ONKI-SKOS – Publishing and Utilizing Thesauri in the Semantic Web

Jouni Tuominen, Matias Frosterus, Kim Viljanen and Eero Hyvönen
*Semantic Computing Research Group (SeCo)
Helsinki University of Technology and University of Helsinki
P.O. Box 5500, 02015 TKK, Finland
first.last@tkk.fi, http://www.seco.tkk.fi

Abstract

Thesauri and other controlled vocabularies act as building blocks of the Semantic Web by providing shared terminology for facilitating information retrieval, data exchange and integration. Representation and publishing methods are needed for utilizing thesauri efficiently, e.g., in content indexing and searching. W3C has provided the Simple Knowledge Organization System (SKOS) data model for expressing concept schemes, such as thesauri. A standard representation format for thesauri eliminates the need for implementing thesaurus specific rules or applications for processing them. However, there do not exist general tools which provide out of the box support for publishing and utilizing SKOS vocabularies in applications, without needing to implement application specific user interfaces for end users. For solving this problem the ONKI-SKOS server is presented.

1 Introduction

Thesauri and other controlled vocabularies are used primarily for improving information retrieval. This is accomplished by using concepts or terms of a thesaurus in content indexing, content searching or in both of them, thus simplifying the matching of query terms and the indexed resources (e.g. documents) compared to using natural language (Aitchison et al., 2000). For users, such as content indexers and searchers, to be able to use thesauri, publishing and finding methods for thesauri are needed (Hyvönen et al., 2008). Thesauri are of great benefit for the Semantic Web, enabling semantically disambiguated data exchange and integration of data from different sources, though not in the same extent as ontologies.

Publishing and utilizing thesauri is a laborous task because representation formats of thesauri and features they provide differ from each other. When utilizing thesauri one has to be familiar with how to locate a given thesaurus and how to use the software the thesaurus is published with. A thesaurus can even be published as a plain text file or even worse, as a paper document, with no proper support for utilizing it. In such a case the users have to implement applications for processing the thesaurus in order to exploit it. Therefore, standard ways for expressing and publishing thesauri would greatly facilitate the publishing and utilizing processes of thesauri.

W3C has proposed a data model for expressing concept schemes (e.g. thesauri), the Simple Knowledge Organization System (SKOS)[1] (Miles et al., 2005), providing a standard way for creating vocabularies and migrating existing vocabularies to the Semantic Web. SKOS solves the problem of diverse, non-interoperable thesaurus representation formats by offering a standard convention for presentation. For expressing existing thesauri in SKOS format conversion methods are needed. When a thesaurus is expressed as a SKOS vocabulary, it can be published as a RDF file on the web, allowing the vocabulary users to fetch the files and process them in a uniform way. However, this does not solve the problem of users having to implement their own applications for processing vocabularies.

For publishing ontologies and vocabularies on the Semantic Web, ontology servers have been proposed in the research community (Ding and Fensel, 2001; Ahmad and Colomb, 2007). Ontology servers are used for managing ontologies and offering users access to them. For accessing SKOS vocabularies, there are some Web Service implementations, namely the SKOS API[2] developed in the SWAD-Europe project and the terminology service by Tudhope et al.[3]. How-

[1] http://www.w3.org/TR/skos-reference/

[2] http://www.w3.org/2001/sw/Europe/reports/thes/skosapi.html

[3] The API of the service is based on a subset of the SKOS API, with extensions for concept expansion.

ever, general tools for providing out of the box support for utilizing SKOS vocabularies in, e.g., content indexing, without needing to implement application specific user interfaces for end users do not exist. For filling this gap, we present the ONKI-SKOS server for publishing and utilizing thesauri.

2 Presenting thesauri with SKOS

W3C's SKOS data model provides a vocabulary for expressing the basic structure and contents of concept schemes, such as thesauri, classification schemes and taxonomies. The concept schemes are expressed as RDF graphs by using RDFS classes and RDF properties specified in the SKOS specification, thus making thesauri compatible with the Semantic Web. SKOS is capable of representing resources which have considerable resemblance to the influential ISO 2788 thesaurus standard (van Assem et al., 2006).

Although semantically richer RDFS/OWL ontologies enable more extensive ways to perform logical inferencing than SKOS vocabularies, in several cases thesauri represented with SKOS are sufficient. In our opinion, the first and the most obvious benefit of using Semantic Web ontologies/vocabularies in content indexing is their ability to disambiguate concept references in a universal way. This is achieved by using persistent URIs as a identification mechanism. Compared to controlled vocabularies using plain concept labels as identifiers, this is a tremendous advantage. When using concept labels as identifiers, identification problems can be encountered. As a thesaurus evolves, the labels of its concepts may change, and concepts may be splitted or merged. In such cases the labels of concepts are not a permanent identification method, and the references to the concepts may become invalid.

Not only being an identification mechanism, URIs provide means for accessing the concept definitions and thesauri. With proper server configuration URIs can act as URLs, thereby providing users additional information about the concepts[4]. In addition to these general RDF characteristics, SKOS provides a way for expressing relations between concepts suitable for the needs of thesauri, thus providing conceptual context for concepts.

As stated by van Assem et al. (2006), using a common representation model (e.g. SKOS) for thesauri either enables or greatly reduces the cost of (a) sharing thesauri; (b) using different thesauri in conjunction within one application; (c) development of standard software to process them.

3 Accessing thesauri

ONKI-SKOS is an ontology server implementation for publishing and utilizing thesauri and lightweight concept ontologies. It conforms to the general ONKI vision and API (Viljanen et al., 2008), and is thus usable via ONKI ontology services as easily integrable user interface components and Web Services.

The Semantic Web applications typically use ontologies which are either straightforward conversions of well-established thesauri, application-specific vocabularies or semantically richer ontologies, that can be presented and accessed in similar ways as thesauri (van Assem et al., 2004; Hyvönen et al., 2008). Since SKOS defines a suitable model for expressing thesauri, it was chosen as the primary data model supported by the ONKI-SKOS server.

ONKI-SKOS can be used to browse, search and visualize any vocabulary conforming to the SKOS specification and also RDFS/OWL ontologies. ONKI-SKOS does simple reasoning (e.g. transitive closure over class and part-of hierarchies). The implementation has been piloted using various thesauri and ontologies, e.g., Medical Subject Headings MeSH[5], the General Finnish Upper Ontology YSO[6] and Iconclass[7].

When utilizing thesauri represented as SKOS vocabularies and published on the ONKI-SKOS server, several benefits are gained. Firstly, SKOS provides a universal way of expressing thesauri. Thus processing different thesauri can be done in the same way, eliminating the use of thesaurus specific processing rules in applications or separate converters between various formats. Secondly, ONKI-SKOS provides access to all published thesauri in the same way, so one does not have to use thesaurus specific implementations of thesaurus browsers and other tools developed by different parties, which is the predominant way. Also, one of the goals of the ONKI ontology services is that all the essential ontologies/thesauri can be found at the same location, thus eliminating the need to search for other thesaurus sources.

The typical way to use thesaurus specific publishing systems in content indexing and searching is either by using their browser user interface for finding desired concepts and then copying and pasting the

http://hypermedia.research.glam.ac.uk/kos/terminology_services/

[4] http://www.w3.org/TR/swbp-vocab-pub/

[5] http://www.nlm.nih.gov/mesh/meshhome.html

[6] http://www.seco.tkk.fi/ontologies/yso/

[7] http://www.iconclass.nl/

concept label to the used indexing system[8], or by using Web Services for accessing and querying the thesaurus (Tudhope and Binding, 2005). Both methods have some drawbacks. The first method introduces rather uncomfortable task of constant switching between two applications and the clumsy copy-paste procedure. The second method leaves the implementation job of the user interface entirely to the parties utilizing the thesaurus.

While ONKI-SKOS supports both the aforementioned thesauri utilizing methods, in addition, as part of the ONKI ontology services, it provides a lightweight web widget for integrating general thesauri accessing functionalities into HTML based applications on the user interface level. The widget depicted in Figure 1 can be used to search and browse thesauri, fetch URI references and labels of desired concepts and storing them in a concept collector. Similar ideas have been proposed by Hildebrand et al. (2007) for providing search widget for general RDF repositories, and by Vizine-Goetz et al. (2005) for providing widget for accessing thesauri through the side bar of the Internet Explorer web browser.

When the desired concepts have been selected with the ONKI Widget they can be stored into, e.g., the database of the application by using an HTML form. Either the URIs or the labels of the concepts can be transferred into the application, thus support for the Semantic Web and legacy applications is provided. For browsing the context of concepts in thesauri, the ONKI-SKOS Browser can be opened by pressing a button. Desired concepts can be fetched from the browser to the application by pressing the "Fetch Concept" button. Thus, there is no need for copy-paste procedures or user interface implementation projects. For content searching use cases, ONKI-SKOS provides support for expanding the query term with the subconcepts of the selected query term concept.

The Web Service interface of the ONKI-SKOS server can be used for querying for concepts by label matching, getting label for a given URI or for querying for supported languages of a thesaurus.

The ONKI-SKOS Browser (see Figure 2) is the graphical user interface of ONKI-SKOS. It consists of three main components: 1) *semantic autocompletion concept search*, 2) *concept hierarchy* and 3) *concept properties*. When typing text to the search field, a query is performed to match the concepts' labels. The result list shows the matching concepts, which

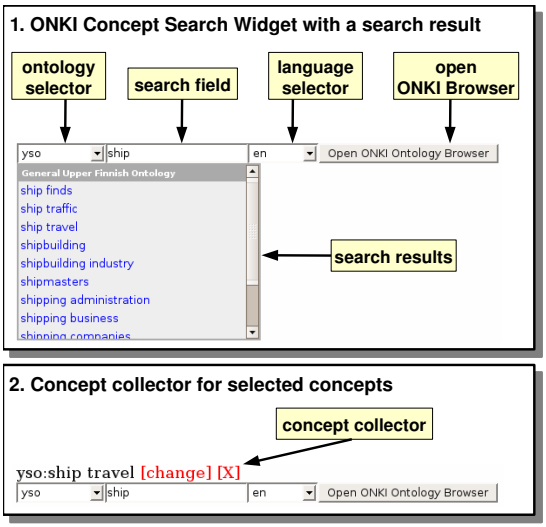

Figure 1: The ONKI Widget for concept searching.

can be selected for further examination.

When a concept is selected, its concept hierarchy is visualized as a tree structure. The ONKI-SKOS Browser supports multi-inheritance of the concepts (i.e. a concept can have multiple parents). Whenever a multi-inheritance structure is met, a new branch is formed to the tree. This leads to cloning of nodes, i.e. a concept can appear multiple times in the hierarchy tree. As a negative side effect, this increases the overall size of the tree. Next to the concept hierarchy tree, the properties of the selected concept are shown in the user interface.

ONKI-SKOS is implemented as a Java Servlet using the Jena Semantic Web Framework[9], the Direct Web Remoting library[10] and the Lucene[11] text search engine.

4 Configuring ONKI-SKOS with SKOS structures

ONKI-SKOS supports straightforward loading of SKOS vocabularies with minimal configuration needs. For using other data models than SKOS, various configuration properties are specified to enable ONKI-SKOS to process the thesauri/ontologies as desired. The configurable properties include the ontological properties used in hierarchy generation, the properties used to label the concepts, the concept to

[8]This is the way the Finnish General Thesaurus YSA has been used previously via the VESA Web Thesaurus Service, http://vesa.lib.helsinki.fi/.

[9]http://jena.sourceforge.net/

[10]http://directwebremoting.org/

[11]http://lucene.apache.org/java

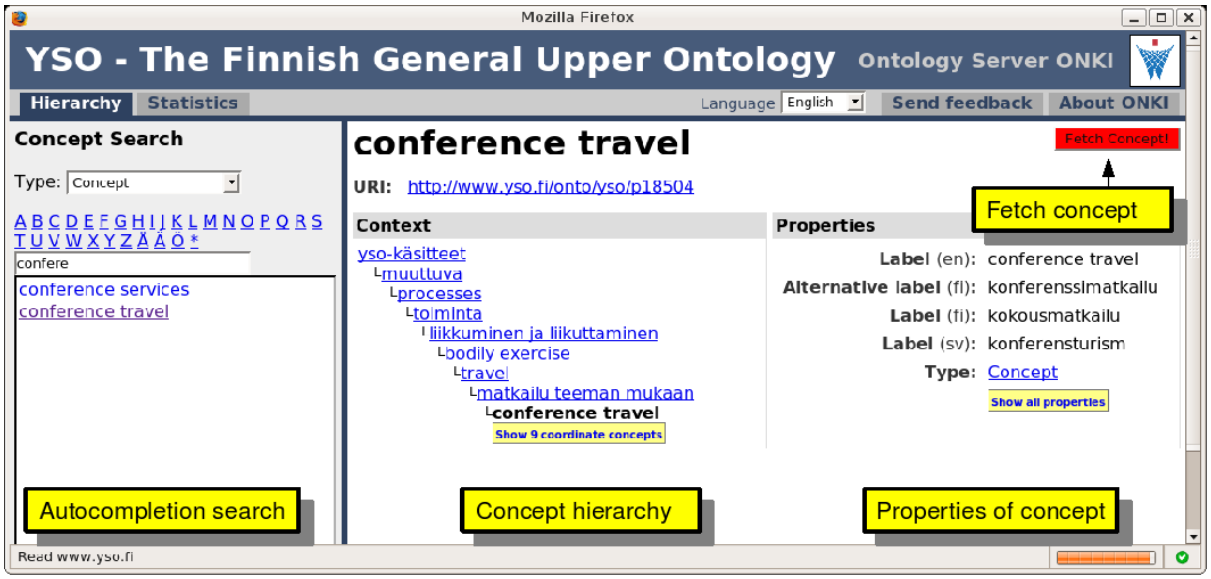

Figure 2: The ONKI-SKOS Browser.

be shown in the default view and the default concept type used in restricting the concept search.

When the ONKI-SKOS Browser is accessed with no URL parameters, information related to the concept configured to be shown as default is shown. Usually this resource is the root resource of the vocabulary, if the vocabulary forms a full-blown tree hierarchy with one single root. In SKOS concept schemes the root resource is the resource representing the concept scheme itself, i.e. the resource of type *skos:ConceptScheme*.

The concept hierarchy of a concept is generated by traversing the configured properties. In SKOS these properties are *skos:narrower* and *skos:broader* and they are used to express the hierarchical relations between concepts . Hierarchical relations between the root resource representing the concept scheme and the top concepts of the concept scheme are defined with the property *skos:hasTopConcept*.

Labels of concepts are needed in visualizing search results, concept hierarchies, and related concepts in the concept property view. In SKOS the labels are expressed with the property *skos:prefLabel*. The label is of the same language as the currently selected user interface language, if such a label exists. Otherwise any label is used.

The semantic autocompletion search of ONKI-SKOS works by searching for concepts whose labels match the search string. To support this, the labels of the concepts are indexed. The indexed properties can be configured. In SKOS these properties are *skos:prefLabel*, *skos:altLabel* and *skos:hiddenLabel*. When the user searches, e.g., with the search term "cat", all concepts which have one of the aforementioned properties with values starting with the string "cat" are shown in the search results. The autocompletion search also supports wildcards, so a search with a string "*cat" returns the concepts which have the string "cat" as any part of their label.

The search can be limited to certain types of concepts only. To accomplish this, the types of the concepts (which are expressed with the property *rdf:type*) are indexed. It is also possible to limit the search to a certain subtree of the concept hierarchy by restricting the search to the children of a specific concept. Therefore also the parents of concepts are indexed.

Many thesauri include structures for representing categories of concepts. To support category-based concept search, another search field is provided. When a category is selected from the category search view, the concept search is restricted to the concepts belonging to the selected category. SKOS includes a concept collection structure, *skos:Collection*, which can be used for expressing such categories. However, *skos:Collection* is often used for slightly different purposes, namely for node labels[12]. For this reason resources of type *skos:Collection* are not used for category-based concept search by default.

[12]A construct for displaying grouping concepts in systematic displays of thesauri. They are not actual concepts, and thus they should not be used for indexing. An example node label is "milk by source animal".

5 Converting thesauri to SKOS – case YSA

Publishing a thesaurus in the ONKI-SKOS server is straightforward. To load a SKOS vocabulary into the server, only the location path of the RDF file of the vocabulary needs to be configured manually. After rebooting the ONKI-SKOS, the RDF file is loaded, indexed and made accessible for users. ONKI-SKOS provides the developers of thesauri a simple way to publish their thesauri.

There exists quite amount of well-established keyword lists, thesauri and other non-RDF controlled vocabularies which have been used in traditional approaches in harmonizing content indexing. In order to reuse the effort already invested developing these resources by publishing these vocabularies in ONKI-SKOS server, conversion processes need to be developed. This idea has also been suggested by van Assem et al. (2006). We have implemented transformation scripts for, e.g., MARCXML format[13], XML dumps from SQL databases and proprietary XML schemas.

An example of the SKOS transformation and publishing process is the case of YSA, the Finnish General Thesaurus[14]. YSA is developed by the National Library of Finland and exported into MARCXML format.

The constantly up-to-date version of the YSA XML file resides at the web server of the National Library of Finland, from where it is fetched via OAI-PMH protocol[15] to our server. This process is automated and the new version of the XML file is fetched daily. After fetching a new version of the file, the transformation process depicted in Figure 3 is started by loading the MARCXML file (ysa.xml). The Java-based converter first creates the necessary structure and namespaces for the SKOS model utilizing Jena Semantic Web Framework. Next, the relations in YSA are translated into their respective SKOS counterparts, which is depicted in Figure 4.

A URI for the new concept entry is created through the unique ID in the source file. The preferred and alternative labels can be converted straightforwardly from one syntax to another. Similarly the type and scheme definitions are added to the SKOS model. Since the relations in the MARCXML refer not to the identifiers but rather to the labels, the source file is searched for an entry that has the given label and then its ID is recorded for the SKOS relation.

[13] http://www.loc.gov/standards/marcxml/

[14] http://www.nationallibrary.fi/libraries/thesauri/ysa.html

[15] http://www.openarchives.org/OAI/openarchivesprotocol.html

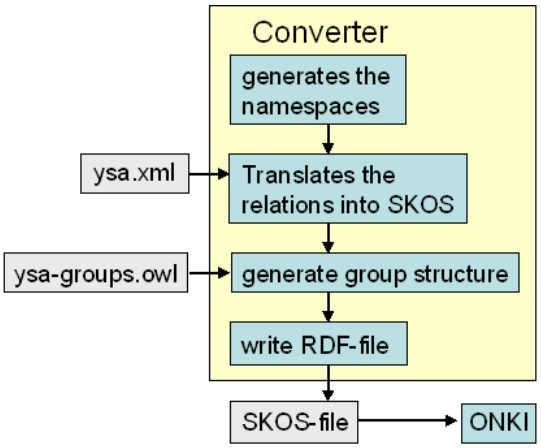

Figure 3: The SKOS transformation process of YSA.

Once the SKOS transformation is ready, the converter fetches the labels for the concept categories from a separate file (ysa-groups.owl) - these labels are not included in the MARCXML file. Finally, a RDF file is written and imported into ONKI-SKOS.

6 Discussion

The main contribution of this paper was depicting how thesauri can be published and utilized easily in the Semantic Web. The benefits of the use of W3C's SKOS data model as a uniform vocabulary representation framework were emphasized. The ONKI-SKOS server was presented as a proof of concept for cost-efficient thesauri utilization method. By using ONKI-SKOS, general thesauri accessing functionalities can be easily integrated into applications without the need for users to implement their own user interfaces for this. The processing of the SKOS structures in an ontology server was depicted in context of the ONKI-SKOS server. The case of the Finnish General Thesaurus was presented as an example how an existing thesaurus can be converted into the SKOS format and published on the ONKI-SKOS server.

Future work includes creating a more extensive Web Service interface for supporting, e.g., querying for properties of a given concept and for discovering concepts which are related to a given concept. The starting point for this API will be the SKOS API.

Related to the ONKI ontology services, there are plans for implementing a web widget intended for content searching. It will help the user to find relevant query concepts from thesauri and perform semantic query expansion (subconcepts, related concepts etc.)

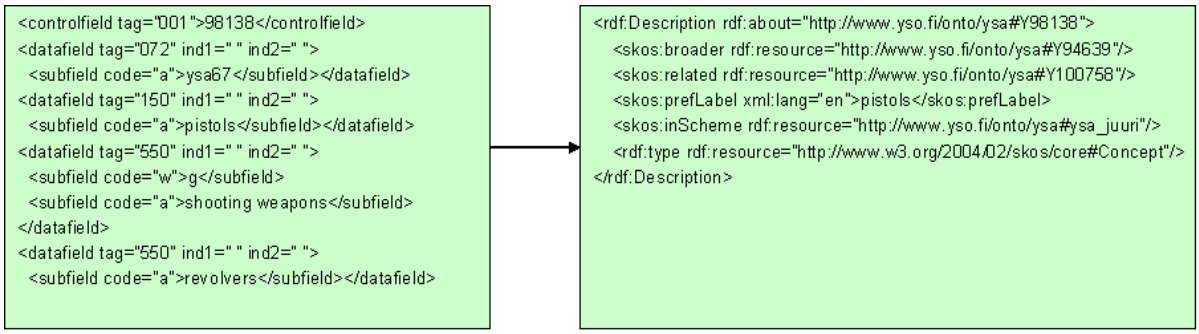

Figure 4: An example of the SKOS transformation of YSA.

for using other relevant concepts in the query. After selecting the desired query terms, the query is passed to the search component of the underlying system. The widget will enable multilingual search based on the languages provided by the used thesaurus. If the thesaurus contains, e.g., English and Finnish labels for the concepts, the search for relevant query concepts can be done in English or Finnish, and in the actual search either the URIs, English labels or Finnish labels can be used as query terms, depending on how the content is annotated in the underlying system.

Acknowledgements

We thank Ville Komulainen for his work on the original ONKI server. This work is a part of the National Semantic Web Ontology project in Finland[16] (FinnONTO) and its follow-up project Semantic Web 2.0[17] (FinnONTO 2.0, 2008-2010), funded mainly by the National Technology and Innovation Agency (Tekes) and a consortium of 38 private, public and non-governmental organizations.

References

Mohammad Nazir Ahmad and Robert M. Colomb. Managing ontologies: a comparative study of ontology servers. In *Proceedings of the eighteenth Conference on Australasian Database (ADC 2007)*, pages 13–22, Ballarat, Victoria, Australia, January 30 - February 2 2007.

Jean Aitchison, Alan Gilchrist, and David Bawden. *Thesaurus Construction and Use: A Practical Manual*. Europa Publications, 4th edition, 2000.

Ying Ding and Dieter Fensel. Ontology library systems: The key to successful ontology reuse. In *Proceedings of SWWS'01, The first Semantic Web Working Symposium, Stanford University, USA*, pages 93–112, August 1 2001.

Michiel Hildebrand, Jacco van Ossenbruggen, Alia Amin, Lora Aroyo, Jan Wielemaker, and Lynda Hardman. The design space of a configurable autocompletion component. Technical Report INS-E0708, Centrum voor Wiskunde en Informatica (CWI), Amsterdam, 2007. URL `http://www.cwi.nl/ftp/CWIreports/INS/INS-E0708.pdf`.

Eero Hyvönen, Kim Viljanen, Jouni Tuominen, and Katri Seppälä. Building a national semantic web ontology and ontology service infrastructure—the finnonto approach. In *Proceedings of the 5th European Semantic Web Conference (ESWC 2008)*, June 1-5 2008.

Alistair Miles, Brian Matthews, Michael Wilson, and Dan Brickley. SKOS Core: Simple knowledge organisation for the web. In *Proceedings of the International Conference on Dublin Core and Metadata Applications (DC 2005)*, Madrid, Spain, September 12-15 2005.

Douglas Tudhope and Ceri Binding. Towards terminology service: experiences with a pilot web service thesaurus browser. In *Proceedings of the International Conference on Dublin Core and Metadata Applications (DC 2005)*, pages 269–273, Madrid, Spain, September 12-15 2005.

Mark van Assem, Maarten R. Menken, Guus Schreiber, Jan Wielemaker, and Bob Wielinga. A method for converting thesauri to RDF/OWL. In *Proceedings of the Third International Semantic*

[16] http://www.seco.tkk.fi/projects/finnonto/
[17] http://www.seco.tkk.fi/projects/sw20/

Web Conference (ISWC 2004), pages 17–31, Hiroshima, Japan, November 7-11 2004.

Mark van Assem, Véronique Malaisé, Alistair Miles, and Guus Schreiber. A method to convert thesauri to SKOS. In *Proceedings of the third European Semantic Web Conference (ESWC 2006)*, pages 95–109, Budva, Montenegro, June 11-14 2006.

Kim Viljanen, Jouni Tuominen, and Eero Hyvönen. Publishing and using ontologies as mash-up services. In *Proceedings of the 4th Workshop on Scripting for the Semantic Web (SFSW 2008), 5th European Semantic Web Conference 2008 (ESWC 2008)*, Tenerife, Spain, June 1-5 2008.

Diane Vizine-Goetz, Eric Childress, and Andrew Houghton. Web services for genre vocabularies. In *Proceedings of the International Conference on Dublin Core and Metadata (DC 2005)*, Madrid, Spain, September 12-15 2005.