SKOS: a guide for information professionals

A guide to representing structured controlled vocabularies in the Simple Knowledge Organization System

Priscilla Jane Frazier

Simple Knowledge Organization System (SKOS) and associated Web technologies aim to enable pre-existing controlled vocabularies to be consumed on the Web and to allow vocabulary creators to publish born-digital vocabularies on the Web.

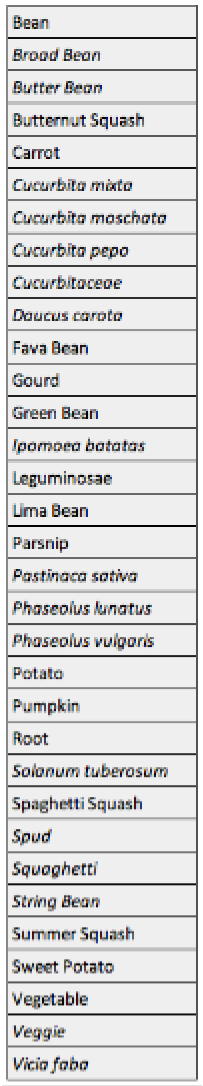
The purpose of this guide is to allow catalogers, librarians, and other information professionals to understand and use SKOS, a World Wide Web Consortium (W3C) standard designed for the representation of controlled vocabularies to be consumed within the Web environment.

Included in this guide is a brief history of classification technologies and the history of SKOS and an examination of SKOS within the context of related technology standards.[1] Following a discussion of the elements and syntax of the SKOS vocabulary, the guide discusses various integrity conditions which are used as a guideline for best practices in constructing SKOS vocabularies. The guide then examines past literature, highlighting the conversion, validation, improvement and automatic generation of SKOS vocabularies. The final section concludes with a discussion of future directions that SKOS development might take.

[1] As they form the foundation upon which SKOS is based, technology standards such as the eXtensible Markup Language (XML) and the Resource Description Framework (RDF) are defined in Appendix I.

Introduction

Throughout history, classification systems have been widely used in the library community. Knowledge organization systems, and more specifically controlled structured vocabularies, are a growing area within the field of classification systems. Within a Web context, formats have been proposed for representing controlled vocabularies in a structured way using the Web standards eXtensible Markup Language (XML) and Resource Description Framework (RDF).



Bean
Broad Bean
Butter Bean
Butternut Squash
Carrot
Cucurbita mixta
Cucurbita moschata
Cucurbita pepo
Cucurbitaceae
Daucus carota
Fava Bean
Gourd
Green Bean
Ipomoea batatas
Leguminosae
Lima Bean
Parsnip
Pastinaca sativa
Phaseolus lunatus
Phaseolus vulgaris
Potato
Pumpkin
Root
Solanum tuberosum
Spaghetti Squash
Spud
Squaghetti
String Bean
Summer Squash
Sweet Potato
Vegetable
Veggie
Vicia faba

Figure 1.

An important distinction needs to be made between controlled vocabularies that have been published to the Web and those that have been published in a structured way specifically *for* the Web. Natural-language vocabularies, such as simple subject heading lists, thesauri and back-of-the-book indexes are extremely useful for humans, but the meaning that machines can derive from them is very limited. Linked open data and linked open vocabularies are both Semantic Web technologies that enable the publishing of controlled vocabularies for the Web in such a way that both humans and machines can recognize meaning (Kaltenböck and Bauer 2012). The creation of linked open vocabularies allows for increased and more meaningful points of access and discovery, and greater effectiveness in information retrieval.

A controlled vocabulary allows for organization of some content, or knowledge, such that it can be easily retrieved at a later time. These vocabularies are “controlled” in that they make use of authorized identification of the content they contain. These groupings of concepts are carefully selected and described so that the information they contain can be retrieved in the most intelligent ways possible. Take, for example, the very small collection of terms called the veggie vocab, found in Figure 1. This group of terms about vegetables is listed in alphabetical order. At a glance, one can recognize that the list is comprised of the names of foods (in English) and the names of species (in Latin). However, the same information can be displayed in a different way that makes it much easier for people to understand the relationships between the different terms.

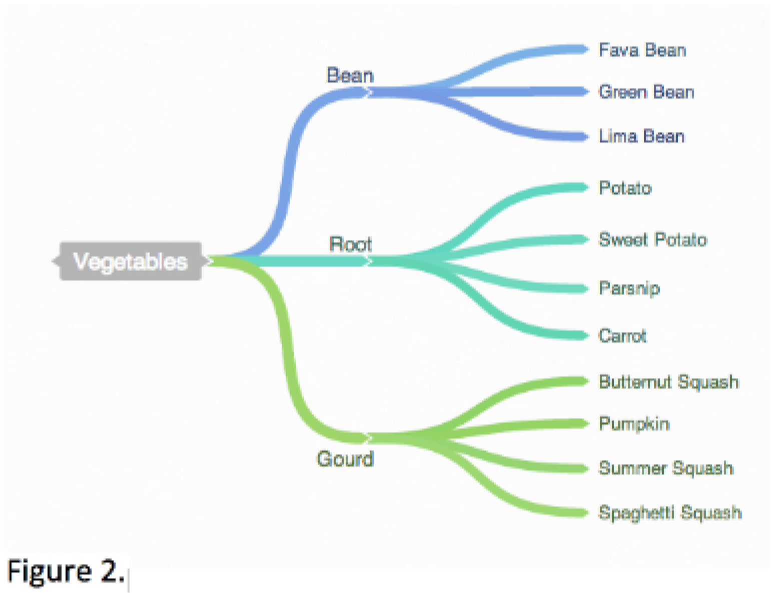
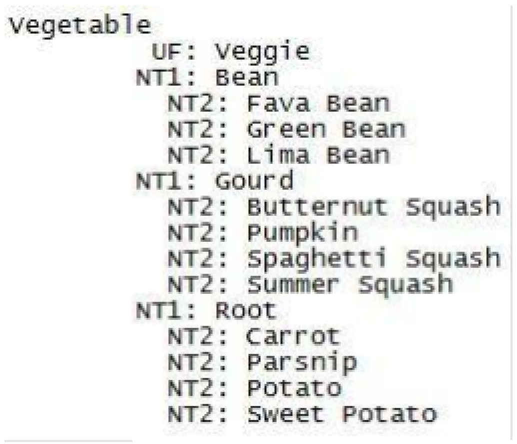


Figure 2.

It is easy to see, from the visualization in Figure 2[2], that the vocabulary is about types of vegetables, Vegetables being the "top" term in the vocabulary. The terms **Bean**, **Root**, and **Gourd** are directly below **Vegetables**. These terms describe varieties of vegetables. Below this second level of terms is a group of terms of actual vegetables, like **Potato** and **Pumpkin**. The relationships between terms are hierarchical; in other words, are broader or narrower in scope with relation to one another. The two types of hierarchical relationships in controlled vocabularies are *broader term* (BT) and *narrower term* (NT). For example, **Root** is a narrower term in relation to **Vegetables**, but a broader term in relation to **Parsnip**.

Another way to visualize the vocabulary is as a hierarchical report. Figure 3 shows the hierarchy of terms and terms to which they are related. This figure also includes the term **Veggie** with the label UF. An equivalency relationship, such as *use for* (UF) or *use* (USE) allows the vocabulary to make connections between synonyms and near-synonyms. In this particular vocabulary, Vegetable is an authorized or preferred term, and **Veggie** is an unauthorized or non-preferred term for the same concept.

[2] Image created using the Coggle mind mapping Web application. (https://coggle.it/)



```
Vegetable
         UF: Veggie
        NT1: Bean
          NT2: Fava Bean
          NT2: Green Bean
          NT2: Lima Bean
        NT1: Gourd
          NT2: Butternut Squash
          NT2: Pumpkin
          NT2: Spaghetti Squash
          NT2: Summer Squash
        NT1: Root
          NT2: Carrot
          NT2: Parsnip
          NT2: Potato
          NT2: Sweet Potato
```

Figure 3.

The final type of relationship that can be reflected in a vocabulary is an associative relationship. This type of relationship allows the vocabulary to make connections between *related terms* (RT), or terms that have neither hierarchical nor equivalency relationships. For example, **Fruit** might be a related term to **Vegetable**.

SKOS Defined

SKOS is a common data model for knowledge organization systems such as thesauri, classification schemes, subject heading systems and taxonomies. Using SKOS, a knowledge organization system or controlled vocabulary can be expressed as machine-readable data. Once expressed in SKOS, data can then be exchanged between computer applications and published in a machine-readable format on the Web.

The SKOS-Core 1.0 Guide was first introduced in 2001 by the W3C[3] Semantic Web Deployment Working Group[4] (SWDWG) in order to develop SKOS as a W3C standardized classification system. The

[3] http://www.w3.org/
[4] http://www.w3.org/2006/07/SWD/

W3C SWDWG currently maintains several pieces of documentation on SKOS which are freely available on the Web. First, the SKOS Reference[5] document, which is currently at the final W3C recommendation or standard stage, defines SKOS. Second, the SKOS Primer[6] document provides a guide for users of the system. Third, the SKOS Use Cases and Requirements[7] document presents a list of representative use cases and a set of requirements derived from these use cases. The SWDWG also maintains an open mailing list[8] and a wiki[9] through which the public may contribute to the development of SKOS.

According to the SKOS Reference document, the aims of the system are:

- "to provide a bridge between different communities of practice within the library and information sciences involved in the design and application of knowledge organization systems."
- "to provide a bridge between these communities and the Semantic Web by transferring existing models of knowledge organization to the Semantic Web technology context, and by providing a low-cost migration path for porting existing knowledge organization systems to RDF."

SKOS is a data-sharing standard which was built upon several pre-existing Semantic Web standards for formal logic and structure. These technologies provide ways of expressing meaning that are amenable to computation and that complement and give structure to information already existing on the Web. SKOS was built on RDF, and thus SKOS data are represented as RDF triples.[10]

The Elements of SKOS

[5] http://www.w3.org/TR/skos-reference/
[6] http://www.w3.org/TR/skos-primer/
[7] http://www.w3.org/TR/skos-ucr/
[8] http://lists.w3.org/Archives/Public/public-swd-wg/
[9] http://www.w3.org/2006/07/SWD/wiki/
[10] RDF, and other related Web technologies, are defined in Appendix I.

The SKOS Vocabulary:

The vocabulary of SKOS includes various elements which work together to represent knowledge organization systems. These elements include concepts, labels, relationships, mapping properties, collections and notes.

Concepts

`skos:Concept` (instance of `owl:Class`[11])

A SKOS concept is any unit of thought: an idea, an object, an event. These concepts are the building blocks of many knowledge organization systems. Because concepts are abstract ideas that exist in the mind, they are independent of the terms used to describe them. Let's take carrots for example. The English word "carrot" which we use to describe the orange vegetable that rabbits like to eat is actually independent of the concept of a carrot. The idea of a concept and its descriptor (or label) being two separate entities is vital to the SKOS model because it allows machines to identify concepts via their identifier and humans to identify concepts via their label. The SKOS "concept" element allows vocabulary builders to describe and distinguish concepts and their descriptors (labels). A SKOS concept can be created in two steps:

1. Create or reuse a uniform resource identifier (URI) to uniquely identify a concept
2. Assert (make a statement) in RDF, using the property **`rdf:type`**, that the resource identified by this URI is of type **`skos:Concept`**.

Example:

[11] Web Ontology Language (OWL) and Resource Description Framework Schema (RDFS) syntactical equivalents to SKOS elements are provided for reference.

```
<http://www.veggievocab.com/vegetable/root/carrot> rdf:type skos:Concept
```

Labels

In SKOS, a label is the element that is the descriptor of a concept. The three SKOS label elements are sub-properties of the RDFS element **`rdfs:label`**. The purpose of these three elements is to link a **`skos:Concept`** to an RDF plain literal, or character string.

`skos:prefLabel` (instance of `owl:AnnotationProperty` and sub-property of `rdfs:label`)

Preferred Label is a SKOS element that makes it possible to assign an authorized name to a concept. The two following examples show that the preferred label for the concept of a vegetable is the word “vegetable” in English and “légume” in French.

Example:

```
ex:vegetable rdf:type skos:Concept;
skos:prefLabel "Vegetable".
```

Example:

```
ex:vegetable rdf:type skos:Concept;
skos:prefLabel "Vegetable"@en;
skos:prefLabel "Légume"@fr.
```

For information retrieval and information organization purposes, no two concepts in the same knowledge organization system should be given the same preferred label for any given language tag.

`skos:altLabel` (instance of `owl:AnnotationProperty` and sub-property of `rdfs:label`)

Alternate Label makes it possible to assign an unauthorized name to a concept. This label allows multiple same-language descriptors for a concept to be stored. The following example shows that the preferred label for the concept fava_bean is the term “fava bean” and an alternate label is the term “broad bean.”

Example:

```
ex:fava_bean rdf:type skos:Concept;
skos:prefLabel "Fava bean"@en;
skos:altLabel "Broad bean"@en.
```

`skos:hiddenLabel` (instance of `owl:AnnotationProperty` and sub-property of `rdfs:label`)

Hidden Label is a label for a resource that a knowledge organization system designer would like to be accessible to applications performing text-based indexing and search operations, but not visible otherwise. The following example shows that the preferred label for the concept “potato” is the term “potato,” and two hidden labels for the concept are “tater” and “spud.” Hidden label may be used for nicknames or colloquialisms which may be used to refer to the concept in question, but are not appropriate for official documentation.

Example:

```
ex:potato rdf:type skos:Concept;
skos:prefLabel "Potato"@en;
skos:hiddenLabel "Tater"@en.
skos:hiddenLabel "Spud"@en.
```

Relationships

`skos:broader` and **`skos:narrower`** (instances of `owl:ObjectProperty`)

These two SKOS labels assert hierarchical relationships between concepts, i.e., that one concept is broader or narrower in meaning than another.

Example:

```
ex:root rdf:type skos:Concept;
skos:prefLabel "Root Vegetable"@en;
skos:narrower ex:sweet_potato.
```

Example:

```
ex:sweet_potato rdf:type skos:Concept;
skos:prefLabel "Sweet Potato"@en;
skos:broader ex:root.
```

`skos:related` (instance of `owl:ObjectProperty`)

This SKOS label allows a designer to assert an associative relationship between two concepts.

Example:

```
ex:vegetable rdf:type skos:Concept;
skos:prefLabel "Vegetable"@en;
skos:related ex:fruit.

ex:fruit rdf:type skos:Concept;
skos:prefLabel "Fruit"@en.
```

In the SKOS data model, **`skos:related`** is not defined as a transitive property, and the transitive closure of **`skos:broader`** must be disjoint from **`skos:related`**. If the concepts **Vegetable** and **Fruit** are related via **`skos:related`**, there must not be a chain of **`skos:broader`** relationships from **Vegetable** to **Fruit**. In other words, there must not be simultaneous instances of hierarchical and associative relationships between concepts.

Semantic Relationships

`skos:semanticRelation` (instance of [`owl:ObjectProperty`])

SKOS semantic relations are connections between SKOS concepts. This type of relation occurs when a link between two concepts is inherent in the meaning of the linked concepts. Each of the SKOS labels **`skos:broader`**, **`skos:narrower`**, **`skos:broaderTransitive`**, **`skos:narrowerTransitive`** and **`skos:related`** are sub-properties of **`skos:semanticRelation`**.

`skos:broaderTransitive` and **`skos:narrowerTransitive`** (instances of [`owl:TransitiveProperty`])

Like **`skos:broader`** and **`skos:narrower`**, these two SKOS labels assert hierarchical relationships between concepts, i.e., that one concept is broader or narrower in meaning than another. The transitive nature of these two labels signifies that statements such as: "if *vegetables* is broader than *gourd* and *gourd* is broader than *pumpkin*, then *vegetables* is assumed to be broader than *pumpkin*" can be expressed in the SKOS data model.

Mapping Properties

`skos:mappingRelation` (instance of [`owl:ObjectProperty`])

The SKOS mapping labels are used to state mapping (or alignment) connections between SKOS concepts existing in different concept schemes, for example, Library of Congress Subject Headings (LCSH), Medical Subject Headings (MeSH) and Thesaurus for Graphic Materials (TGM).

`skos:closeMatch` (instance of [`owl:ObjectProperty`] and [`owl:SymmetricProperty`])

`skos:exactMatch` (instance of [`owl:ObjectProperty`], [`owl:SymmetricProperty`] and owl:TransitiveProperty)

skos:exactMatch is a sub-property of **skos:closeMatch**.

skos:broadMatch and skos:narrowmatch (instances of owl:ObjectProperty)

skos:broadMatch and **skos:narrowMatch** are used to specify a hierarchical link between concepts. These two properties are inversely related to one another. **skos:broadMatch** is a sub-property of **skos:broader** and **skos:narrowMatch** is a sub-property of **skos:narrower**.

skos:relatedMatch (instance of owl:ObjectProperty and owl:SymmetricProperty)

Collections of Concepts

skos:Collection (instance of owl:Class)

The SKOS concept collections labels are used to describe labeled or ordered groups of SKOS concepts. For example, the veggie vocab can be considered a collection of SKOS concepts because it is a group of concepts that have something in common.

Example:

```
<Veggie_Vocab> rdf:type skos:Collection;
   skos:prefLabel “Salad ingredients”@en;
   skos:member ex:lettuce;
   skos:member ex:tomato;
   skos:member ex:onion;
   skos:member ex:cucumber;
   skos:member ex:carrot.
```

skos:OrderedCollection (instance of owl:Class)

The SKOS concept ordered collection is used to capture a list of items which have been put into some type of order, i.e. chronology or alphabetization.

Example:

```
<Veggie_Vocab> rdf:type skos:OrderedCollection;
   skos:prefLabel “Vegetables by size”@en;
   skos:memberList :b1.
   _:b1 rdf:first ex:pumpkin;
   rdf:rest _:b2.
   _:b2 rdf:first ex:potato;
   rdf:rest _:b3.
   _:b3 rdf:first ex:limaBean;
   rdf:rest rdf:nil.
```

`skos:member` (instance of `owl:ObjectProperty`)

The SKOS concept member is used to define multiple members of a collection.

Example:

```
<Veggie_Vocab> rdf:type skos:Collection;
skos:member <bean> , <gourd> , <root> .
```

`skos:memberList` (instance of `owl:ObjectProperty` and `owl:FunctionalProperty`)

The SKOS concept member is used to define multiple members of a collection in a list format.

Example:

```
<Veggie_Vocab> rdf:type skos:OrderedCollection;
skos:memberList ( <bean> , <gourd> , <root> ) .
```

Notes

`skos:note`

This SKOS label was created for general documentation purposes. There is a hierarchical link between **`skos:note`** and its different specializations which allows multiple notes to be captured separately and appropriately and for additional information associated with a concept to be retrieved in a straightforward way.

`skos:scopeNote` (instance of `owl:AnnotationProperty`)

This label supplies some information about the intended meaning of a concept. It is usually used as an indication of how the use of a concept is limited in indexing practice.

Example:

```
ex:root skos:scopeNote
"Used for plant roots used as vegetables"@en.
```

`skos:definition` (instance of `owl:AnnotationProperty`)

This label supplies a complete explanation of the intended meaning of a concept.

Example:

```
ex:parsnip skos:scopeNote
"The parsnip (Pastinaca sativa) is a root vegetable related to the carrot. Parsnips resemble
carrots, but are paler in color than most carrots, and have a sweeter taste, especially when
cooked."@en.
```

`skos:example` (instance of `owl:AnnotationProperty`)

This label supplies an example of the use of a concept.

Example:

```
ex:pumpkin skos:example
"baking, cooking, pumpkin seeds, pumpkin seed oil, pumpkin carving, jack o’lanterns,
etc."@en.
```

skos:historyNote (instance of owl:AnnotationProperty)

This label describes significant changes that have been made to the meaning or form of a concept.

Example:

```
ex:green_bean skos:historyNote
"The first "stringless" bean was bred in 1894 by Calvin Keeney while working in Le Roy,
New York."@en.
```

skos:editorialNote (instance of owl:AnnotationProperty)

This label supplies information that is an administrative aid, such as reminders of editorial work still to be done, notifications that future editorial changes might be made, etc.

Example:

```
ex:lima_bean skos:editorialNote "Check for alternate terms"@en.
```

skos:changeNote (instance of owl:AnnotationProperty)

This label documents fine-grained changes to a concept for the purposes of administration and maintenance.

Example:

```
ex:Parsnip skos:changeNote
"Moved from under 'Gourd' to under 'Root' by Priscilla Jane Smith"@en.
```

It should also be mentioned that it is possible to use non-SKOS properties to document concepts (i.e. Dublin Core's **`dct:creator`**).

Example:

```
ex:spaghetti_squash dct:creator [ foaf:name "Priscilla Jane Frazier" ].[12]
```

SKOS Integrity Conditions

The SKOS Reference document includes several integrity conditions. Integrity conditions are statements that help to determine whether or not given data (for example, a vocabulary) are consistent with respect to the SKOS data model. The purpose of SKOS integrity conditions is to encourage the construction of well-formed and consistent data and to promote interoperability between data represented in SKOS.

`skos:ConceptScheme` ***is disjoint with*** **`skos:Concept`**

This condition states that SKOS concept schemes, or groups of SKOS concepts, must not be on the same hierarchical level as SKOS concepts and vice versa. For example, in the veggie vocab, Vegetables is

[12] In this example, the `dct:creator` property is further defined by the use of the Friend-of-a-Friend system (FOAF) in order to state that the identity of the "creator" of this concept (Priscilla Jane Frazier) is connected to other individuals via the FOAF community.

a **`skos:ConceptScheme`**. This means that Vegetables must not also be a **`skos:Concept`** and one of the concepts, like Lima Bean, may not be a **`skos:ConceptScheme`**.

`skos:prefLabel`, `skos:altLabel` *and* `skos:hiddenLabel` *are pairwise disjoint properties*

This condition states that no SKOS concept may be a member of more than one preferred label, alternate label and hidden label.

A resource has no more than one value of `skos:prefLabel` *per language tag*

This condition states that no SKOS concept may have more than one preferred label for each language tag. For example, the concept summer_squash has the preferred label of “Summer Squash” in English and of “Cucurbita pepo” in Latin, and there may not be any other preferred labels in English or Latin.

`skos:related` *is disjoint with the property* `skos:broaderTransitive`

This condition states that no two SKOS concepts may be connected by both related and broader transitive relationships.

`skos:Collection` *is disjoint with each of* `skos:Concept` *and* `skos:ConceptScheme`

This condition states that SKOS collections, or labeled or ordered groups of SKOS concepts, must not be on the same hierarchical level as SKOS concepts and vice versa. For example, in the veggie vocab, Vegetables is a **`skos:Collection`**. This means that Vegetables must not also be a **`skos:Concept`** and one of the concepts, like Lima Bean, may not be a **`skos:Collection`**.

`skos:exactMatch` *is disjoint with each of the properties* `skos:broadMatch` *and*

`skos:relatedMatch`

This condition states that no two SKOS concepts may be related by more than one of exact match, broader match, and related match.

Literature Review

This literature review covers known SKOS conversion and validation techniques. Also included are numerous SKOS custom expansions and improvement techniques that have been implemented in the past. Finally, the literature review will discuss the new technique of vocabulary creation using SKOS, and the state of the field as it stands today.

SKOS Tools

In the information profession, research and development efforts on the conversion of controlled vocabularies to the SKOS format are being pursued, and a number of technologies and methods have been proposed. Although a manual conversion of a controlled vocabulary by a person or group of people is possible, it is time-consuming and likely prone to error. Applications have been developed to automatically convert vocabularies into the SKOS format, to validate the quality of the formatting of a vocabulary in SKOS format,[13] and most recently, to improve the quality and validity of SKOS vocabularies.[14] This literature review covers SKOS conversion and validation techniques. Numerous SKOS custom expansions and improvement techniques that have been implemented in the past are also documented. Finally, the literature review will discuss new techniques for vocabulary creation using SKOS, and the state of the field as it stands today.

[13] For example, the PoolParty online SKOS Consistency Checker. (http://poolparty.biz/)

[14] For example, Skosify is a tool that can be used to convert and improve vocabularies expressed as RDFS and OWL into SKOS format. (http://demo.seco.tkk.fi/skosify/skosify)

Conversion techniques

In 2001, the Semantic Web Advanced Development for Europe (SWAD-E) published *Migrating Thesauri to the Semantic Web: Guidelines and Case Studies for Generating RDF Encodings of Existing Thesauri*, a thesaurus research prototype that presents guidelines and methods for migrating traditional thesaurus systems to RDF based thesaurus systems. The process described in this document consists of three stages. First, an RDF encoding of the thesaurus is generated. Second, the encoding is taken through error checking and validation processes. Third, the encoding is published on the Web. During the first step, a traditional, or term-oriented, view of a thesaurus is converted to a concept-oriented view. Thus, each "preferred term" in a thesaurus becomes a "preferred label" for a "concept." Each "preferred label" is given a tag of **`skos:prefLabel`**, and each concept is given a tag of **`skos:concept`**. Each "concept" in the thesaurus is assigned a unique URI which can be linked, through the Web, to URIs for related concepts. The technique of designating unique and persistent URIs for all concepts in a vocabulary allows machines to understand the relationships between concepts similar to the way humans understand these types of relationships intrinsically.

Assem, Malaisé and Miles (2006) expand on the technique used in step one of the SWAD-E document. Their paper suggests three activities that will effectively link the term-oriented view of a thesaurus to its concept-oriented view. First, the digital format and the documentation of the thesaurus are analyzed to determine the features of the thesaurus and how it is encoded. Second, a mapping between the thesaurus data items and the SKOS RDF is defined. Third, a transformation program, or algorithm, is created. The authors mention a sub-activity in which pre-existing URIs, if present, are identified. If no URIs exist, in the term-oriented view of a thesaurus, the authors suggest creating randomly generated unique identifiers or using the name of unique preferred terms to generate URIs.

Assem, Malaisé and Miles apply their new technique to three existing thesauri: Integrated Public

Sector Vocabulary (IPSV), Common Thesaurus for Audiovisual Archives (GTAA) and Medical Subject Headings (MeSH). These particular thesauri are chosen because of their popularity and range in complexity. The authors find that conversion of the largest and most complex thesaurus, MeSH, is problematic but those difficulties assist in identifying the boundaries of the applicability of their technique. The fact that MeSH contains textual notes which combine several types of knowledge, or compound concepts, leads the authors to conclude that some thesauri have complex structures for which no direct SKOS counterpart exists. Additionally, IPSV and MeSH contain management information about their terms, which cannot be represented within the SKOS standard. The authors also mention that at the time of their study validation of SKOS RDF is difficult, due to the lack of validation technologies.

In 2008 Summers, Isaac, Redding and Krech presented a technique for converting Library of Congress Subject Headings (LCSH) to SKOS in their paper "LCSH, SKOS and Linked Data." In their research, the authors use content in MARC bibliographic records to harvest LCSH terms and map them to corresponding SKOS concepts. For example, Library of Congress Control Numbers are given to every LCSH, are mapped onto `skos:Concept` and are used to create unique URIs for each concept. Pre-coordinated LCSH terms, or LCSH terms which have been previously combined together in anticipation of a search on that combination of terms, which have the potential to create problems in SKOS because they represent more than one term or concept, are simply flattened into one concept in this technique. To accomplish the conversion, the authors write code using the Python programming language and use open-source MARCXML and RDF processing tools to create an object-oriented streaming interface to mint URIs and link them together. Additionally, the authors suggest that an extension of SKOS would allow the full meaning of LCSH terms to be captured in SKOS form and that the integration of other Semantic Web vocabularies such as Dublin Core could allow SKOS vocabularies to be even more meaningful.

The earlier conversion methods of SWAD-E were adapted by Neubert in his 2009 paper "Bringing the 'Thesaurus for Economics' on to the Web of Linked Data." Neubert found that SKOS's built-in multilingual features are useful, given that the Thesaurus for Economics, or STW, is made up of both English and German terms. The author states that the conversion of STW into SKOS was straightforward. However, at the time of his research, several new SKOS classes, including **`skos:notation`**, had been introduced by W3C. Neubert was able to take advantage of the fact that SKOS could now accommodate internal management information notation, something that Assem, Malaisé, and Miles (2006) had observed SKOS was lacking in 2006.

Validation techniques

Neubert (2009) also discusses the use of SPARQL queries to check SKOS vocabularies for inconsistencies. The author describes the process of loading SKOS vocabularies into a SPARQL server and running inconsistency checks, where queries that check for illogical links between concepts would bring these inconsistencies to the attention of the implementer. Neubert also mentions that this process could potentially improve thesaurus maintenance if inconsistency checks could be performed routinely. The concept of automatic validation of SKOS vocabularies was new at the time of his research. One drawback to this validation process is that an implementer inputting the SPARQL queries would only be able to locate inconsistencies that had previously been anticipated. Inconsistencies that are not queried would not be identified by the SPARQL server.

In their 2012 paper "Improving the Quality of SKOS Vocabularies with Skosify," Suominen and Hyvönen propose a tool with the ability to check the quality of a SKOS vocabulary. The authors cite a lack of quality and validity of existing SKOS vocabularies as the reason for the development of this new tool, named the PoolParty Online SKOS Consistency Checker, or PoolParty. The researchers created a list of eleven validation criteria by which SKOS vocabularies are checked. The criteria were gathered from the

W3C SKOS Reference document and from the authors' examination of fourteen freely-available SKOS vocabularies. The authors found that all of the medium- and large-sized vocabularies failed at least one validation check, meaning that they did not meet some of the SKOS integrity constraints. Only one of the fourteen vocabularies passed all eleven validation criteria.

Custom expansions

As seen in Assem, Malaisé and Miles (2006) and Neubert (2009), controlled vocabularies often contain constructs which have no direct counterpart in SKOS. As stated in the *SKOS Primer*, SKOS is designed for simple extension of language constructs for the specialization of a particular vocabulary. This is made possible by expounding upon a SKOS construct using the extension **`rdfs:subClassOf`**. Neubert proposes two SKOS extensions for use with the STW thesaurus. First, he splits **`skos:Concept`** into two "subclasses." This change is chosen because along with the standard terms, or descriptors, STW includes a taxonomy of approximately 500 classes which are used to aid user information retrieval. Neubert splits **`skos:Concept`** into two parts: **`zbwext:Descriptor rdfs:subClassOf skos:Concept`** for concepts and **`zbwext:Thsys rdfs:subClassOf skos:Concept`** for classes. This approach allows for well-defined semantics in terms of broader and narrower relationships between concepts and allows users to search for both concepts and classes. Additionally, Neubert extended the newly introduced **`skos:note`** using the following notation: **`zbwext:useInsteadNote rdfs:subClassOf skos:note`** for notes which guide users to designated preferred labels rather than other preferred labels in certain circumstances.

Improvement techniques

Along with their validation tool PoolParty, Suominen and Hyvönen (2012) introduce Skosify, a tool created to improve the quality and validity of SKOS vocabularies. Skosify is a command line tool capable of reading one or more SKOS files and outputting a file in which errors and problems have not only been identified, but corrected. Skosify is able to address nine of the eleven validation criteria mentioned earlier in their paper. After being tested on fourteen vocabularies, Skosify corrects problems in all nine of these categories. The tool has the ability to correct missing language tags if a default language is provided, to detect unlabeled concept schemes and add specified labels and to remove unnecessary whitespace surrounding property values. Skosify can recognize and correct more sophisticated problems having to do with the relationships between concepts in vocabularies. The tool is able to identify top level concepts and add **`hasTopConcept`** and **`topConceptOf`** relationships to that concept scheme; recognize the designation of more than one **`prefLabel`** for a single concept and correct the error; and identify as well as correct concepts that have been mislabeled as collections. Additionally, the tool recognizes when a concept is linked to a label using two different label properties and removes the less important property; recognizes when concepts are linked together in a way that is disjointed and removes the related relationship assertion without disabling the broader hierarchy; and recognizes cycles or concepts that have a broader relationship with themselves and removes the offending relationship.

Vocabulary Creation with SKOS

In 2010, Gerbé and Kerhervé proposed a new approach to vocabulary creation with their paper "A Model-Driven Approach to SKOS Implementation." Their technique involves viewing the SKOS conceptual model as a metamodel for structured controlled vocabularies. Using a model management and model engineering approach, the authors state that flexible and extensible vocabularies can be managed and created, which may be useful for non-traditional or highly complex vocabularies. Model operators, or prompts used in model management, such as map, match, merge, and compose, can be used to map term-oriented vocabularies onto a concept-oriented platform. Gerbé and Kerhervé introduced a metamodel and SKOS engine for use in the development of SKOS vocabularies. The metamodel is expressed in much the same way as a database schema, and is supported by the MySQL database system. The SKOS engine is built as a database with an interface for use in populating and visualizing the vocabulary content. The authors state that the tool is also capable of importing, exporting, and merging vocabularies. This model-driven database approach to SKOS vocabulary creation takes the capabilities of tools for working with SKOS vocabularies to a new level of sophistication by offering an advanced data storage system and a graphical user interface.

State of the art

In 2012, Manaf et al. present a survey of the current state of SKOS vocabularies on the Web. A total of 478 vocabularies were identified as complying with the given definition of a SKOS vocabulary. Analyses of those vocabularies include investigation of the use of SKOS constructs, the use of SKOS semantic relations and lexical labels, and the structure in terms of the hierarchical and associative relations, branching factors, and depth of the vocabularies.

The researchers collect the following data on each SKOS vocabulary: number of SKOS concepts, depth of each SKOS concept and depth of the concept hierarchy, number of links for **`skos:broader`**, **`skos:narrower`** and **`skos:related`** properties, total number of concepts not connected to any other concepts, total number of concepts with **`skos:narrower`** relations but no **`skos:broader`** relations, and maximum number of **`skos:broader`** properties. According to the researchers, SKOS concepts and concept labeling is core to SKOS vocabularies, but not all SKOS vocabularies in the study use SKOS lexical labels for their concepts. Approximately one-third of the SKOS vocabularies studied fall into the category of term lists, with no use of any SKOS semantic relations. The researchers find that not all published SKOS vocabularies explicitly declare SKOS concepts present in the vocabularies. The survey results can serve to provide a better understanding of the modeling styles of the SKOS vocabularies published on the Web, especially when considering the creation of applications that utilize these vocabularies.

Future Research

Currently, there are various initiatives which examine alternate and future uses of SKOS. Putkey (2011) discusses the use of SKOS in order to express the hierarchical and associative relationships in faceted classification, that is, the organization of objects into facets, sub-facets and facet values on the Semantic Web. In her research, she discusses the use of the PoolParty application to create SKOS-compliant faceted classification schemes. Putkey also mentions that future research could examine SKOS expressed purely in triples or other alternatives rather than in the more common RDF/XML. In addition, the National Information Standards Organization (NISO) Bibliographic Roadmap Project is currently engaged in discussion about interoperability across different standards and technologies, including SKOS.[15] Finally, Zou (2013) concludes that a future robust infrastructure for building and aggregating metadata registries would allow vocabularies in multiple formats, including SKOS, to be interoperable and synchronized with one another.

Conclusion

The purpose of this guide is to bring a better understanding of the SKOS to the community of librarians, catalogers, and information professionals. SKOS has been brought into context of other RDF-related topics through discussion of the history of classification technologies and the foundational standards upon which SKOS is based. As a W3C standard designed for the representation of controlled vocabularies on the Web, the aim of SKOS technology is to enable controlled vocabularies to be consumed on the Web. Through an understanding of the elements, syntax, and integrity conditions of

[15] "Work to make vocabularies work across systems." Last modified December 2013. http://niso.ideascale.com/a/dtd/Work-to-make-vocabularies-work-across-systems/539645-18685.

SKOS and the growth and current state of SKOS technology development, members of the wider information professional community will be able to advance this field and use SKOS in ways previously unexplored.

References

Abbas, June. *Structures for Organizing Knowledge: Exploring Taxonomies, Ontologies, and Other Schemas*. New York: Neal-Schuman Publishers, Inc., 2010.

Assem, Mark van, Véronique Malaisé, Alistair Miles, and Guus Schreiber. “A Method to Convert Thesauri to SKOS.” *Lecture Notes in Computer Science* 4011 (2006): 95-109. doi: 10.1007/11762256_10.

Gerbé, Olivier and Brigitte Kerhervé. “A Model-Driven Approach to SKOS Implementation.” Paper presented at the International Conference on Internet and Web Applications and Services (5th), St. Maarten, AN., 2010. doi:10.1109/ICIW.2010.79.

Kaltenböck, Martin and Florian Bauer. “Linked Open Data: The Essentials.” Last modified 2012. http://www.semantic-web.at/LOD-TheEssentials.pdf

Manaf, Nor Azlinayati, Sean Bechhofer and Robert Stevens. “The Current State of SKOS Vocabularies on the Web.” *Lecture notes in Computer Science* 7295 (2012): 270-284. doi:10.1007/978-3-642-30284- 8_25.

Miles, Alistair, Brian Matthews, Dave Beckett, Dan Brickley, Michael Wilson and Nikki Rogers. “SKOS: a language to describe simple knowledge structures for the web.” Paper presented at the XTech 2005: XML, the Web and Beyond conference (5th) Amsterdam, NL, 2005.

http://epubs.cclrc.ac.uk/work-details?w=33893.

National Information Standards Organization Bibliographic Roadmap Project Input Forum. “Work to Make Vocabularies Work Across Systems.” Last modified December 2013. http://niso.ideascale.com/a/dtd/Work-to-make-vocabularies-work-across-systems/539645-18685.

Neubert, Joachim. “Bringing the ‘Thesaurus for Economics’ on to the Web of Linked Data.” Paper presented at the WWW Workshop on Linked Data on the Web (2nd) Madrid, ES., 2009. http://citeseerx.ist.psu.edu/viewdoc/similar;jsessionid=36C9F3DE0244DA85D5CFAD2A744E771D?doi=10.1.1.184.2819&type=ab.

Putkey, Theresa. “Using SKOS to Express Faceted Classification on the Semantic Web.” *Library Philosophy and Practice* 620 (2011). http://digitalcommons.unl.edu/libphilprac/620/.

Semantic Web Advanced Development for Europe. “Migrating Thesauri to the Semantic Web: Guidelines and Case Studies for Generating RDF Encodings of Existing Thesauri.” Last modified 2001. http://www.w3.org/2001/sw/Europe/reports/thes/8.8/.

Semantic Web Advanced Development for Europe. “SKOS-Core 1.0 Guide: An RDF Schema for Thesauri and Related Knowledge Organisation Systems.” Last modified 2001. http://www.w3.org/2001/sw/Europe/reports/thes/1.0/guide/20040504/.

Summers, Ed, Antoine Isaac, Clay Redding and Dan Krech. “LCSH, SKOS, and Linked Data.” Paper contributed to the International Conference on Dublin Core and Metadata Applications (8th) Berlin, DE., 2008. http://arxiv.org.libproxy.lib.unc.edu/abs/0805.2855.

Suominen, Osma and Eero Hyvönen. “Improving the Quality of SKOS Vocabularies with Skosify.“ *Lecture* *notes in Computer Science* 7603 (2012): 383-397. doi:10.1007/978-3-642-33876-2_34.

W3C Semantic Web Deployment Working Group. “Development & Participation.” Last modified January 01, 2012. http://www.w3.org/2004/02/skos/development.

W3C Semantic Web Deployment Working Group. “public-swd-wg@w3.org Mail Archives.” Accessed November 5, 2012. http://lists.w3.org/Archives/Public/public-swd-wg/.

W3C Semantic Web Deployment Working Group. “Simple Knowledge Organization System (SKOS).” Accessed November 4, 2012. http://www.w3.org/2001/sw/wiki/SKOS.

W3C Semantic Web Deployment Working Group. “SKOS Simple Knowledge Organization System Primer.” Last modified August 18, 2009. http://www.w3.org/TR/skos-primer.

W3C Semantic Web Deployment Working Group. “SKOS Simple Knowledge Organization System Reference.” Last modified August 18, 2009. http://www.w3.org/TR/skos-reference.

W3C Semantic Web Deployment Working Group. “SKOS Use Cases and Requirements.” Last modified August 18, 2009. http://www.w3.org/TR/skos-ucr.

Zou, Qing. “Building a Platform for Linked Open Thesauri.” *Library Hi Tech* 31 (2013): 620-637. doi:10.1108/LHT-03-2013-0022.

Appendix I.

Foundational Standards and Context

SKOS is a data-sharing standard and was built upon several pre-existing Semantic Web standards for formal logic and structure. These technologies provide ways of expressing meaning that are amenable to computation and that complement and give structure to information already existing on the Web. The terms and definitions in this appendix aim to provide a context for how SKOS fits into the wider Semantic Web vision.

XML

The eXtensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human- and machine-readable. XML documents use markup tags to describe elements of a given type of content. The language is "extensible" because markup tags are not predefined and must be invented by some human author. The markup describing an element's content may include attributes that aid in description of that content. For example:

```
<?xml version="1.0" encoding="utf-8" ?>
    <vegetable>
        <name_of_vegetable lang="eng">Carrot</name_of_vegetable>
        <name_of_vegetable lang="lat">Daucus carota</name_of_vegetable>
    </vegetable>
```

Line one of the code includes an XML declaration, which declares information about the type of document that follows. Line two includes a markup tag of an element called *vegetabl*e. Line three includes a child element with some content (the word "Carrot"). This content is described as a name of the parent element (vegetable), and it is given an attribute (lang) of English. Line four includes a child element with some content (the term "Daucus carota"). This content is described as another name of the parent element (vegetable), and it is given an attribute (lang) of Latin. Line five includes a closing tag which signals the end of the element.

RDF

The Resource Description Framework (RDF) is a metadata data model and method for conceptual description of information that provides a common syntax for the Web. Using various languages such as

RDFS or OWL (see below), RDF can be implemented in Web resources by way of triples. A triple is a subject-predicate-object statement about a Web resource, which helps to describe that resource. For example, “a carrot is a vegetable” is a triple: a subject (“carrot”), a predicate (“is a”), and object (“vegetable”). RDF triples are unique because each component is associated with a unique Uniform Resource Identifier (URI). A collection of these RDF statements can form a powerful data model that can be used for many types of information organization and management. Take, for example, the triple

```
http://www.veggievocab.com/vegetable/root/carrot
<http://purl.org/dc/elements/1.1/title> "Carrot" .
```

This triple statement means that the resource http://www.veggievocab.com/vegetable/root/carrot has a title of “Carrot.” The subject of the triple is a URI for a term in a vocabulary (the veggie vocab). The predicate of this triple is a URI for the Dublin Core metadata element “title.” The object of this triple is the string “Carrot.” This same triple might be expressed in RDF as:

```
<rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:dc="http://purl.org/dc/elements/1.1/">
<rdf:Description rdf:about="http://www.veggievocab.com/vegetable/root/carrot">
            <dc:title>Carrot</dc:title>
        </rdf:Description>
</rdf:RDF>
```

RDFS

The Resource Description Framework Schema (RDFS) is a formally defined knowledge representation language that provides a common data modeling language for data on the Web. It can also be thought of as a “semantic extension” to RDF. Information represented in RDFS is described through classes and properties of those classes.

OWL

Like RDFS, the Web Ontology Language (OWL and OWL 2) is another, more expressive knowledge representation language that also provides a common data modeling language for data on the Web. While fully compatible with RDFS, OWL is also able to augment the meaning of existing RDFS vocabularies. In addition, OWL includes several variants, or sub-languages, including OWL Lite, OWL DL, and OWL Full. OWL Full was designed to be compatible with RDFS, and it allows an ontology to extend the meaning of a given vocabulary. The following example uses OWL 2 syntax to declare that the veggie vocab is an ontology.

```
Ontology(<http://www.veggievocab.com/vegetable.owl>
  Declaration( Class( :Vegetable ) )
)
```

Turtle

The Terse RDF Triple Language (Turtle) is a textual syntax for RDF that allows RDF statements to be written in compact and natural text format, with abbreviations for common usage patterns and data types. The example shows the use of Turtle syntax stating a triple using a SKOS relationship.

```
<http://www.veggievocab.com/vegetable/root>
<http://www.w3.org/2004/02/skos/core#broader>
<http://www.veggievocab.com/vegetable/root/carrot> .
```

SPARQL

The SPARQL Protocol and RDF Query Language (SPARQL) is an RDF query language that provides a standard means for interacting with data on the Web. This language allows for the retrieval and manipulation of data stored in RDF format. The following example demonstrates a query in which the user requests some information about a specific term in the veggie vocab. In this example the request is for the color of the term "Carrot."

```
PREFIX veggieVocab: <http://www.veggievocab.com>
SELECT ?vegetable ?color
WHERE {
  ?x abc:colorname ?color ;
     abc:isColorOf ?y .
  ?y abc:vegetablename ?vegetable ;
     abc:isColorOf abc:Carrot .
}
```

Line one of this SPARQL statement specifies the vocabulary from which to pull information. Line two specifies the specific type of information which will be queried. Line three signals the beginning of the query. Lines four, five, and six specify the way in which the information specified in line two will be queried, and line seven specifies the precise query "What color is a carrot?"

The application of any of these technologies over large bodies of information requires the construction of detailed maps of particular knowledge domains and the accurate description of information resources on a large scale. Most of this work cannot be done automatically, and this is where SKOS comes into play. Simply put, the SKOS data model is an OWL Full ontology and its data are expressed as RDF triples.