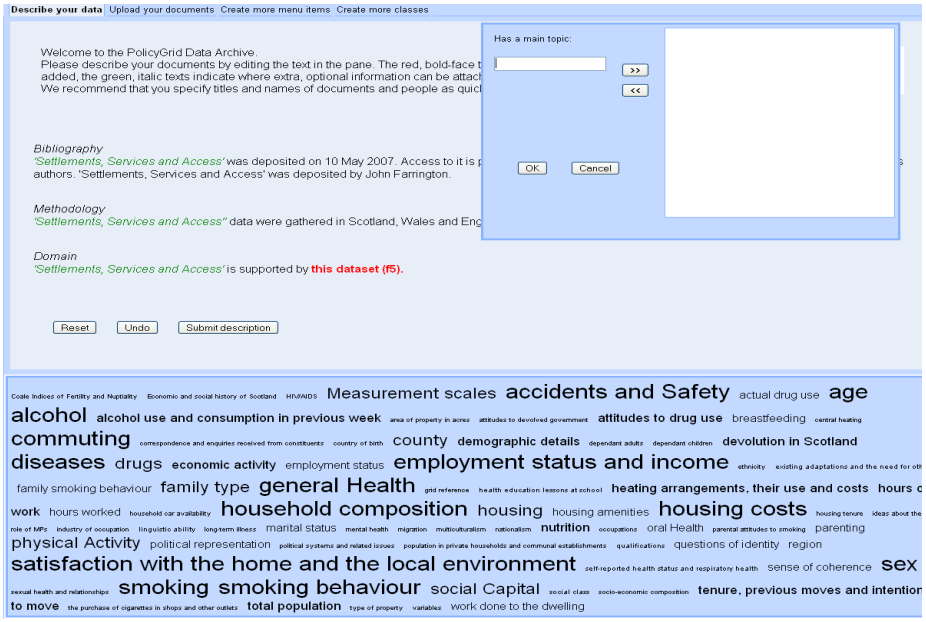

Fig. 1. Metadata elicitation tool: a user supplies values for the property *main topics*.

2 Supporting Natural Language Access to Metadata

A number of tools exist that provide natural language access to metadata. One such system for the Semantic Web is GINO [9], an ontology editor with an approach reminiscent of NL-Menus [10], where the user constructs a query word for

word by selecting terms from menus. Some systems use a controlled language, e.g. PENG-D [11], to restrict the input. A controlled language takes a small subset of natural language and guarantees successful parsing of all those utterances; however, the user has to learn what the boundaries of this subset are. Systems usually restrict what can be said one way or another because parsing all user utterances is beyond the current state-of-the-art. Tools that generate natural language from ontologies include Wilcock [12] and ONTOSUM [13]. Wilcock uses templates, achieving portability but paying a price in expressivity and accuracy, as template-generated text is frequently rather stilted and repetitive. ONTOSUM assumes the ontologies, provided by the user, contain labels with the appropriate lexicalisation of their resources. SEWASIE [7] lets users construct their query graphically but gives a natural language representation of their query as feedback.

A natural language generation approach to knowledge editing is WYSIWYM (What You See Is What You Meant) [14], where the system does not parse the user's utterances but instead generates a feedback text for the user to edit; the user can add information by clicking on portions of the text and choosing options from pop-up menus. As the text is not parsed but generated by the system, there is no need to restrict its expressivity, and the user does not need to learn what is acceptable. WYSIWYM has been used successfully both for database query composition [15] and multi-lingual document authoring [16]. Evaluation in [15] was very positive; the time needed for users to familiarise themselves with the system was very brief.

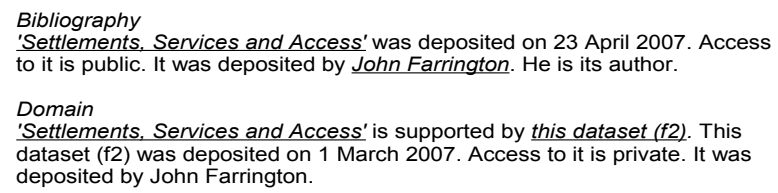

Fig. 2. WYSIWYM feedback text example.

We believe that WYSIWYM can provide the basis for a single style of interface that supports the user in building, querying or receiving information based on Semantic Web metadata. To date, we have implemented a tool (see Figure 1) for RDF metadata elicitation; Section 5 discusses our intention to implement tools for querying and browsing metadata. Our goal is to make this tool easy to use for users who have no previous experience with metadata and RDF. The user uploads his resources (e.g. datasets or publications) and describes them in natural language by editing a WYSIWYM-style feedback text; when he is finished the text is translated by the system to RDF-triples. Figure 2 shows an example of a natural language description taken from our interface.

The system is driven by an underlying, lightweight OWL-lite ontology (see Figure 3) based on the Economic & Social Research Council's UK Data Archive[2]. The UKDA is the largest collection of digital data in the social sciences and humanities in the UK, and uses a schema based on the Data Documentation Initiative[3] (DDI) and the Text Encoding Initiative[4] (TEI), which both map onto Dublin Core but provide more relevant detailed descriptions. We have tried to keep contested domain information out of the ontology, to avoid the kind of controversy described in Section 1. Instead, it is very much a utility ontology, resembling a library catalogue system. It contains a number of 'domain-neutral' classes, mostly corresponding to concrete objects, such as *Document* and *Person*. These classes have some Dublin Core-like properties such as *HasDepositor* and *HasAuthor*, but also properties with which more about the domain can be stated (*HasMainTopic*, *UsesMethodology*) and how resources relate to each other(*Supports*, *Contests*).

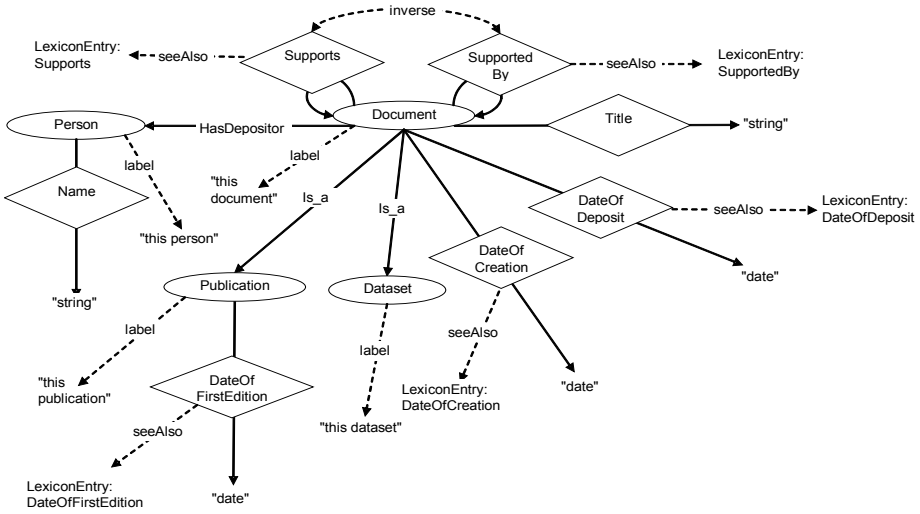

Fig. 3. Detail from the UKDA ontology.

The tool employs an underlying semantic graph to store information (see Figure 4). Initially this is a generic graph, containing only a document and a date of deposit; all information the user specifies is immediately added to the semantic graph. This graph is used to generate the feedback text, which is regenerated every time the user adds information. The feedback text contains

[2] http://www.data-archive.ac.uk/
[3] http://www.icpsr.umich.edu/DDI/
[4] http://www.tei-c.org/

expansion points (anchors) marked in red boldface and green italics, which open pop-up menus when clicked. The anchors correspond to individuals of classes in the ontology, while the menu items of an anchor correspond to the properties that individual can have. When the user selects a menu item and/or specifies a value, this information is added to the graph and the text is immediately regenerated. In Figure 1, the user has already supplied some information, e.g. the title, depositor and authors of the resource he is depositing, where its data were gathered, and that it is supported by some dataset. He is now supplying one or more main topics, advised by a 'tag cloud' of topics that have been supplied on earlier occasions by other users.

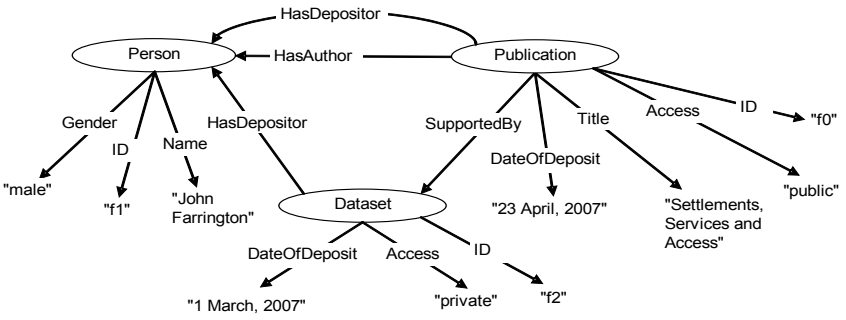

Fig. 4. Semantic graph corresponding to the feedback text in Figure 2.

OWL has two types of property: object and datatype properties. When the user selects a menu item that corresponds to a datatype property, he is asked to provide a value for it. The user is free to enter whatever values he wishes. If he selects an object property, he can select an existing instance (e.g. select an existing *Person* as value for the *HasAuthor* property on a *Document*). Alternatively he can have the system create a new instance in the appropriate range; for instance, *HasAuthor* has as range the class *Person*, so the system will create a new instance of that class.

3 Attaching Linguistic Information to Ontologies

Natural language access to ontologies relies on there being a lexicon of some kind which maps between ontology resources and natural language terms. We can attach lexicalisation information with RDFS via the *rdfs:label* and *rdfs:comment* properties, the former envisaged for specifying the 'name' of a resource and the latter for providing a longer 'description'. This is essentially the approach assumed by ONTOSUM [13]. But, in spite of the fact that linguistic descriptions have important internal structure, the values of these properties are simply lit-

erals, which are not controlled in any way and which are not amenable to automated processing. This seems contrary to the spirit of the Semantic Web. There is an increasing need for some standard and semantics-aware way of attaching linguistic information to ontologies.

Our system maps classes (and instances) to noun phrases (e.g. 'This person', 'Thomas', 'That Dutch guy in wooden shoes'), while properties are mapped to *dependency trees* [17] that represent sentences (with the source and target individuals inserted, e.g. 'This person is the author of this document'). These dependency trees (see Figure 5 for an example) state syntactic properties and semantic roles (dependency labels) for all words and word groups in the sentence; nodes representing individual words also have morphological information, stating verb tense, singular vs. plural, etc.

The advantage of having linguistic structure represented as dependency trees, rather than fixed templates or strings, comes from the flexibility in how they can be used. If every use of a property was simply mapped to a fixed sentence and then shown, the text would risk becoming very repetitive (e.g. 'This document was deposited on 11 May 2007. John is the author of this document. Paul is the author of this document.') To make the text more concise, we can perform operations on the dependency trees, e.g. aggregation and pronominalisation. One case of aggregation is the process of combining two sentences into one; pronominalisation is the substitution of pronouns (he, it) for names and descriptions ('John', 'this document'). Applying these processes, the text given above would become: 'This document was deposited on 11 May 2007. John and Paul are its authors.' This is rather more concise and actually more coherent as well. If the properties were mapped simply onto strings, this effect would be impossible to achieve; but with dependency trees, these operations are easily implemented. Another advantage of having these complex dependency trees is that, to realise a property, the lexicon entry of its inverse can be used, with only some minor tweaking (e.g. setting the main verb from active to passive). So, for instance, the information underlying 'This document was deposited by John Farrington' could be better realised as 'John Farrington deposited this document' in some circumstances. For these reasons, we deem it desirable to have a more complex linguistic representation than a simple string; the extra complexity significantly improves the quality of the generated text.

Figure 3 shows a detail from the UKDA ontology[5] that drives the current system. Each of its classes and properties need to have associated linguistic information - strings for the classes, dependency trees for the properties. The best solution is to store this information somewhere within the ontology. But what is the appropriate way to do this? We believe that the correct approach to representing linguistic structures is as resources in their own ontology. Our approach to associate linguistic information with ontologies is then to provide resources with annotation values that point to these linguistic resources. Although there may be different ways to represent linguistic information (i.e. different linguistic

[5] http://www.csd.abdn.ac.uk/research/policygrid/ontologies/UKDA/UKDA.owl

ontologies, just as there can be different ontologies in any other domain), the actual linking process can and should be standardised.

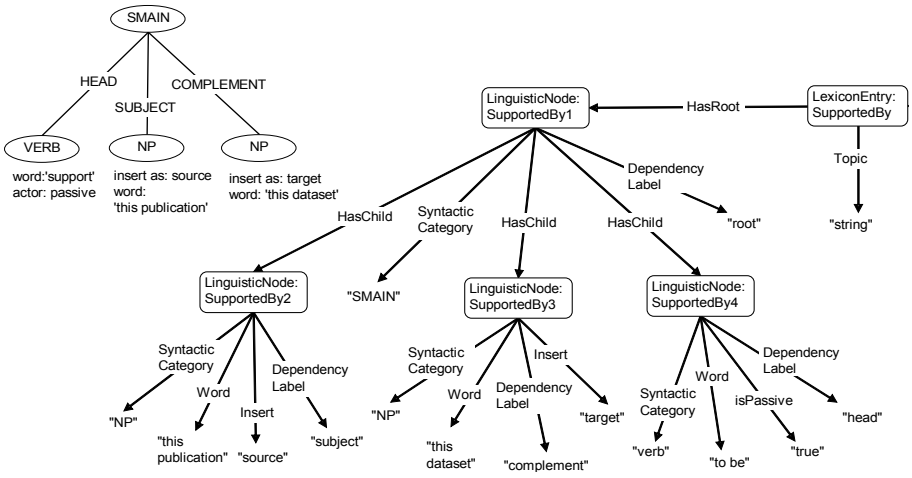

Fig. 5. Dependency tree (left) and lexicon entry (right) for 'This publication is supported by this dataset.'

In our current system, for classes a simple *rdfs:label* annotation value with the corresponding noun phrase as its value suffices. We have developed a separate, linguistic ontology[6] in which we can represent dependency trees (see Figure 6). This ontology is used to represent lexicon entries, as illustrated by Figure 5. A *LexiconEntry* points to an individual *LinguisticNode*, which represents the root of the represented dependency tree. *LinguisticNodes* can be *Nonterminals* or *Leaves*. The *LinguisticNodes* contain lexical information such as the syntactic category (noun, verb) and the dependency label (denoting the role it plays in the parent node, e.g. 'subject'). The *Leaves* also have a word or phrase, and morphological information detailing how to inflect that word. They can also have a property *insert*, signalling that they are placeholders for the noun phrases that represent the source or target of the property that is being realised.

Each property in the UKDA ontology has a *rdfs:seeAlso* annotation pointing to its corresponding *LexiconEntry* in the Lexicon ontology. Figure 5 shows the lexicon entry for the property *SupportedBy*. It is a simple tree consisting of a root (SMAIN, the full sentence), a subject, a head verb and a complement. The verb is set to passive. To get a suitable entry for *Supports*, the inverse property, all the system needs to do is take this entry, switch the source and target, and set the verb to active. For most properties (though not all) the lexical entry of their inverse can be derived successfully with some minor operations, such as

[6] http://www.csd.abdn.ac.uk/research/policygrid/ontologies/Lexicon/Lexicon.owl

switching the locations of the source and target in the sentence, or switching the main verb between 'active' and 'passive'.

The problem with using the *seeAlso* and *label* tags is that they may also be used for other things. A more elegant solution would be to define a new annotation property in OWL/RDFS, perhaps called *LinguisticInformation*, which always points to a dependency tree (or at least a linguistic structure in whatever ontology is chosen). The string corresponding to a class could be stored as a dependency tree with one single node, with syntactic category 'noun phrase'. To be suitable for natural language access, ontologies would then have to be supplied with *rdfs:LinguisticInformation* annotation values for all their resources.

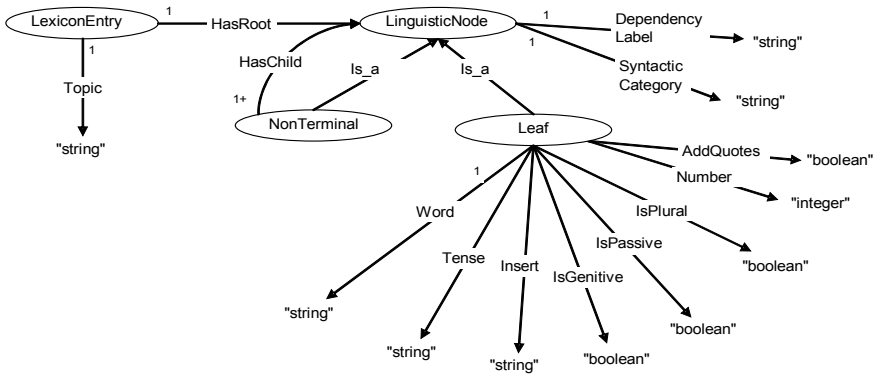

Fig. 6. Lexicon ontology.

4 Introducing Open-endedness

Earlier we discussed the problems associated with using static ontologies in e-social science. Instead of forcing users to use terms or ontologies they (partly) disagree with, we want to make our ontologies open-ended. This section describes how we are trying to achieve this.

As stated above, for datatype properties we let users supply whatever values they wish. However, this is likely to result in an immense proliferation of different tags, with every user specifying his own personal favourite terms. Moreover, it leaves the system vulnerable to typographical and other errors on the part of the user. To alleviate this problem, we integrate our ontologies with folksonomies, which advises the user of tags that are popular with other users. To avoid restricting users within a specific ontological framework, we also enable them to extend the ontology underlying the system by creating new classes and properties. There are linguistic implications for both these extensions.

4.1 Integrating Ontologies with Folksonomies

By giving users freedom, we have no control whatsoever over the values that are specified for datatype properties; in other words, we can expect a diversity of tags, and we cannot prevent erroneous metadata from being created. There is no way to prevent this without taking away the user's freedom. However, if we can advise the user in his choice of tags, this should help to prevent typographical errors, and support the development of community sets of tags. This should improves the consistency of resource descriptions (as the likelihood increases that the same tags are applied rather than synonyms), making it easier to find closely related resources. To this end we use folksonomies. In terms of the metadata generated, our folksonomy tags remain as unanalysed literals, rather than being elements in a tag ontology [5] or being mapped to ontology concepts [18].

Our approach is to have users create semantic metadata, and use folksonomies to advise them in their choice of tags. If the datatype is a string, a tag cloud is presented to the user (an example is shown in the lower half of Figure 1); we chose not to employ the folksonomy technique for numbers, dates and booleans. Every string datatype property has its own folksonomy, as different values apply to properties such as *HasCountry* and *UsesMethodology*. Every value that users supply for a string-datatype property is stored in that property's folksonomy. These folksonomies are shared by all users. The frequency with which a tag has been used is reflected in the relative font size when the tag cloud is displayed. The user can select a tag from the cloud, but can also supply his own tags. The user therefore retains complete freedom in specifying tags, but is advised by the folksonomy, which should alleviate the problems of spelling mistakes and inconsistent tagging.

Linguistic Implications We have no influence over what goes into a folksonomy, which means that we cannot guarantee that its values will 'fit' into the linguistic tree that a property is mapped to. For example, a user is entirely free to enter a meaningless value such as '–x–'. Currently, the linguistic trees expect two noun phrases, representing the individuals at the source and the target of the property, to be inserted; the linguistic tree determines their correct place and supplies their morphological information. But can we assume that folksonomy values will be proper noun phrases, amenable to the necessary inflections? As an example of the latter, imagine someone supplies 'individuals' as a value, and the system sets this to plural: the result would be 'individualses', a rather Gollum-like enunciation.

It is likely that we will be unable to guarantee language that is always correct in terms of syntax and semantics. However, as stated earlier, every property in the ontology has its own folksonomy, meaning that when a user selects a tag, it was frequently used before on this property, with this particular textual representation. We imagine that users are more likely to use those tags that fit into the text in a conventional, grammatical manner (e.g. they may steer away from tags resulting in 'This document's observation units were individualses'). If, however, they use tags that result in grammar errors or unclear language,

only the textual representation suffers. Whether the textual representation is entirely grammatical makes no difference to the underlying metadata.

4.2 Extending the Ontology

As we have discussed, it is desirable for the user to be able to use ontologies and create metadata, without being restricted by them. The use of folksonomies to guide (but not dictate) the choice of values helps, but is not the entire answer. The user has to be able to extend the underlying ontology, so he can add information that the creators of the ontology had not anticipated. Therefore, in our tool we are implementing facilities to add new classes and properties to the underlying ontology. The user has to provide standard ontology features such as placing classes in the class hierarchy and specifying which properties apply to them, and specifying domain and range of properties. There are many tools (e.g. Protégé[7]) that provide such functionality; however, we aim for something simpler, for ontology-naive users. Users can only specify the above plus cardinality restrictions, and whether properties are functional; the language constructs of OWL-lite.

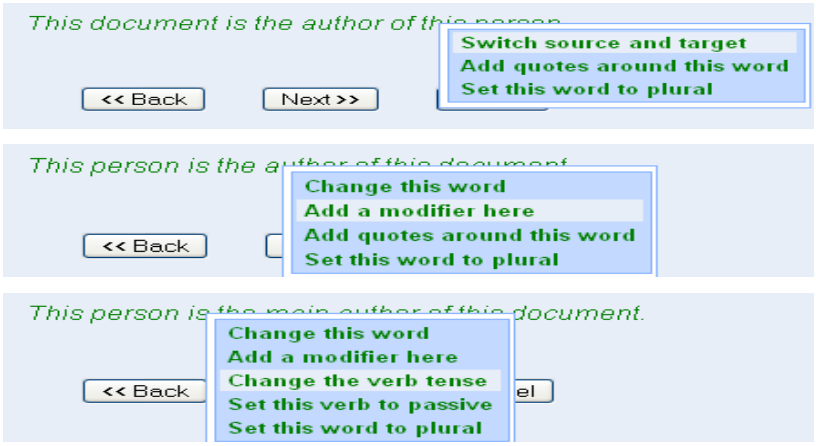

Fig. 7. Creating linguistic information for the property *HasMainAuthor*.

Linguistic Implications As stated above, for natural language access to be possible each resource in an ontology needs a lexicon entry. It is straightforward for the user to provide a noun phrase for a class, but a full dependency tree is a different matter. We cannot require users to have the necessary linguistic expertise to create such a structure; neither can the system generate these automatically because of all the inherent quirks of natural language (whether 'This

[7] http://protege.stanford.edu/

document was authored by this person' is valid is not a question that can be answered by looking at its syntax). Fortunately, if the system has the linguistic knowledge, and the user knows what the textual representation should be, they should be able to generate a lexicon entry together. Our system has a number of template dependency trees, corresponding to common sentence types with known syntactic structure. When the user creates a new property, the system inserts the semantic root of the property name into each template and generates the corresponding sentence. The user then selects the one closest to what he has in mind, and fine-tunes it by manipulating individual words, verb tense, adding or removing modifiers, in ways that are guaranteed to be syntactically legal. An example is shown in Figure 7, where the user is fine-tuning the natural language representation for *HasMainAuthor*. The template selected by the user, one of those generated by the system, is 'This document is the author of this person.' The user first switches the locations of the source and the target in the sentence, then adds a modifier ('main') in front of 'author', and finally changes the tense of the main verb to past tense. The final sentence is thus: 'This person was the main author of this document.' When the user is satisfied the system stores the corresponding lexicon entry for future use.

5 Future Work

In Section 1, we stated that an important challenge facing us was stimulating collaboration in and between communities. We believe that the use of folksonomies will lead to the emergence of community sets of tags, which should make it easier to locate resources. We anticipate that the same approach will also help in bridging between communities, especially if we allow collective tagging, i.e. enabling users to add to the description of resources deposited by others. Users should be able to annotate resources belonging to other communities, thereby stimulating collaboration. This mechanism does of course raise issues of fairness and safety. Can one user render another's data useless by giving it spurious annotations? How can this be prevented? How are different opinions fairly represented? These are as yet open questions.

We are now designing tools for querying and browsing metadata, building upon our experience of developing a metadata elicitation tool. One can imagine a researcher formulating a query by describing the resource he is looking for, e.g. 'The authors of this document include Hielkema, Mellish and Edwards.' could initiate a search for this paper. When presenting information, anchors in the feedback text would signal points where the user could click to see additional, more detailed or related information. This would enable the user to browse through a 'community' of related papers and data.

We are about to commence two user studies to evaluate the metadata elicitation tool. In the first, our purpose is to measure the tool's usability and its usefulness compared to writing natural language. Participants have three main tasks: (i) describe one of their own datasets using a text editor; (ii) after a short demonstration of the tool, each user is given the same four resource descriptions

(in randomised order) and is asked to create similar descriptions using the tool. By comparing the time and the number of operations it takes participants to complete each description, and the percentage of correct descriptions, we hope to prove our hypothesis that users will become proficient after only a few attempts; (iii) users take the dataset they described in task *i*, and describe it again using our tool. Using the differences between the two descriptions, and user feedback, we want to determine whether the tool was limiting what they could express, and conversely, if it ever inspired them to add additional information. The second experiment focuses on the creation of new ontology resources and their lexicon entries. Each user is given three class and three property names, and asked to write down a suitable natural language representation. They are then asked to create these resources and their lexicon entries with the tool. Again we want to measure usability by comparing completion time and the number of operations used. We can measure how many users succeeded in producing the linguistic representation they wanted; we will ask them to report difficulties, whether the natural language they created was satisfactory, and whether they think this open-ended functionality is worthwhile.

6 Conclusion

We have discussed four challenges which have to be addressed to access collaborative metadata using natural language: what type of interface to use, how to associate linguistic information with ontologies, how to achieve open-endedness, and how to stimulate collaboration. We believe that WYSIWYM, a natural language generation technique is the best approach, to avoid the problems inherent in parsing. Evaluation of another WYSIWYM-tool[15] has shown that this method is quickly mastered by novices, as it requires no experience with formal languages but instead relies on the user's natural language capabilities. We believe that the same approach should be suitable for query building and information browsing. We have developed a *Lexicon* ontology that stores the linguistic information needed to generate language from ontologies. We propose the creation of a new annotation value, e.g. *rdfs:LinguisticInformation*, which would associate resources with their lexicon entry. Open-endedness is achieved by integrating ontologies with folksonomies, and enabling users to extend the underlying ontologies by creating new properties and classes. The fourth challenge, how to stimulate collaboration, is still largely unaddressed; we hope to tackle the issues associated with this in future.

References

1. De Roure, D., Jennings, N., Shadbolt, N.: The Semantic Grid: Past, Present and Future. In: Proceedings of the IEEE 93(3). (2005) 669–681
2. Edwards, P., Chorley, A., Pignotti, E., Hielkema, F.: Using the Semantic Grid to Support Evidence-Based Policy Assessment. In Chen, H., Wang, Y., Cheung, K., Studer, R., eds.: Semantic eScience (Papers from the 2007 AAAI Workshop), AAAI Technical Report WS-07-12, AAAI Press, 2007, to appear. (2007)

3. Edwards, P., Aldridge, J., Clarke, K.: A Tree Full of Leaves: Description Logic and Data Documentation. In: Proceedings of the Second International Conference on e-Social Science, Manchester, UK (2006)
4. Guy, M., Tonkin, E.: Folksonomies: Tidying up Tags? D-Lib Magazine **12**(1) (2006)
5. Gruber, T.: Ontology of Folksonomy: A Mash-up of Apples and Oranges. http://tomgruber.org/writing/ontology-of-folksonomy.htm (2005)
6. Handschuh, S., Staab, S., Maedche, A.: 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, New York, NY, USA, ACM Press (2001) 76–83
7. Catarci, T., Dongilli, P., Mascio, T.D., Franconi, E., Santucci, G., Tessaris, S.: An Ontology-based Visual Tool for Query formulation Support. In: Proceedings of the Sixteenth European Conference on Artificial Intelligence (ECAI 2004). (2004)
8. Petre, M.: Why Looking isn't always Seeing: Readership Skills and Graphical Programming. Communications of the ACM **38**(6) (1995) 33–44
9. Bernstein, A., Kaufmann, E.: Gino - a guided input natural language ontology editor. In: International Semantic Web Conference 2006. (2006) 144–157
10. Tennant, H., Ross, K., Saenz, R., C.W.Thompson, Miller, J.: Menu-based Natural Language Understanding. In: Proceedings of the Twenty-first Annual Meetings on Association for Computational Linguistics, Cambridge, Massachusetts (1983) 151–158
11. Schwitter, R., Tilbrook, M.: Controlled natural language meets the semantic web. In: Proceedings of the Australasian Language Technology Workshop 2004. (2004)
12. Wilcock, G.: Talking OWLs: Towards an Ontology Verbalizer. In: Human Language Technology for the Semantic Web and Web Services (ISWC'03), Sanibel Island, Florida (2003) 109–112
13. Bontcheva, K.: Generating tailored textual summaries from ontologies. In: ESWC. (2005) 531–545
14. Power, R., Scott, D., Evans, R.: 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 (1998)
15. Hallett, C., Scott, D., Power, R.: Composing Questions through Conceptual Authoring. Computational Linguistics **33**(1) (2007) 105–133
16. Bouayad-Agha, N., Power, R., Scott, D., Belz, A.: PILLS: Multilingual Generation of Medical Information Documents with Overlapping Documents. In: Proceedings of the Third International Conference on Language Resources and Evaluation (LREC 2002), Las Palmas (2002) 2111–2114
17. Mel'cuk, I.: Dependency Syntax: Theory and Practice. State University of New York (1988)
18. Al-Khalifa, H., Davis, H.: FolksAnnotation: A Semantic Metadata Tool for Annotating Learning Resources Using Folksonomies and Domain Ontologies. In: Proceedings of the Second International IEEE Conference on Innovations in Information Technology, Dubai, UAE (2006)