

ACTIVE

Deliverable <1 . 4 . 1>

Knowledge leveraging and repair Early prototypes

Editor:	Denny Vrandečić, AIFB
Deliverable nature:	Prototype (P)
Dissemination level: (Confidentiality)	Public (PU)
Contractual delivery date:	28 February 2009
Actual delivery date:	
Suggested readers:	Both researchers and practitioners in the area of leveraging and repairing formal knowledge
Version:	Interim
Total number of pages:	32
Keywords:	semantic technologies, ontology language, ontology evaluation, knowledge representation

Abstract

This deliverable reports on investigations of formal techniques for automatically (i) spotting expedient extensions and refinements to initial lightweight knowledge structures and leveraging the knowledge base accordingly and for (ii) detecting and repairing inconsistencies or invalidities in collaboratively maintained knowledge bases. Conceptually, this work will transfer results from the fields of ontology evaluation to the overall task setting. The first year presents early prototypes and how they can be used within the project case studies.

Disclaimer

This document contains material, which is the copyright of certain ACTIVE consortium parties, and may not be reproduced or copied without permission.

All ACTIVE consortium parties have agreed to full publication of this document.

The commercial use of any information contained in this document may require a license from the proprietor of that information.

Neither the ACTIVE consortium as a whole, nor a certain party of the ACTIVE consortium warrant that the information contained in this document is capable of use, or that use of the information is free from risk, and accept no liability for loss or damage suffered by any person using this information.

Impressum

[Full project title] ACTIVE – Knowledge-Powered Enterprise

[Short project title] ACTIVE

[Number and title of work-package] WP1 Enterprise Knowledge Structures

[Document title] Knowledge Leveragin and Repair – Early Prototypes

[Editor: Name, company] Denny Vrandečić, AIFB

[Work-package leader: Name, company] Markus Krötzsch, AIFB

[Estimation of PM spent on the Deliverable] Interim

Copyright notice

©2008-2012 Participants in project ACTIVE

Executive summary

This deliverable presents research results and applications in the area of evaluating knowledge structures. It introduces a number of novel approaches toward detecting and repairing problems in ontologies and ontology-like structures. A number of evaluation criteria are identified based on a literature survey. These criteria are consolidated and succinctly described. We describe the evaluation of the vocabulary aspect of web ontologies. With this evaluation we evaluate ontologies on the so-called *Web of Data* and report on the results. We describe a number of evaluation approaches that are implemented within a content quality check framework for Semantic MediaWiki. We allow these checks to be created and maintained by the users of the wiki themselves, not requiring further programming knowledge. We show a set of tools and approaches towards enabling the encoding and maintenance of consistency checks within the wiki itself.

List of authors

Company	Author
<AIFB>	<Denny Vrandečić>
<STI>	<Stephan Wölger>

Contents

Executive summary	3
List of authors	4
1 Introduction	8
2 Criteria	9
2.1 Accuracy	11
2.2 Adaptability	11
2.3 Clarity	11
2.4 Completeness	11
2.5 Computational efficiency	12
2.6 Conciseness	12
2.7 Consistency	12
2.8 Organizational fitness	12
3 Vocabulary	14
3.1 URI references	14
Linked data	15
Hash vs slash	17
Opacity of URIs	18
URI reuse	19
URI declarations and punning	19
3.2 Literals	20
Typed literals and datatypes	20
Language tags	22
Labels and comments	23
3.3 Blank nodes	23
3.4 Selection of the corpus	24
4 Content Evaluation in Semantic MediaWiki	25
4.1 Cardinality checks	25
4.2 Discovering redundancies	25
4.3 Statistical approaches	26
4.4 Social and usability aspects	26
5 Conclusions and outlook	28

List of Figures

- 3.1 Distribution of the HTTP response codes on the HTTP URIs from the Watson EA corpus. The left hand side shows the slash URIs, the right hand side hash URIs. 16
- 3.2 The fifteen most often used data types in the Watson corpus. Note the logarithmic scale. 21

List of Tables

- 3.1 An overview of what different response codes imply for the resolved HTTP URI reference *U*. *I* is the information resource that is returned, if any. *L* is the URI given in the location field of the response. The table covers the most important responses only, the others do not imply any further facts. 15
- 3.2 The five hash and slash namespaces with the biggest number of names. 17

Chapter 1

Introduction

This deliverable presents research results and applications in the area of evaluating knowledge structures. It introduces a number of novel approaches toward detecting and repairing problems in ontologies and ontology-like structures. As discussed in Deliverable D1.1.1, even though other knowledge structures are important in enterprise environments, we have shown that all enterprise knowledge structures can be represented by ontologies. Therefore in this deliverable we will speak of ontology evaluation, but mean the evaluation of all interesting enterprise knowledge structures.

A number of evaluation criteria are identified based on a literature survey in Chapter 2. These criteria are consolidated and succinctly described. The resulting criteria are:

- Accuracy
- Adaptability
- Clarity
- Completeness
- Computational efficiency
- Conciseness
- Consistency
- Organizational fitness

Chapter 3 describes the evaluation of the vocabulary aspect of web ontologies. The vocabulary is the most basic layer of ontologies. With this evaluation we can evaluate ontologies on the so-called *Web of Data*, in order to see if reusing such ontologies makes sense – or not. Our results are reported in Chapter 3, showing that the overall quality of ontologies on the Web of Data is indeed promising, and that we should reuse existing ontologies if possible – not only for their vocabularies, as it is done in Deliverable D1.1.1, but also as a data source and for connecting emerging data in the case studies with the Web of Data. In the first year of Active we have focused on the vocabulary aspect, but expect to have a more comprehensive report in the following two years.

In Chapter 4 we describe a number of evaluation approaches that are implemented within a content quality check framework for Semantic MediaWiki. Normal wikis hardly allow for any automatic evaluation of their content, but with Semantic MediaWiki a number of consistency checks can be introduced. It is important to allow these checks to be created and maintained by the users of the wiki themselves, not requiring further programming knowledge. We show a set of tools and approaches towards enabling the encoding and maintenance of consistency checks within the wiki itself.

The deliverable ends with an outlook on the plans for Y2 and Y3.

Chapter 2

Criteria

Ontology evaluation can target a number of several different criteria. In this chapter we will list criteria from literature, aggregate them to form a coherent and succinct set, and discuss their applicability and relevance for web ontologies. The goal of an evaluation is not to perform equally well for all these criteria – some of the criteria are even contradicting, like *minimal ontological commitment* and *completeness*. It is therefore the first task of the evaluator to choose the criteria relevant for the given evaluation and then to choose the proper evaluation methods to assess how well the ontology meets these criteria.

We selected five important papers from literature, where each defined their own set of ontology quality criteria or principles for good ontologies [20, 23, 24, 19, 43]. These quality criteria need to be regarded as desiderata, goals to guide the creation and evaluation of the ontology. None of them can be directly measured, and most of them cannot be perfectly achieved.

Asunción Gómez-Pérez lists the following criteria [20]:

- **Consistency:** capturing both the logical consistency (i.e. no contradictions can be inferred) and the consistency between the formal and the informal descriptions (i.e. the comments and the formal descriptions match)
- **Completeness:** All the knowledge that is expected to be in the ontology is either explicitly stated or can be inferred from the ontology.
- **Conciseness:** if the ontology is free of any unnecessary, useless, or redundant axioms.
- **Expandability:** refers to the required effort to add new definitions without altering the already stated semantics.
- **Sensitiveness:** relates to how small changes in an axiom alter the semantics of the ontology.

Thomas Gruber defines the following criteria [23]:

- **Clarity:** An ontology should effectively communicate the intended meaning of defined terms. Definitions should be objective. When a definition can be stated in logical axioms, it should be. Where possible, a definition is preferred over a description. All entities should be documented with natural language
- **Coherence:** Inferred statements should be correct. At the least, the defining axioms should be logically consistent. Also, the natural language documentation should be coherent with the formal statements.
- **Extendibility:** An ontology should offer a conceptual foundation for a range of anticipated tasks, and the representation should be crafted so that one can extend and specialize the ontology monotonically. New terms can be introduced without the need to revise existing axioms.
- **Minimal encoding bias:** An encoding bias results when a representation choices are made purely for the convenience of notation or implementation. Encoding bias should be minimized, because knowledge-sharing agents may be implemented with different libraries and representation styles.
- **Minimal ontological commitment:** The ontology should specify the weakest theory (i.e. allowing the most models) and defining only those terms that are essential to the communication of knowledge consistent with that theory.

Grüninger and Fox define a single criteria, **competency** (or, in extension, **completeness** if all the required competencies are fulfilled). In order to measure competency they introduce informal and formal **competency questions** [24].

Obrst *et al.* name the following criteria [43]:

- **coverage** of a particular domain, and the richness, complexity, and granularity of that coverage
- **intelligibility** to human users and curators
- **validity and soundness**
- evaluation against the specific use cases, scenarios, requirements, applications, and data sources the ontology was developed to address
- **consistency**
- **completeness**
- the sort of **inferences** for which they can be used
- **adaptability** and *reusability* for wider purposes
- **mappability** to upper level or other ontologies

Gangemi *et al.* define the following criteria [19]:

- **Cognitive ergonomics**: this principle prospects an ontology that can be easily understood, manipulated, and exploited.
- **Transparency** (explicitness of organizing principles): this principle prospects an ontology that can be analyzed in detail, with a rich formalization of conceptual choices and motivations.
- **Computational integrity and efficiency**: this principle prospects an ontology that can be successfully/easily processed by a reasoner (inference engine, classifier, etc.).
- **Meta-level integrity**: this principle prospects an ontology that respects certain ordering criteria that are assumed as quality indicators.
- **Flexibility** (context-boundedness): this principle prospects an ontology that can be easily adapted to multiple views.
- **Compliance to expertise**: this principle prospects an ontology that is compliant to one or more users.
- **Compliance to procedures for extension, integration, adaptation, etc.**: this principle prospects an ontology that can be easily understood and manipulated for reuse and adaptation.
- **Generic accessibility** (computational as well as commercial): this principle prospects an ontology that can be easily accessed for effective application.
- **Organizational fitness**: this principle prospects an ontology that can be easily deployed within an organization, and that has a good coverage for that context.

We have taken the freedom to summarize the mentioned criteria, and create a concise set. Eight criteria result from this literature survey: **accuracy**, **adaptability**, **clarity**, **completeness**, **computational efficiency**, **conciseness**, **consistency**, and **organizational fitness**. In the following, we define the criteria.

We have ignored evaluation criteria that deal with the underlying language used for describing the ontology instead of evaluating the ontology itself. Before OWL became widespread, a plethora of knowledge representation languages were actively used. For some ontologies, specific ontology languages were developed in order to specify that one ontology. Today, OWL is used for the vast majority of ontologies. Therefore we disregard criteria that are based on the ontology language, like expressivity, decidability, complexity, etc.

One example is the criteria *expandability* from [20]. It is defined as the required effort to add new definitions without altering the already stated semantics. Since OWL is a non-monotonic languages it is not possible to retract any inferences that have already been made. Thus the feature of non-monotonicity of OWL guarantees a certain kind of expandability for all ontologies in OWL.

2.1 Accuracy

Accuracy is a criteria that states if the axioms of the ontology comply to the expertise of the users. A higher accuracy comes from correct definitions and descriptions of classes, properties, and individuals. Correctness in this case means compliance to defined “gold standards”, be it other data sources, conceptualizations, or even reality ([12] introduces an approach to use reality as a benchmark, i.e. if the terms of the ontology capture the intended portions of reality). The axioms should constrain the possible interpretations of an ontology so that the resulting models are compatible with the conceptualizations of the users.

For example, all inferences of an ontology should be true. When stating that the `foaf:knows` property is a superproperty of a `married` property, then this axiom would only be accurate if indeed all married couples know their respective spouses. If we find counterexamples (for example, arranged pre-natal marriages), then the ontology is inaccurate.

2.2 Adaptability

Adaptability measures how far the ontology anticipates its uses. An ontology should offer the conceptual foundation for a range of anticipated tasks (ideally, on the web, it should also offer the foundation for tasks not anticipated before). It should be possible to extend and specialize the ontology monotonically, i.e. without the need to remove axioms (note that in OWL, semantic monotonicity is given by syntactic monotonicity, i.e. in order to retract inferences explicit stated axioms need to be retracted). An ontology should react predictably and intuitively to small changes in the axioms. It should allow for methodologies for extension, integration, and adaptation, i.e. include required meta-data. New tools and unexpected situations should be able to use the ontology.

For example, many terms of the FOAF ontology [11] are often used to describe contact details of persons. FOAF was originally designed to describe social networks, but its vocabulary also allows to formalize address books of all kinds.

2.3 Clarity

Clarity measures how effectively the ontology communicates the intended meaning of the defined terms. Definitions should be objective and independent of the context. Names of elements should be understandable and unambiguous. An ontology should use definitions instead of descriptions for classes. Entities should be documented sufficiently and be fully labeled in all necessary languages. Complex axioms should be documented.

For example, an ontology may choose to use URIs like `ex:a734` or `ex:735` to identify their elements (and may even omit the labels). In this case, users of the ontology needs to regard the whole context of the elements in order to find a suitable mapping to their own conceptualizations. Instead, the URIs could already include hints to what they mean, like in `ex:Jaguar` or `Lion`.

2.4 Completeness

Completeness measures if the domain of interest is appropriately covered. All questions the ontology should be able to answer can be answered. We define a number of completeness measures that measure different aspects of completeness: completeness with regards to the language (is everything stated that could be stated using the given language?), completeness with regards to the domain (are all individuals present, are all relevant concepts captured?), etc. Completeness also covers the granularity and richness of the ontology.

For example, an ontology to describe the nationalities of all members of the groups should provide the list of all relevant countries. Especially such closed sets (like countries, states in countries, members of a group) can often be provided as external resources to link to on the Semantic Web, and thus promise completeness.

2.5 Computational efficiency

Computational efficiency measures the ability of the used tools to work with the ontology, i.e. especially the speed that reasoners need to fulfill the required tasks, be it query answering, classification, or consistency checking. Some types of axioms may cause problems for certain reasoners. The size of the ontology also affects the efficiency of the ontology.

For example, using certain types of axioms will increase the reasoning complexity. But more important than theoretical complexity is the actual efficiency of the implementation used in a certain context. It is known that number restriction may severely hamper the efficiency of the KAON2 reasoner [39], and should thus be avoided when that system is used.

2.6 Conciseness

Conciseness is the criteria that states if the ontology includes irrelevant elements with regards to the domain to be covered (i.e. a book ontology including axioms about African lions) or redundant representations of the semantics. An ontology should impose a minimal ontological commitment, i.e. specify the weakest theory possible. Only essential terms should be defined. The ontology's underlying assumptions about the wider domain (especially about reality) should be as weak as possible, in order to allow the reuse within and communication between stakeholders that commit to different theories.

For example, an ontology about human resource department organization may take a naïve view on what a *human* actually is. It is not required to state if a human has a soul or not, if humans are the result of evolution or created directly by God, when human life starts or ends. The ontology would remain silent on all these issues, and thus allows both creationists and evolutionists to use it in order to make statements about which department has hired whom, and later exchange that data.

2.7 Consistency

Consistency describes that the ontology does not include or allow for any contradictions. Whereas accuracy states the compliance of the ontology with an external source, consistency states that the ontology itself can be interpreted at all. Logical consistency is just one part of it, but also the formal and informal descriptions in the ontology should be consistent, i.e. the documentation and comments should be aligned with the axioms. Representation choices should not be made for the convenience of the notation or implementation, i.e. the encoding bias should be minimized. Further ordering principles can be defined that the ontology has to be consistent with, like the OntoClean constraints on the taxonomy [26].

Note that within this deliverable we will deal with logical consistency and coherence only superficially. There is an active research community in the area of ontology debugging, that covers discovering, explaining, and repairing errors that lead to consistency and coherence, see for example [45, 36, 27].

An example for a non-logical inconsistency is the description of the element `ex:Jaguar` being "*The Jaguar is a feral cat living in the jungle.*", but having a logical axiom `ClassAssertion(ex:Operating_system ex:Jaguar)`. Such discrepancies are often the result of distributed ontology engineering or a badly implemented change management procedures in ontology maintenance.

2.8 Organizational fitness

Organizational fitness aggregates several criteria that decide how easily an ontology can be deployed within an organization. Tools, libraries, data sources, and other ontologies that are used constrain the ontology, and the ontology should fulfill these constraints. Ontologies are often specified using an ontology engineering methodology or by using specific data sets. The ontology metadata could describe the applied methodologies, tools, and data sources, and the organization. Such metadata can be used by the organization to decide if an ontology should be applied or not.

For example, an organization may decide that all ontologies used have to align to the DOLCE upper level ontology [18]. This will help the organization to align the ontologies and thus reduce costs when integrating data from different sources.

Chapter 3

Vocabulary

Evaluating the vocabulary aspect of an ontology means to evaluate the names used in the ontology. In this chapter we discuss methods for evaluating the vocabulary of an ontology, and how they map to the criteria given in Chapter 2.

The **vocabulary** of an ontology is the set of all *names* used in it. Names can be either URIs or literals. The set of all URIs of an ontology is called the *signature* of the ontology (and is thus the subset of the vocabulary without the literals). URIs are discussed in Section 3.1. *Literals* are names that are mapped to a concrete data value, i.e. instead of using a URI to identify an external entity, literals can be directly interpreted. Literals are presented in Section 3.2. Finally, we will also discuss *blank nodes*, i.e. unnamed entities within ontologies (Section 3.3).

3.1 URI references

Most names in ontologies are URI references (Uniform Resource Identifier, [7]). URIs are more generic forms of URLs (Uniform Resource Locator, [8]). Unlike URLs, URI references are not limited to identifying entities that have network locations, or use other access mechanisms available to the computer. They can be used to identify anything, from a person over an abstract idea to a simple information resource on the web [34].

An URI reference should identify one specific resource, i.e. the same URI reference should not be used to identify several distinct resources. A URI reference may be used to identify a collection of resources, and this is not a contradiction to the previous sentence: in this case the identified resource is the *collection* of resources, and thus a resource of its own. Classes and properties in OWL ontologies are also resources, and thus are identified by a URI reference.

A particular type of resources are *information resources*. Information resources are resources that consist of information, i.e. the digital representation of the resource captures the resource completely. This means that an information resource can be copied without loss, and it can be downloaded via the Internet. Therefore information resources can be located and retrieved with the use of a URL. An example of an information resource is the text of Shakespeare's "*Romeo & Juliet*" (from which this chapter's introductory quote is taken) which may be referenced, resolved, and downloaded via its URL `http://www.gutenberg.org/dirs/etext97/1ws1610.txt`

Non-information resources can not be downloaded via the Internet. There may be metadata about non-information resources available, describing the resource. An example is the book "*Romeo & Juliet*": the book itself can not be downloaded via the Internet (in contrast to its content). There may be metadata stating e.g. the weight or the prize of the book. In order to state such metadata we need to be able to identify the book, e.g. using its ISBN number [30]. In this case we can not use an URL: since the resource is not an information resource, it can not be located on the web, and thus can not be accessed with an URL. Nevertheless it may (and should) have an URI in order to identify the resource.

Non-information resources and information resources are disjoint classes (i.e. no information resource can at the same time be a non-information resource and vice versa). A further formalization of information resources can be found e.g. in the "*Functional Requirements for Bibliographic Records*" ontology (FRBR [49], widely used in the bibliographic domain) or in the DOLCE-based "*Ontology of Information Objects*" [25].

Table 3.1: An overview of what different response codes imply for the resolved HTTP URI reference U . I is the information resource that is returned, if any. L is the URI given in the location field of the response. The table covers the most important responses only, the others do not imply any further facts.

Response code	U has a fragment identifier	U has no fragment identifier
200 OK	I should describe U	U is the name of I . I is an information resource.
301 Moved Permanently	L should describe U	L and U are names of I . I is an information resource.
303 See Other	L should describe U	L should describe U
<i>Any other</i>	Nothing implied for I	Nothing implied for I

Linked data

URI references are strings that start with a protocol. If the protocol is known and implemented by the ontology based application, then the application may *resolve* the URI, i.e. use the URI according to the protocol in order to find more information about the identified resource. In case the URI is an URL, the identified information resource can be accessed and downloaded by using the protocol.

URI references consist of an URI with an optional fragment identifier. Most URI references in web ontologies are indeed using a protocol that can be resolved by the machine in order to fetch further information about the given URI reference. Most commonly this is achieved by using the HyperText Transport Protocol (HTTP, [17]). We have examined the Watson corpus (see Section 3.4) to figure out the usage of protocols on the Web. The Watson corpus contains 108,085,656 URIs. Only 491,710 (0.45%) of them are URIs not using the HTTP protocol.

Other prominent protocols besides HTTP are *file* (37,023 times; for local files), *mailto* (22,971 times; for email addresses), *mid* (13,448 time; for emails), *irc* (3,260 times; for internet relay chat channels and user ids), *ftp* (1,716 times, for the file transfer protocol), *tel* (703 times, for telephone numbers), or *https* (186 times; for secure HTTP connections). Sometimes these protocols are just mistyped (like *hhpt*).

Sometimes, QNames (qualified names, as used in XML, e.g. `foaf:knows`) can mistakenly be interpreted as an URI (especially in XML serialization, see [34]), and the namespace prefix will then be interpreted as the protocol scheme. We discovered that this makes up a good deal of the Non-HTTP protocols: the two most prominent non-HTTP schemes were *UnitOfAssessment* (46,691 times, or 0.043%) and *Institute* (42,331 times, or 0.039%). Both of them are meant to represent namespace prefixes in their ontologies, not URI schemes. These are errors in the ontology that can be easily discovered using the described method. Further examples for mis-interpreted namespace prefixes include *xs* (19,927 times; used for XML schema datatypes), *rdf* (272 times), *rdfs* (7 times), and *owl* (84 times).

Check used protocols

All URIs in the ontology are checked to be well-formed URIs. The evaluator has to choose a set of allowed protocols for the evaluation task. The usage of any protocol other than HTTP should be explained. All URIs in the ontologies have to use one of the allowed protocols.

Resolving an HTTP URI reference will return an HTTP response code and usually some content related to the URI reference. Certain HTTP responses imply facts about the used URI reference. These facts are given in Table 3.1, e.g. a 303 response on an URI without a fragment identifier implies the equality of the requested URI and the URI returned in the location field of the response. If the response code is a 200, it has even stronger implications: then the URI is actually the name of the served resource, i.e. the URI is a information resource that can (and is) accessible over the web [48]. Section 3.1 gives more details on fragment identifiers and how they can be used for evaluation.

Both OWL classes and OWL properties are not information resources. With Table 3.1 this allows us to infer that in OWL DL both class and property names (without fragment identifiers) should not return a 200 or a 301 when resolved via the HTTP protocol. In OWL 2 this is not true anymore, since punning allows URIs to be individuals, properties, and classes at the same time [40]. But as we will discuss later, punning should be avoided (see Section 3.1).

The table also lists the cases when the served resource should describe the name. We can easily check if this is the case, if the description is sufficiently useful, and if it is consistent with the knowledge we already

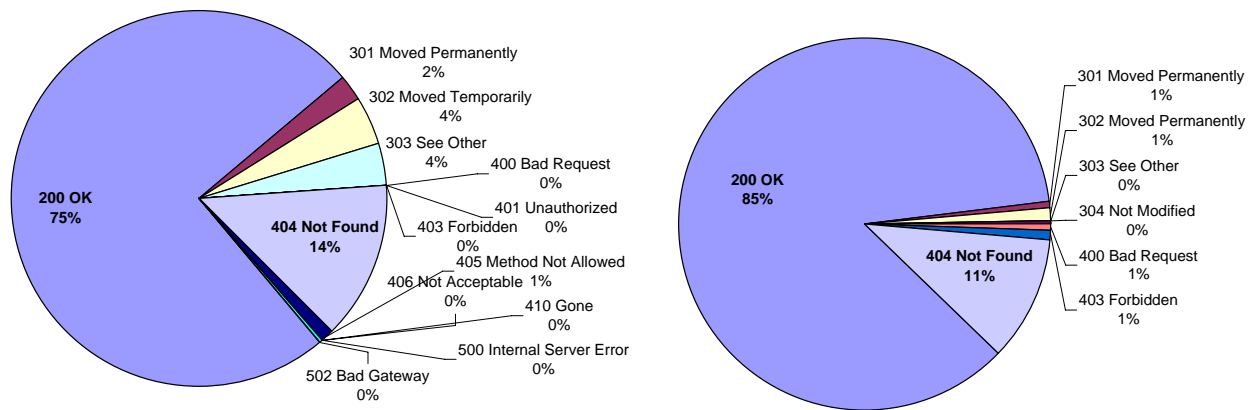


Figure 3.1: Distribution of the HTTP response codes on the HTTP URIs from the Watson EA corpus. The left hand side shows the slash URIs, the right hand side hash URIs.

have about the resource.

We have tested all HTTP URIs from the Watson EA corpus (see Section 3.4). For the slash namespaces URIs, we got 14,856 responses. Figure 3.1 shows the distribution of the response codes. It can be easily seen that for the vast majority (about 75%) of URIs we get 200 OK, which means that the current Semantic Web is indeed a web of metadata describing the connections between resources on the Web. 85% of the tested URIs return a 200 or 3XX response. We got 509 responses on the URIs with hash namespaces, of which significantly more (about 85%) responded with a 200 OK. This shows that in general hash URIs are better suited for terminological entities, and there should be good reasons for using a slash namespace.

Significance of the difference between hash and slash URIs

For significance testing we apply the two-proportion z-test. Let the null hypothesis be $P_s = P_h$, i.e. the probability for a slash URI to return a 200 OK being the same as the probability for a hash URI. According to our tests on the Watson EA corpus, we set $p_s = n_s/t_s \approx 0.7513$ and $p_h = n_h/t_h \approx 0.8585$ (with sample sizes $n_s = 14,856$ resp. $n_h = 509$ and $t_s = 11,161$ resp. $t_h = 437$ positive samples, i.e. URIs returning 200 OK codes). We calculate the pooled sample proportion $\hat{p} = \frac{p_s n_s + p_h n_h}{n_s + n_h} \approx 0.7548$. The standard error results in $e = \sqrt{\hat{p}(1-\hat{p})(1/n_s + 1/n_h)} \approx 0.0194$. The test statistic is a z-score defined as $z = \frac{p_s - p_h}{e} \approx -5.5316$. From that it follows that the probability that the null hypothesis is true is $p < 0.0001$, which means the result is statistically highly significant.

Names from the same slash namespace should always return the same response code. Differing response codes indicate some problems with the used terms. For example, the social website *LiveJournal*¹ exports FOAF profiles about their members, including their social contacts, interests, etc. Some of the terms return a 303 See Other (e.g. foaf:nick, foaf:knows, foaf:Person), whereas others return a 404 Not Found (e.g. foaf:tagLine, foaf:member_name, foaf:image). Investigating this difference uncovers that the first set are all terms that are indeed defined in the FOAF ontology, whereas the second set of terms does not belong to the FOAF ontology.

Check response codes

For all HTTP URIs, make a HEAD call (or GET call) on them. The response code should be 200 OK or 303 See Other. Names with the same slash namespace should return the same response codes, otherwise this indicates an error.

Note that this method can only be applied after the ontology has been published and is thus available on the Web.

In summary, the usage of resolvable URI references allows us to use the Web to look up more information on a given URI reference. This can help us to discover available mappings, or to explore and add new information to a knowledge based system. This is the major advantage of using the *Semantic Web* instead of simply an ontology based application.

¹<http://www.livejournal.com>

Table 3.2: The five hash and slash namespaces with the biggest number of names.

Hash namespace	# names
http://purl.org/obo/owl/FMA#	75,140
http://www.loa-cnr.it/ontologies/OWN/OWN.owl#	65,975
http://www.hero.ac.uk/rae/#	64,799
http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#	58,077
http://www.geneontology.org/owl/#	29,370
Slash namespace	# names
http://www.livejournal.com/	454,376
http://www.ecademy.com/	104,987
http://www.deadjournal.com/	79,093
http://www.aktors.org/scripts/	41,312
http://www.hi5.com/profile/	32,652

Hash vs slash

There was a long running debate in the Semantic Web community on the usage of fragment identifiers in URI references. The basic question is on the difference between using `http://example.org/ontology#joe` and `http://example.org/ontology/joe` in order to refer to a non-information resource. The former type of URI is called a hash URI (since the local part is separated by the hash character # from the namespace), the latter type a slash URI (since the local part is separated by the slash character / from the namespace). The discussion was resolved by the W3C Technical Advisory Group [37, 34, 10].

When resolving a hash URI, only the namespace is resolved. All hash URIs with the same namespace thus resolve to the same resource. This has the advantage that the ontology can be downloaded in one pass, but it also has the disadvantage that the file can become very big. Therefore, terminological ontologies and ontologies with a closed, rarely changing, and rather small set of individuals (like a list of all countries) would use hash URIs, whereas open domains with often changing individuals often use slash URIs (see for example in Semantic MediaWiki).

We analyzed the Watson corpus to see if there is a prevalence towards one or the other on the Web. We found 107,533,230 HTTP URIs that parse. 50,366,325 were hash URIs, 57,166,905 were slash URIs. Discounting repetitions, there were 5,815,504 different URIs in all, 2,247,706 of them hash URIs, 3,567,789 slash URIs.

Regarding their distribution over namespaces, there are much bigger differences: we find that there are only 46,304 hash namespaces compared to 2,320,855 slash namespaces. The hash namespaces are, in average, much bigger than the slash namespaces. 2,197,267 slash namespaces (94.67%) contain only a single name, whereas only 16,990 hash namespaces (36.69%) contain only one name. On the other extreme, only 253 slash namespaces (0.01%) contain more than a 100 names, in contrast to 2,361 hash namespaces (5.10%) that have more than 100 names. Still, as we can see in Table 3.2, the namespaces with the biggest names are slash namespaces, being several times as big as the biggest hash namespace. What does that mean?

Slash namespaces are used in two very different ways in ontologies. First, they are used to identify resources on the web and to provide metadata about them. This explains the huge number of slash namespaces with only a few names: only 3,142 slash namespaces (0.14%) have more than 10 names (compared to 4,575 hash namespaces, or 9.88%). In this case, when providing metadata, the names are often distributed all over the web, and thus introduce many different namespaces. The other usage of slash namespaces is for collectively built ontologies: since hash namespaces reside in one single file, they can not deal well with very dynamic vocabularies, that add and remove names all the time and change their definitions. Looking at the five biggest slash namespaces in Table 3.2, we see that four of five namespaces belong to social networks (*aktors.org* is the Website of a UK-based Semantic Web research project). Looking at the hash namespaces, we also see huge ontologies, but they represent much more stable domains, providing vocabularies for the life sciences (FMA, Gene ontology, NCI thesaurus), a vocabulary for education assessment (RAE), and an OWL translation of WordNet (OWN). None of them are interactively built on the Web by an open community, but rather curated centrally by editors.

Look up names For every name that has a hash namespace make a GET call against the namespace. For every name that has a slash namespace make a GET call against the name. The content type should be set

correctly. Resolve redirects, if any. If the returned resource is an ontology, check if the ontology describes the name. If so, *N* is a *linked data conformant name*. If not, the name may be wrong.

Opaqueness of URIs

URIs should be named in an intuitive way. Even though the URI standard [7] states that URIs should be treated opaque and no formal meaning should be interpreted into them besides their usage with their appropriate protocols, it is obvious that a URI like `http://aifb.de/person/Rudi_Studer` will invoke a certain denotation in the human reader: the user will assume that this is the URI for the person Rudi Studer working at the AIFB, and it would be quite surprising if it were the URI for the movie Casablanca.

On the other hand, an URI like `http://aifb.de/uri/p67` does not have an intuitive denotation, and so they become hard to debug, write manually, and remember. Their advantage is that they are not dependent on a specific natural language. An unfortunate number of tools still displays the URI when providing a user interface to the ontology. Therefore intuitive URIs, that actually identify the entities to which their names allude to (for the human reader) and that have readable URIs, should be strongly preferred. Also, since URIs unlike labels should not change often [48], it is important to catch typos when writing URIs.

URIs should follow a naming convention. When using natural language based names, the local name of classes may use plural (*Cities*) or singular forms (*City*), the local name of properties may use verbs (*marries*), verbal phrases (*married_to*), nouns (*spouse*), or nominal phrases (*spouse_of* or *has_spouse*). All of these naming conventions have certain advantages and disadvantages: phrases make the direction of the property explicit and thus reduce possible sources of confusion (given the triple *Aristotle Teacher Plato*, is it immediately clear who the teacher is, and who the student?). But using simple nouns can help with form based interfaces like Tabulator [6], a Semantic Web browser. Tabulator also constructs a name for the inverse property by appending “*of*” to the word, e.g. the inverse of *Teacher* would be *Teacher of*.

Capitalization and the writing of multi-word names should also be consistent. The ontology authors should decide which names to capitalize (often, classes and individuals are capitalized, whereas properties are not). Multi-word names (i.e. names that consist of several words, like *Wide square*) need to escape the whitespace between the words (since whitespaces are not allowed in URIs). Often this is done by camel casing (i.e. the space is removed and every word after a removed space starts with a capital letter, like *WideSquare*), by replacing the space with a special character (often an underscore like in *Wide_square*, but also dashes or fullstops), or by simple concatenation (*Widesquare*).

Many of these choices are just conventions. The used naming conventions should be noted down explicitly. Metadata about the ontology should state the naming convention for a given vocabulary. Many of the above conventions can then be tested automatically. It is more important to consistently apply the chosen convention than to choose the best convention (especially, since the latter is often unclear).

In general, URIs on the Semantic Web should follow also the same rules that URIs on the hypertext Web should follow. These are [5]:

- don't change (i.e. don't change what the resource refers to, nor change the URI of a resource without redirecting the old URI)
- be human guessable (i.e. prefer `http://example.org/movie/The_Matrix` over `http://example.org/movie/tt0133093`)
- be reasonably short (which also means not to use deep hierarchical nesting, but a rather flat structure)
- don't show query parameters (i.e. don't use URIs like `http://example.org/interest?q=Pop+Music`, use `http://example.org/interest/pop_music` instead)
- don't expose technology (e.g. don't use file extensions like `http://example.org/foaf.rdf`, use `http://example.org/foaf` instead)
- don't include metadata (e.g. don't add the author or access restrictions into the URI, like in `http://example.org/style/timbl/private/uri`, just use `http://example.org/style/uri`. Authorship and access level may change)

Check naming conventions A proper naming can be checked by comparing the local part of the URI with the label given to the entity or by using lexical resources like Wordnet [16]. Formalize naming conventions (like multi-word names and capitalization) and test if the convention is applied throughout all names of a namespace. Check if the URI fulfills the general guidelines for good URIs, i.e. check length, inclusion of query parameters, file extensions, depth of directory hierarchy, etc.)

Note that only local names from the same *namespace*, not all local names in the *ontology*, need to consistently use the same naming convention.

URI reuse

In order to enable easier sharing, exchange, and aggregation of information on the Semantic Web, the reuse of commonly used URIs proves to be helpful (instead of introducing new names). At the time of writing most domains do not provide an exhaustive lexicon of URIs yet, but it is expected that projects like *Freebase*², *Swoogle* [14], *Sindice* [44], or *Semantic Wikipedia* [35] will change that soon. Some domains, like life sciences, music, computer science literature, or geography already have very exhaustive knowledge bases of their domains. These knowledge bases can easily be reused.

Analyzing the Watson EA corpus, we find that 75% of the ontologies use 10 or more namespaces, in 95.2% of the ontologies the average number of names per namespaces is lower than 10, in 76.5% it is lower than 3, in 46.4% lower than 2. This means that most ontologies use many namespaces, but only few names from each namespace. Considering knowledge bases, this makes perfect sense: in their FOAF files persons may add information about their location by referencing the Geonames ontology, and about their favourite musician referencing the MusicBrainz ontology. Terminological ontologies often reference the parts of external ontologies relevant to them in order to align and map to their names.

Metrics of ontology reuse

We define the following measures and metrics:

- Number of namespaces used in the ontology N_{NS}
- Number of unique URIs used in the ontology N_{UN}
- Number of URI name references used in the ontology N_N (i.e. every mention of a URI counts)
- Ratio of name references to unique names $R_{NU} = \frac{N_N}{N_{UN}}$
- Ratio of unique URIs to namespaces $R_{UNS} = \frac{N_{UN}}{N_{NS}}$

Check the following constraints. The percentages show the proportion of ontologies that fulfill this constraint within the Watson EA corpus, thus showing the probability that ontologies not fulfilling the constraint are outliers.

- $R_{NU} < 0.5$ (79.6%)
- $R_{UNS} < 5$ (90.3%)
- $N_{NS} \geq 10$ (75.0%)

URI declarations and punning

Web ontologies do not require names to be declared. This leads to the problem that it is impossible for a reasoner to discern if e.g. `ex:Address` is a new entity or merely a typo of `ex:Address`. This can be circumvented by requiring to *declare* names, so that tools can check if all used names are properly declared. This further brings the additional benefit of a more efficient parsing of ontologies [41].

The declarations are axioms, stating not only that a name exists but also its type, i.e. if it is declared as a class, an individual, a datatype, object or annotation property. This feature was introduced in OWL 2, and thus does not yet appear in ontologies outside of test cases.

Check name declarations

²<http://www.freebase.com>

Check for every URI if there exists a declaration of the URI. If so, check if the declared type is consistent with the usage. This way it is possible to detect erroneously introduced punning.

In OWL 1 DL the set of class names, individual names, and property names were disjoint. This restriction was removed in OWL 2, and not it allows to use the names for either the individual, the property or the class. Based on the position in the axiom it is always clear which type of entity the name refers to. There are good reasons to allow punning: for example, the entity *lion*, depending on the context, may represent the individual *lion* that is of the type *species*, or it may be the type of the lion *Simba* [9]. There is no necessity to introduce different names for the two, or to render the merger of two ontologies inconsistent where *lion* is used as a class in the one ontology, and as an individual in the other. Nevertheless, punning may cause confusion with the user, especially since most tools are not yet well equipped to deal with punning. Punning should only be carefully applied.

3.2 Literals

Ontologies often contain **literals** that represent **data values**. These data values can be very varied, e.g. numbers (like *4.75*), points in time (like *2:14 pm CEST on 6th March 2009*), or strings (e.g. *Jaguar*). The importance of literals on the Semantic Web can be shown by their sheer number of occurrences: the Watson corpus consists of 59,749,786 triples, and 26,750,027 of them (42.8%) include a literal.

Anything that can be represented by a literal could also be represented by an URI. For example, we could introduce the URI *ex:Number4dot75* to be the URI to represent the number *4.75*. Using OWL Full, we could state that the literal and the URI are the same individual. Often it is more convenient to use a literal instead of a URI, especially since their meaning is already agreed on. Literals have the disadvantage that in triples they are only allowed to be objects. This means that we cannot make statements about literals directly, e.g. say that *4* is an instance of the class *ex:EvenNumber*. This can be circumvented by using URIs as proxies for data values. It is currently discussed to drop this limitation and to allow literals to also be subjects in triples.

Literals can be typed (see Section 3.2) or plain. A plain literal may be tagged with a language tag (Section 3.2). The standard does not allow literals to be both typed and language tagged. Language tagged literal would often make little sense: the integer *4* is the same number regardless of the language. Since it makes sense for text, a specification for the new data type `rdf:text` is currently being created in order to allow for language tagged typed text literals [3].

The RDF standard states that plain literals denote themselves [28], i.e. the plain literal *Jaguar* denotes the ordered list of characters *Jaguar*. Most of the literals on the Web are plain literals – only 1,123,704 (4.2%) of them are typed.

Typed literals and datatypes

A typed literal is a pair of a lexical representation and a data type. The data type is given by an URI that defines the interpretation of the lexical representation. Most ontologies use data types defined by the XML Schema Definition [15]. The OWL standard requires all tools to support `xsd:string` and `xsd:integer` [4] and names 33 further data types that should be implemented. OWL 2 extends the number of required data types considerably, providing nineteen data types for numbers, eight data types for text, one for boolean values, two for binary data, one for URIs, one for time instants, and one for XML literals [42] (note that even though this also are 33 data types, they are not the same as the one OWL 1 recommends). Some of the data types are noted as features that may be removed from the final specification of OWL 2.

Figure 3.2 shows the most often used data types. The most often used data types all belong to the set of recommended data types by the specification. The only two non-recommended data types that are used more often than a hundred times are from a deprecated XML schema version³ and from the W3C's calendar working group.⁴

A fairly common error in data types is to use a namespace prefix (`xs:string` appears 19,927 times, `xsd:string` 518 times). Other common errors include misspelling of the data type URIs (e.g. forgetting the hash, or miscapitalizing the local name) or using deprecated versions of the XML schema.

³<http://www.w3.org/2000/10/XMLSchema\#nonNegativeInteger>, used 157 times

⁴<http://www.w3.org/2002/12/cal/icaltzd\#dateTime>, used 128 times

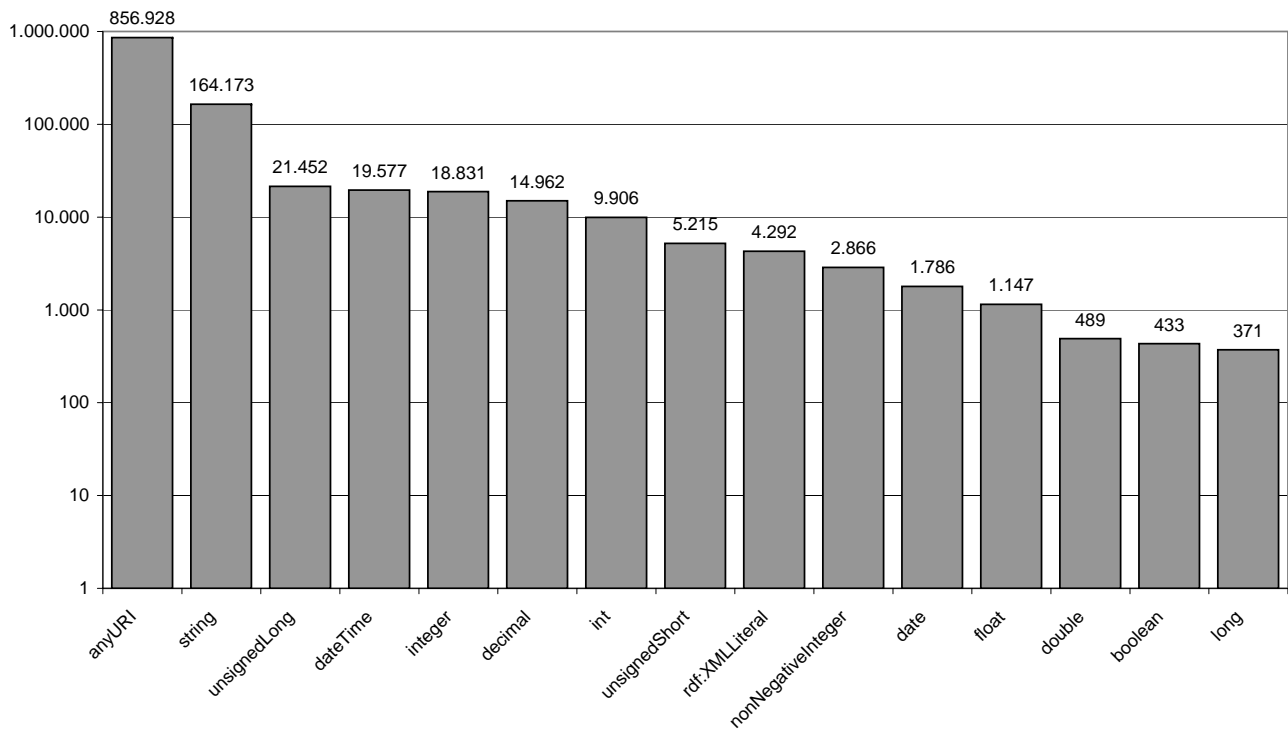


Figure 3.2: The fifteen most often used data types in the Watson corpus. Note the logarithmic scale.

The Semantic Web standards allow to define new custom data types, but this option is very rarely used. It makes it nevertheless hard to automatically discover if a data type URI is just an unknown data type, or if it is indeed a typo. Data type URIs should be resolvable just as all other URIs and thus allow a tool to make an automatic check. When evaluating ontologies, the evaluator should decide on a set of allowed data types. This set depends on the use case and the tools being used. All tools should be able to deal with the data types in this set. This closed set will help to discover syntactic errors in the data type declarations of the literals in the ontology.

An ontology should be checked if all used data types in the ontology are understood by the tools dealing with these data types. That does not mean that all tools need to understand all the used data types. A tool may be perfectly capable of dealing with an unknown data type as far as the task of the tool itself is concerned. For example, a tool used for converting RDF/XML-serialization into OWL Abstract Syntax does not need to understand the data types. A reasoner who needs to check equality of different values on the other hand needs to understand the used data types.

The second check that is relevant for typed literals is to check if the literals are syntactically correct for the given data type. A typed literal of the form "Four" and data type `http://www.w3.org/2001/XMLSchema#integer` is an error and needs to be corrected. For this it is important for the evaluation tool to be able to check the syntactic correctness of all used data types. This should be considered when choosing the set of allowed data types. Otherwise it will be hard to discover simple syntactic errors within literals.

Check literals and data types

A set of allowed data types should be created. All data types beyond besides those recommended by the OWL specifications should be avoided. Creating a custom data type should have a very strong reason. `xsd:integer` and `xsd:string` should be the preferred data types (since they have to be implemented by all OWL conformant tools).

Check if the ontology uses only data types from the set of allowed data types. All typed literals must be syntactically valid with regards to their data type. The evaluation tool needs to be able to check the syntactical correctness of all allowed data types.

Language tags

Language tags can be used on plain literals to state the natural language used by the literal. This enables tools to choose to display the most appropriate literals based on their user's language preferences. An example for a literal with a language tag is "university"@en or "Universität"@de. Language tags look rather simple, but are based on a surprisingly large set of specifications and standards.

Language tags are specified in the IETF RFC 4646 [47]. IETF RFC 4647 in turn specifies the matching of language tags [46]. Language tags are based on the ISO codes for languages, if possible taking the Alpha-2 code (i.e. two ASCII letters representing a language) as defined by [32], otherwise the Alpha-3 code (three ASCII letters representing a language), defined by [33]. Not all languages have an Alpha-2 code. The specification allows to use Alpha-3 codes only if they do not have an Alpha-2 code. For example, to state that the literal *Gift* is indeed the English word, we would tag it with en, the Alpha-2 code for the English language. If we wanted to state that it is a German word, we would have tagged it with de, the Alpha-2 code for German. If it were the Middle English word, it would have to be tagged with enm (since no two letter code exists).

The language tags can be further refined by a script, regional differences, and variants. All these refinements are optional. The script is specified using ISO codes for scripts [29]. To state that we use Russian with a latin script would be ru-latn. Every language has a default script, that should not be specified when used, e.g. en always assumes to be en-latn, i.e. English is by default written in latin script. The IANA registry maintains a complete list of all applicable refinement subtags, and also specifies the default scripts for the used languages.⁵ Following the optional script parameter, the language can be specified to accommodate regional differences. The codes for the regions are based on either the countries and regions ISO codes [31] or, alternatively, on the UN M.49 numeric three-digits codes [50]. English as spoken in Hong Kong would either be defined as en-hk or en-344. The regional modifiers should only be used when needed, and preferably omitted. For example, instead of using ja-jp for *Japanese as spoken in Japan* the tag ja would be preferred. Finally, a relatively small number of further variants can be specified defined for special cases like historic language deviations, sign languages, and similar.

It is also possible to define private language tags or tag refiners, denoted by an x. So one could use en-x-semweb as the language tag for the kind of English that is spoken by the Semantic Web research community, where certain terms have meanings deviating from standard English. Private tags should be avoided, and indeed, our analysis of the Watson corpus shows that none are used.

Language tags are case insensitive. So it does not matter if one uses en-us, EN-US, en-US, En-us, or any other combination of upper and lower case characters. On the Semantic Web it seems to be usual to use lower case characters only, with less than 200 occurrences of upper case codes.

We examined the usage of language tags on the Semantic Web to find if the standards are applied correctly. For such a complex mesh of standards we found surprisingly few errors. All in all, 17,313,981 literals have a language tag (67.6% of all plain literals). English (en) was by far the most often used tag, applied 16,767,502 times (96.8%), followed by Japanese (ja) with 519,191 tags (3.0%). The other languages are less widely used by far, Suomi (fi) is used 9,759 times, German (de) 4,893 times, French (fr) 1,810 times, and Spanish (es) 866 times.

The most often applied refined tags are en-us with 2,284 tags, en-gb 1,767 times, and fr-fr 288 times. The latter could be regarded as redundant, since fr would be sufficient. Further inappropriate usages or errors that can be found in the language tags are en-uk (it should be en-gb, used 90 times), frp (undefined tag, used 30 times), and jp (should be ja). In summary, with respect to the complexity of the used standards, the number of errors regarding language tags is extremely low. This probably stems from the fact that most of the advanced features are never used: no single tag specifies a script (besides one example literal) or a variant, and only a handful of tags specify a region, never using the UN M-49 standard (besides one example literal)⁶ but always ISO 3166 codes.

Check language tags

Check that all language tags are valid with regards to their specification. Check if the shortest possible language tag is used (i.e. remove redundant information like restating default scripts or default regions). Check if the stated language and script is actually the one used in the literal.

⁵<http://www.iana.org/assignments/language-subtag-registry>

⁶Both cited example literals are based on the W3C's write up on its internationalization effort, highlighting the possibilities of languages tags. See here: <http://www.w3.org/International/articles/language-tags/Overview.en.php>

Check if the literals are tagged consistently within the ontology. This can be checked by counting every occurrence n_l of each language tag l that occurs at all in the ontology. All n_l should be roughly the same. Outliers should be inspected.

Labels and comments

Labels are used in order to provide a human readable name for an ontological entity. Every ontological entity should have labels in all relevant languages. Almost none of the ontologies in the Watson corpus have a full set of labels in more than one language, i.e. most ontologies are not multi-lingual. Thus they miss a potential benefit of the Semantic Web, i.e. the language independence of ontologies. Comments add further human-readable explanations to a specific entity, and should also be language tagged.

Labels and comments should follow a style guide and be used consistently. A style guide should define if classes are labeled with plural or singular noun, if properties are labeled with nouns or verbs, and under what circumstances comments should be used. Labels and comments should never use camel case or similar escape mechanisms for multi word terms, but instead simple use space characters (or whatever is most suitable for the given language). I.e. an URI `http://example.org/LargeCity` should have a label "large city"@en. External dictionaries like WordNet [16] can be used to check consistency with regards to a style guide.

In an environment where ontologies are assembled on the fly from smaller ontologies [1], the assembled parts may follow different style guides. The assembled ontology will then not adhere to a single style guide and thus offer an inconsistent user interface. It is not expected that a single style guide will become ubiquitous on the whole web. Instead, an ontology may specify explicitly what style guide it follows, and even provide labels following different style guides. For example, the SKOS ontology [38] offers more specific subproperties for labels like `skos:prefLabel`. This would allow to introduce a subproperty of label that is style guide specific, which would in return allow for the consistent display of assembled ontologies.

Even when subproperties of `rdfs:label` are defined, there should always be one label (per supported language) given explicitly by using `rdfs:label` itself. Even though this is semantically redundant, many tools (especially visualization tools) do not apply reasoning for fetching the labels of an entity but simply look for the explicit triple stating the entity's label.

Check labels and comments

Define the set of relevant languages for an ontology. Check if all label and comment literals are language tagged. Check if all entities have a label in all languages defined as being relevant. Check if all entities that need a comment have one in all relevant languages. Check if the labels and comments follow the style guide defined for the ontology.

3.3 Blank nodes

Blank nodes are an RDF feature that allows to use a node in the RDF graph without giving it a URI. This way the node can only be indirectly referenced (if at all), for example by using an inverse functional property. Blank nodes relieve the author of an RDF graph to come up with good URIs for every node, which would impose additional costs on creating the graph.

There are two different scenarios for using blank nodes. First, blank nodes are used in the structural representation of certain OWL axioms within RDF graphs. Second, blank nodes are used for anonymous ontology entities. An example for the first scenario is the representation of a disjoint union axiom in RDF. The axiom `DisjointUnion(C D E)` will be represented by the following RDF triples:

```
C owl:disjointUnionOf _:x .
_:x rdfs:first D .
_:x rdfs:rest _:y .
_:y rdfs:first E .
_:y rdfs:rest rdfs:nil .
```

Blank nodes in RDF are represented by using the namespace prefix `_` (underscore). In the given example, there are two blank nodes, `_:x` and `_:y`. They do not represent any entities in the domain, but are introduced

only out of structural necessity since we cannot state the relationship between three entities (C, D, and E) from the original axiom directly with triples. We defined these kind of blank nodes to be *structural blank nodes*. Even though they could be given URIs, these URIs would not represent any concept in our conceptualization and thus should be avoided.

The second scenario uses blank nodes to represent anonymous ontology entities. For example, in the first view years it was regarded as good practice *not* to define a URI for persons in FOAF documents but to use blank nodes instead. The argument in favor of using blank nodes for persons was that it was regarded inappropriate to name people via URIs. This echoes the sentiment of "*I am not a number*", or rather, "*I am not a URI*". FOAF preferred to identify persons by using inverse functional properties like eMail-adresses or their hash sums.

Web architecture later suggested that all entities of interest should have a URI [34]. The FOAF project also deprecated the use of blank nodes for persons (since they are definitively entities of interest). Using a URI for an entity allows for all the advantages described earlier about linked data (see Section 3.1), most importantly the possibility to look up further data about the given entity by resolving the URI.

In summary, blank nodes should be avoided unless structurally necessary.

Check for superfluous blank nodes

The RDF syntax serialization of OWL2 [21] lists all cases of structurally necessary blank nodes in RDF graphs. Check for every blank node if it belongs to one of these cases. Besides those, no further blank nodes should appear in the RDF graph. All blank nodes not being structurally necessary should be listed as potential errors.

3.4 Selection of the corpus

In order to test our approach on a realistic corpus of web ontologies we have created and made available a corpus of ontologies based on the Watson collection. Watson is a search engine developed by the Knowledge Media Institute [13]. We took the copy of the Watson corpus that came as part of the Billion Triple Challenge.⁷ Parts of the experiments were done by selecting a random sample of ontologies from the corpus. The EA corpus is the corpus whose ontology name's hash values start with EA. This corpus contains 515 ontologies.

⁷<http://challenge.semanticweb.org>

Chapter 4

Content Evaluation in Semantic MediaWiki

Wikis have proved to be systems that enable communities to collaboratively create knowledge. Wikipedia, the best known wiki, has shown that wikis can grow to a truly global scale. Wikipedia does not work solely because of its underlying software MediaWiki, but it is a rather complex socio-technical entity that works due to often implicit community processes and rules [2].

Semantic wikis have shown to be feasible systems to enable communities to collaboratively create semantically rich content. Additionally to classic wikis they also allow the community to add more structured information to the textual and multimedia content. Such structured content is successfully used in order to reduce redundancy and thus increase consistency within the wiki. In this chapter we discuss how the structured data can additionally be used in order to further ensure the quality of the wiki content. Whereas the formal basis of these content checks is well understood, it is unclear how the social processes will play out.

In this chapter we present a number of means to cope with possible problems when using SMW introduced due to the heterogeneity of its users. These means aim at keeping the knowledge base consistent and compact by giving feedback to the user regarding e.g. redundancies.

Note that in this Chapter, we use the terms *category* and *class* interchangeable, since categories in SMW represent class in OWL DL.

4.1 Cardinality checks

Concept cardinality checks, i.e. explicit statements on how many results shall a query within the wiki have. Besides exact numbers also minimal and maximal cardinalities are allowed. Note that this allows to state disjointness (by stating that the intersection has a cardinality of 0).

It has to be noted that the above checks do not follow the usual OWL semantics. Concept cardinality in SMW does not relate to nominals and OWL property cardinalities, but follow rather the semantics of autoepistemic operators [22]. Within the wiki, we assume a closed world and unique names if not otherwise stated.

The deviation from OWL semantics has to be carefully considered when exporting data from the wiki. It is obvious that domain and range statements in the wiki should not be exported as OWL domain constructs, since that would lead to different semantics of the export and within the system.

An interesting sidenote is that the implementation of cardinality checks can be done completely without adding any proprietary code to the wiki but with merely using what is already available by using templates and count queries (for a description of both features, refer to Deliverable D1.3.1).

Using these, we can simply instantiate an appropriate template on the concept page like this:

```
{{Concept cardinality|20}}
```

which could display a big warning box whenever the result of a count query against itself (i.e. the concept of the page) is different from the provided value (e.g. 20 in the given example).

4.2 Discovering redundancies

One issue we want to deal with are similarities with names. Since there are different terms referring to the same entity different users might introduce the same entity but give different names to that entity. So here the point is to give the user a hint to resolve such an issue like we have it in the following example: one user might call

a vehicle with four wheels and an engine *auto*, another might call it *automobile*. Here the system will compare the names of two concepts by checking whether one name is part of the other. The user then has to decide whether there is really one concept denoted by different terms and therefore resolve that issue or whether the two terms denote different concepts.

Another way to introduce redundancies is by having too many subsumption relations. Imagine a user introducing the categories *dog*, *pet* and *animal*, with *dog* being a subcategory of *pet*, *dog* a subcategory of *animal*, and *pet* a subcategory of *animal*. The system could now discover that the explicit subcategorization of *dog* by *animal* is redundant. The system points this out to the user who then has to decide which relation is redundant or even wrong.

For the just mentioned functionality it is necessary to check whether there are cycles (regarding the subsumption) in the knowledge base. For the redundant relations issue this check is performed for a specific class and its related classes. However there is also the possibility to give an overview of all the classes which are part of cycles.

Regarding classes, individuals and properties there has been implemented another functionality which compares the propertysets (that is the set of properties of all individuals of a class) of classes, which have the same superclass. The idea there is that if a class has a similar propertyset (with a certain percentage e.g. 50%) as a sibling class then there might be the need for integrating the two classes among each other. Let us assume we have two classes *airliner* and *jet*. They both have the same superclass *aircraft*. Regardless which individuals belong to the two classes we assume that the propertysets of both classes contain the properties *speed* and *altitude*. This information would then be delivered to the user who then has to decide whether the classes refer to the same concept.

Like the function of displaying all the classes which have no superclass, which is already implemented in SMW, we have adapted the system in a way that all classes which have no subclass will be displayed to the user. There might be classes which need to be subclassed further since their granularity is not fine enough. The user can add additional subclasses by having a compact overview of all the leaves in the knowledge base.

4.3 Statistical approaches

Beside the now mentioned, rather specific functionalities there are also means for getting general information about the knowledge base. There we have the possibility to display histograms relating the names of classes in the knowledge base to the letters of the alphabet. If the distribution does not follow the natural distribution of words starting with a specific letter then this is a hint that there might be something wrong. This of course is also applicable with properties and individuals. There is several more information that can be displayed to the user like the number of subclasses, the number of superclasses, the number of individuals, the steps needed to reach the top class or the leave class and others.

In SMW the functionalities are used differently. On the one hand the system gives feedback to the user during the creation process. Whenever a user creates a new class for example the system will invoke some functionalities and provide feedback. This feedback can be structured by adding different levels of importance which means that there are issues which have to be resolved as well as issues that merely should be resolved.

On the other hand we use special pages which can be accessed by the regular user as well, however, there the focus is to provide knowledge experts with a comprehensive view on certain issues (other than the regular user who is interested in a small amount of classes the knowledge expert seeks to resolve issues regardless any preferences).

4.4 Social and usability aspects

Adding constraints in the wiki must be simple enough to allow contributors to actually apply these features. The given selection is based on the fact that they can be represented within the wiki simply and unambiguously, i.e. contributors will always know where to look for a specific piece of information. This is a necessary requirement in order to keep a wiki maintainable.

A major decision to make is whether to allow contributors to introduce inconsistencies or not. When a page is modified, the wiki could check if that edit would turn it inconsistent and then cancel the edit. Disregarding if a real time check would be computationally feasible, it seems to conflict with the wiki paradigm. Instead of

prohibiting edits that lead to inconsistencies we plan to introduce special pages that report discovered problems and allow to repair them efficiently.

Chapter 5

Conclusions and outlook

We have collected a succinct set of quality criteria for ontologies, that will guide the evaluation of ontologies within the project. We have provided a comprehensive study of the quality of the vocabulary aspect of ontologies on the Web, and have shown that reusing terms from Web ontologies is feasible. During the second year, all ontologies chosen by the case studies or chosen by the technical partners (e.g. in Deliverable 1.1.1) will be evaluated with the approaches developed by this Work package, like those presented in Chapter 3.

We presented approaches towards enabling user communities to automatically detect quality problems within the content of their semantic wikis. This enables computers and humans to collaborate in novel ways in order to achieve high quality content in wikis more efficiently. We have argued that it is feasible to enable end users to define and manage such quality checks themselves. Our next step is to gather feedback from the community, use the suggested approach, and to evaluate them in order to discover their effects on content quality and especially community development.

We expect that these features will free up valuable time and effort of contributors who hitherto had to check such information manually, who in turn could engage in other projects towards improving their wiki. This is especially true for enterprise wikis, since unlike open community wikis like Wikipedia they often have a very different work structure.

In the second year, we plan to gain experience and introduce further content checking mechanisms to the wiki. One further feature that we expect is to add more powerful consistency checks. We need to find a way to easily express such powerful checks and to evaluate if they get accepted. We expect that in all case studies that use Semantic MediaWiki they will be uses for the consistency checking features. Here we list a few examples:

- When using SMW for managing content snippets for answering a request for a bid, the wiki could check for snippets that are older than a specified time range, or that have no editor associated with them
- When using SMW for creating a complex bid, complex dependencies in the text – like *"if a bid is valued for more than 5 Million Euro, the bid manager should be at least a level 3 accountant"* – can be formulated and automatically checked
- the representation of knowledge structures within the company – e.g. the social graph – can be automatically checked for consistency, i.e. each person should only have one direct supervisor, each person should have at least one person to report to, etc.

What we also will aim at in the second year (what has been omitted during the first year) was the leverage of weaker knowledge structures. This especially involves approaches for conceptual tagging, i.e. lifting simple keyword taggings to become taggings pointing to a well defined concept, e.g. defined in a wiki or vocabularies from external knowledge sources (as those evaluated in Chapter 3).

Bibliography

- [1] Harith Alani. Position paper: ontology construction from online ontologies. In Les Carr, David De Roure, Arun Iyengar, Carole A. Goble, and Michael Dahlin, editors, *Proceedings of the 15th international conference on World Wide Web, WWW 2006, Edinburgh, Scotland, UK, May 23-26, 2006*, pages 491–495. ACM, 2006.
- [2] Phoebe Ayers, Charles Matthews, and Ben Yates. *How Wikipedia works*. No Starch Press, San Francisco, CA, October 2008.
- [3] Jie Bao, Axel Polleres, and Boris Motik. `rdftext`: A datatype for internationalized text, 2008. W3C Working Draft 2 December 2008, available at <http://www.w3.org/TR/2008/WD-rdf-text-20081202/>.
- [4] Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. OWL Web Ontology Language Abstract Reference, 2004. W3C Rec. 10 February 2004.
- [5] Tim Berners-Lee. Cool URIs don't change. W3C Style, 1998. available at <http://www.w3.org/Provider/Style/URI.html>.
- [6] Tim Berners-Lee, Yuhsin Chen, Lydia Chilton, Dan Connolly, Ruth Dhanaraj, James Hollenbach, Adam Lerer, and David Sheets. Tabulator: Exploring and analyzing linked data on the Semantic Web. In Lloyd Rutledge, m.c. schraefel, Abraham Bernstein, and Duane Degler, editors, *Proceedings of the Third International Semantic Web User Interaction Workshop SWUI2006 at the International Semantic Web Conference ISWC2006*, 2006.
- [7] Tim Berners-Lee, Roy Fielding, and Larry Masinter. Uniform Resource Identifier (URI): Generic Syntax. Technical Report 3986, Internet Engineering Task Force, June 2005. RFC 3986 (available at <http://www.ietf.org/rfc/rfc3986.txt>).
- [8] Tim Berners-Lee, Larry Masinter, and Mark McCahill. Universal Resource Locators (URL). Technical Report 1738, Internet Engineering Task Force, December 1994.
- [9] Diego Berrueta and Jon Phipps. Representing classes as property values on the semantic web, 2005. W3C Working Group Note 5 April 2005, avail. at <http://www.w3.org/TR/swbp-vocab-pub/>.
- [10] Diego Berrueta and Jon Phipps. Best practice recipes for publishing RDF vocabularies, 2008. W3C Working Group Note 28 August 2008, avail. at <http://www.w3.org/TR/swbp-vocab-pub/>.
- [11] Dan Brickley and Libby Miller. The Friend Of A Friend (FOAF) vocabulary specification, July 2005.
- [12] Werner Ceusters and Barry Smith. A realism-based approach to the evolution of biomedical ontologies. In *Proceedings of the AMIA 2006 Annual Symposium*, November 2006.
- [13] Mathieu d'Aquin, Claudio Baldassarre, Laurian Gridinoc, Marta Sabou, Sofia Angeletou, and Enrico Motta. Watson: Supporting next generation semantic web applications. In *WWW/Internet conference*, Vila real, Spain, 2007.

- [14] Li Ding, Tim Finin, Anupam Joshi, Rong Pan, R. Scott Cost, Yun Peng, Pavan Reddivari, Vishal Doshi, and Joel Sachs. Swoogle: a search and metadata engine for the semantic web. In *CIKM '04: Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 652–659, New York, NY, USA, 2004. ACM.
- [15] David C. Fallside and Priscilla Walmsley. XML schema part 0: Primer second edition, 2004. W3C Rec. 28 October 2004.
- [16] Christine Fellbaum. *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*. MIT Press, May 1998.
- [17] Roy Fielding, James Gettys, Jeffrey Mogul, Henrik Frystyk, Larry Masinter, Paul Leach, and Tim Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616, June 1999.
- [18] A. Gangemi, N. Guarino, C. Masolo, A. Oltramari, and L. Schneider. Sweetening ontologies with DOLCE. In A. Gómez-Pérez and V. R. Benjamins, editors, *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web (EKAW 2002)*, volume 2473 of *Lecture Notes in Artificial Intelligence (LNAI)*, pages 166–181, Siguenza, Spain, 2002. Springer.
- [19] Aldo Gangemi, Carola Catenacci, Massimiliano Ciaramita, and Jens Lehmann. Ontology evaluation and validation: an integrated formal model for the quality diagnostic task. Technical report, Laboratory of Applied Ontologies – CNR, Rome, Italy, 2005. http://www.loa-cnr.it/Files/OntoEval4OntoDev_Final.pdf.
- [20] Asunción Gómez-Pérez. Ontology evaluation. In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies in Information Systems, First Edition*, International Handbooks on Information Systems, chapter 13, pages 251–274. Springer, 2004.
- [21] Bernardo Cuenca Grau, Ian Horrocks, Boris Motik, Bijan Parsia, Peter Patel-Schneider, and Ulrike Sattler. OWL 2: The next step for OWL. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(4):309–322, 2008.
- [22] Stephan Grimm and Boris Motik. Closed world reasoning in the semantic web through epistemic operators. In Bernardo Cuenca Grau, Ian Horrocks, Bijan Parsia, and Peter Patel-Schneider, editors, *Second International Workshop on OWL: Experiences and Directions (OWLED 2006)*, Galway, Ireland, 2005.
- [23] Thomas R. Gruber. Towards principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies*, 43(5/6):907–928, 1995.
- [24] Michael Grüninger and Mark S. Fox. Methodology for the design and evaluation of ontologies. In *IJCAI95 Workshop on Basic Ontological Issues in Knowledge Sharing*, Montreal, 1995.
- [25] Nicola Guarino. Ontology of information objects. FOFIS Deliverable 2, Istituto di Scienze e Tecnologie della Cognizione del Consiglio Nazionale delle Ricerche, Mar 2006.
- [26] Nicola Guarino and Christopher Welty. Evaluating ontological decisions with OntoClean. *Communications of the ACM*, 45(2):61–65, February 2002.
- [27] Peter Haase and Guilin Qi. An analysis of approaches to resolving inconsistencies in DL-based ontologies. In *Proceedings of International Workshop on Ontology Dynamics (IWOD'07)*, pages 97–109, June 2007.
- [28] Patrick Hayes. RDF Semantics. W3C Recommendation 10 February 2004, 2004. available at <http://www.w3.org/TR/rdf-mt/>.
- [29] ISO 15924. Codes for the representation of names of scripts. Technical report, International Standard ISO, 2004.
- [30] ISO 2108. Information and documentation – International standard book number (ISBN). Technical report, International Standard ISO, 2005.

- [31] ISO 3166. Codes for the representation of names of countries and their subdivisions. Technical report, International Standard ISO, 1999.
- [32] ISO 639-1. Codes for the representation of names of languages – Part 1: Alpha-2 code. Technical report, International Standard ISO, 2002.
- [33] ISO 639-2. Codes for the representation of names of languages – Part 2: Alpha-3 code. Technical report, International Standard ISO, 1998.
- [34] Ian Jacobs and Norman Walsh. Architecture of the World Wide Web Vol. 1, 2004. W3C Recommendation 15 December 2004, avail. at <http://www.w3.org/TR/webarch/>.
- [35] Markus Krötzsch, Denny Vrandečić, Max Völkel, Heiko Haller, and Rudi Studer. Semantic wikipedia. *Journal of Web Semantics*, 5:251–261, September 2007.
- [36] Joey Lam. *Methods for resolving inconsistencies in ontologies*. PhD thesis, University of Aberdeen, 2007.
- [37] Rhys Lewis. Dereferencing HTTP URIs, 2007. Draft TAG Finding 31 August 2007, avail. at <http://www.w3.org/2001/tag/doc/httpRange-14/2007-08-31/HttpRange-14.html>.
- [38] Alistair Miles and Sean Bechhofer. SKOS Simple Knowledge Organization System Reference, 2008. W3C Working Draft 29 August 2008, available at <http://www.w3.org/TR/2008/WD-skos-reference-20080829/>.
- [39] Boris Motik. *Reasoning in Description Logics using Resolution and Deductive Databases*. PhD thesis, Universität Fridericiana zu Karlsruhe (TH), Germany, 2006.
- [40] Boris Motik. On the properties of metamodeling in OWL1. *Journal of Logic and Computation*, 17(4):617–637, 2007.
- [41] Boris Motik and Ian Horrocks. Problems with OWL syntax. In *OWLED2006 Second Workshop on OWL Experiences and Directions*, Athens, GA, USA, 2006.
- [42] Boris Motik, Peter F. Patel-Schneider, and Bijan Parsia. OWL2 Web Ontology Language: Structural specification and functional-style syntax, 2008. W3C Working Draft 2 December 2008, available at <http://www.w3.org/TR/2008/WD-owl2-syntax-20081202/>.
- [43] Leo Obrst, Werner Ceusters, Inderjeet Mani, Steve Ray, and Barry Smith. The evaluation of ontologies. In Christopher J.O. Baker and Kei-Hoi Cheung, editors, *Revolutionizing Knowledge Discovery in the Life Sciences*, chapter 7, pages 139–158. Springer, 2007.
- [44] Eyal Oren, Renaud Delbru, Michele Catasta, Richard Cyganiak, Holger Stenzhorn, and Giovannia Tummarello. Sindice.com: A document-oriented lookup index for open linked data. *International Journal of Metadata, Semantics and Ontologies*, 3(1), 2008.
- [45] Bijan Parsia, Evren Sirin, and Aditya Kalyanpur. Debugging OWL ontologies. In *Proceedings of the 14th World Wide Web Conference (WWW2005)*, Chiba, Japan, May 2005.
- [46] Addison Phillips and Mark Davis. Matching of Language Tags. Technical Report RFC 4647, Internet Engineering Task Force, September 2006. RFC 4647 (available at <http://www.ietf.org/rfc/rfc4647.txt>).
- [47] Addison Phillips and Mark Davis. Tags for Identifying Languages. Technical Report RFC 4646, Internet Engineering Task Force, September 2006. RFC 4646 (available at <http://www.ietf.org/rfc/rfc4646.txt>).
- [48] Leo Sauermann and Richard Cyganiak. Cool URIs for the semantic web. Interest group note, W3C, March 2008.

- [49] Barbara Tillett. What is FRBR? a conceptual model for the bibliographic universe. *The Australian Library Journal*, 54(1), February 2005.
- [50] UN M.49. Standard Country or Area Codes for Statistical Use, Revision 4. Technical Report Sales No. 98.XVII.9, United Nations Statistics Division UNSD, 1998.