
Ontology Learning

Philipp Cimiano¹, Alexander Mädche², Steffen Staab³, and Johanna Völker¹

¹ Institute AIFB, University of Karlsruhe, Karlsruhe, Germany,
cimiano.voelker@aifb.uni-karlsruhe.de

² SAP AG, Walldorf, Germany, alexander.maedche@sap.com

³ ISWEB Group, University of Koblenz-Landau, Koblenz, Germany,
staab@uni-koblenz.de

Summary. Ontology learning techniques serve the purpose of supporting an ontology engineer in the task of creating and maintaining an ontology. In this chapter, we present a comprehensive and concise introduction to the field of ontology learning. We present a generic architecture for ontology learning systems and discuss its main components. In addition, we introduce the main problems and challenges addressed in the field and give an overview of the most important methods applied. We conclude with a brief discussion of advanced issues which pose interesting challenges to the state-of-the-art.

1 Introduction

Ontology engineering is slowly changing its status from an art to a science and in fact, during the last decade, several ontology engineering methodologies (see chapters “Ontology Engineering Methodology” and “Ontology Engineering and Evolution in a Distributed World Using DILIGENT”) have been examined. But still, as pointed out in chapter “Exploring the Economical Aspects of Ontology Engineering”, the task of engineering an ontology remains a resource-intensive and costly task. Therefore, techniques which support the task of ontology engineering are necessary to reduce the costs associated with the engineering and maintenance of ontologies. As data in various forms (textual, structured, visual, etc.) is massively available, many researchers have developed methods aiming at supporting the engineering of ontologies by data mining techniques, thus deriving meaningful relations which can support an ontology engineer in the task of modeling a domain. Such data-driven techniques supporting the task of engineering ontologies have become to be known as *ontology learning*. Ontology learning has indeed the potential to reduce the cost of creating and, most importantly, maintaining an ontology. This is the reason why a plethora of ontology learning frameworks have been developed in the last years and integrated with standard ontology engineering tools. Text-ToOnto [55], for example, was originally integrated into the KAON ontology

engineering environment [27], OntoLT [11] was integrated with Protégé and Text2Onto [22] has been recently integrated with the NeOn Toolkit.¹

There are three kinds of data to which ontology learning techniques can be applied: structured (such as databases), semi-structured (HTML or XML, for example) as well as unstructured (e.g., textual) documents. The methods applied are obviously dependent on the type of data used. While highly structured data as found in databases facilitates the application of pure machine learning techniques such as Inductive Logic Programming (ILP), semi-structured and unstructured data requires some preprocessing, which is typically performed by natural language processing methods.

Ontology Learning builds upon well-established techniques from a variety of disciplines, including natural language processing, machine learning, knowledge acquisition and ontology engineering. Because the fully automatic acquisition of knowledge by machines remains in the distant future, the overall process is considered to be semi-automatic with human intervention.

Organization

This chapter is organized as follows: Sect. 2 introduces a generic architecture for ontology learning and its relevant components. In Sect. 3 we introduce various complementary basic ontology learning algorithms that may serve as a basis for ontology learning. Section 4 describes ontology learning frameworks and tools which have been implemented in the past. In particular, we also discuss our own system, Text2Onto, the successor of the TextToOnto framework [55].

2 An Architecture and Process Model for Ontology Learning

The purpose of this section is to introduce a generic ontology learning architecture and its major components. The architecture is graphically depicted in Fig. 1. In general, the process of ontology learning does not differ substantially from a classical data mining process (e.g., [15]) with the phases of *business and data understanding*, *data preparation*, *modeling*, *evaluation* and *deployment*. The key components of an architecture for ontology learning are the following: an *ontology management*, a *coordination*, a *resource processing* and an *algorithm library component*. We describe these components in more detail in the following.

2.1 Ontology Management Component

The ontology engineer uses the ontology management component to manipulate ontologies. Ontology management tools typically facilitate the import,

¹ <http://www.neon-toolkit.org>

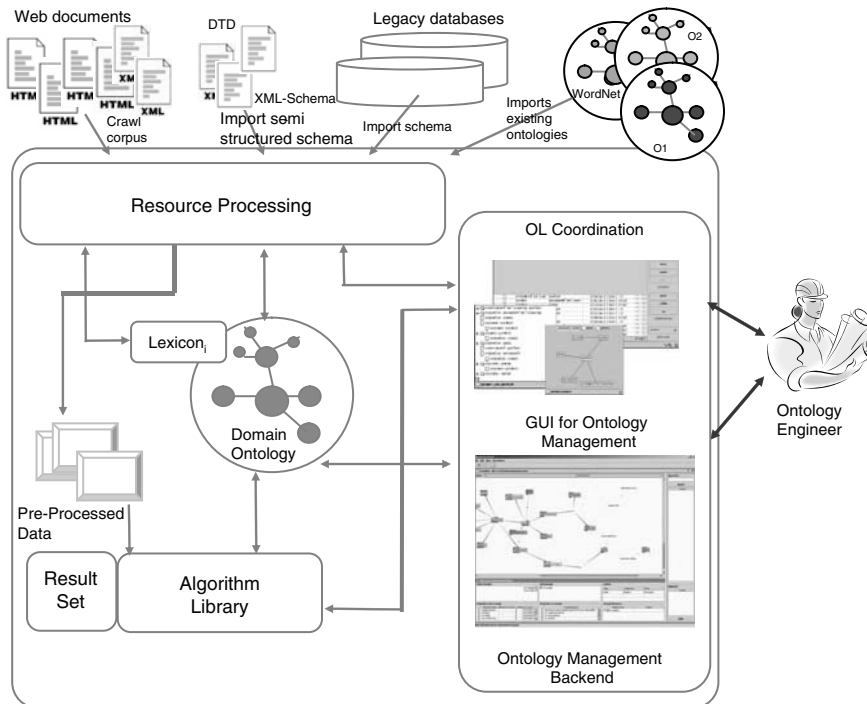


Fig. 1. Ontology learning conceptual architecture

browsing, modification, versioning as well as evolution of ontologies. However, the main purpose of the ontology management component in the context of ontology learning is to provide an interface between the ontology and the learning algorithms. When learning new concepts, relations or axioms, the learning algorithms should add them into the ontology model accessing the Application Programming Interface (API) of the ontology management component. Thus, the ontology management API should at least contain methods for creating new concepts, relations, axioms, individuals, etc. Most available APIs indeed fulfill this requirement. Further important functionalities for ontology learning are: *evolution*, *reasoning* and *evaluation*. Techniques for ontology evolution as presented in [52] or [35] are very important for ontology learning as it is an inherently dynamic process. As the underlying data changes, the learned ontology should change as well and this requires not only incremental ontology learning algorithms, but also some support for ontology evolution at the ontology management level. Reasoning and evaluation play a crucial role in guiding the ontology learning process. In case the ontology learning system faces several alternatives, it should definitely choose that alternative which preserves the consistency of the underlying ontology [36] or the one which maximizes certain quality criteria.

2.2 Coordination Component

The ontology engineer uses this component to interact with the ontology learning components for resource processing as well as with the algorithm library. Comprehensive user interfaces should support the user in selecting relevant input data that are exploited in the further discovery process. Using the coordination component, the ontology engineer also chooses among a set of available resource processing methods and among a set of algorithms available in the algorithm library. A central task of the coordination component is further to sequentially arrange and apply the algorithms selected by the user, passing the results to each other.

2.3 Resource Processing Component

This component contains a wide range of techniques for *discovering, importing, analyzing and transforming* relevant input data. An important sub-component is the natural language processing system. The general task of the resource processing component is to generate a pre-processed data set as input for the algorithm library component.

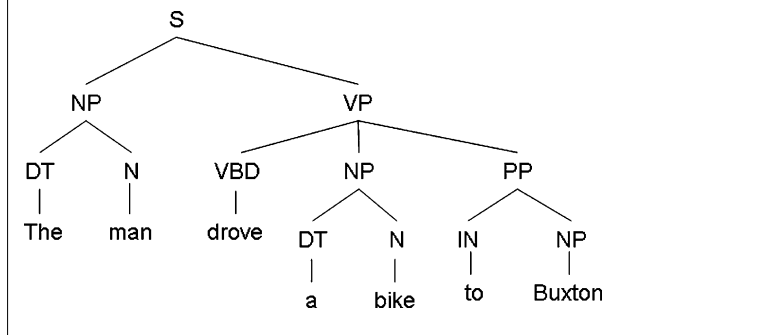
Resource processing strategies differ depending on the type of input data made available. Semi-structured documents, like dictionaries, may be transformed into a predefined relational structure. HTML documents can be indexed and reduced to free text. For processing free text, the system must have access to language-specific natural language processing systems. Nowadays, off-the-shelf frameworks such as GATE [24] already provide most of the functionality needed by ontology learning systems. The needed NLP components could be the following ones:

- A *tokenizer* and a *sentence splitter* to detect sentence and word boundaries.
- A *morphological analyser*. For some languages a lemmatizer reducing words to their base form might suffice, whereas for languages with a richer morphology (e.g., German) a component for structuring a word into its components (lemma, prefix, affix, etc.) will be necessary. For most machine learning-based algorithms a simple stemming of the word might be sufficient (compare [60]).
- A *part-of-speech (POS) tagger* to annotate each word with its syntactic category in context, thus determining whether it is a noun, a verb, an adjective, etc. An example for a POS tagger is the TreeTagger [63].
- *Regular expression matching* allowing to define regular expressions and match these in the text. This functionality is provided for example by GATE's Java Annotation Pattern Engine (JAPE).
- A *chunker* in order to identify larger syntactic constituents in a sentence. Chunkers are also called *partial parsers*. An example of a publicly available chunker is Steven Abney's CASS [1].
- A *syntactic parser* determining the full syntactic structure of a sentence might be needed for some ontology learning algorithms (compare [20]).

Example 1. Given a sentence such as *The man drove a bike to Buxton.* we would for example yield the following tokenization, lemmatization as well as result of the POS tagging:

Tokens The man drove the bike to Buxton.
 Lemma The man drive the bike to Buxton.
 POS DT NN VBD DT NN IN NP.

where we use the Penn Treebank² tagset in which DT stands for a determiner, NN for a singular noun, IN for a preposition, NP for a proper noun and VBD for a past tense verb. The parse tree for the above sentence looks as follows:



2.4 Algorithm Library Component

This component acts as the algorithmic backbone of the framework. A number of algorithms are provided for the extraction and maintenance of the ontology modeling primitives contained in the ontology model. Thus, the algorithm library contains the actual algorithms applied to learning. In particular, the algorithm library consists mainly of machine learning algorithms and versions of these customized for the purpose of ontology learning. In particular, machine learning algorithms typically contained in the library are depicted in Table 1.

Most of these machine learning algorithms can be obtained off-the-shelf in various versions from standard machine learning frameworks such as WEKA [76]. Additionally, the library should also contain a comprehensive number of implemented distance or similarity measures such as Jaccard, Dice, the cosine measure, the Kullback–Leibler divergence, etc. (compare [49]) to support semantic clustering. In addition, the algorithm library could also contain traditional measures for discovering collocations between words known from computational linguistics research (e.g., [43]). In order to be able to combine the extraction results of different learning algorithms, it is necessary to standardize the output in a common way. In general, a common result structure for all learning methods is needed. In the Text2Onto system [22],

² See <http://www.cis.upenn.edu/~treebank>

Table 1. Typical machine learning algorithms in the algorithm library

Algorithm	Generic use	Use in ontology learning
Association rule discovery (e.g., [2])	Discovery of “interesting” transactions in itemsets (e.g., customer data)	Discovery of interesting associations between words
(Hierarchical) Clustering	Discovery of groups in data (unsupervised)	Clustering of words
Classification (e.g., SVMs, Naive Bayes, kNN, etc.)	Prediction (supervised)	Classification of new concepts into an existing hierarchy
Inductive logic programming ([48])	Induction of rules from data (supervised)	Discovery of new concepts from extensional data
Conceptual clustering (e.g., FCA – see chapter “Formal Concept Analysis”)	Concept discovery (extension and intension)	Learning concepts and concept hierarchies

for example, there is a blackboard-style result structure – the POM (Possible Ontologies Model) – where all algorithms can update their results.

3 Ontology Learning Algorithms

The various tasks relevant in ontology learning have been previously organized in a layer diagram showing the conceptual dependencies between different tasks. This *ontology learning layer cake* was introduced in [18] and is shown in Fig. 2. It clearly focuses on learning the TBox part of an ontology. With respect to information extraction techniques to populate the ABox of an ontology, the interested reader is referred to chapter “Information Extraction”. The layers build upon each other in the sense that results of tasks at lower layers typically serve as input for the higher layers. For example, in order to extract relations between concepts, we should consider the underlying hierarchy to identify the right level of generalization for the domain and range of the relation. The two bottom layers of the layer cake correspond to the lexical level of ontology learning. The task in this part of the layer is to detect the relevant terminology as well as groups of synonymous terms, respectively. The extracted terms and synonym groups can then form the basis for the formation of concepts. Concepts differ from terms in that they are ontological entities and thus abstractions of human thought in the sense of Ganter and Wille [32]. According to our formalization, concepts are triples $c := \langle i(c), \llbracket c \rrbracket, Ref_c \rangle$ consisting of an intensional description $i(c)$, an extension $\llbracket c \rrbracket$ and a reference function Ref_c representing how the concept is symbolically realized in a text corpus, an image, etc. (see [10]). At higher levels of the layer cake, we find the layers corresponding to the tasks of learning a concept hierarchy, relations, a relation hierarchy as well as deriving arbitrary rules and axioms. The top two

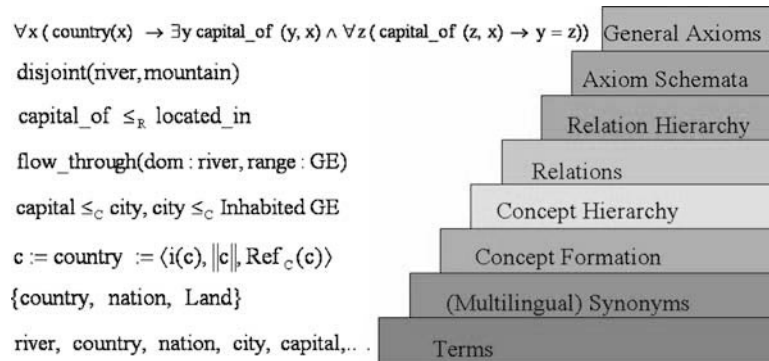


Fig. 2. Ontology learning layer cake from [18]

layers correspond certainly to the most challenging task as in principle there is no limit on the type and complexity of axioms and rules to be learned. In practice, however, as we commit to a specific knowledge representation language – the Web Ontology Language (OWL) for example (see chapter “Web Ontology Language: OWL”) – the types of axioms that are allowed are more restricted. In what follows, we discuss the various tasks layer by layer and point the reader to relevant approaches in the literature of the field.

3.1 Term Extraction

The task at the lexical layers is to extract terms and arrange these into groups of synonymous words. A simple technique for extracting relevant terms that may indicate concepts is counting frequencies of terms in a given set of (linguistically preprocessed) documents, the corpus \mathcal{D} . In general this approach is based on the assumption that a frequent term in a set of domain-specific texts indicates the occurrence of a relevant concept. Research in information retrieval has shown that there are more effective methods of term weighting than simple counting of frequencies. Weighting measures well-known from information retrieval such as *tf.idf* (see [6]) might also be applied here.

Further, the computational linguistics community has proposed a wide range of more sophisticated techniques for term extraction. An interesting measure is the C-value/NC-value measure presented in [31] which does not only take into account the frequency of terms but also the fact that terms can be nested into each other. Further, the approach also takes into account contextual clues which are strong indicators of the “*termhood*” of some sequence of words. Overall, while the field of term extraction seems quite mature and a plethora of techniques have been suggested and examined, there is not yet a clear understanding of which measures work best for which purpose. Clearly, specific domains, such as genomics, medicine or E-commerce need corresponding adaptations of tools and methods with respect to their specific characteristics.

3.2 Synonym Extraction

In order to extract synonyms, most approaches rely on the *distributional hypothesis* claiming that words are semantically similar to the extent to which they share syntactic contexts [39]. This hypothesis is also in line with Firth’s well known statement that “*you shall know a word by the company it keeps*” [30]. For each word w , a distributional representation is computed and represented as a vector \mathbf{v}_w on the basis of the word’s context. Features used to represent a word are typically other words appearing within a certain window from the target word, syntactic information and dependencies. The similarity in vector space between different word vectors can then be computed and highly similar words can be regarded as synonyms.

Example 2. Assuming that we parse a text corpus and identify, for each noun, the verbs for which it appears at the object position, we can construct a matrix as follows:

	Book _{obj}	Rent _{obj}	Drive _{obj}	Ride _{obj}	Join _{obj}
Hotel	x				
Apartment	x	x			
Car	x	x	x		
Bike	x	x	x	x	
Excursion	x				x
Trip	x				x

Each row represents the context of a word, while each column corresponds to one dimension of the context representation, in our case the different verbs that the nouns appear at the object position. Assuming the representation as binary vectors shown in the matrix above, we can for example calculate the similarity between the different terms using the Jaccard coefficient which compares the sets A and B of the non-negative dimensions of the vector representations two words a and b: $Jaccard := \frac{|A \cap B|}{|A \cup B|}$. The resulting similarities are thus:

	Hotel	Apartment	Car	Bike	Excursion	Trip
Hotel	1.0	0.5	0.33	0.25	0.5	0.5
Apartment		1.0	0.66	0.5	0.33	0.33
Car			1.0	0.75	0.25	0.25
Bike				1.0	0.2	0.2
Excursion					1.0	1.0
Trip						1.0

Important approaches along these lines include the work of Grefenstette [33] as well as Lin et al. [50]. Some researchers have also combined different similarity extractors using ensemble methods [25]. Other techniques for extracting synonyms include the application of statistical methods to the Web (cf. [7, 69]) or the calculation of semantic relatedness with respect to

a taxonomy or semantic network such as WordNet (compare [61]) or more recently also the Wikipedia categories (see [67]). WordNet [29] is a lexical database organizing words in terms of synonym sets (synsets) and providing lexical relations between these synsets, i.e., hypernymy/hyponymy (“is a kind of”) as well as meronymy/holonymy (“part of”) relations. In addition, WordNet provides *glosses*, which are natural language definitions of the synsets. Turney [69] for example relies on the well-known Pointwise Mutual Information (PMI) measure to extract synonyms. The pointwise mutual information of two events x and y is defined as:

$$PMI(x, y) := \log_2 \frac{P(x, y)}{P(x) P(y)}$$

where $P(x, y)$ is the probability for a joint occurrence of x and y and $P(x)$ is the probability for the event x . The PMI is thus in essence the (logarithmic) ratio of the joint probability and the probability under the assumption of independence. In fact, if $P(x, y) \leq P(x)P(y)$, we will have a negative (or zero) value for the PMI, while in case $P(x, y) > P(x)P(y)$, we will have a positive PMI value. The PMI can be calculated using *Google* and counting hits as follows:

$$PMI_{Web}(x, y) := \log_2 \frac{Hits(x \text{ AND } y) \text{ MaxPages}}{Hits(x) Hits(y)}$$

where *MaxPages* is an approximation for the maximum number of English web pages. This measure can thus be used to calculate the statistical dependence of two words on the Web. If they are highly dependent, we can assume they are synonyms or at least highly semantically related. This approach to discover synonyms has been successfully applied to the TOEFL test (see [69]).

3.3 Concept Learning

In this section we focus on approaches inducing concepts by clearly defining the intension of the concept. We will distinguish the following three paradigms:

- Conceptual clustering
- Linguistic analysis
- Inductive methods

Conceptual Clustering

Conceptual clustering approaches such as Formal Concept Analysis ([32], chapter “Formal Concept Analysis”) have been applied to form concepts and to order them hierarchically at the same time. Conceptual clustering approaches typically induce an intensional description for each concept in terms of the attributes that it shares with other concepts as well as those that distinguish it from other concepts.

Linguistic Analysis

Linguistic analysis techniques can be applied to derive an intensional description of a concept in the form of a natural language description. The approach of Velardi et al. [70] for example relies on WordNet to compositionally interpret a compound term and as a byproduct produce a description on the basis of the WordNet descriptions of the single terms constituting the compound [70]. The definition of the term *knowledge management practices*: “a kind of practice, knowledge of how something is customarily done, relating to the knowledge of management, the process of capturing value, knowledge and understanding of corporate information, using IT systems, in order to maintain, re-use and de-ploy that knowledge.” is compositionally determined on the basis of the definitions of *knowledge management*³ and *practice*.⁴ For this purpose, disambiguation with respect to the different senses of a word with respect to its several meanings in a lexical database (such as WordNet) is required. Further, a set of rules is specified which drive the above compositional generation of definitions.

Finally, given a populated knowledge base, approaches based on inductive learning such as Inductive Logic Programming can be applied to derive rules describing a group of instances intentionally. Such an approach can for example be used to reorganize a taxonomy or to discover gaps in conceptual definitions (compare [51]).

3.4 Concept Hierarchy

Different methods have been applied to learn taxonomic relations from texts. In what follows we briefly discuss approaches based on matching *lexico-syntactic patterns*, *clustering*, *phrase analysis* as well as *classification*.

Lexico-Syntactic Patterns

In the 1980s, people working on extracting knowledge from machine readable dictionaries already realized that regularities in dictionary entries could be exploited to define patterns to automatically extract hyponym/hypernym and other lexical relations from dictionaries (compare [3, 4, 13]). This early work was continued later in the context of the ACQUILEX project (e.g., [23]).

In her seminal work, Hearst [40] proposed the application of so-called lexico-syntactic patterns to the task of automatically learning hyponym relations from corpora. In particular, Hearst defined a collection of patterns

³ Knowledge management: the process of capturing value, knowledge and understanding of corporate information, using IT systems, in order to maintain, re-use and re-deploy that knowledge.

⁴ Practice: knowledge of how something is customarily done.

indicating hyponymy relations. An example of a pattern used by Hearst is the following:

such NP_0 as NP_1, \dots, NP_{n-1} (or|and) other NP_n

where NP_i stands for a noun phrase. If such a pattern is matched in a text, according to Hearst we could derive that for all $0 < i \leq n$ *hyponym*(NP_i, NP_0).⁵ For example, from the sentence *Such injuries as bruises, wounds and broken bones...*, we could derive the relations: *hyponym*(*bruise, injury*), *hyponym*(*wound, injury*) and *hyponym*(*broken bone, injury*).

The patterns used by Hearst are the following:

- Hearst1: NP_{hyper} such as $\{NP_{hyppo},\}^* \{(and | or)\} NP_{hyppo}$
- Hearst2: such NP_{hyper} as $\{NP_{hyppo},\}^* \{(and | or)\} NP_{hyppo}$
- Hearst3: $NP_{hyppo} \{,NP\}^* \{,\}$ or other NP_{hyper}
- Hearst4: $NP_{hyppo} \{,NP\}^* \{,\}$ and other NP_{hyper}
- Hearst5: NP_{hyper} including $\{NP_{hyppo},\}^* NP_{hyppo} \{(and | or)\} NP_{hyppo}$
- Hearst6: NP_{hyper} especially $\{NP_{hyppo},\}^* \{(and|or)\} NP_{hyppo}$

Overall, lexico-syntactic patterns have been shown to yield a reasonable precision for extracting *is-a* as well as *part-of* relations (e.g., [14, 16, 59]).

Clustering

Clustering can be defined as the process of organizing objects into groups whose members are similar in some way based on a certain representation, typically in the form of vectors (see [46]). In general, there are three major styles of clustering:

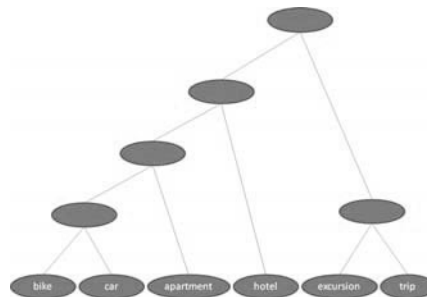
1. *Agglomerative*: In the initialization phase, each term is defined to constitute a cluster of its own. In the growing phase, larger clusters are iteratively generated by merging the most similar/least dissimilar ones until some stopping criterion is reached. Examples of uses of agglomerative clustering techniques in the literature are [8, 20, 28].
2. *Divisive*: In the initialization phase, the set of all terms constitutes a cluster. In the refinement phase, smaller clusters are (iteratively) generated by splitting the largest cluster or the least homogeneous cluster into several subclusters. Examples for divisive clustering can be found in [20, 58].
Both agglomerative and divisive clustering techniques are used to produce hierarchical descriptions of terms. Both rely on notions of (dis-)similarity, for which a range of measures exist (e.g., Jaccard, Kullback–Leibler divergence, L1-norm, cosine; cf. [49]).
3. *Conceptual*: Conceptual clustering builds a lattice of terms by investigating the exact overlap of descriptive attributes between two represented terms. In the worst case, the complexity of the resulting concept lattice is exponential in n . Thus, people either just compute a sublattice [68] or rely

⁵ From a linguistic point of view, a term t_1 is a hyponym of a term t_2 if we can say *a t_1 is a kind of t_2* . Correspondingly, t_2 is then a hypernym of t_1 .

on certain heuristics to explore and/or prune the lattice. Examples of applications of conceptual clustering techniques to ontology learning can be found in [20, 38].

Either way one may construct a hierarchy of term clusters for detailed inspection by the ontology engineer.

Example 3. Using hierarchical agglomerative clustering, we can build a cluster tree for the objects in Example 2. Let us assume we are using *single linkage* as measure of the similarity between clusters. First, we cluster *excursion* and *trip* as they have a similarity of 1.0. We then cluster *bike* and *car* as this is the next pair with the highest degree of similarity. We then build a cluster consisting of *bike*, *car* and *apartment*. Next, we either join the latter cluster with *hotel* or build a cluster between *hotel* and the already created cluster consisting of *excursion* and *trip*. Assuming that we traverse the similarity matrix from the upper left corner to the lower right one, we can add *hotel* to the cluster consisting of *bike*, *car* and *apartment*. At the top level we then join the clusters $\{\textit{hotel}, \textit{apartment}, \textit{bike}, \textit{car}\}$ and $\{\textit{excursion}, \textit{trip}\}$ producing a universal cluster containing all elements. The corresponding cluster tree would then look as follows:



Phrase Analysis

Some approaches rely on the fact that the internal structure of noun phrases can be used to discover taxonomic relations (compare [11, 62], but also [21]). In essence, these methods build on the heuristic that additional modifiers (adjectival or nominal) added to the front of a noun typically define a subclass of the class denoted by the noun. That means, for example, that *focal epilepsy* is interpreted as a subclass of *epilepsy*. This is in essence the approach implemented in the OntoLT system (see below and [11]). Sanchez and Moreno [62] exploit this heuristic in a web setting to find terms which occur to the left

of a term to be refined into subclasses. A measure inspired in the Pointwise Mutual Information (PMI) is used to assess the degree of correlation between the term in question and the modifier to the left.

Classification

When a substantial hierarchy is already given, e.g., by basic level categories from a general resource like WordNet [29], one may rather decide to refine the taxonomy by classifying new relevant terms into the given concept hierarchy. The distributional representation described above is then used to learn a classifier from a training corpus and the set of predefined concepts with their lexical entries. Afterwards, one may construct the distributional representations of relevant, unclassified terms and let the learned classifier propose a node to which to classify the new term. While many researchers have considered lexical databases such as WordNet to test such algorithms (e.g., [41] and [75]), other researchers have indeed considered domain-specific ontologies see, e.g., the work of Pekar and Staab [57]). Pekar and Staab have in particular considered different algorithms to classify a new term into an existing concept hierarchy without testing all concepts, e.g., by exploiting the hierarchical structure using tree-ascending or tree-descending algorithms.

3.5 Relations

In order to discover arbitrary relations between words, different techniques from the machine learning and statistical natural language processing community have found application in ontology learning. In order to discover “anonymous” associations between words, one can look for a strong co-occurrence between words within a certain boundary, i.e., a window of words, a sentence or a paragraph. Mädche et al. [53] apply the well-known association discovery algorithm and represent co-occurrences of words within a sentence as transactions. This representation allows to calculate the support and confidence for binary transactions and thus to detect anonymous binary associations between words.

In the computational linguistics community, the task of discovering strong associations between words is typically called *collocation discovery*. In essence, the idea is to discover words which co-occur beyond chance in a statistically significant manner. Statistical significance is typically checked using some test such as the Student’s t-test or the χ^2 -test (compare [43,56]). Other researchers have aimed at learning labeled relations by relying on linguistic predicate-argument dependencies. Typically, verb structures are considered for this purpose (compare [17,19,64]). When learning relations, a crucial issue is to find the right level of abstraction with respect to the concept hierarchy for the domain and range of the relation in question. This issue can be addressed in different ways. While Mädche et al. [53] incorporate the concept hierarchy into the association discovery process, Ciaramita et al. [17] as well as Cimiano et al. [19] formulate this as a problem of generalizing along a hierarchy as long as the statistical significance does not diminish.

3.6 Axioms and Rules

Ontology learning approaches so far have focused on the acquisition of rather simple taxonomic hierarchies, properties as well as lexical and assertional knowledge. However, the success of OWL which allows for modeling far more expressive axiomatizations has led to some advances in the direction of learning complex ontologies and rules.

Völker et al. [71] propose an approach for generating formal class descriptions from natural language definitions extracted, e.g., from online glossaries and encyclopedias. The implementation of this approach is essentially based on a syntactic transformation of natural language definitions into OWL DL axioms in line with previous work on lexico-syntactic patterns (cf. Sect. 3.4) and lexical entailment.

One of the first methods for learning disjointness axioms relies on a statistical analysis of enumerations which has been implemented as part of the Text2Onto framework [37]. Völker et al. [73] developed a supervised learning approach based on an extended set of methods yielding both lexical and logical evidence for or against disjointness.

3.7 Pruning/Domain Adaptation

One relatively straightforward approach towards generating an appropriate domain ontology given a corpus is to prune an existing general ontology. Along these lines, Buitelaar et al. [12] propose a method by which WordNet synsets can be ranked by relevance with respect to the corpus in question on the basis of a frequency-based measure. The techniques used to prune WordNet, thus adapting it to a certain domain and corpus, can be also applied to prune a given ontology. Kietz et al. [47] for example present a method which additionally uses a general corpus as contrast. They only select a concept as relevant in case it appears a factor c times more relevant in the domain-specific than in the general corpus. Hereby, c is a user-specified constant and the relevance measure used is tf.idf.

4 Ontology Learning Systems

In the last years, many different tools and frameworks for ontology learning have emerged. Needless to say that it is out of the scope of this chapter to discuss them all. Instead, we will provide a rather subjective snapshot of the current tool landscape. Some well-known and frequently cited tools are for example: OntoLearn [70], OntoLT [11], Terminae [5] as well as TextToOnto [55] and its successor Text2Onto [22]. All these tools implement various and different methods, such that a detailed discussion and comparison is out of the scope

of this chapter. OntoLearn for example integrates a word sense disambiguation component to derive intensional descriptions of complex domain-specific terms, which are assumed to denote concepts, on the basis of WordNet glosses (compare Sect. 3.3). In this sense, OntoLearn also induces intensionally defined domain concepts and ingeniously exploits the knowledge available in general resources for a specific domain. OntoLT, which is available as a plugin to the Protégé ontology editor [34], allows for term extraction using various measures such as tf.idf and extraction of taxonomic relations relying on interpreting modifiers (nominal or adjectival) as introducing subclasses (compare Sect. 3.4).

TextToOnto [55] is a framework containing various tools for ontology learning. It includes standard term extraction using a number of different measures, the algorithm for mining relations based on association rules described in [53] (see Sect. 3.5) as well as hierarchical clustering algorithms based on Formal Concept Analysis (compare Sect. 3.4). Its successor, Text2Onto, besides implementing most of the algorithms available also in TextToOnto, abstracts from a specific knowledge representation language and stores the learned ontology primitives in the form of a meta-model called *Possible Ontologies Model* (POM), which can then be translated to any reasonably expressive knowledge representation language, in particular to OWL and RDFS. On the other hand, it implements a framework for data-driven and incremental learning in a sense that changes in the underlying corpus are propagated to the algorithms, thus leading to explicit changes to the POM. The advantage is that these changes can be easily traced back to the original corpus changes, which gives more control to the ontology engineer.

5 Advanced Issues

In this section, we briefly discuss some advanced and open issues in ontology learning that are still under research. This section should help newcomers to get a feeling for the open questions and allow for a quicker entry into the field.

5.1 Methodology

Certainly, besides providing tool support for ontology learning methods, it is crucial to define how ontology learning methods can be integrated into the process of engineering an ontology. Blueprints in this direction can be found in the work of Simperl et al. [65] as well as Aussenec-Gilles et al. [5]. Simperl et al. for example provide a methodology defining the necessary activities and roles for ontology engineering supported by ontology learning methods. In particular, they argue that without a clear methodology to be followed by an ontology engineering project, ontology learning techniques can not reasonably support ontology engineering activities. Aussenec-Gilles et al. [5] have also conducted research on methodological issues in the context of their Terminae

method. In particular, they have argued that knowledge models, in particular ontologies, need to be anchored in language. Therefore, they emphasize the role of language in the process of ontology engineering. In general, though first attempts have been provided, there is still much further work to do in order to clarify the benefits and drawbacks of different methodologies for integrating ontology learning into available ontology engineering methodologies. As argued by Simperl et al., ontology learning tools need to improve on their usability and intuitiveness in order to be useful for the purpose of ontology engineering.

5.2 Evaluation

A crucial part of ontology learning is to evaluate how good the learned ontologies actually are. Such an evaluation can in turn guide and control the ontology learning process in the search towards an “optimal” ontology. However, the evaluation of ontology learning tools is a quite delicate issue as it is not clear what one could compare to. The critical issue in many cases is to define a *gold standard* which we can regard as ground truth an one can compare with (see [26]). However, it is well known that there is no ground truth for ontologies as different people will surely come up with very different ontologies when asked to model a certain domain (see, e.g., the experiments described in [54]).

Other approaches aim at approximating the appropriateness of some ontology by other means. Brewster et al. [9], for example, try to measure the “corpus fit” of the ontology by considering the frequency with which the terms in the ontology appear in the corpus. A completely different way to check the quality of an ontology is pursued by the AEON framework [72], that aims to automatize the application of the OntoClean methodology (see chapter “An Overview of OntoClean”), hence ensuring the formal consistency of an ontology. Finally, an integration of ontology learning and evaluation is proposed by Haase et al. [37]. They describe an approach to exploiting contextual information such as OntoClean meta-properties, or confidence and relevance values for resolving logical inconsistencies in learned ontologies, and to optimize the outcome of the ontology learning process.

5.3 Expressivity

Many people argue that the main benefits of using ontologies for knowledge modeling become most evident in reasoning-based applications. Inferring new knowledge and drawing conclusions beyond explicit assertions is an important aspect of “intelligent” applications. However, the power of reasoning largely depends on the expressivity of the underlying knowledge representation formalism and its instantiation by means of a concrete ontology.

The vast majority of today's lexical ontology learning focuses on the extraction of simple class descriptions and axioms, i.e., atomic concepts, subsumption and object properties, as well as ABox statements expressing concept or property instantiation. The expressivity of ontologies generated by lexical approaches, e.g., based on natural language processing techniques is mostly restricted to \mathcal{ALC} (Attributive Language with Complements) or similar DL fragments such as $\mathcal{AL-log}$. These rather simple, often informal ontologies have proven to be useful for many applications, or as Jim Hendler has put it "A little semantics goes a long way" (see [42]). But semantic applications relying on reasoning over very complex domains such as bioinformatics or medicine require more precise and accurate knowledge representation.

Learning more expressive ontologies greatly facilitates the acquisition and evaluation of complex domain knowledge. But it also brings new challenges, e.g., with respect to logical inconsistencies that may arise as soon as any kind of negation or cardinality constraints are introduced into learned ontologies [45]. Methods for debugging, consistent ontology evolution, or inconsistency reasoning will be required to face these challenges.

Finally, a tighter integration of lexical and logical ontology learning approaches will be required in order to prevent problems resulting from different semantic interpretations, e.g., of lexical and ontological relations (see discussion in [71]). A first approach in this line is the RELExO framework by Völker and Rudolph [74], which combines a lexical approach to the acquisition of complex class descriptions with the FCA-based technique of relational exploration.

5.4 Combination of Evidence

As it is very unlikely that we will be able to derive high-quality ontologies from one single source of evidence and using one single approach, a few researchers have addressed the challenge of learning ontologies by considering multiple sources of evidence. Cimiano et al. [21] have for example presented a classification-based approach in which a classifier is trained with features derived from various approaches and data sources. The approach is shown to outperform any of the single algorithms considered. Snow et al. [66] have phrased the problem in probabilistic terms and considered the task of adding new concepts (synsets) to the WordNet taxonomy. These approaches have so far focused only on learning taxonomic relations with the notable exception of initial approaches to the automatic generation of disjointness axioms [73]. In general, there is a lot of further research needed in this direction.

5.5 Dynamics and Evolution

Most ontology learning approaches assume that there is one static corpus from which to learn. However, collecting such a corpus can sometimes be a non-trivial task. Currently, some researchers are attempting to frame ontology

learning as the task of keeping an *equilibrium* between a (growing) corpus, a (growing) set of extracted ontological primitives and a (changing) set of extraction patterns. While it seems very hard to define such an equilibrium in such a way that certain actions are triggered towards restoring it, first attempts in this direction can be found in the work of Iria et al. [44]. Further, as the underlying corpus can and will evolve, it is an important question to explicitly track changes in the ontology model with respect to changes in the corpus, thus enhancing the transparency of the ontology learning process and allowing human inspection. From a performance point of view, an incremental approach to ontology learning has moreover the benefit that the whole corpus will not need to be processed each time it changes. A first approach in this direction has been implemented in the Text2Onto system (compare [22]).

6 Conclusion

Ontology learning is a challenging and exciting research field at the intersection of machine learning, data and text mining, natural language processing and knowledge representation. While fully automatic knowledge acquisition techniques are not yet feasible (and possibly will nor should ever be), ontology learning techniques have a high potential to support ontology engineering activities. In fact, according to our view, ontology engineering can not be considered without the automatic or semi-automatic support of ontology learning methods. Future work should and will surely aim at developing a new generation of intuitive ontology learning tools which are able to learn expressive ontologies, but at the same time hide their internal complexity from the user. These new generation of tools should feature intuitive user interfaces as well as smoothly integrate into existing methodologies for ontology engineering.

References

1. S. Abney. Partial parsing via finite-state cascades. In *Proceedings of the ESS-LLI '96 Robust Parsing Workshop*, pages 8–15, 1996.
2. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on Very Large Databases (VLDB)*, 1994.
3. H. Alshawi. Processing dictionary definitions with phrasal pattern hierarchies. *Computational Linguistics*, 13(3–4):195–202, 1987. Special Issue of the Lexicon.
4. R. A. Amsler. A taxonomy for english nouns and verbs. In *Proceedings of the 19th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 133–138, 1981.
5. N. Aussenac-Gilles, S. Despres, and S. Szulman. The TERMINAE method and platform for ontology engineering from text. In P. Buitelaar and P. Cimiano, editors, *Bridging the Gap between Text and Knowledge: Selected Contributions to Ontology Learning and Population from Text*, volume 167 of *Frontiers in Artificial Intelligence*. IOS Press, 2007.

6. R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, 1999.
7. M. Baroni and S. Bisi. Using cooccurrence statistics & the web to discover synonyms in a technical language. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC)*, pages 1725–1728, 2004.
8. G. Bisson, C. Nédellec, and L. Ca namerio. Designing clustering methods for ontology building – The Mo’K workbench. In *Proceedings of the ECAI Ontology Learning Workshop*, pages 13–19, 2000.
9. C. Brewster, H. Alani, S. Dasmahapatra, and Y. Wilks. Data-driven ontology evaluation. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*, Lisbon, 2004. European Language Resources Association.
10. P. Buitelaar, T. Declerck, A. Frank, S. Racioppa, M. Kiesel, M. Sintek, R. Engel, M. Romanelli, D. Sonntag, B. Loos, V. Micelli, R. Porzel, and P. Cimiano. Linginfo: Design and applications of a model for the integration of linguistic information in ontologies. In *Proceedings of the OntoLex06 Workshop at LREC*, 2006.
11. P. Buitelaar, D. Olejnik, and M. Sintek. A Protégé plug-in for ontology extraction from text based on linguistic analysis. In *Proceedings of the 1st European Semantic Web Symposium (ESWS)*, pages 31–44, 2004.
12. P. Buitelaar and B. Sacaleanu. Ranking and selecting synsets by domain relevance. In *Proceedings of the NAACL Workshop on WordNet and Other Lexical Resources: Applications, Extensions and Customizations*, 2001.
13. N. Calzolari. Detecting patterns in a lexical data base. In *Proceedings of the 22nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 170–173, 1984.
14. S. Cederberg and D. Widdows. Using LSA and noun coordination information to improve the precision and recall of automatic hyponymy extraction. In *Conference on Natural Language Learning (CoNLL)*, pages 111–118, 2003.
15. P. Chapman, R. Kerber, J. Clinton, T. Khabaza, T. Reinartz, and R. Wirth. The CRISP-DM process model. Discussion Paper, March 1999.
16. E. Charniak and M. Berland. Finding parts in very large corpora. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 57–64, 1999.
17. M. Ciaramita, A. Gangemi, E. Ratsch, J. Šarić, and I. Rojas. Unsupervised learning of semantic relations between concepts of a molecular biology ontology. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 659–664, 2005.
18. P. Cimiano. *Ontology Learning and Population from Text: Algorithms, Evaluation and Applications*. Springer, 2006.
19. P. Cimiano, M. Hartung, and E. Ratsch. Finding the appropriate generalization level for binary ontological relations extracted from the Genia corpus. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, 2006.
20. P. Cimiano, A. Hotho, and S. Staab. Learning concept hierarchies from text corpora using formal concept analysis. *Journal of Artificial Intelligence Research (JAIR)*, 24:305–339, 2005.
21. P. Cimiano, L. Schmidt-Thieme, A. Pivk, and S. Staab. Learning taxonomic relations from heterogeneous evidence. In P. Buitelaar, P. Cimiano, and B. Magnini,

- editors, *Ontology Learning from Text: Methods, Applications and Evaluation*, number 123 in *Frontiers in Artificial Intelligence and Applications*, pages 59–73. IOS Press, 2005.
22. P. Cimiano and J. Völker. Text2onto – A framework for ontology learning and data-driven change discovery. In E. Métais, A. Montoyo, and R. Muñoz, 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, 2005.
 23. A. Copestake. An approach to building the hierarchical element of a lexical knowledge base from a machine readable dictionary. In *Proceedings of the 1st International Workshop on Inheritance in Natural Language Processing*, pages 19–29, 1990.
 24. H. Cunningham, K. Humphreys, R.J. Gaizauskas, and Y. Wilks. GATE – A general architecture for text engineering. In *Proceedings of Applied Natural Language Processing (ANLP)*, pages 29–30, 1997.
 25. J. Curran. Ensemble methods for automatic thesaurus construction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 222–229, 2002.
 26. K. Dellschaft and S. Staab. On how to perform a gold standard based evaluation of ontology learning. In *Proceedings of the International Semantic Web Conference*, pages 228–241, 2006.
 27. E. Bozsak et al. KAON – Towards a large scale Semantic Web. In *Proceedings of the Third International Conference on E-Commerce and Web Technologies (EC-Web)*. Springer Lecture Notes in Computer Science, 2002.
 28. D. Faure and C. Nédellec. A corpus-based conceptual clustering method for verb frames and ontology. In P. Velardi, editor, *Proceedings of the LREC Workshop on Adapting lexical and corpus resources to sublanguages and applications*, pages 5–12, 1998.
 29. C. Fellbaum. *WordNet, an electronic lexical database*. MIT Press, 1998.
 30. J. Firth. *A synopsis of linguistic theory 1930–1955*. Studies in Linguistic Analysis, Philological Society, Oxford. Longman, 1957.
 31. K. Frantzi and S. Ananiadou. The C-value/NC-value domain independent method for multi-word term extraction. *Journal of Natural Language Processing*, 6(3):145–179, 1999.
 32. B. Ganter and R. Wille. *Formal Concept Analysis – Mathematical Foundations*. Springer, 1999.
 33. G. Grefenstette. SEXTANT: Exploring unexplored contexts for semantic extraction from syntactic analysis. In *Meeting of the Association for Computational Linguistics*, pages 324–326, 1992.
 34. W. Grosso, H. Eriksson, R. Ferguson, J. Gennari, S. Tu, and M. Musen. Knowledge modelling at the millenium: The design and evolution of Protégé. In *Proceedings of the 12th International Workshop on Knowledge Acquisition, Modeling and Management (KAW'99)*, 1999.
 35. P. Haase and L. Stojanovic. Consistent evolution of owl ontologies. In A. Gomez-Perez and J. Euzenat, editors, *Proceedings of the Second European Semantic Web Conference*, volume 3532 of *LNCS*, pages 182–197, 2005.
 36. P. Haase and J. Völker. Dealing with uncertainty and inconsistency. In P. C. G. da Costa, K. B. Laskey, K. J. Laskey, and M. Pool, editors, *Proceedings of the Workshop on Uncertainty Reasoning for the Semantic Web (URSW)*, pages 45–55, 2005.

37. P. Haase and J. Völker. Ontology learning and reasoning – Dealing with uncertainty and inconsistency. In P. C. G. da Costa, K. B. Laskey, K. J. Laskey, and M. Pool, editors, *Proceedings of the Workshop on Uncertainty Reasoning for the Semantic Web (URSW)*, pages 45–55, 2005.
38. H.-M. Haav. An application of inductive concept analysis to construction of domain-specific ontologies. In *Proceedings of the VLDB Pre-conference Workshop on Emerging Database Research in East Europe*, 2003.
39. Z. S. Harris. *Mathematical Structures of Language*. Wiley, 1968.
40. M. A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING)*, pages 539–545, 1992.
41. M. A. Hearst and H. Schütze. Customizing a lexicon to better suit a computational task. In *Proceedings of the ACL SIGLEX Workshop on Acquisition of Lexical Knowledge from Text*, 1993.
42. J. Hendler. On beyond ontology. Keynote Talk at the International Semantic Web Conference (ISWC), 2004.
43. G. Heyer, M. Läuter, U. Quasthoff, T. Wittig, and C. Wolff. Learning relations using collocations. In *Proceedings of the IJCAI Workshop on Ontology Learning*, 2001.
44. J. Iria, C. Brewster, F. Ciravegna, and Y. Wilks. An incremental tri-partite approach to ontology learning. In *Proceedings of the Language Resources and Evaluation Conference (LREC-06), Genoa, Italy 22–28 May, 2006*.
45. P. Haase J. Völker and P. Hitzler. Learning expressive ontologies. In P. Buitelaar and P. Cimiano, editors, *Bridging the Gap between Text and Knowledge: Selected Contributions to Ontology Learning and Population from Text*, volume 167 of *Frontiers in Artificial Intelligence*. IOS Press, 2007.
46. L. Kaufman and P. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley, 1990.
47. J.-U. Kietz, R. Volz, and A. Mädche. Extracting a domain-specific ontology from a corporate intranet. In *Proceedings of the 2nd Learning Language in Logic (LLL) Workshop*, 2000.
48. N. Lavrac and S. Dzeroski. *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood, 1994.
49. L. Lee. Measures of distributional similarity. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 25–32, 1999.
50. D. Lin. Automatic retrieval and clustering of similar words. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING-ACL)*, pages 768–774, 1998.
51. F. A. Lisi and F. Esposito. Two orthogonal biases for choosing the intensions of emerging concepts in ontology refinement. In G. Brewka, S. Coradeschi, A. Perini, and P. Traverso, editors, *Proceedings of the 17th European Conference on Artificial Intelligence (ECAI)*, pages 765–766. IOS Press, 2006.
52. A. Mädche, B. Motik, and L. Stojanovic. Managing multiple and distributed ontologies in the semantic web. *VLDB Journal*, 12(4):286–302, 2003.
53. A. Mädche and S. Staab. Discovering conceptual relations from text. In *Proceedings of the 14th European Conference on Artificial Intelligence (ECAI)*, pages 321–325, 2000.

54. A. Mädche and S. Staab. Measuring similarity between ontologies. In *Proceedings of the European Conference on Knowledge Acquisition and Management (EKAW)*, pages 251–263, 2002.
55. A. Mädche and R. Volz. The Text-To-Onto ontology extraction and maintenance system. In *Workshop on Integrating Data Mining and Knowledge Management, collocated with the 1st International Conference on Data Mining*, 2001.
56. C. Manning and H. Schütze. *Foundations of Statistical Language Processing*. MIT Press, 1999.
57. V. Pekar and S. Staab. Taxonomy learning: Factoring the structure of a taxonomy into a semantic classification decision. *Proceedings of the 19th Conference on Computational Linguistics (COLING)*, 2:786–792, 2002.
58. F. Pereira, N. Tishby, and L. Lee. Distributional clustering of english words. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 183–190, 1993.
59. M. Poesio, T. Ishikawa, S. Schulte im Walde, and R. Viera. Acquiring lexical knowledge for anaphora resolution. In *Proceedings of the 3rd Conference on Language Resources and Evaluation (LREC)*, 2002.
60. M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.
61. P. Resnik. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *Journal of Artificial Intelligence Research (JAIR)*, 11:95–130, 1999.
62. D. Sanchez and A. Moreno. Web-scale taxonomy learning. In C. Biemann and G. Pass, editors, *Proceedings of the Workshop on Extending and Learning Lexical Ontologies using Machine Learning Methods*, 2005.
63. H. Schmid. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the International Conference on New Methods in Language Processing*, 1994.
64. A. Schutz and P. Buitelaar. RelExt: A tool for relation extraction from text in ontology extension. In *Proceedings of the International Semantic Web Conference (ISWC)*, pages 593–606, 2005.
65. E. Simperl, C. Tempich, and D. Vrandečić. A methodology for ontology learning. In P. Buitelaar and P. Cimiano, editors, *Bridging the Gap between Text and Knowledge: Selected Contributions to Ontology Learning and Population from Text*, volume 167 of *Frontiers in Artificial Intelligence*. IOS Press, 2007.
66. R. Snow, D. Jurafsky, and Y. Ng. Semantic taxonomy induction from heterogeneous evidence. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, pages 801–808, 2006.
67. M. Strube and S. Paolo Ponzetto. Wikirelate! computing semantic relatedness using wikipedia. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 1419–1424, 2006.
68. G. Stumme, R. Taouil, Y. Bastide, N. Pasquier, and L. Lakhal. Computing iceberg concept lattices with titanic. *Journal of Knowledge and Data Engineering (KDE)*, 42(2):189–222, 2002.
69. P. D. Turney. Mining the web for synonyms: PMI-IR versus LSA on TOEFL. In *Proceedings of the 12th European Conference on Machine Learning (ECML)*, pages 491–502, 2001.
70. P. Velardi, R. Navigli, A. Cuchiarrelli, and F. Neri. Evaluation of OntoLearn, a methodology for automatic population of domain ontologies. In P. Buitelaar,

- P. Cimiano, and B. Magnini, editors, *Ontology Learning from Text: Methods, Applications and Evaluation*, number 123 in *Frontiers in Artificial Intelligence and Applications*, pages 92–106. IOS Press, 2005.
71. J. Völker, P. Hitzler, and P. Cimiano. Acquisition of OWL DL axioms from lexical resources. In *Proceedings of the 4th European Semantic Web Conference (ESWC'07)*, pages 670–685, 2007.
 72. J. Völker, D. Vrandečić, and Y. Sure. Automatic evaluation of ontologies (AEON). In Y. Gil, E. Motta, V. R. Benjamins, and M. A. Musen, editors, *Proceedings of the 4th International Semantic Web Conference (ISWC)*, volume 3729 of *LNCS*, pages 716–731. Springer, 2005.
 73. J. Völker, D. Vrandečić, Y. Sure, and A. Hotho. Learning disjointness. In *Proceedings of the 4th European Semantic Web Conference (ESWC'07)*, pages 175–189, 2007.
 74. J. Völker and S. Rudolph. Lexico-logical acquisition of OWL DL axioms – An integrated approach to ontology refinement. In R. Medina and S. Obiedkov, editors, *Proceedings of the 6th International Conference on Formal Concept Analysis (ICFCA'08)*, volume 4933 of *Lecture Notes in Artificial Intelligence*, pages 62–77. Springer, 2008.
 75. D. Widdows. Unsupervised methods for developing taxonomies by combining syntactic and statistical information. In *Proceedings of the Human Language Technology Conference/North American Chapter of the Association for Computational Linguistics (HLT/NAACL)*, pages 276–283, 2003.
 76. I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 1999.