Common Slips in SKOS Vocabularies

Nor Azlinayati Abdul Manaf, Sean Bechhofer, and Robert Stevens

School of Computer Science, The University of Manchester, United Kingdom
{abdulman,seanb,robert.stevens}@cs.man.ac.uk

Abstract. Following our analysis of SKOS vocabularies publicly available on the Web, we found several types of 'irregularity' in some of the vocabularies' representation [1]. We have considered these 'defects' as *slips* made by vocabulary engineers when authoring the vocabularies. In many cases these slips are apparently due to either syntactic error or accidental misuse of the SKOS core vocabulary. In this paper, we present a typology of these slips, and describe possible *patches* that can be applied to deal with the problems. By 'patching' these slips, vocabularies can be adjusted to conform to the SKOS standard and thus enable the usage of SKOS tools/applications.

1 Introduction

The Simple Knowledge Organization System (SKOS)[1] provides vocabulary to represent traditional knowledge organization systems (KOSs) that are often used for retrieval and navigation systems. Such representations include thesauri, subject headings, classification schemes, taxonomies, glossaries and other structured controlled vocabularies [2]. As described in [3], SKOS itself is not a language for ontologies. However, the SKOS data model, which is described using vocabulary taken from RDF and OWL, is in fact described as an OWL ontology. The `skos:Concept` and `skos:ConceptScheme` are defined as OWL Classes, the SKOS semantic relations such as `skos:broader`, `skos:narrower` and `skos:related` are OWL Object Properties and the labelling and documentation properties are OWL Annotation Properties. Since a SKOS vocabulary is a representation of a particular KOS as an instantiation of the SKOS data model ontology, therefore all SKOS vocabularies are OWL documents that describe KOSs artefacts.

Since SKOS was accepted as a W3C Recommendation, a number of SKOS tools have been developed to interact with and manipulate SKOS vocabularies. Some of the relevant tools are listed in the SKOS wiki page[2]. In [1] we report an apparently low number of SKOS vocabularies on the Web. Some of this small corpus of vocabularies did not pass our test of a valid SKOS vocabulary and so will not meet the expectations of SKOS tools. Here we report on an experiment that analyses available SKOS vocabularies. We describe a typology of defects in these vocabularies and the patches that can be used to make them valid SKOS vocabularies.

[1] `http://www.w3.org/2004/02/skos/`
[2] `http://www.w3.org/2001/sw/wiki/SKOS`

2 Materials and Methods

The results from our previous survey of SKOS on the Web [1] revealed that a small number of URLs that were expected to be identified as a SKOS vocabulary failed our test of validity. Further investigation shows some irregularities in the representation of these vocabularies, caused by apparent 'mistakes' by ontology authors, which we called 'slips'. We detect and classify these slips through the following steps:

1. Preparation of a corpus of candidate SKOS vocabularies.
2. Validation of the candidates as SKOS vocabularies.
3. Detection and classification of detected 'slips'.

Apparatus All experiments were performed on a 2.4GHz Intel Core 2 Duo MacBook running Mac OS X 10.6.8 with a maximum of 3 GB of memory allocated to the Java virtual machine. Two reasoners were used: JFact[3], which is a Java version of the FaCT++ [4] reasoner, and Pellet [5]. We used the OWL API [6] version 3.2.4[4] for handling and manipulating the vocabularies.

Preparing a corpus of candidate SKOS vocabularies As described in [1], the SKOS vocabularies came from several sources. The first collection of candidate SKOS vocabularies was gathered from two dedicated collections, the SKOS Implementation Report[5] and the SKOS/Datasets[6]. The second collection of candidate SKOS vocabularies was gathered by utilising Semantic Web search engines such as Swoogle [7] and Watson [8].

Validating SKOS vocabularies All candidate SKOS vocabularies gathered in the previous step were tested for SKOS 'validity'. We proposed in [1] this definition of SKOS vocabulary:

Definition 1. *A SKOS vocabulary is a vocabulary that at the very least contains SKOS concept(s) used directly, or SKOS constructs that indirectly infer the use of a SKOS concept, such as SKOS semantic relations.*

Each candidate SKOS vocabulary was screened in the following way to identify it as a SKOS vocabulary:

1. Check for the existence of direct instances of type `skos:Concept`; if Yes, then accept the vocabulary as a SKOS vocabulary.
2. Check for the existence of implied instances of `skos:Concept` due to domain and range restrictions on SKOS relationships (for example the subject of a `skos:broader`, `skos:narrower` or `skos:related` relationship is necessarily a `skos:Concept`); if Yes, then accept the vocabulary as a SKOS vocabulary.

[3] `http://jfact.sourceforge.net/`
[4] `http://owlapi.sourceforge.net/`
[5] `http://www.w3.org/2006/07/SWD/SKOS/reference/20090315/implementation.html`
[6] `http://www.w3.org/2001/sw/wiki/SKOS/Datasets`

3. Otherwise, do not accept this vocabulary as a SKOS vocabulary.

Consider the following vocabulary snippets written in Manchester Syntax[7]. `Vocabulary 1` and `Vocabulary 2` are accepted as SKOS vocabularies based on tests in Step 1 and Step 2, respectively. Meanwhile, `Vocabulary 3` is not accepted as a SKOS vocabulary according to our definition, even though this vocabulary uses SKOS constructs such as `skos:prefLabel` and `skos:altLabel`. If Vocabulary 3 was to be included as a SKOS vocabulary, one could expect any ontology that uses SKOS annotation properties for labelling their entities to be included in the survey.

```
Vocabulary 1:
Individual: Emotion
  Types:
    Concept
Individual: Love
  Types:
    Concept
Individual: Beauty
  Types:
    Concept
```

```
Vocabulary 2:
Individual: Love
  Types:
    Thing
  Facts:
    broader Emotion
Individual: Emotion
  Types:
    Thing
```

```
Vocabulary 3:
Individual: Love
  Types:
    Thing
  Facts:
    prefLabel "Love",
    altLabel "Affection"
```

Detecting and classifying slips In the SKOS vocabulary validation stage, we recorded the list of URLs of candidate SKOS vocabularies that do not pass the 'SKOS validity test'. These URLs are then screened for SKOS constructs used in the vocabularies using the OWLAPI. In this screening stage, we are looking for candidate SKOS vocabularies that use to following SKOS constructs, but were failed to be detected in the previous stage.

- `skos:broader`
- `skos:narrower`
- `skos:related`
- `skos:Concept`
- `skos:hasTopConcept`
- `skos:topConceptOf`

Utilising the functionality provided by the OWL API, we then checked the entity types of the listed SKOS constructs recognised by the OWL API and recorded them for each candidate SKOS vocabulary.

We also kept a list of the URLs of candidate SKOS vocabularies that were inconsistent when we classify with an automatic reasoner such as JFact or Pellet. For each of the candidate SKOS vocabularies that failed to be classified, we recorded the exception message thrown by the reasoner together with the cause of the inconsistency.

Each *impaired* SKOS vocabularies was manually inspected for any sign of deviation from SKOS that would account for the irregularity. We classified these irregularities into several types.

[7] `http://www.w3.org/TR/owl2-manchester-syntax/`

3 Typology of slips

Out of 6819 URLs in the corpus, 5751 URLs failed to be validated as a SKOS vocabulary [1]. Out of this number, 2986 URLs were plain HTML pages/blogs/forum page URIs. The second largest portion of the URLs (1199 URLs) were actually OWL documents, but failed the SKOS validity test. Almost all of these OWL documents used at least one of the SKOS constructs from the SKOS labelling and documentation properties. 93 URLs referred to the actual SKOS Core data model[8], while the remaining 2087 were 'unreachable documents' due to 'connection refused', network problems and 'failed to load import ontology' errors.

There were 47 URLs identified in the slips detection stage, with 18 documents detected using the listed SKOS constructs, and the rest were 'inconsistent' ontologies. Table 1 shows a summary of the results.

Table 1. Summary results specifying the number of SKOS vocabularies

Stages	vocabs
Corpus preparation	6819
URIs not validated as SKOS vocabularies	5751
- Plain HTML/blogs/forum	2986
- OWL documents that are not SKOS vocabularies	**1199**
- Actual SKOS Core data model	93
- Others	2087
Detecting and classifying slips	**47**
- Listed SKOS constructs in use	18
- Inconsistent ontologies	29

We classified the types of slips into three categories as follows:

1. Type 1: Undeclared property type.
2. Type 2: Mis-use of SKOS constructs.
   (a) Mistyping of an individual to be an instance of both `skos:ConceptScheme` and `skos:Concept`.
   (b) Incorrect use of `skos:narrower` property to relate a concept to a collection.
3. Type 3: Use of an invalid or user-defined datatype.

3.1 Type 1: Undeclared property type.

The information regarding the entity types returned by the OWLAPI, revealed that all SKOS properties such as `skos:broader`, `skos:narrower`, etc. are of type `owl:AnnotationProperty`. Further inspection of the source of the vocabulary showed that each SKOS property used in the vocabulary was not explicitly typed as any of the possible property types such as `owl:ObjectProperty`,

[8] `http://www.w3.org/2004/02/skos/core.rdf`

`owl:DataProperty` or `owl:AnnotationProperty`. Note that the SKOS specification [2] does not enforce explicit declarations. In fact this type of slip is a consequence of managing and processing the SKOS vocabularies using tools such as the OWLAPI, which due to OWL 2 DL perspective require explicit declarations of properties used in the OWL documents. An example of this type of slip is shown in Figure 1. 18 candidate SKOS vocabularies were classified to have this type of slip.

```
AnnotationProperty(http://www.w3.org/2004/02/skos/core#broader)
AnnotationProperty(http://www.w3.org/2004/02/skos/core#narrower)
```

Fig. 1. A snippet of SKOS vocabulary with a Type 1 slip

3.2 Type 2: Mis-use of SKOS constructs.

This type of slip was identified through the exception thrown by the reasoner when it failed to classify the vocabularies. We found 6 candidate SKOS vocabularies that were inconsistent, caused by a 'mis-use' of SKOS constructs. From our inspection of the SKOS constructs usage in the vocabularies, we can categorised this type of slip into 2 categories.

(a) Mistyping of an individual to be an instance of both `skos:ConceptScheme` and `skos:Concept`. The SKOS Reference [2] has defined that the `skos:ConceptScheme` and `skos:Concept` classes are disjoint. This means that in a SKOS vocabulary, an individual cannot be an instance of both classes at the same time without the ontology being inconsistent. Five SKOS vocabularies were found to be inconsistent due to having a condition where one individual had been declared as a `skos:Concept`, and the `skos:inScheme` property was used to relate other SKOS concepts to this individual. Since the `rdfs:range` of `skos:inScheme` is the class `skos:ConceptScheme` as defined in the SKOS Reference, this individual was indirectly defined as type `skos:ConceptScheme` through the use of the `skos:inScheme` property. Figure 2 shows a snippet of a vocabulary that illustrates the situation for this type of slip. 5 candidate SKOS vocabularies were classified to have this type of slip.

(b) Incorrect use of a `skos:narrower` property to relate a concept to a collection. The SKOS data model provides the property `skos:narrower` to show a hierarchical relationship between SKOS concepts. For example, assertion A `skos:narrower` B means that concept A has a narrower concept B. However we found 1 vocabulary that used the property `skos:narrower` to relate a concept to a collection. The classes `skos:Concept` and `skos:Collection` are defined as

```
Individual: urn:cgi:classifierScheme:CGI:StratigraphicRank:200811
    Types:
        Concept

Individual: urn:cgi:classifier:CGI:StratigraphicRank:200811:lithodeme
    Types:
        Concept
    Facts:
        inScheme  urn:cgi:classifierScheme:CGI:StratigraphicRank:200811,
        prefLabel  "Lithodeme"@en
```

Fig. 2. A snippet of SKOS vocabulary with a Type 2 slip

disjoint classes in the SKOS data model. Therefore, since the `rdfs:domain` of the `skos:narrower` property is `skos:Concept`, using a `skos:narrower` to relate a concept to a collection will violate this constraint, causing the vocabulary to be inconsistent. In the SKOS data model, the correct property to use to relate a member to a collection is `skos:member`. Further inspection of the vocabulary also showed that the `skos:member` property was declared but never used. Figure 3 shows a snippet of a vocabulary with this type of slip. 1 candidate SKOS vocabulary was classified to have this type of slip.

```
Class: Collection
    Individuals:
        _:genid1

Individual: milk
    Types:
        Concept
    Facts:
        narrower  genid1,
        prefLabel  "milk"@
```

Fig. 3. A snippet of SKOS vocabulary with Type 2 slips

3.3 Type 3: Use of an invalid or use-defined datatype.

This type of slip was also identified based on an exception thrown by the reasoner when it failed to classify the vocabularies. This type of slip was due to a user-defined or invalid datatype not being recognised by a reasoner. Besides, the use of user-defined datatypes is not a problem from the SKOS point of view,

instead it is a problem in the context of OWL 2 DL. We found 23 candidate SKOS vocabularies having this type of slip, 9 vocabularies due to user-defined datatypes and 14 vocabularies due to invalid datatypes.

4 Patching

As reported in the previous section, some of the slips are merely syntactic errors which could be fixed by 'simple' patching to the syntax, while others may require some judgement in order to avoid altering the intended meaning of the SKOS vocabulary. In this section, we introduce approaches to fix these slips.

4.1 Type 1: Undeclared property type.

There are two possible *patches* to fix the slip described in Section 3.

1. Patch 1: Addition of missing declarations.
   Search for all SKOS-related constructs in the vocabulary and add the missing declarations for these constructs. For example, if the properties `skos:broader`, and `skos:narrower` were found in the vocabulary, we would add declarations for both of these properties to be of type `owl:ObjectProperty`.
2. Patch 2: Import the SKOS core vocabulary.
   Another possible approach to *fix* this problem is by importing the SKOS core vocabulary[9].

Applying either *patch* fixed the problem. We applied the Patch 1 *fixing* procedure and fixed 18 SKOS vocabularies of this category.

4.2 Type 2: Mis-use of SKOS constructs.

Mistyping of an individual to be instances of both `skos:ConceptScheme` and `skos:Concept` classes.

To 'fix' this type of slip, we propose the following procedures:

1. Search the vocabulary for the mentioned individual X.
2. Check the existence of axiom(s) relating other SKOS concept(s) to individual X through `skos:inScheme` property. For example, `<Y> skos:inScheme <X>`.
   If Yes, this indicates that individual X, is inferred to be of type `skos:ConceptScheme`.
3. Check if the existing declaration for individual X as type `skos:Concept`. If Yes, then remove this declaration from the vocabulary.

Applying these 'fixing' procedures fixed the five SKOS vocabularies in this category.

[9] `http://www.w3.org/2004/02/skos/core.rdf`

Incorrect use of `skos:narrower` property to relate a concept to a collection.

To 'fix' this type of slip, we propose the following procedures:

1. Search the vocabulary for the mentioned individual X.
2. Check if the existing declaration for individual X as type `skos:Collection`.
3. Check for the existence of axiom(s) relating other SKOS concept(s) to individual X through `skos:narrower` property. For example, `<Y> skos:narrower <X>`. If Yes, this indicates that individual X, is inferred to be of type `skos:Concept`.
4. Then replace the axiom `<Y> skos:narrower <X>` by `<Y> skos:member <X>`.

Applying these 'fixing' procedures fixed the one SKOS vocabulary in this category.

4.3 Type 3: Use of an invalid or user-defined datatype.

The patch this type of slip, we do the following. For the user-defined datatype problem, we first checked whether the user-defined datatype was actually in use to type the data in the vocabulary. If the datatype was not in use, we excluded the datatype from the datatype list and reclassified the vocabulary. For the invalid datatypes problem, further judgement was needed in fixing this problem. We fixed 0 vocabularies for this type of slip.

The number of SKOS vocabularies for each type of slips and the results of patching are summarised in Table 2

Table 2. Summary of types of slips and their patching

Types	Total vocabularies	Fixed	Unfixed
Type 1: Undeclared property type	18	18	0
Type 2: Mis-use of SKOS constructs	6		
- Type 2a: Mistyped an individual	5	5	0
- Type 2b: Incorrect use of `skos:narrower`	1	1	0
Type 3: Use of invalid datatype	23	0	23

5 Discussion

We conjecture that some of these slips are merely due to authoring tools used by the ontology engineer to author the vocabulary, while some others are due to mistakes in authoring the vocabulary.

The 'patching' procedures proposed in Section 4 could be done manually or implemented through an automated 'patching' process. Having an automated 'patching' process would make the vocabulary 'repairing' more scalable than the manual approach we used. These procedures could also be incorporated into SKOS parsers or validator tools to help avoid errors at source.

Efforts in detecting and patching errors in Semantic Web documents were described in [9, 10]. However, these works focused on OWL ontologies rather than KOS style artefacts. The PoolParty Consistency Checks for SKOS Thesauri[10] provides an on-line service for checking integrity and consistency of a given SKOS thesaurus. At the end of the checking process, the check result will be displayed, including error(s), if any. So the survey and repairs we report here fills a potentially useful gap for authors of semantic artefacts.

5.1 Recommendations for SKOS Vocabulary Best Practices

We reported in [1], several issues regarding usage of SKOS constructs such as undeclared SKOS concepts, use of `skos:broader` and `skos:narrower` properties, usage of SKOS lexical labels, etc. Based on our discussion of slips in this paper and the SKOS constructs usage described in [1], we propose the following SKOS vocabulary best practices. We give recommendations for Best Practices for both vocabulary engineers (VE) and application designers (AD) point of views.

1. **Declare SKOS Concept.**
   VE: Declare all concepts used in the vocabulary as type `skos:Concept`. This is because some SKOS tools like SKOS Reader rely heavily on this declarations as the first step in identifying instances in the vocabulary. Without this declaration, this kind of tool is not able to display the vocabulary correctly.
   AD: If no SKOS concept were found, look for use of SKOS semantic relations that can infer the use of SKOS concept through the domain and range constraints.
2. **Use of `skos:broader` and `skos:narrower` properties.**
   VE: Whenever an assertion is made between two concepts using either the `skos:broader` or `skos:narrower` property, always add the inverse relation for this assertion.
   AD: When found only one assertion of `skos:broader` or `skos:narrower` property between two SKOS concepts, always infer the inverse relation for this assertion.
3. **Import SKOS core vocabulary.**
   VE: The best way to avoid slip Type 1 is to import the SKOS core vocabulary[11]. I doing so, the problem of missing type will be solved.
   AD: Similarly, an application designer could import SKOS core vocabulary when loading the vocabulary and use the reasoner to classify the vocabulary.

For Best Practices 1 and 2, the application designers could also consider applying a reasoner to classify the vocabularies to be used with their tools [11]. For Best Practice 1, if the vocabularies do not typed their SKOS concepts, but instead used SKOS semantic relations like `skos:broader`, the knowledge about this `skos:Concept` could be inferred from domain and range constraints

[10] `http://demo.semantic-web.at:8080/SkosServices/check`
[11] `http://www.w3.org/2004/02/skos/core.rdf`

of these relations. Similarly, as shown in our previous survey [1], not all SKOS vocabularies asserted the relationships, like `skos:broader` and `skos:broader` in both ways. For this type of SKOS vocabulary, if the tools do not apply a reasoner, it will not get the inverse relation of either `skos:broader` or `skos:narrower` relations. Thus, applying a reasoner before using the vocabulary could be a best practice for the application designers.

6 Conclusion

We have presented a typology of slips in SKOS vocabularies on the Web. We found that some of these slips are syntactic errors, while others appear to be mistakes in understanding the use of SKOS constructs by vocabulary engineers. We admit that the types of slips presented in this paper are based on a rather small collection of SKOS vocabularies in our corpus. We showed that some of these slips can be corrected by applying 'simple' patching procedures, while others require some judgement and further knowledge of the intentions of the vocabulary engineers. We also discussed other issues from the survey in [1] and proposed recommendations for Best Practices in SKOS vocabularies for both ontology engineers and application developers. By patching these slips, we are able to handle more vocabularies, which are consequently made conformant to the SKOS standards and thus enable the usage of SKOS tools/applications.

Acknowledgement: Nor Azlinayati Abdul Manaf is in receipt of a scholarship from Majlis Amanah Rakyat (MARA), an agency under the Malaysian Government, for her doctorate studies.

References

1. Abdul-Manaf, N.A., Bechhofer, S., Stevens, R.: The current state of SKOS vocabularies on the Web. In: Proceedings of the 9th Extended Semantic Web Conference (ESWC2012). (May 2012)
2. Miles, A., Bechhofer, S.: SKOS simple knowledge organization system reference. W3C recommendation, W3C. (2009)
3. Jupp, S., Bechhofer, S., Stevens, R.: SKOS with OWL: Dont be Full-ish! In: Proceedings of the Fifth OWLED Workshop on OWL: Experiences and Directions, collocated with the 7th International Semantic Web Conference (ISWC-2008), Karlsruhe, Germany, October 26-27, 2008. (2008)
4. Tsarkov, D., Horrocks, I.: FaCT++ description logic reasoner: System description. In Furbach, U., Shankar, N., eds.: IJCAR. Volume 4130 of Lecture Notes in Computer Science., Springer (2006) 292297
5. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. Journal of Web Semantics 5(2) (2007) 5153
6. Horridge, M., Bechhofer, S.: The OWL API: A Java API for OWL ontologies. Journal of Semantic Web 2(1) (2011) 1121

7. Finin, T., Peng, Y., Scott, R., Joel, C., Joshi, S.A., Reddivari, P., Pan, R., Doshi, V., Ding, L.: Swoogle: A search and metadata engine for the semantic web. In: Proceedings of the Thirteenth ACM Conference on Information and Knowledge Management, ACM Press (2004) 652659
8. d'Aquin, M., Baldassarre, C., Gridinoc, L., Angeletou, S., Sabou, M., Motta, E.: Watson: A gateway for next generation semantic web applications. Poster, ISWC 2007, (2007)
9. Bechhofer, S., Carroll, J.J.: Parsing OWL DL: trees or triples? In Feldman, S.I., Uretsky, M., Najork, M., Wills, C.E., eds.: WWW, ACM (2004) 266275
10. Bechhofer, S., Volz, R.: Patching syntax in OWL ontologies. In: Proceedings of the 3rd International International Semantic Web Conference. (2004)
11. Solomou, G.D., Koutsomitropoulos, D.A.: Support of SKOS vocabularies in the DSpace digital repository system. In DSpace Federation 5th User Group Meeting (Poster) (2009)