Evaluating an Ontology-Driven WYSIWYM Interface

Feikje Hielkema **Chris Mellish** **Peter Edwards**
Computing Science
School of Natural & Computing Sciences
University of Aberdeen
Aberdeen, AB24 3FX, UK
{f.hielkema, c.mellish, p.edwards}@abdn.ac.uk

Abstract

This paper describes an evaluation study of an ontology-driven WYSIWYM interface for metadata creation. Although the results are encouraging, they are not as positive as those of a similar tool developed for the medical domain. We believe this may be due, not to the WYSIWYM interface, but to the complexity of the underlying ontologies and the fact that subjects were unfamiliar with them. We discuss the ways in which ontology development might be influenced by issues stemming from using an NLG approach for user access to data, and the effect these factors have on general usability.

1 Introduction

In the PolicyGrid[1] project we are investigating how best to support social science researchers through the use of Semantic Grid (De Roure et al., 2005) 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, e.g. through the use of RDF[2] (Resource Description Framework, a metadata model for making statements about resources) or OWL[3] (knowledge representation language for authoring ontologies).

Numerous e-science applications rely on metadata descriptions of resources. But how does metadata come into existence? Ideally the user should create it. However, metadata creation is a complex task, and few users know how to create them in RDF. To enable our users to describe their resources, we need to provide a tool that facilitates creation, querying and browsing of metadata by users with no prior experience of such technologies.

Existing tools that provide access to RDF metadata are often graphical, e.g. (Handschuh et al., 2001; Catarci et al., 2004). However, 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. Natural language approaches include GINO (Bernstein and Kaufmann, 2006), an ontology editor with an approach reminiscent of Natural Language Menus (Tennant et al., 1983), and using Controlled languages such as PENG-D (Schwitter and Tilbrook, 2004). Such natural language approaches tend to restrict expressivity to ensure that every entry can be parsed, limiting the language and often making it stilted, so that there is a small learning curve before the user knows which structures are allowed. 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 feed-

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

[2]http://www.w3.org/RDF/

[3]http://www.w3.org/TR/owl-features/

back text for the user that is based on a semantic representation. This representation is edited directly by the user by manipulating the feedback text. WYSIWYM has been used by a number of other projects, such as MILE (Piwek et al., 2000) and CLEF (Hallett, 2006). As evaluation results in both of these projects were very positive (Piwek, 2002; Hallett et al., 2007), we felt that WYSIWYM would be a suitable approach to use in our work.

We have developed a metadata elicitation tool that enables users to create metadata in the shape of *ontology instance data*; the tool is driven by the ontologies that define those instances. We are currently implementing a WYSIWYM tool for querying, that uses the same interface as the metadata creation tool. We also aim to develop a tool for presenting the results of the query, and for browsing the descriptions in the database. These three tools will be integrated into one consistent interface, so that users can switch effortlessly between querying, browsing and editing ontology instance data. This aim is similar to the support that the graphical tool SHAKEN provides for ontology editing and browsing (Thoméré et al., 2002). We want to ensure that these tools are generic, so that if the ontologies change over time or are replaced, the tools will still function. That means that all domain specific information (as much as is possible) should be contained in the ontologies. In this paper we explore the ways in which Natural Language Generation issues influence ontology building and vice versa.

This paper is structured as follows: section 2 describes the tool for metadata creation that we have implemented; section 3 discusses issues in ontology development and Natural Language Generation; and section 4 presents an evaluation study of the metadata creation tool. In section 5 the results of this study are discussed and compared to those of the CLEF project; we argue that different domains and ontologies affect the usability and complexity of metadata access interfaces.

2 The Metadata Creation Tool

We have developed a WYSIWYM tool that enables users to upload resources (e.g. acadamic papers, statistical datasets, interview transcripts) and create metadata descriptions for them, even if these users

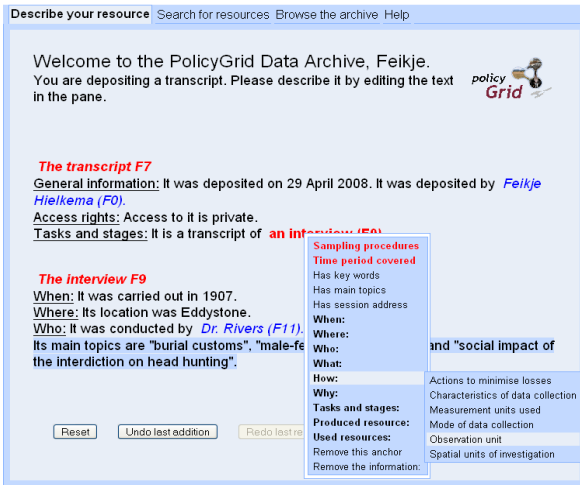

Figure 1: The Metadata Creation Tool.

are unfamiliar with ontologies. First, the user selects the type of resource he is depositing (e.g. a *Transcript*). The tool then generates a brief *feedback text* that presents the information specified by the user. The feedback text contains *anchors*, phrases in red boldface and blue italics that signal where new information can be added. When the user clicks on an anchor, a menu pops up listing the kinds of information that can be added here (see Figure 1). After selecting a menu item, the user is prompted to enter an appropriate value; this may be a date, a free-text string, or another object that may or may not be in the text already. The feedback text is regenerated whenever the user has added some information.

The tool is driven by one or more ontologies. Their class hierarchies are presented when users are selecting a resource type, or creating a new object as range for a property. The anchors correspond to individuals in the ontology; the menu items to the properties of those individuals. The feedback text is divided into paragraphs which correspond to the individuals; each property of an individual is realised as (part of) a sentence in its paragraph. Each property in the ontology is associated with a linguistic specification, a Dependency Tree (Mel'cuk, 1988) that corresponds to a sentence. The specification has slots where the source and target of the property should be inserted, and is sufficiently detailed to support processes such as aggregation, through which the feedback text is made more fluent. For a more extensive description of the metadata cre-

ation tool and its implementation, see Hielkema et al. (2007b).

In August 2007 we ran a pilot evaluation study (Hielkema et al., 2007a) on this tool. This study was heuristic in nature, with subjects discussing the interface with the experimenter while performing set tasks. It highlighted a number of aspects which we felt it was necessary to improve before embarking on the formal evaluation. Apart from there being standard usability considerations such as a need for better undo and help functions, it became evident that the underlying ontology was neither extensive enough nor sufficiently well-structured: subjects struggled to find the options they needed, and were often not satisfied with the options' names or their location in the sub-menus. We therefore decided that, as well as improving the basic usability of the interface, we needed to redevelop the ontology that was driving the interface. Users, we felt, would find it easier to navigate the menus when this ontology matched their mental model of the domain. Throughout the development of this new ontology, user requirements and feedback were gathered through a number of focus group sessions. The next section describes the ways in which this ontology development was affected by the demands of the metadata interface.

3 Ontologies in NLG

Portability has always been a major issue in NLG. Language generation involves the use of much information that is domain-specific, and cannot be generalised without a cost in the expressivity of the resulting text. If we want to create an application that is domain-independent, we have to find a way to store all domain-specific information in a structure that is easily extended or replaced.

We have decided to use an ontology, a common structure whose use has become widespread in knowledge representation. Ideally, we would like to create a generator that can be applied to any domain, provided there is an appropriate domain-specific ontology. But what information should such an ontology contain? How should it be structured? In this section we explore issues that occur when developing or adapting ontologies for use in the WYSIWYM tool; we believe that this can at least in part be generalised to NLG. The ontologies we have used so far were developed at the same time as the WYSIWYM tool, so that both tool and ontology influenced each other's development. We are currently adapting an ontology from another e-science project for use in our WYSIWYM interface, to further investigate such issues (see section 5).

There are a number of existing tools that generate language from ontologies, using various approaches. Wilcock (2003) describes an ontology verbaliser using XML-based generation. As Wilcock states, his approach is domain-specific, and therefore probably incompatible with more general ontologies (and presumably with ontologies from a different domain).

MIAKT (Bontcheva and Wills, 2004) is a system that generates textual medical reports from an RDF description. It uses a medical domain ontology and an NLG lexicon that contains lexicalisations for the concepts and instances in the ontology. In order to verbalise properties, MIAKT's surface realiser needs lexical specifications for them. Four basic property types are distinguished whose sub-properties can mostly be realised automatically through the grammar rules in the realiser. This technique increases the portability of the system, but does affect the variability and expressivity of the generator.

We do not aim to generate from any ontology in a domain, but to generate texts with high expressivity and clarity from ontologies that are designed in an 'NLG-aware' way. We are investigating what requirements an ontology has to meet in order to be usable for our application, so that for any domain an ontology can be built or adapted which we can use to produce a usable NL-interface. As many ontology developers are not linguists, ideally we want to support the adaptation to 'NLG-aware' ontologies without requiring linguistic expertise, for instance through a supporting software tool. Ontologies are primarily built to model domain-specific knowledge, making domain assumptions explicit, and to facilitate reasoning with this knowledge. These aims may sometimes conflict with the requirements of NLG applications, but they do frequently coincide (e.g. the need for clear, unambiguous resource names.

3.1 Domain Ontologies for WYSIWYM

What information does our WYSIWYM application need its ontologies to provide? First of all, the parts of it that will be shown to the user need to be easily mapped to natural language. The purpose of the tool is to support creation of ontology instance data by users unfamiliar with ontologies, so the parts they see should be comprehensible to novices. The names of properties are used to populate the pop-up menus, while the class names are shown in the class hierarchy. These names are mapped to natural language by replacing capitals and underscores with whitespace, and if necessary adding a determiner. Therefore, they need to correspond to phrases in natural language in order to be understood by the user, with individual words separated by capitals or underscores. If there is no intuitive NL-phrase to represent a class, it probably does not correspond to a concept in the domain either and might confuse the user, so it should be removed from the hierarchy. Classes whose instances are best presented by some distinctive name (e.g. *Person* or *Paper*) should have a *name* or *title* property whose value can be used (e.g. 'John'). For other classes (e.g. *Interview*), the class name can be used (e.g. 'some interview').

We need a linguistic specification for each property, sufficiently detailed to support aggregation and pronominalisation, but also to produce more than one surface form: a query is presented differently than a description, even if it contains the same information (compare the texts in Figure 1 and 2). The linguistic specification should be sufficiently rich to support the generation of these different surface forms. For this purpose we are using Dependency Trees, whose richness in both syntactic and semantic information provides ample support for such transformations. These trees can be associated with the domain ontology[4]. This specification also contains the header of the submenu in which the property should appear.

Some peculiarities in natural language are domain-independent. For instance, an address is presented in a very specific way and cannot be realised in the standard manner without sacrificing

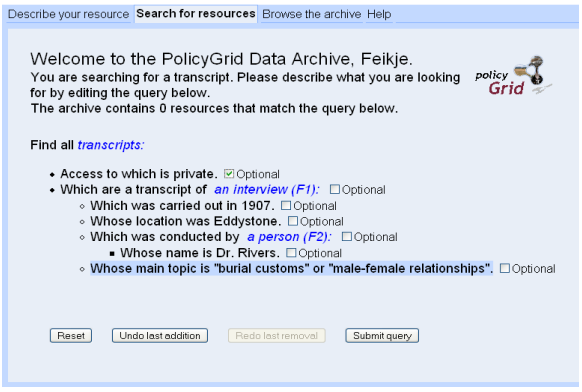

Figure 2: The Query Tool.

clarity (e.g. 'The address' street is Union Street. Its place is Aberdeen'). Such 'utility' classes are used across domains. In PolicyGrid we have created a utility ontology that contains classes such as 'Person', 'Address' and 'Date' [5]. Instances of these classes are generated to a special surface form. In order to get the best realisation from the WYSIWYM tool, domain ontologies should use the classes from this utility ontology. As the properties of the utility classes are already furnished with linguistic specifications, they are already NLG-aware. Another way to hasten the process is to use, where possible, properties from this ontology instead of those from the domain ontology.

3.2 WYSIWYM for Ontologies

What should the WYSIWYM application do in order to provide access to ontologies? For metadata creation it is essential that users can only produce 'correct' metadata, which does not violate the constraints in the ontology. The feedback text should be presented coherently, while the Text Planner only uses information that is either domain independent or present in the ontology. Perhaps most importantly, the application should support easy creation of the linguistic information that the ontology must contain, as we cannot expect ontology developers to have the linguistic expertise to create Dependency Trees. We are devising a way for users to create a specification by manipulating the surface form of a 'template' specification. We currently have 12 templates which represent commonly used sen-

[4]For an example of how this is done, see http://www.csd.abdn.ac.uk/research/policygrid/ontologies/Lexicon/Lexicon.owl

[5]http://www.policygrid.org/utility.owl

tences to present ontology properties in text. The user can fine-tune the surface form by adding adjectives, changing morphological information and the root of individual words; actions for which only a basic linguistic knowledge is needed. This approach is outlined in more detail in (Hielkema et al., 2007b). The main challenge with this approach is that the specification is used to generate two surface forms; it remains to be seen whether a specification that is fine-tuned through one surface form will accommodate the accurate generation of another.

The Penman Upper Model (Bateman, 1990) supports the specification of linguistic information through a different approach. The Upper Model is a domain-independent ontology that supports sophisticated NLP. To make a domain ontology available for NLP, its resources have to be placed in the hierarchy of the Upper Model; their place there determines their surface realisation. This task appears to require considerable linguistic expertise, but like the creation of our Dependency Trees could probably be made easier for non-linguists through some special-purpose interface.

4 Usability Evaluation

The best evaluation of our tool would be to let users deposit their resources in real-life contexts, but our tool is not ready for a full deployment. Another way would be to compare its usability to another metadata creation tool in an experiment where users completed the same tasks with both tools. Unfortunately, most metadata tools focus on providing support for ontology editing (e.g. Protégé[6] or GINO (Bernstein and Kaufmann, 2006)), or query formulation (e.g. SEWASIE (Catarci et al., 2004)). A number of tools for metadata creation use formal (RDF) or controlled languages, which are difficult to use for those wholly unfamiliar with formal logic. Other tools were developed for one specific purpose, e.g. CREAM (Handschuh et al., 2001) which was developed for the annotation of web pages, and could not easily be adapted to our purposes. We were not aware of any tool that we could adapt to the e-social science ontologies and thus use in an experiment.

Alternatively, we could have compared our interface to direct authoring of RDF; but in an environment where most users have no experience of ontologies or metadata this seemed spurious. Instead, we adopted an approach similar to that used in the CLEF project (Hallett et al., 2007). They evaluated their WYSIWYM system (which enabled users to create SQL queries for a database in a medical domain) by measuring the performance of fifteen subjects on four consecutive tasks, after a brief introduction. These subjects were all knowledgeable in the domain, and all but two knew the representation language of the repository and how the data contained in it was structured. These subjects achieved perfect results from the second task onwards, and became faster with each task, especially after the first. We also expected users to become faster and more accurate with each completed task, and indeed hoped for perfect scores on their last task.

Subjects Sixteen researchers and PhD students from various social science-related disciplines participated in the experiment. None of them had prior experience of the metadata elicitation interface, and only two of the subjects had any previous experience of using ontologies. The ontology driving the system models the description of social science resources and was based on requirements gathering sessions, in which a few subjects had participated. None of the subjects knew its precise structure.

Methodology After providing some information about their background, subjects viewed a video introduction[7] of six minutes. This video showed the construction of a simple resource description, highlighting the main functionalities of the interface, while a voice-over explained what was happening on the screen.

Subjects were then handed four short resource descriptions expressed as paragraphs of English (see 'Materials') and asked to reproduce these descriptions as closely as possible using the tool. To avoid making the choice of the correct options too obvious, we tried to avoid phrases that corresponded literally to those in the menus. Each subject received the descriptions in a different order, in case there were differences in the complexity of the tasks. Subjects were allowed as much time as they needed to

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

[7] This video can be viewed at http://www.csd.abdn.ac.uk/ research/policygrid/demos/WysiwymIntroduction1.mov

Task order	Completion time		Operations		Total errors		Avoidable errors	
	μ	σ	μ	σ	μ	σ	μ	σ
First	512.81	366.132	48.38	24.527	3.31	1.922	1.56	.727
Second	329.50	95.716	37.75	12.228	2.69	2.442	1.38	.957
Third	260.06	90.542	35.13	9.749	2.75	2.720	1.63	1.310
Fourth	309.81	106.049	39.38	10.844	2.00	1.966	1.44	1.504

Table 1: Mean completion times, operations and errors per completed task.

complete each task.

For each task, the tool recorded the completion time, the produced description, the number of operations used to produce it, and the frequency with which various operation types were used, such as 'undo' or the 'help' functions. After the subjects had completed all four tasks, they were asked to rate the usability (very difficult - difficult - OK - easy - very easy) and usefulness (useless - not much use - adequate - useful - very useful) of the tool on a five-point Likert scale, and to note any feedback they might have. The entire experiment took on average 50 min. per subject.

Materials We used four resource descriptions, one of which was:

> You are depositing the transcript of an interview that was held by Dr. Rivers in 1907, at Eddystone. The interview mainly discussed 'male-female relationships', 'burial practices' and 'the social impact of the interdiction on head hunting'. Access to this transcript should remain private.

Figure 1 shows the corresponding description that could be produced with the tool. The separation of the transcript from the interview is an example of the clear distinctions necessary for knowledge representation. In natural language, this distinction would not necessarily be made, and indeed this step was missed by a number of users.

To ensure that tasks did not repeat identical subtasks, we tried to use different parts of the ontology in each task. Every task described a different resource type (conference paper, transcript, academic paper, report), which corresponded to a different class in the ontology. We were also careful to choose varying menu items (corresponding to properties in the ontology), although some repetition was unavoidable (e.g. specifying names). In fact, a real-life use of the tool would involve rather more task repetition (specifying titles, authors and dates would be necessary for practically any resource) than the artificial descriptions in this study.

Results To analyse the accuracy of the produced descriptions, we divided each description task into 8 to 10 subtasks. For the task shown in the previous paragraph, these subtasks were:

- Specify that you are depositing a 'Transcript'
- Specify that access is private
- Specify that it is a transcript of an 'Interview' (creating an 'interview' object)
  - Specify the three main topics
  - Add an interviewer (creating a 'Person' object)
    - Call this person 'Dr. Rivers'
  - Specify the location of the interview
  - Specify the date of the interview

As some subtasks are more complicated than others and take longer, we did not try to give each task exactly the same number of subtasks, but instead ensured that all tasks needed the same number of operations (e.g. menu item selections, button clicks, etc.) in order to be completed. Each subtask that was missing or completed differently than in the description shown in 'Materials' was counted as one error. Erroneous ways to complete subtasks included choosing a different menu item and adding information to the wrong object. For instance, a number of subjects, instead of specifying an interviewer for the interview, added a creator for the transcript; this was

counted as one erroneously completed subtask, and therefore one error.

The list of subtasks above shows that some subtasks depend on the successful completion of other tasks; for instance, you cannot add an interviewer unless you have created an 'interview' object. We therefore analysed two error counts: the total number of errors, and the 'avoidable' errors. The 'avoidable' errors were the total number of errors minus those subtasks that depended on another subtask that was missing or had been completed incorrectly.

We analysed the mean completion times, number of operations used and the two error counts of the tasks that were completed first, second, third and last, using a repeated measures ANOVA (see Table 1 for the means and standard deviations). Mean completion times went down significantly (Huynh-Feldt p-value < 0.01). Tukey's HSD post-hoc (applied to a univariate ANOVA, with task order as the independent variable) test shows that both the third (p-value < 0.01) and the fourth (p-value 0.030) were completed significantly faster than the first task. However, no significant differences were found for the number of operations (Huynh-Feldt p-value 0.062), the total number of errors (Huynh-Feldt p-value .322) or the number of avoidable errors (Huynh-Feldt p-value .931).

Subject feedback on the tool was positive: it was perceived as useful (μ 3.94; 1='useless', 5='very useful'), and OK or easy to use (μ 2.69; 1='very easy', 5='very difficult'). Five subjects expressed a preference for a form-based interface, and five others for a NL-interface such as the one tested. In feedback, subjects indicated a desire for more form-based elements in the interface, to speed up the creation of the standard description elements (e.g. name/title, author), and complained that the environment was initially unfamiliar, with some menu items overlapping. This unfamiliarity meant that items that were necessary to complete the description were often overlooked; subjects often solved this by choosing the closest approximation they could find, e.g. 'creator' instead of 'interviewer'.

5 Discussion and Future Work

Although users quickly gained speed using the tool, and were positive in their feedback, the evaluation results are not nearly as positive as those found for CLEF (see section 4). The mean number of errors decreased, but this effect was not significant and only five out of sixteen subjects received a perfect score on the last task (four other subjects performed some earlier task(s) perfectly). Evidently there is a difference in usability of both tools - but what causes it? No doubt the difference can partly be ascribed to differences in the implementation of the interface. However, the most common feedback we received from the subjects was that they were overwhelmed by the large number of options available to them. Each class in the social science ontology has on average 30 properties, which means a description with three objects provides 90 options. In contrast, the number of available options in the CLEF system was deliberately kept small (max. three) for 'non-terminal anchors'. Especially in the first task, users had trouble finding the option they wanted, and although it became easier in the later tasks as they familiarised themselves with the system, the results indicate that it remained a problem. This was likely aggravated by our deliberate avoidance of subtask-repetition; more standard descriptions, which always involve titles and authors, might have produced a greater learning effect. CLEF was developed for a medical domain, which is well defined and understood by the experimental subjects. The social science domain encompasses many different theories and concepts, not just about what subjects are investigated, but also about how the research should be conducted. PolicyGrid has tried to develop an ontology that the different disciplines in social science could be satisfied with. As a result, it is quite large and complex, and most users will only recognise parts of it. Thus the number of available options in the tool driven by this ontology is large, and users have to explore the ontology and learn to navigate it where their domain knowledge does not suffice. This flattens the learning curve and decreases the usability of the tool.

Half the users preferred a form-based interface over an NLG interface. Although forms are an easily understood mechanism which are just as familiar to users as natural language, we have three reasons for preferring the WYSIWYM approach. First, the large number of options in the ontology means that a form would reach truly daunting proportions. Sec-

ond, we want our resource descriptions to be connected through shared people, projects, institutions, etc; using the expressivity that RDF offers us. This would be more difficult to achieve in an interface where the user completes a form by providing each property with a free-text description. Thirdly, forms can be confusing for the user as well; the brief descriptions provided for each element are frequently ambiguous and therefore misunderstood. An NLG-interface, which provides feedback by presenting the property in a complete sentence, should help to clarify the meaning of the property name for the user.

As we discussed earlier, there are many constraints on the development of domain ontologies that can be accessed through NLG, and the evaluation indicates that the structure of the ontology is essential for the tool's usability. Still, the evaluation is sufficiently positive that we believe the WYSIWYM approach suitable for providing access to ontologies, especially for users who are unfamiliar with ontologies or their graphical representations. Navigation could be made easier by providing users with an overview of the underlying ontology, possibly presented as an index of objects, and the information that can be specified about each object. An online manual with some worked examples and screenshots might also help users get started on the more obvious parts of a description. We are currently attempting to adapt an ontology developed in another UK e-science project[8] for use in the WYSIWYM metadata elication tool. Instead of assuming the depositing of a resource, this ontology was developed to capture user-elicited metadata for video annotation. Part of this metadata is captured automatically, part of it is elicited from the user. We hope that the adaptation of an ontology that was originally developed for a different purpose for use in an NLG application will highlight other issues involved in the use of ontologies in NLG.

One way in which subjects did tasks erroneously was by using the 'hasComment' property when they could not find the option they wanted. This is not precisely *wrong*: the metadata it produces is correct and any human readers will understand the description. But it is not the best description for querying purposes. We think some subjects may have had trouble grasping the exact purpose of the produced descriptions. We hope that users who have used the query tool to find (the descriptions of) resources, will have a better understanding of what an effective metadata description is.

We intend to run more evaluation experiments, to assess the usability but also the usefulness of the combined toolset. Rather than asking subjects to copy descriptions or queries, we may ask them to find a particular resource, or to try to deposit and describe one of their own papers. If possible, it would also be interesting to see how they perform the same tasks using a different interface for metadata access, e.g. a graphical interface such as SHAKEN (Thoméré et al., 2002).

6 Conclusion

We have presented a WYSIWYM interface for the creation of RDF metadata, which will be extended by the addition of querying and browsing tools. This tool is driven by an ontology that contains all domain-specific information needed to present it in natural language. We have highlighted a number of issues in ontology development for access through NLG. We have evaluated the tool's usability through an experiment with potential users. The results were encouraging, but indicate that the structure and familiarity of the underlying ontology strongly influence the usability of the interface.

Acknowledgments

Many thanks to Catalina Hallett and Richard Power from the CLEF project, for their help in comparing the two tools and their different evaluation results.

References

J.A. Bateman. 1990. Upper Modelling: A General Organisation of Knowledge for Natural Language Processing. In *Proceedings of the International Language Generation Workshop*, Pittsburgh, USA.

A. Bernstein and E. Kaufmann. 2006. GINO - A Guided Input Natural Language Ontology Editor. In *International Semantic Web Conference 2006*, pages 144–157.

K. Bontcheva and Y. Wills. 2004. Automatic Report Generation from Ontologies: the MIAKT approach. In *Nineth International Conference on Appli-*

[8] http://www.ncess.ac.uk/research/digital_records/

cations of Natural Language to Information Systems (NLDB'2004), Manchester, UK.

T. Catarci, P. Dongilli, T. Di Mascio, E. Franconi, G. Santucci, and S. Tessaris. 2004. An Ontology-based Visual Tool for Query Formulation Support. In *Proceedings of the Sixteenth European Conference on Artificial Intelligence (ECAI 2004)*.

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.

C. Hallett, D. Scott, and R. Power. 2007. Composing Questions through Conceptual Authoring. *Computational Linguistics*, 33(1):105–133.

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.

F. Hielkema, P. Edwards, C. Mellish, and J. Farrington. 2007a. A Flexible Interface to Community-Driven Metadata. In *Proceedings of the Third International Conference on eSocial Science*.

F. Hielkema, C. Mellish, and P. Edwards. 2007b. Using WYSIWYM to Create an Open-ended Interface for the Semantic Grid. In S. Busemann, editor, *Proceedings of the 11th European Workshop on Natural Language Generation*.

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

P. Piwek, R. Evans, L. Cahil, and N. Tipper. 2000. Natural Language Generation in the MILE System. In *Proceedings of IMPACTS in NLG workshop*, pages 33–42, Schloss Dagstuhl, Germany.

P. Piwek. 2002. Requirements Definition, Validation, Verification and Evaluation of the CLIME Interface and Language Processing Technology. Technical Report ITRI-02-03, ITRI, University of Brighton.

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.

J. Thoméré, K. Barker, V. Chaudhri, P. Clark, M. Eriksen, S. Mishra, B. Porter, and A. Rodriguez. 2002. A Web-based Ontology Browsing and Editing System. In *Eighteenth National Conference on Artificial Intelligence*, pages 927–934, Menlo Park, CA, USA. American Association for Artificial Intelligence.

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.