# Inductive Learning of Disjointness Axioms

Daniel Fleischhacker and Johanna Völker*

KR & KM Research Group, University of Mannheim, Germany
{daniel,johanna}@informatik.uni-mannheim.de

**Abstract.** The tremendous amounts of linked data available on the web are a valuable resource for a variety of semantic applications. However, these applications often need to face the challenges posed by flawed or underspecified representations. The sheer size of these data sets, being one of their most appealing features, is at the same time a hurdle on the way towards more accurate data because this size and the dynamics of the data often hinder manual maintenance and quality assurance. Schemas or ontologies constraining, e.g., the possible instantiations of classes and properties, could facilitate the automated detection of undesired usage patterns or incorrect assertions, but only few knowledge repositories feature schema-level knowledge of sufficient expressivity. In this paper, we present several approaches to enriching learned or manually engineered ontologies with disjointness axioms, an important prerequisite for the applicability of logical approaches to knowledge base debugging. We describe the strengths and weaknesses of these approaches and report on a detailed evaluation based on the DBpedia dataset.

**Keywords:** Linked Data, Ontology Learning, OWL, Data Mining

## 1 Motivation

The success of the Open Linked Data initiative and the fast growing number of knowledge repositories on the web have paved the way for the development of various semantic mashups and applications. However, these applications often need to face the challenges posed by flawed or underspecified knowledge representations. While the redundancy of structured data on the web can help to compensate for many of those problems, even applications based on lightweight knowledge representations benefit from more accurate semantic data in repositories such as DBpedia, Freebase as well as in other, domain-specific knowledge bases.

Ontologies, or generally speaking schemas, constraining the possible instantiations of classes and properties are a valuable means to improve the quality of linked data sets. They can enable the automated detection of undesired usage patterns or incorrect assertions. However, only few knowledge repositories feature schema-level knowledge of sufficient expressivity, and thus Auer and Lehmann [3] demanded that "algorithms and tools have to be developed for improving the structure, semantic richness and quality of Linked Data". In particular, disjointness axioms would be useful to enable more

* Johanna Völker is financed by a Margarete-von-Wrangell scholarship of the European Social Fund (ESF) and the Ministry of Science, Research and the Arts Baden-Württemberg.

expressive query answering as well as the detection of logical inconsistencies indicating potential modeling errors. Hitzler and van Harmelen [13] point out, for instance, that annotating linked data with ontologies that include class disjointness can help to solve the object reconciliation problem, i.e., the discovery of identical individuals across data sets. The following example taken from the DBpedia data set illustrates further benefits potentially provided by the addition of disjointness axioms:[1]

*Dirk_Bouts* dbo:nationality *Netherland*
*Netherland* rdf:type dbo:Book
dbo:nationality rdfs:domain dbo:Person

Considering these RDF triples, we find the dbo:nationality relation linking *Dirk_Bouts* to the DBpedia resource *Netherland*, which is explicit asserted to be of type dbo:Book in DBpedia. This error stems from a spelling mistake in the Wikipedia infobox of the article about Dirk_Bouts, as the value of the `nationality` property is given as `[[Netherland]]` and thus points to the wrong Wikipedia article. Note that detecting the error by logical means would not only require a properly specified range restriction of the dbo:nationality relation,[2] but it would also demand for the existence of a disjointness axiom:

dbo:nationality rdfs:range dbo:Country
dbo:Country owl:disjointWith dbo:Book

These two statements would allow us to infer that *Netherland* must be a country, and hence can *not* be a book since dbo:Book and dbo:Country are declared disjoint. This would be a logical contradiction to the previous rdf:type assertion. Since such logical contradictions can be spotted by automated means [18,21], such a manual or automatic enrichment of ontologies with further axioms can provide the maintainers of a knowledge base with valuable pointers to potential problems. In practice, of course, it will not always be clear whether the newly added or any of the existing statements are incorrect and thus should be removed – especially, if the former were generated automatically they should therefore be associated with provenance information such as certainty values to increase the efficiency of manual or automated debugging (e.g., [19]).

In this paper, we present a set of novel inductive methods for automatically enriching ontologies with disjointness axioms which could be used to extend previous approaches to inducing schema-level knowledge from knowledge bases. These methods exhibit three characteristics that we consider essential, especially for ontology generation from linked data: They are scalable enough to work on large RDF repositories such

[1] For the sake of brevity, we assume `http://dbpedia.org/resource/` to be the default namespace and use the prefix dbo: for abbreviating the URI of the DBpedia ontology (`http://dbpedia.org/ontology/`).

[2] There are several approaches to the automatic acquisition of domain or range restrictions from text or linked data including, for example, early work by Mädche and Staab [16]. Note that it is also possible to induce these types of axioms from linked data, e.g., by association rule mining [25]. For a comprehensive overview of approaches to mining structured web data, see Stumme et al. [23].

as DBpedia, and robust insofar as they can tolerate a certain number of incorrect assertions. Moreover, these methods provide users and applications with a certainty value for each of the generated disjointness axioms.

The remainder of this paper is structured as follows. After giving a brief overview of related work (cf. Section 2), we describe the three approaches that we developed in order to enrich ontologies with disjointness axioms (Section 3), including statistical correlation analysis as well as two algorithms for mining association rules. In Section 4, we report on a comparative evaluation of these approaches, before concluding with a summary and an outlook to future work (cf. Section 5).

## 2 Related Work

The work presented in this paper relates to previous approaches in the field of ontology learning and the automated generation of disjointness axioms. It also follows a variety of automated approaches supporting the evaluation and maintenance of knowledge bases on the web.

Early work on learning class disjointness has been done by Haase and Völker [11], who suggested an unsupervised method for mining disjointness axioms from natural language text. Their method based on lexico-syntactic patterns was later incorporated into the LeDA framework for acquiring disjointness by supervised machine learning [26]. Unlike the approaches suggested by this paper, LeDA does not crucially hinge on the existence of class membership assertions. However, the learning algorithm needs to be trained on a manually created set of disjointness axioms and requires various kinds of background knowledge, whose fit to the data set at hand can be assumed to have a huge impact on the quality of the generated disjointness axioms.

Disjointness axioms are also generated by more general ontology learning approaches. Especially, inductive methods based on inductive logic programming [14] or formal concept analysis [4] are applicable to the task at hand, but so far none of them has been evaluated with regard to the quality of acquired disjointness axioms. This also holds for methods based on association rule mining as proposed, e.g., by Völker and Niepert [25]. Their approach referred to as statistical schema induction is complemented by a simple heuristic for introducing disjointness axioms into schemas automatically generated from RDF data, that assumes non-overlapping classes with more than a hundred individuals to be disjoint – a rough rule of thumb that cannot be expected to work in the general case.

A more well-known heuristic for introducing disjointness axioms into existing ontologies has been proposed by Schlobach [20]. His approach known as *semantic clarification* aims to make logical debugging techniques applicable to lightweight ontologies. It relies on the "strong disjointness assumption" [9], which postulates disjointness among sibling classes, as well as on the pinpointing technique for discovering and fixing the causes of logical incoherence. A similar strategy was later adopted by Meilicke et al. [17], who showed that automatically generated disjointness axioms can facilitate the detection of incorrect correspondences between classes in different lightweight ontologies.

Particularly related to our approaches is recent work by Lehmann and Bühmann [15], who developed ORE, a tool for repairing different types of modeling errors in ontologies. It uses the DL-Learner framework [14], which has been shown to scale up to large knowledge bases [12], in order to enrich ontologies by class expressions automatically induced from existing instance data. Inconsistencies or incoherences resulting from this enrichment serve as indicators for modeling flaws in the knowledge base. While their approach does not focus on disjointness axioms, they emphasize the usefulness of negation in debugging knowledge bases, it would be worthwhile investigating ways to integrate our methods into ORE.

In Section 4, we will take a closer look at LeDA as well as the strong disjointness assumption and how their performance compares to the methods presented in this paper.

## 3 Methods for Learning Disjointness

In this section, we present three approaches to enriching the schemas of large knowledge repositories with disjointness axioms. First, after briefly introducing the syntax and semantics of disjointness axioms in the Web Ontology Language OWL, we describe an approach that is based on statistical correlation analysis (cf. Section 3.1). We then elaborate on the use of association rule mining techniques for learning disjointness, and outline the ideas underlying two alternative methods supporting the discovery of negative association rules (see Section 3.2). For a detailed comparison of these methods with state-of-the-art approaches to generating disjointness axioms, see Section 4.

Both RDF Schema[3] and the Web Ontology Language (OWL)[4] are standards proposed by the W3C for expressing schema-level knowledge on the web. RDFS allows for modeling lightweight schemas consisting of classes (or concepts), individuals (or instances), as well as properties (or relations) connecting these individuals. It also provides means to express class subsumption, domain and range restrictions of properties, and equality of individuals, for example, but the RDFS standard does not contain a negation operator or other means to model negative knowledge. Additional expressivity required, e.g., by reasoning-based applications, is offered by OWL, which extends RDFS by additional constructs, such as class and property disjointness.[5] Note that there is an RDF-based serialization for every OWL ontology. For the sake of brevity, however, we will henceforth use the description logic notation for talking about disjointness axioms.

Using class disjointness, it is possible to state that two classes *cannot* have any common individuals according to their intended interpretation, i.e., that the intersection of these classes is necessarily empty in all possible worlds. For example, the OWL axiom

$$\mathsf{Person} \sqsubseteq \neg \mathsf{Plant}$$

[3] `http://www.w3.org/TR/rdf-schema/`

[4] `http://www.w3.org/TR/owl2-overview/`

[5] Property disjointness has been added as part of OWL 2 which has become a W3C recommendation on October 27, 2009.

or equivalently, $\mathsf{Person} \sqcap \mathsf{Plant} \sqsubseteq \bot$, expresses the fact that nothing can be both a person and a plant, as the intersection of these classes is subsumed by $\bot$ and hence necessarily empty. This does *not* imply, however, that if two classes do not have any common individuals in a particular knowledge base, they are meant to be disjoint. This is because of the Open World Assumption holding in OWL as well as in RDFS, which states that knowledge not explicitly (or implicitly) said to be true is not treated as being false, but as being unknown. For this reason, the assertions $\mathsf{Person}(Tom)$ and $\mathsf{Plant}(Tom)$ would not cause a logical contradiction, and thus would not necessarily be recognized as a modeling error by a mere reasoning-based approach, unless we add an axiom stating that Person and Plant are disjoint classes.

Ontologies using disjointness may exhibit two kinds of logical contradiction: incoherence and inconsistency. An ontology is incoherent if it contains a class $C$ which is not satisfiable, i.e., which is empty according to every possible interpretation. An incoherent class could be introduced in an ontology by the following axioms.

$$\mathsf{Human} \sqsubseteq \mathsf{Animal}$$
$$\mathsf{Human} \equiv \mathsf{Person}$$
$$\mathsf{Person} \sqsubseteq \neg\mathsf{Animal}$$

Since all humans are defined to be animals and the classes Person and Human are defined to be equivalent, the disjointness between Person and Animal renders the class Person unsatisfiable. Incoherences are rarely introduced on purpose, since in most real-world application scenarios there is little reason for creating a named class that is not intended to contain individuals. When it comes to logical inference over an ontology, incoherent classes mainly have a local effect as they are subsumed by and at the same time disjoint to all other classes.

Inconsistency usually has a more significant impact on the practical usefulness of an ontology for reasoning-based applications. An ontology being inconsistent means that there is no model for this ontology which, e.g., could be caused by an individual belonging to a non-satisfiable class. Since a fact can be inferred from an ontology iff it is valid in all models of the ontology (and this trivially holds if no model exists), inconsistencies prevent most standard reasoners from performing meaningful inference. In an ontology containing the axioms from our incoherence example, the axiom

$$\mathsf{Person}(Kim)$$

would lead to an inconsistent ontology because the unsatisfiable class Person is assigned the instance $Kim$.

Both inconsistencies and incoherences can be detected by automated means for inconsistency diagnosis. Often, the results of a diagnosis, a set of axioms that together cause a logical contradiction, indicate some kind of modeling error in the knowledge base, that might have remained unnoticed if the ontology had not turned inconsistent. For this reason, a certain level of logical expressivity introduced, e.g., by axioms containing negation operators, is desirable as it facilitates the occurrence of logical contradictions whenever classes, individuals or properties are not used in agreement with their intended semantics. However, many of the available linked data repositories such

as DBpedia only use lightweight schemas – either in formats not supporting class disjointness, like RDFS, or just not stating disjointness though possible format-wise. Thus, it is not possible to apply logical debugging methods to these semantic resources. To enable more elaborate maintenance and quality assurance on linked data, we thus explored different ways to automatically enrich lightweight schemas by disjointness axioms.

For all approaches which we present in the following, we assume the data to be represented as depicted in Table 1. In this case, we have one row per instance contained in the data set and one column per class mentioned in the dataset resp. the corresponding ontology. For each instance, all existing rdf:type assertions are marked by a 1 in the corresponding column while 0 means that the instance is not assigned to a certain class. This table is a structured representation of a so-called *transaction database* which we will introduce more formally later-on in Section 3.2.

**Table 1.** Excerpt from a transaction database for the DBpedia dataset

| IRI | Place | City | Person | OfficeHolder |
|---|---|---|---|---|
| $Berlin$ | 1 | 1 | 0 | 0 |
| $Charles\_Darwin$ | 0 | 0 | 1 | 0 |
| $Eiffel\_Tower$ | 1 | 0 | 0 | 0 |
| $John\_F.\_Kennedy$ | 0 | 0 | 1 | 1 |
| $Golden\_Gate\_Bridge$ | 1 | 0 | 0 | 0 |

The approaches that we present in the remainder of this section are based on the paradigm of statistical inductive learning, i.e., they are based on the assumption that schema-level knowledge can be derived from an analysis of existing class membership (or rdf:type) assertions – either by association rule mining or the computation of statistical correlation values. In this respect, our approaches bear some resemblance with previous work on concept learning in description logics [14]. Even several features used in the LeDA framework, including the taxonomic overlap in terms of existing or automatically acquired individuals, can be considered inductive or extensional. In Section 4, we will take a closer look at LeDA as the only existing framework for learning disjointness, and how it compares to the new, purely inductive methods.

### 3.1 Correlation

The first approach we applied for generating disjointness axioms is measuring the correlation between class rdf:type assertions. Correlation coefficients are commonly used to rate the strength of linear relationships between two value sequences. One widely known correlation coefficient is Pearson's correlation coefficient which is also used in a similar fashion by Antonie and Zaïane [2] who combine it with association rules and use it for filtering.

For our experiments, the values we consider for computing the correlation coefficient are the values stating which instances belong to a given class. Given two classes $C_1$ and $C_2$, we take the sequences formed by the appropriate columns of our transaction

database and compute the correlation between these two sequences. For each instance, there exist four possibilities of class combinations which are shown in the following table, e.g., $n_{10}$ is the number of transactions containing class $C_1$ but not class $C_2$.

| | $C_1$ | $\neg C_1$ | |
|---|---|---|---|
| $C_2$ | $n_{11}$ | $n_{01}$ | $n_{*1}$ |
| $\neg C_2$ | $n_{10}$ | $n_{00}$ | $n_{*0}$ |
| | $n_{1*}$ | $n_{0*}$ | |

For this specific variant, the Pearson correlation coefficient can be reduced to the so-called $\phi$-coefficient given by

$$\phi = \frac{n_{11}n_{00} - n_{10}n_{01}}{\sqrt{n_{1*}n_{0*}n_{*0}n_{*1}}}$$

Given the resulting correlation coefficient, we can assess the strength of the correlation between the occurrences of the classes. According to Cohen [8] the results of the Pearson correlation coefficient can be coarsely divided into the categories of strong correlation for absolute values larger than $0.5$, medium correlation for absolute values in the range from $0.3$ to $0.5$ and small correlation for absolute values from $0.1$ to $0.3$. Absolute values of less than $0.1$ are inexpressive.

Since disjointness of two classes means that both classes must not have any common instantiations, classes being clearly disjoint would lead to $\phi = -1.0$. In contrast, a $\phi$-value of $1.0$ would show two perfectly equivalent classes based on the set of instances. Thus, negative correlation values having a high absolute value give the most evidence for both classes being disjoint and can be considered as a confidence value for the validity of the corresponding disjointness axiom.

For the transaction database shown in Table 1 and the classes Place and OfficeHolder, we would get $\phi = \frac{-3}{\sqrt{24}} = -0.61$. From this strong negative correlation the correlation-based algorithm could propose a disjointness axiom between both classes using the absolute correlation value as confidence.

### 3.2 Association Rule Mining

The other two approaches, we evaluated for inductively learning disjointness are based on association rules. Association rules are implication patterns originally developed for large and sparse datasets such as transaction databases of international supermarket chains. A typical dataset in such a setting can have up to $10^{10}$ transactions (rows) and $10^6$ attributes (columns). Hence, the mining algorithms developed for these applications are also applicable to the large data repositories in the open Linked Data cloud. Formally, the transaction database $D = (t_1, t_2, \ldots, t_n)$ contains transactions $t_j \subseteq I$ where $I = \{i_1, i_2, \ldots\}$ is the set of all possible items. As already described, each individual in the data set has a corresponding transaction which contains items representing classes the individual belongs to.

To mine association rules from such a transaction database the first step is to generate frequent itemsets contained in this database. For this purpose, there are multiple algorithms, the Apriori algorithm [1] being the most commonly used one. Frequent

itemsets are thereby identified by having a support value greater than a specified minimum support threshold whereas support is defined as

$$supp(X) = |\{t_i \in D : X \subseteq t_i\}|$$

In some cases, the support value is also defined relatively to the number of transactions contained in the database. Given these frequent itemsets, it is possible to generate association rules of the form $A \Rightarrow B$ where $A \subseteq I$ and $B \subseteq I$ are both itemsets. The confidence for a certain association rule is given by

$$conf(A \rightarrow B) = \frac{supp(A \cup B)}{supp(A)}$$

Thus, it shows the conditional probability of an itemset $B$ occurring in a transaction given the occurrence of an itemset $A$.

In this work, we do not want to generate rules like $A \rightarrow B$, which would to a degree resemble subsumption and equivalence relations in ontologies, but negative association rules where either $A$ or $B$ is negated like $A \rightarrow \neg B$. It is important to note that negative association rules, despite being similar to logical implications, do not capture the same logical meaning of implication or, in our special case, of disjointness. Association rules are not definitive rules but there may be some transactions which violate their proposition. This fact is partly represented by the confidence values which incorporate the fraction of transactions transgressing the association rule.

**Naïve Negative Association Rule Mining** Mining negative association rules poses different requirements to the association rule mining algorithms since typically there are many more items *not* contained in an itemset than items contained in it. In the typical problem domain of association rules, the number of possible items is too large to apply regular algorithms on the data set. However, our problem has a much more limited problem space. In our domain, we usually have to deal with a few hundreds or thousands of items (i.e., classes defined in the provided schema) whereas a typical association rule application deals with itemset sizes of up to $10^6$. Therefore, we are able to apply standard algorithms to our problem of mining negative association rules sometimes referred to as the naïve approach of mining negative association rules [2, 24]. To do this, for all classes the corresponding complements are also added to the transaction database and all instances not belonging to a class are marked as belonging to its respective complements. Applying this transformation to the transaction database depicted in Table 1, we get a the transaction database shown in Table 2. Using this approach, the standard association rule mining methods generate not only positive items but also negative ones and thus negative association rules. The example transaction database also illustrates one major shortcoming of this approach. Because of the addition of complement classes, we lose much of the original database's sparsity which greatly increases the space required to store such a transformed database.

As an example, we consider the itemset $\{\mathsf{Place}, \neg\mathsf{OfficeHolder}\}$ which reaches a support value of 3 because these items are contained in the transactions for $Berlin$, $Eiffel\_Tower$ and $Golden\_Gate\_Bridge$. For the negative association rule

**Table 2.** Transaction database containing materialized class complements

| IRI | Place | City | Person | OfficeHolder | ¬Place | ¬City | ¬Person | ¬OfficeHolder |
|---|---|---|---|---|---|---|---|---|
| $Berlin$ | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| $Charles\_Darwin$ | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| $Eiffel\_Tower$ | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| $John\_F.\_Kennedy$ | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| $Golden\_Gate\_Bridge$ | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

$\mathsf{Place} \rightarrow \neg\mathsf{OfficeHolder}$, we thus get a confidence value of

$$\frac{supp(\mathsf{Place}, \neg\mathsf{OfficeHolder})}{supp(\mathsf{Place})} = \frac{3}{3} = 1$$

**Negative Association Rule Mining** Typically, the datasets association rule mining is performed on are much larger than the ones used in our case. The number of items contained in one transaction is much smaller than the number of items not contained in a transaction and thus the number of frequent itemsets gets an enormous boost by such a naïve transformation of the transaction database. Since this greatly reduces the usefulness of standard positive association rules mining algorithms for mining negative association rules, there are several works regarding the development of special negative association rule mining algorithms [2]. In this work, we apply the negative association rule algorithm proposed by Zhang and Zhang [28]. Because of the complexity of this algorithm, we only give a short overview on it.

The negative association rule mining approach does not rely on materializing the item complements but instead searches for infrequent positive itemsets. Because of the sparsity of the original transaction database there is an almost exponential number of such infrequent positive itemsets. To reach a well enough performance for such an approach, pruning the search space is an important concern. After generating infrequent itemsets, Zhang and Zhang prune those itemsets which are not considered interesting given the minimum interest level. In this context, an itemsets is called *interesting* if its support exceeds the expected level of support. Based on the remaining negative itemsets, they define an approach to create all possible negative association rules using the probability ratio of each association rule as the corresponding confidence.

## 4 Experiments

The three approaches described in Section 3 are expected to perform differently on the task of generating disjointness axioms. Thus, in order to assess the quality of the produced disjointness axioms, we did an extensive comparison. As a state of the art approach for creating this type of axioms, we also included the LeDA framework [26] into our comparison which implements a supervised machine learning approach to acquiring disjointness axioms. Additionally, we compared the results to the heuristic proposed by Schlobach [20] (cf. Section 2). All of these methods have been tested against a gold standard of manually created disjointness axioms.

### 4.1 Setting

For our experiments,[6] we used the data set provided by the DBpedia project [5]. We applied the aforementioned approaches to a set of transaction tables containing the DBpedia data as of December 2010. In addition, we used the DBpedia ontology[7] version 3.5.1.

***Implementation.*** The values for the *correlation-based approach* were computed by our own implementation that determines the Pearson correlation coefficient as described in Section 3.1 for each pair of two classes that are stated to be of the types of some resources in DBpedia. The resulting correlation coefficients were used as confidence values for the corresponding disjointness axioms.

For mining the *association rules*, we used the Apriori miner system version 5.39 by Borgelt and Kruse [6], while the transaction tables for the naïve way of mining were generated by our own implementation. It is worthwhile mentioning that the materialization of the transaction database with representations for the respective complements of all contained classes, increased the size of the transaction data file from 13 MB to about 1.7 GB. On this materialized data file, we did multiple runs of Apriori miner with different settings for minimum support, minimum confidence and minimum interest. In addition, to actually make the amount of data manageable for the mining system, we had to limit the computation of frequent itemsets to sets consisting of two elements – no real limitation as we anyway only consider disjointness axioms relating two atomic classes.

Since we were unable to find a publicly available implementation of *negative association rule mining* suitable for our experiments, we implemented the algorithm described by Zhang and Zhang [28] ourselves and used it in the experiments described in the following.[8] This mining approach is also influenced by the parameters minimum support, minimum confidence and minimum interest.

***Configuration of LeDA.*** In order to evaluate our methods against LeDA [26], a state-of-the-art framework for learning disjointness axioms from heterogeneous sources of evidence, we updated the latest LeDA release as follows. We replaced the original KAON2-based implementation of the ontology backend by a version that uses Pellet and the Manchester OWL API. For our experiments, we used the set of features that performed well in recent experiments by Meilicke et al. [17]. These features are listed in Table 3. The naive bayes classifier of LeDA was trained on the upper module of the PROTON ontology[9] using the partial disjointness gold standard created by Völker et al. After this training phase, we applied the resulting classifier to the DBpedia ontology.

The background ontologies for the respective features have been automatically generated by using the Text2Onto tool [7]. We extracted these ontologies from a corpus of Wikipedia articles which we automatically assembled by downloading the articles

[6] All data used in our experiments is available from `http://code.google.com/p/gold-miner/`.

[7] `http://dbpedia.org/Ontology`

[8] The negative association rule mining system is integrated into the gold-miner system available from `http://code.google.com/p/gold-miner/`.

[9] `http://proton.semanticweb.org/`

corresponding to the class names contained in the ontologies. During this process only an automatic transformation from the camel-case naming style used in the ontologies to a sequence of single words has been done but no further disambiguation steps were applied. Ontologies automatically generated by Text2Onto contain instances and thus provide LeDA with a limited amount of instance data, e.g., for determining the instance overlap of two classes. Since the PROTON ontology used during training does not contain instance data, the instance-based features of LeDA could only be applied to the background ontologies but not to the original ontologies, which also resembles the setup employed by Meilicke et al.

**Table 3.** Features of the classification model used by LeDA [26].

| Feature | Description |
|---|---|
| $f_{doc}$ | Lexical context similarity (Wikipedia) |
| $f_{jaro-winkler}$ | Label similarity (JaroWinkler) |
| $f_{levenshtein}$ | Label similarity (Levenshtein) |
| $f_{overlap_c}$ | Taxonomic overlap wrt. subclasses |
| $f^b_{overlap_c}$ | Taxonomic overlap wrt. subclasses (learned ontology) |
| $f^b_{overlap_i}$ | Taxonomic overlap wrt. instances (learned ontology) |
| $f_{path}$ | Semantic distance |
| $f^b_{path}$ | Semantic distance (learned ontology) |
| $f_{prop}$ | Object properties |
| $f_{qgrams}$ | Label similarity (QGrams) |
| $f_{sub}$ | Subsumption |
| $f^b_{sub}$ | Subsumption (learned ontology) |
| $f_{wn_1}$ | WordNet similarity (Patwardhan-Pedersen v1) |
| $f_{wn_2}$ | WordNet similarity (Patwardhan-Pedersen v2) |

***Thresholds.*** For the association rule mining approaches, we chose an absolute support value of $10$ transactions and a confidence value of $0.8$ for both approaches. We did not set an interest threshold, i.e., our minimum interest value was $0$. For LeDA we just used the default thresholds given by the type of classifier and thus considered those classes as being disjoint for which the disjointness has been determined with a confidence of at least $0.5$. On the correlation approach, we applied the threshold values $0.05$ and $0.005$. Even if these values are both beneath the limits for meaningful correlations we nevertheless chose these after some first experiments since the results seemed to be promising.

***Baselines.*** In addition to these automatic approaches, we considered two more baselines and a gold standard of disjointness axioms manually added to the DBpedia ontology. For our gold standard, we asked several experienced ontology engineers to collaboratively enrich the DBpedia ontology with a full set of disjointness axioms. The first baseline approach is based on the *strong disjointness assumption* [9] used by *Schlobach*, thus it considers all siblings as being disjoint. This baseline approach reaches an accuracy of $92\%$ with respect to our gold standard. The second baseline approach is a simple *majority* approach. Since the vast majority of all possible class pairs is considered as

disjoint in the gold standard, the majority vote would be setting all pairs to disjoint. Regarding our gold standard, this method would reach an accuracy of $91\%$.

### 4.2 Handling Logical Inconsistency & Incoherence

After generating the list of disjointness axioms, we ordered the axioms by descending confidences and enriched the DBpedia ontology incrementally always adding the axiom with the highest confidence to the ontology. After each addition, the ontology is checked for coherence and, by also considering the instances contained in the DBpedia dataset, for consistency. If an incoherence or inconsistency is detected the lastly added axiom is removed from the ontology and pruned from the axiom list.

Note that checking for consistency and coherence is a non-trivial task. Due to the high number of instances contained in the DBpedia dataset it was not possible to use standard OWL reasoners like Pellet [22] which are not suited for such large repositories. Therefore, we applied a two-step approach regarding the detection of incoherence and inconsistence in the enriched ontology. Incoherence of a class means that this class is not satisfiable, i.e., it has to be empty to conform to the schema. Incoherence is not directly related to the actual existence of instances asserted to this or any other class. Thus, it is sufficient to only consider the raw schema and ignore the instance data which allows to query Pellet for satisfiability of each single class with good performance. To also preserve consistency, instances have to be considered either way. To do this, we combined results returned by the Pellet reasoner and from SPARQL queries sent to a Virtuoso RDF database containing the instances for the DBpedia ontology. We identified three different cases which he had to deal with to catch possible inconsistencies while enriching the ontology. We present these three cases in the following, all of them are checked after adding a new disjointness axiom to the ontology.

The most obvious case of inconsistency is caused by individuals assigned to the classes defined to be disjoint. This type of inconsistency is detectable by a non-empty result set for the SPARQL query[10]

```
SELECT ?x WHERE { ?x a <ClassURI1> . ?x a <ClassURI2> . }
```

Furthermore, the ontology is inconsistent if the SPARQL query

```
SELECT ?x WHERE { ?x a <ClassURI1> . ?x <PropURI> ?y . }
```

returns a non-empty result while the ontology entails $< \mathsf{ClassURI2} > \mathsf{rdf:domain} < \mathsf{PropURI} >$. Eventually, the third type of inconsistencies is detected if there are individuals fulfilling the query

```
SELECT ?x WHERE { ?x a <ClassURI1> . ?y <PropURI> ?x . }
```

while the ontology entails $< \mathsf{ClassURI2} > \mathsf{rdf:range} < \mathsf{PropURI} >$. It is worth noting that these patterns should be able to detect most inconsistencies which can occur in the DBpedia ontology by adding disjointness axioms. This is caused by the fact that the ontology only employs a limited set of the features provided by OWL, e.g., it does not contain cardinality restrictions.

[10] $< \mathsf{ClassURI1} >$ resp. $< \mathsf{ClassURI2} >$ are used as placeholders for the actual URIs of the classes the disjointness is stated for.

### 4.3 Creation of a Gold Standard

We created a gold standard of disjointness axioms on the DBpedia ontology for our experiments. During its manual creation the human ontology engineers came across some points which led to discussions. In the following, we describe some of these problems.

One point which turned out to be problematic is the distinction between different functions of buildings, e.g., *shopping mall* and *airport*. Since there are several airport buildings which also include shopping malls, it would be reasonable not to set both classes as disjoint. On the other side, the actual functions of buildings are intensionally disjoint because the airport functionality has no relation to the shopping mall function. In this specific case, the way of modeling used in the DBpedia ontology favors the first interpretation because the building may at the same time serve both functions, airport and shopping mall, without the possibility to divide both parts. Thus, the subclasses of *building* have been modeled to be pairwise disjoint.

Many problems during the creation of the gold standard were similar to these ones. Another example is the differentiation between *continents* and *countries* since there is, e.g., Australia which could be considered to be both a continent and a country. In this case, we opted for a way of modeling that takes into consideration the difference between the continent being a landmass and a country being an organizational unit possibly located at a landmass.

For some classes, their intension was not clear by just using the information available from the ontology itself. In these cases, the contents of the corresponding Wikipedia articles were used to clarify the respective notions and for some cases the extension of the specific class, i.e., the DBpedia instances assigned to these classes, were considered for clarification purposes.

### 4.4 Results & Discussion

**Analysis of Inconsistencies** After creating the gold standard, we materialized all disjointness axioms inferable from the gold standard. Based on this set of axioms, we performed an analysis of all axioms contained in the different automatically generated axiom lists. This way, we were able to compute a precision[11] regarding the actually inferable set of axioms in the gold standard. The results are shown in Table 4. For our computations, we assumed our gold standard to be complete, i.e., classes not being implicitly or explicitly stated as disjoint are considered *not* to be disjoint. Furthermore, since it is not meaningful to compute recall and accuracy on this level without any semantics of ontologies included, we left out these measures.

While enriching the DBpedia ontology with disjointness axioms our greedy approach raised various inconsistency errors caused by instances explicitly or implicitly asserted to both classes of a disjoint class pair. Usually, there are two sources of such errors. The first one is simply that the disjointness axiom determined by the automatic learning process is incorrect, the other one are incorrect, explicit or implicit rdf:type assertions. As described in Section 4.2, we only apply a heuristic approach for inconsistency detection which checks for three different kinds of inconsistencies. Table 5 shows the number of the different contradiction types.

[11] For definitions of precision, recall and accuracy, see [27].

**Table 4.** Statistics for automatically generated axioms without materialization (compared to materialized gold standard)

| | Total Axioms over Threshold | Correct Axioms | Precision |
|---|---|---|---|
| *LeDA* | 62,115 | 57,422 | 0.92 |
| *Correlation (0.005)* | 10,218 | 9,562 | 0.94 |
| *Correlation (0.05)* | 424 | 418 | 0.99 |
| *Naive ARM* | 14,994 | 13,623 | 0.91 |
| *Negative ARM* | 58 | 58 | 1.00 |

**Table 5.** Incoherences and inconsistencies detected while adding axioms

| | Axioms | Incoherences | Inconsistencies: Total | Range Conflict | Domain Conflict |
|---|---|---|---|---|---|
| *Gold Standard* | 59,914 | 0 | 1,339 | 1,302 | 37 |
| *LeDA* | 62,115 | 0 | 1,837 | 1,759 | 78 |
| *Correlation (0.005)* | 10,218 | 0 | 2,068 | 2,028 | 40 |
| *Correlation (0.05)* | 424 | 0 | 262 | 257 | 5 |
| *Naive ARM* | 14,994 | 230 | 1,025 | 980 | 45 |
| *Negative ARM* | 176 | 0 | 70 | 69 | 1 |

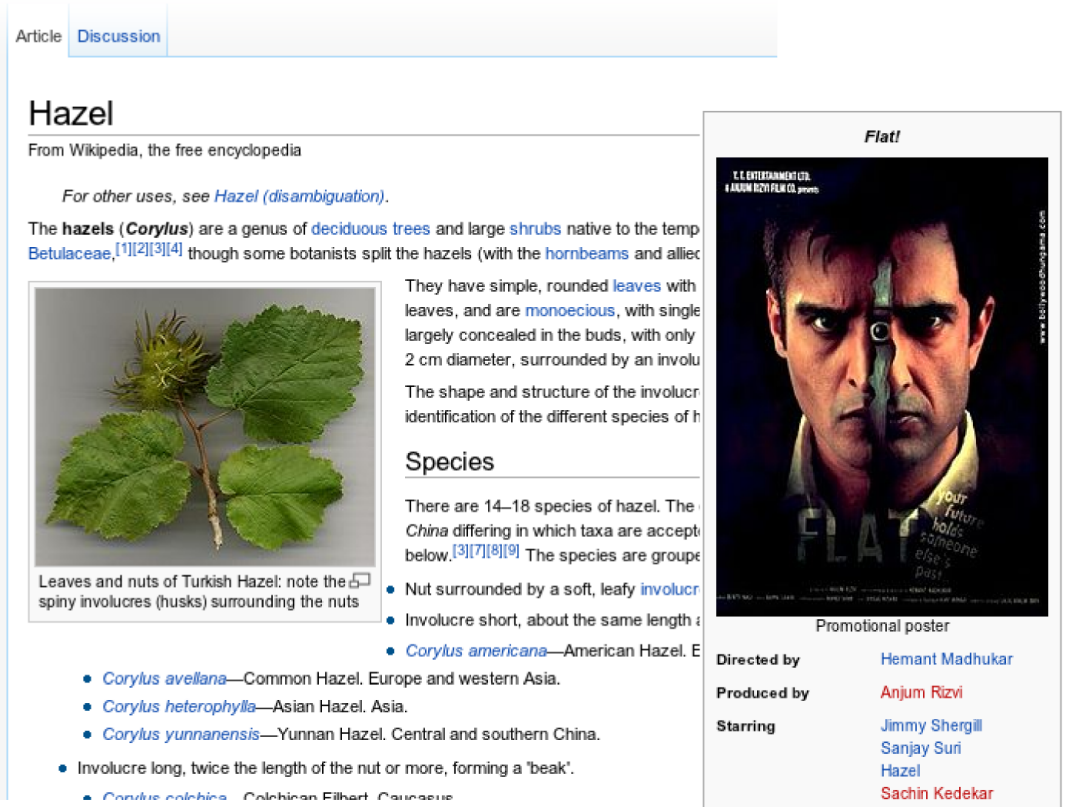
As we are able to see from the numbers, most errors are caused by range restrictions. This means that the range assertion of a specific property allows to infer a class assertion for an instance which conflicts with the generated disjointness axiom. According to our exploration of the inconsistencies, the most common error type is a *disambiguation error*. An example for this kind of error is revealed by the obviously correct disjointness axiom between the classes *Person* and *Plant* raised by the naive association rule mining approach. While adding this disjointness axiom to the DBpedia ontology, a range conflict for the property *starring* has been detected by our enriching process. A more elaborate examination of the DBpedia data showed that the range of *starring* is set to be the class *Person* but there exists a property *starring* between the instances *Flat!* (which is an Indian movie) and *Hazel* (the tree). This is caused by an incorrect cast reference in the infobox of the corresponding Wikipedia page of *Flat!*.[12] The correct reference would point to the actress *Hazel Crowney*.[13] Thus, the learned disjointness axiom helped to detect an error in Wikipedia which led to a wrong DBpedia information.

**Semantically Founded Evaluation** For the automatically generated axiom lists, we not only computed the precision for the raw lists (see Table 4) but also the measures of precision, recall and accuracy on their materializations. For this purpose, we enriched the DBpedia ontology by the automatically generated disjointness axioms using our greedy approach and afterwards computed a list of all disjointness axioms inferable from the ontology. The most important results are shown in Table 6.

The best performance regarding the automatically generated disjointness axioms is achieved by LeDA using the schema-based approach. Even if it did not reach the highest precision value, the recall and accuracy values are the highest of all automatic approaches. The precision with respect to the non-materialized list is at $0.92$ which is

[12] `http://en.wikipedia.org/w/index.php?title=Flat!&oldid=409238040`

[13] `http://en.wikipedia.org/wiki/Hazel_Crowney`




```
{{Multiple issues|cleanup=March 2010|unreferenced=November 2010}}
{{Infobox film
| name           = Flat!
| image          = Flat.JPG
| caption        = Promotional poster
| director       = [[Hemant Madhukar]]
| producer       = [[Anjum Rizvi]]
| Story Writer   = Hemant Madhukar
| starring       = [[Jimmy Shergill]]<br />[[Sanjay Suri]]<br />[[Hazel]]<br
/>[[Sachin Kedekar]]<br />[[Kaveri Jha]]
| music          = Bappa Lahiri
| cinematography = Manoj Shaw
| editing        = Bunty Nagi
| released       =
| country        = {{FilmIndia}}
| language       = Hindi
| budget         =
| gross          =
}}
```

**Fig. 1.** Disambiguation error in Wikipedia infobox for movie *Flat!*

**Table 6.** Evaluation of generated axioms[14]

| | Inferable Axioms | Correct Decisions | Correct Axioms | Precision | Recall | Accuracy |
|---|---|---|---|---|---|---|
| *Gold Standard* | 59,914 | - | - | - | - | - |
| *Schlobach* | 65,006 | 60,597 | 60,597 | 0.93 | 0.92 | 0.92 |
| *Majority* | 66,049 | 59,914 | 59,914 | 0.91 | 1.00 | 0.91 |
| *LeDA* | 60,314 | 57,557 | 55,868 | 0.93 | 0.85 | 0.87 |
| *Correlation (0.005)* | 41,786 | 43,049 | 39,350 | 0.94 | 0.60 | 0.65 |
| *Correlation (0.05)* | 3,246 | 9,381 | 3,246 | 1.00 | 0.05 | 0.14 |
| *Naive ARM* | 47,358 | 49,173 | 45,198 | 0.95 | 0.68 | 0.74 |
| *Negative ARM* | 10,790 | 16,925 | 10,790 | 1.00 | 0.16 | 0.26 |

quite high, though not as high as the correlation-based approaches. The main advantage of the correlation-based approaches is their high precision for certain thresholds but they suffer from a relatively low recall. To have a more detailed insight into the good performance of LeDA in this task, we also had a look at the feature usage by means of gain ratio evaluation of Weka. According to this analysis, the classifier mostly uses the features $f_{overlap_c}$ (gain ratio of $0.59$), $f_{qgrams}$ $(0.11)$ and $f^b_{overlap_c}$ $(0.11)$. The other activated features only gave a maximum gain ratio of $0.02$ or less. The negative association rule mining approach suffers from the same problem as the correlation-based one with respect to recall but in exchange reaches $1.0$ for precision which makes it very suitable if high precision is more important than high recall.

Taking into consideration the feature usage of LeDA, we can conclude that it is very dependent on the available training data and its similarity to the actual data which has to be classified. This is the main advantage of the induction-based approaches and regarding the statistics both association rule mining approaches perform well putting their specific emphasis either on precision or recall. Thus, if there is no appropriate training data available, these association rule mining-based approaches should be considered and chosen depending on the desired balance between precision and recall.

Regarding the baselines it is important to mention that while delivering some of the best results regarding precision, recall and accuracy, they are not suited for being used in general. For the majority approach, one has to determine first what is more common, disjointness or non-disjointness, which means creating a disjointness gold standard for the ontology. The strong disjointness assumption proposed by Schlobach is only this successful because for the DBpedia ontology the majority of siblings is in fact disjoint but in general there might be ontologies following other characteristics, e.g., for the *Person* subtree of the DBpedia ontology the assumption does not hold.

## 5 Conclusion & Outlook

In this paper, we presented a set of inductive methods for generating disjointness axioms from large knowledge repositories. We performed a comparative evaluation with the most commonly used methods and heuristics for generating disjointness axioms, and discussed the respective advantages of inductive, i.e. instance-based, and schema-based methods for learning disjointness. Our experiments indicate that it is possible to induce

[14] correct decisions = true positives + true negatives; correct axioms = true positives

disjointness axioms from an existing knowledge base with an accuracy that is well enough to help detecting inconsistencies in datasets. This is particularly true if there is no appropriate training data available to use tools like LeDA. As we have also shown, we were able to identify various problems in DBpedia by adding the automatically generated axioms to the DBpedia ontology.

While further experiments with other data sets will be indispensable, we are confident that our methods will facilitate the development of more efficient means to supporting users of large RDF repositories in detecting and fixing potential problems in the data sets. Future work includes the integration of our methods with existing approaches to acquiring schema-level knowledge from linked data, e.g., based on inductive logical programming [12] or association rule mining [25]. The incremental induction of schemas including disjointness axioms could facilitate, for example, an automated synchronization of the DBpedia ontology with DBpedia Live[15] and immediate diagnoses whenever changes to the underlying Wikipedia articles are submitted. Logical inconsistencies provoked by the enrichment of learned or manually engineered schemas would need to be resolved by methods for consistent ontology evolution [10]. Finally, we will investigate the applicability of our approaches to the problem of learning property disjointness.

## References


1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: Proceedings of the 20th International Conference on Very Large Data Bases (VLDB). pp. 487–499. Morgan Kaufmann (1994)
2. Antonie, M.L., Zaïane, O.R.: Mining positive and negative association rules: An approach for confined rules. In: Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD). pp. 27–38 (2004)
3. Auer, S., Lehmann, J.: Creating knowledge out of interlinked data. Semantic Web 1(1-2), 97–104 (2010)
4. Baader, F., Ganter, B., Sertkaya, B., Sattler, U.: Completing description logic knowledge bases using formal concept analysis. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI). pp. 230–235. AAAI Press (2007)
5. Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S.: DBpedia - a crystallization point for the web of data. Web Semantics 7(3), 154–165 (2009)
6. Borgelt, C., Kruse, R.: Induction of association rules: Apriori implementation. In: Proceedings of the 15th Conference on Computational Statistics (COMPSTAT). pp. 395–400. Physica Verlag (2002)
7. Cimiano, P., Völker, J.: Text2Onto. In: Natural Language Processing and Information Systems, Lecture Notes in Computer Science, vol. 3513, pp. 257–271. Springer Berlin / Heidelberg (2005)
8. Cohen, J.: Statistical power analysis for the behavioral sciences. Larwence Erlbaum, New Jersey, 2nd edn. (1988)
9. Cornet, R., Abu-Hanna, A.: Usability of expressive description logics – a case study in UMLS. In: Proceedings of the AMIA Annual Symposium. pp. 180–184 (2002)
10. Haase, P., van Harmelen, F., Huang, Z., Stuckenschmidt, H., Sure, Y.: A framework for handling inconsistency in changing ontologies. In: The Semantic Web – ISWC 2005. Lecture Notes in Computer Science, vol. 3729, pp. 353–367. Springer Berlin / Heidelberg (2005)


[15] `http://live.dbpedia.org`

11. Haase, P., Völker, J.: Ontology learning and reasoning – dealing with uncertainty and inconsistency. In: Uncertainty Reasoning for the Semantic Web I, Lecture Notes in Computer Science, vol. 5327, pp. 366–384. Springer Berlin / Heidelberg (2008)
12. Hellmann, S., Lehmann, J., Auer, S.: Learning of OWL class descriptions on very large knowledge bases. International Journal on Semantic Web and Information Systems 5(2), 25–48 (2009)
13. Hitzler, P., van Harmelen, F.: A reasonable semantic web. Journal of Semantic Web 1(1-2), 39–44 (2010)
14. Lehmann, J.: DL-Learner: learning concepts in description logics. Journal of Machine Learning Research (JMLR) 10, 2639–2642 (2009)
15. Lehmann, J., Bühmann, L.: ORE – a tool for repairing and enriching knowledge bases. In: The Semantic Web – ISWC 2010, Lecture Notes in Computer Science, vol. 6497, pp. 177–193. Springer Berlin / Heidelberg (2010)
16. Mädche, A., Staab, S.: Discovering conceptual relations from text. In: Proceedings of the 14th European Conference on Artificial Intelligence (ECAI). pp. 321–325. IOS Press (2000)
17. Meilicke, C., Völker, J., Stuckenschmidt, H.: Learning disjointness for debugging mappings between lightweight ontologies. In: Knowledge Engineering: Practice and Patterns, Lecture Notes in Computer Science, vol. 5268, pp. 93–108. Springer Berlin / Heidelberg (2008)
18. Parsia, B., Sirin, E., Kalyanpur, A.: Debugging owl ontologies. In: Proceedings of the 14th international conference on World Wide Web (WWW). pp. 633–640. ACM (2005)
19. Qi, G., Haase, P., Huang, Z., Ji, Q., Pan, J.Z., Völker, J.: A kernel revision operator for terminologies - algorithms and evaluation. In: The Semantic Web – ISWC 2008. pp. 419–434. Springer Berlin / Heidelberg (2008)
20. Schlobach, S.: Debugging and semantic clarification by pinpointing. In: The Semantic Web: Research and Applications. Lecture Notes in Computer Science, vol. 3532, pp. 27–44. Springer Berlin / Heidelberg (2005)
21. Schlobach, S.: Diagnosing terminologies. In: Proceedings of the 20th National Conference on Artificial Intelligence (NCAI). vol. 2, pp. 670–675. AAAI Press (2005)
22. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical owl-dl reasoner. Web Semantics 5, 51–53 (2007)
23. Stumme, G., Hotho, A., Berendt, B.: Semantic web mining: State of the art and future directions. Journal of Web Semantics 4(2), 124–143 (2006)
24. Teng, W.G., Hsieh, M.J., Chen, M.S.: On the mining of substitution rules for statistically dependent items. In: Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM). pp. 442–449. IEEE Computer Society (2002)
25. Völker, J., Niepert, M.: Statistical schema induction. In: The Semantic Web: Research and Applications. Lecture Notes in Computer Science, vol. 6643, pp. 124–138. Springer Berlin / Heidelberg (2011)
26. Völker, J., Vrandečić, D., Sure, Y., Hotho, A.: Learning disjointness. In: The Semantic Web: Research and Applications. pp. 175–189. Lecture Notes in Computer Science, Springer Berlin / Heidelberg (2007)
27. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann Series in Data Management Systems, Morgan Kaufmann, 2nd edn. (2005)
28. Zhang, C., Zhang, S.: Association rule mining: models and algorithms. Springer Berlin / Heidelberg (2002)