# RDFS vs SKOS, when to use what?

Asked 3 years ago  Modified 2 years, 5 months ago  Viewed 2k times

8

As I'm learning semantic-web & sparql, sensing that RDFS & SKOS seem to offer very similar semantic relations modeling capabilities. For example,

- RDFS - rdfs:subClassOf, rdfs:superClassOf can be used to model the hierarchy
- SKOS - skos:narrower, skos:broader can be used to model the hierarchy

Both offer 2-way transitivity.

Though

- SKOS offers more explicit properties to model transitivity, related relationships and matching thru skos:narrowerTransitive, skos:broaderTransitive, skos:related, skos:closeMatch, etc

1. Is this correct understanding?
2. Is there any guidance to pick the right pattern while modeling?
3. If I consider that skos semantics offer above said advantages, Why does dbpedia uses a lot of rdfs vs skos?

Thanks!

semantic-web rdfs skos

Share

Improve this question

Follow

asked Nov 28, 2021 at 16:22

Sahas

3,186 ● 6 ● 38 ● 59

## 2 Answers

Sorted by: Highest score (default)

10

RDFS and SKOS can be easily used side by side, because their focus is somewhat different.

RDFS is something that directly emerges from the RDF data model and its serializations. It starts with the RDF inference/entailment regime that models the most basic thing anyone could infer from a triple `a b c` – that `b` is a property, thus we need `rdf:type` and `rdf:Property` to express that fact (that's why those two are not in `rdfs:`). One step further, we have RDFS that again starts at `rdf:type` (making its range a class) and `rdf:Property` (inferring it is a class, and allowing describing subproperties). Then we have OWL that allows even greated degree of inference, but also makes it possible to detect (or create) contradictions.

RDFS is focused on inference; all of its classes and properties are centered around it: `rdfs:domain` and `rdfs:range` are defined in terms of inference of `rdf:type`; `rdfs:subClassOf` and `rdfs:subPropertyOf`

are defined in terms of inference of arbitrary properties etc. Of course (along with OWL) one can also use it to describe a vocabulary itself, making it a sort of meta-RDF language (but, unlike OWL, it can describe itself).

RDFS is similar to classical object-oriented languages, but it also makes it possible to express classes of classes etc. There is also a difference between "being an instance of" (`rdf:type`) and "being a subclass of" (`rdfs:subClassOf`).

SKOS is oblivious to classes and properties as seen by RDF, as it stems from what we humans observe around us: we don't usually talk about a "class of cars", we talk about a car. Sets and classes are left to logicians and philosophers, and we are left with *concepts*. Concepts are the things we see around us, some apply to a broad range of things (a car, an electric car), some to just one thing (my neighbor's car). We don't need higher-order set theory to talk about them, and neither does SKOS, everything is described as an individual concept, some more general/abstract than others. Moreover, getting into higher orders is, in a sense, explicitly disallowed by SKOS by the rule that a concept is not a concept scheme.

SKOS may be set-theoretically limited but it is nonetheless powerful. Sometimes, it is easier to use it to talk about vague concepts (is "my neighbor's car" a class or an individual? SKOS doesn't care), sometimes it is necessary to describe concepts that don't fit in the RDFS

hierarchy at all (related, close/exact match), not necessarily for computers to understand but definitely for humans.

SKOS's model reminds me of JavaScript, where you don't have traditional classes but use prototypes to express inheritance, similarly to `skos:broader`. In this regard, a class is an object and an object is a class; there is no difference between an instance and a subclass.

Hopefully you see now that the answer to whether to use one or the other is to use *both*, in their respective places. You use RDFS/OWL for vocabularies or inference, when you need strict definitions, and you use SKOS for concepts/taxonomies, when you need vague definitions, but there are places where both are perfectly usable.

Share Improve this answer Follow

edited Jul 23, 2022 at 18:42

answered Dec 2, 2021 at 12:38

IS4

**13.1k** ● 2 ● 55 ● 90

1 @IS4- Thank you for a more easily understandable explanation – Sahas Dec 8, 2021 at 1:35

6

The main difference between RDFS and SKOS is outlined in the SKOS specs:

https://www.w3.org/TR/skos-reference/#L1045

> The elements of the SKOS data model are classes and properties, and the structure and integrity of the data model is defined by the logical characteristics of, and interdependencies between, those classes and properties. This is perhaps one of the most powerful and yet potentially confusing aspects of SKOS, because SKOS can, in more advanced applications, also be used side-by-side with OWL to express and exchange knowledge about a domain. However, SKOS is **not** a formal knowledge representation language.

Not being a formal knowledge representation language, inferences are not standardised and there might be less interoperability with other knowledge bases.

I can't speak for dbpedia as to the reasons for the choice, but this seems a good enough reason to me, so I wouldn't be surprised if this was part of them.

Share Improve this answer

Follow

answered Nov 28, 2021 at 21:36

Ignazio

**10.7k** ● 1 ● 16 ● 25