Automatic Evaluation of Ontologies (AEON)

Johanna Völker, Denny Vrandečić, and York Sure

Institute AIFB, University of Karlsruhe
{voelker, vrandecic, sure}@aifb.uni-karlsruhe.de

Abstract. OntoClean is a unique approach towards the formal evaluation of ontologies, as it analyses the intensional content of concepts. Although it is well documented in numerous publications, and its importance is widely acknowledged, it is still used rather infrequently due to the high costs for applying OntoClean, especially on tagging concepts with the correct meta-properties. In order to facilitate the use of OntoClean and to enable proper evaluation of it in real-world cases, we provide AEON, a tool which automatically tags concepts with appropriate OntoClean meta-properties. The implementation can be easily expanded to check the concepts for other abstract meta-properties, thus providing for the first time tool support in order to enable intensional ontology evaluation for concepts. Our main idea is using the web as an embodiment of objective world knowledge, where we search for patterns indicating concepts meta-properties. We get an automatic tagging of the ontology, thus reducing costs tremendously. Moreover, AEON lowers the risk of having subjective taggings. As part of the evaluation we report our experiences from creating a middle-sized OntoClean-tagged reference ontology.

1 Introduction

Providing a shared conceptualization of a domain of interest, ontologies have become an important means for knowledge interchange and integration. The raise 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. Their wide-spread use leads to an increasing need for domain-independent methodologies and guidelines for ontology engineering and evaluation.

OntoClean [10] is a well-known methodology for the formal analysis of taxonomic relationships based on philosophical notions such as *essence*, *unity* or *identity*. Several tools supporting the OntoClean methodology have been developed and integrated into ontology editors such as ODEClean for WebODE [7], OntoEdit [17] or Protégé [15]. Given a taxonomy of concepts annotated with respect to a set of meta-properties, all these tools are able to perform an automatic analysis of the taxonomic relationships in order to detect cases of invalid generalization. Nevertheless, since this annotation has to be done manually, the evaluation of ontologies according to the OntoClean methodology remains a difficult and time consuming, thus very expensive task.

Y. Gil et al. (Eds.): ISWC 2005, LNCS 3729, pp. 716–731, 2005.

In order to solve this problem, we have developed an approach for the automatic tagging of concepts with respect to the meta-properties which constitute the basis for the OntoClean methodology. We provide an implementation of our approach, AEON[1], which makes use of the World Wide Web as the currently biggest existing source of common sense knowledge. In line with several approaches such as [4] and [5] we defined a set of domain independent patterns which can be considered as indicators for or against Rigidity, Identity, Unity and Dependence of given concepts in an ontology.

In the next section we give a brief introduction to the OntoClean methodology, in particular to the core notions of Rigidity, Unity, Dependence and Identity. Thereafter, we describe our approach to an automatic annotation of concepts with respect to these meta-properties (section 3). For the evaluation we needed a tagged ontology. In Section 4 we describe its creation and the problems we faced. Section 5 presents the evaluation setting and the results of the evaluation. In section 6 we discuss some related work, before we finally conclude with a short summary and an outlook to future work (section 7).

2 OntoClean in Theory

We provide a brief introduction to OntoClean, for a more thorough description refer to [10], for example. In the OntoClean vocabulary, *properties* are what is commonly called *concepts* or *classes*. *Meta-properties* are therefore properties of properties. Within this paper 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*. OntoClean consists of two steps: first every single concept needs to be tagged with occurrences of the core meta-properties, which are described below. Thus, every concept will have a certain tagging like *+R+U-D+I*. We call an ontology with tagged concepts a tagged ontology (wrt. OntoClean, to be precise). After the tagging, the second step of OntoClean is to check all subsumption relations of the ontology (also called Subclass-relations). OntoClean constraints 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.

We now briefly present the four main meta-properties and rules which belong to OntoClean. The four meta-properties are: *Rigidity (R)*, *Unity (U)*, *Dependence (D)* and *Identity (I)*. They base on philosophical notions 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* it 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

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

all of its instances. It is tagged $\sim R$. 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 is about "What is part of something and what is not?" This answer is given by an **Unity Criterion** (UC), which is true for all parts of an instance of this concept, and for nothing else. For example, there is an unitiy 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. Another way to regard dependency is to distinguish between intrinsic and extrinsic concepts. Intrinsic concepts are independent, whereas extrinsic concepts need to be given to an instance by circumstances or definitions.

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.

On a tagged ontology, we can use the existing OntoClean rules to check the ontology for consistency. Here, we will give only one illustrative example for these rules. For a full list refer to [11]. As shown in [17] such rules can be formalized as logical axioms and validated by an inference engine.

$\sim R$ can't 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 can not 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. The classic example is *student*, an anti-rigid concept, subsuming *human*, a rigid concept, which is obviously wrong: whereas every student is free to leave the university and stop being a student, humans cannot stop being humans. As every human would be a student, according to the example, they never could stop being a student, which contradicts the previous sentence.

3 Approach

Our approach for 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 [9], [14], [16], [3] and [4] 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

In order to evaluate our approach we developed AEON, a tool which 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 of AEON is roughly depicted by figure 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 1.1). Each of these patterns include 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

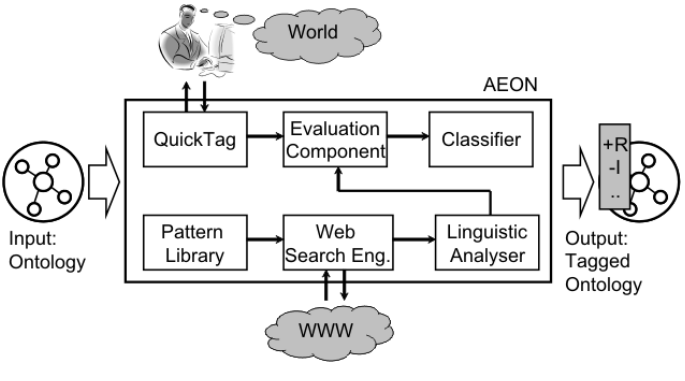

Fig. 1. Architecture of AEON

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).

Listing 1.1. Negative Evidence for Rigidity (R)

```
<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> %
```

Given a set of instantiated patterns (e.g. *"is no longer a student"*) 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 *"student"* for example.

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, for example, all those snippets returned by Google™, which satisfy the query "is no longer a computer" (cf. listing 1.1). 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 false matches in case of statements such as "He is no longer a computer hacker." or (this would yield false evidence for the unity of `employee`) when we find a phrase like "the computer of an employee consists of". Of course, linguistic filtering is also applied in the normalization process (see above).

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:

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

where Q_i is the set of queries associated with pattern i, q_c is the instantiation of query q for concept c, and $hits(q_c)$ and $hits(c)$ are 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 5). 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 and ontological relationships such as hyponymy and meronymy (cf. [13], [2], [12]). 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. So, we performed an extensive evaluation of many different pattern candidates 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.

`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 [11] identity is given by the fact that two instances of a concept are the same *iff* they have the same parts. This is known as *mereological extensionality* and can be expressed by the following patterns providing **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 the same patterns that we have already used as positive or negative evidence for Unity.

`(one|two)` *CONCEPT*

`amount of` *CONCEPT*

Dependence. Among the meta-properties Rigidity, Unity, Identity and Dependence 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. Due to the flexible architecture of AEON, adding further patterns is a very easy task. It simply requires the addition of the pattern in described format to the XML file.

We had some more patterns in mind, but preliminary testing in Google™ revealed often only a small number of hits, which would only lower the efficiency of the system and not improve the output of the system adequately.

3.3 Discussion

The described approach is original, and quite a number of problems were raised. We solved many of them, but some remain for further research. Both kinds 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.

The combination of negative and positive evidence right now just happens by simple subtraction. Maybe more complex combinations will yield even better results. This is an open issue. So is the difference between Non-, Anti- and Semi-Rigidity. 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 OntoClean in Practice

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 [11] and some examples in the other publications. This amounted to about 30-40 tagged concepts. [20] describes 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.

So we decided to tag an ontology on our own. We wanted a generic, domain-independent ontology with a not too small number of concepts. This is to ensure that the experience we gain and the classifiers trained will be most reusable for further ontologies evaluated with AEON in the future. We chose Proton[2], a freely available upper level ontology developed by OntoText within the European IST project SEKT[3]. We merged the System, Top and Upper modules of Proton, and the merged ontology contained 266 concepts, as diverse as *Accident*, *Alias*, *Happening* or *Woman*.

We asked methodology and ontology engineering experts to tag Proton 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 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. The third tagging we used for our evaluation was done by one of the authors of this paper.

The tagging 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). Concepts similar to those in the example ontology in [11] 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 often used ontology evaluation method, a far more precise and yet broader understandable description of OntoClean must become available, or else an approach for the automatic tagging of concepts must lower the time to

[2] `http://proton.semanticweb.org`
[3] `http://www.sekt-project.com`

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 [8], 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* since birth - or not. It is not a role a person plays, but rather it is the identity moving from body to body through the centuries. Simply tagging an ontology therefore reduces its possible audience by further ontological commitments.

We see that this contradicts to the definition of Rigidity, as there seem to be possible worlds where the concept is rigid and possible worlds in which it is not. 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.

5 Evaluation

As described in section 4 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 considered 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 labels 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 labelled A_1/A_2, A_1/A_3, A_2/A_3 were obtained by building the intersection of two of the single data sets. 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. *strict* means that two taggings were considered equal only if they were totally identical. *relaxed* means that $-$ and $\sim$ were

Table 1. Tagged Concepts

	R			U			I			D		
	+	-	∼	+	-	∼	+	-	∼	+	-	∼
A_1	147	69	50	156	81	29	194	61	11	151	110	3
A_2	208	39	0	103	138	3	189	58	0	31	203	13
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**	**53.7**	**3.7**	**81.7**	**104.7**	**5.3**
A_1 / A_2	122	3	20	77	61	11	134	17	4	23	94	3
A_1 / A_3	125	27	15	0	0	0	171	18	1	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**	**156.0**	**15.7**	**1.7**	**26.3**	**31.7**	**1.0**
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%	64.8%	71.7%	71.3%	71.1%	71.1%	54.5%	54.5%
D	48.9%	45.7%	75.0%	75.0%	15.0%	15.0%	15.0%	15.0%
avg	**58.8%**	**54.2%**	**69.9%**	**67.9%**	**52.4%**	**52.4%**	**38.6%**	**37.8%**

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.

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 how 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

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

Table 4. Rigidity (Best Results with Linguistic Filtering)

	P		R		F					Classifier
	+	-	+	-	+	-	M-avg	**baseline**	no LF	
A_1	59.0	51.4	69.5	40.0	63.8	45.0	54.4	49.9	61.6	RandomForest
A_2	86.9	31.8	91.0	23.3	88.9	26.9	57.9	43.4	47.8	ADTree
A_3	76.5	23.5	76.1	24.0	76.3	23.8	50.1	**46.4**	**44.8**	RandomTree
avg	**74.1**	**35.6**	**78.9**	**29.1**	**76.3**	**31.9**	**54.1**	**46.6**	**51.4**	
A_1 / A_2	91.3	64.3	94.9	50.0	93.1	56.3	74.7	43.4	69.4	ADTree
A_1 / A_3	78.2	66.7	98.0	12.9	87.0	21.6	54.3	46.7	62.6	DecisionStump
A_2 / A_3	93.8	11.1	93.8	11.1	93.8	11.1	52.5	39.3	48.2	RandomTree
avg	**87.8**	**47.4**	**95.6**	**24.7**	**91.3**	**29.7**	**60.5**	**43.1**	**60.1**	
A_1 / A_2 / A_3	95.5	0.0	100.0	0.0	97.7	0.0	48.9	38.7	48.4	NBTree

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 an *macro-average F-Measure* of $F = \frac{1}{2} * (F_+ + F_-)$, which 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.

Setting. Since we decided to evaluate our system separately for R, U, I and D, we made 2*7*4=56 experiments (one for each human annotator, each meta-property, with and without linguistic filtering) using a number of Weka[4] classifiers. In order to detect the limitations of our approach and to see what we can potentially get out of the data 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

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

Table 5. Identity (Best Results with Linguistic Filtering)

	P		R		F					Classifier
	+	-	+	-	+	-	M-avg	**baseline**	no LF	
A_1	75.0	34.8	84.1	23.2	79.3	27.8	53.6	47.2	49.5	ADTree
A_2	79.4	37.5	86.3	26.8	82.7	31.3	57.0	**46.2**	**45.0**	ADTree
A_3	87.3	55.0	95.8	26.8	91.4	36.1	63.8	43.4	47.9	RandomForest
avg	**80.6**	**42.4**	**88.7**	**25.6**	**84.5**	**31.7**	**58.1**	**45.6**	**47.5**	
A_1 / A_2	87.0	13.3	90.7	10.0	88.8	11.1	50.0	42.0	54.1	RandomTree
A_1 / A_3	93.0	46.7	95.2	36.8	94.1	41.2	67.7	40.5	50.8	NBTree
A_2 / A_3	95.7	57.1	98.1	36.4	96.9	44.4	70.7	38.6	48.2	ADTree
avg	**91.9**	**39.0**	**94.7**	**27.7**	**93.3**	**32.2**	**62.8**	**40.4**	**51.0**	
A_1 / A_2 / A_3	95.3	66.7	99.2	25.0	97.2	36.4	66.8	38.0	48.5	RandomForest

Table 6. Unity (Best Results with Linguistic Filtering)

	P		R		F					Classifier
	+	-	+	-	+	-	M-avg	**baseline**	no LF	
A_1	69.5	49.5	63.6	56.2	66.4	52.6	59.5	49.6	58.8	DecisionStump
A_2	43.0	61.2	46.0	58.3	44.4	59.7	52.1	47.7	57.8	ADTree
A_3	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
avg	**56.3**	**55.4**	**54.8**	**57.3**	**55.4**	**56.2**	**55.8**	**48.7**	**58.3**	
A_1 / A_2	57.6	53.6	51.5	59.7	54.4	56.5	55.5	50.0	60.2	ADTree
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	**57.6**	**53.6**	**51.5**	**59.7**	**54.4**	**56.5**	**55.5**	**50.0**	**60.2**	
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

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. Please 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 we get by using a pattern-based approach.

Results. One of the main findings of our experiments was that linguistic filtering really helps in the task of pattern-based ontology evaluation. As shown by tables 4, 5 and 7 without linguistic filtering the baseline for macro-average F-Measure was missed several times. And especially for *Identity* we noticed that the results could be improved by around 30% with the help of linguistic filtering. Another

interesting result of the evaluation was that on average our system performed significantly better on the agreement, i.e. the intersection of two or three data sets, than on the single 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.

Table 7. Dependence (Best Results with Linguistic Filtering)

	P		R		F					Classifier
	+	-	+	-	+	-	M-avg	**baseline**	no LF	
A_1	68.2	40.9	69.8	39.1	69.0	40.0	54.5	**49.7**	**39.1**	RandomTree
A_2	30.0	81.5	23.1	86.3	26.1	83.8	55.0	41.8	56.7	RandomForest
A_3	100.0	0.0	100.0	0.0	100.0	0.0	50.0	34.7	50.0	ADTree
avg	**66.1**	**40.8**	**64.3**	**41.8**	**65.0**	**41.3**	**53.2**	**42.1**	**48.6**	
A_1 / A_2	45.5	70.0	45.5	70.0	45.5	70.0	57.8	**44.7**	**35.3**	ADTree
A_1 / A_3	100.0	0.0	100.0	0.0	100.0	0.0	50.0	35.1	40.0	ADTree
A_2 / A_3	100.0	0.0	100.0	0.0	100.0	0.0	50.0	N/A	50.0	ADTree
avg	**81.8**	**23.3**	**81.8**	**23.3**	**81.8**	**23.3**	**52.6**	**39.9**	**41.8**	
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

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 without much parameter tuning we hope to get even better results in future experiments.

6 Related Work

Applying OntoClean for ontology evaluation has been proposed e.g. for traditional ontology engineering methodologies such as [6,18]. Checking for the described constraint violations after tagging reveals any 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 [15], WebODE [1] and OntoEdit [17]. 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, 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. DILIGENT [19] is the only known ontology engineering methodology right now, that explicitly integrates computational agents to be actors participating in ontology engineering tasks just like human users. The integration of our approach into DILIGENT is on our agenda.

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 only known approach, where the intension of the concepts are taken into account when checking the taxonomic structure of the ontology. 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 a small building block for a next generation integrated ontology engineering environment. While the user is creating or evolving an ontology, the system checks the taxonomical relationships in the background, pointing to possible inconsistencies and likely errors. For those taggings where the system's confidence is not that high, suggestions will be given. These suggestions can be substantiated with an explanation based on the patterns found on the Web, which is much more intuitive 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 full agent, who does not just point to errors, but also explains why a certain change is needed.

With the availability of 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 every ontology, 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). Special thanks go to Aldo Gangemi and Daniel Oberle for their time spent on tagging our reference ontology and thus making the evaluation possible. We thank Andreas Hotho, Philipp Cimiano, Peter Haase, Stephan Bloehdorn and Christoph Tempich for helpful comments and interesting discussions.

References

1. J. C. Arpírez et al. WebODE: a scalable workbench for ontological engineering. In *Proc. of Int. Conf. on Knowledge Capture (K-CAP), Victoria, Canada*, 2001.
2. E. Charniak and M. Berland. Finding parts in very large corpora. In *Proc. of the 37th Annual Meeting of the ACL*, pages 57–64, 1999.

3. P. Cimiano, S. Handschuh, and S. Staab. Towards the self-annotating web. In *Proceedings of the 13th World Wide Web Conference*, pages 462–471, 2004.
4. P. Cimiano, G. Ladwig, and S. Staab. Gimme' the context: Context-driven automatic semantic annotation with c-pankow. In *Proc. 14th WWW*. ACM, 2005.
5. O. Etzioni et al. Web-scale information extraction in KnowItAll (preliminary results). In *Proc. 13th WWW Conf.*, pages 100–109, 2004.
6. M. Fernández-López et al. Building a chemical ontology using Methontology and the Ontology Design Environment. *IEEE Int. Systems*, 14(1), Jan/Feb 1999.
7. M. Fernández-López and A. Gómez-Pérez. The integration of ontoclean in webode. In *Proc. of the EON2002 Workshop at 13th EKAW*, 2002.
8. A. Gangemi et al. Sweetening WordNet with Dolce. *AI Magazine*, Fall 2003.
9. G. Grefenstette. The WWW as a resource for example-based MT tasks. In *Proc. of ASLIB'99 Translating and the Computer 21*, 1999.
10. N. Guarino and C. A. Welty. A formal ontology of properties. In *Knowledge Acquisition, Modeling and Management*, pages 97–112, 2000.
11. N. Guarino and C. A. Welty. An overview of OntoClean. In S. Staab and R. Studer, editors, *Handbook on Ontologies in Inf. Sys.*, pages 151–172. Springer, 2004.
12. U. Hahn and K. Schnattinger. Towards text knowledge engineering. In *Proc. of AAAI'98/IAAI'98*, 1998.
13. M. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proc. 14th Int. Conf. on Computational Linguistics*, pages 539–545, 1992.
14. F. Keller, M. Lapata, and O. Ourioupina. Using the web to overcome data sparseness. In *Proc. of EMNLP-02*, pages 230–237, 2002.
15. N. Noy, R. Fergerson, and M. Musen. The knowledge model of Protégé-2000: Combining interoperability and flexibility. In R. Dieng and O. Corby, editors, *Proc. of the 12th EKAW*, LNAI, pages 17–32, Juan-les-Pins, France, 2000. Springer.
16. P. Resnik and N. A. Smith. The Web as a parallel corpus. *Computational Linguistics*, 29(3):349–380, 2003.
17. Y. Sure, J. Angele, and S. Staab. OntoEdit: Multifaceted inferencing for ontology engineering. *Journal on Data Semantics*, LNCS(2800):128–152, 2003.
18. Y. Sure, S. Staab, and R. Studer. Methodology for development and employment of ontology based knowledge management applications. *SIGMOD Rec.*, 31(4), 2002.
19. C. Tempich et al. An argumentation ontology for distributed, loosely-controlled and evolving engineering processes of ontologies (DILIGENT). In C. Bussler et al., editors, *ESWC 2005*, LNCS, Heraklion, Crete, Greece, 2005. Springer.
20. C. Welty, R. Mahindru, and J. Chu-Carroll. Evaluating ontology cleaning. In D. McGuinness and G. Ferguson, editors, *AAAI2004*. AAAI / MIT Press, 2004.