Learning Disjointness

Johanna Völker[1], Denny Vrandečić[1], York Sure[1] and Andreas Hotho[2]

[1]Institute AIFB, University of Karlsruhe, Germany
[2]University of Kassel, Germany
{voelker,vrandecic,sure}@aifb.uni-karlsruhe.de,
hotho@cs.uni-kassel.de

Abstract. An increasing number of applications benefits from light-weight ontologies, or to put it differently *"a little semantics goes a long way"*. However, our experience indicates that more expressiveness can offer significant advantages. Introducing disjointness axioms, for instance, greatly facilitates consistency checking and the automatic evaluation of ontologies. In an extensive user study we discovered that proper modeling of disjointness is a difficult and very time-consuming task. We therefore developed an approach to automatically enrich learned or manually engineered ontologies with disjointness axioms. This approach relies on several methods for obtaining syntactic and semantic evidence from different sources which we believe to provide a solid base for learning disjointness. After thoroughly evaluating the implementation of our approach we think that in future ontology engineering environments the automatic discovery of disjointness axioms may help to increase the richness, quality and usefulness of any given ontology.

1 Introduction

An increasing number of applications benefits from light-weight ontologies, or, to put it differently, *"a little semantics goes a long way"* (Jim Hendler). Our experience in building ontology-based systems indicates, however, that adding more expressivity in a controlled manner can reap further benefits. Introducing disjointness axioms, for example, greatly facilitates consistency checking and the automatic evaluation of individuals in a knowledge base with regards to a given ontology.

In description logics two classes are considered as disjoint *iff* their *taxonomic overlap*, i.e. the set of common individuals, *must* be empty. This does not include classes with actual extensions that coincidentally do not have common individuals, for instance *Woman* and *US President*, but only those where the common subset must be empty in all possible worlds – like, for example, *Woman* and *Car*.

Disjointness allows for far more expressive and meaningful ontologies, as shown exemplary in the following. An ontology language with the expressivity of RDFS does not constrain the possible assertions in any way. Even after we set up an ontology defining terms like *Book*, *Student* and *University*, stating that *John* is both a *Student* and a *University* is logically perfectly viable, and would not be recognized as an error by a reasoner. Only if we define these classes as being disjoint, the reasoner will be able to infer the error in the above ontology, guaranteeing that particular constraints are met by the knowledge base and a certain quality of facts is achieved – thus raising the quality of the whole ontology-based system [17].

Despite the obvious importance of stating disjointness among classes, many of today's ontologies do not contain any disjointness axioms. In fact, a survey of 1,275 ontologies [22] recently found only 97 of them to include disjointness axioms. We can only speculate about the reasons, but it is very likely that ontology engineers often forget to introduce disjointness axioms, simply because they are not aware of the fact that classes which are not explicitly declared to be disjoint will be considered as overlapping. Particularly, inexperienced users usually assume the semantics of partitions, or even complete partitions, when they build a subsumption hierarchy (see [15]). Also, as the size of an ontology is a major cost driver for ontologies [2], the manual engineering and addition of the axioms actually costs more time, and thus money.

Therefore, we believe that an approach to automatically introduce disjointness axioms into an ontology would be a valuable addition to any ontology learning or engineering framework. The principle feasibility of learning disjointness based on simple lexical evidence has already been shown by [9]. However, our experiments indicate that a single heuristic is not suitable for detecting disjointness with sufficiently high precision, i.e. better than an average human could do.

For this paper, we performed an extensive survey in order to collect experience with modeling disjoint classes, and identified several problems frequently encountered by users who try to introduce disjointness axioms. Based on the results of our survey we developed a variety of different methods in order to automatically extract lexical *and* logical features which we believe to provide a solid basis for learning disjointness. These methods take into account the structure of the ontology, associated textual resources, and other types of data sources in order to compute the likeliness of two classes to be disjoint. The features obtained from these methods are used to build an overall classification model which we evaluated against more than 10,000 disjointness axioms provided by 30 human annotators. Due to the encouraging evaluation results we are confident that our implementation can be used, for example, to extend state-of-the-art ontology learning systems, to support ontology debugging [17], or to evaluate manually added disjointness axioms.

The survey also showed that deciding if two classes are disjoint is far from trivial. Although experts have a higher agreement on disjointness than non-expert users, their agreement is still lower than we expected. Discussing these problematic formalizations, we uncovered a number of problems humans have with formal disjointness.

In this paper, we will, in Section 2, first present the features we have used in order to automatically learn disjointness axioms. Section 3 describes the set up and execution of the experiments we conducted in order to train a classifier and evaluate the results of our implementation (Section 4). We close with an overview of related work in Section 5 and a summary of the key contributions and remaining open questions in Section 6.

2 Features for Learning Disjointness

Assuming that there is not the one and only approach to determine the disjointness of two classes in an ontology, we developed a variety of different methods to obtain evidence for or against disjointness from different sources. The features delivered by these methods will help us to train a classifier which is able to distinguish between disjoint and non-disjoint classes.

Preliminaries: In this paper we adopt the OWL ontology model, although we do not restrict our approach to OWL. Any ontology model that allows to state disjointness between two classes can be used with all the methods described in this paper.

The methods are provided with an unsorted list of all the pairs previously tagged by human annotators. In the following the set of pairs will be denoted by $P = \{p_1, ...p_n\}$ for $0 \leq n \leq |C|^2$, where C is the set of all classes in the ontology. Each pair $p_k = (c_{k_1}, c_{k_2})$ consists of two classes $c_{k_1}, c_{k_2} \in C$ and $c_{k_1} \neq c_{k_2}$. The confidence of the system in c_{k_1} and c_{k_2} being (not) disjoint is denoted by *conf*$(p_k, +)$ or *conf*$(p_k, -)$ respectively.

All methods are allowed to look up these classes within their semantic context, i.e. the domain **ontology** they have been extracted from (see Section 3.1). And finally, as additional sources of background knowledge, the methods may make use of a **corpus** of textual resources associated with the ontology. We automatically selected a subset of 957 documents from the Reuters corpus[1] [16]. For efficiency reasons we only chose those documents with at least 20 occurrences of any of the classes in the ontology.

It is important to mention, that we assume 'meaningful' labels for all classes in the ontology, i.e. labels which may be understood by humans even without knowing the whole taxonomy. This assumption is particularly relevant for all methods which make use of textual resources such as the pattern-based disjointness extraction (cf. Section 2.4), the computation of extensional overlap with respect to Del.icio.us[2] and the algorithms for learning taxonomic relationships (see Section 2.1).

2.1 Taxonomic Overlap

In description logics two classes are disjoint *iff* their taxonomic overlap, i.e. the set of common individuals, *must* be empty. Because of the open world assumption in OWL, these individuals do not necessarily have to *exist* in the ontology. The taxonomic overlap of two classes is considered not empty as long as there *could* be common individuals within the domain of interest which is modeled by the ontology.

We developed three methods which determine the likeliness for two classes to be disjoint by considering their overlap with respect to (i) **individuals** and **subclasses** in the ontology – or learned from a corpus of associated textual resources – and (ii) **Del.icio.us documents** tagged with the corresponding class labels.

Ontology Both, individuals and subclasses can be imported from an ontology (see Section 3.1) or from a given corpus of text documents. In the latter case, `subclass-of` and `instance-of` relationships are extracted by different algorithms provided by the Text2Onto[3] ontology learning framework. A detailed description of these algorithms

[1] `http://trec.nist.gov/data/reuters/reuters.html`
[2] `http://del.icio.us/`
[3] `http://ontoware.org/projects/text2onto/`

can be found in [4]. All taxonomic relationships – learned and imported ones – are associated with rating annotations $r_{subclass-of}$ (or $r_{instance-of}$ respectively) indicating the certainty $x > 0$ of the underlying ontology learning framework in the correctness of its results. For imported relationships the confidence is 1.0.

$$r_{subclass-of}(c_1, c_2) = \begin{cases} x & c_1 \texttt{ subclass-of } c_2 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

The following formula defines the confidence $conf(p, -)$ for a pair $p = (c_1, c_2)$ to be **not** disjoint based on the taxonomic overlap of c_1 and c_2 with respect to common **subclasses** (the same for **instance**):

$$conf(p, -) = \frac{\sum_{c \in sub_1 \cap sub_2} (r_{subclass-of}(c, c_1) \cdot r_{subclass-of}(c, c_2))}{\sum_{c \in sub_1} r_{subclass-of}(c, c_1) + \sum_{c \in sub_2} r_{subclass-of}(c, c_2)} \tag{2}$$

where sub_i denotes the set of subclasses of c_i.

Del.icio.us Del.icio.us is a server-based system with a simple-to-use interface that allows users to organize and share bookmarks on the internet. It associates each URL with a description, a note, and a set of tags (i.e. arbitrary class labels). For our experiments, we collected $|U| = 75,242$ users, $|T| = 533,191$ tags and $|R| = 3,158,297$ resources, related by in total $|Y| = 17,362,212$ triples. The idea underlying the use del.icio.us in this case is that two labels which are frequently used to tag the same resource are likely to be disjoint, because users tend to avoid redundant labeling of documents.

$$conf(p, -) = \frac{|\{d | c_1 \in t(d), c_2 \in t(d)\}|}{\sum_{c \in C} |\{d | c_1 \in t(d), c \in t(d)\}| + \sum_{c \in C} |\{d | c_2 \in t(d), c \in t(d)\}|} \tag{3}$$

where $t(d)$ is the set of del.icio.us *tags* associated with document d. The normalized number of co-occurrences of c_1 and c_2 (their respective labels to be precise) as del.icio.us tags aims at capturing the degree of association between the two classes.

2.2 Subsumption

If one class is a subclass of the other we assume the two classes of a pair $p = (c_1, c_2)$ to be **not** disjoint with a confidence equal to the likeliness associated with the `subclass-of` relationship (cf. Section 2.1).

$$conf(p, -) = \max(r_{subclass-of}(c_1, c_2), r_{subclass-of}(c_2, c_1)) \tag{4}$$

2.3 Semantic Similarity

The assumption that a direct correspondence between the semantic similarity of two classes indicates their likeliness to be disjoint led to the development of three further methods: The first one implements the similarity measure described by [24] to compute the semantic similarity sim of two classes c_1 and c_2 with respect to **WordNet** [6]:

$$conf(p,-) = sim(s_1, s_2) = \frac{2 * depth(lcs(s_1, s_2))}{depth(s_1) + depth(s_2)} \tag{5}$$

where $s_i = first(c_i)$ denotes the first sense of c_i, $i \in \{1,2\}$ with respect to WordNet, and $lcs(s_1, s_2)$ is the least common subsumer of s_1 and s_2. The depth of a node n in WordNet is recursively defined as follows: $depth(root) = 1$, $depth(child(n)) = depth(n) + 1$.

The second method measures the distance of c_1 and c_2 with respect to the given background **ontology** (see Section 3.1) by computing the minimum length of a path q of `subclass-of` relationships connecting c_1 and c_2.

$$conf(p,+) = \min_{p \in paths(c_1,c_2)} length(q) \tag{6}$$

And finally, the third method computes the similarity of c_1 and c_2 based on their **lexical context**. Along with the ideas described in [5] we exploit Harris' distributional hypothesis [10] which claims that two words are semantically similar to the extent to which they share syntactic contexts.

For each occurrence of a class label in a corpus of textual documents (see preliminaries of this section) we consider all the lemmatized tokens in the same sentence (except for stop words) as potential features in the context vector of the corresponding class. After the context vectors for both classes have been constructed, we assign weights to all features using a modified version of the tf-idf formula:

Let $v_i = (f_1^i ... f_n^i)$ be the context vector of class c_i where each f_j^i, $n \geq 1$ is the frequency of token j in the context of c_i. Then we define $TF(f_j^i) = \sum_{d \in doc(c_i)} freq(f_j^i, d)$ and $N = |doc(c_i)|$ and $DF = |doc(c_i) \cap doc(f_j^i)|$, where $doc(t)$ is the set of documents containing term t and $freq(t,d)$ is the frequency of term t in document d. And finally, we get $TFIDF(f_j^i) = TF(f_j^i) \cdot \log(\frac{N}{DF})$.

Given the *weighted* context vectors v_1' and v_2' the confidence in c_1 and c_2 being **not** disjoint is defined as $conf(p,-) = \cos(v_1', v_2')$.

2.4 Patterns

Since we found that disjointness of two classes is often reflected by human language, we defined a number of lexico-syntactic patterns to obtain evidence for disjointness relationships from a given corpus of textual resources. The first type of pattern is based on enumerations as described in [9]. The underlying assumption is similar to the idea described in section 2.1, i.e. terms which are listed separately in an enumeration mostly denote disjoint classes. Therefore, from the sentence

The pigs, cows, horses, ducks, hens and dogs all assemble in the big barn, thinking that they are going to be told about a dream that Old Major had the previous night.

we would conclude that *pig*, *cow*, *horse*, *duck*, *hen* and *dog* are disjoint classes. This is because we believe that – except for some idiomatic expressions it would be rather unusual to enumerate overlapping classes such as *dogs* and *sheep dogs* separately which would result in semantic redundancy. More formally:

Given an enumeration of noun phrases $NP_1,\ NP_2,\ \ldots,\ (and|or)\ NP_n$ we conclude that the concepts $c_1,\ c_2,\ \ldots,\ c_k$ denoted by these noun phrases are pairwise disjoint, where the confidence for the disjointness of two concepts is obtained from the number of evidences found for their disjointness in relation to the total number of evidences for the disjointness of these concepts with other concepts.

The second type of pattern is designed to capture more explicit expressions of disjointness in natural language by phrases such as $either\ NP_1\ or\ NP_2$ or $neither\ NP_1\ nor\ NP_2$. For both types of patterns we compute the confidence for the disjointness of two classes c_1 and c_2 as follows:

$$conf(p,+) = \frac{freq(c_1,c_2)}{\sum_{j\neq 1} freq(c_1,c_j) + \sum_{i\neq 2} freq(c_i,c_2)} \tag{7}$$

where $freq(c_i,c_j)$ is the number of patterns providing evidence for the disjointness of c_i and c_j with $0 \leq i,j \leq |C|^2$ and $i \neq j$.

2.5 OntoClean

In [20] we introduced AEON, an approach to automatically evaluate ontologies according to the OntoClean methodology [8]. The basic idea is to use a pattern-based approach on top of the Web (and other textual data sources) for annotating classes of a given ontology with the OntoClean properties such as unity, identity and rigidity. Parts of the approach can be reused for learning disjointness axioms.

Two classes are disjoint if they have incompatible unity or identity criteria. This implies that a class carrying anti-unity ($\sim$U) must be disjoint of a class carrying unity (+U) – and similarly for identity. Since we use the same subset of the PROTON ontology as in our AEON experiments, we can rely on the manual OntoClean taggings we collected earlier for the evaluation of AEON.

$$conf(p,+) = \begin{cases} 1 & \text{if } c_1 \text{ tagged with } \phi\Omega,\ c_2 \text{ tagged with } \psi\Omega, \\ & \text{for } \Omega \in \{U,I\},\ \phi,\psi \in \{\sim,+\},\ \phi \neq \psi \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

2.6 Meta Algorithm

The meta algorithm considers superclasses known to be disjoint (from previously computed confidence values) and propagates this information downwards in the taxonomic hierarchy[4]. For $p = (c_1, c_2)$ the confidence for c_1 and c_2 being disjoint is computed as follows:

[4] This algorithm was not used in the final evaluation, since early experiments indicated that it introduces too much noise. However, we report on it for reasons of completeness. And we still believe that it constitutes a potentially interesting direction of future work, because it allows for integrating subsumption information into any other algorithm.

$$conf(p,+) = \frac{\sum_{p^s}(conf(p^s,+) - conf(p^s,-))}{|super(c_1)| \cdot |super(c_2)|} \tag{9}$$

where $p^s = (c_1^s, c_2^s)$ with $c_i^s \in \{c|subclass - of(c_i, c)\}$ for $i \in \{1,2\}$ and $subclass - of(c_i, c_j)$ being the `subclass-of` relationship between c_i and c_j. Moreover, $super(c)$ denotes the set of superclasses of c.

3 Experiment: Human Annotation of Disjointness

We thoroughly evaluated our approach by performing a comparison of learned disjointness axioms against a large number of manually created ones to calculate (among other things) the degree of overlap. This section describes the generation of the evaluation dataset consisting of 2000 pairs of classes tagged by 30 annotators and discusses methodological aspects related to the manual creation of disjointness axioms. The complete dataset is available from `http://www.aifb.uni-karlsruhe.de/WBS/jvo/data/disjointness-111206.zip`.

3.1 Ontology

As a basis for the creation of the evaluation datasets and as background knowledge for the ontology learning algorithms we took a subset (*system*, *top* and *upper* module) of the freely available PROTON ontology (PROTo ONtology)[5]. In total our subset of PROTON contains 266 classes, 77 object properties, 34 datatype properties and 1388 siblings.

PROTON is a basic upper-level ontology to facilitate the use of background or preexisting knowledge for automatic metadata generation. PROTON covers the general concepts necessary for a wide range of tasks, including semantic annotation, indexing, and retrieval of documents. The design principles can be summarized as follows (as described in [19]) (i) domain-independence; (ii) light-weight logical definitions; (iii) alignment with popular standards; (iv) good coverage of named entities and concrete domains (i.e. people, organizations, locations, numbers, dates, addresses).

3.2 Evaluation Setting: Manual Taggings

To be able to compare the results of our trained model with the results generated by manual annotation we created a dataset consisting of 2000 pairs of classes as follows: First, we manually selected 200 (potentially) *non-disjoint* pairs from the ontology, since we assumed the set of non-disjoint pairs to constitute a weak minority class (which would have hampered the construction of a good model for our classifier). Then, we randomly chose 500 *siblings* – which constitute a subset of the data, which is of particular interest from a practical and theoretical aspect. And finally, we added another 1300 pairs chosen *randomly* without any selection criteria.

[5] PROTON is available from `http://proton.semanticweb.org`.

Once the dataset was complete, each pair was randomly assigned to 6 different people – 3 from each of two groups, the first one consisting of PhD students from our institute (all of them professional ”ontologists”), the second being composed of under-graduate students without profound knowledge in ontological engineering. Each of the annotators was given between 385 and 406 pairs along with natural language descriptions of the classes whenever those were available. Possible taggings for each pair were $+$ (disjoint), $-$ (not disjoint) and $?$ (unknown). The result were two datasets A and B for ”ontologists” and ”students”. A third dataset C was created by merging A and B (cf. table 1a). Dataset D is a subset of C consisting of all siblings, whereas E contains all those pairs of classes which were randomly selected.

In order to get cleaner and less ambiguous training data for our classification model (see Section 4) we computed the *majority votes* for all the above mentioned datasets by considering the individual taggings for each pair (3 in the case of A and B, and 6 for C). If at least 50% (or 100% respectively) of the human annotators agreed upon $+$ or $-$ this decision was assumed to be the majority vote for that particular pair. In case of equally many positive and negative taggings, the majority vote was defined as $?$ or *unknown*. These pairs were not used for training purposes. In this way we reduced the noise the classifier had to deal with in the training phase, and obtained a better overall model. Some statistical properties of the majority vote datasets are given by table 2.

3.3 Analysis of Human Annotations

In order to determine how difficult it is for humans to tag pairs of classes as being disjoint or not we measured the human agreement within and across the different subsets of the data. Table 1b shows the average agreement among the individual taggers, i.e. the average maximum ratio of annotators who agreed upon the same tag for a pair of classes. By analysing the figures we find that the average agreement for D is significantly lower than the agreement for any of the other datasets – which seems to imply that pairs of siblings (classes with a common direct superclass) are much more difficult to tag for human annotators than randomly chosen pairs of classes. This might be due to the fact that it is comparably hard to determine the differences between the intension and extension of classes which are semantically very close.

Table 1. a) Evaluation Datasets

ID	Dataset	Annotators	Tags per Pair	Pairs
A	Experts	15	3	2000
B	Students	15	3	2000
C	All	30	6	2000
D	Siblings	30	6	541
E	Random	30	6	1300

b) Tagged Pairs (Individual)

Dataset	Individual Taggings					
	$+$	$-$	$?$	all	$-/+$	avg. agree.
A	3849	2007	144	6000	0.521	0.869
B	3881	2106	13	6000	0.543	0.858
$avg.$	3865.0	2056.5	78.5	6000	0.532	0.864
C	7730	4113	157	12000	0.532	0.824
D	1362	1822	62	3246	1.338	0.754
E	6166	1554	80	7800	0.252	0.853

In addition to the computation of the agreement within each of the datasets, we also tried to capture commonalities and differences between the taggings of people from the two groups of annotators – ontologists (A) and students (B).

First, we measured the average agreement of the individual taggings of the experts with the majority vote 100% of the students and vice versa. The figures – 0.852 for the agreement between A and the majority vote of B, and a slightly lower value of 0.834 for the agreement between B and the majority vote of A – indicate that, maybe due to the relatively higher disagreement among the students (see table 1b), those tend to agree mainly on very evident cases of disjointness.

The hypothesis that there is a considerable number of pairs which are comparably easy to tag, thus provoking a high agreement, is supported by the figures we get for the agreement among the majority votes 100% (0.964) and 50% (0.793) of A and B.

And finally, we completed our analysis of the annotation results by inspecting concrete examples of differently tagged pairs. Table 3 shows the listing of all pairs of classes which were assigned different tags by the majority votes 100% (which means that all 3 annotators of A or B agreed upon each tag) of experts and students. An extensive discussion of the differences which tries to explain some of the problems the human annotators encountered can be found in the following section.

3.4 Discussion

During the creation of the human annotations, we had the chance to study the problems humans face when using disjointness. Even in the taggings of the experts group – consisting of post-graduates all involved in Semantic Web research – the overlap of the taggings was lower than expected (cf. Section 3.3). Table 3 shows all pairs where all experts agreed on one tagging, and all students agreed on the other. Based on an analysis of the taggings and subsequent discussions with the taggers, we identified several types of problems regarding disjointness:

1. The label and comment of a class often do not provide an unambiguous idea of what is meant with this class.

Table 2. Tagged Pairs (Majority Vote)

Dataset	Majority Vote 50%					Majority Vote 100%				
	$+$	$-$	?	all	$-/+$	$+$	$-$	?	all	$-/+$
A	1297	649	54	2000	0.500	931	330	739	2000	0.354
B	1346	648	6	2000	0.481	846	307	847	2000	0.363
$avg.$	1321.5	648.5	30.0	2000	0.490	888.5	318.5	793.0	2000	0.359
C	1276	537	187	2000	0.421	616	194	1190	2000	0.315
D	188	274	79	541	1.457	28	96	417	541	3.429
E	1072	140	88	1300	0.131	588	35	677	1300	0.060

Table 3. Differences between Majority Votes 100% of A (Experts) and B (Students)

Vote		Disjoint Classes ?		Vote		Disjoint Classes ?	
A	B			A	B		
−	+	RailroadFacility	Pipeline	+	−	Canal	Harbor
−	+	Order	Abstract	+	−	OfficialPoliticalMeeting	Parliament
−	+	Newspaper	HomePage	+	−	Week	Month
−	+	School	MineSite	+	−	Mountain	Peninsula
−	+	TelecomFacility	Monument	+	−	Island	Valley
−	+	ReligiousLocation	Canal	+	−	Government	Parliament
−	+	InternationalOrganization	StockExchange	+	−	Service	Telecom
−	+	WaterRegion	PoliticalRegion	+	−	Park	Festival
−	+	InternetDomain	EntitySource	+	−	OilField	Province
−	+	ReligiousOrganization	Airline	+	−	Patent	AirplaneModel
−	+	RecreationalFacility	Capital	+	−	Ministry	Location
−	+	City	Archipelago	+	−	Delta	River
−	+	Pipeline	LaunchFacility	+	−	TVCompany	Movie
−	+	AstronomicalObject	Mountain				
−	+	GovernmentOrganization	AmusementPark				
−	+	AmusementPark	Galaxy				
−	+	LaunchFacility	Bridge				

2. Some disjointness axioms may depend on the context: whereas *Dog* and *Livestock* may be disjoint in most parts of Europe, in the Chinese Wordnet[6] the latter is actually a hypernym of the former.
3. Classes can have abstract individuals, like *Money*, *Message* or *Idea*.
4. Often the extension of two classes are disjoint, although their intension is not, e.g. *US President* and *Woman*. Annotators struggle with this difference.
5. Also, the extensions of two classes might be not disjoint, even though their intensions are: although *Weapon* and *Pitchfork* are disjoint intensionally (in the literal sense), their extensions do not need to be.
6. Roles and so called basic classes are often mixed, e.g. the role *Professor* and the *Person* itself that plays the role, which may be defined disjoint (depending on how roles are modeled [11]).
7. Mereological and instantiation relations can be mixed: a *Week* is part of a *Month*, so are these two classes disjoint? What about *Delta* and *River*?
8. Mixing other types of relations with instantiation relations may lead to misunderstandings: see for example the pairs *Movie*/*TVCompany*, *Government*/*Parliament*, or *Patent*/*AirplaneModel*, where the instances have close relations and thus seem to confuse the annotators.
9. Instantiations can occur at different levels of abstraction. E.g., when describing animals, *Eagle* may be the label of both an individual (e.g. of the class *Species*) and of a class itself. Are then the two classes *Species* and *Eagle* disjoint? Note that the individual *Eagle* is not the same as the class *Eagle*, but they may be connected via an axiom like $\mathit{Class{:}Eagle} \equiv \exists \mathit{species}.\{\mathit{Individual{:}Eagle}\}$.
10. Sometimes, lexical information is mixed with ontological one. The PROTON ontology contains concepts like *Alias* that form lexical information. Is a *JobTitle* disjoint from a *Job* or the *Person* having the *Job* or *JobTitle*?

[6] `http://www.keenage.com/`

Note that this list does not speak about problems of disjointness with regards to its definition in description logics, but rather with the problems our annotators had when they had to decide if two classes are disjoint or not. Many of the above problem types have a well-defined answer with regards to the formal semantics of disjointness, e.g. #7, where *Week* and *Month* are disjoint as they don't have common instances (since a week consists of seven days, and months consist of around 28-31 days. Note that the definition of week and month can change, but this basically means that we introduce new concepts which may or may not have the same name).

Recognizing the problem type would allow an ontology development environment to offer much more appropriate help than just a general description of the meaning of the disjointness axiom, which can be hard to apply at times.

Often the decision, if two classes are disjoint or not, will uncover underspecified or ambiguous classes, i.e. moot points in the description of one or both classes. Instead of simply adding (or, which is far harder to tract, *not* adding) a disjointness axiom, the rationale behind this decision should also be documented, following an ontology lifecycle methodology like DILIGENT [21] for the continuous evolution and refinement of the ontology.

4 Evaluation: Learning a Classifier

In this section we present the evaluation procedure and analyse the results of the comparison between the classifier which has been trained on the features described in Section 2 and the sets of manual annotations (see Section 3).

4.1 Experimental Settings

To train the classifier we skipped pairs of classes tagged with ? since the definition of disjointness only distinguishes between disjoint and not disjoint classes. For the rest of the evaluation we will consider this two-class problem. We evaluate our learned classifier against two baseline: the random and majority baseline.

Random Baseline: The idea of the random baseline is to randomly choose the target class of the classifier. As we have a two-class problem we will distribute the pairs equally over the two classes. This will result in a 50% baseline for accuracy as 50% of the $+$ examples will be classified in $+$ which means that these examples are classified correctly. The same holds for the $-$ class.

Majority Baseline: The majority baseline is determined by taking the largest class as default classification. This way, we will get a high accuracy if the classes are unequally distributed. In this case, of course, the majority baseline is much more difficult to beat than the random baseline. Nevertheless, since in the experiments at hand we only have to deal with two classes ($+$ or $-$) which are not equally distributed, the majority baseline should be considered as more realistic than the random baseline.

Classifier settings: In order to be able to classify each pair of classes as being disjoint ($+$) or not ($-$), we trained a classifier based on the manual taggings created by human annotators. The features for the classifier are the confidence values obtained from various sources as described in section 2.

We tested a couple of different classifiers made available by the Weka package[7]. In general, decision trees outperformed all other classifiers – maybe, because of the highly selective character of our features – while the performance of different types of decision trees was more the less comparable. Therefore, we finally chose the *ADTree* classifier [7] with default settings for our experiments which shows very good performance while at the same time providing interpretable results.

First, we performed a 10-fold cross-validation against the majority votes 100% and 50% of the datasets A (ontologists), B (students), C (all) and E (random) (cf. table 1a). The results for the random dataset are included to show the performance of our approach for an unbiased dataset (E contains examples chosen randomly from the set of all possible pairs without any selection criteria). To get the results for dataset D (siblings), we split dataset C into two independent parts - one for evaluation and one for training. The training set for the evaluation with dataset D consists of all manually tagged pairs except for the siblings.

4.2 Results

Table 5 and 4 list the results of our evaluation experiments by means of Precision (P), Recall (R), F-Measure (F) and Accuracy (Acc) (for definitions cf. [23]). From the tables it becomes evident that we easily beat the baselines for the datasets A (experts), B (students) and C in both cases majority vote 50% and 100%. With an accuracy of over 90% the performance of our system for dataset C is remarkable, especially in the case of the total majority vote. These results are comparable with the human inter-annotator agreement for experts and students – and even better for dataset C (90.9%) in comparison to the human agreement of 86.4%.

Dataset D, which only contains pairs of siblings, is certainly the most difficult to handle – for the classifier, but also for the human annotators – because, as explained in Section 3.3, siblings are semantically close, so that differences between their intensions and extensions may often be hard to grasp. As dataset D shows a relatively low average agreement compared to the other datasets (cf. table 1b) the classifier seems to have more

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

Table 4. Evaluation against Majority Vote 50% (ADTree)

Dataset	P			R			F			Acc	Acc_{random}	$Acc_{majority}$
	$+$	$-$	avg.	$+$	$-$	avg.	$+$	$-$	avg.			
A	0.815	0.638	0.727	0.823	0.626	0.725	0.819	0.632	0.726	0.757	0.500	0.666
B	0.807	0.642	0.725	0.844	0.580	0.712	0.825	0.609	0.717	0.758	0.500	0.675
$avg.$	0.811	0.640	0.726	0.834	0.603	0.719	0.822	0.621	0.722	0.758	0.500	0.671
C	0.854	0.682	0.768	0.874	0.644	0.759	0.864	0.663	0.764	0.806	0.500	0.704
D	0.558	0.628	0.593	0.255	0.861	0.558	0.350	0.726	0.538	0.615	0.500	0.593
E	0.910	0.761	0.836	0.990	0.250	0.620	0.948	0.376	0.662	0.904	0.500	0.884

Table 5. Evaluation against Majority Vote 100% (ADTree)

Dataset	P			R			F			Acc	Acc_{random}	$Acc_{majority}$
	$+$	$-$	avg.	$+$	$-$	avg.	$+$	$-$	avg.			
A	0.896	0.720	0.808	0.903	0.703	0.803	0.899	0.712	0.806	0.851	0.500	0.738
B	0.866	0.790	0.828	0.942	0.599	0.771	0.903	0.681	0.792	0.851	0.500	0.734
$avg.$	0.881	0.755	0.818	0.923	0.651	0.787	0.901	0.697	0.799	0.851	0.500	0.736
C	0.934	0.823	0.879	0.946	0.789	0.868	0.940	0.805	0.873	0.909	0.500	0.760
D	0.237	0.806	0.522	0.786	0.260	0.523	0.364	0.394	0.379	0.379	0.500	0.774
E	0.977	0.955	0.966	0.998	0.600	0.799	0.987	0.737	0.862	0.976	0.500	0.944

difficulties to learn it. This is also expressed by the very bad classification accuracy with 37% for majority vote 100%.

An investigation of the learned classifier revealed that the rather important taxonomic feature (see Section 2.2) is not well populated in the siblings part of the dataset. To analyse the influence of this feature we constructed a dataset without this feature. As expected the accuracy for the training dataset drops, whereas for the evaluation set it is improved considerably from 37.9% to 74.2%. Moreover, the results for the majority vote 50% rise to 76.6% which can be interpreted as an indication to the noise insert by this feature.

Our approach seems to work very well also for the random dataset E as we got a better accuracy in both cases. The difference to the majority baseline is much smaller than for A, B, and C but the baseline of around 90% is very difficult to beat. To conclude, the results – not only for the random dataset – are very promising and allow us to setup a competitive classifier to support ontology engineering.

In order to find out which classification features contributed most to the overall performance of the classifier we performed an analysis of our initial feature set with respect to the gain ratio measure [14]. The ranking produced for data set C clearly indicates an exceptionally good performance of the features taxonomic overlap (Section 2.1), similarity based on WordNet and lexical context (Section 2.3), and del.icio.us (Section 2.1). The contribution of other features such as the one presented in Section 2.4 relying on lexico-syntactic patterns seems to be less substantial. However as the classification accuracy tested on every single feature is always below the overall performance the combination of all features is necessary to achieve a very good overall result.

5 Related Work

Several ontology learning frameworks have been designed and implemented in the last decade. The Mo'K workbench [1], for instance, basically relies on unsupervised machine learning methods to induce concept hierarchies from text collections. In particular, the framework focuses on agglomerative clustering techniques and allows ontology engineers to easily experiment with different parameters. OntoLT [3] is an ontology learning plug-in for the Protégé ontology editor. It is targeted at end users and heavily relies on linguistic analysis, i.e. it makes use of the internal structure of noun phrases to derive ontological knowledge from texts. JATKE[8] is a Protégé based unified platform for ontology learning which allows for inclusion of modules for ontology learning. The OntoLearn framework [13] mainly focuses on the problem of word sense disambiguation, i.e. of finding the correct sense of a word with respect to a general ontology or lexical database. TextToOnto [12] is a framework implementing a variety of algorithms for diverse ontology learning subtasks. In particular, it implements diverse relevance measures for term extraction, different algorithms for taxonomy construction as well as techniques for learning relations between concepts. The recent RelExt approach [18] focusses on the extraction of triples, i.e. classes connected by a relation. None of the mentioned approaches deals with disjointness.

6 Conclusion and Future Work

Learning of disjointness axioms is an intuitive and useful extension of existing ontology learning frameworks. We have motivated the need for richter ontologies which include disjointness axioms and presented an approach consisting of a number methods to extract expressive feature for learning disjointness from different sources of evidence. In a thorough evaluation our learning approach behaved competitive to human annotators. As a by-product we captured lessons learned from human annotators with respect to their difficulties when modeling disjointness axioms.

Future work includes a combination with ontology evaluation approaches for richly axiomatized ontologies such as [17]. Moreover, we want to integrate the novel methods into the Text2Onto [4] framework for ontology learning from texts.

Acknowledgments: Research reported in this paper has been partially financed by the EU in the IST project SEKT (IST-2003-506826) (`http://www.sekt-project.com`). We would like to thank our colleagues, especially Peter Haase, for fruitful discussions, and all the students and colleagues at the AIFB and the University of Kassel for providing us with more than 10,000 taggings.

References

1. G. Bisson, C. Nedellec, and L. Canamero. Designing clustering methods for ontology building - The Mo'K workbench. In *Proc. of the ECAI Ontology Learning Workshop*, pages 13–19, 2000.
2. E. P. Bontas, C. Tempich, and Y. Sure. ONTOCOM: A cost estimation model for ontology engineering. In I. Cruz et al., editors, *Proceedings of the 5th International Semantic Web Conference (ISWC 2006)*, volume 4273 of *LNCS*, pages 625–639. Springer-Verlag Berlin Heidelberg, 2006.
3. P. Buitelaar, D. Olejnik, and M. Sintek. OntoLT: A protégé plug-in for ontology extraction from text. In *Proc. of the 2nd Int. Semantic Web Conference (ISWC2003)*, 2003.
4. P. Cimiano and J. Völker. Text2onto – a framework for ontology learning and data-driven change discovery. In *Proc. of the 10th Int.l Conf. on Applications of Natural Language to Information Systems (NLDB'05)*, June 2005.

[8] `http://jatke.opendfki.de/`

5. P. Cimiano and J. Völker. Towards large-scale, open-domain and ontology-based named entity classification. In G. Angelova, K. Bontcheva, R. Mitkov, and N. Nicolov, editors, *Proc. of the International Conference on Recent Advances in Natural Language Processing (RANLP)*, pages 166–172, Borovets, Bulgaria, September 2005. INCOMA Ltd.
6. C. Fellbaum. *WordNet, an electronic lexical database*. MIT Press, 1998.
7. Y. Freund and L. Mason. The alternating decision tree learning algorithm. In *ICML*, pages 124–133, 1999.
8. N. Guarino and C. A. Welty. A formal ontology of properties. In *Knowledge Acquisition, Modeling and Management*, pages 97–112, 2000.
9. P. Haase and J. Völker. Ontology learning and reasoning - dealing with uncertainty and inconsistency. In *Proc. of the Workshop on Uncertainty Reasoning for the Semantic Web (URSW)*, pages 45–55, 2005.
10. Z. Harris. Distributional structure. In J. Katz, editor, *The Philosophy of Linguistics*, pages 26–47, New York, 1985. Oxford University Press.
11. K. Kozaki, E. Sunagawa, Y. Kitamura, and R. Mizoguchi. Fundamental considerations of role concepts for ontology evaluation. In *Proc. of the Workshop EON – Evaluation of Ontologies for the Web*, 2006.
12. A. Maedche and S. Staab. Ontology learning for the semantic web. *IEEE IS*, 16(2), 2001.
13. R. Navigli, P. Velardi, A. Cucchiarelli, and F. Neri. Extending and enriching WordNet with OntoLearn. In *Proc. of the GWC 2004*, pages 279–284, 2004.
14. J. R. Quinlan. *C4.5 Programs for Machine Learning*. Morgan Kaufmann, California, 1993.
15. A. Rector, N. Drummond, M. Horridge, J. Rogers, H. Knublauch, R. Stevens, H. Wang, and C. Wroe. OWL pizzas: Practical experience of teaching OWL-DL – common errors & common patterns. In *Proc. of EKAW 2004*, pages 63–81, 2004.
16. T. Rose, M. Stevenson, and M. Whitehead. The reuters corpus volume 1-from yesterdays news to tomorrows language resources. *Proc. of the Third International Conference on Language Resources and Evaluation*, pages 29–31, 2002.
17. S. Schlobach. Debugging and semantic clarification by pinpointing. In *Proc. of the 2nd European Semantic Web Conference (ESWC2005)*, volume 3532 of *LNCS*, pages 226–240. Springer, 2005.
18. A. Schutz and P. Buitelaar. RelExt: A tool for relation extraction in ontology extension. In *Proc. of the 4th International Semantic Web Conference (ISWC2005)*, 2005.
19. I. Terziev, A. Kiryakov, and D. Manov. Base upper-level ontology (BULO) guidance. SEKT deliverable 1.8.1, Ontotext Lab, Sirma AI EAD (Ltd.), 2004.
20. J. Völker, D. Vrandecic, and Y. Sure. Automatic evaluation of ontologies (AEON). In *Proc. of the 4th International Semantic Web Conference (ISWC2005)*, volume 3729 of *LNCS*, pages 716–731. Springer, 2005.
21. D. Vrandečić, H. S. Pinto, Y. Sure, and C. Tempich. The DILIGENT knowledge processes. *Journal of Knowledge Management*, 9(5):85–96, 2005.
22. T. D. Wang. Gauging ontologies and schemas by numbers. In *Proc. of the Workshop EON – Evaluation of Ontologies for the Web*, 2006.
23. I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Series in Data Management Sys. Morgan Kaufmann, 2nd edition, June 2005.
24. Z. Wu and M. Palmer. Verb semantics and lexical selection. In *32nd. Annual Meeting of the Ass. for Computational Linguistics*, pages 133–138, New Mexico, 1994.