Applied Ontology 3 (2008) 41–62
DOI 10.3233/AO-2008-0048
IOS Press

AEON – An approach to the automatic evaluation of ontologies

Johanna Völker[a,*], Denny Vrandečić[a], York Sure[a] and Andreas Hotho[b]
[a] *Institut AIFB, Universität Karlsruhe (TH), Karlsruhe, Germany*
E-mail: {voelker, vrandecic, sure}@aifb.uni-karlsruhe.de
[b] *Universität Kassel, Germany*
E-mail: hotho@cs.uni-kassel.de

Abstract. OntoClean is an approach towards the formal evaluation of taxonomic relations in ontologies. The application of OntoClean consists of two main steps. First, concepts are tagged according to meta-properties known as rigidity, unity, dependency and identity. Second, the tagged concepts are checked according to predefined constraints to discover taxonomic errors. Although OntoClean is well documented in numerous publications, it is still used rather infrequently due to the high costs of application. Especially, the manual tagging of concepts with the correct meta-properties requires substantial efforts of highly experienced ontology engineers. In order to facilitate the use of OntoClean and to enable the evaluation of real-world ontologies, we provide AEON, a tool which automatically tags concepts with appropriate OntoClean meta-properties and performs the constraint checking. We use the Web as an embodiment of world knowledge, where we search for patterns that indicate how to properly tag concepts. We thoroughly evaluated our approach against a manually created gold standard. The evaluation shows the competitiveness of our approach while at the same time significantly lowering the costs. All of our results, i.e. the tool AEON as well as the experiment data, are publicly available.

Keywords: OntoClean, ontology evaluation

1. Introduction

Ontologies (Staab & Studer, 2004) have become an important means for knowledge interchange and integration as they provide a shared conceptualization of a domain of interest. The raise of the Semantic Web (Berners-Lee et al., 2001) fuels the need for domain-independent methodologies and guidelines for ontology engineering to efficiently and effectively build ontologies (Fernández-López et al., 1999; Noy & McGuinness, 2001; Sure & Studer, 2002; Tempich et al., 2005). Industrial strength ontology engineering additionally asks for cost-effective engineering of ontologies (Bontas et al., 2006). By nature, the realization of the Semantic Web leads to distributed nets of knowledge, and plenty of reasoning will take place on heterogeneously created ontologies. For reasoning algorithms to yield useful results the underlying ontologies need to offer a high quality. To ensure high quality, ontologies can be evaluated according to different criteria (Gómez-Pérez, 2004). Despite the need for high quality ontologies only very few approaches for ontology evaluation exist so far.

OntoClean (Guarino & Welty, 2000) is the most well-known methodology for ontology evaluation. More precisely, OntoClean enables the formal analysis of concepts and taxonomic relationships based on the philosophical notions *rigidity*, *unity*, *dependency* and *identity* (known as OntoClean meta-properties).

[*]Corresponding author.

By defining the OntoClean meta-properties, the ontology engineer can capture more of what the ontology means in a concise and formal way. For example, by stating if a certain concept is rigid or not helps to understand if this concept is meant to be used as a role, that applies to an individual, or rather as an essential type. By raising this questions and the ontology engineer answering them, the ontology will be more specific and easier to be used consistently.

The application of OntoClean consists of two main steps. First, all concepts are tagged with regards to the OntoClean meta-properties. Second, the tagged concepts are checked against predefined constraints, with constraint violations indicating potential misconceptualisations in the subsumption hierarchy. Although OntoClean is well documented in numerous publications, and its importance is widely acknowledged, it is still used rather infrequently due to the high costs for application. Several tools supporting the OntoClean methodology have been developed and integrated into ontology editors such as ODEClean for WebODE (Fernández-López & Gómez-Pérez, 2002), OntoEdit (Sure et al., 2003) or Protégé (Noy et al., 2000). The main obstacle for applying OntoClean is still the manual tagging of concepts with the correct meta-properties which requires substantial efforts of highly experienced ontology engineers.

In order to solve this problem, we have developed AEON, an approach to automatising both steps of OntoClean. By means of AEON, we can automatically tag any given ontology with respect to the OntoClean meta-properties and perform the constraint checking. For creating the taggings, our implementation of AEON[1] makes extensive use of the World Wide Web (WWW) as the currently biggest existing source of common sense knowledge. In line with several approaches such as Cimiano et al. (2005) and Etzioni et al. (2004) we defined a set of domain independent patterns which can be considered as indicators *for* or *against* Rigidity, Unity, Dependence and Identity of given concepts in an ontology.

To evaluate our automatic tagging approach we created a gold standard, i.e. a manually tagged middle-sized real-world ontology, and compared AEON results against it. A number of OntoClean experts as well as ontology engineering experts were involved in the creation of the more than 2000 taggings in the gold standard. Each expert had to tag the same ontology with OntoClean meta-properties. Although the purpose of OntoClean is to force the ontology designers to make explicit their own ontological assumptions concerning the concepts used for a given ontology, with no claim for homogeneous ontological choices, yet the extent of experts disagreement resulting from our experiments suprised us, showing again the difficulty of applying OntoClean in real-world settings. We see it as an advantage of our approach that it is based on the text corpus of the whole web, instead of being defined by a small group or a single person. As key result of our evaluation our approach compares favorably with respect to the quality of the automatic taggings while reducing significantly the time needed to do the tagging.

In order to check the OntoClean constraints automatically, we decided to reuse an existing OWL DL formalization of the OntoClean meta-properties and constraints (OntoClean ontology). We reified the tagged ontology and were then able to automatically check the tagged ontology according to the OntoClean ontology. We expected two types of errors when analysing the inconsistencies. First, the tagging of a concept is incorrect, and second, the corresponding taxonomic relationship is incorrect. We found both kinds of errors in our experimental data and looked at some of the errors in more detail to understand some rationale behind. In the end, we were thus able to point the creators of the ontology to the inconsistencies and we requested changes in the ontology to fix them.

The outline of the paper is as follows. In the next section, we briefly introduce the OntoClean meta-properties and the most important OntoClean constraints. In Section 3, we present the conceptual ideas of the AEON tool which creates meta-property taggings in an automated way. We start by describing

[1] http://ontoware.org/projects/aeon/.

the architecture of its implementation which based on the application of patterns to the Web. For each meta-property we defined a set of patterns allowing to obtain evidence for or against a certain meta-property tagging (e.g. `+R` or `-R`). We give some examples for those patterns and describe the intuition behind them. In Section 4.2 we then present our evaluation of AEON. In particular, we illustrate the methodology we adopted for the creation of a gold standard, report lessons learned, describe our technical settings and present the results of the experiment in which we compared the automatically created taggings against the gold standard. In Section 5, we describe AEON's support for the second step of OntoClean, i.e. the checking of constraints based on the meta-property taggings. We also present some examples for the kind of inconsistencies we found in the tagged ontology. Finally, we discuss some related work (cf. Section 6) and conclude in Section 7.

2. OntoClean in theory

We provide a brief introduction to OntoClean, for a more thorough description refer to Guarino & Welty (2000) and Guarino & Welty (2004), for example. In the OntoClean vocabulary, *properties* are what is commonly called *concepts* or *classes* (e.g. in RDF/S or OWL). *Meta-properties* are, therefore, properties of properties. Within this work we will use the term *meta-property* in the usual OntoClean way, whereas we will refrain from using the term *property* but rather stick to the more common term *concept*.

2.1. OntoClean process

Applying the OntoClean methodology consists of two main steps.

- First, every single concept of the ontology to be evaluated is tagged with occurrences of the core meta-properties, which are described below. Thus, every concept has a certain tagging like `+R+U-D+I`, where for example `+R` denotes that a concept carries *Rigidity* and `+U` denotes that the concept carries *Unity*. We call an ontology with tagged concepts a tagged ontology (wrt. OntoClean, to be precise).
- Second, after the tagging, all subsumption relations of the ontology (in the following also called subClassOf relations) are checked according to predefined constraints. Any violation of a constraint indicates a potential misconceputalisations in the subsumption hierarchy.

The key idea of OntoClean is to constrain the possible taxonomic relations by disallowing subsumption relations between specific combinations of tagged concepts. This way, OntoClean provides a unique approach by formally analyzing the concepts intensional content and their subsumption relationships. In other words, applying OntoClean means comparing the taxonomical part of a tagged ontology versus a predefined ideal taxonomic structure which is defined by the combination of meta-properties and constraints.

After performing the two steps the result is a tagged ontology and a (potentially empty) list of misconceptualisations. According to this list an ontology engineer may repair (in an OntoClean sense) the ontology. It may make sense to apply OntoClean after repairing and evolving the ontology again.

2.2. OntoClean meta-properties

As already indicated, the main ingredients of OntoClean are four meta-properties and a number of rules. The four meta-properties are: *rigidity (R)*, *unity (U)*, *dependence (D)* and *identity (I)*. They base on philosophical notions as developed by Strawson and others, even dating back to Aristotle. Here we will offer a short description of these meta-properties.

Rigidity. Rigidity is based on the notion of *essence*. A concept is essential for an instance *iff* such instance is necessarily an instance of this concept, in all worlds and at all times. *Iff* a concept is essential to all of its instances, the concept is called rigid and is tagged with +R. *Iff* it is not essential to some instances, it is called non-rigid, tagged with -R. An anti-rigid concept is one that is not essential to all of its instances. It is tagged ∼R. An example of an anti-rigid concept would be TEACHER, as no teacher has always been, nor is necessarily, a teacher, whereas HUMAN is a rigid concept because all humans are necessarily humans and neither became nor can stop being a human at some time.

Unity. Unity tells us what is part of the object, what is not, and under what conditions the object is *whole* (Guarino & Welty, 2004). This answer is given by an *unity criterion* (UC), which describes the conditions that most hold among the parts of a certain entity to consider that entity as a whole. For example, there is an unity criterion for the parts of a human body, as we can say for every human body which parts belong to it. Concepts carrying an UC have Unity and are tagged +U else -U.

Dependence. A concept C_1 is dependent on a concept C_2 (and thus tagged +D), *iff* for every instance of C_1 an instance of C_2 must exist. An example for a dependent concept would be FOOD, as instances of FOOD can only exist if there is something for which these instances are food. This does not mean that an entity being food ceases to exist the moment all animals die out that regarded it as food, it just stops being food.

Identity. A concept with identity is one, where the instances can be identified as being the same at any time and in any world, by virtue of this concept. This means that the concept carries an *identity criterion* (IC). It is tagged with +I, and with -I otherwise. It is not important to answer the question of what this IC is (this may be hard to answer), it is sufficient to know that the concept carries an IC. For example, the concept HUMAN carries an IC, as we are able to identify someone as being the same or not, even though we may not be able to say what IC we actually used for that. On the other hand, a concept like RED would be tagged -I, as we cannot tell instances of red apart because of its color.

OntoClean differentiates between the two tags I and O, whereby the first means, that the concept simply carries an IC and the second that it carries an own IC. The difference is not relevant for this work, as the tagging +O may just be treated like the tagging +I, as +O implies +I anyway and there are no subsumption rules about the tag O.

2.3. OntoClean constraints

A number of OntoClean rules is applied on the tagged ontology. We may use the existing OntoClean rules to check a tagged ontology for consistency. Here, we will give some illustrative example for these rules. For a full list refer to Guarino & Welty (2004). As shown in (Sure et al., 2003) such rules can be formalized as logical axioms and validated automatically by an inference engine.

∼R cannot subsume +R. Having a concept C subsuming the concept D, with C tagged $\sim R$ and D tagged $+R$, would lead to the following inconsistency: D must always hold true for all of its instances. D, as a subsumed concept, would always imply C for all of its instances. Therefore, there are at least some instances of C that are necessarily C as they are D. Thus C cannot be anti-rigid, as the tagging says, because this would mean that it is not necessarily true for any of its instances – which would be a contradiction. An example is FOOD, an anti-rigid concept, subsuming APPLE, a rigid concept. As it is explained in (Guarino & Welty, 2004), nothing is essentially food – it may or may not be eaten. On the other hand, an apple is always an apple. But if apples were subsumed by FOOD, there would be some food that would essentially be food, namely apples, since every apple would always be an apple and thus food – which would be a contradiction to the statement that no food is essentially food.

+I cannot subsume −I. If this rule was broken, it would mean that instances of the subsumed concept can not be identified – although they are also instances of the subsuming concept, which explicitly allows for the identification of the instances. This would be a contradiction, revealing an error in our taxonomy (or tagging).

+D cannot subsume −D. FOOD is an example for a dependent concept. Modeling the concept CANDY, we decide that everything with more than 20% sugar is candy, thus the concept would be independent. We let FOOD subsume CANDY, and the formal analysis shows this rule is broken. This points us to either an error in the taxonomy or in the tagging. In the last example we see that the quality of the taxonomical analysis is only as good as the quality of the applied tagging.

3. Tagging approach

Our approach to the automatic assignment of meta-properties according to the OntoClean methodology is based on three fundamental assumptions. First, we believe that the nature of concepts is to some degree reflected by human language and what is said about instances of these concepts in the language corpus. Because of this, we consider statistics about the occurrences of lexico-syntactic patterns (see Section 3.2) as a feasible means to capture the meta-properties of ontological concepts. Second, in line with similar approaches by Grefenstette (1999), Keller et al. (2002), Resnik & Smith (2003), Cimiano et al. (2004) and Cimiano et al. (2005) we think that using the Web as a corpus is an effective way of addressing the typical data sparseness problem one encounters when working with natural language corpora. Finally, from our point of view, the Web being the biggest source of common-sense knowledge available constitutes a perfect basis for computational comprehension of human intuition as to the philosophical notions of essence, unity and identity.

3.1. Architecture and implementation

Our core contribution is the development and the evaluation of the tagging component of AEON. The tagging component matches lexico-syntactic patterns on the Web to obtain positive and negative evidence for rigidity, unity, dependence and identity of concepts in an RDFS or OWL ontology. The architecture is roughly depicted by Fig. 1. It consists of an *evaluation component*, which is responsible for training and evaluation, a *classifier* for mapping given sets of evidence to meta-properties such as `+R` or `-U`, a *pattern library* and a *search engine wrapper*.

The pattern library is initialized by means of an XML file containing a set of abstract patterns for each meta-property (see Listing 1 for an example). Each of these patterns includes a specification of the type of evidence it produces, e.g., negative evidence for rigidity. Moreover, it contains a declaration of one or more variables and a set of Web queries which can be instantiated by replacing the regarding variables by the labels of the concepts to be analysed. Finally, a linguistic filter, i.e. a regular expression over tokens and part-of-speech tags, is defined for filtering the results obtained by the above mentioned queries (see Section 3.3).

Given a set of instantiated patterns (e.g. "*is no longer an apple*") the search engine wrapper uses the Google™ API in order to retrieve web pages or snippets, i.e. parts of web pages containing the regarding search string, from the Web. For normalization purposes (see below) it also queries the web for all occurrences of the regarding concept, such as "*apple*" for example.

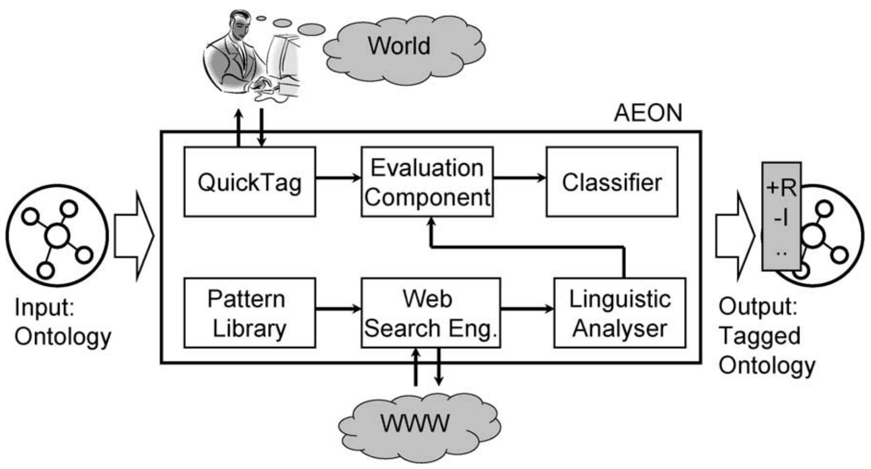

Fig. 1. Tagging architecture of AEON.

```
<pattern>
  <variable name="x" />
  <evidence type="false" for="R" />
  <google regex="is\t\w+ no\t\w+ longer\t(DT\w+\t)?(NN|NP|NNS|NPS)
     x\t[^(NN|NP|NNS|NPS)]">
    <query string="is no longer a x" />
    <query string="is no longer an x" />
    <query string="is no longer x" />
  </google>
</pattern>
```

Listing 1. Negative evidence for rigidity (R).

The linguistic analyser provides methods for tokenization, lemmatizing and part-of-speech (POS) tagging, which are required for some fundamental preprocessing of the snippets and HTML pages obtained from the Web and for an appropriate matching of the linguistic patterns described above. By what we call *Linguistic Filtering* we analyse, e.g., all those snippets returned by Google™, which satisfy the query "*is no longer a computer*" (cf. Listing 1). If the regular expression associated with the query does not match, the particular snippet is not counted as a hit and thus does not provide any evidence with respect to the rigidity of COMPUTER. This way, we avoid, for instance, false matches in case of statements such as "*He is no longer a computer hacker*". The assumption underlying our implementation of linguistic filtering is that any common or proper noun immediately following the concept variable within the regular expression (e.g. by the term "computer") would imply that the lexicalization of the concept does not represent the head of the corresponding noun phrase, i.e. the part which carries the essential semantic information. Patterns which are structurally different from the one shown by Listing 1 may therefore require a similar treatment of preceding noun phrases, e.g., to avoid false evidence for the unity of EMPLOYEE when a phrase like "*the computer of an employee consists of*" is returned by Google™. For the same reasons, linguistic filtering is also applied in the normalization process.

Finally, for each pattern i contained in the above mentioned pattern library the positive or negative evidence *evidence*(p, i, c) for a concept c having a certain meta-property $p \in \{R, U, D, I\}$ is given by:

$$\mathit{evidence}(p,i,c) = \frac{\sum_{q \in Q_i} \mathit{lf}(\mathit{hits}(q_c))}{\mathit{lf}(\mathit{hits}(c))},$$

where Q_i is the set of queries associated with pattern i, q_c is the instantiation of query q for concept c, and *hits*(q_c) as well as *hits*(c) represent the number of hits obtained for q_c or c, respectively. *lf* is a function implementing the linguistic filtering described above.

Given a concept c and the evidence values obtained for all patterns the decision whether or not a meta-property p applies to c is made by a classifier. A set of classifiers – one for each meta-property – has been trained on a small number of examples provided by human annotators (cf. Section 4.2). The manual effort rests with the creating of a gold standard ontology and classifiers to be trained on this ontology.

3.2. Patterns

During the last decades, lexico-syntactic patterns have become generally accepted as an effective means for extracting various types of lexical or ontological relationships such as hyponymy and meronymy (cf. Hearst, 1992; Charniak & Berland, 1999; Hahn & Schnattinger, 1998). Nevertheless, there has been little if any work on the use of pattern-based approaches towards the extraction of meta-properties, i.e. properties of concepts or relations. In order to find a suitable set of patterns, we therefore evaluated an overall number of 19 pattern candidates[2] against the original OntoClean example ontology (Guarino & Welty, 2004) before finally choosing a small subset of particularly promising patterns for the evaluation of our approach. All of these patterns are domain-independent, thus being well suited for the WWW as a very heterogeneous corpus.

Rigidity. The intuition behind the patterns we defined for Rigidity is the following: if any individual can become or stop being a member of a certain class, then it holds that the membership of this class, e.g. the property *being a student*, is not essential for all its individuals. Therefore, we can obtain *negative* evidence with respect to Rigidity from the following patterns:

```
is no longer (a|an)? CONCEPT
became (a|an)? CONCEPT
while being (a|an)? CONCEPT
```

Unity. As explained in Section 2, a concept is tagged with $+U$ if for each of its instances all parts can be identified and if they share a common Unity Criterion which holds true for exactly these parts. Because of this, in order to determine whether a given concept has unity or not we have to find answers to questions such as "*what is part of an object? and what is not?*" or "*under which conditions is the object a whole?*". If we can answer these questions for at least most of the instances of the concept, we can take this as *positive* evidence for Unity.

[2]The initial set of patterns is included in the AEON release which can be downloaded from http://ontoware.org/projects/aeon/.

```
part of (a|an)? CONCEPT
```

Moreover, since instances of concepts which are not countable usually do not carry a unity criterion, we can get *positive* evidence for Unity by searching for the following patterns:

```
(one|two) CONCEPT
```

Of course, `one` and `two` seem to be somewhat arbitrary, but since Google™ is not yet able to process queries containing regular expressions we had to confine ourselves to what we considered as the most frequent of all possible variations of this pattern.

Similarly, *negative* evidence can be obtained by a pattern which indicates non-countability of a concept.

```
amount of CONCEPT
```

Identity. According to Guarino & Welty (2004), identity criteria are often based on the identity of certain parts or characteristics. Therefore, the following patterns can provide, in general, some positive evidence for Identity:

```
CONCEPT consists of (two|three) parts
CONCEPT is composed of (two|three) parts
```

Additional *positive* evidence for identity can be obtained by the rather straight-forward pattern:

```
CONCEPT is identified by
```

Negative and *positive* evidence, respectively, can be obtained by these merely linguistic patterns checking whether the name of the concept is an adjective or a noun.

Both patterns are matched on the results of Google™ queried for nothing but the concept name. Please note that linguistic preprocessing as described in Section 3.1 is required to allow this kind of lexico-syntactic pattern matching, since these patterns assume the text to be an alternate sequence of words and POS tags. The tags *JJ*, *JJR* and *JJS* indicate an adjective, whereas *NN*, *NP*, *NNS* and *NPS* are indicators for a common or proper noun.

```
(JJ|JJR|JJS) CONCEPT
(NN|NP|NNS|NPS) CONCEPT
```

Also, countability means that the instances of a concept are obviously identifiable (or else they would not be countable). Therefore we reuse some of the patterns that we have already used as positive or negative evidence for Unity. Note that even identical feature vectors would lead to different classification results for Unity and Identity as long as the training data is different.

```
(one|two) CONCEPT
amount of CONCEPT
```

Dependence. Among the meta-properties rigidity, unity, dependence and identity, we consider dependence as the most difficult one to learn automatically. Maybe, this is because of the fact that relational knowledge, i.e. knowledge involving more than one concept, is required in order to detect dependence. Nevertheless, we tried to capture dependence of concepts by the following pattern:

```
cannot be (a|an)? CONCEPT without
```

Additional patterns. Thanks to the flexible architecture of AEON adding further patterns is a very easy task which simply requires an extension of the pattern library in XML format. This way, we also tested several more patterns which we initially had in mind. However, preliminary testing in Google™ mostly resulted in a very small number of hits that would have lowered the efficiency of AEON without significantly improving its output.

3.3. Discussion

The described approach is original and poses quite a number of problems. We solved many of them, but some remain for further research. Both kinds of problems are described in this section.

Certain patterns could return a lot of inappropriate evidence. Searching for the fragment "*is no longer a computer*" would also return "*is no longer a computer hacker*", which is false evidence about the Rigidity of computers. To solve this problem we introduced linguistic preprocessing and patterns that recognize "computer" not being the subject of the given example. Thus we can get rid of a lot of false evidence.

The other problem occurs with high level, abstract or seldom used concepts: they just do not return hits, or return only a small, and thus usually unreliable number of evidence. However, we do not consider this as a big problem in general, since this kind of very abstract concepts mostly appear in upper-level ontologies which are typically smaller and less dynamic than domain ontologies. If we do not get any hits, the concept will not be part of possible constraint errors. So it does not really bother the user with wrong warnings but rather simply ignores this concept.

A much bigger problem is given by the highly ambiguous nature of human language. So far, our approach does not distinguish between different concepts which could be meant by the word "glass", for example. Whereas the "glass" which can be used to drink water certainly has Unity, the "glass" windows are made of does not have Unity. Linguistic patterns do not help in this case. We will try to solve this problem by comparing the context of the word – given by a Google™ snippet or a Web page – with the semantic neighborhood of the regarding concept.

Natural language is not as strict and formal as the OntoClean meta-properties. The best known example is the English verb *to be*, which can have various meanings based heavily on context, like subsumption, definition or constitution. But exactly these different meanings play a crucial role within the OntoClean methodology. Thus, the translation of the OntoClean definitions of meta-properties to commonly used language patterns was quite challenging. With the patterns given in this section we hope to have achieved a good balance between language ambiguity, pragmatic indication of meta-properties and number of occurrences for a wide range of concepts.

An open issue is the differentiation between Non-, Anti- and Semi-Rigidity by automatic means. Right now we just consider Rigidity and Non-Rigidity, but the more detailed division may lead to an even better evaluation of the ontology.

4. Tagging evaluation

4.1. Manual tagging

For the evaluation and training of our automatic methods, we needed a gold standard tagging of an ontology with the OntoClean meta-properties. Although OntoClean is already some years old and appeared in a number of publications, actual tagged ontologies were found only extremely scarcely. Our best resource was the example ontology in (Guarino & Welty, 2004) and some examples in the other publications. This amounted to about 30–40 tagged concepts. Welty et al. (2004) describe the creation of another ontology evaluated with OntoClean, but this is not publicly available. To the best of our knowledge there are no further available tagged ontologies.

For this reason, we had to acquire a new gold standard and searched for a generic, domain-independent ontology with a reasonably big number of concepts. By choosing these selection criteria we intended to ensure the reusability of the gained tagging experience and the trained classifiers for future ontology evaluation efforts. We finally decided to use the freely available PROTON[3] (Terziev et al., 2004) ontology, which has been developed within the European IST project SEKT[4] to facilitate the use of background or preexisting knowledge for automatic metadata generation. After merging the *System*, *Top* and *Upper* modules of PROTON we obtained an ontology consisting of 266 concepts, as diverse as ACCIDENT, ALIAS, HAPPENING or WOMAN.

We asked three methodology and ontology engineering experts to tag the PROTON ontology according to the OntoClean methodology, because we wanted to base the evaluation of our own techniques on this human tagging. Most of them told us that based on their experience with OntoClean the manual tagging of an ontology such as PROTON with Rigidity, Unity, Identity and Dependence would take more than one week. Some even considered this as an effort of one month – which would of course render any evaluation of the ontology far too expensive to be efficient.

Finally, we were able to convince two of them to create a manual tagging of PROTON, whereas the third tagging was done by one of the authors of this paper. Each of these experts tagged 266 concepts with three to four meta-properties – which gave us a total number of 2926 taggings. The tagging process itself was very strenuous, and often uncertainty arose. Decisions were debatable and the documentation of OntoClean was open to interpretation. The experts tagged the ontology in the given time of four to six hours, but they achieved an agreement far lower than expected (refer to Table 2). We assume that the allocated time presents a realistic investment in an ontology building setting. Concepts similar to those in the example ontology in (Guarino & Welty, 2004) were often tagged consistently, but the agreement on the other concepts was low (close to the baseline given by random tagging). This suggests that the experts rather worked by analogies (not surprisingly, given the time constraints) to the examples (an approach that is very common for humans) than by applying the definitions of the meta-properties.

Taking into account that OntoClean is only a method to evaluate the taxonomic relationships of an ontology, these findings point to doubts concerning the efficiency of manual tagging. Although there are some implementations that support the tagging with OntoClean meta-properties in existing ontology engineering environments (refer to Section 6), the number of actually tagged ontologies is obviously far too low. This again points to a discrepancy between the expected work and the expected benefit of using OntoClean. To turn OntoClean into a feasible and more frequently used ontology evaluation method, a far more precise and yet broader understandable description of OntoClean must become available, or else

[3]http://proton.semanticweb.org.

[4]http://www.sekt-project.com.

an approach for the automatic tagging of concepts must lower the time to tag ontologies dramatically. The latter approach requires far less training to the individual ontology engineer and evaluator.

The upper level ontology DOLCE was created with the principles of OntoClean in mind. WordNet on the other hand was not created with ontological categories in mind, but rather adhering to linguistic structures. Aligning those two should reveal numerous errors in WordNet, by OntoClean standards, due to the different nature of the two. In Gangemi et al. (2003), where this task is described, the authors say that the alignment of DOLCE and WordNet yielded almost only constraint violations regarding rigidity and much less on all other meta-properties. Thus, it was essential to get reliable results for rigidity, more than for the other meta-properties.

Another problem is that tagging an ontology implies further ontological decisions possibly unintended by the ontology creators. Subjective point of views going further than the ontology is already committed to can be introduced through the tagging. For example, regarding the concept *Dalai Lama* we could state this concept is not rigid: a person is chosen to become the *Dalai Lama*. Thus a question of believe becomes relevant: buddhist religion claims that one does not become the *Dalai Lama*, but rather that one is the *Dalai Lama* across multiple reincarnations. Tagging an ontology, therefore, increases the ontological commitment. Our approach dodges this problem by basing the taggings on statistics over a large corpus instead of an individual or small group's subjective point of view, and then pointing to possible errors in the ontology.

4.2. Setting and results

As described in Section 4.1 we decided to use the System, Top and Upper module of the PROTON ontology for the evaluation of our approach. The merged ontology consists of 266 concepts, most of them annotated with a short natural language description. The list of all concepts together with their descriptions was given to three human annotators in the following called A_1, A_2 and A_3. All of them were assumed to be experts in using the OntoClean methodology. Nevertheless, whereas Rigidity, Identity and Dependence were considered by all annotators, only two of them also assigned Unity tags to some of the concepts. Table 1 shows the number of concepts and their corresponding taggings created by each of the human annotators. The data sets labeled A_1/A_2, A_1/A_3, A_2/A_3 were obtained by the intersection

Table 1

Tagged concepts

	R			U			I			D		
	$+$	$-$	$\sim$	$+$	$-$	$\sim$	$+$	$-$	$\sim$	$+$	$-$	$\sim$
A_1	147	69	50	156	81	29	194	72	0	151	113	0
A_2	208	39	0	103	138	3	189	58	0	31	216	0
A_3	201	64	0	0	0	0	223	42	0	63	1	0
avg	185.3	57.3	16.7	86.3	73.0	10.7	202.0	53.3	0.0	81.7	110.0	0.0
A_1/A_2	122	3	20	77	61	11	143	21	0	23	94	0
A_1/A_3	125	27	15	0	0	0	171	19	0	47	1	0
A_2/A_3	161	14	0	0	0	0	163	12	0	9	0	0
avg	136.0	14.7	11.7	25.7	20.3	3.7	159.0	17.3	0.0	26.3	31.7	0.0
$A_1/A_2/A_3$	106	2	6	0	0	0	126	8	0	9	0	0

Table 2

Human agreement

	A_1/A_2		A_1/A_3		A_2/A_3		$A_1/A_2/A_3$	
	Relaxed	Strict	Relaxed	Strict	Relaxed	Strict	Relaxed	Strict
R (%)	58.7	50.6	63.0	57.4	71.1	71.1	46.3	43.9
U (%)	61.1	56.6	N/A	N/A	N/A	N/A	N/A	N/A
I (%)	66.4	66.4	71.7	71.7	71.1	71.1	54.5	54.5
D (%)	48.6	48.6	75.0	75.0	15.0	15.0	15.0	15.0
avg (%)	58.7	55.6	69.9	68.0	52.4	52.4	38.6	37.8

of two of the single data sets.[5] Obviously, $A_1/A_2/A_3$, which is the intersection of all three data sets – the set of concepts which are tagged identically by all human annotators – is extremely sparse.

In order to illustrate how difficult it was for the human annotators to tag the ontology according to the OntoClean methodology we measured the human agreement between the data sets (cf. Table 2). *Strict* means that two taggings were considered equal only if they were totally identical. *relaxed* means that $-$ and $\sim$ were considered the same. Since our approach so far does not distinguish between Semi- and Anti-Rigidity, for example, the strict agreement can be neglected for the following evaluation. As shown by Table 2 the average human agreement is extremely low, which means close to the random baseline and sometimes much lower than the results we obtained by automatic tagging. Given these figures indicating the difficulty of this task, we believe any kind of automatic support could be of great use for formal ontology evaluation. Just to give one example on the problem of agreement we look at the concept ISLAND: two annotators tagged it with being rigid, one said it is non-rigid. The third tagger explained, that there exist islands that can turn into peninsulas due to tidal effects. Such problems were encountered throughout the PROTON ontology. A complete list of the taggings can be downloaded at the AEON website.[6]

Baseline. In order to obtain an objective baseline for the evaluation of AEON which is statistically more meaningful than the human agreement (see Table 2) we computed a random baseline for the F-measure as follows: let x be the overall number of concepts to be tagged, p the number of positive and $n = x - p$ the number of negative examples. Given a random tagging for all n concepts we can assume that half of them are tagged as $+$ and equally many are tagged as $-$. Of course, the fraction of positives within the whole data set tends to be the same as in each of the randomly chosen subsets S_+ and S_- of size $\frac{n}{2}$. Therefore, the number of true positives (*TP*) and true negatives (*TN*) is given by $TP = \frac{p}{x} * \frac{x}{2} = \frac{p}{2}$ and $FP = (1 - \frac{p}{x}) * \frac{x}{2} = \frac{x}{2} - \frac{p}{2} = \frac{x-p}{2} = \frac{n}{2}$ whereas the false positives (*FP*) and false negatives (*FN*) can be computed by $TN = \frac{n}{x} * \frac{x}{2} = \frac{n}{2}$ and $FN = (1 - \frac{n}{x}) * \frac{x}{2} = \frac{x}{2} - \frac{n}{2} = \frac{x-n}{2} = \frac{p}{2}$.

Obviously, the Precision P_+ for the positive examples (for example, all concepts tagged as $+R$) is given by $P_+ = TP/(TP + FP)$, whereas the Precision for the negative examples can be obtained by $P_- = TN/(TN + FN)$. Recall can be computed by $R_+ = TP/(TP + FN)$ and $R_- = TN/(TN + FP)$, respectively.

Given Recall and Precision we can obtain the F-measure for positive and negative examples by $F_+ = \frac{2 * P_+ * R_+}{P_+ + R_+}$ and $F_- = \frac{2 * P_- * R_-}{P_- + R_-}$. This leads to a *macro-average F-measure* of $F = \frac{1}{2} * (F_+ + F_-)$, which

[5]In case a concept was tagged by only one of the annotators we assumed the agreement to be equal to the tag assigned by this annotator. Please note that all the data sets are available for download from http://ontoware.org/projects/aeon/.

[6]http://ontoware.org/projects/aeon.

Table 3
Random baseline (F-measure)

	R			U			I			D		
	+	−	M-avg	+	−	M-avg	+	−	M-avg	+	−	M-avg
A_1	52.5	47.2	49.9	54.0	45.3	49.6	59.3	35.1	47.2	53.5	45.9	49.7
A_2	62.7	24.0	43.4	45.8	53.6	49.7	60.5	32.0	46.2	20.1	63.6	41.8
A_3	60.1	32.6	46.4	N/A	N/A	N/A	62.7	24.1	43.4	66.3	3.0	34.7
avg	58.4	34.6	46.6	49.9	49.5	49.7	60.8	30.4	45.6	46.6	37.5	42.1
A_1/A_2	62.7	24.1	43.4	50.8	49.1	50.0	63.6	20.4	42.0	27.7	61.8	44.7
A_1/A_3	60.0	33.5	46.7	N/A	N/A	N/A	64.3	16.7	40.5	66.2	4.0	35.1
A_2/A_3	64.8	13.8	39.3	N/A	N/A	N/A	65.1	12.1	38.6	66.7	N/A	N/A
avg	62.5	23.8	43.1	50.8	49.1	50.0	64.3	16.4	40.4	53.5	32.9	39.9
$A_1/A_2/A_3$	65.0	12.3	38.7	N/A	N/A	N/A	65.3	10.7	38.0	66.7	N/A	N/A

we consider as a reasonable baseline for the evaluation of our approach. A detailed overview of the concrete baselines we determined for all data sets is given by Table 3.

Settings. Since we decided to evaluate our system separately for R, U, I and D, we made $7*4*2 = 56$ experiments (for each of the human annotators and the agreement data sets, each of the meta-properties, as well as with and without linguistic filtering) using a number of Weka[7] classifiers. In order to detect the limitations of our approach and to see what we can potentially get out of the data which we are able to provide, we first tried many different types of classifiers, such as Support Vector Machines, Bayesian classifiers and Decision Trees. Since the latter turned out to perform best we finally decided to focus on the class of Decision Trees – among them ADTree, RandomForest and J48, for example. The features given to these classifiers were sets of evidences obtained by all patterns for the regarding meta-property (see Section 3.1). Precision, Recall and F-measure for both positive and negative examples as well as the macro-average F-measure were determined by a 10-fold cross-validation (Kohavi, 1995) with stratified sampling (cf. Särndal et al., 2003). That is we divided each of the data sets into $k = 10$ partitions, trained on $k - 1$ and tested on the remaining one part of the data set. This procedure was repeated k times before the results were averaged to compute the overall performance or error rate, respectively. In the general case, such k-fold cross validation leads to a better error estimation than a single split into test and training data set, in particular if only very few samples are available.

Note that for training and evaluation we only used those concepts which were annotated in the regarding data set and for which we obtained at least some evidence. The percentage of tests which failed, because we did not get any Google™ hits for the instantiated patterns was about 20% for rigidity, 5% for identity and around 10% for unity. Because of this, in many cases the number of examples we gave to the classifiers was extremely low – especially for the agreement data sets A_1/A_2, A_1/A_3, A_2/A_3 and $A_1/A_2/A_3$. The reason why the results are nevertheless very promising, certainly is the good quality of the classification features which we achieved by using a pattern-based approach.

Results. One of the main findings of our experiments was that linguistic filtering can help in the task of pattern-based ontology evaluation. Whereas the differences between the results with and without

[7]http://www.cs.waikato.ac.nz/ml/weka/.

Table 4
Rigidity (best results with linguistic filtering)

	P		R		F					Classifier
	$+$	$-$	$+$	$-$	$+$	$-$	M-avg	Baseline	No LF	
A_1	65.6	57.5	71.7	50.5	68.5	53.8	61.2	49.9	63.5	RandomForest
A_2	87.7	41.6	91.5	32.3	89.6	36.4	63.0	43.4	65.9	RandomTree
A_3	79.6	36.4	82.6	32.0	81.1	34.0	57.6	46.4	52.9	RandomTree
avg	77.6	45.2	81.9	38.3	79.7	41.4	60.6	46.6	60.8	
A_1/A_2	91.3	60.0	94.0	50.0	92.6	54.5	73.6	43.4	72.1	RandomForest
A_1/A_3	80.8	39.3	83.2	35.5	82.0	37.3	59.6	46.7	62.1	ADTree
A_2/A_3	93.8	20.0	93.8	20.0	93.8	20.0	56.9	39.3	61.4	RandomTree
avg	88.6	39.8	90.3	35.2	89.5	37.3	63.4	43.1	65.2	
$A_1/A_2/A_3$	94.5	0.0	100.0	0.0	97.2	0.0	48.6	38.7	47.9	NBTree

Table 5
Identity (best results with linguistic filtering)

	P		R		F					Classifier
	$+$	$-$	$+$	$-$	$+$	$-$	M-avg	Baseline	No LF	
A_1	75.9	34.9	78.3	31.9	77.1	33.3	55.2	47.2	51.7	RandomForest
A_2	80.9	37.5	80.9	37.5	80.9	37.5	59.2	46.2	54.3	ADTree
A_3	88.7	38.6	87.5	41.5	88.1	40.0	64.1	43.4	52.6	RandomForest
avg	81.8	37.0	82.2	37.0	82.0	36.9	59.5	45.6	52.9	
A_1/A_2	88.7	25.0	89.3	23.8	89.0	24.4	56.7	42.0	51.5	RandomTree
A_1/A_3	92.6	60.0	97.6	31.6	95.0	41.4	68.2	40.5	53.3	NBTree
A_2/A_3	96.8	60.0	97.5	54.5	97.1	57.1	77.1	38.6	59.8	RandomForest
avg	92.7	48.3	94.8	36.3	93.7	41.0	67.3	40.4	54.9	
$A_1/A_2/A_3$	97.5	55.6	96.7	62.5	97.1	58.8	78.0	38.0	58.4	RandomForest

linguistic filtering tend to be marginal for Rigidity and Unity (cf. Tables 4 and 6), linguistic filtering clearly improved the results for Identity and Dependence (Tables 5 and 7). And in fact, the only time the baseline was missed in our experiments was in a run-through without linguistic filtering (see Table 7).

Another interesting result of the evaluation was that (except for the very sparse data sets of Unity and Dependence) our system performed significantly better on the agreement, i.e. the intersection of two or three data sets. This is probably due to the fact that those concepts which were tagged identically by at least two of the human annotators are easier to tag – maybe, because they are less ambiguous.

The overall conclusion we draw from the evaluation of AEON was that despite the weaknesses of our pattern-based approach (see Section 3.3) the first results are already very promising. Given the small amount of training data we had and the fact that we used standard Weka classifiers with marginal parameter tuning we hope to get even better results in future experiments after analyzing the data in more detail.

Table 6

Unity (best results with linguistic filtering)

	P		R		F					Classifier
	$+$	$-$	$+$	$-$	$+$	$-$	M-avg	Baseline	No LF	
A_1	66.9	49.4	72.1	43.2	69.4	46.1	57.7	49.6	50.9	DecisionStump
A_2	51.7	63.2	34.5	77.8	41.4	69.8	55.6	47.7	60.1	ADTree
A_3	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
avg	59.3	56.3	53.3	60.5	55.4	58.0	56.7	48.7	55.5	
A_1/A_2	54.8	51.9	60.6	45.9	57.6	48.7	53.1	50.0	57.0	RandomForest
A_1/A_3	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
A_2/A_3	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
avg	54.8	51.9	60.6	45.9	57.6	48.7	53.1	50.0	57.0	
$A_1/A_2/A_3$	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Table 7

Dependence (best results with linguistic filtering)

	P		R		F					Classifier
	$+$	$-$	$+$	$-$	$+$	$-$	M-avg	Baseline	No LF	
A_1	67.5	42.9	69.2	40.9	68.4	41.9	55.1	49.7	57.7	RandomForest
A_2	33.3	81.9	38.5	78.3	35.7	80.0	57.9	41.8	42.6	RandomForest
A_3	100.0	0.0	100.0	0.0	100.0	0.0	50.0	34.7	50.0	RandomForest
avg	66.9	41.6	69.2	39.7	68.0	40.6	54.3	42.1	50.1	
A_1/A_2	42.9	64.0	25.0	80.0	31.6	71.1	51.3	44.7	36.4	DecisionStump
A_1/A_3	100.0	0.0	100.0	0.0	100.0	0.0	50.0	35.1	50.0	RandomForest
A_2/A_3	100.0	0.0	100.0	0.0	100.0	0.0	50.0	N/A	50.0	RandomForest
avg	81.0	21.3	75.0	26.7	77.2	23.7	50.4	39.9	43.2	
$A_1/A_2/A_3$	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

5. Constraint checking

5.1. Implementation

Creating correct taggings is only one part (although the harder one) in discovering taxonomic errors according to the OntoClean methodology. Equipped with these taggings we are able to check the hierarchical part of the ontology with regards to the meta-property constraints defined by OntoClean. In order to check these constraints automatically, we decided to reify the ontology and use a formalization of the constraints in OWL DL[8] in order to check the reified ontology. This section describes the approach in detail. The approach is based on the work described in (Welty, 2006).

We took the formalisation of OntoClean in OWL DL[9] as depicted in Table 8. Axioms (1)–(7) formalize the reification of the subsumption axioms with the transitive subClassOf relation and its inverse, domain,

[8]http://www.w3.org/TR/owl-ref/.

[9]The ontology is taken from http://www.ontoclean.org/ontoclean-dl-v1.owl.

Table 8

OntoClean constraints meta-ontology in DL

$\textsc{Trans}(\mathsf{subClassOf})$			(1)
$\mathsf{hasSubClass}$	$\equiv$	$\mathsf{subClassOf}^{-1}$	(2)
$\mathsf{subClassOf}$	$\equiv$	$\mathsf{hasSubClass}^{-1}$	(3)
$\top$	$\sqsubseteq$	$\forall\mathsf{hasSubClass}.\textsc{Class}$	(4)
$\top$	$\sqsubseteq$	$\forall\mathsf{subClassOf}.\textsc{Class}$	(5)
$\top$	$\sqsubseteq$	$\forall\mathsf{hasSubClass}^{-1}.\textsc{Class}$	(6)
$\top$	$\sqsubseteq$	$\forall\mathsf{subClassOf}^{-1}.\textsc{Class}$	(7)
$\textsc{RigidClass}$	$\sqsubseteq$	$\textsc{Class}$	(8)
$\textsc{NonRigidClass}$	$\sqsubseteq$	$\textsc{Class}$	(9)
$\textsc{AntiRigidClass}$	$\sqsubseteq$	$\textsc{NonRigidClass}$	(10)
$\textsc{UnityClass}$	$\sqsubseteq$	$\textsc{Class}$	(11)
$\textsc{NonUnityClass}$	$\sqsubseteq$	$\textsc{Class}$	(12)
$\textsc{AntiUnityClass}$	$\sqsubseteq$	$\textsc{NonUnityClass}$	(13)
$\textsc{DependentClass}$	$\sqsubseteq$	$\textsc{Class}$	(14)
$\textsc{NonDependentClass}$	$\sqsubseteq$	$\textsc{Class}$	(15)
$\textsc{SortalClass}$	$\sqsubseteq$	$\textsc{Class}$	(16)
$\textsc{NonSortalClass}$	$\sqsubseteq$	$\textsc{Class}$	(17)
$\textsc{Class}$	$\equiv$	$(\textsc{NonRigidClass} \sqcup \textsc{RigidClass})\sqcap$ $(\textsc{NonDependentClass} \sqcup \textsc{DependentClass})\sqcap$ $(\textsc{SortalClass} \sqcup \textsc{NonSortalClass})\sqcap$ $(\textsc{UnityClass} \sqcup \textsc{NonUnityClass})$	(18)
$\textsc{NonDependentClass} \sqcap \textsc{DependentClass}$	$\sqsubseteq$	$\bot$	(19)
$\textsc{NonSortalClass} \sqcap \textsc{SortalClass}$	$\sqsubseteq$	$\bot$	(20)
$\textsc{UnityClass} \sqcap \textsc{NonUnityClass}$	$\sqsubseteq$	$\bot$	(21)
$\textsc{NonRigidClass} \sqcap \textsc{RigidClass}$	$\sqsubseteq$	$\bot$	(22)
$\textsc{RigidClass}$	$\equiv$	$\forall\mathsf{subClassOf}.\neg\textsc{AntiRigidClass}$	(23)
$\textsc{UnityClass}$	$\sqsubseteq$	$\forall\mathsf{subClassOf}.\neg\textsc{AntiUnityClass}$	(24)
$\textsc{DependentClass}$	$\sqsubseteq$	$\forall\mathsf{hasSubClass}.\textsc{DependentClass}$	(25)
$\textsc{SortalClass}$	$\sqsubseteq$	$\forall\mathsf{hasSubClass}.\textsc{SortalClass}$	(26)

and range. Axioms (8)–(17) describe the tagging hierarchy as described in Section 2.2 (note that besides axioms (10) and (13) these axioms are redundant due to axiom (18). The class $\textsc{Sortal}$ describes the identity meta-property). Axioms (18)–(22) state the complete partition of all classes by the tags, i.e. each class is either `+R` or `-R`, `+U` or `-U`, etc. Finally, the axioms (23)–(26) describe the actual constraints, as partially given in Section 2.3. Just to take an example, axiom (25) describes the constraint with regards to dependency, i.e. `+D` cannot subsume `-D`. Note that the ontology, in particular axioms (1) and (23), infer that all rigid classes have to be subsumed by rigid classes.

We took the tagging created in the previous sections and formalised them in OWL DL as well, taking each tagging and interpreting it as a concept instantiation of the respective concept described in the OWL DL constraint ontology. Then we added the reification of the class hierarchy. This is done with the following steps:

1. Create an individual i_{C} for every class C of the original ontology;
2. For each tag, declare the individual that relates to the class from the original ontology to belong to the class corresponding to that tag within the constraint ontology. For example, if a class C is

tagged +R, take the related individual i_C and add the fact RIGIDCLASS(i_C). If a class D is tagged −U, take the individual i_D and add the fact NONUNITYCLASS(i_D), etc.;

3. For each axiom of the form $C \sqsubseteq D$ add a property instance subClassOf(i_D, i_C).

Diagnosis

The thus created ontology can be simply checked for satisfiability by an OWL DL reasoner (actually, even much weaker reasoners would be sufficient due to the limited language fragment used in the constraint ontology). Standard reasoning services and ontology debugging tools can be applied in order to discover and repair inconsistencies.

For our experiments, we used RaDON,[10] a tool for inconsistency diagnosis based on KAON2.[11] RaDON features a number of algorithms for checking consistency and coherence of both TBox and ABox – among them an algorithm for identifying a set of minimal inconsistent subontologies for any given ontology. The algorithm starts with the inconsistent ontology, and (i) tries to find *any* minimal inconsistent subontology (Haase et al., 2005). Then, (ii) it removes *any* axiom from this minimal inconsistent subontology – which fixes (at least) one inconsistency in this part of the ontology. Finally, (iii) the algorithm starts all over again beginning with step (i) until the whole ontology is consistent. Obviously, this algorithm is non-deterministic, but it gives us a good approximation of the total number of inconsistencies in the ontology.

Example

Take the classes APPLE and FOOD. APPLE is tagged +R, whereas FOOD is tagged ∼R (as described in Section 2.3). Now for the sake of the example let us assume that APPLE is defined as a subclass of FOOD. We reify this axiom as described above, which results in the following formalization:

ANTIRIGIDCLASS(*food*)	(a)
RIGIDCLASS(*apple*)	(b)
subClassOf(*apple*, *food*)	(c)
Together with axiom (23) from the constraint ontology (cf. Table 8)	
$\text{RIGIDCLASS} \equiv \forall \text{subClassOf}.\neg\text{ANTIRIGIDCLASS}$	(23)

This leads to an unsatisfiability: *apple* is a RIGIDCLASS (b), which has a local range axiom for the subClassOf relation (23) so that from the instantiated relation (c) we must infer that *food* belongs to the ¬ANTIRIGIDCLASS class description – which is a clear contradiction to the given fact ANTIRIGIDCLASS(*food*) (a).

5.2. Analysis and examples

Table 9 shows the number of inconsistencies (each of them corresponding to a constraint violation) which were detected by RaDON for the sets of taggings manually created by A_1, A_2 and A_3. On average, 17 constraint violations were found per annotator – most of them related to rigidity and identity. This also holds for the agreements, i.e. the data sets consisting of those taggings where two or three annotators agreed upon the same meta-property tagging for each concept. As shown by the lower part of the table the overall number of inconsistencies drops significantly for the intersection of any two human taggings

[10]http://radon.ontoware.org/.
[11]http://kaon2.semanticweb.org/.

Table 9
Constraint violations for manual taggings

	Inconsistencies	Constraint violations			
		R	U	D	I
A_1	24	20	2	1	1
A_2	14	7	0	1	6
A_3	13	3	0	0	10
avg	17.0	10.0	0.7	0.7	5.7
A_1/A_2	5	1	1	1	2
A_1/A_3	2	1	0	0	1
A_2/A_3	2	1	0	0	1
avg	3.0	0.3	0.3	1.0	1.3
$A_1/A_2/A_3$	2	0	0	0	2

Table 10
Constraint violations for automatic taggings

	Inconsistencies	Constraint violations			
		R	U	D	I
A_1	74	23	43	1	7
A_2	17	3	8	5	1
A_3	31	1	0	0	30
avg	40.7	9.0	17.0	2.0	12.7
A_1/A_2	13	1	6	1	5
A_1/A_3	5	2	0	0	3
A_2/A_3	7	3	0	0	4
avg	8.3	2.0	2.0	0.3	4.0
$A_1/A_2/A_3$	3	0	0	0	3

to an average of 3.0 constraint violations per data set.[12] This can be explained by the fact that the number of agreed taggings is much lower than the overall number of tagged concepts (cf. Tables 1 and 2).

How does this compare to the automatically generated taggings? After training a classifier on each of the data sets we obtained seven fully automatic taggings. Since AEON so far has not been trained to distinguish between an anti-rigid and non-rigid (respectively, anti-unity and non-unity) tagging we converted all taggings to their stricter counterpart wherever possible, in order to obtain an upper bound for the number of constraint violations. The results of the inconsistency diagnosis computed for these taggings are presented by Table 10. As expected, the average number of constraint violations per data set increased significantly from 17 to 40.7. The automatic unity taggings seems to cause far more inconsistencies than it is the case for any of the manual taggings – probably, due to the fact that anti-unity tags are very rare in the manually created tagging sets.

[12]The agreement statistics represent lower bounds as they are computed in a cautious manner with respect to the number of possible inconsistencies. If at least one of the individual annotators tagged the regarding concept as `-R` whereas the others agreed upon `~R`, we assumed the agreement to be `-R` (or `-U`, respectively).

In the following we present some illustrative examples for inconsistencies which were detected in the ontology, and discuss possible reasons for the corresponding constraint violations.

Discussion
We expected two different kinds of errors when analysing the unsatisfiabilities: (i) the tagging was incorrect, e.g., because the annotators interpreted a concept in a different way than its author (presumably) did, or (ii) the subsumption relation was wrong, i.e. contradictory to a plausible meta-property assignment. We assumed that the OntoClean constraints – in our reification they are represented by the TBox axioms in Table 8 – are correct.

An example of an error of the first kind is given by the following facts:

ANTIRIGIDCLASS(*company*) (a)
RIGIDCLASS(*mediaCompany*) (b)
subClassOf(*mediaCompany*, *company*) (c)
The facts contradict axiom (23) from the constraint ontology:
$\text{RIGIDCLASS} \equiv \forall \text{subClassOf}.\neg \text{ANTIRIGIDCLASS}$ (23)

This error uncovers the improper tagging given by the taggers: COMPANY and MEDIACOMPANY should be tagged in a compatible way. As of now, COMPANY is said to be not rigid for all of its instances, whereas MEDIACOMPANY is rigid for its instances. Granted that the subsumption of MEDIACOMPANY by COMPANY is correct, the error must be with the tagging of the two concepts. But here the taggers seem to have two different notions of companies in mind when they tagged COMPANY and MEDIACOMPANY. If COMPANY is understood as an anti-rigid concept, then being a company is the role an organization can have: a university, which is an educational organisation, can become a company; or a company can turn to become a not-for-profit charity organisation. In this case, the concept company means an organisation that is meant to generate profit. On the other hand, if COMPANY is tagged as a rigid concept actually is a type for individuals: now, a company can not cease to be a company any more, but a change as described above would basically require to generate a new individual. It depends heavily on the conceptualisation which of the two company concepts are useful within a given ontology. The example given above shows a confusion with regards to the concepts, and thus a possible source for errors.

An error of the second kind was discovered in the subsumption relation between GROUP and POLITICALPARTY, which is only an indirect relation: GROUP is, in PROTON, actually a superclass of ORGANIZATION which in turn is a superclass of POLITICALENTITY which is a superclass of POLITICALPARTY. The problem here is indeed in the subsumption relation between GROUP and ORGANIZATION: a group is defined in PROTON as "*a group of agents, which is not organized in any way*" (Terziev et al., 2004). This description is not applicable for an organization (since an organization is, by its very nature and as shown in its name, organized). In formal terms, group was tagged as ∼U, whereas organization (and also political party) were tagged as +U, which causes an inconsistency (based on axioms 1 and 24 of the constraints meta-ontology). The ontology would need to be changed to reflect that (such a change is discussed in (Guarino & Welty, 2004), where, incidentally, this very same pair, organization and group, is taken as an example).

Another example of a subsumption relation that required closer inspection was the subsumption of HOTEL by BUILDING. Is HOTEL rather the role or the function of a building, whereas the building describes the object itself? Then the building would have a height, whereas the hotel would have a manager. A good example to illustrate the difference would be the number of rooms: it is counted differently for hotels than for buildings. This illustrates that it is not obvious if HOTEL should be subsumed by BUILDING or not, as the constraint violation suggested.

All the given examples are taken from the automatically tagged ontology. They illustrate that AEON points to possible taxonomic errors in an ontology, and guides the ontology engineer towards problematic parts of the ontology.

6. Related work

Applying OntoClean for ontology evaluation has been proposed, e.g., for traditional ontology engineering methodologies such as Fernández-López et al. (1999) and Sure et al. (2002). Checking for the described constraint violations after tagging reveals design errors during the cyclic engineering of ontologies. There are several OntoClean plug-ins created for ontology engineering suites to support this, in particular for Protégé (Noy et al., 2000), WebODE (Arpírez et al., 2001) and OntoEdit (Sure et al., 2003). They allow the manual tagging of ontologies, integrated within the ontology engineering task, and also partially check the consistencies according to the OntoClean rules described in Section 2. As we have seen in Section 4.1, the biggest problem when applying OntoClean is not the proper user interface for a manual tagging nor the possibility to check the ontology for formal taxonomic constraints, but rather the high cost of tagging itself. This is where the work presented here comes into play. To the best of our knowledge no other approach is known which automatizes the OntoClean tagging task as we do.

The constraint checking described here is based on ideas presented in (Welty, 2006). There an OWL ontology is presented that captures the taxonomic restrictions given by OntoClean in OWL DL, and also shows how off the shelf tools can be used to clean an ontology. We reuse the ontology offered there in our experiments. The paper further investigates the source of the problem with the low agreement between taggers that we encountered in Section 4.1.

Recently a refinement of OntoClean was suggested (Welty & Andersen, 2005). There the meta-property of Rigidity is further investigated under temporal and modal aspects. The framework we present here is extensible with regards to further or refined meta-properties, but finding appropriate patterns will remain a challenge. New patterns need to be evaluated experimentally.

Among the related work there is also a vast amount of research on text-mining using lexico-syntactic patterns. Most of them are based on the assumption that certain natural language expressions can provide evidence for lexical relationships such as hyponymy or meronymy (Hearst, 1992; Charniak & Berland, 1999). Given that these can be mapped more or less directly to particular ontological relationships pattern-based approaches have been frequently applied within the area of ontology learning, i.e. the automatic generation of ontologies from natural language text. Whereas some of them use locally available text corpora as the most important resource for ontology extraction others have specifically been developed for Google-based pattern matching (Cimiano et al., 2005). Obviously, the latter are most similar to our approach, being different only in the ultimate goals of the learning process.

7. Conclusion and outlook

Despite the fact that ontology evaluation is a critical task for ontology engineering there currently exist only few approaches. OntoClean is the most well-known approach that takes into account the intension of the concepts when checking the taxonomic structure of the ontology. First, applying OntoClean per se helps ontology engineers to better understand an ontology. Second, applying OntoClean allows for an evaluation of the formal properties of an ontology to detect potential misconceptualisations. Our approach focusses on facilitating the latter.

Tagging ontological concepts according to OntoClean is very expensive as it requires a lot of experts time and knowledge. The approach provided in this paper is giving a helpful hand by enabling an automatic tagging. Instead of claiming full automatic tagging and evaluation against OntoClean's meta-properties, we only take into account the concepts we are pretty sure of in our tagging and point to potential formal errors in the taxonomy at hand. But, such a tagging is only the beginning and can be considered a crucial building block for the next generation of integrated ontology engineering environments such as developed by the NeOn project.[13] While the user of such a system is creating or evolving an ontology, the system checks automatically the taxonomical relationships in the background, pointing to possible inconsistencies and likely errors. For those taggings where the system's confidence is high enough, suggestions will be given to the user. These suggestions can be substantiated with an explanation based on the patterns found on the Web, which is much more intuitive for most users than the formal definition of a meta-property.

The flexible architecture described in Section 3 can easily be extended to check for further constraints, not represented by OntoClean's rules. For example, if we find evidence that HUMAN BEING consists of AMOUNT OF MATTER then we could conclude that there is probably no taxonomic relationship between both concepts. Mereological relationships may be regarded as well. Due to the strong usage of Google™ and its snippets, we are even able to pinpoint to the very evidence of why two relationships should or should not exist. This way the automatic tagger can act as a kind of agent, who does not just point to errors, but also explains why a certain change is needed.

With the availability of the methods and the software presented in this paper we hope to turn the usage of OntoClean from a few experts' method to a widespread and standard technique for the intensional ontological analysis for large numbers of ontologies, raising the quality of ontologies in common use.

Acknowledgements

Research reported in this paper has been partially financed by the EU in the IST-2003-506826 project SEKT (http://www.sekt-project.com) and the IST-2005-027595) project NeOn (http://www.neon-project.com). Special thanks to Aldo Gangemi and Daniel Oberle for their time spent on tagging our reference ontology and thus making the evaluation possible. We also thank our students, Hua Gao and Qiu Ji, who did parts of implementation work. And finally, many thanks to Philipp Cimiano, Peter Haase, Stephan Bloehdorn and Christoph Tempich for helpful comments and interesting discussions.

References

Arpírez, J.C. et al. (2001). WebODE: a scalable workbench for ontological engineering. In *Proceedings of Int. Conference on Knowledge Capture (K-CAP)*, Victoria, Canada.

Berners-Lee, T., Hendler, J. & Lassila, O. (2001). The semantic web – a new form of web content that is meaningful to computers will unleash a revolution of new possibilities. *Scientific American*, *284*(5), 28–37.

Bontas, E.P., Tempich, C. & Sure, Y. (2006). ONTOCOM: A cost estimation model for ontology engineering. In I. Cruz et al. (eds), *Proceedings of the 5th International Semantic Web Conference (ISWC 2006)*, Lecture Notes in Computer Science (LNCS) (Vol. 4273, pp. 625–639). Berlin–Heidelberg: Springer-Verlag.

[13]http://www.neon-project.org/.

Charniak, E. & Berland, M. (1999). Finding parts in very large corpora. In *Proceedings of the 37th Annual Meeting of the ACL*, College Park, MD, USA (pp. 57–64).

Cimiano, P., Handschuh, S. & Staab, S. (2004). Towards the self-annotating web. In *Proceedings of the 13th World Wide Web Conference*, New York, NY, USA (pp. 462–471).

Cimiano, P., Ladwig, G. & Staab, S. (2005). Gimme the context: Context-driven automatic semantic annotation with C-PANKOW. In A. Ellis & T. Hagino (eds), *Proceedings of the 14th World Wide Web Conference* Chiba, Japan (pp. 332–341). New York: ACM Press.

Etzioni, O. et al. (2004). Web-scale information extraction in KnowItAll (preliminary results). In *Proceedings of the 13th WWW Conference*, New York, NY, USA (pp. 100–109).

Fernández-López, M. & Gómez-Pérez, A. (2002). The integration of OntoClean in WebODE. In *Proceedings of the EON2002 Workshop at 13th EKAW*, Siguenza, Spain.

Fernández-López, M., Gómez-Pérez, A., Sierra, J.P. & Sierra, A.P. (1999). Building a chemical ontology using methontology and the ontology design environment. *Intelligent Systems*, *14*(1), 37–46.

Gangemi, A. et al. (2003). Sweetening WordNet with Dolce. *AI Magazine*, *24*(3), 13–24.

Gómez-Pérez, A. (2004). *Ontology Evaluation*. In *Handbook on Ontologies in Information Systems*. Berlin: Springer.

Grefenstette, G. (1999). The WWW as a resource for example-based MT tasks. In *Proceedings of ASLIB'99 Translating and the Computer 21*, London, UK.

Guarino, N. & Welty, C.A. (2000). A formal ontology of properties. In *Knowledge Acquisition, Modeling and Management*, Juan-les-Pins, France (pp. 97–112).

Guarino, N. & Welty, C.A. (2004). An overview of OntoClean. In *Handbook on Ontologies in Information Systems* (pp. 151–172). Berlin: Springer.

Haase, P., van Harmelen, F., Huang, Z., Stuckenschmidt, H. & Sure, Y. (2005). A framework for handling inconsistency in changing ontologies. In Y. Gil, E. Motta, V.R. Benjamins & M.A. Musen (eds), *Proceedings of the Fourth International Semantic Web Conference (ISWC2005)*, LNCS (Vol. 3729, pp. 353–367). Berlin: Springer.

Hahn, U. & Schnattinger, K. (1998). Towards text knowledge engineering. In *Proceedings of AAAI'98/IAAI'98*, Madison, WI, USA.

Hearst, M. (1992). Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th International Conference on Computational Linguistics*, Nantes, France (pp. 539–545).

Keller, F., Lapata, M. & Ourioupina, O. (2002). Using the web to overcome data sparseness. In *Proceedings of EMNLP-02*, Pennsylvania, PA, USA (pp. 230–237).

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence* (pp. 1137–1145). San Mateo, CA: Morgan Kaufmann.

Noy, N., Fergerson, R. & Musen, M. (2000). The knowledge model of Protégé-2000: Combining interoperability and flexibility. In R. Dieng & O. Corby (eds), *Proceedings of the 12th EKAW*, LNAI, Juan-les-Pins, France (pp. 17–32). London: Springer.

Noy, N. & McGuinness, D.L. (2001). Ontology development 101: A guide to creating your first ontology. Technical Report KSL-01-05 and SMI-2001-0880, Stanford Knowledge Systems Laboratory and Stanford Medical Informatics.

Resnik, P. & Smith, N.A. (2003). The Web as a parallel corpus. *Computational Linguistics*, *29*(3), 349–380.

Staab, S. & R. Studer, eds (2004). *Handbook on Ontologies in Information Systems*. Berlin: Springer.

Sure, Y. & Studer, R. (2002). *On-To-Knowledge Methodology* (Chapter 3, pp. 33–46). England: Wiley and Sons.

Sure, Y., Angele, J. & Staab, S. (2003). OntoEdit: Multifaceted inferencing for ontology engineering. *Journal on Data Semantics*, *1*, 128–152.

Sure, Y., Staab, S. & Studer, R. (2002). Methodology for development and employment of ontology based knowledge management applications. *SIGMOD Rec.*, *31*(4), 18–23.

Särndal, C.-E., Swensson, B. & Wretman, J. (2003). Model assisted survey sampling (Springer series in statistics).

Tempich, C. et al. (2005). An argumentation ontology for distributed, loosely-controlled and evolving engineering processes of ontologies (DILIGENT). In C. Bussler et al. (eds), *ESWC 2005*, Heraklion, Crete, Greece, LNCS. Berlin: Springer.

Terziev, I., Kiryakov, A. & Manov, D. (2004). Base upper-level ontology (bulo) guidance. SEKT deliverable 1.8.1, Ontotext Lab, Sirma AI EAD (Ltd.).

Welty, C. (2006). OntOWLClean: Cleaning OWL ontologies with OWL. In B. Bennet & C. Fellbaum (eds), *Proceedings of Formal Ontologies in Information Systems FOIS 2006*, Baltimore, MD. Amsterdam: IOS Press.

Welty, C. & Andersen, W. (2005). Towards OntoClean 2.0: a framework for rigidity. *Journal of Applied Ontology*, *1*(1), 107–116.

Welty, C., Mahindru, R. & Chu-Carroll, J. (2004). Evaluating ontology cleaning. In D. McGuinness & G. Ferguson (eds), *AAAI2004*. San Jose, CA: AAAI/MIT Press.