Learning Disjointness Axioms
Technical Report

Johanna Völker, Alexander Kesseler

Institute AIFB
Universität Karlsruhe (TH), Germany

1 Introduction

Our approach to the automatic acquisition of disjointness axioms relies on a machine learning classifier that determines disjointness of any two classes. The classifier is trained based on a "Gold Standard" of manually created disjointness axioms, i.e. pairs of classes each of which is associated with a label – "disjoint" or "not disjoint" – and a vector of feature values. As in our earlier experiments [7], we used a variety of lexical and logical features, which we believe to provide a solid basis for learning disjointness. These features are used to build an overall classification model on whose basis the classifier can predict disjointness for previously unseen pairs of classes. We implemented all features and auxiliary methods for training and classification within the open-source framework LeDA[1] (*Learning Disjointness Axioms*), a complete redesign and re-implementation of our original prototype. LeDA is open-source and publicly available under the LGPL license.

2 Classification Features

In the following, we give a brief overview of the 14 features we used for the experiments that we report on in Section 4. The current feature set differs from the original one [7] in that it focuses more on lexical and ontology-based similarity, which turned out to work very well in previous experiments. At the same time, we omitted several "weak" features including, e.g., OntoClean meta-properties and enumerations.

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, the individuals of a class do not necessarily have to *exist* in the ontology. Hence, the taxonomic overlap of two classes is considered not empty as long as there *could* be common individuals within the domain that is modeled by the ontology. Following these considerations, we developed several methods to compute the actual or possible overlap of two classes. Both of the following formulas are based on the Jaccard similarity coefficient [3].

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

$$\mathrm{f}_{overlap_i}(c_1, c_2) = \frac{|\{i \in I | c_1(i) \wedge c_2(i)\}|}{|\{i \in I | c_1(i) \vee c_2(i)\}|} \quad \mathrm{f}_{overlap_c}(c_1, c_2) = \frac{|\{c \in C | c \sqsubseteq c_1 \sqcap c_2\}|}{|\{c \in C | c \sqsubseteq c_1 \sqcup c_2\}|}$$

These two features are complemented by f_{sub}, that represents a particular case of taxonomic overlap, while at the same time capturing negative information such as class complements or already existing disjointness contained in the ontology. The value of f_{sub} for any pair of classes c_1 and c_2 is 1 for $c_1 \sqsubseteq c_2 \vee c_2 \sqsubseteq c_1$, 0 for $c_1 \sqsubseteq \neg c_2$ and $undefined$ otherwise.

$$\mathrm{f}_{sub}(c_1, c_2) = \begin{cases} 1 & c_1 \sqsubseteq c_2 \vee c_2 \sqsubseteq c_1 \\ 0 & c_1 \sqsubseteq \neg c_2 \\ \text{undefined} & \text{otherwise} \end{cases} \tag{1}$$

Note that subsumption as well as taxonomic overlap greater than zero mostly, but not necessarily implies non-disjointness. This particularly holds when the respective feature values are computed based on a learned background ontology (see further below), but also for many of the lightweight ontologies that we target with LeDA.

Semantic Distance. The semantic distance between two classes c_1 and c_2 is the minimum length of a path consisting of subsumption relationships between atomic classes that connects c_1 and c_2 (as defined in [7]).

$$\mathrm{f}_{path}(c_1, c_2) = \min_{p \in \text{paths}(c_1, c_2)} \text{length}(p) \tag{2}$$

Object Properties. This feature encodes the semantic relatedness of two classes, c_1 and c_2, based on the number of object properties they share. More precisely, we divided the number of properties p with $p(c_1, c_2)$ or $p(c_2, c_1)$ by the number of all properties whose domain subsumes c_1 whereas their range subsumes c_2 or vice-versa. This measure can be seen as a variant of the Jaccard similarity coefficient with object properties considered as undirected edges.

Label Similarity. The semantic similarity of two classes is in many cases reflected by their labels – especially, in case their labels share a common prefix or postfix. This is because the right-most constituent of an English noun phrase[2] can be assumed to be the lexical head, that determines the syntactic category and usually indicates the semantic type of the noun phrase. A common prefix, on the other hand, often represents a nominal or attribute adjunct which describes some semantic characteristics of the noun phrase referent. In order to compute the lexical similarity of the two class labels, we therefore used three different similarity measures:

[2] At least in English, people seem to prefer noun phrases for labeling classes.

- *Levenshtein*. The Levenshtein distance measures the edit distance of two strings, i.e. it returns the number of insertion, deletion and substitution operations that are required to transform one string into the other.
- *QGrams.* The idea of the QGrams metric is that two strings have a small edit distance if they have many q-grams in common. A q-gram is a substring of the original string with length q. Our implementation of the QGrams feature is based on the *SimMetrics*[3] library, with $q = 3$.
- *Jaro-Winkler.* The Jaro-Winkler distance is a variant of the Jaro distance metric taking into account the number of matching characters, the number of transpositions and the length of a common prefix.

WordNet Similarity. In order to compute the lexical similarity of two classes (their labels, to be precise), we applied two variants of a WordNet-based similarity measure by Patwardhan and Pedersen [4][4]. This similarity measure computes the cosine similarity between vector-based representations of the glosses, that are associated with the two synsets.[5]. We omitted any sort of word sense disambiguation at this point, assuming that every class label refers to the most frequently used synset it is contained in.

Features based on Learned Ontology. As an additional source of background knowledge about the classes in our input ontology we used an automatically acquired corpus of Wikipedia articles. By querying Wikipedia for each class label[6] we obtained an initial set of articles some of which were disambiguation pages. We followed all content links and applied a simple word sense disambiguation method in order to obtain the most relevant article for each class: For each class label we considered the article to be most relevant, which had, relative to its length, the highest "terminological overlap" with all of the labels used in the ontology. The resulting corpus of Wikipedia articles was fed into Text2Onto [1] to generate an additional background ontology for each of the original ontologies in our data set (cf. Section 3), consisting of classes, individuals, subsumption and class membership axioms.

Based on this newly acquired background knowledge, we defined the following features: *subsumption*, *taxonomic overlap of subclasses* and *individuals* – all of these are defined as their counterparts described above – as well as document-based *lexical context similiarity*, which we computed by comparing the Wikipedia article associated with the two classes. This type of similarity is in line with Harris' distributional hypothesis [2] claiming that two words are semantically similar to the extent to which they share syntactic contexts.

[3] `http://www.dcs.shef.ac.uk/~sam/simmetrics.html`

[4] `http://www.d.umn.edu/~tpederse/similarity.html`

[5] In WordNet, a synset is a set of (almost) synonymous words, roughly corresponding to a class or concept in an ontology. A gloss is a textual description of a synset's meaning, that most often also contains usage examples.

[6] Labels that were written as one word, though consisting of nominal compounds or other types of complex noun phrases.

Note that in order to enable the computation of feature values on the background ontology (e.g. taxonomic overlap of two classes), we had to map each class in the original ontology, i.e. the one to be enriched with disjointness axioms, to its counterpart in the automatically generated background ontology. We did this by determining for each class the one with the most similar label according to the Jaro-Winkler similarity measure.

3 Scenario

For the evaluation of our approach we used 6 ontologies from the OntoFarm [6] data set.[7] Since all of them represent knowledge about the same domain, namely the one of scientific conferences and workshops, we hoped that training performed on any of these ontologies would enable us to classify all other ontologies in the data set. Although our approach is domain-independent in principle, it most probably works best if the ontologies used for training and testing share lexical and structural characteristics, which is less likely to be the case for ontologies from completely different domains.

Gold Standard. In order to obtain a reference set of disjointness axioms for training and evaluating LeDA as well as to get an upper bound for the evaluation of mapping debugging, we manually added a minimal and complete number of disjointness axioms to the ontologies described above.[8] For these sets of explicit disjointness axioms, we computed the transitive closure by "materializing" all implicit disjointness relationships (positive examples). All pairs of classes whose disjointness could not be inferred from the initial, minimal set of axioms were considered not disjoint, thus serving as negative examples in the Gold Standard. This way we obtained a logically "cleaner" and much bigger data set than in of our earlier experiments with learning disjointness.

Ontology	**Classes**	**Properties**	**Disjointness**	**Added Axioms**
CMT	30	59	27	8
CRS	14	17	12	0
CONFTOOL	39	36	43	1
EKAW	77	33	83	25
PCS	24	38	0	23
SIGKDD	51	28	0	64

Table 1. Evaluation data sets. The last column shows the number of (explicit) disjointness axioms that were added in the creation of the Gold Standard.

[7] The ontologies can be obtained from `http://nb.vse.cz/~svabo/oaei2006/`.

[8] A set of disjointness axioms D is minimal with respect to ontology $\emptyset$ iff for all $d \in D$ we have $O \cup D \setminus \{d\} \not\models d$.

4 Evaluation

4.1 Setting

Training and Test Data. Unlike in our earlier experiments where a single ontology had to serve as a basis for both training and testing, the conference ontologies data set allows us to use $6 \times 5 = 30$ different combinations of ontologies for the evaluation of learning disjointness: for each of the 6 ontologies, we thus performed 5 experiments using each of the remaining ontologies as training data, and finally averaged over the individual results. Note that we removed all previously existing disjointness axioms from the ontologies prior to training and classification, because we wanted to get comparable results for all ontologies, independently of their respective numbers of existing disjointness axioms.

When testing on any of the ontologies, we always classified (and evaluated against) all possible pairs of classes – not just those explicitly marked as disjoint by the user. This is because we hoped that the resulting redundancy would help to rule out incorrectly classified pairs of classes in a post-processing (debugging) step. As a classifier for all experiments, we used Weka's implementation of NaiveBayes with default parameters[9], which turned out to perform slightly better in our initial tests than Decision Trees and SVMs – especially on the smaller data sets.

Baseline and Evaluation Measures. We generated macro-average values for precision, recall and F-measure[10] by averaging over the respective results on the sets of positive and negative examples. As a reasonable baseline for our evaluation, we computed a majority baseline for accuracy (Acc_{base}), that is defined as the number of examples in the majority class (e.g. "not disjoint") divided by the overall number of examples. The majority baseline represents the performance level that would be achieved by a naïve classifier that labels all entities in the test set with the majority class, i.e. "disjoint" for all ontologies in our data set. This simple, yet efficient strategy is hard to beat, especially for data sets that are relatively unbalanced and biased towards one of the target classes.

4.2 Results

	Training on CMT										
Classification	P_+	P_-	P	R_+	R_-	R	F_+	F_-	F	Acc	Acc_{base}
CMT	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	68.5
CONFTOOL	90.7	96.6	93.7	99.5	58.2	78.9	94.9	72.6	83.8	91.3	80.3
CRS	93.8	100.	96.9	100.	68.8	84.4	96.8	81.5	89.2	94.5	82.4
EKAW	92.2	95.9	94.1	99.6	52.0	75.8	95.8	67.4	81.6	92.5	85.1
PCS	80.7	98.0	89.4	99.5	53.2	76.4	89.1	69.0	79.1	83.9	66.3
SIGKDD	77.0	84.8	80.9	97.7	30.9	64.3	86.1	45.3	65.7	77.9	70.4

Table 2. Results (NaiveBayes)

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

[10] In the following we use the term F-measure to refer to the F_1-measure, where recall and precision are evenly weighted.

	Training on CONFTOOL										
Classification	P_+	P_-	P	R_+	R_-	R	F_+	F_-	F	Acc	Acc_{base}
CMT	83.8	85.0	84.4	95.1	59.9	77.5	89.1	70.2	79.7	84.0	68.5
CONFTOOL	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	80.3
CRS	93.6	84.6	89.1	97.3	68.8	83.1	95.4	75.9	85.7	92.3	82.4
EKAW	92.5	79.1	85.8	97.5	54.5	76.0	94.9	64.5	79.7	91.1	85.1
PCS	81.0	86.7	83.9	95.6	55.9	75.8	87.7	68.0	77.9	82.2	66.3
SIGKDD	77.1	81.0	79.1	96.9	31.7	64.3	85.9	45.5	65.7	77.6	70.4

Table 3. Results (NaiveBayes)

	Training on CRS										
Classification	P_+	P_-	P	R_+	R_-	R	F_+	F_-	F	Acc	Acc_{base}
CMT	81.1	92.0	86.6	98.0	50.4	74.2	88.8	65.1	77.0	83.0	68.5
CONFTOOL	88.7	75.3	82.0	96.0	50.0	73.0	92.2	60.1	76.2	86.9	80.3
CRS	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	82.4
EKAW	89.6	83.0	86.3	98.8	34.1	66.5	93.9	48.3	71.1	89.2	85.1
PCS	78.4	95.6	87.0	98.9	46.2	72.6	87.4	62.3	74.9	81.2	66.3
SIGKDD	76.1	90.5	83.3	98.8	26.2	62.5	86.0	40.6	63.3	77.3	70.4

Table 4. Results (NaiveBayes)

	Training on EKAW										
Classification	P_+	P_-	P	R_+	R_-	R	F_+	F_-	F	Acc	Acc_{base}
CMT	82.6	69.5	76.1	87.9	59.9	73.9	85.2	64.3	74.8	79.1	68.5
CONFTOOL	91.2	68.4	79.8	92.8	63.7	78.3	92.0	66.0	79.0	87.0	80.3
CRS	93.9	47.1	70.5	82.0	75.0	78.5	87.5	57.8	72.7	80.8	82.4
EKAW	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	85.1
PCS	81.0	76.1	78.6	90.7	58.1	74.4	85.6	65.9	75.8	79.7	66.3
SIGKDD	77.0	68.3	72.7	93.4	33.9	63.7	84.4	45.3	64.9	75.8	70.4

Table 5. Results (NaiveBayes)

	Training on PCS										
Classification	P_+	P_-	P	R_+	R_-	R	F_+	F_-	F	Acc	Acc_{base}
CMT	85.6	86.8	86.2	95.5	65.0	80.3	90.2	74.3	82.3	85.9	68.5
CONFTOOL	89.9	52.4	71.2	86.5	60.6	73.6	88.2	56.2	72.2	81.4	80.3
CRS	93.3	66.7	80.0	92.7	68.8	80.8	93.0	67.7	80.4	88.5	82.4
EKAW	91.9	91.8	91.9	99.2	50.0	74.6	95.4	64.7	80.1	91.9	85.1
PCS	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	66.3
SIGKDD	78.8	59.0	68.9	87.0	44.4	65.7	82.7	50.6	82.7	74.4	70.4

Table 6. Results (NaiveBayes)

	Training on SIGKDD										
Classification	P_+	P_-	P	R_+	R_-	R	F_+	F_-	F	Acc	Acc_{base}
CMT	94.7	84.4	89.6	92.4	88.7	90.1	93.5	86.5	90.0	91.3	68.5
CONFTOOL	92.6	53.3	73.0	84.4	72.6	78.5	88.3	61.4	74.9	82.1	80.3
CRS	93.5	75.9	84.7	95.3	68.8	82.1	94.4	72.1	83.3	90.7	82.4
EKAW	95.5	65.4	80.5	93.1	75.0	84.1	94.3	69.9	82.1	90.4	85.1
PCS	83.4	83.1	83.3	93.4	63.4	78.4	88.1	72.0	80.1	83.3	66.3
SIGKDD	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	70.4

Table 7. Results (NaiveBayes)

4.3 Feature Ranking

Table 8 shows a ranking of our features (see Section 2) with respect to their performance on our data set. The first column contains the gain ratio [5] values that were computed by averaging over all training data sets.

Not surprisingly, $f_{overlap_c}$ and f_{sub} performed best on our data set. This is because we exploited the taxonomy in the creation of the gold standard, assuming the ontologies to be carefully designed. A different methodology for acquiring the reference set of disjointness axioms and taxonomies of lower quality as in our earlier experiments would probably have led to different results.

Two features, $f_{overlap_i}$ and f_{doc} did not contribute at all, which is easy to explain for the first one, because there are no individuals contained in any of the ontologies, but not completely obvious for the f_{doc}, i.e. the document-based lexical context similarity. We assume that the performance of this feature suffers from the fact that only very few classes had associated Wikipedia articles in our experiments.

Gain Ratio	Feature	Description
0.53953417	$f_{overlap_c}$	Taxonomic overlap wrt. subclasses
0.52329850	f_{sub}	Subsumption
0.10352250	f_{prop}	Object properties
0.06150700	$f^b_{overlap_c}$	Taxonomic overlap wrt. subclasses (learned ontology)
0.04459383	f^b_{sub}	Subsumption (learned ontology)
0.03102233	f_{qgrams}	Label similarity (QGrams)
0.02297483	f_{wn_1}	WordNet similarity (Patwardhan-Petersen v1)
0.02070767	$f_{jaro-winkler}$	Label similarity (JaroWinkler)
0.01720867	$f_{levenshtein}$	Label similarity (Levenshtein)
0.00706467	f_{path}	Semantic distance
0.00089500	f_{wn_2}	WordNet similarity (Patwardhan-Petersen v2)
0.00010067	$f^b_{overlap_i}$	Taxonomic overlap wrt. instances (learned ontology)
0.0	f_{doc}	Lexical context similarity (Wikipedia articles)
0.0	$f_{overlap_i}$	Taxonomic overlap wrt. instances

Table 8. Feature ranking: average gain ration (full training set)

5 Conclusion

In this report, we have presented the implementation and evaluation of LeDA, an open-source tool for the automatic acquisition of disjointness axioms.

In the next weeks and month, we are going to perform a more detailed evaluation on different subsets of our data (e.g. all sibling classes) and a deeper analysis of errors made by the trained classifiers. By evaluating our approach on data from multiple domains we further hope to find out more about content related and structural characteristics of ontologies that hinder or facilitate learning disjointness. Finally, we will

implement a LeDA plugin for the NeOn Toolkit, in order to make our approach more easily applicable within a standard ontology engineering process.

Acknowledgements: Research reported in this paper has been partially financed by the EU under the IST-2006-027595 project NeOn (`http://www.neon-project.com`).

References

1. P. Cimiano and J. Völker. Text2onto - a framework for ontology learning and data-driven change discovery. In A. Montoyo, R. Munoz, and E. Metais, editors, *Proceedings of the 10th International Conference on Applications of Natural Language to Information Systems (NLDB)*, volume 3513 of *Lecture Notes in Computer Science*, pages 227–238, Alicante, Spain, JUN 2005. Springer.
2. Z. Harris. Distributional structure. In J. Katz, editor, *The Philosophy of Linguistics*, pages 26–47, New York, 1985. Oxford University Press.
3. P. Jaccard. The distribution of flora in the alpine zone. 11:37–50, 1912.
4. B. Patwardhan and Pedersen. Using measures of semantic relatedness for word sense disambiguation. In *Proceedings of the Fourth International Conference on Intelligent Text Processing and Computational Linguistics*, pages 241–257, FEB 2003.
5. J. R. Quinlan. *C4.5 Programs for Machine Learning*. Morgan Kaufmann, California, 1993.
6. O. Svab, S. Vojtech, P. Berka, D. Rak, and P. Tomasek. Ontofarm: Towards an experimental collection of parallel ontologies. In *Poster Proceedings of the International Semantic Web Conference 2005*, 2005.
7. J. Völker, D. Vrandecic, Y. Sure, and A. Hotho. Learning disjointness. In E. Franconi, M. Kifer, and W. May, editors, *Proceedings of the 4th European Semantic Web Conference (ESWC'07)*, volume volume 4519 of *Lecture Notes in Computer Science*, pages 175–189. Springer, JUN 2007.