# Perspectives on Ontology Learning

Jens Lehmann and Johanna Völker (Eds.)

*Informatics Institute, University of Leipzig, Germany*
*Data & Web Science Research Group, University of Mannheim, Germany*

IOS Press

# Foreword

Philipp CIMIANO [a],
[a] *Semantic Computing Group, Bielefeld University*

Equipping machines with commonsense as well as domain-specific knowledge in order to endow them with human-like understanding of certain problem domains has been and still is a main goal of artificial intelligence research. In this context, a crucial question is how high the cost actually is for encoding all the relevant knowledge in such a way that it can be exploited by machines for automatic reasoning, inconsistency detection, etc. While there has been some recent work on developing methodologies allowing us to estimate the cost of knowledge engineering projects [12], it is legitimate to assume that not all the relevant knowledge can be encoded manually. Techniques that can extract and discover knowledge by analyzing human behaviour and data produced as a result thereof can offer an important contribution in this respect.

The field of ontology learning, a term coined by Alexander Mädche and Steffen Staab in 2001 [7], is concerned with the development of methods that can induce relevant ontological knowledge from data. The field can by now look back into more than ten years of intense research. Early research in the field focused on applying shallow methods to term and concept extraction as well as hierarchical and non-hierarchical relation extraction [7]. Later on, in my PhD thesis with the title "Ontology Learning and Population from Text: Algorithms, Evaluation and Applications", I defined ontology learning as the acquisition of a domain model from data and attempted to provide a systematic overview of ontology learning tasks by introducing the so called *ontology learning layer cake*, which has received wide attention since then. In recent years, several researchers have attempted to increase the expressivity of the ontologies learned from text data, in particular by attempting to extract deeper axiomatic knowledge (e.g. see [13], [14] and [4]). Some contributions along these lines can also be found in this volume, e.g. aiming at learning OWL axioms by applying inductive techniques (cf. Lehmann et al. [5] and Lisi [6] in this volume).

The problem of ontology learning has turned out to be much more difficult than expected. The main reason for this is, in my view, that an ontology always reflects a way of conceptualizing the world or a given domain, while the results of an ontology learning algorithm that learns from a set of data essentially reflects the idiosyncrasies of the dataset in question. As such, turning the results of an ontology algorithm into an ontology that actually reflects the conceptualization one has of a domain can be more costly than actually building the ontology from scratch. The problem of ontology learning has turned out to be much more difficult than expected. The main reason for this is, in my view, that an ontology always reflects a way of conceptualizing the world or a given domain, while the results of an ontology learning algorithm that learns from a set of data essentially reflects the idiosyncrasies of the dataset in question. As such, turning the results of an ontology

learning algorithm into an ontology that actually reflects the conceptualization one has of a domain can be more costly than actually building the ontology from scratch.

A second problem so far has been the lack of applications for automatically learned ontologies. While Hotho, Bloehdorn and myself [2] showed some positive impacts of automatically learned ontologies on classification and clustering tasks, not many other convincing applications were at sight at that stage. Recently, ontology learning has, however, seen interesting applications for inducing the semantics of tags used in social media data and folksonomies [1]. Recently, Meilicke et al. have shown that automatically induced knowledge, disjointeness axioms in particular, can be deployed to debug ontology mappings [9]. The fact that such applications are emerging is clearly a good sign that there is definitely progress in the field. A further interesting and very promising application potential for ontology learning lies in the field of Linked Data. Learning from Linked Data will allow us to induce schemata in a bottom-up fashion and let the schema evolve with the data. Ontology population will also continue to play a crucial role in taming and structuring the large amount of unstructured data available, e.g. in Scientific Publications. In many applications domains, one needs to consider all data together in order to extract key facts and knowledge, structure this knowledge in the form of a database in order to aggregate and summarize the data and provide analytical procedures that support decision making by experts.

In the mid-term, I foresee two very interesting research directions for ontology learning. When modelling knowledge, it is relatively easy to model "the obvious" and straightforward knowledge in a particular domain. However, the "not-so-obvious" and more complex relationships are harder to come up with for a knowledge engineer. This is where algorithms that induce more complex and non-trivial relationships from data can assist a human in the process of modelling more complex axioms. This is especially relevant and valuable in the Linked Data era where not many people seem to want to put effort into axiomatizing the vocabulary used in their datasets.

This is tightly related to the second future direction I regard as crucial within the field of ontology learning. So far, there has not been too much focus on how humans and machines can collaborate on the task of modelling relevant knowledge for a given domain. The role of machine agents should be to derive interesting axioms and relationships from data, generating hypotheses induced from data and asking the human for validation and clarification of them. Humans would then rely on their domain knowledge to confirm the induced knowledge or reject it by providing counter-examples. Methodologies that define and clarify the role of machines and humans in ontology engineering and ontology learning are urgently needed in order to exploit the capabilities of both humans and machines optimally. In this volume, for instance, there is one contribution by Simperl et al. [11] that shows how humans can be involved in the process of ontology learning through games with a purpose. Unless we have good methodologies incorporating the human in the loop, I dare to predict that major breakthroughs in ontology learning can not be expected.

Concerning applications, in my view there is one very important application for ontology learning, i.e. natural language processing (NLP). Given that language processing inherently requires world knowledge to guide the interpretation process [3], it strikes that so far – at least to my knowledge – there have not been too many convincing applications of proper OWL ontologies within NLP. We may wonder why this is the case. Is the field of NLP not aware of the techniques developed in the Semantic Web community?

Or are existing ontology learning techniques too noisy for researchers that are used to work with highly curated, hand-crafted resources such as WordNet, FrameNet etc.? Or is it simply the case that the knowledge contained in OWL ontologies and produced by state-of-the-art ontology learning tools is not adequate for NLP purposes? Being far from knowing the answer, let me speculate a bit about the reasons:

- **Lack of coverage**: domain ontologies – whether learned or not – typically have a limited coverage and scope, whereas NLP researchers are typically used to work with domain-independent resources such as WordNet, FrameNet, etc. An NLP researcher would have to work with many, possibly overlapping ontologies.
- **Limited quality**: Automatically learned ontologies might be noisy, but still useful as some applications have been shown. This requires also a paradigm shift towards working with non-perfect, possibly noisy, incomplete or even logically inconsistent ontologies.
- **Limitations of crisp knowledge**: In human language, the meaning of words is often vague so that prototypes or distributions might be better suitable for NLP applications.rather than crisp knowledge representations formalisms.
- **No need for expressive knowledge (yet!)**: With the statistical turn in the 80s and 90s, NLP focused mainly on statistical approaches to natural language processing, moving away from purely symbolic approaches. However, one observes a move back into symbolic approaches as people realize more and more that inference is crucial for NLP. New models such as Markov Logic Networks [10] that combine statistic with symbolic, first-order theories have received in fact wide attention in the NLP and Semantic Web communities. They might, thus, offer a point of convergence for both communities. Concentrating on learning probabilistic knowledge will thus be an important avenue for future work.
- **Lack of awareness:** For sure, the NLP community is not particularly aware of what is going on in the Semantic Web and ontology learning communities. This is clearly corroborated by the fact that only few NLP researchers attend Semantic Web conferences. The number of references to Semantic Web related work in NLP papers at major NLP conference converges practically to zero, as an informal empirical study by Josef van Genabith showed[1]. For sure, there is a lot that the Semantic Web community could do about this, i.e. organizing tutorials and workshops at NLP conferences, but also regarding NLP as a potential consumer of ontologies – whether learned or not. A number of recent activities in the Semantic Web field, such as the development of NIF (Hellmann et. al, this volume) or *lemon*[2], a model for the lexicon-ontology interface [8] have contributed already to creating important synergies between and to the convergence of both communities.

In the future, ontology learning research should in my opinion not only concentrate on what we can learn, but also for which purpose or which applications the learned knowledge might be useful. Only then will we be able to create a strong case for Semantic Web technologies, ontology learning techniques in particular, outside the Semantic Web

[1] Josef van Genabith presented this observation at the Dagstuhl Seminar on the "Multilingual Semantic Web" in September of 2012

[2] See also the standardization activities by the ontolex group: `http://www.w3.org/community/ontolex`

community. Having said this, I kindly invite you as reader to immerse into the current state-of-the-art in ontology learning research and enjoy the great book that Jens Lehmann and Johanna Völker have compiled together. For sure, there is no better book to learn about recent developments in ontology learning research. Thanks to Jens and Johanna for editing this great book and for working on this topic so enthusiastically in order to contribute to the further evolution of this (still) very exciting and important field.

## References


[1] Dominik Benz, Christian Körner, Andreas Hotho, Gerd Stumme, and Markus Strohmaier. One tag to bind them all : Measuring term abstractness in social metadata. In Grigoris Antoniou, Marko Grobelnik, Elena Simperl, Bijan Parsia, Dimitris Plexousakis, Jeff Pan, and Pieter De Leenheer, editors, *Proceedings of the 8th Extended Semantic Web Conference (ESWC 2011)*, 2011.

[2] Stephan Bloehdorn, Philipp Cimiano, and Andreas Hotho. Learning ontologies to improve text clustering and classification. In Myra Spiliopoulou, Rudolf Kruse, Andreas Nürnberger, Christian Borgelt, and Wolfgang Gaul, editors, *From Data and Information Analysis to Knowledge Engineering: Proceedings of the 29th Annual Conference of the German Classification Society (GfKl 2005), March 9-11, 2005, Magdeburg, Germany*, volume 30, pages 334–341. Springer, Berlin–Heidelberg, Germany, 2006.

[3] Philipp Cimiano, Christina Unger, and John McCrae. *Ontology-based Interpretation of Natural Language*. Synthesis Lectures on Human Language Technology. Morgan & Claypool Publishers. to appear.

[4] Daniel Fleischhacker, Johanna Völker, and Heiner Stuckenschmidt. Mining rdf data for property axioms. In *Proceedings of the Confederated International Conferences (OTM)*, pages 718–735, 2012.

[5] Jens Lehmann, Nicola Fanizzi, and Claudia d'Amato. Concept learning. In Jens Lehmann and Johanna Völker, editors, *Perspectives on Ontology Learning*, Studies on the Semantic Web. AKA Heidelberg / IOS Press, 2014.

[6] Francesca Lisi. Learning onto-relational rules with inductive logic programming. In Jens Lehmann and Johanna Völker, editors, *Perspectives on Ontology Learning*, Studies on the Semantic Web. AKA Heidelberg / IOS Press, 2014.

[7] Alexander Maedche and Steffen Staab. Ontology learning for the semantic web. *IEEE Intelligent Systems*, 16(2):72–79, 2001.

[8] J. McCrae, G. Aguado-de Cea, P. Buitelaar, P. Cimiano, T. Declerck, A. Gómez-Pérez, J. Gracia, L. Hollink, E. Montiel-Ponsoda, D. Spohr, and T. Wunner. Interchanging lexical resources on the Semantic Web. *Language Resources and Evaluation*, 2012.

[9] Christian Meilicke, Johanna Völker, and Heiner Stuckenschmidt. Learning disjointness for debugging mappings between lightweight ontologies. In *Proceedings of the 16th International Conference on Knowledge Engineering: Practice and Patterns*, pages 93–108, 2008.

[10] Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine Learning*, 62(1-2):107–136, 2006.

[11] Elana Simperl, Stephan Wölger, Stefan Thaler, and Katharina Siorpaes. Learning ontologies via games with a purpose. In Jens Lehmann and Johanna Völker, editors, *Perspectives on Ontology Learning*, Studies on the Semantic Web. AKA Heidelberg / IOS Press, 2014.

[12] Elena Simperl, Tobias Bürger, Simon Hangl, Stephan Wörgl, and Igor O. Popov. Ontocom: A reliable cost estimation method for ontology development projects. *Journal of Web Semantics*, 16:1–16, 2012.

[13] Johanna Völker, Pascal Hitzler, and Philipp Cimiano. Acquisition of owl dl axioms from lexical resources. In *Proceedings of the 4th European Semantic Web Conference (ESWC)*, pages 670–685, 2007.

[14] Johanna Völker, Denny Vrandecic, York Sure, and Andreas Hotho. Learning disjointness. In *Proceedings of the 4th European Semantic Web Conference (ESWC)*, pages 175–189, 2007.

# An Introduction to Ontology Learning

Jens LEHMANN [a] and Johanna VÖLKER [b,1]
[a] *Informatics Institute, University of Leipzig, Germany*
[b] *Data & Web Science Research Group, University of Mannheim, Germany*

Ever since the early days of Artificial Intelligence and the development of the first knowledge-based systems in the 70s [32] people have dreamt of self-learning machines. When knowledge-based systems grew larger and the commercial interest in these technologies increased, people became aware of the knowledge acquisition bottleneck and the necessity to (partly) automatize the creation and maintenance of knowledge bases. Today, many applications which exhibit 'intelligent' behavior thanks to symbolic knowledge representation and logical inference rely on ontologies and the standards provided by the World Wide Web Committee (W3C). Supporting the construction of ontologies and populating them with instantiations of both concepts and relations, commonly referred to as *ontology learning*.

Early research in ontology learning has concentrated on the extraction of facts or schema-level knowledge from textual resources building upon earlier work in the field of computational linguistics and lexical acquisition. However, as we will show in this book, ontology learning is a very diverse and interdisciplinary field of research. Ontology learning approaches are as heterogeneous as the sources of data on the web, and as different from one another as the types of knowledge representations called "ontologies".

In the remainder of this introduction, we briefly summarize the state-of-the-art in ontology learning and elaborate on what we consider as the key challenges for current and future ontology learning research.

### Ontology-based Knowledge Representation

Ontologies in computer science are usually regarded as formal representations of knowledge – often restricted to a particular domain. There is, however, no general agreement on which requirements the formal representation needs to satisfy in order to be appropriately be called an ontology. Depending on the particular point of view, ontologies can be simple dictionaries, taxonomies, thesauri, or richly axiomatized top-level formalisations. In this book, we do not limit ourselves to a particular definition of the term ontology, since use case specific requirements determine the needed expressivity of knowledge bases: the contributions range from approaches to learning lightweight structures to methods for generating complex definitions of classes.

Ontologies play a central role in data and knowledge integration. By providing a shared schema, they facilitate query answering and reasoning over disparate data sources.

[1] 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.

Since the amount of data on the web as well as in corporate intranets, for example, is growing quickly, structured access to data becomes more important. In addition to data integration, reasoning and querying scenarios, ontologies are also a means to document the structure of a particular domain, which helps to develop a common understanding of its concepts.

However, the construction of ontologies is a highly expensive task which crucially hinges on the availability of scarce expert resources [39]. In order to build a formal ontology for a particular domain of interest, for instance, specialized domain knowledge needs to be acquired and formalized in a way that automated inference will yield the expected results. This goal can only be achieved if domain experts collaborate with skilled ontology engineers familiar with the theory and practice of knowledge representation – and once the ontology has been constructed, evolving knowledge and application requirements will demand for continuous maintenance efforts.

### Ontology Learning

It is more than ten years now that Mädche and Staab [29] coined the term "ontology learning" for a newly emerging field of research aiming at nothing less than the automatic generation of ontologies. The first ontology learning workshop[2], held in 2000 and co-organized by Claire Nédellec and Peter Wiemer-Hastings, brought together people from very different research communities. Looking into the proceedings, we can distinguish works based on ripple down rules, word sense clustering, and information extraction, for example. It is remarkable that there were only few contributions from the field of concept learning at that time, although researchers have investigated the use of inductive logic programming for learning logical theories since the mid 80s. From today's perspective, ontology learning is a use case for concept learning, but the collaboration and exchange between this and other parts of the ontology learning community is still limited. Ten years after the first ontology learning workshop, we tried to compile a book which brings together the most diverse works in the area of ontology learning including contributions by the concept learning community as well as "classical" works on ontology learning from text or other semi-structured resources. It is designed to give an overview of a broad range of ontology learning approaches, logical and statistical ones, and to outline the synergies that may arise from bringing together the various methods and methodologies.

While we do not intend to draw a sharp line between different types of ontology learning, approaches can be roughly classified into the following areas:

**Ontology Learning from Text** mostly focuses on the automatic or semi-automatic generation of lightweight taxonomies by means of text mining and information extraction. Many of the methods used in ontology learning from text (e.g. lexico-syntactic patterns for hyponymy detection or named-entity classification) are inspired by previous work in the field of computational linguistics, essentially designed in order to facilitate the acquisition of lexical information from corpora. Some ontology learning approaches do not derive schematic structures, but focus on the data level. Such *ontology population* methods derive facts from text. A popular example is the Never-Ending Language Learning (NELL) project [10], which reads the web to add statements to its knowledge base and improves its performance over time, e.g. via user feedback.

[2] `http://ol2000.aifb.uni-karlsruhe.de`

**Linked Data Mining** refers to the process of detecting meaningful patterns in RDF graphs. One of the motivations behind this research area is that Linked Data publishers sometimes do not create an explicit schema for their dataset upfront, but focus on publishing data first. Being able to detect the structure within published RDF graphs can, on the one hand, simplify the later creation of schemata and, on the other hand, allow to detect interesting associations between elements in the RDF graph. This can be achieved via statistical schema induction [7,8,43] or statistical relational learning methods, which mine frequent patterns and correlations in large data sets. In Linked Data mining, clustering approaches can be used to group related resources and provide an enhanced structure for the underlying data.

**Concept Learning in Description Logics and OWL** is a direction of research that aims at learning schema axioms, such as definitions of classes, from existing ontologies and instance data. Most methods in this area are based on Inductive Logic Programming methods [33]. While many algorithms, such as DL-FOIL [16] and OCEL [25] are generic supervised machine learning approaches for description logics, there are also specific adaptations to ontology learning [22], e.g., in terms of performance and usability. Closely related to concept learning in Description Logics is *onto-relational learning*, which combines methods for learning OWL axioms with rule learning approaches [27].

**Crowdsourcing** ontologies is an interesting alternative to purely automatic approaches as it combines the speed of computers with the accuracy of humans. Provided that the task to be completed is simple enough, it only requires the right incentives for people to contribute. Examples of crowdsourcing in the field of ontology learning include taxonomy construction via Amazon mechanical turk, and games with a purpose for ontology population (see, e.g., [12,19]).

Other approaches include, e.g., transfer learning and ontology re-use, which try to adapt existing ontologies to new domains by partially re-using existing schematic structures. Furthermore, apart from the above mentioned combination of rules and ontologies, direct representations of uncertainty, e.g. via Markov Logic Networks, are also investigated.

Major and minor distinctions between these approaches make it difficult to come up with a formal definition [41], or a breakdown into concrete subtasks (see, e.g., the *ontology learning layer cake* [35]) which is neither too general nor limited to one particular type of approach [15,29,45]. However, it is this variety that makes the ontology learning community so rich and inspiring. A lot of progress has been made in each of the abovementioned fields, and we can observe a trend towards hybrid and integrated approaches. For this book, we assembled contributions by researchers addressing some of the key challenges in ontology learning:

***Heterogeneity.*** Data on the web differs largely, e.g., with respect to formats, languages, domains and quality.[3] Approaches to learning from heterogeneous sources of evidence [9] can effectively leverage this huge variety by increasing the accuracy as well as the coverage of learned ontologies. However, neither the integration of methods nor the homogenization of data has attracted high attention within the ontology learning commu-

[3] See [28] for an early attempt to categorize the different types of data that can serve as input to ontology learning approaches.

nity so far and remains to be a challenge for the application of ontology learning methods in practice.

***Uncertainty.*** Low-quality or unstructured data, which is hard to interpret by computational means, as well as inherently imperfect methods for learning ontologies can lead to results that are less likely to be correct. Thus, many ontology learning methods are designed in a way that they associate each individual outcome with a certainty value reflecting the methods' *confidence* with regard to the correctness of particular results. Uncertainty values and other types of provenance information such as timestamps or authorship annotations are especially important when it comes to manual or automatic debugging of learned ontologies [14].

***Reasoning.*** Often, ontologies are learned or manually created for applications which are based on logical inference. In case these applications require the ontology to be logically consistent, ontology learning approaches should be capable of generating consistent (and coherent) ontologies. Therefore, not only methods for concept learning in description logics, but also other ontology learning approaches rely on logical reasoning. Experiments have shown that a tight coupling between ontology debugging, i.e. inconsistency diagnosis and repair, and ontology learning may be beneficial [31,23].

***Scalability.*** Extracting knowledge from the growing amounts of data on the web – unstructured, textual data on the one hand and structured data such as databases, linked data[4] or ontologies on the other hand – requires scalable and efficient approaches. Especially when it comes to learning from distributed, loosely interconnected data and the integration of knowledge from multiple sources, ontology learning methods face big challenges. In order to address these challenges, various strategies are currently being developed, such as distributed computation for horizontally scaling ontology learning, incremental learning approaches for re-using existing knowledge, or sampling [17] and modularization to improve the efficiency of ontology learning algorithms.

***Quality.*** We can measure the quality of an automatically generated ontology as we can measure the quality of any ontology, be it learned or manually engineered. However, *ontology evaluation* is not an easy task (see [44] for a comprehensive overview of the state-of-the-art in ontology evaluation). Formal correctness, completeness and consistency are only a few of many possible criteria for judging the quality of an ontology, and it is the application context of an ontology which ultimately determines the choice of evaluation criteria as well as the required quality standard. Ideally, each step of an ontology learning process, including the choice of input data as well as preprocessing and relation extraction, for example, should thus be optimized with regard to the particular domain or application context that the learned ontology will be used for. Ontology learning methodologies [40] and the adoption of ontology design patterns [4] can help to further improve the results.

***Interactivity.*** In practice, the quality of a learned ontology often depends on the degree of automation. The lesser the extent to which humans are involved in a semi-automatic ontology generation process, the lower the quality we can expect. An ontology insufficient for the intended application in terms of quality, e.g., after having been generated in a fully automatic way, will eventually require a significant amount of post-processing.

[4] See `http://stats.lod2.eu` and `http://lod-cloud.net`.

While the revision of learned ontologies is not generally considered part of the actual ontology learning process, methods for automatic ontology generation can support this sort of post-processing by providing detailed provenance information. Provenance information acquired in the course of ontology learning typically comprises, for example, confidence and relevance values for individual axioms as well as time stamps and key facts about the employed learning procedure. Nevertheless, the amount of post-processing can be a significant burden for knowledge engineers, and innovative methods are required in order to overcome the so-called *knowledge acquisition bottleneck*. Generally, it is advisable to integrate methods for ontology learning and revision into popular ontology engineering frameworks, in order to reduce the overhead for human participation in the overall process [22,37]. Crowdsourcing and games with a purpose can help to lower the costs of revising learned ontologies by involving non-experts, but translating their interactions into ontology modeling decisions is a non-trivial problem. Systematic expert interrogation, known as relational exploration, has been found to be an efficient way of asking people the right questions, while at the same time reducing the overall number of decisions to be made [2]. Finally, experiments have shown that ontology design patterns, which capture the knowledge and experience of human ontology engineers, can beneficially be integrated into the ontology learning process [5].

**About this Book**

In the following, we describe the structure of the book and give an outline of the content of individual chapters.

*Foundations* This part of the book covers the most basic concepts which are important to understand state-of-the-art ontology learning approaches. First, Krötzsch et al. [20] give a primer on description logics, the family of knowledge representation formalisms which underly most of the learned ontologies. The second chapter contributed by Jentzsch, Vrandečić and Usbeck [18] introduces RDF and the Linked Data principles as well as fundamental semantic web technologies such as SPARQL, for example. It is followed by an overview of typical machine learning approaches which are applied in ontology learning (see Ławrynowicz and Tresp [21]). Finally, as the vast majority of ontology learning algorithms still relies on textual input, Maynard and Bontcheva [30] cover the most relevant natural language processing techniques.

*Logical Learning* The second part of this book focuses on approaches which have been developed to derive ontologies from structured knowledge. Logical learning methods are introduced by Lehmann et al. [24], who cover the foundations of learning description logic concepts, as well as by Francesca Lisi [26] who contributed a chapter on learning onto-relational rules. The presented methods are based on the assumption that schema axioms, such as concept definitions, can be learned from existing instance level knowledge.

*Lexical Learning* Given the vast amounts of textual contents on the web it is not astonishing that previous research in ontology learning mainly concentrated on the acquisition of ontologies from unstructured data. In this part of the book, we therefore present contributions to ontology learning from natural language text based on information extraction and text mining techniques. The first chapter (cf. Coppola et al. [13]) introduces a methodology for generating domain-specific ontologies by specializing and instantiating

frames from the FrameNet lexicon. In the second chapter, Fabian Suchanek [42] gives an overview of well-known methods for acquiring facts and taxonomies from Wikipedia articles, while putting an emphasis on the synergies between ontological reasoning and information extraction. Finally, the chapter contributed by Nováček and Handschuh [34] outlines a layered ontology learning framework, which facilitates the integration of facts extracted from textual documents with existing schema-level knowledge.

*Learning from Web Data* The increasing amounts of data available through the web and the growing need for automatic approaches to the access and usage of information on the web, pose particular challenges to ontology learning methods. This part of the book is therefore dedicated to methods which take into account the specific characteristics of web data, such as heterogeneity, volume, decentralization and a large variety of different formats. A part of the world wide web, which has seen a tremendous rise in importance over the past decade, are social websites and crowdsourced tagging applications. The first chapter in this part of the book gives a systematic overview over the trends and future developments in the area of knowledge extraction from tagging systems or folksonomies (see Benz and Hotho [3]). Spatial applications and annotations of points of interests are investigated by Alves and Pereira [1] in the second chapter, which deals with approaches to semantically enrich the descriptions of spatial objects. Finally, in the last chapter in this part, Cerbah and Lammari [11] address the problem of web application backends. Motivated by the fact that the vast majority of web applications are driven by relational databases, they present methods for deriving ontologies from schemata of relational databases.

*Dynamics and User Interaction* In many cases, purely automatic learning approaches will fail to generate ontologies which are good enough for a particular, e.g. reasoning-based, application. For this reason, we would like to emphasize the role users and knowledge engineers can play in an ontology learning process. The first chapter of this part written by Simperl et al. [38] explains how games with a purpose can help to leverage human resources for learning ontologies. A completely different approach to "putting the human in the loop" is presented by Rudolph and Sertkaya [36], who introduce formal concept analysis as a means to interactive ontology learning and as an effective way to minimize the required human effort. Last, but not least, Blomqvist et al. [6] elaborate on key challenges in ontology learning and how those can be addressed by the use of ontology design patterns, which encode best-practices in ontology engineering.

## References

[1] Ana Oliveira Alves and Francisco Camara Pereira. Semantic enrichment of places: From public places descriptions to linked data. In Jens Lehmann and Johanna Völker, editors, *Perspectives on Ontology Learning*, Studies on the Semantic Web. AKA Heidelberg / IOS Press, 2014.

[2] Franz Baader, Bernhard Ganter, Barış Sertkaya, and Ulrike Sattler. Completing Description Logic knowledge bases using Formal Concept Analysis. In Manuela M. Veloso, editor, *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 230–235, 2007.

[3] Dominik Benz and Andreas Hotho. Capturing emergent semantics from social tagging systems. In Jens Lehmann and Johanna Völker, editors, *Perspectives on Ontology Learning*, Studies on the Semantic Web. AKA Heidelberg / IOS Press, 2014.

[4] Eva Blomqvist. Ontocase – a pattern-based ontology construction approach. In Robert Meersman and Zahir Tari, editors, *On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS*, volume 4803 of *Lecture Notes in Computer Science*, pages 971–988. Springer Berlin Heidelberg, 2007.

[5] Eva Blomqvist. *Semi-automatic Ontology Construction based on Patterns*. PhD thesis, Linköping University, Department of Computer and Information Science at the Institute of Technology, 2009.

[6] Eva Blomqvist, Aldo Gangemi, and Francesco Draicchio. Ontology design patterns in ontology learning. In Jens Lehmann and Johanna Völker, editors, *Perspectives on Ontology Learning*, Studies on the Semantic Web. AKA Heidelberg / IOS Press, 2014.

[7] Lorenz Bühmann and Jens Lehmann. Universal OWL axiom enrichment for large knowledge bases. In *Proceedings of EKAW 2012*, pages 57–71. Springer, 2012.

[8] Lorenz Buhmann and Jens Lehmann. Pattern based knowledge base enrichment. In *12th International Semantic Web Conference, 21-25 October 2013, Sydney, Australia*, 2013.

[9] Paul Buitelaar, Philipp Cimiano, Anette Frank, Matthias Hartung, and Stefania Racioppa. Ontology-based information extraction and integration from heterogeneous data sources. *Journal on Human-Computer Studies*, 66(11):759–788, 2008.

[10] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E.R. Hruschka Jr, and T.M. Mitchell. Toward an architecture for never-ending language learning. In *Proceedings of the 24th Conference on Artificial Intelligence (AAAI)*, volume 2, pages 1306–1313, 2010.

[11] Farid Cerbah and Nadira Lammari. Ontology learning from databases: Some efficient methods to discover semantic patterns in data. In Jens Lehmann and Johanna Völker, editors, *Perspectives on Ontology Learning*, Studies on the Semantic Web. AKA Heidelberg / IOS Press, 2014.

[12] Lydia B. Chilton, Greg Little, Darren Edge, Daniel S. Weld, and James A. Landay. Cascade: Crowdsourcing taxonomy creation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 1999–2008, New York, NY, USA, 2013. ACM.

[13] Bonaventura Coppola, Aldo Gangemi, Alfio Gliozzo, Davide Picca, and Valentina Presutti. Learning domain ontologies by corpus-driven framenet specialization. In Jens Lehmann and Johanna Völker, editors, *Perspectives on Ontology Learning*, Studies on the Semantic Web. AKA Heidelberg / IOS Press, 2014.

[14] Paulo Cesar G. da Costa, Kathryn B. Laskey, Kenneth J. Laskey, and Michael Pool, editors. *International Semantic Web Conference, ISWC 2005, Galway, Ireland, Workshop 3: Uncertainty Reasoning for the Semantic Web, 7 November 2005*, 2005.

[15] X.Y. Du, M. Li, and S. Wang. A survey on ontology learning research. *Journal of Software*, 17(9):1837–1847, 2006.

[16] Nicola Fanizzi, Claudia d'Amato, and Floriana Esposito. DL-FOIL: Concept learning in description logics. In F. Zelezný and N. Lavrac, editors, *Proceedings of the 18th International Conference on Inductive Logic Programming (ILP)*, volume 5194 of *LNAI*, pages 107–121. Springer, 2008.

[17] Sebastian Hellmann, Jens Lehmann, and Sören Auer. Learning of OWL class descriptions on very large knowledge bases. *International Journal on Semantic Web and Information Systems*, 5(2):25–48, 2009.

[18] Anja Jentzsch, Ricardo Usbeck, and Denny Vrandečić. An incomplete and simplifying introduction to linked data. In Jens Lehmann and Johanna Völker, editors, *Perspectives on Ontology Learning*, Studies on the Semantic Web. AKA Heidelberg / IOS Press, 2014.

[19] Dimitris Karampinas and Peter Triantafillou. Crowdsourcing taxonomies. In *Proceedings of the 9th International Conference on The Semantic Web: Research and Applications*, ESWC'12, pages 545–559, Berlin, Heidelberg, 2012. Springer-Verlag.

[20] Markus Krötzsch, Frantisek Simančík, and Ian Horrocks. A description logic primer. In Jens Lehmann and Johanna Völker, editors, *Perspectives on Ontology Learning*, Studies on the Semantic Web. AKA Heidelberg / IOS Press, 2014.

[21] Agnieszka Ławrynowicz and Volker Tresp. Introducing machine learning. In Jens Lehmann and Johanna Völker, editors, *Perspectives on Ontology Learning*, Studies on the Semantic Web. AKA Heidelberg / IOS Press, 2014.

[22] Jens Lehmann, Sören Auer, Lorenz Bühmann, and Sebastian Tramp. Class expression learning for ontology engineering. *Journal of Web Semantics*, 9:71 – 81, 2011.

[23] Jens Lehmann and Lorenz Bühmann. ORE – a tool for repairing and enriching knowledge bases. In *Proceedings of the 9th International Semantic Web Conference (ISWC)*, Lecture Notes in Computer Science, Berlin / Heidelberg, 2010. Springer.

[24] Jens Lehmann, Nicola Fanizzi, and Claudia d'Amato. Concept learning. In Jens Lehmann and Johanna Völker, editors, *Perspectives on Ontology Learning*, Studies on the Semantic Web. AKA Heidelberg / IOS Press, 2014.

[25] Jens Lehmann and Pascal Hitzler. Concept learning in description logics using refinement operators.

*Machine Learning journal*, 78(1-2):203–250, 2010.

[26] Francesca Lisi. Learning onto-relational rules with inductive logic programming. In Jens Lehmann and Johanna Völker, editors, *Perspectives on Ontology Learning*, Studies on the Semantic Web. AKA Heidelberg / IOS Press, 2014.

[27] Francesca A. Lisi and Floriana Esposito. Nonmonotonic onto-relational learning. In Luc De Raedt, editor, *ILP*, volume 5989 of *Lecture Notes in Computer Science*, pages 88–95. Springer, 2009.

[28] Alexander Mädche. *Ontology Learning for the Semantic Web*. PhD thesis, Universität Karlsruhe (TH), Germany, 2001.

[29] Alexander Mädche and Steffen Staab. Ontology learning for the semantic web. *IEEE Intelligent Systems*, 16(2):72–79, 2001.

[30] Diana Maynard and Kalina Bontcheva. Natural language processing. In Jens Lehmann and Johanna Völker, editors, *Perspectives on Ontology Learning*, Studies on the Semantic Web. AKA Heidelberg / IOS Press, 2014.

[31] Christian Meilicke, Johanna Völker, and Heiner Stuckenschmidt. Learning disjointness for debugging mappings between lightweight ontologies. In Aldo Gangemi and Jerome Euzenat, editors, *Knowledge Engineering: Practice and Patterns*, volume 5268 of *Lecture Notes in Computer Science*, pages 93–108. Springer Berlin Heidelberg, 2008.

[32] M. Minsky. A framework for representing knowledge. In P. Winston, editor, *The Psychology of Computer Vision*, pages 211–277. McGraw-Hill, New York, 1975.

[33] Shan-Hwei Nienhuys-Cheng and Ronald de Wolf, editors. *Foundations of Inductive Logic Programming*, volume 1228 of *Lecture Notes in Computer Science*. Springer, 1997.

[34] Vit Novacek and Siegfried Handschuh. Empirically grounded emergent knowledge. In Jens Lehmann and Johanna Völker, editors, *Perspectives on Ontology Learning*, Studies on the Semantic Web. AKA Heidelberg / IOS Press, 2014.

[35] Bernardo Magnini Paul Buitelaar, Philipp Cimiano. *Ontology Learning from Text: Methods, Evaluation and Applications*. IOS Press, 2008.

[36] Sebastian Rudolph and Baris Sertkaya. Formal concept analysis methods for interactive ontology learning. In Jens Lehmann and Johanna Völker, editors, *Perspectives on Ontology Learning*, Studies on the Semantic Web. AKA Heidelberg / IOS Press, 2014.

[37] Barış Sertkaya. OntoComP: A protégé plugin for completing OWL ontologies. In *The Semantic Web: Research and Applications*, volume 5554 of *LNCS*, pages 898–902. Springer Berlin Heidelberg, 2009.

[38] Elana Simperl, Stephan Wölger, Stefan Thaler, and Katharina Siorpaes. Learning ontologies via games with a purpose. In Jens Lehmann and Johanna Völker, editors, *Perspectives on Ontology Learning*, Studies on the Semantic Web. AKA Heidelberg / IOS Press, 2014.

[39] Elena Simperl, Tobias Buerger, Simon Hangl, Stephan Woelger, and Igor Popov. Ontocom: A reliable cost estimation method for ontology development projects. *Web Semantics: Science, Services and Agents on the World Wide Web*, 16(0):1 – 16, 2012.

[40] Elena Simperl, Christoph Tempich, and Denny Vrandečić. A methodology for ontology learning. In *Proceedings of the Workshop on Ontology Learning and Population: Bridging the Gap between Text and Knowledge*, pages 225–249. IOS Press, 2008.

[41] Michael Sintek, Paul Buitelaar, and Daniel Olejnik. A formalization of ontology learning from text. In *Proceedings of the Workshop on Evaluation of Ontology-based Tools (EON) at the International Semantic Web Conference (ISWC)*, 2004.

[42] Fabian Suchanek. Information extraction for ontology learning. In Jens Lehmann and Johanna Völker, editors, *Perspectives on Ontology Learning*, Studies on the Semantic Web. AKA Heidelberg / IOS Press, 2014.

[43] Johanna Völker and Mathias Niepert. Statistical schema induction. In Grigoris Antoniou, Marko Grobelnik, Elena Paslaru Bontas Simperl, Bijan Parsia, Dimitris Plexousakis, Pieter De Leenheer, and Jeff Z. Pan, editors, *The Semantic Web: Research and Applications - 8th Extended Semantic Web Conference (ESWC), Heraklion, Crete, Greece, May 29-June 2, 2011, Proceedings*, volume 6643 of *Lecture Notes in Computer Science*, pages 124–138. Springer, 2011.

[44] Denny Vrandečić. Ontology evaluation. In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, International Handbooks on Information Systems, pages 293–313. Springer, 2009.

[45] W. Wong, W. Liu, and M. Bennamoun. Ontology learning from text: A look back and into the future. *ACM Computing Surveys (CSUR)*, 44(4):20, 2012.

# Contents

# Part I

# Foundations

# A Description Logic Primer

Markus KRÖTZSCH, František SIMANČÍK, Ian HORROCKS
*Department of Computer Science, University of Oxford, UK*

**Abstract.** This chapter provides a self-contained first introduction to description logics (DLs). The main concepts and features are explained with examples before syntax and semantics of the DL $\mathcal{SROIQ}$ are defined in detail. Additional sections review light-weight DL languages, discuss the relationship to the Web Ontology Language OWL and give pointers to further reading.

**Keywords.** Description Logics, OWL Web Ontology Language, Knowledge Representation, Ontological Modelling, $\mathcal{SROIQ}$

## Introduction

Description logics (DLs) are a family of knowledge representation languages that are widely used in ontological modelling. An important practical reason for this is that they provide one of the main underpinnings for the OWL Web Ontology Language as standardised by the World Wide Web Consortium (W3C). However, DLs have been used in knowledge representation long before the advent of ontological modelling in the context of the Semantic Web, tracing back to first DL modelling languages in the mid 1980s.

As their name suggests, DLs are logics (in fact most DLs are decidable fragments of first-order logic), and as such they are equipped with a *formal semantics*: a precise specification of the meaning of DL ontologies. This formal semantics allows humans and computer systems to exchange DL ontologies without ambiguity as to their meaning, and also makes it possible to use logical deduction to *infer* additional information from the facts stated explicitly in an ontology – an important feature that distinguishes DLs from other modelling languages such as UML.

The capability of inferring additional knowledge increases the modelling power of DLs but it also requires some understanding on the side of the modeller and, above all, good tool support for computing the conclusions. The computation of inferences is called *reasoning* and an important goal of DL language design has been to ensure that reasoning algorithms of good performance are available. This is one of the reasons why there is not just a single description logic: the best balance between expressivity of the language and complexity of reasoning depends on the intended application.

In this chapter we provide a self-contained first introduction to description logics. We start by explaining the basic way in which knowledge is modelled in DLs in Section 1 and continue with an intuitive introduction to the most important DL modelling features in Section 2. This leads us to the rather expressive DL called $\mathcal{SROIQ}$, the syntax of which we summarise in Section 3. In Section 4, we explain the underlying ideas of DL semantics and use it to define the meaning of $\mathcal{SROIQ}$ ontologies. Many DLs can be obtained by omitting some features of $\mathcal{SROIQ}$ and in Section 5 we review some of the

most important DLs obtained in this way. In particular, this includes various lightweight description logics that allow for particularly efficient reasoning. In Section 6 we discuss the relationship of DLs to the OWL Web Ontology Language. We conclude with pointers to further reading in Section 7.

## 1. Basic Building Blocks of DL Ontologies

Description logics (DLs) provide means to model the relationships between entities in a domain of interest. In DLs there are three kinds of entities: concepts, roles and individual names.[1] Concepts represent sets of individuals, roles represent binary relations between the individuals, and individual names represent single individuals in the domain. Readers familiar with first-order logic will recognise these as unary predicates, binary predicates and constants.

For example, an ontology modelling the domain of people and their family relationships might use concepts such Parent to represent the set of all parents and Female to represent the set of all female individuals, roles such as parentOf to represent the (binary) relationship between parents and their children, and individual names such as julia and john to represent the individuals Julia and John.

Unlike a database, a DL ontology does not fully describe a particular situation or "state of the world"; rather it consists of a set of statements, called axioms, each of which must be true in the situation described. These axioms typically capture only partial knowledge about the situation that the ontology is describing, and there may be many different states of the world that are consistent with the ontology. Although, from the point of view of logic, there is no principal difference between different types of axioms, it is customary to separate them into three groups: assertional (ABox) axioms, terminological (TBox) axioms and relational (RBox) axioms.

### *1.1. Asserting Facts with ABox Axioms*

ABox axioms capture knowledge about named individuals, i.e., the concepts to which they belong and how they are related to each other. The most common ABox axioms are *concept assertions* such as

$$\mathsf{Mother}(\mathsf{julia}), \tag{1}$$

which asserts that Julia is a mother or, more precisely, that the individual named julia is an *instance* of the concept Mother.

*Role assertions* describe relations between named individuals. The assertion

$$\mathsf{parentOf}(\mathsf{julia}, \mathsf{john}), \tag{2}$$

for example, states that Julia is a parent of John or, more precisely, that the individual named julia is in the relation that is represented by parentOf to the individual named john. The previous sentence shows that it can be rather cumbersome to explicitly point out that the relationships expressed by an axiom are really relationships between the individuals,

[1] In OWL concepts and roles are respectively known as classes and properties; see Section 6.

sets and relations that are represented by the respective individual names, concepts and roles. Assuming that this subtle distinction between syntactic identifiers and semantic entities is understood, we will thus often adopt a more sloppy and readable formulation. Section 4 below explains the underlying semantics with greater precision.

Although it is intuitively clear that Julia and John are different individuals, this fact does not logically follow from what we have stated so far. DLs do not make the *unique name assumption*, so different names might refer to the same individual unless explicitly stated otherwise. The *individual inequality* assertion

$$\mathsf{julia} \not\approx \mathsf{john} \tag{3}$$

is used to assert that Julia and John are actually different individuals. On the other hand, an *individual equality* assertion, such as

$$\mathsf{john} \approx \mathsf{johnny}, \tag{4}$$

states that two different names are known to refer to the same individual. Such situations can arise, for example, when combining knowledge about the same domain from several different sources, a task that is known as *ontology alignment*.

### *1.2. Expressing Terminological Knowledge with TBox Axioms*

TBox axioms describe relationships between concepts. For example, the fact that all mothers are parents is expressed by the *concept inclusion*

$$\mathsf{Mother} \sqsubseteq \mathsf{Parent}, \tag{5}$$

in which case we say that the concept Mother is *subsumed* by the concept Parent. Such knowledge can be used to infer further facts about individuals. For example, (1) and (5) together imply that Julia is a parent.

*Concept equivalence* asserts that two concepts have the same instances, as in

$$\mathsf{Person} \equiv \mathsf{Human}. \tag{6}$$

While synonyms are an obvious example of equivalent concepts, in practice one more often uses concept equivalence to give a name to complex expressions as introduced in Section 2.1 below. Furthermore, such additional concept expressions can be combined with equivalence and inclusion to describe more complex situations such as the disjointness of concepts, which asserts that two concepts do not share any instances.

### *1.3. Modelling Relationships between Roles with RBox Axioms*

RBox axioms refer to properties of roles. As for concepts, DLs support *role inclusion* and *role equivalence* axioms. For example, the inclusion

$$\mathsf{parentOf} \sqsubseteq \mathsf{ancestorOf} \tag{7}$$

states that parentOf is a *subrole* of ancestorOf, i.e., every pair of individuals related by parentOf is also related by ancestorOf. Thus (2) and (7) together imply that Julia is an ancestor of John.

In role inclusion axioms, *role composition* can be used to describe roles such as uncleOf. Intuitively, if Charles is a brother of Julia and Julia is a parent of John, then Charles is an uncle of John. This kind of relationship between the roles brotherOf, parentOf and uncleOf is captured by the *complex role inclusion* axiom

$$\mathsf{brotherOf} \circ \mathsf{parentOf} \sqsubseteq \mathsf{uncleOf}. \tag{8}$$

Note that role composition can only appear on the left-hand side of complex role inclusions. Furthermore, in order to retain decidability of reasoning (see the end of Section 4 for a discussion on decidability), complex role inclusions are governed by additional structural restrictions that specify whether or not a collection of such axioms can be used together in one ontology.

Nobody can be both a parent and a child of the same individual, so the two roles parentOf and childOf are disjoint. In DLs we can write *disjoint roles* as follows:

$$\mathit{Disjoint}(\mathsf{parentOf}, \mathsf{childOf}). \tag{9}$$

Further RBox axioms include *role characteristics* such as reflexivity, symmetry and transitivity of roles. These are closely related to a number of other DL features and we will discuss them again in more detail in Section 2.5.

## 2. Constructors for Concepts and Roles

The basic types of axioms introduced in Section 1 are rather limited for accurate modelling. To describe more complex situations, DLs allow new concepts and roles to be built using a variety of different constructors. We distinguish concept and role constructors depending on whether concept or role expressions are constructed. In the case of concepts, one can further separate basic Boolean constructors, role restrictions and nominals/enumerations. At the end of this section, we revisit the additional kinds of RBox axioms that have been omitted in Section 1.3.

### *2.1. Boolean Concept Constructors*

Boolean concept constructors provide basic Boolean operations that are closely related to the familiar operations of intersection, union and complement of sets, or to conjunction, disjunction and negation of logical expressions.

For example, concept inclusions allow us to state that all mothers are female and that all mothers are parents, but what we really mean is that mothers are *exactly* the female parents. DLs support such statements by allowing us to form complex concepts such as the *intersection* (also called *conjunction*)

$$\mathsf{Female} \sqcap \mathsf{Parent}, \tag{10}$$

which represents the set of individuals that are both female and parents. A complex concept can be used in axioms in exactly the same way as an atomic concept, e.g., in the equivalence $\mathsf{Mother} \equiv \mathsf{Female} \sqcap \mathsf{Parent}$.

*Union* (also called *disjunction*) is the dual of intersection. For example, the concept

$$\mathsf{Father} \sqcup \mathsf{Mother} \tag{11}$$

describes those individuals that are either fathers or mothers. Again, it can be used in an axiom such as $\mathsf{Parent} \equiv \mathsf{Father} \sqcup \mathsf{Mother}$, which states that a parent is either a father or a mother (and vice versa).

Sometimes we are interested in individuals that do *not* belong to a certain concept, e.g., in women who are not married. These could be described by the complex concept

$$\mathsf{Female} \sqcap \neg\mathsf{Married}, \tag{12}$$

where the *complement* (also called *negation*) $\neg\mathsf{Married}$ represents the set of all individuals that are not married.

It is sometimes useful to be able to make a statement about every individual, e.g., to say that everybody is either male or female. This can be accomplished by the axiom

$$\top \sqsubseteq \mathsf{Male} \sqcup \mathsf{Female}, \tag{13}$$

where the *top concept* $\top$ is a special concept with every individual as an instance; it can be viewed as an abbreviation for $C \sqcup \neg C$ for an arbitrary concept $C$. Note that this modelling is rather coarse as it presupposes that every individual has a gender, which may not be reasonable for instances of a concept such as Computer. We will see more useful applications for $\top$ later on.

To express that, for the purposes of our modelling, nobody can be both a male and a female at the same time, we can declare the set of male and the set of female individuals to be disjoint. While ontology languages like OWL provide a basic constructor for disjointness, it is naturally captured in DLs with the axiom

$$\mathsf{Male} \sqcap \mathsf{Female} \sqsubseteq \bot, \tag{14}$$

where the *bottom concept* $\bot$ is the dual of $\top$, that is the special concept with no individuals as instances; it can be seen as an abbreviation for $C \sqcap \neg C$ for an arbitrary concept $C$. The above axiom thus says that the intersection of the two concepts is empty.

### *2.2. Role Restrictions*

So far we have seen how to use TBox and RBox axioms to express relationships between concepts and roles, respectively. The most interesting feature of DLs, however, is their ability to form statements that link concepts and roles together. For example, there is an obvious relationship between the concept Parent and the role parentOf, namely, a parent is someone who is a parent of at least one individual. In DLs, this relationship can be captured by the concept equivalence

$$\mathsf{Parent} \equiv \exists \mathsf{parentOf}.\top, \tag{15}$$

where the *existential restriction* $\exists \mathsf{parentOf}.\top$ is a complex concept that describes the set of individuals that are parents of at least one individual (instance of $\top$). Similarly, the concept $\exists \mathsf{parentOf}.\mathsf{Female}$ describes those individuals that are parents of at least one female individual, i.e., those that have a daughter.

To represent the set of individuals all of whose children are female, we use the *universal restriction*

$$\forall \mathsf{parentOf}.\mathsf{Female}. \tag{16}$$

It is a common error to forget that (16) also includes those individuals that have no children at all. More accurately (and less naturally), the axiom can be said to describe the set of all individuals that have "no children other than female ones," i.e., that have "no children that are not female." Following this wording, the concept (16) could indeed be equivalently expressed as $\neg\exists \mathsf{parentOf}.\neg\mathsf{Female}$. If this meaning is not intended, one can describe the individuals who have at least one child and with all their children being female by the concept $(\exists \mathsf{parentOf}.\top) \sqcap (\forall \mathsf{parentOf}.\mathsf{Female})$.

Existential and universal restrictions are useful in combination with the top concept for expressing *domain* and *range restrictions* on roles; that is, restrictions on the kinds of individual that can be in the domain and range of a given role. To restrict the domain of $\mathsf{sonOf}$ to male individuals we can use the axiom

$$\exists \mathsf{sonOf}.\top \sqsubseteq \mathsf{Male}, \tag{17}$$

and to restrict its range to parents we can write

$$\top \sqsubseteq \forall \mathsf{sonOf}.\mathsf{Parent}. \tag{18}$$

In combination with the assertion $\mathsf{sonOf}(\mathsf{john}, \mathsf{julia})$, these axioms would then allow us to deduce that John is male and Julia is a parent. It is interesting to note how this behaviour contrasts with the meaning of *constraints* in databases. Constraints would also allow us to state, e.g., that all sons must be male. However, given only the fact that John is a son of Julia, such a constraint would simply be violated (leading to an error) rather than implying that John is male. Mistaking DL axioms for constraints is a very common source of modelling errors.

*Number restrictions* allow us to restrict the number of individuals that can be reached via a given role. For example, we can form the *at-least restriction*

$$\geqslant 2\, \mathsf{childOf}.\mathsf{Parent} \tag{19}$$

to describe the set of individuals that are children of at least two parents, and the *at-most restriction*

$$\leqslant 2\, \mathsf{childOf}.\mathsf{Parent} \tag{20}$$

for those that are children of at most two parents. The axiom $\mathsf{Person} \sqsubseteq \geqslant 2\, \mathsf{childOf}.\mathsf{Parent} \sqcap \leqslant 2\, \mathsf{childOf}.\mathsf{Parent}$ then states that every person is a child of exactly two parents.

Finally, *local reflexivity* can be used to describe the set of individuals that are related to themselves via a given role. For example, the set of individuals that talk to themselves is described by the concept

$$\exists \mathsf{talksTo}.\mathit{Self}. \tag{21}$$

### 2.3. Nominals

As well as defining concepts in terms of other concepts (and roles), it may also be useful to define a concept by simply enumerating its instances. For example, we might define the concept Beatle by enumerating its instances: john, paul, george, and ringo. Enumerations are not supported natively in DLs, but they can be simulated in DLs using *nominals*. A nominal is a concept that has exactly one instance. For example, {john} is the concept whose only instance is (the individual represented by) john. Combining nominals with union, the enumeration in our example could be expressed as

$$\mathsf{Beatle} \equiv \{\mathsf{john}\} \sqcup \{\mathsf{paul}\} \sqcup \{\mathsf{george}\} \sqcup \{\mathsf{ringo}\}. \tag{22}$$

It is interesting to note that, using nominals, a concept assertion Mother(julia) can be turned into a concept inclusion {julia} $\sqsubseteq$ Mother and a role assertion parentOf(julia, john) into a concept inclusion {julia} $\sqsubseteq \exists$parentOf.{john}. This illustrates that the distinction between ABox and TBox does not have a deeper logical meaning.

### 2.4. Role Constructors

In contrast to the variety of concept constructors, DLs provide only few constructors for forming complex roles. In practice, *inverse roles* are the most important such constructor. Intuitively, the relationship between the roles parentOf and childOf is that, for example, if Julia is a parent of John, then John is a child of Julia and vice versa. More formally, parenfOf is the inverse of childOf, which in DLs can be expressed by the equivalence

$$\mathsf{parentOf} \equiv \mathsf{childOf}^{-}, \tag{23}$$

where the complex role childOf$^{-}$ represents the inverse of childOf.

In analogy to the top concept, DLs also provide the *universal role*, represented by $U$, which always relates all pairs of individuals. It typically plays a minor role in modelling,[2] but it establishes symmetry between roles and concepts w.r.t. a top element. Similarly, an *empty role* that corresponds to the bottom concept is also available in OWL but has rarely been introduced as a constructor in DLs; however, we can define any role $R$ to be empty using the axiom $\top \sqsubseteq \neg\exists R.\top$ ("all things do not relate to anything through $R$"). Interestingly, the universal role cannot be defined by TBox axioms using the constructors introduced above, and in particular universal role restrictions cannot express that a role is universal.

[2] Although there are a few interesting things that could be expressed with $U$, such as *concept products* [17], tool support is rarely sufficient for using this feature in practice.

### 2.5. More RBox Axioms: Role Characteristics

In Section 1.3 we introduced three forms of RBox axioms: role inclusions, role equivalences and role disjointness. OWL provides a variety of others, namely role transitivity, symmetry, asymmetry, reflexivity and irreflexivity. These are sometimes considered as basic axiom types in DLs as well, using some suggestive notation such as *Trans*(ancestorOf) to express that the role ancestorOf is transitive. However, such axioms are just syntactic sugar; all role characteristics can be expressed using the features of DLs that we have already introduced.

*Transitivity* is a special form of complex role inclusion. For example, transitivity of ancestorOf can be captured by the axiom $\mathsf{ancestorOf} \circ \mathsf{ancestorOf} \sqsubseteq \mathsf{ancestorOf}$. A role is *symmetric* if it is equivalent to its own inverse, e.g., $\mathsf{marriedTo} \equiv \mathsf{marriedTo}^-$, and it is *asymmetric* if it is disjoint from its own inverse, as in *Disjoint*(parentOf, $\mathsf{parentOf}^-$). If desired, *global reflexivity* can be expressed by imposing local reflexivity on the top concept as in $\top \sqsubseteq \exists \mathsf{knows}.\mathit{Self}$. A role is *irreflexive* if it is never locally reflexive, as in the case of $\top \sqsubseteq \neg\exists \mathsf{marriedTo}.\mathit{Self}$.

## 3. The Description Logic $\mathcal{SROIQ}$

In this section, we summarise the various features that have been introduced informally above to provide a comprehensive definition of DL syntax. Doing so yields the description logic called $\mathcal{SROIQ}$, which is one of the most expressive DLs commonly considered today. It also largely agrees in expressivity with the ontology language OWL 2 DL, though there are still some differences as explained in Section 6.

Formally, every DL ontology is based on three finite sets of signature symbols: a set $\mathsf{N}_I$ of *individual names*, a set $\mathsf{N}_C$ of *concept names* and a set $\mathsf{N}_R$ of *role names*. Usually these sets are assumed to be fixed for some application and are therefore not mentioned explicitly. Now the set of $\mathcal{SROIQ}$ *role expressions* $\mathbf{R}$ (over this signature) is defined by the following grammar:

$$\mathbf{R} ::= U \mid \mathsf{N}_R \mid \mathsf{N}_R{}^-$$

where $U$ is the universal role (Section 2.4). Based on this, the set of $\mathcal{SROIQ}$ *concept expressions* $\mathbf{C}$ is defined as:

$$\mathbf{C} ::= \mathsf{N}_C \mid (\mathbf{C} \sqcap \mathbf{C}) \mid (\mathbf{C} \sqcup \mathbf{C}) \mid \neg\mathbf{C} \mid \top \mid \bot \mid \exists \mathbf{R}.\mathbf{C} \mid \forall \mathbf{R}.\mathbf{C} \mid \geqslant n\, \mathbf{R}.\mathbf{C} \mid \leqslant n\, \mathbf{R}.\mathbf{C} \mid \exists \mathbf{R}.\mathit{Self} \mid \{\mathsf{N}_I\}$$

where $n$ is a non-negative integer. As usual, expressions like $(\mathbf{C} \sqcap \mathbf{C})$ represent any expression of the form $(C \sqcap D)$ with $C, D \in \mathbf{C}$. It is common to omit parentheses if this cannot lead to confusion with expressions of different semantics. For example, parentheses do not matter for $A \sqcup B \sqcup C$ whereas the expressions $A \sqcap B \sqcup C$ and $\exists R.A \sqcap B$ are ambiguous.

Using the above sets of individual names, roles and concepts, the *axioms* of $\mathcal{SROIQ}$ can be defined to be of the following basic forms:

| | | | | |
|---|---|---|---|---|
| ABox: | $\mathbf{C}(\mathsf{N}_I)$ | $\mathbf{R}(\mathsf{N}_I, \mathsf{N}_I)$ | $\mathsf{N}_I \approx \mathsf{N}_I$ | $\mathsf{N}_I \not\approx \mathsf{N}_I$ |
| TBox: | $\mathbf{C} \sqsubseteq \mathbf{C}$ | $\mathbf{C} \equiv \mathbf{C}$ | | |
| RBox: | $\mathbf{R} \sqsubseteq \mathbf{R}$ | $\mathbf{R} \equiv \mathbf{R}$ | $\mathbf{R} \circ \mathbf{R} \sqsubseteq \mathbf{R}$ | $\mathit{Disjoint}(\mathbf{R}, \mathbf{R})$ |

with the intuitive meanings as explained in Section 1 and 2.

Roughly speaking, a $\mathcal{SROIQ}$ ontology (or *knowledge base*) is simply a set of such axioms. To ensure the existence of reasoning algorithms that are correct and terminating, however, additional syntactic restrictions must be imposed on ontologies. These restrictions refer not to single axioms but to the structure of the ontology as a whole, hence they are called *structural restrictions*. The two such conditions relevant for $\mathcal{SROIQ}$ are based on the notions of *simplicity* and *regularity*. Notably, both are automatically satisfied for ontologies that do not contain complex role inclusion axioms.

A role $R$ in an ontology $\mathcal{O}$ is called *non-simple* if some complex role inclusion axiom (i.e., one that uses role composition $\circ$) in $\mathcal{O}$ implies instances of $R$; otherwise it is called *simple*. A more precise definition of the non-simple role expressions of the ontology $\mathcal{O}$ is given by the following rules:

- if $\mathcal{O}$ contains an axiom $S \circ T \sqsubseteq R$, then $R$ is non-simple,
- if $R$ is non-simple, then its inverse $R^-$ is also non-simple,[3]
- if $R$ is non-simple and $\mathcal{O}$ contains any of the axioms $R \sqsubseteq S$, $S \equiv R$ or $R \equiv S$, then $S$ is also non-simple.

All other roles are called simple.[4] Now for a $\mathcal{SROIQ}$ ontology it is required that the following axioms and concepts contain simple roles only:

| | | | |
|---|---|---|---|
| Restricted axioms: | $\mathit{Disjoint}(\mathbf{R}, \mathbf{R})$ | | |
| Restricted concept expressions: | $\exists \mathbf{R}.\mathit{Self}$ | $\geqslant n\, \mathbf{R}.\mathbf{C}$ | $\leqslant n\, \mathbf{R}.\mathbf{C}$. |

The other structural restriction that is relevant for $\mathcal{SROIQ}$ is called *regularity* and is concerned with RBox axioms only. Roughly speaking, the restriction ensures that cyclic dependencies between complex role inclusion axioms occur only in a limited form. For details, please see the pointers given in Section 7. For the introductory treatment in this chapter, it suffices to note that regularity, just like simplicity, is a property of the ontology as a whole that cannot be checked for each axiom individually. An important practical consequence is that the union of two regular ontologies may no longer be regular. This must be taken into account when merging ontologies in practice.

## 4. Description Logic Semantics

The formal meaning of DL axioms is given by their model-theoretic semantics. In particular, the semantics specifies what the logical consequences of an ontology are. The formal semantics is therefore the main guideline for every tool that computes logical consequences of DL ontologies, and a basic understanding of its working is vital to make

[3] If $R = S^-$ already is an inverse role, then $R^-$ should be read as $S$. We do not allow expressions like $S^{--}$.

[4] Whether the universal role $U$ is simple or not is a matter of preference that does not affect the computational properties of the logic [18]. However, the universal role in OWL 2 is considered non-simple.

reasonable modelling choices and to comprehend the results given by software applications. Luckily, the semantics of description logics is not difficult to understand provided that some common misconceptions are avoided.

Intuitively speaking, an ontology describes a particular situation in a given domain of discourse. For example, the axioms in Sections 1 and 2 describe a particular situation in the "families and relationships" domain. However, ontologies usually cannot fully specify the situation that they describe. On the one hand, there is no formal relationship between the symbols we use and the objects that they represent: the individual name julia, for example, is just a syntactic identifier with no intrinsic meaning. Indeed, the intended meaning of the identifiers in our ontologies has no influence on their formal semantics: what we know about them stems only from the ontological axioms. On the other hand, the axioms in an ontology typically do not provide complete information. For example, (3) and (4) in Section 1.1 state that some individuals are equal and that others are unequal, but in many other cases this information might be left unspecified.

Description logics have been designed to deal with such incomplete information. Rather than making default assumptions in order to fully specify one particular interpretation for each ontology, the DL semantics generally considers all the possible situations (i.e., states of the world) where the axioms of an ontology would hold (we also say: where the axioms are *satisfied*). This characteristic is sometimes called the *Open World Assumption* since it keeps unspecified information open.[5] A logical consequence of an ontology is an axiom that holds in all interpretations that satisfy the ontology, i.e., something that is true in all conceivable states of the world that agree with what is said in the ontology. The more axioms an ontology contains, the more specific are the constraints that it imposes on possible interpretations, and the fewer interpretations exist that satisfy all of the axioms. Conversely, if fewer interpretations satisfy an ontology, then more axioms hold in all of them, and more logical consequences follow from the ontology. The previous two sentences imply that the semantics of description logics is *monotonic*: additional axioms always lead to additional consequences, or, more informally, the more knowledge we feed into a DL system the more results it returns.

An extreme case is when an ontology is not satisfied in any interpretation. The ontology is then called *unsatisfiable* or *inconsistent*. In this case *every* axiom holds vacuously in all of the (zero) interpretations that satisfy the ontology. Such an ontology is clearly of no utility, and avoiding inconsistency (and checking for it in the first place) is therefore an important task during modelling.

We have outlined above the most important ideas of DL semantics. What remains to be done is to define what we really mean by an "interpretation" and which conditions must hold for particular axioms to be satisfied by an interpretation. For this, we closely follow the intuitive ideas established above: an interpretation $\mathcal{I}$ consists of a set $\Delta^{\mathcal{I}}$ called the *domain* of $\mathcal{I}$ and an interpretation function $\cdot^{\mathcal{I}}$ that maps each atomic concept $A$ to a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, each atomic role $R$ to a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and each individual name $a$ to an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. The interpretation of complex concepts and roles follows from the interpretation of the basic entities. Table 1 shows how to obtain the semantics of each compound expression from the semantics of its parts. By "$R^{\mathcal{I}}$-successor of $x$" we mean any individual $y$ such that $\langle x, y\rangle \in R^{\mathcal{I}}$. The definition should confirm the intuitive

[5] A *Closed World Assumption* "closes" the interpretation by assuming that every fact not explicitly stated to be true is actually false. Both terms are not formally specified and rather outline the general flavour of a semantics than any particular definition.

**Table 1.** Syntax and semantics of $\mathcal{SROIQ}$ constructors

| | Syntax | Semantics |
|---|---|---|
| *Individuals:* | | |
| individual name | $a$ | $a^{\mathcal{I}}$ |
| *Roles:* | | |
| atomic role | $R$ | $R^{\mathcal{I}}$ |
| inverse role | $R^{-}$ | $\{\langle x, y\rangle \mid \langle y, x\rangle \in R^{\mathcal{I}}\}$ |
| universal role | $U$ | $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ |
| *Concepts:* | | |
| atomic concept | $A$ | $A^{\mathcal{I}}$ |
| intersection | $C \sqcap D$ | $C^{\mathcal{I}} \cap D^{\mathcal{I}}$ |
| union | $C \sqcup D$ | $C^{\mathcal{I}} \cup D^{\mathcal{I}}$ |
| complement | $\neg C$ | $\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$ |
| top concept | $\top$ | $\Delta^{\mathcal{I}}$ |
| bottom concept | $\bot$ | $\emptyset$ |
| existential restriction | $\exists R.C$ | $\{x \mid$ some $R^{\mathcal{I}}$-successor of $x$ is in $C^{\mathcal{I}}\}$ |
| universal restriction | $\forall R.C$ | $\{x \mid$ all $R^{\mathcal{I}}$-successors of $x$ are in $C^{\mathcal{I}}\}$ |
| at-least restriction | $\geqslant n\, R.C$ | $\{x \mid$ at least $n$ $R^{\mathcal{I}}$-successors of $x$ are in $C^{\mathcal{I}}\}$ |
| at-most restriction | $\leqslant n\, R.C$ | $\{x \mid$ at most $n$ $R^{\mathcal{I}}$-successors of $x$ are in $C^{\mathcal{I}}\}$ |
| local reflexivity | $\exists R.Self$ | $\{x \mid \langle x, x\rangle \in R^{\mathcal{I}}\}$ |
| nominal | $\{a\}$ | $\{a^{\mathcal{I}}\}$ |
| where $a, b \in \mathsf{N}_I$ are individual names, $A \in \mathsf{N}_C$ is a concept name, $C, D \in \mathbf{C}$ are concepts, $R \in \mathbf{R}$ is a role | | |

**Table 2.** Syntax and semantics of $\mathcal{SROIQ}$ axioms

| | Syntax | Semantics |
|---|---|---|
| *ABox:* | | |
| concept assertion | $C(a)$ | $a^{\mathcal{I}} \in C^{\mathcal{I}}$ |
| role assertion | $R(a, b)$ | $\langle a^{\mathcal{I}}, b^{\mathcal{I}}\rangle \in R^{\mathcal{I}}$ |
| individual equality | $a \approx b$ | $a^{\mathcal{I}} = b^{\mathcal{I}}$ |
| individual inequality | $a \not\approx b$ | $a^{\mathcal{I}} \neq b^{\mathcal{I}}$ |
| *TBox:* | | |
| concept inclusion | $C \sqsubseteq D$ | $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ |
| concept equivalence | $C \equiv D$ | $C^{\mathcal{I}} = D^{\mathcal{I}}$ |
| *RBox:* | | |
| role inclusion | $R \sqsubseteq S$ | $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$ |
| role equivalence | $R \equiv S$ | $R^{\mathcal{I}} = S^{\mathcal{I}}$ |
| complex role inclusion | $R_1 \circ R_2 \sqsubseteq S$ | $R_1^{\mathcal{I}} \circ R_2^{\mathcal{I}} \subseteq S^{\mathcal{I}}$ |
| role disjointness | $\mathit{Disjoint}(R, S)$ | $R^{\mathcal{I}} \cap S^{\mathcal{I}} = \emptyset$ |

explanations given for each case in Section 2. For example, the semantics of Female ⊓ Parent is indeed the intersection of the semantics of Female and Parent.

Since an interpretation $\mathcal{I}$ fixes the meaning of all entities, we can unambiguously say for each axiom whether it holds in $\mathcal{I}$ or not. An axiom $\alpha$ *holds* in $\mathcal{I}$ (we also say $\mathcal{I}$ *satisfies* $\alpha$ and write $\mathcal{I} \models \alpha$) if the corresponding condition in Table 2 is met. Again, these definitions fully agree with the intuitive explanations given in Section 1. If all axioms in an ontology $\mathcal{O}$ hold in $\mathcal{I}$ (i.e., if $\mathcal{I}$ satisfies $\mathcal{O}$, written $\mathcal{I} \models \mathcal{O}$), then $\mathcal{I}$ is a *model*

of $\mathcal{O}$. Thus a model is an abstraction of a state of the world that satisfies all axioms in the ontology. An ontology is *consistent* if it has at least one model. An axiom $\alpha$ is a *consequence* of an ontology $\mathcal{O}$ (or $\mathcal{O}$ *entails* $\alpha$, written $\mathcal{O} \models \alpha$) if $\alpha$ holds in every model of $\mathcal{O}$. In particular, an inconsistent ontology entails every axiom.

A noteworthy consequence of this semantics is the meaning of individual names in DL ontologies. We already remarked that DLs do not usually make the Unique Name Assumption, and indeed our formal definition allows two individual names to be interpreted as the same individual (element of the domain). Possibly even more important is the fact that the domain of an interpretation is allowed to contain many individuals that are not represented by any individual name. A common confusion in modelling arises from the implicit assumption that interpretations must only contain individuals that are represented by individual names (such individuals are also called *named individuals*). For example, one could wrongly assume the ontology consisting of the axioms

$$\mathsf{parentOf}(\mathsf{julia}, \mathsf{john}) \qquad \mathsf{manyChildren}(\mathsf{julia}) \qquad \mathsf{manyChildren} \sqsubseteq \geqslant 3\, \mathsf{parentOf}.\top$$

to be inconsistent since it requires Julia to have at least 3 children when only one (John) is given. However, there are many conceivable models where Julia does have three children, even though only one of the children is explicitly named. A significant number of modelling errors can be traced back to similar misconceptions that are easy to prevent if the general open world assumption of DLs is kept in mind.

Another point to note is that the above specification of the semantics does not provide any hint as to how to compute the relevant entailments in practical software tools. There are infinitely many possible interpretations, each of which may have an infinite domain (in fact there are some ontologies that are satisfied only by interpretations with infinite domains). Therefore it is impossible to test all interpretations to see if they model a given ontology, and impossible to test all models of an ontology to see if they entail a given axiom. Rather, one has to devise deduction procedures and prove their correctness with respect to the above specification. The interplay of certain expressive features can make reasoning algorithms more complicated and in some cases it can even be shown that no correct and terminating algorithm exists at all (i.e., that reasoning is undecidable). For our purposes it suffices to know that entailment of axioms is decidable for $\mathcal{SROIQ}$ (with the structural restrictions explained in Section 3) and that a number of free and commercial tools are available. Such tools are typically optimised for more specific reasoning problems, such as consistency checking, the entailment of concept subsumptions (subsumption checking) or of concept assertions (instance checking). Many of these standard inferencing problems can be expressed in terms of each other, so they can be handled by very similar reasoning algorithms.

## 5. Important Fragments of $\mathcal{SROIQ}$

Many different description logics have been introduced in the literature. Typically, they can be characterised by the types of constructors and axioms that they allow, which are often a subset of the constructors in $\mathcal{SROIQ}$. For example, the description logic $\mathcal{ALC}$ is the fragment of $\mathcal{SROIQ}$ that allows no RBox axioms and only $\sqcap$, $\sqcup$, $\neg$, $\exists$ and $\forall$ as its concept constructors. The extension of $\mathcal{ALC}$ with transitive roles is traditionally denoted

by the letter $\mathcal{S}$. Some other letters used in DL names hint at a particular constructor, such as inverse roles $\mathcal{I}$, nominals $\mathcal{O}$, qualified number restrictions $\mathcal{Q}$, and role hierarchies (role inclusion axioms without composition) $\mathcal{H}$. So, for example, the DL named $\mathcal{ALCHIQ}$ extends $\mathcal{ALC}$ with role hierarchies, inverse roles and qualified number restrictions. The letter $\mathcal{R}$ most commonly refers to the presence of role inclusions, local reflexivity *Self*, and the universal role $U$, as well as the additional role characteristics of transitivity, symmetry, asymmetry, role disjointness, reflexivity, and irreflexivity. This naming scheme explains the name $\mathcal{SROIQ}$.

In recent years, fragments of DLs have been specifically developed in order to obtain favourable computational properties. For this purpose, $\mathcal{ALC}$ is already too large, since it only admits reasoning algorithms that run in worst-case exponential time. More lightweight DLs can be obtained by further restricting expressivity, while at the same time a number of additional $\mathcal{SROIQ}$ features can be added without loosing the good computational properties. The three main approaches for obtaining lightweight DLs are $\mathcal{EL}$, *DLP* and *DL-Lite*, which also correspond to language fragments OWL EL, OWL RL and OWL QL of the Web Ontology Language.

The $\mathcal{EL}$ family of description logics is characterised by allowing unlimited use of existential quantifiers and concept intersection. The original description logic $\mathcal{EL}$ allows only those features and $\top$ but no unions, complements or universal quantifiers, and no RBox axioms. Further extensions of this language are known as $\mathcal{EL}^+$ and $\mathcal{EL}^{++}$. The largest such extension allows the constructors $\sqcap$, $\top$, $\bot$, $\exists$, *Self*, nominals and the universal role, and it supports all types of axioms other than role symmetry, asymmetry and irreflexivity. Interestingly, all standard reasoning tasks for this DL can still be solved in worst-case polynomial time. One can even drop the structural restriction of regularity that is important for $\mathcal{SROIQ}$. $\mathcal{EL}$ has been used to model large but lightweight ontologies that consist mainly of terminological data, in particular in the life sciences. A number of reasoners are specifically optimised for handling $\mathcal{EL}$-type ontologies, the most recent of which is the ELK reasoner for OWL EL.[6]

DLP is short for *Description Logic Programs* and comprises various DLs that are syntactically restricted in such a way that axioms could also be read as rules in first-order Horn logic without function symbols. Due to this, DLP-type logics can be considered as kinds of rule languages (hence the name OWL RL) contained in DLs. To accomplish this, one has to allow different syntactic forms for subconcepts and superconcepts in concept inclusion axioms. We do not provide the details here. While DLs in general may require us to consider domain elements that are not represented by individual names, for DLP one can always restrict attention to models in which all domain elements are represented by individual names. This is why DLP is often used to augment databases (interpreted as sets of ABox axioms), e.g., in an implementation of OWL RL in the Oracle 11g database management system.

DL-Lite is a family of DLs that is also used in combination with large data collections and existing databases, in particular to augment the expressivity of a query language that retrieves such data. This approach, known as Ontology Based Data Access, considers ontologies as a language for constructing *views* or *mapping rules* on top of existing data. The core feature of DL-Lite is that data access can be realised with standard query languages such as SQL that are not aware of the DL semantics. Ontological

[6] `http://elk-reasoner.googlecode.com/`

information is merely used in a query preprocessing step. Like DLP, DL-Lite requires different syntactic restrictions for subconcepts and superconcepts. We do not present the details here.

## 6. Relationship to OWL

The *OWL Web Ontology Language* is a knowledge representation language standardised by the World Wide Web Consortium (W3C). OWL is one of the most important applications of description logics today. In this section, we briefly outline the relationship of the two languages. A comprehensive treatment is beyond the scope of this chapter; see Section 7 for pointers to further reading. The current version of the OWL specification is OWL 2 as standardised in 2009. This supersedes the earlier OWL 1 standard of 2004.

The main building blocks of OWL are indeed very similar to those of DLs, with the main difference that concepts are called *classes* and roles are called *properties*. It is therefore not surprising that description logics have had a major influence on the development of OWL and the expressive features that it provides. Historically, however, OWL has also been conceived as an extension to RDF, a Web data modelling language whose expressivity is comparable to DL ABoxes. The formal semantics of RDF is subtly different from that of DLs, even though both lead to the same consequences in many common cases. Extending the RDF semantics to the expressive features of OWL improves the compatibility between the two, but it also makes reasoning undecidable. Therefore, it has been decided to specify both styles of formal semantics for OWL: the *Direct Semantics* based on DLs and the *RDF-based Semantics*.

In this section, we are therefore mainly interested in the Direct Semantics of OWL. This semantics is only defined for OWL ontologies that abide by certain syntactic restrictions (essentially the restriction that the OWL axioms can be read as $\mathcal{SROIQ}$ axioms for which the structural restrictions of Section 3 are satisfied). This syntactic fragment of OWL is called *OWL DL*.[7] Under the Direct Semantics, large parts of OWL DL can indeed be considered as a syntactic variant of $\mathcal{SROIQ}$. For example, the axiom $\mathsf{Mother} \equiv \mathsf{Female} \sqcap \mathsf{Parent}$ would be written as follows in OWL:

```
EquivalentClasses( Mother ObjectIntersectionOf( Female Parent ) )
```

where the symbols Mother, Female and Parent would be identifier strings that conform to the OWL specification.[8] The above example illustrates the close relationship between the syntax of $\mathcal{SROIQ}$ and that of OWL. In many cases, it is indeed enough to translate an operator symbol of $\mathcal{SROIQ}$ into the corresponding operator name in OWL, which is then written in prefix notation like a function. This is also why the above form of syntax is called *Functional-Style Syntax*. The OWL standard provides a number of syntactic forms that can be used to express OWL ontologies. The most prominent among these is the RDF/XML serialisation since it is the only format that all conforming OWL tools

[7] In contrast, the OWL language without any syntactic constraints is called *OWL Full*. It comprises ontologies that can only be interpreted under the RDF-based Semantics.

[8] Entity names in OWL are generally based on Uniform Resource Identifiers (URIs). The details are not relevant here. Additional information can be found in the chapter on linked data included in this volume [10].

need to understand. On the other hand, it is more difficult for humans to read and we do not present it here.

It is interesting to note that there are still a few differences between OWL DL under the Direct Semantics and $\mathcal{SROIQ}$. On a syntactic level, OWL provides a lot more operators that, though logically redundant, can be convenient as shortcuts for compound DL axioms. For example, OWL has special constructs for specifying domain and range of a property, even though these could equally well be expressed as in Section 2.2. These kinds of features also include the empty (bottom) property, which can easily be defined but is not included as a language feature in DLs.

However, OWL also includes some expressive features that we did not include in our treatment of $\mathcal{SROIQ}$ above. Most notably, this includes support for datatypes and datatype literals. These behave like classes and individual names but come with a fixed, pre-defined interpretation. For example, the datatype for Boolean values has exactly two elements – true and false – in any interpretation. This can also be introduced in DLs by so-called *concrete domains*, i.e., pre-defined interpretation domains. Both DLs and OWL in this case strictly distinguish roles/properties that relate to "abstract" individuals from those that relate to values from some datatype. In OWL, the constructs that relate to datatypes include "Data" in their name while constructs that relate to abstract individuals include "Object." For example, OWL distinguishes `ObjectIntersectionOf` (used above) from `DataIntersectionOf` (the intersection of datatypes).

The only other logical feature that is missing in DLs are so-called *Keys*. These are special forms of rules that can be used for data integration. Roughly speaking, a key specifies that two named individuals are entailed to be equal if they agree on certain property values and class memberships, similar to key constraints in databases. For example, the combination of nationality and registration number might be treated as a key for (i.e., sufficient to uniquely identify) motor vehicles.

Besides the logical features, OWL also includes a number of other aspects that are not considered in description logics at all. For example, it includes means of naming an ontology and of importing ontological axioms from one ontology into another. Further extra-logical features include a simple form of *meta-modelling* called *punning*, non-logical axioms to *declare* identifiers, and the possibility to add *annotations* to arbitrary axioms and entities similar to comments in a programming language.

## 7. Further Reading

This chapter can only provide a first introduction to description logics and OWL. More detailed introductory texts can be found in the lecture notes of the Reasoning Web Summer School: Rudolph provides a detailed discussion of DL semantics and modelling [16], Baader gives a general overview with extended historical notes [1], and Sattler focusses on tableau-based reasoning methods [19]. An extensive introduction to lightweight description logics is given by Krötzsch [11].

For a more detailed coverage of OWL and its relationship to DL, we recommend the textbook *Foundations of Semantic Web Technologies* [8]. This introductory text also treats the relationship of DLs to first-order logic, DL query answering and extensions for rule-based modelling (related to keys in OWL), which we have omitted here. An in-depth treatment of description logics and related research topics is provided by the *Description*

*Logic Handbook* [3], which also covers interesting aspects of deduction algorithms and computational complexity that are beyond the scope of this chapter.

A number of research papers focus on specific topics in DLs. Closely related to this chapter is the original article on $\mathcal{SROIQ}$, which also provides the details on regularity conditions that have been skipped above [9]. A detailed discussion of OWL datatypes and their description logic semantics is given by Motik and Horrocks [14]. There are also various works that focus on $\mathcal{EL}$ [2,12], DLP/OWL RL [6,13] and DL-Lite [4]. Current developments in DL research are discussed at the annual DL Workshop[9] and at the major Semantic Web and Artificial Intelligence conferences.

The primary resources on OWL 2 are the online documents of the specification [15] where the OWL Primer provides a first introduction [7]. The differences of the 2009 OWL 2 standard to its predecessor are explained in [5].

Many related tools such as reasoners and ontology editors are available. The most popular free ontology editor is Protégé,[10] which can be used with a variety of OWL reasoners. Pointers to current OWL reasoners are best found online.[11] Popular systems for large parts of OWL 2 DL ($\mathcal{SROIQ}$) include FaCT++, HermiT, Pellet and RacerPro. Some typical lightweight systems are ELK (OWL EL), jCEL (OWL EL), Owlgress (OWL QL), OWLIM (OWL RL and QL), Quonto (OWL QL) and Snorocket (OWL EL). Details about these tools and related publications can be found on the respective homepages.

*Acknowledgements* We thank Fernando Bobillo, Peter Patel-Schneider and Evgeny Zolin for helpful comments on an earlier version of this text.

## References

[1] Franz Baader. Description logics. In Sergio Tessaris, Enrico Franconi, Thomas Eiter, Claudio Gutierrez, Siegfried Handschuh, Marie-Christine Rousset, and Renate A. Schmidt, editors, *Reasoning Web. Semantic Technologies for Information Systems – 5th International Summer School, 2009*, volume 5689 of *LNCS*, pages 1–39. Springer, 2009. Available at `http://lat.inf.tu-dresden.de/research/papers.html`.

[2] Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ envelope. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *Proc. 19th Int. Joint Conf. on Artificial Intelligence (IJCAI'05)*, pages 364–369. Professional Book Center, 2005.

[3] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, second edition, 2007.

[4] Diego Calvanese, Guiseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *J. of Automated Reasoning*, 39(3):385–429, 2007.

[5] Bernardo Cuenca Grau, Ian Horrocks, Boris Motik, Bijan Parsia, Peter Patel-Schneider, and Ulrike Sattler. OWL 2: The next step for OWL. *J. of Web Semantics*, 6:309–322, 2008.

[6] Benjamin N. Grosof, Ian Horrocks, Raphael Volz, and Stefan Decker. Description logic programs: combining logic programs with description logic. In *Proc. 12th Int. Conf. on World Wide Web (WWW'03)*, pages 48–57. ACM, 2003.

[9] See `http://dl.kr.org/` for proceedings.

[10] `http://protege.stanford.edu/`

[11] A list of reasoners can be found, e.g., at `http://semanticweb.org/wiki/Category:Reasoner`.

[7] Pascal Hitzler, Markus Krötzsch, Bijan Parsia, Peter F. Patel-Schneider, and Sebastian Rudolph, editors. *OWL 2 Web Ontology Language: Primer*. W3C Recommendation, 27 October 2009. Available at `http://www.w3.org/TR/owl2-primer/`.

[8] Pascal Hitzler, Markus Krötzsch, and Sebastian Rudolph. *Foundations of Semantic Web Technologies*. Chapman & Hall/CRC, 2009.

[9] Ian Horrocks, Oliver Kutz, and Ulrike Sattler. The even more irresistible $\mathcal{SROIQ}$. In Patrick Doherty, John Mylopoulos, and Christopher A. Welty, editors, *Proc. 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'06)*, pages 57–67. AAAI Press, 2006.

[10] Anja Jentzsch, Ricardo Usbeck, and Denny Vrandečić. An incomplete and simplifying introduction to linked data. In Jens Lehmann and Johanna Völker, editors, *Perspectives on Ontology Learning*, Studies on the Semantic Web. AKA Heidelberg / IOS Press, 2014.

[11] Markus Körtzsch. OWL 2 Profiles: An introduction to lightweight ontology languages. In Thomas Eiter and Thomas Krennwallner, editors, *Proceedings of the 8th Reasoning Web Summer School, Vienna, Austria, September 3–8 2012*, volume 7487 of *LNCS*, pages 112–183. Springer, 2012. Available at `http://korrekt.org/page/OWL_2_Profiles`.

[12] Markus Krötzsch. Efficient rule-based inferencing for OWL EL. In Toby Walsh, editor, *Proc. 22nd Int. Conf. on Artificial Intelligence (IJCAI'11)*, pages 2668–2673. AAAI Press/IJCAI, 2011.

[13] Markus Krötzsch. The not-so-easy task of computing class subsumptions in OWL RL. In Philippe Cudré-Mauroux, Jeff Heflin, Evren Sirin, Tania Tudorache, Jérôme Euzenat, Manfred Hauswirth, Josiane Xavier Parreira, Jim Hendler, Guus Schreiber, Abraham Bernstein, and Eva Blomqvist, editors, *Proc. 11th Int. Semantic Web Conf. (ISWC'12)*, volume 7649 of *LNCS*, pages 279–294. Springer, 2012.

[14] Boris Motik and Ian Horrocks. OWL datatypes: Design and implementation. In Amit Sheth, Steffen Staab, Mike Dean, Massimo Paolucci, Diana Maynard, Timothy Finin, and Krishnaprasad Thirunarayan, editors, *Proc. 7th Int. Semantic Web Conf. (ISWC'08)*, volume 5318 of *LNCS*, pages 307–322. Springer, 2008.

[15] W3C OWL Working Group. *OWL 2 Web Ontology Language: Document Overview*. W3C Recommendation, 27 October 2009. Available at `http://www.w3.org/TR/owl2-overview/`.

[16] Sebastian Rudolph. Foundations of description logics. In Axel Polleres, Claudia d'Amato, Marcelo Arenas, Siegfried Handschuh, Paula Kroner, Sascha Ossowski, and Peter F. Patel-Schneider, editors, *Reasoning Web. Semantic Technologies for the Web of Data – 7th International Summer School 2011*, volume 6848 of *LNCS*, pages 76–136. Springer, 2011. Available at `http://www.aifb.kit.edu/web/Incollection3026/en`.

[17] Sebastian Rudolph, Markus Krötzsch, and Pascal Hitzler. All elephants are bigger than all mice. In Franz Baader, Carsten Lutz, and Boris Motik, editors, *Proc. 21st Int. Workshop on Description Logics (DL'08)*, volume 353 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.

[18] Sebastian Rudolph, Markus Krötzsch, and Pascal Hitzler. Cheap Boolean role constructors for description logics. In Steffen Hölldobler, Carsten Lutz, and Heinrich Wansing, editors, *Proc. 11th European Conf. on Logics in Artificial Intelligence (JELIA'08)*, volume 5293 of *LNAI*, pages 362–374. Springer, 2008.

[19] Ulrike Sattler. Reasoning in description logics: Basics, extensions, and relatives. In Grigoris Antoniou, Uwe Aßmann, Cristina Baroglio, Stefan Decker, Nicola Henze, Paula-Lavinia Patranjan, and Robert Tolksdorf, editors, *Reasoning Web – 3rd International Summer School, 2007*, volume 4636 of *LNCS*, pages 154–182. Springer, 2007.

# An Incomplete and Simplifying Introduction to Linked Data

Anja JENTZSCH [a], Ricardo USBECK [c] and Denny VRANDEČIĆ [b]
[a] *Hasso-Plattner-Institut, Potsdam, Germany;*
*E-mail: Anja.Jentzsch@hpi.uni-potsdam.de*
[b] *Wikimedia Deuschland, Germany;*
[c] *Agile Knowledge Engineering and Semantic Web (AKSW), Leipzig University, Leipzig, Germany; E-mail: ricardo.usbeck@informatik.uni-leipzig.de*

**Abstract.** Linked Data is becoming an increasingly popular method to publish data. It is based on a simple set of rules and a number of widely adopted standards. This introductory article aims at capturing the simplicity of Linked Data, and the concepts behind the related standards. It will help the reader to understand the basic elements of Linked Data and its related technologies. Using the example of a calendar app, we will highlight the practical advantages of using Linked Data for writing apps and publishing data in a completely decoupled way.

## 1. Introduction

Linked Data is a method to publish data on the Web, so that it can be connected to other datasets and create a Web of Data. Four rules need to be followed in order to publish data as Linked Data [7]:

1. Use URIs to name things
2. Use HTTP URIs so that they can be looked up
3. On lookup, provide useful information in standard formats
4. Provide links to other things

This chapter will describe the standards around Linked Data and their basic concepts. Instead of focusing on completeness and technical correctness, we want to achieve an intuitive understanding in the interested but so far uninformed reader. We refer to more comprehensive primers and the technical specifications in each section, so that the reader interested in more details can further deepen their knowledge.

Prerequisites for this chapter are merely a basic understanding of how the Web works. It would be helpful if the reader was exposed to some basic ideas from data management, data structures, and knowledge representation.

Throughout the chapter we will use the example of writing a calendar application. Our app will be able to deal with different time zones: if the user enters a meeting like *"Meeting at 9am in Atlanta"* the calendar tool should be able to

translate the time to the local time zone, even if the meeting is being entered in Istanbul and shared with participants in Athens.

The chapter is structured as follows: Section 2 explains the connection between Linked Data and the Semantic Web Vision. Section 3 introduces the basic notions of RDF, especially the idea of an RDF graph and RDF triples. Section 4 describes URIs, the basic building blocks for such triples, and how they refer to the Web. Section 5 then discusses how the Web is used for the distributed publishing of knowledge using the HTTP standard. Secton 6 extends the very simple expressivity of RDF, and discusses how languages like RDFS, OWL, and RIF are layered on top of it. Section 7 introduces SPARQL, a query language for RDF graphs. In Section 8 we then go into more detail about the question how RDF is and can be serialized, before we devote Section 9 to the sometimes confusing relationship of RDF and XML. We close in Section 10 with a resumé on the state of Linked Data and an outlook on current developments.

## 2. Vision of the Semantic Web

Linked Data and *Semantic Web* are sometimes used synonymously, although these two are rather different concepts. The Semantic Web [36] is a movement from the `Web of Documents` to the `Web of Data`. The intention is to develop a Web that enables humans as well as machines to interact and work on existing data in way that creates an additional value. To achieve this the data needs to be machine understandable, human manageable, linkable and storable in a standardized way. Linked Data is the materialized implementation of the Semantic Web vision, executed by technologies like RDF, SPARQL, ontologies and many more decribed below.

**Further readings**: To get to know more about the Semantic Web vision we refer to the book [2] or the W3C Semantic Web website at `http://www.w3.org/standards/semanticweb/`.

## 3. RDF: The basics

RDF is a standardized model used for representing knowledge on the Web. Knowledge representation in general enables to separate what a system does from what it knows [12], i.e., to make its knowledge explicit and not simply implicit in the program. Decoupling knowledge from a program has several advantages, for example, it allows the knowledge base to be maintained externally, so that changes in the world may not even affect the code.

In the calendar app, one approach to deal with the different time zones would be to code the information into the program itself. We could simply have a function that, given a country or city name, returns the offset, implemented with a lot of `switch` or `if then else` statements. A change in the timezone of a country would require rewriting the function and recompiling the whole application. Instead, we could externalize the knowledge about countries and cites and their

time zones. The application would then load the knowledge base whenever it is needed, and behave accordingly.

There are many different ways to represent the knowledge in a way that the application can read it. For example, we could have a database with tables of cities and countries and associated time zones. We could have comma separated lists of cities for each time zone. Or we could gather a consortium and define a standard for expressing time zone information, so that different developers can independently access this knowledge in their apps.[1]

RDF provides a generic, standardized model for representing knowledge. Generic means, that it can be used in any domain, be it time zones, geographic data, genes, books, or anything else. Standardized means that RDF has been specified in an explicit, reimplementable way, so that everyone can create software that can correctly read and write RDF documents.

The basic notions of RDF are triples and graphs. Triples are used to express the given knowledge, piece by piece. Each triple is built of three elements: a subject, a predicate, and an object. In our example, a piece of knowledge could be expressed with the following triple:

```
Turkey timezone EET .
```

The subject in this triple is `Turkey`, the predicate is `timezone`, and the object `EET`. The triple can intuitively be understood as having the same meaning as the English sentence *"Turkey is in the timezone EET."*.

A graph consists of a set of triples. It is called a graph, because the subjects and objects can be understood as nodes, connected with directed edges labeled with the given property. The following example graph is visualized in Figure 1.

```
Turkey timezone EET .
Greece timezone EET .
Georgia timezone GET .
EET borders GET .
EET offset "2"^^int .
GET offset "4"^^int .
```

The last two triples in the example graph demonstrate the use of literal values in the object position: instead of a named node, that refers to an entity, we have a typed literal value. The type in this case is `int`, which means that the values should be interpreted as integers. The types available in RDF cover numbers, strings, booleans, dates, time durations, URIs, XML, and others.

**Further readings**: If you want to get deeper into the formalisms behind RDF have a look at the the RDF specification [25], the RDF primer [27], the specification for XSD datatype [30]. Relevant concepts we did not introduce are blank nodes, reification, named graphs and extensible datatypes cf. [15,27].

[1] For time zones, this is roughly what happened, in form of the *tz database*.

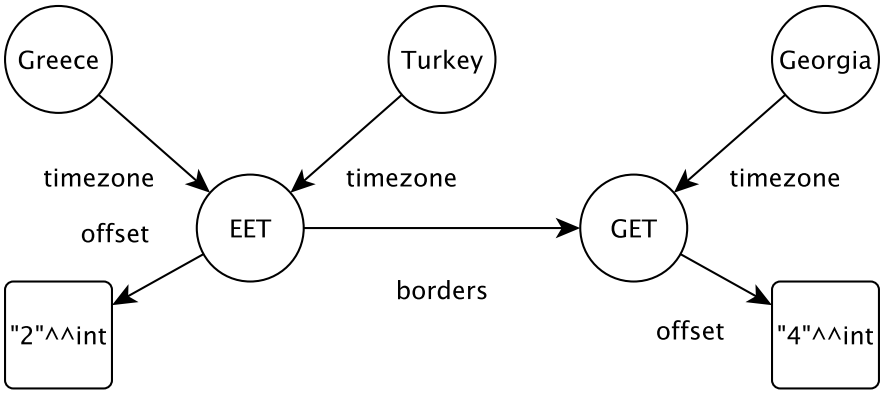



**Figure 1.** Example of Linked Data used for the calender application

## 4. URIs: The words of the language

The big advantage of a graph-based model is that they can be easily be merged, by simply regarding a merger as a union of the triples in both graphs. Many other models for knowledge representations, like tables or XML files, do not allow for such a generic merging. But in order to provide a knowledge representation language that allows these kind of mergers, naming conflicts must be avoided. In our example, `Georgia` refers to the caucasian country. But now consider a second knowledge base about US states:

```
Georgia timezone EST .
SouthCarolina timezone EST .
EST offset "-5"^^int .
```

Since both Georgias – the caucasian country and the US state – are named the same, we would not be able to differentiate them. Georgia now seems to be both in the timezone `GET` and `EST`!

To avoid this situation, all entities in Linked Data have to be referred to by unique names. To achieve this, the names are given as URIs, most often HTTP URIs (just like websites). The advantage of URIs is that anyone can register a prefix, and then create new names with this prefix. The owner of the prefix is responsible for what the given name means. So, the country Georgia could be named `http://example.org/countries#Georgia` and the us state could be `http://example.org/usstates#Georgia`. As long as everyone defines new names in their own namespace only, naming conflicts can be avoided without constant coordination between all parties.

As URIs can be quite lengthy, often qualified names (or *QNames*) are used. They have the form `prefix:localName`. Prefixes are defined locally to expand to a certain namespace, e.g., in our example we could define that the prefix `us` means the namespace `http://example.org/usstates#`, and thus we could use the qualified name `us:Georgia` in order to refer to the complete URI `http://example.org/usstates#Georgia`.

**Further readings**: To find more about the specification of URIs, see [6] and for to get to know how to register a new URI scheme, have a look at [22]. We did not speak about IRIs [18] and other protocols besides HTTP, like FTP [31] or URN [34].

## 5. HTTP: Distributed knowledge

The second rule for publishing linked data is to use HTTP URIs. The advantage is, that HTTP is a widely implemented protocol, that can be used over the Internet for accessing resources with a given HTTP URI. For example, the above knowledge base could have itself the name `http://example.org/countries`. Now a simple HTTP `GET` command like

```
GET http://example.org/countries
```

will return the knowledge base (it can be tried by entering the URI in a browser, but it is suggested for later as the result might be confusing at this point).

All entities are named with URIs, and the third rule of Linked Data asks to return information about the entity identified with a given URI when the URI is being dereferenced via HTTP. This way, the Web can be used as vast knowledge space, where everyone can publish what they know about a given entity.

We can also use the URIs of others – we do not have to publish URIs for all entities that we want to use ourselves. For example, DBpedia is a widely used resource that publishes Linked Data based on Wikipedia's infoboxes [10]. DBpedia offers URIs for all entities that have a Wikipedia page. For example, Greece' Wikipedia page is `http://en.wikipedia.org/wiki/Greece` and, based on that, DBpedia defines the URI for Greece to be `http://dbpedia.org/resource/Greece`. If this is entered into a browser, the browser redirects the user to a Web page about Greece in DBpedia. If a Semantic Web application would have asked, DBpedia (prefix dbpedia) would have returned the RDF data instead.

So we could replace

```
countries:Greece tz:timezone tz:EET .
```

with using the DBpedia URI for Greece and get

```
dbpedia:Greece tz:timezone tz:EET .
```

This would have the advantage that now we can learn much more about Greece: the name of the country in different languages and alphabets, its population, the name of the head of state, etc. Suddenly, our application can use knowledge from all over the Web.

Instead of simply replacing the URI, in our case we actually state that the URIs refer to the same entity, like this:

```
countries:Greece owl:sameAs dbpedia:Greece .
```

We see, that also properties can be reused from all over the Web. In this case we use the term `sameAs` from the OWL vocabulary, which we will look at in the next section.

One advantage of using the Web as a knowledge base is that much knowledge is already published: whereas our knowledge bases had information on some countries, Websites like Geonames or DBpedia offer lists of cities, and in which countries or US states they are located. So regarding the cities we mentioned in the beginning of the article, the Web already offers the following pieces of knowledge:

```
dbpedia:Istanbul dbo:country dbpedia:Turkey .
dbpedia:Athens dbo:country dbpedia:Greece .
dbpedia:Atlanta dbo:isPartOf dbpedia:Georgia_(U.S._state) .
```

This allows us to just reuse the knowledge about cities from the Web of Data, for very little cost.

**Further readings**: For a deeper introduction of the HTTP protocol have a look at the HTTP specification [20].

## 6. RDFS, OWL and others: Adding expressivity

RDF allows us to express triples directly. A very powerful method is to allow for implicit triples, by using more expressive semantics than simple triples. We have seen one example already: `owl:sameAs` states that two URIs refer to the same entity. That means that anything we say about `dbpedia:Greece` is also true about `countries:Greece`. So now that we learnt that `dbpedia:Athens` is in `dbpedia:Greece`, we know that it is also in our own `countries:Greece`.

A number of languages build on top of RDF and extend it with more expressive semantics. We will look at three of them, that are standardized by the W3C: RDFS, OWL, and RIF.

RDFS is the simplest one of them. It allows us to describe class and property hierarchies: for example, we have found on the Web cities connected to countries resp. US states. The connection to the country was done using the `dbo:country`, to the state with `dbo:isPartOf`. Now we can also define that everything that is connected via the former should also be connected through the latter. In RDFS we do that with the following triple:

```
dbo:country rdfs:subPropertyOf dbo:isPartOf .
```

Now a reasoner who follows the RDFS semantics can infer that

```
dbpedia:Istanbul dbo:isPartOf dbpedia:Turkey .
```

is true, even though it was never stated explicitly.

OWL is much more expressive regarding the description of classes and properties: for example, we can state that every city has to be in exactly one time zone, or that nothing can be both a U.S. state and a country, which could have helped to discover the error with the two Georgias automatically.

Since RDF allows only triples, such more complex statements need to be broken down to triples. The statement *"Every city is in exactly one time zone."* translates in RDF to the following four triples:

```
x:statement1 rdf:type owl:Restriction .
x:statement1 owl:onProperty tz:timezone .
x:statement1 owl:qualifiedCardinality "1"^^xsd:nonNegativeInteger .
x:statement1 owl:onClass dbo:City .
```

Although this might seem a bit daunting, in reality these kind of triples are hidden either by more high-level syntaxes (see Section 8) or query tools (see Section 7).

RIF is a different beast. Whereas OWL is an expressive language to describe classes and properties, RIF is a way to express rules of them form *if... then...*. In our example, we might want to state that whenever a city is part of a country or state, and the country or state has a time zone, this is also the time zone of the city. Or, using the variables `?x`, `?y`, and `?z`:[2]

If

```
?x dbo:isPartOf ?y .
?y tz:timezone ?z .
```

then

```
?x tz:timezone ?z .
```

RDFS, OWL, and RIF can, at least in theory, be all used together. It depends on the used tools if a given semantics is understood: some reasoners support parts of OWL (so called fragments), some support only RDFS, other RIF, and a very few claim to support interesting combinations of all three languages, and sometimes beyond.

**Further readings**: The whole formalism can be found in the specifications for RDFS [14], OWL [4], and RIF [11],but especially at the OWL primer in this book [24]. We did not discuss questions of how to reason and about the complexity and decidability of reasoning. This is a big research topic, and in the last few years, it was advanced tremendously. We point to the DL Handbook as an entry point into this topic [3]. We also did not mention the different fragments of OWL, RIF, and the differences between OWL and OWL2, which can all be found in detail in the respective standards. Besides the languages presented here, other languages like SKOS [28] or WSML [26] exist, that can define other semantics.

## 7. SPARQL: Querying RDF

So far, we have described how to express knowledge: both simple facts (with RDF) and more expressive statements that enrich the knowledge base (in the previous section). SPARQL provides a query language for RDF knowledge bases.

[2]We will not show how the rule is represented in RDF, as this looks even worse than the OWL statement above.

For example, assume that we have all the triples we have mentioned so far in one graph that we can query via SPARQL. Let's also assume that the system providing the SPARQL endpoint understands the semantics of RDFS, OWL, and RIF. Now we can ask for the offset for Athens:

```
SELECT ?offset WHERE {
  dbpedia:Athens tz:timezone ?tz .
  ?tz tz:offset ?offset .
}
```

The system will return as a result the integer 2.

Athens is in the country Greece. We know that from the Web. Due to the OWL subproperty triple, we also know that to be in a country means to be part of it. Because of the RIF rule, we can infer that if something is part of something, it also has the same timezone. Based on this, the following two triples, the first one implicit, the second given explicitly, are in our knowledge base:

```
dbpedia:Athens tz:timezone tz:EET .
tz:EET tz:offset "2"^^xsd:int .
```

A SPARQL query describes a triple pattern (similar to the one in the *if*-part of RIF), where symbols with a leading question mark are variables. A SPARQL processor now tries to find values for the variables, so that the whole SPARQL pattern can be fulfilled by the knowledge base. The SPARQL processors then returns a list of all possible answers for the selected variables,[3] i.e. in this case for all values that `?offset` can have so that the SPARQL query pattern matches in the knowledge base. Given our query pattern, the two triples above are the only match in our knowledge base, and thus the result, `"2"^^xsd:int` will be returned as the only possible value for `?offset`.

SPARQL can be regarded as the main interface to access knowledge on the Web of Data. Currently, the usual workflow to work with Linked Data is to find and gather trustworthy data from the Web, include some knowledge created for the task or tying together the data from the Web, put it all in one knowledge base, and then use SPARQL to get answers to the queries of interest to the given task.

**Further readings**: To find out more about SPARQL look at the specification for SPARQL [32]. We did not discuss that SPARQL is not only a query language but also a protocol of how to acccess SPARQL endpoints. We also did not discuss other types of queries: `DESCRIBE`, `ASK`, and `CONSTRUCT`, nor the powerful features of SPARQL to count, do math, regular expressions, named graphs, etc.

## 8. RDFa and Co.: Serializations of RDF

What is a serialization? To send around RDF graphs through the Web, we need somehow to write them down in documents, i.e. to serialize them in a sequence of

[3] That is, all variables following the introductory `SELECT` keyword, in this case `?offset`.

tokens. Throughout this chapter we have used a slightly simplified, triple-based serialization, N3 [8]. N3 has the advantage, that the triple structure of the graph is very obvious. Although, it is widely used, it has the disadvantage of not being standard.

There are a confusing number of serializations of RDF around, mostly due to the fact that the orginally standardized serialization in RDF/XML is considered to be not very pleasing. Soon, further syntaxes were created, some of them also in XML (like TriX [16]), some of them not (like N3 and its constrained version N-Triples [21]. Expressions in other languages like OWL and RIF were often very cumbersome to be translated to RDF (as shown in Section 6), and thus introduced serializations of their own, like the OWL Functional Syntax [29], the OWL XML presentation syntax (a serialization of OWL directly in XML, instead of going through RDF [23]), the RIF syntaxes [11], etc. Lately, JSON became a more prominent serialization format on the Web, and standards to represent the RDF data model in JSON are being worked on [35].

Besides serializations for pure RDF, there has been a second strand of embedding RDF in other file formats. One of the main use cases for RDF is to provide flexible metadata about a file. Embedding that metadata in the file itself has the advantage that the metadata is easier retained if the file gets moved, shared, changed, etc. A growing number of file formats, like Adobe's (PDF, Photoshop, etc.) allow to embed RDF [17].

The most relevant file format for the Web is obviously HTML itself, the language to describe Web pages and applications. The RDFa standard offers how to markup and annotate elements of a Web page with RDF. This allows a tool understanding RDFa to directly extract structured data out of a Website: a page about an event can be pulled into a calendar, a restaurant can be automatically filtered with the allergies of the user, different shopping Websites can be thrown into one knowledge base and be compared directly, etc.

**Further readings**: If you want to find more serializations have a look at the specifications of RDFa [1], RDF in JSON [35], and RDF/XML [5]. Also relevant is the ongoing conversation between the communities supporting Microformats [9], Microdata [33], and RDFa.

## 9. XML: The confusing older brother

XML became the de-facto lingua france for data on the Web and beyond. So it was natural, that it was assumed that RDF would be build on top of XML. But in reality, the two are very different beasts: XML describes a tree-based model, with a single root element, that has child elements in a strict order, and, who in turn, might have further child elements, in strict orders too, all strictly hierarchical. RDF describes a graph-based model, where the order of the edges does not matter, and that is expressed as a simple set of triples. XML schema defines a strict grammar for the elements in an XML documents, determining if an XML file is valid or not. RDF schemas provide additional knowledge to infer implicit knowledge from a given RDF file, and can hardly be used to check the

validity of an RDF file. It is often easier to deal with valid XML files than with RDF files, because the developer has guarantees about the structure of the file. On the other hand, RDF files are much easier to extend: one can simply add further triples, and, as long as they don't contradict the existing triples, the knowledge simply grows. Two RDF files can always be simply merged automatically. For XML files in general, such an operation makes no sense.

With the benefit of hindsight, forcing RDF into an XML-based serialization was bound to lead to numerous problems without gaining the hoped-for advantages. Many of the existing XML tools and workflows were actually unable to deal with RDF/XML files, so that the existing huge pool of experience and software could not be used to kickstart the Web of Data.

Today, XML does not play a prominent role for the Web of Data anymore. Even if it gets further used as the main serialization format for RDF, its data model and the tools used with XML are loosing relevance. It is an open question if this might change again, or if the rich set of software and experiences surrounding the XML world can be unlocked in favor of the Web of Data – or the other way around.

**Further readings**: For more information about XML, see the XML specification [13] or the XML Schema primer [19].

## 10. Conclusions and outlook

RDF is increasingly becoming the standard way to share data on the Web. Using and publishing RDF is not an academic exercise anymore. The flexibility and extensibility of RDF, together with the possibility to merge arbitrary RDF graphs, gives it a unique advantage compared to other wide spread data models. Confusions surrounding its several serializations, especially the ill-received standard RDF/XML-serialization, a maybe too early focus on OWL, and the late availability of SPARQL, have probably hampered uptake. Meanwhile, simple standards like Microformats and JSON have received considerable uptake.

The advantages and the genericity of Linked Data standards are being increasingly recognized. Instead of introducing hundreds, if not thousands of APIs and heterogeneous formats, one common data model and query language can substantially decrease costs of data integration and data reuse.

End user interfaces to the Web of Data are still missing – but maybe they always will. Maybe the role of Linked Data is to be background technology: no one asks for generic interfaces for end-users to SQL databases. Maybe the Web of Data has a similar fate.

Still several practical issues remain unresolved:

- In general, SPARQL is too powerful and too expensive for the Web. It is far too easy to bring a SPARQL endpoint down with a few queries.
- The multitude of serialization formats in practical use combined with the lack of standard formats besides RDF/XML hampers teaching about RDF and its uptake.

- For a number of wide-spread use cases, no standards or even widely acknowledged best practices exist: how to express numbers with units, especially imperial units? How to express data that was valid at a given point in time? How to express time spans? How to deal with numerical precision? How to work with simple geographical and temporal reasoning, like inclusion?
- The current standards allow fine-grained provenance information only through reification, a method, that is often strongly discouraged for several reasons.
- The semantics break down under inconsistencies. There is currently no accepted way to deal with diversity in knowledge bases, even though this will play a crucial role on the Web. This ties in with questions of trust that have not yet been sufficiently tackled: given diverse data about an entity, maybe even contradictions, how to choose which sources to trust? How to make this trust transferable to the user?

The Web of Data, as part of the Web, is getting increasingly tangled with all aspects of our lives. The growing number of intelligent apps and devices in our environment will have an ever-growing need to communicate with each other. Imagining a future where our calendar app can support the flight finder app by restricting the departure and arrival times based on our agenda and the locations of our meetings and the airport, has become much easier today than it used to be only a few years ago. Such a future is much easier to achieve when the applications and devices can all communicate in the same common and standard data model, and using the same interfaces.

## Acknowledgements

We thank the students, who listened to previous versions of the contents of this chapters, during courses given at KIT and FU Berlin, and during several summer school courses (ESWC summer school, Berkeley Summer School for Semantic Technologies, Asian Summer School for the Semantic Web). Their questions and the discussions with them helped enormously in sharpening the didactic approach and our own ideas of the topic. Writing of this chapter was partially funded by the EU project RENDER (FP7 Grant agreement ICT-257790-STREP).[4]


This work has been supported by the ESF and the Free State of Saxony.

## References

[1] B. Adida, I. Herman, M. Sporny, and M. Birbeck. Rdfa 1.1 Primer. Technical report, World Wide Web Consortium, http://www.w3.org/TR/2012/NOTE-rdfa-primer-20120607/, June 2012.

[2] Grigoris Antoniou and Frank vanHarmelen. *A Semantic Web Primer*. MIT Press, Cambridge, MA, USA, 2004.

[4] `http://www.render-project.eu`

[3] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider. *The description logic handbook: theory, implementation, and applications*. Cambridge University Press, New York, NY, USA, 2003.
[4] Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. OWL Web Ontology Language Reference. Technical report, W3C, http://www.w3.org/TR/owl-ref/, February 2004.
[5] David Beckett. RDF/XML Syntax Specification (Revised). Technical report, W3C, http://www.w3.org/TR/REC-rdf-syntax/, February 2004.
[6] T. Berners-Lee, R. Fielding, and L. Masinter. Rfc 3986, uniform resource identifier (uri): Generic syntax, 2005.
[7] Tim Berners-Lee. Linked data. World wide web design issues, July 2006.
[8] Tim Berners-Lee and Dan Connolly. Notation3 (n3): A readable rdf syntax. Technical report, W3C, http://www.w3.org/TeamSubmission/n3/, January 2008.
[9] Frances Berriman, Dan Cederholm, Dan Tantek Çelik, Dan Rohit Khare, Dan Ryan King, Dan Kevin Marks, and Dan Ben Ward. Microformats, 2004.
[10] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. Dbpedia - a crystallization point for the web of data. *Web Semant.*, 7(3):154–165, 2009.
[11] Harold Boley, Gary Hallmark, Michael Kifer, Adrian Paschke, Axel Polleres, and Dave Reynolds. RIF Core Dialect (Second Edition). Technical report, W3C, http://www.w3.org/TR/rif-core/, February 2013.
[12] Ronald Brachman and Hector Levesque. *Knowledge Representation and Reasoning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.
[13] Tim Bray, Jean Paoli, C. Michael Sperberg-McQueen, Eve Maler, and François Yergeau. Extensible markup language (xml) 1.0 (fifth edition), November 2008.
[14] Dan Brickley and Ramanathan V. Guha. Rdf vocabulary description language 1.0: Rdf schema. Technical report, http://www.w3.org/TR/rdf-schema/, 2004.
[15] Jeremy J. Carroll, Christian Bizer, Pat Hayes, and Patrick Stickler. Named graphs, provenance and trust. In *Proceedings of the 14th international conference on World Wide Web*, WWW '05, pages 613–622, http://doi.acm.org/10.1145/1060745.1060835, 2005. ACM.
[16] Jeremy J. Carroll and Patrick Stickler. Trix: Rdf triples in xml. Technical Report HPL-2004-56, HP Labs, May 2004.
[17] Adobe Corp. Xmp specification. Technical report, http://partners.adobe.com/public/developer/en/xmp/sdk/XMPspecification.pdf, 2005.
[18] M. Duerst and M. Suignard. RFC 3987: Internationalized Resource Identifiers (IRIs), January 2005.
[19] David C. Fallside and Priscilla Walmsley. XML Schema part 0: Primer second edition. Technical report, http://www.w3.org/TR/xmlschema-0/, October 2004.
[20] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Rfc 2616, hypertext transfer protocol – http/1.1, 1999.
[21] J. Grant and D. Beckett. RDF test cases, February 2004.
[22] T. Hansen, T. Hardie, and L. Masinter. RFC 4395: Guidelines and Registration Procedures for New URI Schemes, 2006.
[23] Masahiro Hori, Jérôme Euzenat, and Peter F. Patel-Schneider. OWL Web Ontology Language - XML Presentation Syntax. Technical report, W3C, http://www.w3.org/TR/owl-xmlsyntax/, June 2003.
[24] Markus Krötzsch, František Simančík, and Ian Horrocks. A description logic primer. *CoRR*, abs/1201.4089, 2012.
[25] Ora Lassila, Ralph R. Swick, World Wide, and Web Consortium. Resource description framework (rdf) model and syntax specification. 1998.
[26] Holger Lausen, Jos de Bruijn, Axel Polleres, and Dieter Fensel. Wsml - a language framework for semantic web services. In *W3C Workshop on Rule Languages for Interoperability*, 2005.
[27] Frank Manola and Eric Miller. RDF primer. Technical report, http://www.w3.org/TR/rdf-primer/, 2004.
[28] Alistair Miles and José R. Pérez-Agüera. Skos: Simple knowledge organisation for the

web. *Cataloging & Classification Quarterly*, 43(3):69–83, 2007.

[29] Boris Motik, Peter F. Patel-Schneider, Bijan Parsia, Conrad Bock, Achille Fokoue, Peter Haase, Rinke Hoekstra, Ian Horrocks, Alan Ruttenberg, Uli Sattler, and Michael Smith. OWL 2 Web Ontology Language - Structural Specification and Functional-Style Syntax (Second Edition). Technical report, W3C, http://www.w3.org/TR/owl2-syntax/, December 2012.

[30] David Peterson, Shudi (Sandy) Gao, Ashok Malhotra, C. M. Sperberg-McQueen, Henry S. Thompson, Paul V. Biron, and Ashok Malhotra. W3C XML schema definition language (XSD) 1.1 part 2: Datatypes. Technical report, http://www.w3.org/TR/xmlschema11-2/, 2012.

[31] J. Postel and J. K. Reynolds. RFC 959: File transfer protocol, 1985.

[32] Eric Prud'hommeaux and Andy Seaborne. Sparql query language for rdf. Technical report, http://www.w3.org/TR/rdf-sparql-query/, January 2008.

[33] Jason Ronallo. HTML5 Microdata and Schema.org. *The Code4Lib Journal*, (16), 2012. schema.org.

[34] Karen Sollins and Larry Masinter. Functional requirements for uniform resource names. Internet RFC 1737, December 1994.

[35] Manu Sporny, Gregg Kellogg, and Markus Lanthaler. Json-ld syntax 1.0. Technical report, http://json-ld.org/spec/latest/json-ld-syntax/, December 2012.

[36] W3C. Semantic Web, 2013.

# Introducing Machine Learning

Agnieszka ŁAWRYNOWICZ [a] and Volker TRESP [b]
[a] *Institute of Computing Science, Poznan University of Technology, Poznań, Poland;*
*E-mail: alawrynowicz@cs.put.poznan.pl*
[b] *Siemens AG, Corporate Technology, München, Germany;*
*E-mail: volker.tresp@siemens.com*

**Abstract.** In this chapter we provide an overview on some of the main issues in machine learning. We discuss machine learning both from a formal and a statistical perspective. We describe some aspects of machine learning such as concept learning, support vector machines, and graphical models in more detail. We also present example machine learning applications to the Semantic Web.

**Keywords.** Machine Learning, Data Mining, Inductive Inference, Models

## Introduction

The goal of *Machine Learning (ML)* is to construct computer programs that can learn from data. The *inductive inference* of machine learning, i.e. the generalizations from a set of observed instances, can be contrasted to early Artificial Intelligence (AI) approaches that dealt mostly with deductive inference (cf. Krötzsch et al. [24] in this volume), i.e., the derivation of theorems from axioms. Although ML is considered a subfield of AI it also intersects with many other scientific disciplines such as statistics, cognitive science, and information theory[1]. An area, closely related to ML is *data mining* [11,12] which deals with the discovery of new and interesting patterns from large data sets. Although ML and data mining are often used interchangeably, one might state that ML is more focused on adaptive behavior and operational use, whereas data mining focusses on handling large amounts of data and the discovery of previously unknown patterns (implicit knowledge, regularities) in the data. Most of this chapter discusses ML in the context of a formal AI system, although when suitable, as in the discussion of graphical models, we assume a more statistical perspective.

ML approaches can be distinguished in terms of representation and adaptation. A machine learning system needs to store the learned information in some knowledge representation structure which is called (an *inductive*) *hypothesis* and is typically of the form of a *model*. Following the Ockham's razor principle, the hypothesis should generalize the training data giving preference for the simplest hypothesis; to obtain valid generalization, the hypothesis should be simpler than the data itself. A *learning algorithm* specifies how to update the learned hypothesis with new *experience* (i.e. *training data*) such that the *performance measure* with regard to the *task* is being optimized (see Figure 1).

[1] Certainly there are many different perspectives on machine learning: some researchers see the strongest link with statistics and even claim that both fields are identical.

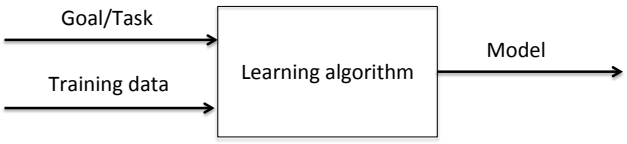



**Figure 1.** A generic machine learning method.

Over the years, machine learning methods have been applied to solve many real-world problems such as spoken language recognition, fraud detection, customer relationship management, gene function prediction etc. To provide a concrete example where machine learning has been effective on a Web service, consider the task of categorizing email messages as spam or non-spam, where the performance of the machine learning method is assessed by the percentage of email messages correctly classified. The training experience in this problem may come in the form of the database of emails that has been labeled as spam or no-spam by humans.

The rest of this chapter is organized as follows. In the next Section we discuss some basic aspects of machine learning and in Section 2 we discuss concept learning, the support vector machine and graphical models. In Section 3 we discuss applications of machine learning to the Semantic Web and Section 4 contains our conclusions.

## 1. Machine Learning Basics

In this section, we introduce the basic components of a machine learning problem. We discuss that learning algorithms are typically implemented as some kind of search and local optimization. The following Section 2 then describes three important machine learning approaches in more detail.

### *1.1. Tasks*

#### *1.1.1. Classification and Regression*

The tasks of *classification* and *regression* deal with the prediction of the value of one field (the target) based on the values of the other fields (*attributes* or *features*). If the target is discrete (e.g. nominal or ordinal) then the given task is called classification. If the target is continuous, the task is called regression. Classification or regression normally are supervised procedures: based on a previously correctly labeled set of training instances, the model learns to correctly label new unseen instances.

An example classification problem may consist in predicting whether to grant or not to grant a credit to a customer. The values of the class $c$ in this problem could be formed by a set {yes, no} representing a positive and a negative decision, respectively. The input to the classification method (that is, to a *classifier*) would consist of information about a customer. In particular, if the hypothesis space consists of rules, the output may be formed by a set of learned rules such as the one presented in Figure 2a.

### 1.1.2. Learning Associations

An association describes a relation between objects, or measured quantities, that is the result of some interaction or of a dependency between the objects. Typically, the learned associations are in the form of *association rules* or sets of *frequent items*. The motivation for this type of task has been provided by market basket analysis where the methods for finding associations between products bought by customers are studied. For example, consider that customers who buy X (e.g. beer) typically also buy Y (e.g. chips); then, if we encounter a customer who buys X but does not buy Y, we may target this customer via cross-selling as a potential customer for Y. An itemset is called frequent if it appears in at least a given percentage (called *support*) of all transactions. Frequent itemsets are often the prerequisite for the learning of association rules.

### 1.1.3. Clustering

*Clustering* is an unsupervised task, whose aim is to group a set of objects into classes of similar objects. A cluster is a collection of objects that are similar to each other within the same cluster, and dissimilar to the objects in other clusters. Therefore, an important notion in clustering (also known as *cluster analysis* in statistics) is the notion of *similarity* (or *distance*). In *conceptual clustering*, a symbolic representation of each cluster is extracted and we may consider each cluster to be a *concept*, closely related to a class in classification.

### 1.1.4. Other Machine Learning Tasks

Some examples of other machine learning tasks are: reinforcement learning, learning to rank and structured prediction.

The reinforcement learning task consists of learning sequential control strategies. It deals with situations, where the output of the system is a sequence of actions that are performed to achieve some goal. An example may be game playing, where the complete sequence of moves is important, rather than a single move.

Learning to rank is a type of a (semi-)supervised learning problem where the goal is an automatic construction of a ranking model from training data, e.g., to learn to rank the importance of returned Web pages in a search application.

Structured prediction deals with prediction problems in which the output is a complex structure. Such problems arise in disciplines such as computational linguistics, e.g. in natural language parsing, speech, vision, and biology.

## 1.2. Training Data

One distinguishes three important classes of feedback: feedback in the form of labeled examples, feedback in the form of unlabeled examples, or feedback in the form of reward and punishment, as in reinforcement learning.

*Supervised learning* consists of learning a function from training examples, based on their attributes (inputs) and labels (outputs). Each training example is a pair $(x, f(x))$, where $x$ is the input, and $f(x)$ is the output of the underlying unknown function. The aim of supervised learning is: given a set of examples of $f$, return a function $h$ that best approximates $f$. For example, given symptoms and corresponding diagnoses for patients,

the goal is to learn a prediction model to make a diagnose based on symptoms for a new patient.

*Unsupervised learning* is concerned with learning patterns in the input, without any output values available for training. Continuing our example, only symptoms of patients may be available and the goal may be to discover groups of similar patients.

In *semi-supervised learning*, both labeled and unlabeled data is used for training, with typically only a small amount of labeled data, but a large amount of unlabeled data. In the clinical example, diagnoses might be available for only a few patients, and the goal would be to use this information for making most probable diagnoses for all patients.

In *reinforcement learning*, input/output pairs are not available to the learning system. Instead, the learning system receives some sort of a reward after each action, and the goal is to maximize the cumulative reward for the whole process. For example, in case of treatment planning, the learning system may receive reinforcement from the patient (e.g., feels better, feels worse, cured) as an effect of actions taken during a treatment.

Typically, a training data set comes in the simple form of *attribute-value* data table. However, more complex input data has also been studied, for example sequences, time series or graphs. From the point of view of this book, an interesting setting is presented by *Inductive Logic Programming (ILP)* [33,36], originally defined as a subfield of machine learning that assumes Logic Programming as a representation formalism of hypotheses and background knowledge. Since then ILP has been further refined to a broader definition that considers not only Logic Programs as a representation, but also other subsets of the first-order logic. In particular, its methodology is well-suited for the tasks where Description Logics are used as representation formalism. The distinguishing feature of the ILP methods is their ability to take into account *background knowledge*, that is the knowledge generally valid in some domain, represented, for example, in the form of ontologies.

### *1.3. Models*

Machine learning hypotheses may come in a variety of knowledge representation forms, such as equations, decision trees, rules, distances and partitions, probabilistic and graphical models. Classically, a division is made between *symbolic* and *sub-symbolic* forms of knowledge representation. The first category consists of representation systems in which the atomic building blocks are formal symbolic representations, often easily readable by a human. Such representation systems have compositional syntax and semantics, and their components may be assigned an interpretation. The system may, for example, be composed of a set of rules such as the one presented in Figure 2a. A good example of a symbolic system is an interpreted logical theory.

In turn, the components of a sub-symbolic representation system do not have a clear interpretation, and are not formal representations by themselves. Knowledge in this approach is represented as numerical patterns determining the computation of an output when being presented a given input. Good examples of sub-symbolic systems are neural networks, where the patterns are represented in the form of interconnected groups of simple artificial neurons.

Figure 2 provides an illustration of the distinction between symbolic and sub-symbolic representations.

Typical machine learning algorithms induce models, that is hypotheses that characterize globally an entire data set.

IF income $>$ 1500 THEN c = yes

(a)

(b)

**Figure 2.** An illustration of a) symbolic and b) sub-symbolic representation.

### *1.4. Generative and Discriminative Models*

So far the discussion was oriented towards a formal description of a learning problem. Here we describe a statistical view based on the concepts of generative and discriminative probabilistic models.

Generative models simulate a data generating process. In an unsupervised learning problem, this would involve a model for $P(X)$, i.e., a probabilistic model for generating the data.[2] In supervised learning, such as classification, one might assume that a class $Y \in \{0, 1\}$ is generated with some probability $P(Y)$ and the class-specific data is generated via the conditional probability $P(X|Y)$. Models are learned for both $P(Y)$ and $P(X|Y)$ and Bayes rule is employed to derive $P(Y|X)$, i.e., the class label probability for a new input $X$. In contrast, discriminative models model the conditional probability distribution $P(Y|X)$ directly, they learn a direct mapping from inputs $X$ to class label probabilities. To illustrate the difference between generative and discriminative models let us discuss an example task consisting in determining the language of a given speaker. In a generative modeling approach this task would be solved by learning language models for each language under consideration, i.e. by learning $P(X|Y)$ for each language $Y$ and by then applying Bayes rule to infer the language for a new text $x$. A discriminate model would not bother modeling the distribution of texts but would focus on the task of language classification directly and could focus on only the differences between languages. Popular examples of generative models are Naive Bayes, Bayesian Networks and Hidden Markov Models. Popular examples of discriminative probabilistic models are logistic regression and support vector machines.

### *1.5. Training Generative Models*

Since our description of graphical models is based on a generative modeling approach for an unsupervised model $P(X)$, we briefly discuss the training of such models. In a simple maximum likelihood model, one assumes a model $P(X|w)$, i.e. a probabilistic model for generating a data point given parameter vector $w$. The maximum likelihood parameters estimate is then defined by the parameters that maximize the likelihood, where the likelihood is $L(w)$ defined as the product of the probabilities of generating the $N$ independent training data points given the parameters and assumes the form

$$L(w) = \prod_{i=1}^{N} P(X_i = x_i|w)$$

[2] In the discussion on generative models, $X$ and $Y$ stand for random variables.

In a Bayesian approach the model is completed with an *a priori* distribution over models $P(M)$ and a prior distribution over parameters given model, i.e., $P(w|M)$. Based on observed data $D$, one can now calculate the most likely model as the one that maximizes

$$P(M|D) = \frac{P(D|M)P(M)}{P(D)}$$

or the parameter distributions given model and data as

$$P(w|D, M) = \frac{L(w)P(w|M)}{P(D|M)}$$

A maximum a posteriori (MAP) estimate is achieved by taking the most likely model and selecting the parameters that maximize $P(w|D, M)$. A more truthfully Bayesian approach would consider the uncertainties in the estimates by integrating over unobserved quantities in the prediction.

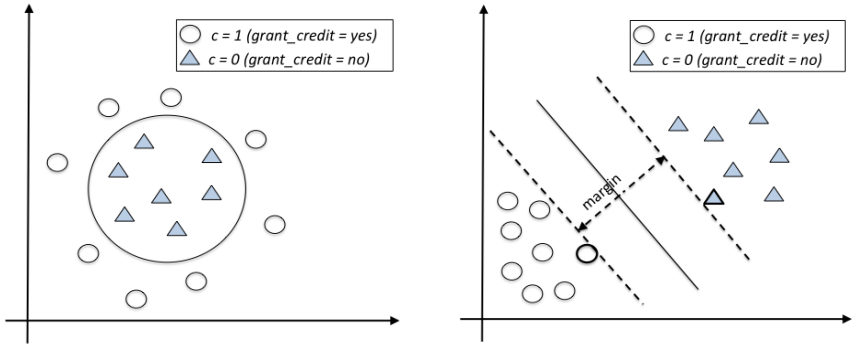
## 2. An Overview of Selected ML Approaches

In this section, we provide an introduction to selected ML approaches, in particular to those that will be further referred to throughout the book.

### *2.1. Concept Learning*

*Concept learning* consists of inducing the general definition of some concept (a category) given training examples labeled as members (positive examples) and nonmembers (negative examples) of the concept. Each training example consists of an instance $x \in X$ and its target concept value $f(x)$. Thus, a training example can be described by the ordered pair $(x, f(x))$. A learned concept is often represented as a boolean valued function. An example is called a positive one if $f(x) = 1$, and a negative one if $f(x) = 0$. Concept learning can be posed as a problem of searching the space of hypotheses to find a hypothesis best fitting the training data. The concept learning task may be thus formulated as follows [32]. Given instances $x \in X$, a target concept $f$ to be learned ($X \rightarrow \{0, 1\}$), hypotheses $H$, described by a set of constraints they impose on instances, training examples $D$ (positive, and negative examples of the target function), determine a hypothesis $h \in H$, such that $h = f(x)$ for all $x \in X$. Such a hypothesis, if learned on sufficiently large set of training examples, should also approximate the target concept well for new examples.

By choosing the hypothesis representation, one determines the space of all hypotheses that can ever be learned by the given method. The hypothesis space may be ordered by a generality relation $\succeq_g$. Let $h_i$ and $h_j$ be two hypotheses. Then $h_i$ is more general or equal to $h_j$ (written $h_i \succeq_g h_j$) if and only if any instance satisfying $h_j$, also satisfies $h_i$, where an instance $x \in X$ is said to *satisfy* $h \in H$ if and only if $h(x) = 1$. The relation $\succeq_g$ is a *partial order* (i.e., it is reflexive, antisymmetric and transitive) over the hypothesis space. Therefore, there may be also cases where two hypotheses are incomparable with $\succeq_g$, what happens if the sets of instances satisfied by the hypotheses are disjoint or intersect (are not subsumed by one another).




(a) (b)

**Figure 3.** An illustration of SVMs. a) a non-linear, circular concept b) linearly separable instances, margin, and support vectors (with thicker border).

A hypothesis $h$ is called *consistent* with a set of training examples $D$ if it correctly classifies all these examples that is if and only if $h(x) = f(x)$ for each training example $(x, f(x))$ in $D$. The set of all hypotheses consistent with the training examples is called the *version space* $VS$ with respect to $H$ and the training examples $D$. The version space $VS$ may be represented by the sets of its maximally specific and maximally general hypotheses that delimit the entire set of hypotheses consistent with the data forming the boundaries of $VS$.

Concept learning algorithms utilize a structure imposed over the hypothesis space by the relation $\succeq_g$ to efficiently search for relevant hypotheses. For example, Find-S algorithm [32] performs the search from most specific to most general hypotheses in order to find the most specific hypothesis consistent with the training examples, while Candidate Elimination algorithm [32] exploits this general-to-specific ordering to compute the version space by an incremental computation of the sets of maximally specific and maximally general hypotheses. The search for hypotheses is steered also by the *inductive bias* of a concept learning algorithm, that is the set of assumptions representing the nature of the target function used by the algorithm to predict outputs given previously unseen inputs. The learning algorithm implicitly makes assumptions on the correct output for unseen examples to select one consistent hypothesis over another.

Some of concept learning algorithms proposed in the context of ILP that are relevant for this book are FOIL [39], and PROGOL [34]. They both induce first-order rules similar to Horn clauses. Concept learning is a very useful technique for ontology learning, and will be discussed in this context in more detail in (cf. Lehmann et al. [28] in this volume).

### *2.2. Support Vector Machines*

*Support Vector Machines* (or *SVMs*) [2,3] may be used for binary classification or for regression. In binary classification, they construct a linear hyperplane (a *decision boundary*) to separate instances of one class from the other class. The separation between the classes is optimized by obtaining the separating hyperplane which is defined as the plane having the largest distance or margin to the nearest training data points of any class. Figure 3 illustrates the general concept of SVMs. Mathematically, the separating hyperplane may be defined by the equation:

$$\mathbf{w} \cdot \mathbf{x} + b = 0$$

where $\mathbf{w}$ is a weight vector, $b$ is a scalar, and $\mathbf{w} \cdot \mathbf{x}$ is the inner product between $\mathbf{w}$ and $\mathbf{x}$.

The distance of the closest points to the decision boundaries *defines* the margin, which is maximized during training. The optimization problem solved by SVMs may be formulated as follows:

$min_{\mathbf{w},\mathbf{b}} \mathbf{w} \cdot \mathbf{w}$
where $\forall \mathbf{x} \in D : f(\mathbf{x})(\mathbf{w} \cdot \mathbf{x} + b) \geq 1$,
$D$ is a set of examples, and $f(\mathbf{x}) = 1$ for $c_i = 1$, and $f(\mathbf{x}) = -1$ for $c_i = 0$

It can be shown that the margin is $\frac{1}{||\mathbf{w}||}$, where $||\mathbf{w}||$ is the Euclidean norm of $\mathbf{w}$. Thus minimizing $||\mathbf{w}||$, maximizes the margin. The optimization problem is usually not solved as posed in the Equation 1, but rewritten into a dual problem known as a constrained (convex) quadratic optimization problem, that can be solved by public domain quadratic programming solvers (for the details see for example [9]).

A new input $\mathbf{x}$ can be labeled as positive (1) or negative (0) based on whether it falls on or "above" the hyperplane:

$$\mathbf{w} \cdot \mathbf{x} + b \geq 0, \text{for } c_i = 1, \text{and}$$
$$\mathbf{w} \cdot \mathbf{x} + b \leq 0, \text{for } c_i = 0$$

SVMs can also form nonlinear classification boundaries. For this purpose, the original input data is transformed into a higher dimensional space by a nonlinear mapping $\phi$, and a linear separating hyperplane in this new space is formed. Fortunately, the nonlinear mapping function $\phi$ does not need to be specified explicitly. While solving the quadratic optimization problem of the linear SVM, the training tuples occur only in the form of dot products, $\phi(\mathbf{x_i}) \cdot \phi(\mathbf{x_j})$. Then, instead of computing the dot product on the transformed tuples, it is mathematically equivalent to apply a *kernel function* $K(\mathbf{x_i}, \mathbf{x_j})$ to the original input data, where the kernel is defined as:

$$K(\mathbf{x_i}, \mathbf{x_j}) = \phi(\mathbf{x_i}) \cdot \phi(\mathbf{x_j})$$

SVMs have attracted a lot of attention in recent years since they are less likely to suffer from overfitting than other methods. A drawback is that training scales unfavorable with the size of the training data set. This section mainly followed [11,40]. A good introduction to SVM methods may be also found in [3].

### *2.3. Learning in Graphical Models*

Given a set of $M$ features or random variables, $X_1, \ldots, X_M$, graphical models are a means to efficiently describe their joint probability distribution $P(X_1, \ldots, X_M)$ by exploiting independencies in $P(\cdot)$. We can consider graphical models as generative models, modeling the statistical dependencies among a potentially large number of variables. In terms of a knowledge representation, $X_i$ might stand for the truth value of a statement, and $X_i = 1$ if the statement is true and $X_i = 0$ otherwise.

In graphical models, independencies in the model can be displayed in form of a graph. We will consider the two most important subclasses of graphical models here, i.e., Bayesian networks (a.k.a directed graphical models) and Markov networks (a.k.a undirected graphical models).

### 2.3.1. Bayesian Networks

*The Basics* Any probability distribution of $M$ random variables can be decomposed in product form as

$$P(X_1, \ldots, X_M) = \prod_{i=1}^{M} P(X_i | \{X_j\}_{j<i})$$

where the order of the variables is arbitrary.

Bayesian networks exploit the fact that the set of all predecessors can be reduced to a set of parent nodes

$$\prod_{i=1}^{M} P(X_i | \{X_j\}_{j<i}) = \prod_{i=1}^{M} P(X_i | \text{par}(X_i))$$

where $\text{par}(X_i) \subseteq \{X_j\}_{j<i}$, thus exploiting independencies in a domain. In a graphical representation, nodes represent the random variables and one draws directed links from parent nodes to child node. Although the decomposition is possible in any order, most independencies are typically exploited when a causal ordering is observed, i.e, when the parents of a node also correspond to its direct causal probabilistic factors. It is of great importance to exploit domain independencies, since otherwise inference and other operations would require resources exponential in the number of variables. Consider, as an example, a diagnostic setting, where the random variables stand for diagnosis, symptoms and influencing factors. Influencing factors (e.g., smoking) are probabilistic causal factors for diseases, and diseases are probabilistic causal factors for symptoms. After the conditional probabilities $P(X_i|\text{par}(X_i))$ are defined by a medical expert, probabilistic inference can be used to calculate the probabilities of a disease given symptoms and given influencing factors.

One typically has the case that a certain probabilistic dependency appears several times in the training data. For example, this could describe the dependency of fever given flue, and this dependency is assumed identical for all patients. We use $P^k(X_i|\text{par}(X_i))$ where $i \in I(k)$ to indicate that the dependency between $X_i$ and its parents is of type $k$ and where $P^k(\cdot)$ now stands for a parameterized function.

Learning can assume varying complexity. In the simplest case, the causal structure and all variables are known in training and the log-likelihood can be written as

$$l(w) = \log L(w) = \sum_{k} \sum_{i \in I(k)} \log P^k(X_i | \text{par}(X_i), w)$$

where $w$ are model parameters. As we see, the log-likelihood nicely decomposes into sums which greatly simplifies the learning task. Typical learning approaches are maximum likelihood learning, penalized maximum likelihood learning, and fully Bayesian learning.

In the next level of complexity, some variables are unknown in the training data and some form of EM (expectation maximization) learning is typically applied.

Finally, we might also assume that the causal structure is unknown. In the most common approach one defines a cost function (e.g., Bayesian Information Criterion (BIC)

or marginal likelihood) and one does heuristic search by removing and adding directed links to find a structure that is at least locally optimal. Alternatively one performs statistical testing to discover independencies in the domain and one defines Bayesian network structures consistent with those (constraint-based approach). Care must be taken, that no directed loops are introduced in the Bayesian network in modeling or structural learning. A large number of models used in machine learning can be considered special cases of Bayesian networks, e.g., Hidden Markov models, Kalman filters, which is the reason for the great importance of Bayesian networks. Readers, who are interested to learn more should consult the excellent tutorial [13].

*Modeling Relationships* Traditionally, Bayesian networks have mostly been applied to attribute-based representations. Recently, there has been increasing interest to applying Bayesian networks to domains with object-to-object relationships for which the term statistical relational learning (SRL) is used. Relationships add to complexity. In an attribute-based setting, one often assumes that objects are sampled independently, which greatly simplifies inference. For example the wealth of a person can be predicted from income and value of the person's home but, given this independence sampling assumption, is independent from the wealth of other people (given parameters). As a consequence, inference can be performed separately for each person. In SRL, one could also consider the wealth of this person's friends. As a consequence, random variables become globally dependent and inference often has to be performed globally as well, in this example potentially considering the wealth of all persons in the domain. A second issue is that directed loops become more problematic: I cannot easily model that my wealth depends on my friends's wealth and vice versa without introducing directed loops, which are forbidden in Bayesian networks. Finally, aggregation plays a more important role. For example, I might want to model that a given teacher is a good teacher, if the teacher's students get good grades in the classes the teacher teaches. This last quantity is probably not represented in the raw data as a random variable but needs to be calculated in a preprocessing step. As one might suspect, aggregation tends to make structural learning more complex.

Probabilistic Relational Models (PRMs) were one of the first published approaches for SRL with Bayesian networks and found great interest in the statistical machine learning community [23,10]. PRMs combine a frame-based logical representation with probabilistic semantics based on directed graphical models. Parameter learning in PRMs is likelihood based or based on empirical Bayesian learning. Structural learning typically uses a greedy search strategy, where one needs to guarantee that the ground Bayesian network does not contain directed loops.

Another important approach is presented by the infinite hidden relational model (IHRM) [50].[3] Here each object is represented by a discrete latent variable. The parents of a node representing a statement are the latent variables of all the objects involved. Thus, if $X_{u,m}$ stands for the statement that user $u$ likes movie $m$, then we would obtain the term $P(X_{u,m}|X_u^l, X_u^l)$ where $X_u^l$ is the latent variable for user $u$ and where $X_m^l$ is the latent variable for user $m$. The resulting network has by construction no loops. Also the need for aggregation is alleviated since information can propagate in the network of latent variables. Finally, no structural learning is required as the structure is given by the typed relations in the domain. The IHRM is applied in the context of a Dirichlet process mixture model where the number of states is automatically tuned in the sampling pro-

[3] Kemp et al. [20] presented an almost identical model independently.

cess. In [42] it was shown how ontological class information can be integrated into the IHRM and in [44] it is shown how OWL constraints can be integrated.

Bayesian networks are also being used in ILP where they form the basis for combinations of rule representations with SRL. A *Bayesian logic program* is defined as a set of Bayesian clauses [21]. A Bayesian clause specifies the conditional probability distribution of a random variable given its parents on a template level, i.e. in a node-class. A special feature is that, for a given random variable, *several* such conditional probability distributions might be given. For each clause there is one conditional probability distribution and for each Bayesian predicate (i.e., node-class) there is one combination rule. *Relational Bayesian networks* [17] are related to Bayesian logic programs and use probability formulae for specifying conditional probabilities.

### *2.3.2. Markov Networks*

*The Basics* The most important parameterization of the probability distribution of a Markov network is

$$P(X_1, \ldots, X_M) = \frac{1}{Z} \exp \sum_k w_k f_k(\{X\}_k)$$

where the feature functions $f_k$ can be any real-valued function, where $\{X\}_k \subseteq \{X_1, \ldots, X_M\}$ and where $w_k \in \mathbb{R}$. $Z$ normalizes the distribution. In a graphical representation, all variables in $\{X\}_k$ would be mutually connected by undirected links and thus would form cliques in the resulting graph.

We consider first that the feature functions $f_k(.)$ are given. Learning then consists of estimating the $w_k$. The log-likelihood is

$$l(w) = -\log Z + \sum_k w_k f_k(\{X\}_k)$$

Even a simple maximum likelihood estimate leads to a non-trivial optimization problem, since $Z$ is a function of all the parameters in the model.

The more complex question is, how application specific feature functions $f_k(.)$ can be defined. In Markov logic networks, as described next, the feature functions are derived from logical constraints.

*Markov Logic Networks (MLN)* Let us consider a simple example with two friends $A$ and $B$. Let $X_{A,r} = 1$ and $X_{B,r} = 1$ stand for the facts that person $A$ is rich, respectively person $B$. Let $X_{A,p} = 1$ and $X_{B,p} = 1$ stand for the facts that person $A$ is poor, respectively person $B$. Then we define the feature function $f_{r,r}(X_{A,r}, X_{A,r})$ which is only equal to one if $X_{A,r} = 1$ and $X_{B,r} = 1$ and is equal to zero else. Similarly, we define $f_{p,p}, f_{p,r}, f_{r,p}$. After training, we might obtain the weights $w_{r,r} = 10$, $w_{p,p} = 10$, $w_{r,p} = -5$, $w_{p,r} = -5$. Thus a situation where both friends are both rich or both are poor is much more likely (for example, $P(X_{A,r} = 1, X_{B,r} = 1, X_{A,p} = 0, X_{B,p} = 0) = Z^{-1} \exp 10$) than the situation where only one of the is rich and the other one is poor (for example, $P(X_{A,r} = 1, X_{B,r} = 0, X_{A,p} = 0, X_{B,p} = 1) = Z^{-1} \exp -5$). We can consider that the features were derived from the logical expressions, $X_{A,r} \wedge X_{B,r}$, $X_{A,p} \wedge X_{B,p}$, $X_{A,r} \wedge X_{B,p}$, $X_{A,p} \wedge X_{B,r}$. Obviously this knowledge-base is not even consistent, but for MLNs this would not hurt. After learning, only the true statements

would survive with $w \to \infty$. Even better, statements which are often but not always true would obtain weights which reflect this frequency. This basic ideas is formalized in MLNs.

Let $F_k$ be a formula of first-order logic and let $w_k \in \mathbb{R}$ be a weight attached to each formula. Then a MLN $L$ is defined as a set of pairs $(F_k, w_k)$ [45] [7]. One introduces a binary node for each possible grounding of each predicate appearing in $L$, given a set of constants $c_1, \ldots, c_{|C|}$. The state of the node is equal to 1 if the ground atom/statement is true, and 0 otherwise (for an $Q$-ary predicate there are $|C|^Q$ such nodes). A grounding of a formula is an assignment of constants to the variables in the formula (considering formulas that are universally quantified). If a formula contains $Q$ variables, then there are $|C|^Q$ such assignments. The nodes in the Markov network $M_{L,C}$ are the grounded predicates. In addition the MLN contains one feature for each possible grounding of each formula $F_k$ in $L$. The value of this feature is 1 if the ground formula is true, and 0 otherwise. $w_k$ is the weight associated with $F_k$ in $L$. A Markov network $M_{L,C}$ is a grounded Markov logic network of $L$ with

$$P(\{X\} = \vec{x}) = \frac{1}{Z} \exp\left(\sum_k w_k n_k(\vec{x})\right)$$

where $n_k(\vec{x})$ is the number of formula groundings that are true for $F_k$. MLN makes the unique names assumption, the domain closure assumption and the known function assumption, but all these assumptions can be relaxed.

A MLN puts weights on formulas: the larger the weight, the higher is the confidence that a formula is true. When all weights are equal and become infinite, one strictly enforces the formulas and all worlds that agree with the formulas have the same probability.

The simplest form of inference concerns the prediction of the truth value of a grounded predicate given the truth values of other grounded predicates (conjunction of predicates) for which the authors present an efficient algorithm. In the first phase, the minimal subset of the ground Markov network is returned that is required to calculate the conditional probability. It is essential that this subset is small since in the worst case, inference could involve alle nodes. In the second phase Gibbs sampling in this reduced network is used.

Learning consists of estimating the $w_k$. In learning, MLN makes a closed-world assumption and employs a pseudo-likelihood cost function, which is the product of the probabilities of each node given its Markov blanket. Optimization is performed using a limited memory BFGS algorithm.

Finally, there is the issue of structural learning, which, in this context, defines the employed first order formulae. Some formulae are typically defined by a domain expert *a priori*. Additional formulae can be learned by directly optimizing the pseudo-likelihood cost function or by using ILP algorithms. For the latter, the authors use CLAUDIEN [41], which can learn arbitrary first-order clauses (not just Horn clauses, as many other ILP approaches).

## 3. Applications of ML to the Semantic Web

The interest of applying ML techniques in the Semantic Web context has been growing over recent years, and at the main Semantic Web conferences special ML tracks and workshops have been formed[4].

One of the applications of ML techniques that is discussed throughout this book is *ontology learning*. ML techniques may be used to both learn ontologies from scratch and enrich already existing ontologies. Learning data originates, e.g., from Linked Data, social networks, tags, textual data [30,8,27]. Another popular use of ML is the learning of the mapping from one ontology to another (e.g. based on association rules [5], or similarity-based methods [6] ).

A number of proposed approaches for *Learning from the Semantic Web* are based on ILP methods (e.g., classification [27] or association learning [19,26]). This kind of approach is supported by recently developed tools for ontology-based data mining such as DL-Learner [27][5], RMonto [38][6]) or SDM-Toolkit [49][7]. An interesting application of this kind was realized within the project e-LICO[8]. It consisted of optimizing knowledge discovery processes through ontology-based meta-learning, that is machine learning from meta data of executed past experiments, where meta data was represented with background ontologies [14][9]. [29] describes a perspective of ILP for the Semantic Web.

Ontology learning and ILP assume deterministic or close-to-deterministic dependencies. The increase of interest in ML techniques has arisen largely due to the open, distributed and inherently incomplete nature of the Semantic Web. Such a context makes it hard to apply purely deductive techniques, which traditionally have been dominating reasoning approaches for ontological data. As part of the LarKC project[10] a scalable machine learning approach has been developed that works well with the high-dimensional, sparse, and noisy data one encounters in those domains [47,15]. The approach is based on matrix factorization and has shown superior performance on a number of Semantic Web data sets [16]. Extensions have been developed that can take into account temporal effects and can model sequences [48] and can include ontological background and textual information [18]. The approach was part of the winning entry in the ISWC 2011 Semantic Web Challenge[11]. Tensor factorization is another promising direction, since the subject-predicate-object structure of the Semantic Web matches perfectly to the modes of a three-way tensor [35]. Another light-weighted approach is presented by [22], where the authors describe SPARQL-ML, a framework for adding data mining support to SPARQL. The approach uses relational bayes classifier (RBC) and relational probabilistic trees (RPT). In turn, [25] proposes to semantically group SPARQL query results via conceptual clustering.

An overview on early work on the application of ML to the Semantic Web can be found in [46] with applications described in [37]. Data mining perspectives for the

[4] Inductive Reasoning and Machine Learning for the Semantic Web (IRMLeS) workshops, `http://irmles.di.uniba.it`

[5] `http://aksw.org/Projects/DLLearner`

[6] `http://semantic.cs.put.poznan.pl/RMonto`

[7] `http://sourceforge.net/p/sdmtoolkit/`

[8] `http://www.e-lico.eu`

[9] `http://www.dmo-foundry.org`

[10] `http://www.larkc.eu`

[11] `http://challenge.semanticweb.org`

Semantic Web have been described by [1,31]. More recent overviews are presented in [4] and in [43].

## 4. Conclusions

In this chapter we have discussed machine learning as the basis for the remaining chapters in this book. With an increasing amount of data published in the format of the Semantic Web, we feel that the number of machine learning applications will certainly grow. Due to space limitations we could only cover a few aspects we felt *are* most relevant. By now machine learning is a large research areas with a multitude of theories, approaches and algorithms. We feel that there will not be one dominating approach towards machine learning on the Semantic Web but that we can expect creative solutions from different machine learning research areas.

## References

[1] Bettina Berendt, Andreas Hotho, and Gerd Stumme. Towards Semantic Web Mining. In *Proc. of the First International Semantic Web Conference on The Semantic Web*, ISWC '02, pages 264–278, London, UK, 2002. Springer-Verlag.

[2] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proc. of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pages 144–152, New York, NY, USA, 1992. ACM.

[3] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, Cambridge, UK, March 2000.

[4] Claudia d'Amato, Nicola Fanizzi, and Floriana Esposito. Inductive learning for the Semantic Web: What does it buy? *Semantic Web*, 1(1-2):53–59, 2010.

[5] Jérôme David, Fabrice Guillet, and Henri Briand. Matching directories and OWL ontologies with AROMA. In *Proc. of the 15th ACM International Conference on Information and Knowledge Management*, CIKM '06, pages 830–831, New York, NY, USA, 2006. ACM.

[6] AnHai Doan, Jayant Madhavan, Pedro Domingos, and Alon Y. Halevy. Ontology matching: A machine learning approach. In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, pages 385–404. Springer, 2004.

[7] Pedro Domingos and Matthew Richardson. Markov logic: A unifying framework for statistical relational learning. In Lise Getoor and Ben Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press, 2007.

[8] Nicola Fanizzi, Claudia D'Amato, and Floriana Esposito. DL-FOIL concept learning in description logics. In *Proc. of the 18th International Conference on Inductive Logic Programming*, ILP '08, pages 107–121, Berlin, Heidelberg, 2008. Springer-Verlag.

[9] Roger Fletcher. *Practical methods of optimization; (2nd ed.)*. Wiley-Interscience, New York, NY, USA, 1987.

[10] Lise Getoor, Nir Friedman, Daphne Koller, Avi Pferrer, and Ben Taskar. Probabilistic relational models. In Lise Getoor and Ben Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press, 2007.

[11] Jiawei Han and Micheline Kamber. *Data Mining. Concepts and Techniques*. Morgan Kaufmann, 2nd edition, 2006.

[12] David J. Hand, Padhraic Smyth, and Heikki Mannila. *Principles of data mining*. MIT Press, Cambridge, MA, USA, 2001.

[13] David Heckerman. A tutorial on learning with Bayesian Networks. Technical report, Microsoft Research, 1995.

[14] Melanie Hilario, Phong Nguyen, Huyen Do, Adam Woznica, and Alexandros Kalousis. Ontology-based meta-mining of knowledge discovery workflows. In Norbert Jankowski, Wlodzislaw Duch, and Krzysztof Grabczewski, editors, *Meta-Learning in Computational Intelligence*, pages 273–316. Springer, 2011.

[15] Yi Huang, Volker Tresp, Markus Bundschus, Achim Rettinger, and Hans-Peter Kriegel. Multivariate prediction for learning on the semantic web. In *Proc. of the 20th International Conference on Inductive Logic Programming*, ILP'10, pages 92–104, Berlin, Heidelberg, 2011. Springer-Verlag.

[16] Yi Huang, Volker Tresp, Maximilian Nickel, Achim Rettinger, and Hans-Peter Kriegel. A scalable approach for statistical learning in semantic graphs. *Semantic Web Interoperability, Usability, Applicability (SWJ)*, 2012.

[17] Manfred Jaeger. Relational bayesian networks. In *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence (UAI)*, 1997.

[18] Xueyan Jiang, Yi Huang, Maximilian Nickel, and Volker Tresp. Combining information extraction, deductive reasoning and machine learning for relation prediction. In *Proc. of the 9th International Conference on the Semantic Web: Research and Applications*, ESWC'12, pages 164–178, Berlin, Heidelberg, 2012. Springer-Verlag.

[19] Joanna Józefowska, Agnieszka Ławrynowicz, and Tomasz Lukaszewski. The role of semantics in mining frequent patterns from knowledge bases in description logics with rules. *TPLP*, 10(3):251–289, 2010.

[20] Charles Kemp, Joshua B. Tenenbaum, Thomas L. Griffiths, Takeshi Yamada, and Naonori Ueda. Learning systems of concepts with an infinite relational model. In *Proc. of the 21st National Conference on Artificial Intelligence - Volume 1*, AAAI'06, pages 381–388. AAAI Press, 2006.

[21] Kristian Kersting and Luc De Raedt. Bayesian logic programs. Technical report, Albert-Ludwigs University at Freiburg, 2001.

[22] Christoph Kiefer, Abraham Bernstein, and André Locher. Adding data mining support to SPARQL via statistical relational learning methods. In *Proc. of the 5th European Semantic Web Conference*, ESWC'08, pages 478–492, Berlin, Heidelberg, 2008. Springer-Verlag.

[23] Daphne Koller and Avi Pfeffer. Probabilistic frame-based systems. In *Proc. of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*, AAAI '98/IAAI '98, pages 580–587, Menlo Park, CA, USA, 1998. American Association for Artificial Intelligence.

[24] Markus Krötzsch, Frantisek Simančík, and Ian Horrocks. A description logic primer. In Jens Lehmann and Johanna Völker, editors, *Perspectives on Ontology Learning*, Studies on the Semantic Web. AKA Heidelberg / IOS Press, 2014.

[25] Agnieszka Ławrynowicz. Grouping results of queries to ontological knowledge bases by conceptual clustering. In Ngoc Thanh Nguyen, Ryszard Kowalczyk, and Shyi-Ming Chen, editors, *ICCCI*, volume 5796 of *LNCS*, pages 504–515. Springer, 2009.

[26] Agnieszka Ławrynowicz and Jedrzej Potoniec. Fr-ONT: An algorithm for frequent concept mining with formal ontologies. In Marzena Kryszkiewicz, Henryk Rybinski, Andrzej Skowron, and Zbigniew W. Ras, editors, *ISMIS*, volume 6804 of *LNCS*, pages 428–437. Springer, 2011.

[27] Jens Lehmann. DL-learner: Learning concepts in description logics. *Journal of Machine Learning Research*, 10:2639–2642, 2009.

[28] Jens Lehmann, Nicola Fanizzi, and Claudia d'Amato. Concept learning. In Jens Lehmann and Johanna Völker, editors, *Perspectives on Ontology Learning*, Studies on the Semantic Web. AKA Heidelberg / IOS Press, 2014.

[29] Francesca A. Lisi and Floriana Esposito. An ILP perspective on the Semantic Web. In *Semantic Web Applications and Perspectives, Proceedings of the 2nd Italian Semantic Web Workshop*, 2005.

[30] Alexander Maedche and Steffen Staab. Ontology learning. In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, pages 173–190. Springer, 2004.

[31] Peter Mika. *Social Networks and the Semantic Web*. Springer, 2007.

[32] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[33] Stephen Muggleton. Inductive logic programming. *New Generation Computing*, 8(3):295–318, 1991.

[34] Stephen Muggleton. Inverse entailment and PROGOL. *New Generation Computing*, 13:245–286, 1995.

[35] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In Lise Getoor and Tobias Scheffer, editors, *Proc. of the 28th International Conference on Machine Learning (ICML-11)*, ICML '11, pages 809–816, New York, NY, USA, June 2011. ACM.

[36] Shan-Hwei Nienhuys-Cheng and Ronald de Wolf. *Foundations of inductive logic programming*, volume 1228 of LNAI. Springer, 1997.

[37] Hector Oscar Nigro, Sandra Gonzalez Cisaro, and Daniel Hugo Xodo. *Data Mining With Ontologies:*

*Implementations, Findings and Frameworks*. Information Science Reference - Imprint of: IGI Publishing, Hershey, PA, 2007.

[38] Jedrzej Potoniec and Agnieszka Ławrynowicz. RMonto: Ontological extension to RapidMiner. In *Poster and Demo Session of the 10th International Semantic Web Conference*, Bonn, Germany, 2011.

[39] J. Ross Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.

[40] Luc De Raedt. *Logical and relational learning*. Cognitive Technologies. Springer, 2008.

[41] Luc De Raedt and Luc Dehaspe. Clausal discovery. *Machine Learning*, 26, 1997.

[42] Stefan Reckow and Volker Tresp. Integrating ontological prior knowledge into relational learning. Technical report, Siemens, 2007.

[43] Achim Rettinger, Uta Lösch, Volker Tresp, Claudia d'Amato, and Nicola Fanizzi. Mining the semantic web - statistical learning for next generation knowledge bases. *Data Min. Knowl. Discov.*, 24(3):613–662, 2012.

[44] Achim Rettinger, Matthias Nickles, and Volker Tresp. Statistical relational learning with formal ontologies. In *Proc. of the European Conference on Machine Learning and Knowledge Discovery in Databases: Part II*, ECML PKDD '09, pages 286–301, Berlin, Heidelberg, 2009. Springer-Verlag.

[45] Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine Learning*, 62(1-2), 2006.

[46] Gerd Stumme, Andreas Hotho, and Bettina Berendt. Semantic Web Mining: State of the art and future directions. *J. Web Sem.*, 4(2):124–143, 2006.

[47] Volker Tresp, Yi Huang, Markus Bundschus, and Achim Rettinger. Materializing and querying learned knowledge. In *Proceedings of the First ESWC Workshop on Inductive Reasoning and Machine Learning on the Semantic Web*, 2009.

[48] Volker Tresp, Yi Huang, Xueyan Jiang, and Achim Rettinger. Graphical models for relations - modeling relational context. In Joaquim Filipe and Ana L. N. Fred, editors, *KDIR*, pages 114–120. SciTePress, 2011.

[49] Anže Vavpetič and Nada Lavrač. Semantic subgroup discovery systems and workflows in the SDM-Toolkit. *The Computer Journal*, 2012.

[50] Zhao Xu, Volker Tresp, Kai Yu, and Hans-Peter Kriegel. Infinite hidden relational models. In *Uncertainty in Artificial Intelligence (UAI)*, 2006.

# Natural Language Processing

Diana MAYNARD and Kalina BONTCHEVA
*Dept. of Computer Science, University of Sheffield, UK*
*diana@dcs.shef.ac.uk*

**Abstract.** This chapter provides a high-level overview of the various NLP processes typically required for an ontology learning system,ranging from low-level linguistic pre-processing, through parsing, term recognition and information extraction. Since ontology learning research tends to reuse many existing NLP tools, this chapter also discusses some of the most widely used, open-source ones, providing references to further reading materials.

**Keywords.** Natural Language Processing, term recognition

## Introduction

Natural Language Processing (NLP) is concerned with building algorithms that understand and generate natural language text (also often referred to as unstructured content). Since ontology learning methods automatically derive (parts of) an ontology from textual sources, they tend to make use of NLP tools and techniques in order to help with the text analysis. The aim of this chapter is to provide a high-level overview of the various NLP tasks, ranging from low-level linguistic pre-processing, through parsing, term recognition and information extraction. Since ontology learning research tends to reuse many existing NLP tools, this chapter also discusses some of the most widely used, open-source ones, providing references to further reading materials. We have, however, focused primarily on the GATE architecture [23] because, in our opinion, it offers one of the most flexible infrastructures for incorporating NLP components, and provides the widest variety of plugins for different NLP tasks.

As the chapter progresses, the NLP tasks and algorithms become progressively more complex and, consequently, more error-prone. For instance, tokenisation, sentence splitting and part of speech (POS) tagging are typically performed with 95-98% accuracy, entity recognition between 90-95%, and even less for parsing. Domain adaptation is also a major issue, and again performance in different domains varies more widely as the tasks become more complex. Consequently, there is a trade-off between the sophistication of the linguistic analysis and its accuracy, especially on unseen, noisier types of text. As a result, it is advisable, as an integral part of system development, to carry out rigorous quantitative evaluation against a gold standard dataset. It is through such experiments that it is possible to establish the usefulness of each of the NLP processing steps for the ontology learning results. On large datasets, computational complexity and implementation efficiency would also need to be considered.

In general, the easiest way to carry out such quantitative experimentation is to build easily reconfigurable NLP pipelines. A typical NLP pipeline consists of a number of

tools applied in sequence (tokenisation, sentence splitting, part-of-speech tagging, entity recognition, etc). To help with the pipeline building and quantitative evaluation tasks, researchers typically use a general purpose NLP infrastructure. Some of the most popular ones are open-source and come with a large number of already implemented NLP tools (e.g. GATE [23], NLTK [46], OpenNLP[1], UIMA [32]). The advantages of reusing NLP tools from such an infrastructure are several:

- They support a variety of text formats including HTML, XML, RTF, email, plain text and in some cases Word, PDF and Excel files. When a document is processed, the format is analysed transparently to the user and converted into a single unified model of annotations. Multilingual support is well-tested and based on Unicode.
- All low-level components within an infrastructure are designed to be interoperable and, consequently, there is no overhead in putting them together into a single pipeline.
- Results storage, evaluation tools, and other facilities are taken care of by the infrastructure. In the case of GATE, this also comes with a graphical development environment, called GATE Developer, which makes it easy to build and test pipelines visually.

Since GATE integrates OpenNLP low-level processing components, it is also possible to mix and match GATE modules and OpenNLP modules, as well as visually and quantitatively (i.e. using precision, recall, and f-measure) compare the performance of these alternative implementations, from within the GATE Developer user interface or programmatically via the GATE API.

NLTK [46] is suitable when Python is preferred as a programming language. Given that NLTK was primarily developed for the purpose of teaching NLP, the efficiency of some of the implementations might not be suitable for the processing of very large datasets.

## 1. Linguistic Pre-Processing Tasks

There are a number of low-level linguistic tasks which form the basis of more complex language processing and ontology learning algorithms. This section will provide an overview and point to some existing open-source implementations which can easily be reused and, in some cases, are easily adaptable as well.

### *1.1. Tokenisation*

Tokenisation is the task of splitting the input text into very simple units, called tokens. Tokenisation is a required step in any linguistic processing application, since more complex algorithms typically work on tokens as their input, rather than using the raw text. Consequently, it is important to use a high-quality tokeniser, as errors are likely to affect the results of all subsequent NLP algorithms.

Commonly distinguished types of tokens are numbers, symbols (e.g., $, %), punctuation and words of different kinds, e.g., uppercase, lowercase, mixed case. Tokenising well-written text is generally reliable and reusable, since it tends to be domain-

[1] `http://incubator.apache.org/opennlp/`

independent. However, such general purpose tokenisers typically need to be adapted to work correctly with, for example, chemical formulae, twitter messages, and other more specific text types.

One widely used tokeniser is bundled in the open-source ANNIE system in GATE [24]. Similar to the way in which programming languages are tokenised, the ANNIE Tokeniser relies on a set of regular expression rules which are then compiled into a finite-state machine. This differs from most other tokenisers in that it maximises efficiency by doing only very light processing, and enabling greater flexibility by placing the burden of deeper processing on the grammar rules, which are more adaptable (see Section 3). For example, there is a specialised set of rules for tokenisation of English, which deals with expressions such as "don't" which would by default be tokenised as 3 tokens, whereas for correct POS tagging and syntactic processing they need to be tokenised as "do" and "n't" as the short form of not. If Python is preferred, NLTK has several similar tokenisers, one based on regular expressions.

Another freely-available tokeniser is the OpenNLP TokenizerME[2], which is a trainable maximum entropy tokeniser. It uses a statistical model, based on a training corpus. There is a method also for re-training the tokeniser on new data. However, this dependency on training data means that tokeniser adaptation to new types of text, e.g., twitter messages, is likely to be expensive, as it would require a substantial amount of training data.

In our experience, human readable tokenisation rules such as those used in the ANNIE and OpenNLP tokenisers tend to be easier to adapt to new languages and text types than those built from statistical models. For example, the following tokeniser rule is from the ANNIE Tokeniser:

```
'UPPERCASE_LETTER' 'LOWERCASE_LETTER'* >
  Token;orth=upperInitial;kind=word;
```

It states that the sequence must begin with an uppercase letter, followed by zero or more lowercase letters. This sequence will then be annotated as type 'Token'. The attribute 'orth' (orthography) has the value 'upperInitial'; the attribute 'kind' has the value 'word'.

### *1.2. Sentence Splitting*

Sentence detection (or sentence splitting) is the task of separating text into its constituent sentences. For plain text, sentence splitting is concerned pretty much solely with determining whether a punctuation token (e.g. ".", "?", ":") marks the end of a sentence or not. More complex cases arise when the text being processed contains tables, titles, formulae, or other formatting markup (e.g. HTML tags, hashtags in tweets).

The GATE sentence splitter is a cascade of finite-state transducers which segment the text into sentences, based on a set of rules. This module is required for the POS tagger in GATE, and is often necessary for other low-level processing. It is domain- and application-independent, although some adaptation to text formats might be required. Each sentence is annotated with the type 'Sentence' and a separate sentence break annotation is also produced.

[2] `http://incubator.apache.org/opennlp/documentation/manual/opennlp.html`

GATE also offers a RegEx splitter, which is based on regular expressions, using the default Java implementation. This has the advantage of being easily customisable by Java programmers. It has three sets of patterns for:

- sentence splits that are part of the sentence, such as sentence-ending punctuation;
- sentence splits that are not part of the sentence, such as 2 consecutive new lines;
- text fragments that might be seen as splits but which should be ignored (such as full stops occurring inside abbreviations).

The OpenNLP Sentence Detector is designed to run prior to tokenisation and is a trainable one. Punctuation marks are required as indicators for sentence boundaries. Conquently, it cannot identify sentence boundaries based on new lines, markup tags, or sentence content, e.g. titles, whereas the GATE ones are more flexible in that respect. The Sentence Detector takes a trained model file as input and produces an array of sentences. Similar to the OpenNLP tokeniser, this might make it harder to adapt to new types of text, compared with the regular expression and rule-based ones, which can be changed directly.

While generally a simple problem for humans, automatic sentence splitting is not without challenges. For instance, abbreviations need to be recognised and dealt with properly, as well as carriage returns and newlines. Some splitters ignore these completely, requiring a punctuation mark as a sentence boundary. Others use two consecutive newlines/carriage returns as an indication of a sentence end, while there are also cases when even a single newline/carriage return character would indicate end of a sentence (e.g. comments in software code or lists which have one entry per line). HTML formatting tags, Twitter hashtags, wiki syntax, and other such special text types are also somewhat problematic for general-purpose sentence splitters which have been trained on well-written corpora, typically newspaper texts.

### *1.3. POS Tagging*

Part-of-Speech (POS) tagging is concerned with tagging words with their part of speech, by taking into account the word itself, as well as the context in which it appears. A key part of this task is the tagset used and the distinctions that it makes. The main categories are verb, noun, adjective, adverb, preposition, etc. However, tagsets tend to be much more specific, e.g. distinguishing between singular and plural nouns. One commonly used tagset is the Penn treebank one [47].

In terms of approaches, researchers have achieved excellent results with Hidden Markov models, rule-based approaches, maximum entropy, and many other methods. GATE's English POS tagger [40] is a modified version of the Brill transformational rule-based tagger [7], which produces a part-of-speech tag as an annotation on each word or symbol, using the Penn treebank tagset. The tagger uses a default lexicon and ruleset (the result of training on a large corpus taken from the Wall Street Journal). Both of these can be modified manually if necessary.

Similarly, the OpenNLP POS tagger uses a model learnt from a training corpus to predict the correct POS tag from the Penn treebank tagset. During training, it is possible to build either a maximum entropy or a perceptron-based model.

For Python, NLTK also has an implementation of the Brill tagger, as well as the TNT statistical tagger [6] and the Stanford POS tagger [65].

The accuracy of these general purpose, reusable taggers is typically excellent (97-98%) on texts similar to those on which the taggers have been trained (mostly news articles). Consequently, when presented with new text types or noisier data, the accuracy declines. In some cases, changes to the tagger rules and/or re-training might be required. For instance, Hearst patterns (see Section 6), which are widely used in ontology learning, need reliable POS tags in order to produce high-quality results.

### *1.4. Stemming and Morphological Analysis*

Another set of useful low-level processing components are stemmers and morphological analysers. Stemmers produce the stem form of each word, e.g. “driving” and “drivers” have the stem “drive”, whereas morphological analysis tends to produce the root/lemma forms of the words and their affixes, e.g. “drive” and “driver” for the above examples, with affixes “ing” and “s” respectively.

GATE provides a wrapper for the widely used, open-source Snowball stemmers, which cover 11 European languages (Danish, Dutch, English, Finnish, French, German, Italian, Norwegian, Portuguese, Russian, Spanish and Swedish) and makes them straightforward to combine with the other low-level linguistic components. The stemmers are rule-based [59] and easy to modify, following the suffix-stripping approach of Porter. NLTK also provides an implementation of the Snowball stemmers for Python.

The English morphological analyser in GATE is also rule-based, with the rule language supporting rules and variables that can be used in regular expressions in the rules. POS tags can taken into account if desired, depending on a configuration parameter. At the time of writing, OpenNLP and NLTK do not provide morphological analysers.

## 2. Named entity recognition

Named entity recognition (NER) is used to automatically derive the semantics from textual content, using linguistic and/or statistical knowledge. It consists of the identification of proper names in texts, and their classification into a set of predefined categories of interest. The core set of traditional named entities are Person, Organisation, Location and Date and Time expressions, such as "John Smith", "IBM", "London", "4th August 2011" etc. respectively. Various other types of named entity are frequently included, as appropriate to the application, e.g. newspapers, ships, monetary amounts, percentages etc. NER provides a foundation from which to build more complex IE systems. For example, extracting the relations between entities can provide the means for entity tracking (finding co-references), ontological information (e.g. distinguishing between “Athens, Georgia” and “Athens, Greece”), and scenario building.

Approaches can be divided into pattern-based and statistical extraction [16], although quite often the two techniques are mixed (see e.g. [58][17][63]). Most information extraction (IE) techniques rely on some form of human supervision, with the exception of purely structural IE techniques performing unsupervised machine learning on unannotated documents, e.g. [22]. A survey of information extraction methods from web data is presented in [12]. Language engineering platforms such as GATE enable the modular implementation of techniques and algorithms for information extraction, and allow repeatable experimentation and evaluation of their results.

Linguistic rule-based methods for NER, such as those used in ANNIE, GATE's information extraction system, comprise a combination of gazetteer lists and hand-coded pattern-matching rules which use contextual information to help determine whether candidate entities are valid, or to extend the set of candidates. The gazetteer lists act as a starting point from which to establish, reject, or refine the final entity to be extracted. A typical processing pipeline consists of linguistic pre-processing (tokenisation, sentence splitting, POS tagging), entity finding (using gazetteers and grammars), co-reference, and finally some kind of export of the results to a database or ontology. Section 3 discusses in more detail some of the techniques for pattern-based rule writing used for NER.

Learning methods for NER can be classified broadly into two main categories: rule learning and statistical learning. The former methods induce a set of rules from training examples, e.g. SRV [34], RAPIER [9], WHISK [64], BWI [35], and $LP^2$ [18]. Statistical systems learn statistical models or classifiers, such as HMMs [55], Maximum Entropy [15], SVM [42] [48] [44] and Perceptron [10][45]. Methods differ widely in the NLP features that they use, including simple features such as token string and capitalisation information, linguistic features such as part-of-speech, semantic information from gazetteer lists, and genre-specific information such as document structure.

The general approach consists of three stages: linguistic pre-processing to obtain the feature vectors, training or applying classifiers, and finally post-processing the results to tag the documents. There are advantages and disadvantages to the Machine Learning approach compared with a knowledge engineering, rule-based approach. First, large quantities of training data are required, which can be problematic, especially as these need to be relevant to the domain and the set of entities required. If any criteria change (such as a new entity type), then the whole training set may need to be reannotated. On the other hand, ML techniques have the advantage of not requiring specialist language engineers to develop hand-coded rules, which can be time-consuming to develop.

GATE's general purpose named entity recognition system is ANNIE, which was designed for traditional NER on news texts, but which, being easily adaptable, can form the starting point for new NER applications in other languages and for other domains. Other well known systems are UIMA[3], developed by IBM, which focuses more on architectural support and processing speed, and offers a number of similar resources to GATE; OpenCalais[4], which provides a web service for semantic annotation of text for traditional named entity types, and LingPipe[5] which provides a (limited) set of Machine Learning models for various tasks and domains: while these are very accurate, they are not easily adaptable to new applications. Components from all these tools are actually included in GATE, so that a user can mix and match various resources as needed, or compare different algorithms on the same corpus.

## 3. Pattern-based rule writing

Using pattern matching for Named Entity Recognition (NER) requires the development of patterns over multi-faceted structures that consider many different token properties (e.g orthography, morphology, part of speech information etc.). Traditional pattern-

[3] `http://uima.apache.org`
[4] `http://www.opencalais.com/`
[5] `http://alias-i.com/lingpipe/index.html`

matching languages such as PERL get "hopelessly long-winded and error prone" [4], when used for such complex tasks. Therefore, attribute-value notations are normally used, which allow for conditions to refer to token attributes arising from multiple analysis levels. Examples of these are the NEA notation [4] and JAPE [23], both of which are declarative notations that allow for context-sensitive rules to be written and for non-deterministic pattern matching to be performed. NEA was used in FACILE, a named entity recognition tool used in the early MUC evaluations, and was then adapted to the needs of the CONCERTO project [3,56]. while JAPE is the standard rule-writing mechanism used in GATE.

Traditional rule-based NER is based on a set of linguistic patterns which aim to identify the relevant entities in text. These rely largely on gazetteer lists which provide all or part of the entity, or clues to its existence, in combination with linguistic patterns. For example, a typical rule to identify a person's name consists of matching the first name of the person via a gazetteer entry (e.g. *John*), followed by an unknown proper noun (e.g. *Smith*, which is POS-tagged as a proper noun). In this section we introduce the concept of pattern-based rule writing, using the example of the JAPE language.

### *3.1. JAPE*

JAPE (Java Annotations Pattern Engine) is a language for writing regular expressions over annotations and for using pattern matching as a basis for creating or manipulating annotations. It is based on the Common Pattern Specification Language (CPSL), developed in the TIPSTER Program. The rules are divided into phases which run sequentially, constituting a cascade of Finite State Transducers over annotations. Each phase consists of rules for the same entity types (e.g "Person") or rules that have the same specific requirements for being run (e.g. rules that cover elision cases; this phase needs to run after normal full-entity expressions have been recognised).

Basic rules in JAPE are of the form:

```
(Optional Context)
(Entity to be recognised):label
(Optional Context)
-->
:label.EntityType={annotation attributes}
```

At the beginning of each phase, one needs to specify the set of *input annotations* on which the current rule phase relies for the matching, and the *matching style*, which constrains the priorities about which rule should be fired when more than one satisfies the match (e.g. longest first, shortest first, explicit priority statements, and so on). The LHS of the rule contains the pattern to be matched in the text: this is always specified in terms of annotations (and optionally features and values) and regular expression iteration operators. The RHS of the rule contains information about the annotation to be generated or other action to be performed. Arbitrary Java code may also be used here for more complex procedures, e.g. feature percolation and manipulation.

### *3.2. Development and reuse of grammars*

The application for which the NER task is designed clearly has an impact on the development of the grammars. It can involve considerable effort to adapt a grammar to a new

domain or task, particulary if different entity types are needed, or if the same entities have different structures and syntactic behaviour in their new context. However, adding rules for new entity types and changing some other rules for the needs of a new domain or text type/genre is less effort and time consuming than building everything from scratch. There should always be some subset of a purpose-built NE recognition grammar set that is application independent; this part can be used as a basis for creating a new grammar set for a new application, no matter how different one application is from another. This set of core rules might correspond to the basic named entities (person, organisation, location names) and fixed data structures (date, time and monetary expressions), traditionally identified by any NER system, which are largely domain independent. There are many examples of adapting grammars to new domains and languages: see for example [49,51].

## 4. Term recognition

Automatic term recognition (also known as term extraction) is a crucial component of many knowledge-based applications such as automatic indexing, knowledge discovery, terminology mining and monitoring, knowledge management and so on. It is particularly important in the healthcare and biomedical domains, where new terms are emerging constantly, and often goes hand in hand with named entity recognition, especially for applications such as ontology learning. The main aim of term recognition is to determine whether a word or a sequence of words is a term that characterises the target domain. This involves the identification and filtering of term candidates for the purpose of identifying domain-relevant terms or entities. Most term extraction methods use a combination of linguistic filtering (e.g. possible sequences of part of speech tags) and statistical measures (e.g. tf.idf [61]), to determine the salience of each term candidate for each document in the corpus [27,54].

Named entity recognition and automatic term recognition are not entirely mutually exclusive. A "term" refers to a specific concept characteristic of a domain, so while a named entity such as Person or Location is generic across all domains, a technical term such as "myocardial infarction" is only considered a relevant term when it occurs in a medical domain. If we were interested in sporting terms then it would probably not be considered a relevant term, even if it occurred in a sports article. As with named entities, however, terms are generally formed from noun phrases (in some contexts, verbs may also be considered terms). As we have already discussed, however, term extraction generally makes use of frequency-based information whereas typically named entity recognition uses a more linguistic basis.

Term recognition has been performed on the basis of various criteria. The main distinction we can make is between algorithms that only take the distributional properties of terms into account, such as frequency and tf.idf, and extraction techniques that use the contextual information associated with terms. In very small and/or specialised domains, as are typically used as a testbed for term recognition, statistical information may be skewed due to data sparsity. On the other hand, it is also difficult to extract suitable semantic information from such specialised corpora, particularly as appropriate linguistic resources may be lacking. Although contextual information has been used, e.g. in general language [36], and in the NC-Value method [33], it only consists of very shallow semantic information. Ranging from basic statistical methods for term recognition, such

as the tf.idf score, methods tend to build on this with further linguistic and contextual information. For example, the NC-Value method which includes part-of-speech information about the candidate term, along with information about frequency of co-occurrence with context words. TRUCKS [53] extends this by also measuring, amongst other things, the strength of association of contextual words with relevant candidate terms.

Unlike many of the other tools described in this chapter, there are many techniques, but few complete open-source off-the-shelf tools available for term recognition that can be easily incorporated into existing frameworks. Some of the more well-known tools for term recognition are LEXTER [5] for French, ACABIT [25], which has been successfully used for English, French and Japanese amongst others, and TERMINO [37], a large-scale terminological resource for text processing in the biomedical domain, available as a web service.

## 5. Parsing and Chunking

Syntactic parsing is concerned with analysing sentences to derive their syntactic structure according to a grammar, e.g. finding the subject and object of a verb. There are many different syntactic theories in computational linguistics research, but due to space limitations here we will discuss briefly several general-purpose, wide-coverage parsers which have been used for ontology learning, such as the Minipar[6] dependency parser, the RASP [8] statistical parser, and the Stanford [43] statistical parser. These are all available within GATE, so using GATE Developer's user interfact to experiment with them is one of the easiest ways to compare them and determine which is the most appropriate one for the task at hand.

Parsing algorithms can be computationally expensive and tend to work best on the kinds of texts that they are trained on. In addition, accuracy of parsing is significantly worse than that of the lower level pre-processing tools. Consequently, using a syntactic parser could introduce significant errors and thus limit the gains in terms of ontology learning performance. Therefore, it is advisable to carry out some cost-benefit experiments in order to determine whether using a parser is worthwhile for the particular ontology learning problem.

In some cases, ontology learning algorithms need to focus only on analysing noun phrases or verb phrases, without taking into account syntactic relationships between them. Shallow parsing (also called chunking) is concerned only with finding the boundaries of phrases without deducing their attachment and internal structure. This makes chunking more efficient and also less error-prone on new types of text.

### *5.1. Large-coverage, Reusable Syntatic Parsers*

Syntactic parsers are widely used to help ontology learning. Due to space limitations, below we cover only some of them in more detail. Other popular parsers are Bikel's statistical parser [2], the Collins statistical parser [20], and Charniak's parser [13].

Minipar is a dependency parser, i.e. it determines the dependency relationships between the words in a sentence. In more detail, it identifies the head modified by each word and the name of the dependency relationship between the two. Examples of depen-

[6] `http://www.cs.ualberta.ca/~lindek/minipar.htm`

dency relations are things like apposition, relative clauses, subjects and objects of verbs, and determiners. Minipar processes the text one sentence at a time and thus only needs a sentence splitter as a prerequisite.

The RASP statistical parser [8] is a domain-independent, robust parser for English. It comes also with its own tokeniser, POS tagger, and morphological analyser included. As with Minipar, it requires the text to be already segmented into sentences. RASP is available under the LGPL license and can thus be used also in commercial applications.

The Stanford statistical parser[7] [43] is a probabilistic parsing system. It provides either a dependency output or a phrase structure output. The latter can be viewed in its own GUI or through the user interface of GATE Developer. The Stanford parser comes with data files for parsing Arabic, Chinese, English, and German and is licensed under GNU GPL.

### *5.2. Reusable Chunkers*

The GATE verb chunker is based on a number of grammar rules for English [19] [1], and contains 68 rules for the identification of non-recursive verb groups. The rules cover finite ('is investigating'), non-finite ('to investigate'), participles (investigated'), and special verb constructs ('is going to investigate'). All the forms may include adverbials and negatives. The rules have been implemented in JAPE (see Section 3.1) and use the output of the POS tagger as well as information about the identity of the tokens (e.g. the token 'might' is used to identify modals).

The GATE Noun Phrase (NP) Chunker is a Java implementation of the Ramshaw and Marcus BaseNP chunker [60] which marks noun phrases in text, based on their POS tags. The output from this version should be identical to the output of the original C++/Perl version. The one major difference is in the assumption that if a POS tag is not in the mapping file then it is tagged as 'I'. The original algorithm simply fails if an unknown POS tag is encountered.

Both chunkers require that the text is first processed with a tokeniser, sentence splitter, and a POS tagger.

OpenNLP also provides a chunker with a pre-packaged English maximum entropy chunker model. The OpenNLP chunker chunks all tokens of a sentence, based on their POS tags (using the Penn Treebank tagset). In other words, it produces NP and VP chunks in one pass, instead of as two separate components, as in GATE. The OpenNLP chunker is trainable and requires a tagged training corpus, one word per line, with POS tag and a chunk type assigned (e.g. B-NP marks the first word of an NP chunk and I-NP is used for all subsequent words in that chunk).

Given that different chunkers tend to produce different kinds of chunks (e.g. include prepositional phrases within noun phrase chunks or not), it is advisable to compare the outputs of several different chunkers on the texts to be analysed and then decide which is the best one to use, given the concrete ontology learning task at hand.

### *5.3. Shallow Semantic Parsing*

Shallow semantic parsing (or semantic role labelling) is concerned with assigning semantic role information to the arguments of a verb in a sentence. Research has been driven

[7] `http://nlp.stanford.edu/software/lex-parser.shtml`

by the PropBank corpus [57], which has manually annotated semantic role labels (e.g. agent, theme) to Wall Street Journal articles. Another relevant resource is FrameNet[8], which is a large lexical database, srtuctured around semantic frames. It is also available in languages other than English, e.g. German, Spanish.

Some freely available tools for shallow semantic parsing are the probabilistic frame-based SEMAFOR parser[14] and Shalmaneser [28].

In the context of ontology learning, shallow semantic parsing is used for many tasks, including domain adaptation (see [21] in this volume).

## 6. Lexico-Syntactic Patterns

The extraction of lexico-syntactic patterns from text is frequently used to identify relationships between terms and entities. For the purposes of ontology creation, these can also then be re-engineered into concepts and instances. An increasing number of atomic ontology editing operations are associated with lexico-syntactic patterns, according to the ontological information these patterns and the participating entities contribute. The flexibility of this association enables the transformation of linguistic structures into lightweight ontological knowledge, performed in an incremental fashion.

Lexico-syntactic pattern-based ontology population has proven to be reasonably successful for a variety of tasks (see e.g. [29] for more details). The idea of acquiring semantic infomration from texts dates back to the early 1960s with Harris' *distributional hypothesis* [38] and Hirschman and Sager's work in the 1970s [41], which focused on determining sets of sublanguage-specific word classes using syntactic patterns from domain-specific corpora. A detailed description and comparison of lexical and syntactic pattern matching can be found in [52], In particular, research in this area has been used in specific domains such as medicine, where a relatively small number of syntactic structures is often found, for example in patient reports. Here the structures are also quite simple, with short and relatively unambiguous sentences typically found: this makes syntactic pattern matching much easier.

### *6.1. Hearst patterns*

One of the most common sets of lexico-syntactic patterns that indicate hyponymic relations are known as the Hearst patterns [39], and have been widely used for relation finding and ontology creation tasks. Typically they achieve a very high level of precision, but quite low recall: in other words, they are very accurate but only cover a small subset of the possible patterns for finding hyponyms and hypernyms. The patterns can be described by the following 5 rules, where NP stands for a Noun Phrase and the regular expression symbols have their usual meanings[9]:

1. `such NP as (NP,)* (or|and) NP`
   **Example:** ... works by such authors as Herrick, Goldsmith, and Shakespeare.
2. `NP (, NP)* , or other NP`
   **Example:** Bruises, wounds, broken bones or other injuries

[8] `http://framenet.icsi.berkeley.edu/`

[9] `()` for grouping; `|` for disjunction; `*`, `+`, and `?` for iteration.

3. `NP (, NP)* , and other NP`
   **Example:** temples, treasuries, and other important civic buildings.
4. `NP (,) including (NP,)* (or | and) NP`
   **Example:** all common-law countries, including Canada and England
5. `NP (,) especially (NP,)* (or | and) NP`
   **Example:** most European countries, especially France, England, and Spain.

Hearst defines the relations as hyponym-hypernym; however, if translating this to an ontology, we need to be more specific as the pairs could represent either instance-class or subclass-superclass relations. There are a number of ways in which this distinction could be made, depending on the ontology. For example, instances are generally represented by proper nouns, so one strategy could be to use POS tagging to indicate this. However, some POS taggers may mistag capitalised common nouns (e.g. at the beginning of sentences) as proper nouns frequently enough that it cannot be relied on as a strategy. Another method could be to look at the presence or absence of a determiner preceding the noun (since proper nouns in English are rarely accompanied by determiners) and whether the noun is singular or plural, but this still leaves the problem of the sentence-initial nouns. A safer solution might be to first recognise named entities in the text, and then only consider certain types of named entities (e.g. *Person*, *Location*, *Organization*) as candidates for instances; all other NPs are then considered to be classes. The exact strategy will depend on the requirements for the ontology, however.

### *6.2. Other lexico-syntactic patterns*

Because the Hearst patterns are very specific, they do not typically achieve wide coverage, so it is likely that further patterns will be necessary to supplement them. KnowItAll is a hybrid named-entity extraction system [30] that finds lists of instances of classes from the web using a search engine, combining Hearst patterns and learned patterns. Another possibility for adding to the Hearst patterns is the set of Lexico-Syntactic Patterns (LSPs) corresponding to Ontology Design Patterns (ODPs) [26]. We give some examples below, implemented in the SPRAT tool [50]. In these rules, `<sub>` and `<super>` are like variable names for the subclasses and superclasses to be generated; `CN` means *class of*, *group of*, etc.; `CATV` is a classification verb[10]; `PUNCT` is punctuation; `NPlist` is a conjoined list of NPs ("X, Y and Z").

1. Subclass rules
   - `NP<sub> be NP<super>`
   - `NPlist<sub> be CN NP<super>`
   - `NPlist<sub> (group (in|into|as) | (fall into) | (belong to)) [CN] NP<super>`
   - `NP<super> CATV CV? CN? PUNCT? NPlist<sub>`

   **Example:** *Frogs* and *toads* are kinds of *amphibian*.
2. Equivalence rules
   - `NP<class> be (the same as|equivalent to|equal to|like) NP<class>`

[10] e.g. *classify in/into*, *comprise*, *contain*, *compose (of)*, *group in/into*, *divide in/into*, *fall in/into*, *belong (to)*.

- `NP<class> (call | denominate | (designate by|as) | name) NP<class>` (where the verb is passive)
- `NP<class> have (the same|equal) (characteristic | feature | attribute | quality | property) as NP<class>`

**Example:** *Poison dart frogs* are also called *poison arrow frogs*.

3. Properties
   - `NP<class> have NP<property>`
   - `NP<instance> have NP <property>`

   **Example:** *Birds* have *feathers*.

While these patterns are quite productive (for example `X is a Y`), most of them are potentially ambiguous and susceptible to overgeneration. For example, in the following sentence:

Mistakenly, some artists and writers have penguins based at the North Pole.

clearly we do not want to recognise *penguin* as a property of *writer*. The difficulty is deciding where to draw the line between acceptable patterns and those that just overgenerate.

### *6.3. Pattern refinement*

One method of pattern refinement, to reduce overgeneration, is to add some semantic restrictions, e.g. using semantic categories from WordNet[31] and/or VerbNet[62]. The idea behind this is to look for verbal patterns connecting terms in a sentence, and to restrict the kinds of noun phrase extracted. This not only reduces the number of errors, but also eliminates the kind of general relations which while not incorrect, are not very useful. For example, knowing that a turtle is a local creature is not of much interest unless more contextual information is provided (i.e. to which region it is local). Combining statistically derived collocational information with lexico-syntactic patterns is another technique which has been proven to improve precision and recall [11].

Restrictions can be applied to various categories: for example, a noun phrase can be prevented from matching if it contains a stop word. The list of stop words might include some very common words which we do not want to recognise as adjectives, and can be determined either heuristically or using frequency analysis. Restrictions on properties is another very useful addition: for example, a pattern such as "X has a Y" is very general and can overgenerate massively. By adding semantic restrictions we can limit this, e.g. we can state that if X is an animal then Y must be a body part. Another restriction is the type of thing that can be considered a property, for example one can restrict the range of the property to certain semantic categories from WordNet, e.g. plant, shape, food, substance, object, body, animal, possession, artifact etc.

## 7. Summary

In this chapter, we have provided an overview of the main NLP tasks and techniques that are used to build up an application for language processing from unstructured text, as

a prerequisite to ontology learning. We have discussed some of the most widely used, open source tools, although we have focused primarily on the GATE architecture. As we have described in this chapter, an NLP system is typically composed of a pipeline of processing resources, which can be adapted for any particular application, language, or domain by adding, removing, replacing or editing individual components as necessary. For example, for adapting an NLP system to a new language, some components may be language-independent (such as tokenisers and sentence splitters), while others such as gazetteers and grammars may need to be modified or retrained for the new language. GATE, in particular, provides an easy mechanism for this, and also enables conditional processing so that the system can automatically choose the relevant resource for a text in a multilingual corpus based on the language of each text. We should stress also that no one system or component should be considered "bes", but only "best for a particular application, domain or task". This is why rigorous evaluation procedures and a good testbed are necessary when choosing an NLP system or a set of resources.

## References


[1] S. Azar. *Understanding and Using English Grammar*. Prentice Hall Regents, 1989.

[2] Daniel M. Bikel. Design of a multi-lingual, parallel-processing statistical parsing engine. In *Proceedings of the second international conference on Human Language Technology Research*, HLT '02, pages 178–182, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[3] B. Black, J. McNaught, F. Rinaldi, M. Ferraro, L. Gilardoni, S. Mazza, G.P. Zarri, A. Brasher, and A. Persidis. Detailed specification of the text extraction and concept recognition components of the concerto architecture. Deliverable 6, version 1.2, CONCERTO Consortium, 1999.

[4] W. Black, F. Rinaldi, and D. Mowatt. Facile: Description of the named entity system used for muc-7. In *Proceedings of the 7th MUC*, 1998.

[5] D. Bourigault. Surface grammatical analysis for the extraction of terminological noun phrases. In *Proc. of 14th International Conference on Computational Linguistics (COLING)*, pages 977–981, Nantes, France, 1992.

[6] Thorsten Brants. Tnt: a statistical part-of-speech tagger. In *Proceedings of the sixth conference on Applied Natural Language Processing*, ANLP '00, pages 224–231, 2000.

[7] E. Brill. A simple rule-based part-of-speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing*, Trento, Italy, 1992.

[8] Ted Briscoe, John Carroll, and Rebecca Watson. The second release of the rasp system. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, pages 77–80, 2006.

[9] M. Califf and R. Mooney. Relational learning of pattern-match rules for information extraction. *Working Papers of the ACL-97 Workshop in Natural Language Learning*, pages 9–15, 1997.

[10] X. Carreras, L. Màrquez, and L. Padró. Learning a perceptron-based named entity chunker via online recognition feedback. In *Proceedings of CoNLL-2003*, pages 156–159. Edmonton, Canada, 2003.

[11] S. Cederberg and D. Widdows. Using LSA and noun coordination information to improve the precision and recall of automatic hyponymy extraction. In *Proceedings of the 7th conference on Natural language learning at HLT-NAACL*, pages 111–118, Morristown, NJ, 2003.

[12] C.H. Chang, M. Kayed, M.R. Girgis, and K.F. Shaalan. A survey of web information extraction systems. *IEEE transactions on knowledge and data engineering*, pages 1411–1428, 2006.

[13] Eugene Charniak. A maximum-entropy-inspired parser. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 132–139. Morgan Kaufmann Publishers Inc., 2000.

[14] Desai Chen, Nathan Schneider, Dipanjan Das, and Noah A Smith. Semafor: Frame argument resolution with log-linear models. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 264–267. Association for Computational Linguistics, 2010.

[15] H. L. Chieu and H. T. Ng. Named entity recognition with a maximum entropy approach. In Walter Daelemans and Miles Osborne, editors, *Proceedings of CoNLL-2003*, pages 160–163. Edmonton, Canada, 2003.

[16] P. Cimiano, S. Handschuh, and S. Staab. Towards the Self-Annotating Web. In *Proc. of the 13th International Conference on World Wide Web (WWW'04)*, 2004.

[17] P. Cimiano, M. Hartung, and E. Ratsch. Learning the appropriate generalization level for relations extracted from the Genia corpus. In *Proc. of the 5th Language Resources and Evaluation Conference (LREC)*, 2006.

[18] F. Ciravegna. $(LP)^2$, an Adaptive Algorithm for Information Extraction from Web-related Texts. In *Proceedings of the IJCAI-2001 Workshop on Adaptive Text Extraction and Mining*, Seattle, 2001.

[19] Collins Cobuild, editor. *English Grammar*. Harper Collins, 1999.

[20] Michael Collins. Head-driven statistical models for natural language parsing. *Computational linguistics*, 29(4):589–637, 2003.

[21] Bonaventura Coppola, Aldo Gangemi, Alfio Gliozzo, Davide Picca, and Valentina Presutti. Learning domain ontologies by corpus-driven framenet specialization. In Johanna Völker and Jens Lehmann, editors, *Perspectives of Ontology Learning*, Studies on the Semantic Web. AKA Heidelberg / IOS Press, 2012.

[22] V. Crescenzi, G. Mecca, P. Merialdo, et al. Roadrunner: Towards automatic data extraction from large web sites. In *Proceedings of the international conference on very large data bases*, pages 109–118. Citeseer, 2001.

[23] H. Cunningham, D. Maynard, K. Bontcheva, V. Tablan, N. Aswani, I. Roberts, G. Gorrell, A. Funk, A. Roberts, D. Damljanovic, T. Heitz, M.A. Greenwood, H. Saggion, J. Petrak, Y. Li, and W. Peters. *Text Processing with GATE (Version 6)*. The University of Sheffield, 2011.

[24] Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. Gate: an architecture for development of robust hlt applications. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, 7–12 July 2002*, ACL '02, pages 168–175, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[25] B. Daille. Study and implementation of combined techniques for automatic extraction of terminology. *The Balancing Act: Combining Symbolic and Statistical Approaches to Language*, 1:49–66, 1996.

[26] G. Aguade de Cea, A. Gómez-Pérez, E. Montiel Ponsoda, and M-C. Suárez-Figueroa. Natural language-based approach for helping in the reuse of ontology design patterns. In *Proceedings of the 16th International Conference on Knowledge Engineering and Knowledge Management Knowledge Patterns (EKAW 2008)*, Acitrezza, Italy, September 2008.

[27] Paul Deane. A nonparametric method for extraction of candidate phrasal terms. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 605–613, 2005.

[28] Katrin Erk and Sebastian Pado. Shalmaneser – a flexible toolbox for semantic role assignment. In *Proceedings of LREC*, volume 6, 2006.

[29] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. Web-scale Information Extraction in KnowItAll. In *Proceedings of WWW-2004*, 2004. `http://www.cs.washington.edu/research/knowitall/papers/www-paper.pdf`.

[30] O. Etzioni, M. Cafarella, D. Downey, A.M. Popescu, T. Shaked, S. Soderland, D.S. Weld, and A. Yates. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence*, 165(1):91–134, 2005.

[31] Christiane Fellbaum, editor. *WordNet - An Electronic Lexical Database*. MIT Press, 1998.

[32] D. Ferrucci and A. Lally. UIMA: An Architectural Approach to Unstructured Information Processing in the Corporate Research Environment. *Natural Language Engineering*, 10(3-4):327–348, 2004.

[33] K.T. 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.

[34] D. Freitag. Information extraction from html: Application of a general learning approach. *Proceedings of the Fifteenth Conference on Artificial Intelligence AAAI-98*, pages 517–523, 1998.

[35] Dayne Freitag and Nicholas Kushmerick. Boosted Wrapper Induction. In *Seventeenth National Conference on Artificial Intelligence (AAAI-2000): Twelfth Innovative Applications of Artificial Intelligence Conference (IAAI-2000).*, pages 577–583, 2000.

[36] G. Grefenstette. *Explorations in Automatic Thesaurus Discovery*. Kluwer Academic Publishers, 1994.

[37] Hepple M Harkema H, Gaizauskas R. A large scale terminology resource for biomedical text processing. In *Proceedings of the HLT-NAACL Workshop: BioLINK 2004, Linking Biological Literature, Ontologies and Databases*, 2004.

[38] Z.S. Harris. *Mathematical Structures of Language*. Wiley (Interscience), New York, 1968.

[39] M. A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Conference on Computa-*

*tional Linguistics (COLING'92)*, Nantes, France, 1992. Association for Computational Linguistics.

[40] M. Hepple. Independence and Commitment: Assumptions for Rapid Training and Execution of Rule-based Part-of-Speech Taggers. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, Hong Kong, October 2000.

[41] L. Hirschman, R. Grishman, and N. Sager. Grammatically based automatic word class formation. *Information Processing and Retrieval*, 11:39–57, 1975.

[42] H. Isozaki and H. Kazawa. Efficient Support Vector Classifiers for Named Entity Recognition. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING'02)*, pages 390–396, Taipei, Taiwan, 2002.

[43] D. Klein and C. Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Meeting of the Association for Computational Linguistics*, 2003.

[44] Y. Li, K. Bontcheva, and H. Cunningham. Using Uneven Margins SVM and Perceptron for Information Extraction. In *Proceedings of Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, 2005.

[45] Yaoyong Li, Kalina Bontcheva, and Hamish Cunningham. Adapting SVM for Data Sparseness and Imbalance: A Case Study on Information Extraction. *Natural Language Engineering*, 15(2):241–271, 2009.

[46] E. Loper and S. Bird. NLTK: The Natural Language Toolkit. In *ACL Workshop on Effective Tools and Methodologies in Teaching NLP*, 2002.

[47] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1994.

[48] J. Mayfield, P. McNamee, and C. Piatko. Named Entity Recognition Using Hundreds of Thousands of Features. In *Proceedings of CoNLL-2003*, pages 184–187. Edmonton, Canada, 2003.

[49] D. Maynard, H. Cunningham, K. Bontcheva, and M. Dimitrov. Adapting a robust multi-genre NE system for automatic content extraction. In *Proceedings of the* $10^{th}$ *International Conference on Artificial Intelligence: Methodology, Systems, Applications (AIMSA'02)*, Varna, Bulgaria, Sep 2002.

[50] D. Maynard, A. Funk, and W. Peters. SPRAT: a tool for automatic semantic pattern-based ontology population. In *International Conference for Digital Libraries and the Semantic Web*, Trento, Italy, September 2009.

[51] D. Maynard, V. Tablan, K. Bontcheva, H. Cunningham, and Y.Wilks. Multi-source entity recognition – an information extraction system for diverse text types. Research Memorandum CS–03–02, Department of Computer Science, University of Sheffield, April 2003.

[52] D. G. Maynard. *Term Recognition Using Combined Knowledge Sources*. PhD thesis, Manchester Metropolitan University, UK, 2000.

[53] D.G. Maynard and S. Ananiadou. Identifying terms by their family and friends. In *Proc. of 18th International Conference on Computational Linguistics (COLING)*, Saarbrücken, Germany, 2000.

[54] Diana Maynard, Yaoyong Li, and Wim Peters. NLP Techniques for Term Extraction and Ontology Population. In P. Buitelaar and P. Cimiano, editors, *Bridging the Gap between Text and Knowledge - Selected Contributions to Ontology Learning and Population from Text*. IOS Press, 2008.

[55] A. McCallum, D. Freitag, and F. Pereira. Maximum entropy markov models for information extraction and segmentation. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 591–598. Citeseer, 2000.

[56] J. McNaught, W. Black, F. Rinaldi, E. Bertino, A. Brasher, D. Deavin, B. Catania, D. Silvestri, B. Armani, A. Persidis, G. Semerano, F. Esposito, V. Candela, G.P. Zarri, and L. Gilardoni. Integrated document and knowledge management for the knowledge-based enterprise. In *Proceedings of the 3rd International Conference on the practical application of Knowledge Management*. The paractical application company, 2000.

[57] Martha Palmer, Daniel Gildea, and Paul Kingsbury. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106, 2005.

[58] P. Pantel and M. Pennacchiotti. Automatically harvesting and ontologizing semantic relations. In *Proceeding of the 2008 conference on Ontology Learning and Population: Bridging the Gap between Text and Knowledge*, pages 171–195. IOS Press, 2008.

[59] M.F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[60] L. Ramshaw and M. Marcus. Text Chunking Using Transformation-Based Learning. In *Proceedings of the Third ACL Workshop on Very Large Corpora*, 1995.

[61] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

[62] Karin Kipper Schuler. *VerbNet: A broad-coverage, comprehensive verb lexicon*. PhD thesis, University of Pennsylvania, 2005.

[63] A. Schutz and P. Buitelaar. Relext: A tool for relation extraction from text in ontology extension. *The Semantic Web–ISWC 2005*, pages 593–606, 2005.

[64] S. Soderland. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34(1):233–272, 1999.

[65] Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Techn@articleAsur2010, annote = Tests the informativeness of twitter when predicting performance of new film releases at the box office. Both simple tweet counts and a classifier are used. The classifier is trained using mechanical turk. A performance of 98% was reported for the classifier. They found that classifying sentiment only lead to a slight improvement over the tweet count., author = Asur, Sitaram and Huberman, Bernardo A, journal = CoRR, keywords = Sentiment, mendeley-tags = Sentiment, title = Predicting the Future with Social Media, url = http://arxiv.org/abs/1003.5699, volume = abs/1003.5, year = 2010 ology*, NAACL '03, pages 173–180, 2003.

# Part II

# Logical Learning

# Concept Learning

Jens LEHMANN [a] and Nicola FANIZZI [b] and Lorenz BÜHMANN [a]
and Claudia D'AMATO [b]
[a] *Univ. Leipzig, Germany*
[b] *Univ. Bari, Italy*

**Abstract.** One of the bottlenecks of the ontology construction process is the amount of work required with various figures playing a role in it: domain experts contribute their knowledge that has to be formalized by knowledge engineers so that it can be mechanized. As the gap between these roles likely makes the process slow and burdensome, this problem may be tackled by resorting to machine learning techniques. By adopting algorithms from inductive logic programming, the effort of the domain expert can be reduced, i.e. he has to label individual resources as instances of the target concept. From those labels, axioms can be induced, which can then be confirmed by the knowledge engineer. In this chapter, we survey existing methods in this area and illustrate three different algorithms in more detail. Some basics like refinement operators, decision trees and information gain are described. Finally, we briefly present implementations of those algorithms.

**Keywords.** Refinement operators, Decision Trees, Information Gain

## 1. Introduction to Concept Learning

One of the bottlenecks of the ontology construction process is represented by the amount of work required with various figures playing a role in it: domain experts contribute their knowledge that is formalized by knowledge engineers so that it can be mechanized. As the gap between these roles makes the process slow and burdensome, this problem may be tackled by resorting to *machine learning* (cf. Lawrynowicz and Tresp [23] in this volume) techniques. Solutions can be based on *relational learning* [36] which requires a limited effort from domain experts (labeling individual resources as instances of the target concepts) and leads to the construction of concepts adopting even very expressive languages [32]. If the concept learning problem is tackled as a search through a space of candidate descriptions in the reference representation guided by exemplars of the target concepts, the same algorithms can be adapted to solve also *ontology evolution* problems. Indeed, while normally the semantics of change operations has been considered from the logical and deductive point of view of automated reasoning, a relevant part of information lying in the data that populates ontological knowledge bases is generally overlooked or plays a secondary role. Early work on the application of machine learning to Description Logics (DLs) [3] essentially focused on demonstrating the PAC-learnability for various terminological languages derived from CLASSIC. In particular, Cohen and Hirsh investigate the CORECLASSIC DL proving that it is not PAC-learnable [9] as well as demonstrating the PAC-learnability of its sub-languages, such

as C-CLASSIC [10], through the bottom-up LCSLEARN algorithm. These approaches tend to cast supervised concept learning to a structural generalizing operator working on equivalent graph representations of the concept descriptions. It is also worth mentioning unsupervised learning methodologies for DL concept descriptions, whose prototypical example is KLUSTER [22], a polynomial-time algorithm for the induction of BACK terminologies, which exploits the tractability of the standard inferences in this DL language [3]. More recently, approaches have been proposed that adopt the idea of *generalization as search* [33] performed through suitable operators that are specifically designed for DL languages [4,14,11,15,19,30,32] on the grounds of the previous experience in the context of ILP. There is a body of research around the analysis of such operators [30,24] along with applications to various problems [31,20] and studies on the practical scalability of algorithms using them [17,18,27]. Supervised (resp., unsupervised) learning systems, such as YINYANG [19] and DL-Learner [25], have been implemented and adoptions implemented for the ontology learning use case [7,27,8,26].

Learning alternative models such as logical decision trees offers another option for concept induction. The induction of decision trees is among the most well-known machine learning techniques [34], also in its more recent extensions that are able to work with more expressive logical representations in clausal form [5]. A new version of the FOIL algorithm [35] has been implemented, resulting in the DL-FOIL system [12]. The general framework has been extended to cope with logical representations designed for formal Web ontologies [13]. The induction of *terminological decision trees* [13], i.e. logical decision trees test-nodes represented with DL concept descriptions, adopts a classical top-down *divide-and-conquer* strategy [6] which differs from previous DL concept learning methods based on sequential covering or heuristic search, with the use of refinement operators for DL concept descriptions [19,12,32]. The main components of this new systems are 1) a set of refinement operators borrowed from other similar systems [19,31]; 2) a specific *information-gain function* which must take into account the open-world assumption, namely, many instances may be available which cannot be ascribed to the target concept nor to its negation. This requires a different setting, similar to learning with unknown class attributes [16], requiring a special treatment of the unlabeled individuals. Once a terminological tree is induced, similarly to the logical decision trees, a definition of the target concepts can be drawn exploiting the nodes in the tree structure. The algorithm has also a useful side-effect: the suggestion of new intermediate concepts which may have no definition in the current ontology.

## 2. Learning as Search in Description Logics

### *2.1. Learning Problem*

In this section, the learning problem in the DL setting is formally defined.

**Definition 2.1 (learning problem)** *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a DL knowledge base.*

**Given**

- *a (new) target concept name $C$*

- *a set of positive and negative examples*[1] *for* $C$*:*
  - $\mathit{Ind}_C^+(\mathcal{A}) = \{a \in \mathit{Ind}(\mathcal{A}) \mid C(a) \in \mathcal{A}\} \subseteq R_{\mathcal{K}}(C)$ *instance retrieval of* $C$
  - $\mathit{Ind}_C^-(\mathcal{A}) = \{b \in \mathit{Ind}(\mathcal{A}) \mid \neg C(b) \in \mathcal{A}\} \subseteq R_{\mathcal{K}}(\neg C)$

**Find** *a concept definition* $C \equiv D$ *such that*

- $\mathcal{K} \models D(a) \quad \forall a \in \mathit{Ind}_D^+(\mathcal{A})$ *and*
- $\mathcal{K} \models \neg D(b) \quad \forall b \in \mathit{Ind}_C^-(\mathcal{A}) \qquad$ *(resp.* $\mathcal{K} \not\models C(b) \quad \forall b \in \mathit{Ind}_C^-(\mathcal{A})$*)*

We prefer the first form for a correct definition w.r.t. negative examples ($\mathcal{K} \models \neg D(b)$) because that seems more coherent with the explicit indication given by the expert. Other settings assume ($\mathcal{K} \not\models C(b)$) which implicitly makes it a binary learning problem.

The definition given above can be interpreted as a generic supervised concept learning problem. In case a previous definition $D'$ for $C$ is already available in $\mathcal{K}$ and $\exists a \in \mathsf{Ind}_C^+(\mathcal{A})\ s.t.\ \mathcal{K} \not\models D'(a)$ or $\exists b \in \mathsf{Ind}_C^-(\mathcal{A})\ s.t.\ \mathcal{K} \not\models \neg D'(b)$ then the problem can be cast as a *refinement problem* which would amount to searching for a solution $D$ starting from the approximation $D'$.

### *2.2. Refinement Operators*

The solution of the learning problem stated above can be cast as a search for a correct concept definition in an ordered space $(\Sigma, \preceq)$. In such a setting, one can define suitable operators to traverse the search space. Refinement operators can be formally defined as:

**Definition 2.2 (refinement operator)** *Given a quasi-ordered*[2] *search space* $(\Sigma, \preceq)$

- *a* downward refinement operator *is a mapping* $\rho : \Sigma \to 2^\Sigma$ *such that*

$$\forall \alpha \in \Sigma \quad \rho(\alpha) \subseteq \{\beta \in \Sigma \mid \beta \preceq \alpha\}$$

- *an* upward refinement operator *is a mapping* $\delta : \Sigma \to 2^\Sigma$ *such that*

$$\forall \alpha \in \Sigma \quad \delta(\alpha) \subseteq \{\beta \in \Sigma \mid \alpha \preceq \beta\}$$

**Definition 2.3 (properties of DL refinement operators)** *A refinement operator* $\rho$ *is*

- (locally) finite *iff* $\rho(C)$ *is finite for all concepts* $C$*.*
- redundant *iff there exists a refinement chain from a concept* $C$ *to a concept* $D$*, which does not go through some concept* $E$ *and a refinement chain from* $C$ *to a concept equal to* $D$*, which does go through* $E$*.*
- proper *iff for all concepts* $C$ *and* $D$*,* $D \in \rho(C)$ *implies* $C \not\equiv D$*.*

*A downward refinement operator* $\rho$ *is called*

- complete *iff for all concepts* $C, D$ *with* $C \sqsubset D$ *we can reach a concept* $E$ *with* $E \equiv C$ *from* $D$ *by* $\rho$*.*
- weakly complete *iff for all concepts* $C \sqsubset \top$ *we can reach a concept* $E$ *with* $E \equiv C$ *from* $\top$ *by* $\rho$*.*

*The corresponding notions for upward refinement operators are defined dually.*

[1] Note that $\mathsf{Ind}_C^+(\mathcal{A}) \cup \mathsf{Ind}_C^-(\mathcal{A}) \subseteq \mathsf{Ind}(\mathcal{A})$ where $\mathsf{Ind}(\mathcal{A})$ is the set of all individuals occurring in $\mathcal{A}$.

[2] A quasi-ordering is a reflexive and transitive relation.

In the following, we will consider a space of concept definitions ordered by the subsumption relationship $\sqsubseteq$ which induces a quasi-order on the space of all the possible concept descriptions [4,11]. In particular, given the space of concept definitions in the reference DL language, say $(\mathcal{L}, \sqsubseteq)$, ordered by subsumption, there is an infinite number of generalizations and specializations. Usually one tries to devise operators that can traverse efficiently throughout the space in pursuit of one of the correct definitions (w.r.t. the examples that have been provided).

### *2.2.1. Refinement Operator for DL-FOIL*

In the definition of refinement operators, the notion of *normal form* for $\mathcal{ALC}$ concept descriptions is given. Preliminarily, a concept is in *negation normal form* iff negation only occurs in front of concept names. Now, some notation is needed to name the different parts of an $\mathcal{ALC}$ description: $\mathsf{prim}(C)$ is the set of all the concepts at the top-level conjunction of $C$; if there exists a universal restriction $\forall R.D$ on the top-level of $C$ then $\mathsf{val}_R(C) = \{D\}$ (a singleton description because many such restrictions can be collapsed into a single one with a conjunctive filler concept) otherwise $\mathsf{val}_R(C) = \{\top\}$. Finally, $\mathsf{ex}_R(C)$ is the set of the concept descriptions $E$ appearing in existential restrictions $\exists R.E$ at the top-level conjunction of $C$.

**Definition 2.4 ($\mathcal{ALC}$ normal form)** *A concept description $D$ is in $\mathcal{ALC}$* normal form *iff $D$ is $\bot$ or $\top$ or if $D = D_1 \sqcup \cdots \sqcup D_n$ with*

$$D_i = \bigsqcap_{A \in \mathsf{prim}(D_i)} A \sqcap \bigsqcap_{R \in N_R} \left( \bigsqcap_{V \in \mathsf{val}_R(D_i)} \forall R.V \sqcap \bigsqcap_{E \in \mathsf{ex}_R(D_i)} \exists R.E \right)$$

*where, for all $i = 1, \ldots, n$, $D_i \not\equiv \bot$ and for any $R$, every sub-description in $\mathsf{ex}_R(D_i)$ and $\mathsf{val}_R(D_i)$ is in normal form.*

We will consider two theoretical refinement operators [19] that, given a starting incorrect definition (too weak or too strong) for the target concept in the search space, can compute one (or some) of its generalizations / specializations. Both are defined (w.l.o.g.) for $\mathcal{ALC}$ descriptions in normal form.

**Definition 2.5 (downward operator $\rho$)** *Let $\rho = (\rho_\sqcup, \rho_\sqcap)$ be a downward refinement operator, where:*
*[$\rho_\sqcup$] given a description in $\mathcal{ALC}$ normal form $D = D_1 \sqcup \cdots \sqcup D_n$:*

- *$D' \in \rho_\sqcup(D)$ if $D' = \bigsqcup_{\substack{1 \le i \le n \\ i \neq j}} D_i$ for some $j \in \{1, \ldots, n\}$*
- *$D' \in \rho_\sqcup(D)$ if $D' = D'_i \sqcup \bigsqcup_{\substack{1 \le i \le n \\ i \neq j}} D_i$ for some $j \in \{1, \ldots, n\}$ and $D'_j \in \rho_\sqcap(D_j)$*

*[$\rho_\sqcap$] given a conjunctive description $C = C_1 \sqcap \cdots \sqcap C_m$:*

- *$C' \in \rho_\sqcap(C)$ if $C' = C \sqcap C_{m+1}$ for some $C_{m+1}$ such that $\bot \sqsubset C' \sqsubseteq C$*
- *$C' \in \rho_\sqcap(C)$ if $C' = \bigsqcap_{\substack{1 \le i \le m \\ i \neq k}} C_i \sqcap C'_k$ for some $k \in \{1, \ldots, m\}$, where:*
  - *$C'_k \sqsubseteq C_k$ if $C_k \in \mathsf{prim}(C)$ or*
  - *$C'_k = \exists R.D'$ if $C_k = \exists R.D$ and $D' \in \rho_\sqcup(D)$ or*

$*$ $C'_k = \forall R.D'$ *if* $C_k = \forall R.D$ *and* $D' \in \rho_\sqcup(D)$

Note that a difference operator for concepts is used to single out the subconcepts to be refined. Further possibilities may be explored using the operator defined $C - D = C \sqcup \neg D$ [38].

The operator for disjunctive concepts $\rho_\sqcup$ simply drops one top-level disjunct or replaces it with a downward refinement obtained with $\rho_\sqcap$. $\rho_\sqcap$ adds new conjuncts or replaces one with a refinement obtained by specializing a primitive concept or the subconcepts in the scope of a universal or existential restriction (again through $\rho_\sqcup$). Note that successive applications of the operator may require intermediate normalization steps.

**Definition 2.6 (upward operator $\delta$)** *Let $\delta = (\delta_\sqcup, \delta_\sqcap)$ be a downward refinement operator, where:*
*[$\delta_\sqcup$] given a description in $\mathcal{ALC}$ normal form $D = D_1 \sqcup \cdots \sqcup D_n$:*

- $D' \in \delta_\sqcup(D)$ *if* $D' = D \sqcup D_{n+1}$ *for some* $D_{n+1}$ *such that* $D_{n+1} \not\sqsubseteq D$
- $D' \in \delta_\sqcup(D)$ *if* $D' = D'_j \sqcup \bigsqcup_{\substack{1 \leq i \leq n \\ i \neq j}} D_i$ *for some* $j \in \{1, \ldots .n\}$, $D'_j \in \delta_\sqcap(D_j)$

*[$\delta_\sqcap$] given a conjunctive description $C = C_1 \sqcap \cdots \sqcap C_m$:*

- $C' \in \delta_\sqcap(C)$ *if* $C' = \bigsqcap_{\substack{1 \leq i \leq m \\ i \neq k}} C_i$ *for some* $k \in \{1, \ldots .m\}$
- $C' \in \delta_\sqcap(C)$ *if* $C' = \bigsqcap_{\substack{1 \leq i \leq m \\ i \neq k}} C_i \sqcap C'_k$ *for some* $k \in \{1, \ldots, m\}$, *where:*
  - $C'_k \sqsupseteq C_k$ *if* $C_k \in \mathsf{prim}(C)$ *or*
  - $C'_k = \exists R.D'$ *if* $C_k = \exists R.D$ *and* $D' \in \delta_\sqcup(D)$ *or*
  - $C'_k = \forall R.D'$ *if* $C_k = \forall R.D$ *and* $D' \in \delta_\sqcup(D)$

$\delta_\sqcup$ and $\delta_\sqcap$ simply perform dual operations w.r.t. $\rho_\sqcup$ and $\rho_\sqcap$, respectively. Some examples of their application can be found in [19].

These operators follow the definition of the $\mathcal{ALC}$ normal form. Hence they cannot be complete for more expressive DLs (see [30] for an analysis of refinement operators in DLs). However, instead of such operators that likely lead to overfit the data (e.g. a generalizing operator based on the computation of the *Least Common Subsumer* (LCS) [10] would amount to a simple union of the input descriptions in $\mathcal{ALC}$) it may be preferable to search the space (incompletely) using the non-$\mathcal{ALC}$ restrictions as atomic features (concepts). Moreover, other operators have been designed to exploit also the knowledge conveyed by the positive and negative examples in order to prune the possible candidate refinements yielded by a single generalization / specialization step and to better direct the search for suitable problem solutions [19]. Even more so, instead of using the examples in a mere *generate-and-test* strategy based on these operators, they could be exploited more directly[3], in order to influence the choices made during the refinement process.

### *2.2.2. A refinement operator for CELOE*

Designing a refinement operator $\rho$ needs to make decisions on which properties are most useful in practice regarding the underlying learning algorithm. Considering the properties *completeness*, *weak completeness*, *properness*, *finiteness*, and *non-redundancy* an

[3] E.g. using the most specific concepts [3] as their representatives to the concept level. But their exact computation is feasible only for very simple DLs.

extensive analysis in [30] has shown that the most feasible property combination for our setting is $\{$*weakly complete*$, complete, proper\}$, which we will justify briefly. Only for less expressive description logics like $\mathcal{EL}$, ideal, i.e. complete, proper and final, operators exist [29]. (Weak) Completeness is considered a very important property, since an incomplete operator may fail to converge at all and thus may not return a solution even if one exists. Reasonable, weakly complete operators are often complete. Consider, for example, the situation where a weakly complete operator $\rho$ allows to refine a concept $C$ to $C \sqcap D$ with some $D \in \rho(\top)$. Then it turns out that this operator is already complete.

Concerning finiteness, having an infinite operator is less critical from a practical perspective since this issue can be handled algorithmically. So it is preferable not imposing finiteness, which allows to develop a proper operator. As for non-redundancy, this appears to be very difficult to achieve for more complex operators. Consider, for example, the concept $A_1 \sqcap A_2$ which can be reached from $\top$ via the chain $\top \rightsquigarrow A_1 \rightsquigarrow A_1 \sqcap A_2$, For non-redundancy, the operator would need to make sure that this concept cannot be reached via the chain $\top \rightsquigarrow A_2 \rightsquigarrow A_2 \sqcap A_1$. While there are methods to handle this in such simple cases via normal forms, it becomes more complex for arbitrarily deeply nested structures, where even applying the same replacement leads to redundancy. In the following example, $A_1$ is replaced by $A_1 \sqcap A_2$ twice in different order in each chain:

$$\begin{aligned}\top \rightsquigarrow & \forall r_1.A_1 \sqcup \forall r_2.A_1 \rightsquigarrow \forall r_1.A_1 \sqcup \forall r_2.(A_1 \sqcap A_2)\\ & \rightsquigarrow \forall r_1.(A_1 \sqcap A_2) \sqcup \forall r_2.(A_1 \sqcap A_2)\\ \top \rightsquigarrow & \forall r_1.A_1 \sqcup \forall r_2.A_1 \rightsquigarrow \forall r_1.(A_1 \sqcap A_2) \sqcup \forall r_2.A_1\\ & \rightsquigarrow \forall r_1.(A_1 \sqcap A_2) \sqcup \forall r_2.(A_1 \sqcap A_2)\end{aligned}$$

To avoid this, an operator would need to regulate when $A_1$ can be replaced by $A_1 \sqcap A_2$, which appears not to be achievable by syntactic replacement rules. Alternatively, a computationally inexpensive redundancy check can be used, which seems to be sufficiently useful in practice.

We now define the refinement operator $\rho$: For each $A \in N_C$, we define ($sh$ stands for subsumption hierarchy):

$$sh_{\downarrow}(A) = \{A' \in N_C \mid A' \sqsubset A, \text{ there is no } A'' \in N_C \text{ with } A' \sqsubset_{\mathcal{T}} A'' \sqsubset_{\mathcal{T}} A\}$$

$sh_{\downarrow}(\top)$ is defined analogously for $\top$ instead of $A$. $sh_{\uparrow}(A)$ is defined analogously for going upward in the subsumption hierarchy. We do the same for roles, i.e. :

$$sh_{\downarrow}(r) = \{r' \mid r' \in N_R, r' \sqsubset r, \text{ there is no } r'' \in N_R \text{ with } r' \sqsubset_{\mathcal{T}} r'' \sqsubset_{\mathcal{T}} r\}$$

*domain*$(r)$ denotes the domain of a role $r$ and *range*$(r)$ the range of a role $r$. A range axiom links a role to a concept. It asserts that the role fillers must be instances of a given concept. Domain axioms restrict the first argument of role assertions to a concept. We define:

$$\begin{aligned}ad(r) = \quad & \text{an } A \text{ with } A \in \{\top\} \cup N_C \text{ and } domain(r) \sqsubseteq A\\ & \text{and there does not exist an } A' \text{ with } domain(r) \sqsubseteq A' \sqsubset A\end{aligned}$$

$ar(r)$ is defined analogously using *range* instead of *domain*. *ad* stands for atomic domain and *ar* stands for atomic range. We assign exactly one atomic concept as domain/range of a role. Since using atomic concepts as domain and range is very common, *domain* and *ad* as well as *range* and *ar* will usually coincide. The set $app_B$ of applicable properties with respect to an atomic concept $B$ is defined as:

$$app_B = \{r | r \in N_R, ad(r) = A, A \sqcap B \not\equiv \bot\}$$

To give an example, for the concept `Person`, we have that the role `hasChild` with $ad(\texttt{hasChild}) = \texttt{Person}$ is applicable, but the role `hasAtom` with $ad(\texttt{hasAtom}) = \texttt{ChemicalCompound}$ is not applicable (assuming `Person` and `ChemicalCompound` are disjoint). We will use this to restrict the search space by ruling out unsatisfiable concepts. The index $B$ describes the context in which the operator is applied, e.g. $\top \rightsquigarrow \texttt{Person}$ is a refinement step of $\rho$. However, $\exists \texttt{hasAtom}.\top \rightsquigarrow \exists \texttt{hasAtom}.\texttt{Person}$ is not a refinement step of $\rho$ assuming $ar(\texttt{hasAtom})$ and `Person` are disjoint. The set of most general applicable roles $mgr_B$ with respect to a concept $B$ is defined as:

$$mgr_B = \{r \mid r \in app_B, \text{ there is no } r' \text{ with } r \sqsubset r', r' \in app_B\}$$

$M_B$ with $B \in \{\top\} \cup N_C$ is defined as the union of the following sets:

- $\{A \mid A \in N_C, A \sqcap B \not\equiv \bot, A \sqcap B \not\equiv B, \text{ there is no } A' \in N_C \text{ with } A \sqsubset A'\}$
- $\{\neg A \mid A \in N_C, \neg A \sqcap B \not\equiv \bot, \neg A \sqcap B \not\equiv B, \text{ there is no } A' \in N_C \text{ with } A' \sqsubset A\}$
- $\{\exists r.\top \mid r \in mgr_B\}$
- $\{\forall r.\top \mid r \in mgr_B\}$

The operator $\rho$ is defined in Figure 1. Note that $\rho$ delegates to an operator $\rho_B$ with $B = \top$ initially. $B$ is set to the atomic range of roles contained in the input concept when the operator recursively traverses the structure of the concept. The index $B$ in the operator (and the set $M$ above) is used to rule out concepts which are disjoint with $B$.

**Example 2.1 ($\rho$ refinements)** *Since the operator is not easy to understand at first glance, we provide some examples. Let the following knowledge base be given:*

$$\mathcal{K} = \{\texttt{Man} \sqsubset \texttt{Person}; \texttt{Woman} \sqsubset \texttt{Person}; \texttt{SUV} \sqsubset \texttt{Car}; \texttt{Limo} \sqsubset \texttt{Car};$$
$$\texttt{Person} \sqcap \texttt{Car} \equiv \bot; \textit{domain}(\texttt{hasOwner}) = \texttt{Car}; \textit{range}(\texttt{hasOwner}) = \texttt{Person}\}$$

*Then the following refinements of $\top$ exist:*

$$\rho(\top) = \{\texttt{Car}, \texttt{Person}, \neg\texttt{Limo}, \neg\texttt{SUV}, \neg\texttt{Woman}, \neg\texttt{Man},$$
$$\exists\texttt{hasOwner}.\top, \forall\texttt{hasOwner}.\top, \texttt{Car} \sqcup \texttt{Car}, \texttt{Car} \sqcup \texttt{Person}, \dots\}$$

*This illustrates how the set $M_\top$ is constructed. Note that refinements like $\texttt{Car} \sqcup \texttt{Car}$ are incorporated in order to reach e.g. $\texttt{SUV} \sqcup \texttt{Limo}$ later in a possible refinement chain. The concept $\texttt{Car} \sqcap \exists\texttt{hasOwner}.\texttt{Person}$ has the following refinements:*

$$\rho(C) = \begin{cases} \{\bot\} \cup \rho_\top(C) & \text{if } C = \top \\ \rho_\top(C) & \text{otherwise} \end{cases}$$

$$\rho_B(C) = \begin{cases} \emptyset & \text{if } C = \bot \\ \{C_1 \sqcup \dots \sqcup C_n \mid C_i \in M_B \ (1 \leq i \leq n)\} & \text{if } C = \top \\ \{A' \mid A' \in sh_\downarrow(A)\} & \text{if } C = A \ (A \in N_C) \\ \quad \cup \{A \sqcap D \mid D \in \rho_B(\top)\} & \\ \{\neg A' \mid A' \in sh_\uparrow(A)\} & \text{if } C = \neg A \ (A \in N_C) \\ \quad \cup \{\neg A \sqcap D \mid D \in \rho_B(\top)\} & \\ \{\exists r.E \mid A = ar(r), E \in \rho_A(D)\} & \text{if } C = \exists r.D \\ \quad \cup \{\exists r.D \sqcap E \mid E \in \rho_B(\top)\} & \\ \quad \cup \{\exists s.D \mid s \in sh_\downarrow(r)\} & \\ \{\forall r.E \mid A = ar(r), E \in \rho_A(D)\} & \text{if } C = \forall r.D \\ \quad \cup \{\forall r.D \sqcap E \mid E \in \rho_B(\top)\} & \\ \quad \cup \{\forall r.\bot \mid & \\ \qquad D = A \in N_C \text{ and } sh_\downarrow(A) = \emptyset\} & \\ \quad \cup \{\forall s.D \mid s \in sh_\downarrow(r)\} & \\ \{C_1 \sqcap \dots \sqcap C_{i-1} \sqcap D \sqcap C_{i+1} \sqcap \dots \sqcap C_n \mid & \text{if } C = C_1 \sqcap \dots \sqcap C_n \\ \qquad D \in \rho_B(C_i), 1 \leq i \leq n\} & (n \geq 2) \\ \{C_1 \sqcup \dots \sqcup C_{i-1} \sqcup D \sqcup C_{i+1} \sqcup \dots \sqcup C_n \mid & \text{if } C = C_1 \sqcup \dots \sqcup C_n \\ \qquad D \in \rho_B(C_i), 1 \leq i \leq n\} & (n \geq 2) \\ \quad \cup \{(C_1 \sqcup \dots \sqcup C_n) \sqcap D \mid & \\ \qquad D \in \rho_B(\top)\} & \end{cases}$$

**Figure 1.** Definition of the refinement operator $\rho$.

$$\begin{aligned} \rho(\texttt{Car} \sqcap \exists \texttt{hasOwner.Person}) = \{&\texttt{Car} \sqcap \exists \texttt{hasOwner.Man}, \\ &\texttt{Car} \sqcap \exists \texttt{hasOwner.Woman}, \\ &\texttt{SUV} \sqcap \exists \texttt{hasOwner.Person}, \\ &\texttt{Limo} \sqcap \exists \texttt{hasOwner.Person}, \dots\} \end{aligned}$$

*Note the traversal of the subsumption hierarchy, e.g.* `Car` *is replaced by* `SUV`*.*

**Proposition 2.1 (Downward Refinement of $\rho$)** *$\rho$ is an $\mathcal{ALC}$ downward refinement operator.*

A distinguishing feature of $\rho$ compared to other DL refinement operators [4,11], is that it makes use of the subsumption and role hierarchy, e.g. for concepts $A_2 \sqsubset A_1$, we reach $A_2$ via $\top \rightsquigarrow A_1 \rightsquigarrow A_2$. This way, we can stop the search if $A_1$ is already too weak and, thus, make better use of TBox knowledge. The operator also uses domain and range of roles to reduce the search space. This is similar to mode declarations in Aleph, Progol, and other ILP programs. However, in DL knowledge bases and OWL ontologies, domain and range are usually explicitly given, so there is no need to define them manually. Overall, the operator supports more structures than those in [4,11] and

tries to intelligently incorporate background knowledge. In [32] further extensions of the operator are described, which increase its expressivity such that it can handle most OWL class expressions. Note that $\rho$ is infinite. The reason is that the set $M_B$ is infinite and we put no bound on the number of elements in the disjunctions, which are refinements of the top concept. Furthermore, the operator requires reasoner requests for calculating $M_B$. However, the number of requests is fixed, so – assuming the results of those requests are cached – the reasoner is only needed in an initial phase, i.e. during the first calls to the refinement operator. This means that, apart from this initial phase, the refinement operator performs only syntactic rewriting rules.

## 3. CELOE

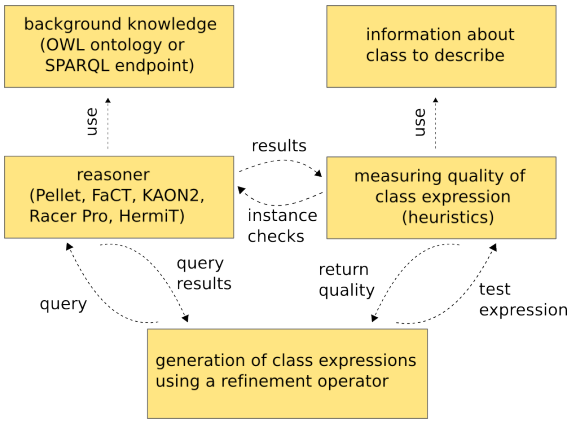



**Figure 2.** Outline of the general learning approach in CELOE: One part of the algorithm is the generation of promising class expressions taking the available background knowledge into account. Another part is a heuristic measure of how close an expression is to being a solution of the learning problem. Figure adapted from [17,18].

Figure 2 gives an overview of our algorithm *CELOE* (standing for "class expression learning for ontology engineering"), which follows the common "generate and test" approach in ILP. Learning is seen as a search process and several class expressions are generated and tested against a background knowledge base. Each of those class expressions is evaluated using a heuristic [27]. A challenging part of a learning algorithm is to decide which expressions to test. Such a decision should take the computed heuristic values and the structure of the background knowledge into account. For CELOE, we use the approach described in [31,32] as base, where this problem has been analysed, implemented, and evaluated. It is based on the *refinement operator* introduced in Sect. 2.2.2.

The approach we used is a top-down algorithm based on refinement operators as illustrated in Figure 3. This means that the first class expression, which will be tested is the most general expression ($\top$), which is then mapped to a set of more specific expressions by means of a downward refinement operator. The refinement operator can be applied to the obtained expressions again, thereby spanning a *search tree*. The search tree can be pruned when an expression does not cover sufficiently many instances of the class $A$ we want to describe. One example for a path in a search tree spanned up by a downward refinement operator is the following ($\rightsquigarrow$ denotes a refinement step):

$$\top \rightsquigarrow \texttt{Person} \rightsquigarrow \texttt{Person} \sqcap \texttt{takesPartinIn}.\top$$
$$\rightsquigarrow \texttt{Person} \sqcap \texttt{takesPartIn}.\texttt{Meeting}$$

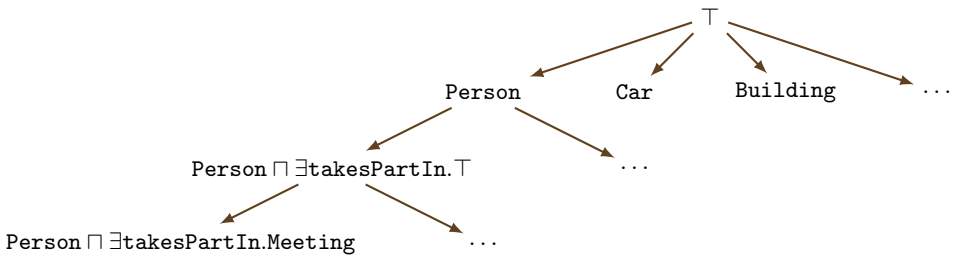



**Figure 3.** Illustration of a search tree in a top down refinement approach.

The heart of such a learning strategy is to define a suitable refinement operator and an appropriate search heuristics for deciding which nodes in the search tree should be expanded. The refinement operator in the considered algorithm is defined in [32]. It is based on [31] which in turn is build on theoretical foundations in [30]. It has been shown to be the best achievable operator with respect to a set of properties (not further described here), which are used to assess the performance of refinement operators. The learning algorithm supports conjunction, disjunction, negation, existential and universal quantifiers, cardinality restrictions, hasValue restrictions as well as boolean and double datatypes.

While the major change compared to other supervised learning algorithms for OWL is the previously described heuristic, there are also further modifications. The goal of those changes is to adapt the learning algorithm to the ontology engineering scenario: For example, the algorithm was modified to introduce a strong bias towards short class expressions. This means that the algorithm is less likely to produce long class expressions, but is almost guaranteed to find any suitable short expression (see [8] for an alternative approach to achieve this). The rationale behind this change is that knowledge engineers can understand short expressions better than more complex ones and it is essential not to miss those. We also introduced improvements to enhance the readability of suggestions: Each suggestion is reduced, i.e. there is a guarantee that they are as succinct as possible. For example, $\exists \texttt{hasLeader}.\top \sqcap \texttt{Capital}$ is reduced to `Capital` if the background knowledge allows to infer that a capital is a city and each city has a leader. This reduction algorithm uses the complete and sound *Pellet* reasoner, i.e. it can take any possible complex relationships into account by performing a series of subsumption checks between class expressions. A caching mechanism is used to store the results of those checks, which allows to perform the reduction very efficiently after a warm-up phase. We also make sure that "redundant" suggestions are omitted. If one suggestion is longer and subsumed by another suggestion and both have the same characteristics, i.e. classify the relevant individuals equally, the more specific suggestion is filtered. This avoids expressions containing irrelevant subexpressions and ensures that the suggestions are sufficiently diverse.

## 4. DL-FOIL

In this section, DL-FOIL [12] algorithm is presented. The main aim of this work was conceiving a learning algorithm that could overcome two limitation of the current DL concept learning systems, namely avoiding the computation of the most specific concepts (which is also language-dependent) and the excessive (syntactic) complexity of the

**Algorithm 1** GENERALIZE(*Positives*, *Negatives*, *Unlabeled*): *Generalization*

**Require:** *Positives*, *Negatives*, *Unlabeled*: positive, negative and unlabeled individuals
**Ensure:** *Generalization*: concept definition solving the learning problem

```
1: Generalization ← ⊥
2: PositivesToCover ← Positives
3: while PositivesToCover ≠ ∅ do
4:    PartialDef ← ⊤
5:    CoveredNegatives ← Negatives
6:    while CoveredNegatives ≠ ∅ do
7:       PartialDef ← SPECIALIZE(PartialDef, PositivesToCover, CoveredNegatives, Unlabeled)
8:       CoveredNegatives ← {n ∈ Negatives | K ⊨ ¬PartialDef(n)}
9:    end while
10:   CoveredPositives ← {p ∈ PositivesToCover | K ⊨ PartialDef(p)}
11:   Generalization ← Generalization ⊔ PartialDef
12:   PositivesToCover ← PositivesToCover \ CoveredPositives
13: end while
14: return Generalization
```

resulting generalizations. For instance, the algorithm presented in [19] requires lifting the instances to the concept level through a suitable approximate MSC operator and then start learning from such extremely specific concept descriptions. This setting has the disadvantages of approximation and language-dependency. In DL-LEARNER [31] these drawbacks are partly mitigated because a learning procedure grounded on a genetic programming based on refinement operators is adopted, whose fitness is computed on the grounds of the covered instances (retrieval). More heuristics and approximated retrieval procedures are further investigated in [17].

The DL-FOIL algorithm essentially adapts the original FOIL algorithm [35] to the different learning problem with DL knowledge bases. Together with a sequential covering procedure, it exploits the (downward) refinement operators defined in Sect. 2.2.1 and a heuristic similar to the *information gain* to select among candidate specialization. Various search strategies have been experimented as well as evaluation measures. Those that we will present in the following are those which gave the best results. A sketch of the main routine of the learning procedure is reported as Alg. 1. Like in the original FOIL algorithm, the generalization routine computes (partial) generalizations as long as they do not cover any negative example. If this occurs, the specialization routine is invoked for solving these sub-problems. This routine applies the idea of specializing using the (incomplete) refinement operator defined in the previous section. The specialization continues until no negative example is covered (or a very limited amount[4] of them). The partial generalizations built on each outer loop are finally grouped together in a disjunction which is an allowed constructor for more expressive logics than (or equal to) $\mathcal{ALC}$. Also the outer while-loop can be exited before covering all the positive examples for avoiding overfitting generalizations.

The specialization function SPECIALIZE (reported as Alg. 2) is called from within the inner loop of the generalization procedure in order to specialize an overly general partial generalization. The function searches for proper refinements that provide at least a minimal gain (see below) fixed with a threshold (*MINGAIN*). Specializations are ran-

[4]The actual exit-condition for the inner loop may be: $1 - |\mathit{CoveredNegatives}|/|\mathit{Negatives}| < \varepsilon$, for some small constant $\varepsilon$.

**Algorithm 2** SPECIALIZE(*PartialDef*, *Positives*, *Negatives*, *Unlabeled*): *Refinement*

**Require:**
*PartialDef*: concept definition
*Positives*, *Negatives*, *Unlabeled*: (positive, negative and unlabeled) individuals
**Ensure:** *Refinement*: concept definition
1: **const:**
*MINGAIN*: minimal acceptable gain;
*NUMSPECS*: number of specializations to be generated
2: *bestGain* $\leftarrow 0$
3: **while** *bestGain* $<$ *MINGAIN* **do**
4: **for** $i \leftarrow 1$ **to** *NUMSPECS* **do**
5: *Specialization* $\leftarrow$ GETRANDOMREFINEMENT($\rho$, *PartialDef*)
6: *CoveredPositives* $\leftarrow \{p \in$ *Positives* $\mid \mathcal{K} \models$ *Specialization*$(p)\}$
7: *CoveredNegatives* $\leftarrow \{n \in$ *Negatives* $\mid \mathcal{K} \models \neg$*Specialization*$(n)\}$
8: *thisGain* $\leftarrow$ GAIN(*CoveredPositives*, *CoveredNegatives*, *Unlabeled*, *Positives*, *Negatives*)
9: **if** *thisGain* $>$ *bestGain* **then**
10: *bestConcept* $\leftarrow$ *Specialization*
11: *bestGain* $\leftarrow$ *thisGain*
12: **end if**
13: **end for**
14: **end while**
15: **return** *Refinement*

domly generated using the $\rho$ operator defined in Sect. 2.2.1, especially $\rho_{\sqcap}$ is exploited with the addition of new conjuncts or the specialization of primitive concepts or role restrictions. This is similar to the original FOIL algorithm, where new random literals are appended to clauses' antecedents. A first random choice is made between atomic concepts or role restrictions. In the latter case another random choice is made between existential and universal restriction. In all cases the required concept and roles names are also randomly selected. This may give a way to impose some further bias to the form of the concept descriptions to be induced.

As regards the heuristic employed to guide the search, it was shown [21] that the gain function has to take into account incomplete examples. Similarly to a semi-supervised learning setting, the gain value $g$ that is computed in GAIN() for selecting the best refinement is obtained as follows:

$$g = p_1 \cdot \left[ \log \frac{p_1 + u_1 w_1}{p_1 + n_1 + u_1} - \log \frac{p_0 + u_0 w_0}{p_0 + n_0 + u_0} \right]$$

where $p_1$, $n_1$ and $u_1$ represent, resp., the number of positive, negative and unlabeled examples covered by the specialization; $p_0$, $n_0$ and $u_0$ stand for the number of positive, negative and unlabeled examples covered by the former definition, the weights $w_0, w_1$ can be determined by an estimate of the prior probability of the positive examples, resp., in the current and former concept definition. To avoid the case of null numerators, a further correction of the probabilities is performed by resorting to an $m$-estimate procedure.

The overall complexity of the algorithm is largely determined by the calls to reasoning services, namely subsumption (satisfiability) and instance-checking. If we consider only concepts expressed in $\mathcal{ALC}$ logic, the complexity of these inferences is P-space. However, should the considered knowledge base contain definitions expressed in more

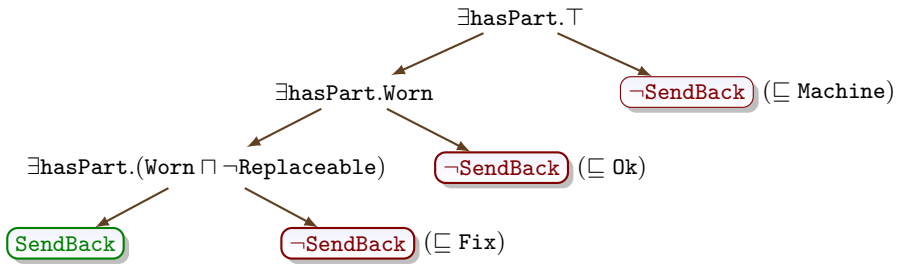



**Figure 4.** A TDT whose leftmost path corresponds to the DL concept definition SendBack ≡ ∃hasPart.(Worn ⊓ ¬Replaceable). Other definitions can be associated to the paths to leaves labeled with ¬SendBack that are related to other (disjoint) concepts.

complex languages which require more complex reasoning procedures. The inductive algorithm can be thought as a means for building (upper) $\mathcal{ALC}$-approximations of the target concepts. The number of nodes visited during the traversal of the search space grows with richness of the vocabulary, yet not with the expressiveness of the underlying DL, because the algorithm searches a sub-space of the actual search space induced by the language.

## 5. Learning Terminological Decision Trees

*First-order logical decision trees* (FOLDTs) [5] are binary decision trees in which

1. the nodes contain tests in the form of conjunctions of literals;
2. left and right branches stand, resp., for the truth-value (resp. true and false) determined by the test evaluation;
3. different nodes may share variables, yet a variable that is introduced in a certain node must not occur in the right branch of that node.

*Terminological decision trees* (TDTs) extend the original definition, allowing DL concept descriptions as (variable-free) node tests. Fig. 4 shows a TDT denoting also the definition of the SendBack concept as in the problem described in [5].

### 5.1. Classification

The TDTs can be used for classifying individuals. Alg. 3 shows the related classification procedure. It uses other functions: LEAF() to determine whether a node is a leaf of the argument tree, ROOT() which returns the root node of the input tree, and INODE() which retrieves the test concept and the left and right subtrees branching from a given internal node. Given an individual $a$, starting from the root node, the algorithm checks the class-membership w.r.t. the test concept $D_i$ in the current node, i.e. $\mathcal{K} \models D_i(a)$, sorting $a$ to the left branch if the test is successful while the right branch is chosen if $\mathcal{K} \models \neg D_i(a)$. Eventually the classification is found as a leaf-node concept.

Note that the open-world semantics may cause unknown answers (failure of both left and right branch tests) that can be avoided by considering a weaker (default) right-branch test: $\mathcal{K} \not\models D_i(a)$. This differs from the FOLDTs where the test actually consists of several conjunctions that occur in the path from the root to the current node.

**Algorithm 3** Classification with TDTs.
CLASSIFY($a$: individual, $T$: TDT, $\mathcal{K}$: KB): concept;

1. $N \leftarrow \text{ROOT}(T)$;
2. **while** $\neg\text{LEAF}(N, T)$ **do**
   (a) $(D, T_{\text{left}}, T_{\text{right}}) \leftarrow \text{INODE}(N)$;
   (b) **if** $\mathcal{K} \models D(a)$ **then** $N \leftarrow \text{ROOT}(T_{\text{left}})$
   (c) **elseif** $\mathcal{K} \models \neg D(a)$ **then** $N \leftarrow \text{ROOT}(T_{\text{right}})$
   (d) **else return** $\top$
3. $(D, \cdot, \cdot) \leftarrow \text{INODE}(N)$;
4. **return** $D$;

### *5.2. From Terminological Decision Trees to Concept Descriptions*

Note that each node in a path may be used to build a concept description through specializations. This can be given 1) by adding a conjunctive concept description, 2) by refining a sub-description in the scope of an existential, universal or number restriction or 3) by narrowing a number restriction (which may be allowed by the underlying language, e.g. $\mathcal{ALN}$ or $\mathcal{ALCQ}$). No special care[5] is to be devoted to negated atoms and their variables.

For each target concept name $C$ it is possible to derive a *single* concept definition from a TDT. The algorithm (see Alg. 4) follows all the paths leading to success nodes i.e. leaves labeled with $C$ (the heads of the clauses in the original setting) collecting the intermediate test concepts (formerly, the body literals). In this way, each path yields a different conjunctive concept description that represents a different version of the target concept in conjunctive form $D_i = D_1^i \sqcap \cdots \sqcap D_l^i$. The final single description for the target concept is obtained as the disjunctive description built with concepts from this finite set $S = \{D_i\}_{i=1}^M$. Hence, the final definition is $C \equiv \bigsqcup_{i=1}^M D_i$. As an example, looking at the TDT depicted in Figure 4, a concept definition that may be extracted is

$$\texttt{Ok} \equiv \exists\texttt{hasPart}.\top \sqcap \neg\exists\texttt{hasPart}.\texttt{Worn} \equiv \exists\texttt{hasPart}.\top \sqcap \forall\texttt{hasPart}.\neg\texttt{Worn}$$

i.e. something that has exclusively parts which are not worn.

Like in the original logic tree induction setting, also internal nodes may be utilized to induce new intermediate concepts.

### *5.3. Induction of TDTs*

The subsumption relationship $\sqsubseteq$ induces a partial order on the space of DL concept descriptions. Then, as seen above, the learning task can be cast as a search for a solution of the problem in the partially ordered space. In such a setting, suitable operators to traverse the search space are required [19,32]. While existing DL concept induction algorithms generally adopt a separate-and-conquer covering strategy, the TDT-learning algorithm adopts a divide-and-conquer strategy [6]. It also tries to cope with the limitations of the other learning systems, namely approximation and language-dependence. Indeed, since the early works [10], instances are required to be transposed to the concept level before the learning can start. This is accomplished by resorting to the computation, for each training individual, of the related MSC the individual belongs to [3], which need not ex-

[5] We are considering expressive (and decidable) DL languages like $\mathcal{ALCQ}$, that are endowed with full negation, hence the situation is perfectly symmetric.

**Algorithm 4** Mapping a TDT onto a DL concept description.

DERIVEDEFINITION($C$: concept name, $T$: TDT): concept description;

1. $S \leftarrow$ ASSOCIATE$(C, T, \top)$;
2. **return** $\bigsqcup_{D \in S} D$;

ASSOCIATE($C$: concept name; $T$: TDT; $D_c$: current concept description): set of descriptions;

1. $N \leftarrow$ ROOT$(T)$;
2. $(D_n, T_{\text{left}}, T_{\text{right}}) \leftarrow$ INODE$(N)$;
3. **if** LEAF$(N, T)$ **then**

   (a) **if** $D_n = C$ **then**
   **return** $\{D_c\}$;
   **else**
   **return** $\emptyset$;

   **else**

   (a) $S_{\text{left}} \leftarrow$ ASSOCIATE$(C, T_{\text{left}}, D_c \sqcap D_n)$;
   (b) $S_{\text{right}} \leftarrow$ ASSOCIATE$(C, T_{\text{right}}, D_c \sqcap \neg D_n)$;
   (c) **return** $S_{\text{left}} \cup S_{\text{right}}$;

ist, especially for expressive DLs, and thus has to be approximated. Even in an approximated version, the MSCs turn out to be extremely specific descriptions which affects both the efficiency of learning and the effectiveness of the learned descriptions as this specificity easily leads to overfitting the data [19].

The algorithms implemented by DL-LEARNER [32] partly mitigate these disadvantages being based on stochastic search using refinement operators and a heuristic computed on the grounds of the covered individuals (and a syntactic notion of concept length). Generate-and-test strategies may fall short when considering growing search spaces determined by more expressive languages. This drawback is hardly avoidable and it has been tackled by allowing more interaction with the knowledge engineer which can be presented with partial solutions and then decide to stop further refinements.

Our TDT-induction algorithm adapts the classic schema implemented by C4.5 [34] and TILDE [5]. A sketch of the main routine is reported as Alg. 5. It reflects the standard tree induction algorithms with the addition of the treatment of unlabeled training individuals. The three initial conditions take care of the base cases of the recursion, namely:

1. no individuals got sorted to the current subtree root then the resulting leaf is decided on the grounds of the prior probabilities of positive and negative instances (resp. $Pr_+$ and $Pr_-$);
2. no negative individual yet a sufficient rate (w.r.t. the threshold $\theta$) of positive ones got sorted to the current node, then the leaf is labeled accordingly;
3. dual case w.r.t. to the previous one.

The second half of the algorithm (randomly) generates a set *Specs* of (satisfiable) candidate descriptions (calling GENERATENEWCONCEPTS), that can specialize the current description $D$ when added as a conjunction. Then, the best one ($D_{best}$) is selected in terms of an improvement of the purity of the subsets of individuals resulting from a split based on the test description. The (im)purity measure is based on the entropic *infor-*

**Algorithm 5** The main routine for inducing terminological decision trees
INDUCETDTREE($C$: concept name; $D$: current description; *Ps*, *Ns*, *Us*: set of (positive, negative, unlabeled) training individuals): TDT;

```
1: const θ;                                                    {purity threshold}
2: Initialize new TDT T;
3: if |Ps| = 0 and |Ns| = 0 then
4:    if Pr+ ≥ Pr− then
5:       T.root ← C
6:    else
7:       T.root ← ¬C;
8:    end if
9:    return T;
10: end if
11: if |Ns| = 0 and |Ps|/(|Ps| + |Us|) > θ then
12:    T.root ← C; return T;
13: end if
14: if |Ps| = 0 and |Ns|/(|Ns| + |Us|) > θ then
15:    T.root ← ¬C; return T;
16: end if
17: Specs ← GENERATENEWCONCEPTS(D, Ps, Ns);
18: D_best ← SELECTBESTCONCEPT(Specs, Ps, Ns, Us);
19: ((P^l, N^l, U^l), (P^r, N^r, U^r)) ← SPLIT(D_best, Ps, Ns, Us);
20: T.root ← D_best;
21: T.left ← INDUCETDTREE(C, D ⊓ D_best, P^l, N^l, U^l);
22: T.right ← INDUCETDTREE(C, D ⊓ ¬D_best, P^r, N^r, U^r);
23: return T;
```

*mation gain* [34] or on the *Gini index* which was finally preferred. In the DL setting the problem is made more complex by the presence of instances which cannot be labelled as positive or negative (see [12]) whose contributions are considered as proportional to the prior distribution of positive and negative examples.

Once the best description $D_{best}$ has been selected (calling SELECTBESTCONCEPT), it is installed as the current subtree root and the sets of individuals sorted to this node are subdivided according to their classification w.r.t. such a concept. Note that unlabeled individuals must be sorted to both subtrees. Finally the recursive calls for the construction of the subtrees are made, passing the proper sets of individuals and the concept descriptions $D \sqcap D_{best}$ and $D \sqcap \neg D_{best}$ related to either path.

The resulting system, TERMITIS (TERMI*nological* T*ree* I*nduction* S*ystem*), ver. 1.2, was applied, for comparative purposes, to ontologies that have been considered in previous experiments with other DL learning systems [13].

## 6. Implementation

### *6.1. The Protégé Plugin*

After implementing and testing the learning algorithm described in Sect. 3, it has been integrated into *Protégé* and *OntoWiki*. We extended the Protégé 4 plugin mechanism to be able to integrate the DL-Learner plugin as an additional method to create class

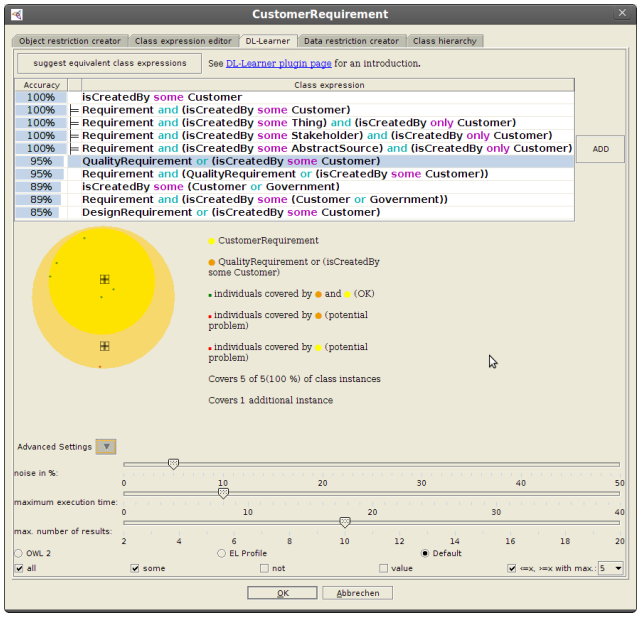



**Figure 5.** A screenshot of the DL-Learner Protégé plugin. It is integrated as additional tab to create class expressions in Protégé. The user is only required to press the "suggest equivalent class expressions" button and within a few seconds they will be displayed ordered by accuracy. If desired, the knowledge engineer can visualize the instances of the expression to detect potential problems. At the bottom, optional expert configuration settings can be adopted.

expressions. The plugin has also become part of the official Protégé 4 repository. A screenshot of the plugin is shown in Fig. 5. To use the plugin, the knowledge engineer is only required to press a button, which then starts a new thread, in the background, that executes the learning algorithm. The used algorithm is an *anytime algorithm*, i.e. at each point in time we can always see the currently best suggestions. The GUI updates the suggestion list each second until the maximum runtime – 10 seconds per default – is reached. For each suggestion, the plugin displays its accuracy. When clicking on a suggestion, it is visualized by displaying two circles: One stands for the instances of the class to describe and another circle for the instances of the suggested class expression. Ideally, both circles overlap completely, but in practice this will often not be the case. Clicking on the plus symbol in each circle shows its list of individuals. Those individuals are also presented as points in the circles and moving the mouse over such a point shows information about the respective individual. Red points show potential problems, where it is important to note that we use a closed world assumption to detect those. If there is not only a potential problem, but adding the expression would render the ontology inconsistent, the suggestion is marked red and a warning message is displayed. Accepting such a suggestion can still be a good choice, because the problem often lies elsewhere in the knowledge base, but was not obvious before, since the ontology was not sufficiently expressive for reasoners to detect it. This is illustrated by a screencast available from the plugin homepage,[6] where the ontology becomes inconsistent after adding the axiom, and the real source of the problem is fixed afterwards. Being able to make such suggestions can be seen as a strength of the plugin.

The plugin allows the knowledge engineer to change expert settings. Those settings include the maximum suggestion search time, the number of results returned and settings related to the desired target language., e.g. the knowledge engineer can choose to stay

[6] `http://dl-learner.org/wiki/ProtegePlugin`

within the OWL 2 EL profile or enable/disable certain class expression constructors. The learning algorithm is designed to be able to handle noisy data and the visualisation of the suggestions will reveal false class assignments so that they can be fixed afterwards.

### 6.2. The OntoWiki Plugin

Analogous to Protégé, we created a similar plugin for OntoWiki [2,1]. OntoWiki is a lightweight ontology editor, which allows distributed and collaborative editing of knowledge bases. The DL-Learner plugin is technically realized by implementing an OntoWiki component, which contains the core functionality, and a module, which implements the UI embedding. The DL-Learner plugin can be invoked from several places in OntoWiki, for instance through the context menu of classes. The plugin accesses DL-Learner functionality through its WSDL-based web service interface. Jar files containing all necessary libraries are provided by the plugin. If a user invokes the plugin, it scans whether the web service is online at its default address. If not, it is started automatically.

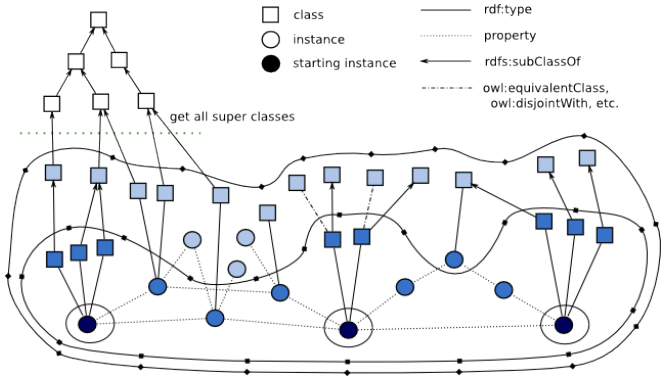



**Figure 6.** Extraction with three starting instances. The circles represent different recursion depths. The circles around the starting instances signify recursion depth 0. The larger inner circle represents the fragment with recursion depth 1 and the largest outer circle with recursion depth 2. Figure taken from [17].

A major technical difference compared to the Protégé plugin is that the knowledge base is accessed via SPARQL, since OntoWiki is a SPARQL-based web application. In Protégé, the current state of the knowledge base is stored in memory in a Java object. As a result, we cannot easily apply a reasoner on an OntoWiki knowledge base. To overcome this problem, we use the DL-Learner fragment selection mechanism described in [17]. Starting from a set of instances, the mechanism extracts a relevant fragment from the underlying knowledge base up to some specified recursion depth. Fig. 6 provides an overview of the fragment selection process. The fragment has the property that learning results on it are similar to those on the complete knowledge base. For a detailed description see [17]. The fragment selection is only performed for medium to large-sized knowledge bases. Small knowledge bases are retrieved completely and loaded into the reasoner. While the fragment selection can cause a delay of several seconds before the learning algorithm starts, it also offers flexibility and scalability. For instance, we can learn class expressions in large knowledge bases such as DBpedia in OntoWiki. Fig. 7 shows a screenshot of the OntoWiki plugin applied to the SWORE [37] ontology. Sug-

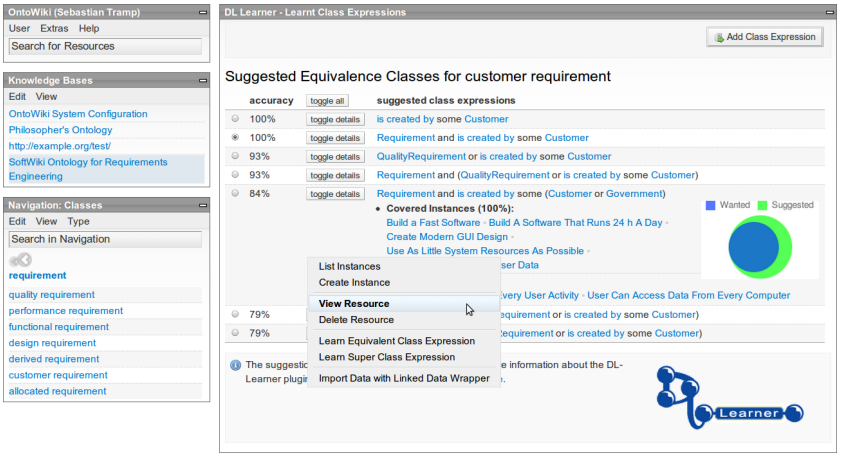



**Figure 7.** Screenshot of the result table of the DL-Learner plugin in OntoWiki.

gestions for learning the class "customer requirement" are shown in Manchester OWL Syntax. Similar to the Protégé plugin, the user is presented a table of suggestions along with their accuracy value. Additional details about the instances of "customer requirement" covered by a suggested class expressions and additionally contained instances can be viewed via a toggle button. The modular design of OntoWiki allows rich user interaction: Each resource, e.g. a class, property, or individual, can be viewed and subsequently modified directly from the result table as shown for "design requirement" in the screenshot. For instance, a knowledge engineer could decide to import additional information available as Linked Data and run the CELOE algorithm again to see whether different suggestions are provided with additional background knowledge.

## 7. Conclusions

Ontology construction may be a burdensome and time consuming task. To cope with this problem, the usage of machine learning techniques has been proposed. Specifically, the problem is regarded as a (supervised) concept learning problem where, given a set of individual resources labeled as instances of a target concept, the goal is to find an intensional concept description for them. Particularly, from those labels, axioms can be induced, which can then be confirmed by the knowledge engineer. The concept learning problem is tackled as a search through a space of candidate descriptions in the reference representation guided by exemplars of the target concepts. Those techniques are also applicable in other domains, e.g. question answering [28]. After surveyed existing methods and some basics on refinement operators, three different algorithms have been presented and compared in more detail: DL-FOIL, CELOE and TERMITIS.

## References

[1] Sören Auer, Sebastian Dietzold, Jens Lehmann, and Thomas Riechert. OntoWiki: A tool for social, semantic collaboration. In Natalya Fridman Noy, Harith Alani, Gerd Stumme, Peter Mika, York Sure, and

Denny Vrandecic, editors, *Proceedings of the Workshop on Social and Collaborative Construction of Structured Knowledge (CKC 2007) at the 16th International World Wide Web Conference (WWW2007) Banff, Canada, May 8, 2007*, volume 273 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2007.

[2] Sören Auer, Sebastian Dietzold, and Thomas Riechert. Ontowiki - a tool for social, semantic collaboration. In *ISWC 2006*, volume 4273 of *LNCS*, pages 736–749. Springer, 2006.

[3] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniel Nardi, and Peter Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003.

[4] Liviu Badea and Shan hwei Nienhuys-Cheng. A refinement operator for description logics. In *Proc. of the Int. Conf. on Inductive Logic Programming*, volume 1866 of *LNAI*, pages 40–59. Springer, 2000.

[5] Hendrik Blockeel and Luc De Raedt. Top-down induction of first-order logical decision trees. *Artificial Intelligence*, 101(1-2):285–297, 1998.

[6] Henrik Boström. Covering vs. divide-and-conquer for top-down induction of logic programs. In *Proc. of the Int. Joint Conf. on Artificial Intelligence, IJCAI95*, pages 1194–1200. Morgan Kaufmann, 1995.

[7] Lorenz Bühmann and Jens Lehmann. Universal OWL axiom enrichment for large knowledge bases. In *Proceedings of EKAW 2012*, 2012.

[8] Lorenz Buhmann and Jens Lehmann. Pattern based knowledge base enrichment. In *12th International Semantic Web Conference, 21-25 October 2013, Sydney, Australia*, 2013.

[9] William W. Cohen and Haym Hirsh. Learnability of description logics. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*. ACM Press, 1992.

[10] William W. Cohen and Haym Hirsh. Learning the CLASSIC description logic. In *Proc. of the Int. Conf. on Principles of Knowledge Representation and Reasoning*, pages 121–133. Morgan Kaufmann, 1994.

[11] Floriana Esposito, Nicola Fanizzi, Luigi Iannone, Ignazio Palmisano, and Giovanni Semeraro. Knowledge-intensive induction of terminologies from metadata. In *The Semantic Web – ISWC 2004: Third International Semantic Web Conference. Proceedings*, pages 441–455. Springer, 2004.

[12] Nicola Fanizzi, Claudia d'Amato, and Floriana Esposito. DL-FOIL: Concept learning in description logics. In F. Zelezný and N. Lavrac, editors, *Proceedings of the 18th International Conference on Inductive Logic Programming, ILP2008*, volume 5194 of *LNAI*, pages 107–121. Springer, 2008.

[13] Nicola Fanizzi, Claudia d'Amato, and Floriana Esposito. Induction of concepts in web ontologies through terminological decision trees. In José L. Balcázar et al., editors, *Proceedings of ECML PKDD 2010, Part I*, volume 6321 of *LNCS/LNAI*, pages 442–457. Springer, 2010.

[14] Nicola Fanizzi, Floriana Esposito, Stefano Ferilli, and Giovanni Semeraro. A methodology for the induction of ontological knowledge from semantic annotations. In *Proc. of the Conf. of the Italian Association for Artificial Intelligence*, volume 2829 of *LNAI/LNCS*, pages 65–77. Springer, 2003.

[15] Nicola Fanizzi, Stefano Ferilli, Luigi Iannone, Ignazio Palmisano, and Giovanni Semeraro. Downward refinement in the $\mathcal{ALN}$ description logic. In *Proceedings of the 4th International Conference on Hybrid Intelligent Systems, HIS2004*, pages 68–73. IEEE Computer Society, 2005.

[16] Sally A. Goldman, Stephen S. Kwek, and Stephen D. Scott. Learning from examples with unspecified attribute values. *Information and Computation*, 180(2):82–100, 2003.

[17] Sebastian Hellmann, Jens Lehmann, and Sören Auer. Learning of OWL class descriptions on very large knowledge bases. *International Journal on Semantic Web and Information Systems*, 5(2):25–48, 2009.

[18] Sebastian Hellmann, Jens Lehmann, and Sören Auer. Learning of owl class expressions on very large knowledge bases and its applications. In Interoperability Semantic Services and Web Applications: Emerging Concepts, editors, *Learning of OWL Class Expressions on Very Large Knowledge Bases and its Applications*, chapter 5, pages 104–130. IGI Global, 2011.

[19] Luigi Iannone, Ignazio Palmisano, and Nicola Fanizzi. An algorithm based on counterfactuals for concept learning in the semantic web. *Applied Intelligence*, 26(2):139–159, 2007.

[20] Josué Iglesias and Jens Lehmann. Towards integrating fuzzy logic capabilities into an ontology-based inductive logic programming framework. In *Proc. of the 11th International Conference on Intelligent Systems Design and Applications (ISDA)*, 2011.

[21] Nobuhiro Inuzuka, Masakage Kamo, Naohiro Ishii, Hirohisa Seki, and Hidenori Itoh. Tow-down induction of logic programs from incomplete samples. In *Selected Papers from the 6th International Workshop on Inductive Logic Programming, ILP96*, volume 1314 of *LNAI*, pages 265–282. Springer, 1997.

[22] Jörg Uwe Kietz and Katharina Morik. A polynomial approach to the constructive induction of structural knowledge. *Machine Learning*, 14(2):193–218, 1994.

[23] Agnieszka Ławrynowicz and Volker Tresp. Introducing machine learning. In Jens Lehmann and Johanna Völker, editors, *Perspectives on Ontology Learning*, Studies on the Semantic Web. AKA Heidelberg /

IOS Press, 2014.

[24] Jens Lehmann. Hybrid learning of ontology classes. In *Proc. of the 5th Int. Conference on Machine Learning and Data Mining MLDM*, volume 4571 of *Lecture Notes in Computer Science*, pages 883–898. Springer, 2007.

[25] Jens Lehmann. DL-Learner: learning concepts in description logics. *Journal of Machine Learning Research (JMLR)*, 10:2639–2642, 2009.

[26] Jens Lehmann. *Learning OWL Class Expressions*. PhD thesis, University of Leipzig, 2010. PhD in Computer Science.

[27] Jens Lehmann, Sören Auer, Lorenz Bühmann, and Sebastian Tramp. Class expression learning for ontology engineering. *Journal of Web Semantics*, 9:71 – 81, 2011.

[28] Jens Lehmann and Lorenz Bühmann. Autosparql: Let users query your knowledge base. In *Proceedings of ESWC 2011*, 2011.

[29] Jens Lehmann and Christoph Haase. Ideal downward refinement in the el description logic. In *Proc. of the Int. Conf. on Inductive Logic Programming*, volume 5989 of *LNCS*, pages 73–87. Springer, 2009.

[30] Jens Lehmann and Pascal Hitzler. Foundations of refinement operators for description logics. In *ILP 2007*, volume 4894 of *LNCS*, pages 161–174. Springer, 2008.

[31] Jens Lehmann and Pascal Hitzler. A refinement operator based learning algorithm for the ALC description logic. In *ILP 2007*, volume 4894 of *LNCS*, pages 147–160. Springer, 2008.

[32] Jens Lehmann and Pascal Hitzler. Concept learning in description logics using refinement operators. *Machine Learning journal*, 78(1-2):203–250, 2010.

[33] Tom M. Mitchell. Generalization as search. *Artificial Intelligence*, 18(2):203–226, 1982.

[34] J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[35] J. Ross Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.

[36] Luc De Raedt. *Logical and Relational Learning*. Springer, 2008.

[37] Thomas Riechert, Kim Lauenroth, Jens Lehmann, and Sören Auer. Towards semantic based requirements engineering. In *I-KNOW 2007*, 2007.

[38] Gunnar Teege. A subtraction operation for description logics. In *Proc. of the Int. Conf. on Principles of Knowledge Representation and Reasoning*, pages 540–550. Morgan Kaufmann, 1994.

# Learning Onto-Relational Rules with Inductive Logic Programming

Francesca A. LISI,
*Dipartimento di Informatica, Università degli Studi di Bari "Aldo Moro", Italy*
*lisi@di.uniba.it*

**Abstract.** Rules complement and extend ontologies on the Semantic Web. We refer to these rules as onto-relational since they combine DL-based ontology languages and Knowledge Representation formalisms supporting the relational data model within the tradition of Logic Programming and Deductive Databases. Rule authoring is a very demanding Knowledge Engineering task which can be automated though partially by applying Machine Learning algorithms. In this chapter we show how Inductive Logic Programming (ILP), born at the intersection of Machine Learning and Logic Programming and considered as a major approach to Relational Learning, can be adapted to Onto-Relational Learning. For the sake of illustration, we provide details of a specific Onto-Relational Learning solution to the problem of learning rule-based definitions of DL concepts and roles with ILP.

**Keywords.** Inductive Logic Programming, Rule Languages and Systems, Integration of Rules and Ontologies, Deductive Databases.

## Introduction

*Rules* are widely used in Knowledge Engineering (KE) and Knowledge Representation (KR) as a powerful way of modeling knowledge. In the broadest sense, a rule could be any statement which says that a certain conclusion must be valid whenever a certain premise is satisfied, *i.e.* any statement that could be read as a sentence of the form "if .. then ..". Rules have been successfully applied in the fields of Logic Programming (LP) and Deductive Databases [6]. Rules play also a role in the *Semantic Web* architecture. Interest in this area has grown rapidly over recent years as testified by the Rules Interchange Format (RIF)[1] activity at W3C. Rules from the RIF perspective would allow the integration, transformation and derivation of data from numerous sources in a distributed, scalable, and transparent manner. Because of the great variety in rule languages and rule engine technologies, RIF consists of a core language[2] to be used along with a set of standard and non-standard extensions. These extensions need not all be combinable into a single unified language. As for the expressive power, two directions are followed: monotonic extensions towards full First Order Logic (FOL) and non-monotonic (NM) extensions based on the LP tradition. The debate around a RIF has taken a long time also due to the controversial issue of having rules on top or aside ontologies [20]. There is

[1] `http://www.w3.org/2005/rules/wiki/RIF_Working_Group`
[2] `http://www.w3.org/TR/rif-core/`

a consensus now on the fact that rules complement and extend ontologies. Indeed, rules can be used in combination with ontologies, or as a means to specify ontologies. They are also frequently applied over ontologies, to draw inferences, express constraints, specify policies, react to events, discover new knowledge, transform data, etc. In particular, RIF rules can refer to RDF and OWL facts. Since the design of OWL has been based on the $\mathcal{SH}$ family of very expressive *Description Logics* (DLs) (cf. Krötzsch et al. [23] in this volume), the NM dialects of RIF will most likely be inspired by those hybrid KR systems that integrate DLs and LP. Such rule formalisms are of interest to this chapter. We shall refer to them as *onto-relational rule languages* from now on. Apart from the specific ontology language, the integration of ontologies and rules is already present in existing knowledge bases (KBs). Notably the Cyc[3] KB consists of terms (which constitute the vocabulary, *i.e.* the ontology) and assertions which relate those terms and include both simple ground assertions and rules [26].

The acquisition of rules for very large KBs like Cyc is a very demanding KE activity. Indeed, according to an estimate from the Cyc project, human experts produce rules at the rate of approximately three per hour but can evaluate an average of twenty rules per hour. Also, for untrained knowledge engineers, while rule authoring may be very difficult, rule reviewing is feasible (although still difficult). A partial automation of the rule authoring task, *e.g.*by applying *Machine Learning* (ML) algorithms (cf. Lawrynowicz and Tresp [24] in this volume), can be of help even though the automatically produced rules are not guaranteed to be correct. In fact, of those rules, some will turn out to be correct, and some will be found to need editing to be assertible. Yet, as mentioned above, rule reviewing is less critical than rule authoring. In order to partially automate the authoring of onto-relational rules, the bunch of ML techniques collectively known under the name of *Inductive Logic Programming* (ILP) [44] seems particularly promising for the following reasons. ILP was born at the intersection of ML and LP [43], and is widely recognized as a major approach to *Relational Learning* [7]. Apart from the KR framework of LP, the distinguishing feature of ILP, also with respect to other ML forms, is the use of prior domain knowledge in the form of a logical theory during the induction process. In this chapter we take a critical look at ILP proposals for learning relational rules while having an ontology as the background theory. These proposals try to overcome the difficulties of accommodating ontologies in Relational Learning. The work of [3] on using semantic meta-knowledge from Cyc as inductive bias in an ILP system is another attempt at solving this problem though more empirically. Conversely, we promote an extension of Relational Learning, called *Onto-Relational Learning* (ORL), which accounts for ontologies in a clear, elegant and well-founded manner by resorting to onto-relational rule languages. In this chapter, for the sake of illustration, we provide details of a specific ORL solution to the problem of learning rule-based definitions of DL concepts and roles with ILP.

The chapter is organized as follows. Section 1 is devoted to preliminaries on LP and its applications to databases and ontologies as well as on ILP. Section 2 provides a state-of-the-art survey of ILP proposals for learning onto-relational rules. Section 3 describes in depth the most powerful of these proposals. Section 4 concludes the chapter with final remarks and outlines directions of future work.

[3] `http://cyc.com/cyc/technology/whatiscyc_dir/`

## 1. Preliminaries

### 1.1. Logic Programming and databases

Logic Programming (LP) is rooted into a fragment of Clausal Logics (CLs) known as Horn Clausal Logic (HCL) [38]. The basic element in CLs is the *atom* of the form $p(t_i, \ldots, t_{k_i})$ such that each $p$ is a predicate symbol and each $t_j$ is a term. A *term* is either a constant or a variable or a more complex term obtained by applying a functor to simpler term. Constant, variable, functor and predicate symbols belong to mutually disjoint alphabets. A *literal* is an atom either negated or not. A *clause* is a universally quantified disjunction of literals. Usually the universal quantifiers are omitted to simplify notation. Alternative notations are a clause as set of literals and a clause as an implication. A *program* is a set of clauses. HCL admits only so-called definite clauses. A *definite clause* is an implication of the form

$$\alpha_0 \leftarrow \alpha_1, \ldots, \alpha_m$$

where $m \geq 0$ and $\alpha_i$ are atoms, *i.e.* a clause with exactly one positive literal. The right-hand side $\alpha_0$ and the left-hand side $\alpha_1, \ldots, \alpha_m$ of the implication are called *head* and *body* of the clause, respectively. Note that the body is intended to be an existentially quantified conjunctive formula $\exists \alpha_1 \wedge \ldots \wedge \alpha_m$. Furthermore definite clauses with $m > 0$ and $m = 0$ are called *rules* and *facts* respectively. The model-theoretic semantics of HCL is based on the notion of *Herbrand interpretation*, *i.e.* an interpretation in which all all constants and function symbols are assigned very simple meanings. This allows the symbols in a set of clauses to be interpreted in a purely syntactic way, separated from any real instantiation. The corresponding proof-theoretic semantics is based on the *Closed World Assumption* (CWA), *i.e.* the presumption that what is not currently known to be true, is false. Deductive reasoning with HCL is formalized in its proof theory. In clausal logic *resolution* comprises a single inference rule which, from any two clauses having an appropriate form, derives a new clause as their consequence. Resolution is sound: every resolvent is implied by its parents. It is also refutation complete: the empty clause is derivable by resolution from any set $S$ of Horn clauses if $S$ is unsatisfiable. *Negation As Failure* (NAF) is related to the CWA, as it amounts to believing false every predicate that cannot be proved to be true. Clauses with NAF literals in the body are called *normal clauses*. The concept of a *stable model*, or *answer set*, is used to define a declarative semantics for normal logic programs [17]. According to this semantics, a logic program may have several alternative models (but possibly none), each corresponding to a possible view of the reality. Also based on the stable model (answer set) semantics, *Answer Set Programming* (ASP) is an alternative LP paradigm oriented towards difficult search problems [39].

Definite clauses played a prominent role in the rise of deductive databases [6]. More precisely, functor-free non-recursive definite clauses are at the basis of the language Datalog for deductive databases [5]. Generally, it is denoted by DATALOG$^\neg$ where $\neg$ is treated as NAF. The restriction of Datalog to only positive rules (*i.e.*, rules without NAF literals) is denoted by DATALOG. Based on the distinction between extensional and intensional predicates, a DATALOG program $\Pi$ can be divided into two parts. The *extensional part*, denoted as $EDB(\Pi)$, is the set of facts of $\Pi$ involving the extensional predicates, whereas the *intensional part* $IDB(\Pi)$ is the set of all other clauses of $\Pi$. The main

reasoning task in DATALOG is *query answering*. A *query* $Q$ to a DATALOG program $\Pi$ is a DATALOG clause of the form

$$\leftarrow \alpha_1, \ldots, \alpha_m$$

where $m > 0$, and $\alpha_i$ is a DATALOG atom. An *answer* to a query $Q$ is a substitution $\theta$ for the variables of $Q$. An answer is correct with respect to the DATALOG program $\Pi$ if $\Pi \models Q\theta$. The *answer set* to a query $Q$ is the set of answers to $Q$ that are correct w.r.t. $\Pi$ and such that $Q\theta$ is ground. In other words the answer set to a query $Q$ is the set of all ground instances of $Q$ which are logical consequences of $\Pi$. Answers are computed by refutation.

Disjunctive Datalog (denoted as DATALOG$^{\vee}$) is a variant of DATALOG where disjunctions may appear in the rule heads [10]. Therefore DATALOG$^{\vee}$ can not be considered as a fragment of HCL. Advanced versions (DATALOG$^{\neg\vee}$) also allow for negation in the bodies, which can be handled according to a semantics for negation in CLs. Defining the semantics of a DATALOG$^{\neg\vee}$ program is complicated by the presence of disjunction in the rules' heads because it makes the underlying disjunctive logic programming inherently nonmonotonic, *i.e.* new information can invalidate previous conclusions. Among the many alternatives, one widely accepted semantics for DATALOG$^{\neg\vee}$ is the extension of the stable model semantics to the disjunctive case.

### *1.2. Logic Programming and ontologies*

The integration of LP and ontologies follows the tradition of KR research on so-called *hybrid systems*, *i.e.* those systems which are constituted by two or more subsystems dealing with distinct portions of a single KB by performing specific reasoning procedures [16]. The motivation for investigating and developing such systems is to improve on two basic features of KR formalisms, namely *representational adequacy* and *deductive power*, by preserving the other crucial feature, *i.e. decidability*. Indeed DLs and CLs are FOL fragments incomparable as for the expressiveness [1] and the semantics [46] but combinable at different degrees of integration: Tight, loose, full.

The semantic integration is *tight* when a model of the hybrid KB is defined as the union of two models, one for the DL part and one for the CL part, which share the same domain. In particular, combining DLs with CLs in a tight manner can easily lead to undecidability if the interaction scheme between the DL and the CL part of a hybrid KB does not solve the semantic mismatch between DLs and CLs [47]. This requirement is known as *DL-safety* [42]. With respect to this property, the hybrid KR system CARIN [27] is *unsafe* because the interaction scheme is left unrestricted. Conversely, $\mathcal{AL}$-LOG [8] guarantees a *safe* interaction scheme by means of syntactic restrictions. Finally, $\mathcal{DL}$+LOG$^{\neg\vee}$ [48][4] is *weakly DL-safe* because it relaxes the condition of DL-safety. The distinguishing features of these three KR frameworks are summarized in Table 1 and further discussed in Section 1.2.1, 1.2.2, and 1.2.3 respectively.

The semantic integration is *loose* when the DL part and the CL part are separate components connected through a minimal interface for exchanging knowledge. An example of one such kind of coupling is the integration scheme for ASP and DLs illustrated

[4] We prefer $\mathcal{DL}$+LOG$^{\neg\vee}$ to the original name $\mathcal{DL}$+LOG in order to emphasize the NM features of the language.

**Table 1.** Three KR frameworks suitable for representing onto-relational rules.

| | **CARIN** [27] | $\mathcal{AL}$**-LOG**[8] | $\mathcal{DL}$**+LOG**$^{\neg\vee}$ [48] |
|---|---|---|---|
| DL language | any DL | $\mathcal{ALC}$ | any DL |
| CL language | Horn clauses | DATALOG clauses | DATALOG$^{\neg\vee}$ clauses |
| integration | tight DL-unsafe | tight DL-safe | tight weakly DL-safe |
| rule head literals | DL/Horn literals | DATALOG literal | DL/DATALOG literals |
| rule body literals | DL/Horn literals | $\mathcal{ALC}$/DATALOG literals (no roles) | DL/DATALOG$^{\neg}$ literals |
| semantics | Herbrand models+DL models | idem | stable models+DL models |
| reasoning | SLD-resolution+tableau calculus | idem | stable model computation + Boolean CQ/UCQ containment |
| decidability | only for some instantiations | yes | for all instantiations with DLs for which the Boolean CQ/UCQ containment is decidable |
| implementation | yes, *e.g.*[19] | yes, *e.g.*[51] | unknown |

in [11]. It derives from the previous work of the same authors on the extension of ASP with higher-order reasoning and external evaluations [12] which has been implemented into the system DLVHEX[5].

The semantic integration is *full* when there is no separation between vocabularies of the two parts of the hybrid KB. One such kind of coupling is achieved by means of the logic of Minimal Knowledge and Negation as Failure in [41].

A complete picture of the computational properties of systems combining DL ontologies and DATALOG rules can be found in [49]. An updated survey of the literature on hybrid DL-CL systems [9] is suggested for further reading.

### *1.2.1.* CARIN

A comprehensive study of the effects of combining DLs and CLs (more precisely, Horn rules) can be found in [27]. Special attention is devoted to the DL $\mathcal{ALCNR}$. The results of the study can be summarized as follows: (i) answering conjunctive queries over $\mathcal{ALCNR}$ TBoxes is decidable, (ii) query answering in $\mathcal{ALCNR}$ extended with non-recursive DATALOG rules, where both concepts and roles can occur in rule bodies, is also decidable, as it can be reduced to answering a *union of conjunctive queries* (UCQ)[6], (iii) if rules are recursive, query answering becomes undecidable, (iv) decidability can be regained by disallowing certain combinations of constructors in the logic, and (v) decidability can be regained by requiring rules to be *role-safe*, where at least one variable from each role literal must occur in some non-DL-atom. The integration framework proposed in [27] and known as CARIN is therefore DL-unsafe. Reasoning in CARIN is based on *constrained SLD-resolution*, *i.e.* an extension of SLD-resolution with a tableau calculus for DLs to deal with DL literals in the rules. Constrained SLD-refutation is a complete and sound method for answering *ground* queries.

[5] `http://www.kr.tuwien.ac.at/research/systems/dlvhex/`

[6] A UCQ over a predicate alphabet $P$ is a FOL sentence of the form $\exists\vec{X}.conj_1(\vec{X}) \vee \ldots \vee conj_n(\vec{X})$, where $\vec{X}$ is a tuple of variable symbols and each $conj_i(\vec{X})$ is a set of atoms whose predicates are in $P$ and whose arguments are either constants or variables from $\vec{X}$. A CQ is a UCQ with $n = 1$.

### 1.2.2. $\mathcal{AL}$-LOG

$\mathcal{AL}$-LOG is a hybrid KR system that integrates safely the DL $\mathcal{ALC}$ and DATALOG [8]. In particular, variables occurring in the body of rules may be constrained with $\mathcal{ALC}$ concept assertions to be used as 'typing constraints'. This makes rules applicable only to explicitly named objects. As in CARIN, query answering is decided using the constrained SLD-resolution which however in $\mathcal{AL}$-LOG is decidable and runs in single non-deterministic exponential time.

### 1.2.3. $\mathcal{DL}$+LOG$^{\neg\vee}$

The hybrid KR framework of $\mathcal{DL}$+LOG$^{\neg\vee}$ allows a $\mathcal{DL}$ KB, *i.e.* a KB expressed in any DL, to be extended with weakly DL-safe DATALOG$^{\neg\vee}$ rules [48]. Weak DL-safeness allows to overcome the main representational limits of the DL-safe approaches, *e.g.* the possibility of expressing UCQs, by keeping the integration scheme still decidable. For $\mathcal{DL}$+LOG$^{\neg\vee}$ two semantics have been defined: a FOL semantics and a NM semantics. In particular, the latter extends the stable model semantics of DATALOG$^{\neg\vee}$. According to it, $\mathcal{DL}$-predicates are still interpreted under OWA, while DATALOG-predicates are interpreted under CWA. Notice that, under both semantics, entailment can be reduced to satisfiability and, analogously, that CQ answering can be reduced to satisfiability. The problem statement of satisfiability for finite $\mathcal{DL}$+LOG$^{\neg\vee}$ KBs relies on the problem known as the *Boolean CQ/UCQ containment problem*[7] in $\mathcal{DL}$. It is shown that the decidability of reasoning in $\mathcal{DL}$+LOG$^{\neg\vee}$, thus of ground query answering, depends on the decidability of the Boolean CQ/UCQ containment problem in $\mathcal{DL}$. Currently, $\mathcal{SHIQ}$ is one of the most expressive DLs for which this problem is decidable [18].

## 1.3. Inductive Logic Programming

Inductive Logic Programming (ILP) was born at the intersection between LP and ML [43]. From LP it has borrowed the KR framework, *i.e.* HCL. From ML (more precisely, from Concept Learning) it has inherited the inferential mechanisms for induction, the most prominent of which is *generalization*. However, a distinguishing feature of ILP with respect to other forms of Concept Learning is the use of prior knowledge of the domain of interest, called *background knowledge* (BK). Therefore, induction with ILP generalizes from individual instances/observations in the presence of BK, finding valid hypotheses. *Validity* depends on the underlying *setting*. At present, there exist several formalizations of induction in ILP that can be classified according to the following two orthogonal dimensions: the *scope of induction* (discrimination vs characterization) and the *representation of observations* (ground definite clauses vs ground unit clauses). *Discriminant induction* aims at inducing hypotheses with discriminant power as required in tasks like classification. In classification, observations encompass both positive and negative examples. *Characteristic induction* is more suitable for finding regularities in a data set. This corresponds to learning from positive examples only. The second dimension affects the notion of *coverage*, *i.e.* the condition under which a hypothesis explains an observation. In *learning from entailment*, hypotheses are clausal theories, observations are ground definite clauses, and a hypothesis covers an observation if the hypothesis logically entails the observation. In *learning from interpretations*, hypotheses are clausal the-

[7] This problem was called *existential entailment* in [27].

ories, observations are Herbrand interpretations (ground unit clauses) and a hypothesis covers an observation if the observation is a model for the hypothesis.

In Concept Learning, generalization is traditionally viewed as search through a partially ordered space of inductive hypotheses [40]. According to this vision, an inductive hypothesis in ILP is a clausal theory and the induction of a single clause requires (i) structuring, (ii) searching and (iii) bounding the space of clauses [44]. First we focus on (i) by clarifying the notion of *ordering* for clauses. An ordering allows for determining which one, between two clauses, is more general than the other. Since partial orders are considered, uncomparable pairs of clauses are admitted. Given the usefulness of BK, orders have been proposed that reckon with it. Among them, *generalized subsumption* [2] is of major interest to this chapter: Given two definite clauses $C$ and $D$ standardized apart and a definite program $\mathcal{K}$, we say that $C \succeq_{\mathcal{K}} D$ iff there exists a ground substitution $\theta$ for $C$ such that (i) $head(C)\theta = head(D)\sigma$ and (ii) $\mathcal{K} \cup body(D)\sigma \models body(C)\theta$ where $\sigma$ is a Skolem substitution for $D$ with respect to $\{C\} \cup \mathcal{K}$. Generalized subsumption is also called *semantic generality* in contrast to other orders which are purely syntactic. In the general case, it is undecidable. However, for DATALOG it is decidable and admits a least general generalization. Once structured, the space of hypotheses can be searched (ii) by means of refinement operators. A *refinement operator* is a function which computes a set of specializations or generalizations of a clause according to whether a top-down or a bottom-up search is performed. The two kinds of refinement operator have been therefore called *downward* and *upward*, respectively. The definition of refinement operators presupposes the investigation of the properties of the various orderings and is usually coupled with the specification of a declarative bias for bounding the space of clauses (iii). *Bias* concerns anything which constrains the search for theories, e.g. a *language bias* specifies syntactic constraints such as *linkedness* and *connectedness* on the clauses in the search space. A definite clause $C$ is linked if each literal $l_i \in C$ is linked. A literal $l_i \in C$ is linked if at least one of its terms is linked. A term $t$ in some literal $l_i \in C$ is linked with linking-chain of length 0, if $t$ occurs in $head(C)$, and with linking-chain of length $d+1$, if some other term in $l_i$ is linked with linking-chain of length $d$. The link-depth of a term $t$ in $l_i$ is the length of the shortest linking-chain of $t$. A clause $C$ is connected if each variable occurring in $head(C)$ also occurs in $body(C)$.

## 2. ILP for Onto-Relational Rule Learning: State of the Art

Hybrid KR systems combining DLs and CLs with a tight integration scheme have very recently attracted some attention in the ILP community: [50] chooses CARIN-$\mathcal{ALN}$, [28] resorts to $\mathcal{AL}$-LOG, and [32] builds upon $\mathcal{SHIQ}$+LOG. A comparative analysis of the three is reported in Table 2. They can be considered as attempts at accommodating ontologies in ILP. Indeed, they can deal with $\mathcal{ALN}$, $\mathcal{ALC}$, and $\mathcal{SHIQ}$ ontologies respectively. We remind the reader that $\mathcal{ALN}$ and $\mathcal{ALC}$ are incomparable DLs whereas DLs in the $\mathcal{SH}$ family enrich $\mathcal{ALC}$ with further constructors.

Closely related to KR systems integrating DLs and CLs are the hybrid formalims arising from the study of many-sorted logics, where a FOL language is combined with a sort language which can be regarded as an elementary DL [14]. In this respect the study of a sorted downward refinement [15] can be also considered as a contribution to the problem of interest to this chapter. Finally, some work has been done on discovering frequent association patterns in the form of DL-safe rules [21].

### 2.1. Learning CARIN-$\mathcal{ALN}$ rules

The framework proposed in [50] focuses on discriminant induction and adopts the ILP setting of learning from interpretations. Hypotheses are represented as CARIN-$\mathcal{ALN}$ non-recursive rules with a Horn literal in the head that plays the role of target concept. The coverage relation of hypotheses against examples adapts the usual one in learning from interpretations to the case of hybrid CARIN-$\mathcal{ALN}$ BK. The generality relation between two hypotheses is defined as an extension of generalized subsumption. Procedures for testing both the coverage relation and the generality relation are based on the existential entailment algorithm of CARIN. Following [50], Kietz studies the learnability of CARIN-$\mathcal{ALN}$, thus providing a pre-processing method which enables ILP systems to learn CARIN-$\mathcal{ALN}$ rules [22].

### 2.2. Learning $\mathcal{AL}$-LOG rules

In [28], hypotheses are represented as constrained DATALOG clauses that are linked, connected (or range-restricted), and compliant with the bias of Object Identity (OI)[8]. Unlike [50], this framework is general, meaning that it is valid whatever the scope of induction is. The generality relation for one such hypothesis language is an adaptation of generalized subsumption, named $\mathcal{B}$-subsumption, to the $\mathcal{AL}$-LOG KR framework. It gives raise to a quasi-order and can be checked with a decidable procedure based on constrained SLD-resolution [35]. Coverage relations for both ILP settings of learning from interpretations and learning from entailment have been defined on the basis of query answering in $\mathcal{AL}$-LOG [31]. As opposed to [50], the framework has been implemented in an ILP system [37,30]. More precisely, an instantiation of it for the case of *characteristic induction from interpretations* has been considered. Indeed, the system supports a variant of a very popular data mining task - frequent pattern discovery - where rich prior conceptual knowledge is taken into account during the discovery process in order to find patterns at multiple levels of description granularity. The search through the space of patterns represented as unary conjunctive queries in $\mathcal{AL}$-LOG and organized according to $\mathcal{B}$-subsumption is performed by applying an ideal downward refinement operator [36].

### 2.3. Learning $\mathcal{SHIQ}$+LOG rules

The ILP framework presented in [32] represents hypotheses as $\mathcal{SHIQ}$+LOG rules and organizes them according to a generality ordering inspired by generalized subsumption. The resulting hypothesis space can be searched by means of refinement operators either top-down or bottom-up. Analogously to [28], this framework encompasses both scopes of induction but, differently from [28], it assumes the ILP setting of learning from entailment only. Both the coverage relation and the generality relation boil down to query answering in $\mathcal{SHIQ}$+LOG, thus can be reformulated as satisfiability problems. Compared to [50] and [28], this framework shows an added value which can be summarized as follows. First, it relies on a more expressive DL, *i.e.* $\mathcal{SHIQ}$. Second, it allows for inducing definitions for new DL concepts, *i.e.* rules with a $\mathcal{SHIQ}$ literal in the head.

[8]The OI bias can be considered as an extension of the UNA from the semantic level to the syntactic one of $\mathcal{AL}$-LOG. It can be the starting point for the definition of either an equational theory or a quasi-order for constrained DATALOG clauses.

**Table 2.** Three ILP frameworks suitable for learning onto-relational rules.

| | **Learning CARIN-$\mathcal{ALN}$ rules** [50] | **Learning $\mathcal{AL}$-LOG rules** [28] | **Learning $\mathcal{SHIQ}$+LOG rules** [32] |
|---|---|---|---|
| prior knowledge | CARIN-$\mathcal{ALN}$ KB | $\mathcal{AL}$-LOG KB | $\mathcal{SHIQ}$+LOG KB |
| ontology language | $\mathcal{ALN}$ | $\mathcal{ALC}$ | $\mathcal{SHIQ}$ |
| rule language | HCL | DATALOG | DATALOG |
| hypothesis language | CARIN-$\mathcal{ALN}$ non-recursive rules | $\mathcal{AL}$-LOG non-recursive rules | $\mathcal{SHIQ}$+LOG non-recursive rules |
| target predicate | Horn predicate | DATALOG predicate | $\mathcal{SHIQ}$/DATALOG predicate |
| logical setting | interpretations | interpretations/entailment | entailment |
| scope of induction | prediction | prediction/description | prediction/description |
| generality order | extension of [2] to CARIN-$\mathcal{ALN}$ | extension of [2] to $\mathcal{AL}$-LOG | extension of [2] to $\mathcal{SHIQ}$+LOG |
| coverage test | CARIN query answering | $\mathcal{AL}$-LOG query answering | $\mathcal{DL}$+LOG$^{\neg\vee}$ query answering |
| ref. operators | n.a. | downward | downward/upward |
| implementation | unknown | yes, see [30] | no |
| application | no | yes, see [37] | no |

Third, it adopts a more flexible form of integration between the DL and the CL part, *i.e.* the weakly-safe one.

The work reported in [34,29] generalizes the results of [32] to any decidable instantiation of $\mathcal{DL}$+LOG$^{\neg\vee}$. The following section illustrates how learning $\mathcal{DL}$+LOG$^{\neg}$ rules can support the evolution of ontologies.

## 3. Learning Rule-based Definitions of $\mathcal{DL}$ Concepts and Roles with ILP

In KE, Ontology Evolution is the timely adaptation of an ontology to changed business requirements, to trends in ontology instances and patterns of usage of the ontology-based application, as well as the consistent management/propagation of these changes to dependent elements [53]. As opposed to Ontology Modification, Ontology Evolution must preserve the consistency of the ontology. According to [13] one can distinguish between conceptual, specification and representation changes.

In this section we consider the conceptual changes of a $\mathcal{DL}$ ontology due to extensional knowledge (*i.e.*, facts of the instance level of the ontology) previously unknown but classified which may become available. In particular, we consider the task of defining new concepts or roles which provide the intensional counterpart of such extensional knowledge and show how this task can be reformulated as an ORL problem [33]. For example, the new facts `LONER(Joe)`, `LONER(Mary)`, and `LONER(Paul)` concerning known individuals may raise the need for having a definition of the concept `LONER` in the ontology. One such definition can be learned from these facts together with prior knowledge about `Joe`, `Mary` and `Paul`, *i.e.* facts concerning them and already available in the ontology. A crucial requirement is that the definition must be expressed as a $\mathcal{DL}$ formula or similar. In the following we provide the means for learning rule-based definitions of $\mathcal{DL}$ concepts/roles in the KR framework of $\mathcal{DL}$+LOG$^{\neg}$.

### *3.1. The learning problem*

We assume that a $\mathcal{DL}$ ontology $\Sigma = \langle \mathcal{T}, \mathcal{A} \rangle$ is integrated with a DATALOG$^{\neg}$ database $\Pi$ to form a $\mathcal{DL}$+LOG$^{\neg}$ KB $\mathcal{B}$. The problem of inducing rule-based definitions of $\mathcal{DL}$ concepts/roles that do not occur in $\mathcal{B}$ can be formalized as follows.

**Definition 1** *Given:*

- *a $\mathcal{DL}$+LOG$^\neg$ KB $\mathcal{B}$ (*background theory*)*
- *a $\mathcal{DL}$ predicate name $p$ (*target predicate*)*
- *a set $\mathcal{E} = \mathcal{E}^+ \cup \mathcal{E}^-$ of $\mathcal{DL}$ assertions that are either true or false for $p$ (*examples*)*
- *a set $\mathcal{L}$ of $\mathcal{DL}$+LOG$^\neg$ definitions for $p$ (*language of hypotheses*)*

*the problem of building a rule-based definition of $p$ is to induce a set $\mathcal{H} \subset \mathcal{L}$ (*hypothesis*) of $\mathcal{DL}$+LOG$^\neg$ rules from $\mathcal{E}$ and $\mathcal{B}$ such that:*

**Completeness** $\forall e \in \mathcal{E}^+ : \mathcal{H}$ *covers* $e$ *w.r.t.* $\mathcal{B}$
**Consistency** $\forall e \in \mathcal{E}^- : \mathcal{H}$ *does not cover* $e$ *w.r.t.* $\mathcal{B}$.

The *background theory* $\mathcal{B}$ in Definition 1 can be split into an intensional part $\mathcal{K}$ (*i.e.*, the TBox $\mathcal{T}$ plus $IDB(\Pi)$) and an extensional part $\mathcal{F}$ (*i.e.*, the ABox $\mathcal{A}$ plus $EDB(\Pi)$). Also we denote by $P_\mathcal{C}(\mathcal{B})$, $P_\mathcal{R}(\mathcal{B})$, and $P_\mathrm{D}(\mathcal{B})$ the sets of concept, role and DATALOG predicate names occurring in $\mathcal{B}$, respectively. Note that $p \notin P_\mathcal{C}(\mathcal{B}) \cup P_\mathcal{R}(\mathcal{B})$.

**Example 1** *Suppose we have a $\mathcal{DL}$+LOG$^\neg$ KB $\mathcal{B}$ (adapted from [48]) built upon the alphabets $P_\mathcal{C}(\mathcal{B}) = \{$`RICH/1`, `UNMARRIED/1`$\}$, $P_\mathcal{R}(\mathcal{B}) = \{$`WANTS-TO-MARRY/2`, `LOVES/2`$\}$, and $P_\mathrm{D}(\mathcal{B}) = \{$`famous/1`, `scientist/1`, `meets/3`$\}$ and consisting of the following intensional knowledge $\mathcal{K}$:*

$[A1]$ `RICH` $\sqcap$ `UNMARRIED` $\sqsubseteq \exists$ `WANTS-TO-MARRY`$^-$ $.\top$
$[A2]$ `WANTS-TO-MARRY` $\sqsubseteq$ `LOVES`
$[R1]$ `RICH(X)` $\leftarrow$ `famous(X)`, $\neg$`scientist(X)`
$[R2]$ `happy(X)` $\leftarrow$ `famous(X)`, `WANTS-TO-MARRY(Y,X)`

*and the following extensional knowledge $\mathcal{F}$:*

```
UNMARRIED(Mary)
UNMARRIED(Joe)
famous(Mary)
famous(Paul)
famous(Joe)
scientist(Joe)
meets(Mary,Paul,Italy)
meets(Mary,Joe,Germany)
meets(Joe,Mary,Italy)
```

*that concerns the individuals* `Mary`, `Joe`, `Paul`, `Italy`, *and* `Germany`.

The *hypothesis language* $\mathcal{L}$ in Definition 1 is given as a set of declarative bias constraints. It allows for the generation of $\mathcal{DL}$+LOG$^\neg$ rules starting from three disjoint alphabets $P_\mathcal{C}(\mathcal{L}) \subseteq P_\mathcal{C}(\mathcal{B})$, $P_\mathcal{R}(\mathcal{L}) \subseteq P_\mathcal{R}(\mathcal{B})$, and $P_\mathrm{D}(\mathcal{L}) \subseteq P_\mathrm{D}(\mathcal{B})$. Also we distinguish between $P^+_\mathrm{D}(\mathcal{L})$ and $P^-_\mathrm{D}(\mathcal{L})$ in order to specify which DATALOG predicates can occur in positive and negative literals, respectively. More precisely, we consider $\mathcal{DL}$+LOG$^\neg$ rules of the form

$$p(\vec{X}) \leftarrow r_1(\vec{Y_1}), \ldots, r_m(\vec{Y_m}), s_1(\vec{Z_1}), \ldots, s_k(\vec{Z_k}), \neg u_1(\vec{W_1}), \ldots, \neg u_q(\vec{W_q}) \qquad (1)$$

where $m, k, q \geq 0$, $p(\vec{X})$ and each $r_j(\vec{Y_j})$, $s_l(\vec{Z_l})$, $u_t(\vec{W_t})$ is an atom with $r_j \in P_{\mathrm{D}}^{+}(\mathcal{L})$, $s_l \in P_{\mathcal{C}}(\mathcal{L}) \cup P_{\mathcal{R}}(\mathcal{L})$, and $u_t \in P_{\mathrm{D}}^{-}(\mathcal{L})$. The admissible rules must be compliant with the following restrictions:

**DATALOG-safeness** every variable occurring in (1) must appear in at least one of the atoms $r_1(\vec{Y_1}), \ldots, r_m(\vec{Y_m}), s_1(\vec{Z_1}), \ldots, s_k(\vec{Z_k})$;
**weak $\mathcal{DL}$-safeness** every head variable of (1) must appear in at least one of the atoms $r_1(\vec{Y_1}), \ldots, r_m(\vec{Y_m})$.

which also guarantee that the conditions of linkedness and connectedness, usually assumed in ILP, are satisfied.

**Example 2** *Suppose that the target predicate is the $\mathcal{DL}$ concept* `LONER`*. If $\mathcal{L}^{\texttt{LONER}}$ is defined over $P_D^{+}(\mathcal{L}^{\texttt{LONER}}) \cup P_D^{-}(\mathcal{L}^{\texttt{LONER}}) \cup P_{\mathcal{C}}(\mathcal{L}^{\texttt{LONER}}) = \{\texttt{famous/1}\} \cup \{\texttt{happy/1}\} \cup \{\texttt{RICH/1}, \texttt{UNMARRIED/1}\}$, then the following $\mathcal{DL}$+LOG$^{\neg}$ rules*

$h_1^{\texttt{LONER}}$ `LONER(X)` $\leftarrow$ `famous(X)`
$h_2^{\texttt{LONER}}$ `LONER(X)` $\leftarrow$ `famous(X), UNMARRIED(X)`
$h_3^{\texttt{LONER}}$ `LONER(X)` $\leftarrow$ `famous(X),` $\neg$`happy(X)`

*belong to $\mathcal{L}^{\texttt{LONER}}$ and represent hypotheses of a definition for* `LONER`*.*

**Example 3** *Suppose now that the $\mathcal{DL}$ role* `LIKES` *is the target predicate and the set $P_D^{+}(\mathcal{L}^{\texttt{LIKES}}) \cup P_{\mathcal{C}}(\mathcal{L}^{\texttt{LIKES}}) \cup P_{\mathcal{R}}(\mathcal{L}^{\texttt{LIKES}}) = \{\texttt{happy/1}, \texttt{meets/3}\} \cup \{\texttt{RICH/1}\} \cup \{\texttt{LOVES/2}, \texttt{WANTS-TO-MARRY/2}\}$ provides the building blocks for the language $\mathcal{L}^{\texttt{LIKES}}$. The following $\mathcal{DL}$+LOG$^{\neg}$ rules*

$h_1^{\texttt{LIKES}}$ `LIKES(X,Y)` $\leftarrow$ `meets(X,Z,Y)`
$h_2^{\texttt{LIKES}}$ `LIKES(X,Y)` $\leftarrow$ `meets(X,Z,Y), happy(X)`
$h_3^{\texttt{LIKES}}$ `LIKES(X,Y)` $\leftarrow$ `meets(X,Z,Y), RICH(Z)`

*belonging to $\mathcal{L}^{\texttt{LIKES}}$ can be considered hypotheses of a definition for* `LIKES`*.*

The *set $\mathcal{E}$ of examples* in Definition 1 contains assertions of the kind $p(\vec{a_i})$ where $p$ is the target predicate and $\vec{a_i}$ is a tuple of individuals occurring in the ABox $\mathcal{A}$. Note that, when $p$ is a role name, the tuple $\vec{a_i}$ is a pair $< a_i^1, a_i^2 >$ of individuals. We assume $\mathcal{B} \cap \mathcal{E} = \emptyset$. However, a possibly incomplete description of each $e_i \in \mathcal{E}$ is in $\mathcal{B}$.

**Example 4** *With reference to Example 2, suppose that the following concept assertions:*

$e_1^{\texttt{LONER}}$ `LONER(Mary)`
$e_2^{\texttt{LONER}}$ `LONER(Joe)`
$e_3^{\texttt{LONER}}$ `LONER(Paul)`

*are examples for the target predicate* `LONER`*.*

**Example 5** *With reference to Example 3, the following role assertions:*

$e_1^{\texttt{LIKES}}$ `LIKES(Mary,Italy)`
$e_2^{\texttt{LIKES}}$ `LIKES(Mary,Germany)`
$e_3^{\texttt{LIKES}}$ `LIKES(Joe,Italy)`

*can be assumed as examples for the target predicate* `LIKES`*.*

### 3.2. The ingredients for an ILP solution

In order to solve the learning problem in hand with the ILP methodological approach , the language $\mathcal{L}$ of hypotheses needs to be equipped with (i) a *coverage relation* which defines the mappings from $\mathcal{L}$ to the set $\mathcal{E}$ of examples, and (ii) a *generality order* $\succeq$ such that $(\mathcal{L}, \succeq)$ is a search space.

The definition of a *coverage relation* depends on the representation choice for examples. The normal ILP setting is the most appropriate to the learning problem in hand and can be extended to the $\mathcal{DL}$+LOG$^\neg$ framework depicted in Definition 1 as follows.

**Definition 2** *We say that a rule $h \in \mathcal{L}$* covers *(does not cover, resp.) an example $e_i = p(\vec{a_i}) \in \mathcal{E}$ w.r.t. a background theory $\mathcal{B}$ iff $\mathcal{B} \cup h \models p(\vec{a_i})$ ($\mathcal{B} \cup h \not\models p(\vec{a_i})$, resp.).*

Note that the coverage test can be reduced to query answering w.r.t. a $\mathcal{DL}$+LOG$^{\neg\vee}$ KB, which in turn can be reformulated as a satisfiability problem of the KB.

**Example 6** *With reference to Example 2 and 4, the rule $h_1^{LONER}$ covers the example $e_1^{LONER}$ because all NM-models for $\mathcal{B}' = \mathcal{B} \cup h_1^{LONER}$ do satisfy $famous(Mary)$. It covers also $e_2^{LONER}$ and $e_3^{LONER}$ for analogous reasons. The rule $h_2^{LONER}$ covers only $e_1^{LONER}$ and $e_2^{LONER}$ whereas $h_3^{LONER}$ covers $e_2^{LONER}$ and $e_3^{LONER}$.*

**Example 7** *With reference to Example 3 and 5, the rule $h_1^{LIKES}$ covers the example $e_1^{LIKES}$ because all NM-models for $\mathcal{B}' = \mathcal{B} \cup h_1^{LIKES}$ do satisfy $meets(Mary, Z, Italy)$. It covers also $e_2^{LIKES}$ and $e_3^{LIKES}$ for analogous reasons. The rule $h_2^{LIKES}$ covers only $e_1^{LIKES}$ and $e_2^{LIKES}$ whereas $h_3^{LIKES}$ covers only $e_1^{LIKES}$ and $e_3^{LIKES}$.*

The definition of a *generality order* for hypotheses in $\mathcal{L}$ must consider the peculiarities of the chosen $\mathcal{L}$. Generalized subsumption, subsequently extended in [52] to deal with NAF literals, is suitable for the problem in hand and can be adapted to the case of $\mathcal{DL}$+LOG$^\neg$ rules. In the following we provide a characterization of the resulting generality order, denoted by $\succeq^\neg_\mathcal{K}$, that relies on the reasoning tasks known for $\mathcal{DL}$+LOG$^{\neg\vee}$and from which a test procedure can be derived.

**Definition 3** *Let $h_1, h_2 \in \mathcal{L}$ be two $\mathcal{DL}$+LOG$^\neg$ rules standardized apart, $\mathcal{K}$ a $\mathcal{DL}$+LOG$^\neg$ KB, and $\sigma$ a Skolem substitution for $h_2$ with respect to $\{h_1\} \cup \mathcal{K}$. We say that $h_1$ is* more general than *$h_2$ w.r.t. $\mathcal{K}$, denoted by $h_1 \succeq^\neg_\mathcal{K} h_2$, iff there exists a ground substitution $\theta$ for $h_1$ such that (i) $head(h_1)\theta = head(h_2)\sigma$ and (ii) $\mathcal{K} \cup body(h_2)\sigma \models body(h_1)\theta$. We say that $h_1$ is* strictly more general than *$h_2$ w.r.t. $\mathcal{K}$, denoted by $h_1 \succ^\neg_\mathcal{K} h_2$, iff $h_1 \succeq^\neg_\mathcal{K} h_2$ and $h_2 \not\succeq^\neg_\mathcal{K} h_1$. We say that $h_1$ is* equivalent to *$h_2$ w.r.t. $\mathcal{K}$, denoted by $h_1 \equiv^\neg_\mathcal{K} h_2$, iff $h_1 \succeq^\neg_\mathcal{K} h_2$ and $h_2 \succeq^\neg_\mathcal{K} h_1$.*

**Example 8** *Let us consider the rules reported in Example 2 up to variable renaming:*

$h_1^{LONER}$ $\quad LONER(A) \leftarrow famous(A)$
$h_2^{LONER}$ $\quad LONER(X) \leftarrow famous(X), UNMARRIED(X)$

*In order to check whether $h_1^{LONER} \succeq^\neg_\mathcal{K} h_2^{LONER}$ holds, let $\sigma = \{X/a\}$ a Skolem substitution for $h_2^{LONER}$ with respect to $\mathcal{K} \cup h_1^{LONER}$ and $\theta = \{A/a\}$ a ground substitution for $h_1^{LONER}$. The condition (i) is immediately verified. The condition*

$$\textit{(ii)}\ \mathcal{K} \cup \{famous(a), UNMARRIED(a)\} \models \{famous(a)\}$$

*is a ground query answering problem in $\mathcal{DL}$+LOG$^\neg$. It can be easily proved that all NM-models for $\mathcal{K} \cup \{famous(a), UNMARRIED(a)\}$ satisfy $famous(a)$. Thus, it is the case that $h_1^{LONER} \succeq_{\mathcal{K}}^{\neg} h_2^{LONER}$. The viceversa does not hold. Also, $h_1^{LONER} \succ_{\mathcal{K}}^{\neg} h_3^{LONER}$ and $h_3^{LONER}$ is incomparable with $h_2^{LONER}$.*

**Example 9** *With reference to Example 3, it can be proved that $h_1^{LIKES} \succ_{\mathcal{K}}^{\neg} h_2^{LIKES}$ and $h_1^{LIKES} \succ_{\mathcal{K}}^{\neg} h_3^{LIKES}$. Conversely, the rules $h_2^{LIKES}$ and $h_3^{LIKES}$ are incomparable. Note that*

$h_4^{LIKES}$ $\quad LIKES(X,Y) \leftarrow meets(X,Z,Y),\ LOVES(X,Z)$
$h_5^{LIKES}$ $\quad LIKES(X,Y) \leftarrow meets(X,Z,Y),\ WANTS\text{-}TO\text{-}MARRY(X,Z)$

*also belong to $\mathcal{L}^{LIKES}$. It can be proved that $h_1^{LIKES} \succ_{\mathcal{K}}^{\neg} h_4^{LIKES}$, $h_1^{LIKES} \succ_{\mathcal{K}}^{\neg} h_5^{LIKES}$, and $h_4^{LIKES} \succ_{\mathcal{K}}^{\neg} h_5^{LIKES}$.*

Note that the decidability of $\succ_{\mathcal{K}}^{\neg}$ follows from the decidability of $\mathcal{DL}$+LOG$^\neg$. Also it can be proved that $\succ_{\mathcal{K}}^{\neg}$ is a quasi-order (*i.e.*, it is a reflexive and transitive relation) for $\mathcal{DL}$+LOG$^\neg$ rules, therefore the space $(\mathcal{L}, \succ_{\mathcal{K}}^{\neg})$ can be searched by refinement operators like the following one able to traverse the hypothesis space top down.

**Definition 4** *Let $\mathcal{L}$ be a $\mathcal{DL}$+LOG$^\neg$ hypothesis language built out of the three finite and disjoint alphabets $P_{\mathcal{C}}(\mathcal{L})$, $P_{\mathcal{R}}(\mathcal{L})$, and $P_{\mathrm{D}}^{+}(\mathcal{L}) \cup P_{\mathrm{D}}^{-}(\mathcal{L})$. We define a* downward refinement operator $\rho^{\mathrm{OR}}$ *for $(\mathcal{L}, \succeq_{\mathcal{K}}^{\neg})$ such that, for each $h \in \mathcal{L}$, the set $\rho^{\mathrm{OR}}(h)$ contains all $h' \in \mathcal{L}$ that can be obtained from $h$ by applying one of the following refinement rules:*

$\langle AddDataLit\_B^{+} \rangle$ $body(h') = body(h) \cup \{r_{m+1}(\vec{Y_{m+1}})\}$ *if*

1. $r_{m+1} \in P_{\mathrm{D}}^{+}(\mathcal{L})$
2. $r_{m+1}(\vec{Y_{m+1}}) \notin body(h)$
3. $var(head(h)) \subseteq var(body(h'))$

$\langle AddOntoLit\_B \rangle$ $body(h') = body(h) \cup \{s_{k+1}(\vec{Z_{k+1}})\}$ *if*

1. $s_{k+1} \in P_{\mathcal{C}}(\mathcal{L}) \cup P_{\mathcal{R}}(\mathcal{L})$
2. *it does not exist any* $s_l(\vec{Z_l}) \in body(h)$ *such that* $s_{k+1} \sqsubseteq s_l$
3. $var(head(h)) \subseteq var(body(h'))$

$\langle SpecOntoLit\_B \rangle$ $body(h') = (body(h) \setminus \{s_l(\vec{Z_l})\}) \cup s'_l(\vec{Z_l})$ *if*

1. $s'_l \in P_{\mathcal{C}}(\mathcal{L}) \cup P_{\mathcal{R}}(\mathcal{L})$
2. $s'_l \sqsubseteq s_l$

$\langle AddDataLit\_B^{-} \rangle$ $body(h') = body(h) \cup \{\neg u_{q+1}(\vec{W_{q+1}})\}$ *if*

1. $u_{q+1} \in P_{\mathrm{D}}^{-}(\mathcal{L})$
2. $u_{q+1}(\vec{W_{q+1}}) \notin body(h)$
3. $\vec{W_{q+1}} \subset var(body^{+}(h))$

**function** OR-FOIL($\mathcal{B}$, $p$, $\mathcal{E}^+$, $\mathcal{E}^-$, $\mathcal{L}$): $\mathcal{H}$
1. $\mathcal{H} := \emptyset$
2. **while** $\mathcal{E}^+ \neq \emptyset$ **do**
3. $\quad h := \{p(\vec{X}) \leftarrow\}$;
4. $\quad \mathcal{E}^-_h := \mathcal{E}^-$
5. $\quad$ **while** $\mathcal{E}^-_h \neq \emptyset$ **do**
6. $\quad\quad \mathcal{Q} := \{h' \in \mathcal{L} | h' \in \rho^{\mathrm{OR}}(h)\}$;
7. $\quad\quad h :=$ OR-FOIL-CHOOSEBEST($\mathcal{Q}$);
8. $\quad\quad \mathcal{E}^-_h := \{e \in \mathcal{E}^- | \mathcal{B} \cup h \models e\}$;
9. $\quad$ **endwhile**
10. $\quad \mathcal{H} := \mathcal{H} \cup \{h\}$;
11. $\quad \mathcal{E}^+_h := \{e \in \mathcal{E}^+ | \mathcal{B} \cup h \models e\}$;
12. $\quad \mathcal{E}^+ := \mathcal{E}^+ \setminus \mathcal{E}^+_h$
13. **endwhile**
14. **return** $\mathcal{H}$

**Figure 1.** OR-FOIL: A FOIL-like algorithm for learning onto-relational rules

All the rules of $\rho^{\mathrm{OR}}$ are correct, *i.e.* the $h'$'s obtained by applying any of the rules of $\rho^{\mathrm{OR}}$ to $h \in \mathcal{L}$ are such that $h \succ^{-}_{\mathcal{K}} h'$. This can be proved intuitively by observing that they act only on $body(h)$. Thus condition (i) of Definition 3 is satisfied. Furthermore, it is straightforward to notice that the application of any of the rules of $\rho^{\mathrm{OR}}$ to $h$ reduces the number of models of $h$. In particular, as for $\langle SpecOntoLit\_B \rangle$, this intuition follows from the semantics of DLs. So condition (ii) also is fulfilled.

**Example 10** *With reference to Example 2, applying* $\langle AddDataLit\_B^+ \rangle$ *to*

$h_0^{LONER}$ `LONER(X)` $\leftarrow$

*produces* $h_1^{LONER}$ *which can be further specialized by means of* $\langle AddOntoLit\_B \rangle$ *and* $\langle AddDataLit\_B^- \rangle$. *Note that no other refinement rule can be applied to* $h_1^{LONER}$ *and that* $h_2^{LONER}$ *and* $h_3^{LONER}$ *are among the refinements of* $h_1^{LONER}$.

**Example 11** *With reference to Example 3, applying* $\langle AddDataLit\_B^+ \rangle$ *to*

$h_0^{LIKES}$ `LIKES(X,Y)` $\leftarrow$

*produces* $h_1^{LIKES}$ *which can be further specialized into* $h_2^{LIKES}$, $h_3^{LIKES}$, $h_4^{LIKES}$ *and* $h_5^{LIKES}$ *by means of* $\langle AddDataLit\_B \rangle$ *and* $\langle AddOntoLit\_B \rangle$. *Note that no other refinement rule can be applied to* $h_1^{LIKES}$ *and that* $h_5^{LIKES}$ *can be also obtained as refinement from* $h_4^{LIKES}$ *via* $\langle SpecOntoLit\_B \rangle$.

### *3.3. An ILP algorithm*

The ingredients identified in the previous section are the starting point for the definition of ILP algorithms. Figure 1 reports the main procedure of a FOIL-like algorithm, named OR-FOIL, for learning onto-relational rules. In OR-FOIL, analogously to FOIL[9],

[9] FOIL is a popular ILP algorithm for learning sets of rules to be used as a classifier [45].

the outer loop (steps 2-12) corresponds to a variant of the sequential covering algorithm, *i.e.*, it learns new rules one at a time, removing the positive examples covered by the latest rule before attempting to learn the next rule (steps 11-12). The hypothesis space search performed by OR-FOIL is best understood by viewing it hierarchically. Each iteration through the outer loop (steps 2-13) adds a new rule to its disjunctive hypothesis $\mathcal{H}$. The effect of each new rule is to generate the current disjunctive hypothesis (*i.e.*, to increase the number of instances it classifies as positive), by adding a new disjunct. Viewed at this level, the search is a bottom-up search through the space of hypotheses, beginning with the most specific empty disjunction (step 1) and terminating when the hypothesis is sufficiently general to cover all positive training examples (step 13). The inner loop (steps 5-9) performs a more fine-grained search to determine the exact definition of each new rule. This loop searches a second hypothesis space, consisting of conjunctions of literals, to find a conjunction that will form the body of the new rule. Within this space, it conducts a top-down, hill-climbing search, beginning with the most general preconditions possible (step 3), then refining the rule (step 6) until it avoids all negative examples. To select the most promising specialization from the candidates generated at each iteration, OR-FOIL-CHOOSEBEST (called at step 7) considers the performance of each candidate over $\mathcal{E}$ and chooses the one which maximizes the *information gain*. This measure is computed according to the following formula

$$\text{GAIN}(h', h) = p * (log_2(cf(h')) - log_2(cf(h))) \,, \tag{2}$$

where $p$ is the number of distinct variable bindings with which positive examples covered by the rule $h$ are still covered by $h'$ and $cf()$ is the confidence degree. Thus, the gain is positive iff $h'$ is more informative in the sense of Shannon's information theory (*i.e.* iff the confidence degree increases). If there are some literals to add which increase the confidence degree, the information gain tends to favor the literals that offer the best compromise between the confidence degree and the number of examples covered.

One may think to use the confidence degree defined for $\mathcal{DL}$-FOIL(cf. Lehmann et. al [25] in this volume) which takes OWA into account. Indeed, many individuals may be available which can not be classified as instances of the target concept nor of its negation. This requires a different setting able to deal with unlabeled individuals.

**Example 12** *With reference to Example 10 and Example 6, we suppose that*

$$\mathcal{E}^+ = \{e_1^{LONER}, e_2^{LONER}\}$$
$$\mathcal{E}^- = \{e_3^{LONER}\}$$

*The outer loop of* OR-FOIL *starts from $h_0^{LONER}$ which is further refined through the iterations of the inner loop, more precisely it is first specialized into $h_1^{LONER}$ which in turn, since it covers negative examples, is then specialized into $h_2^{LONER}$ and $h_3^{LONER}$ out of which the rule $h_3^{LONER}$ is added to $\mathcal{H}^{LONER}$ the hypothesis because it does not cover negative examples. At this point the algorithm stops because $\mathcal{H}^{LONER}$ covers both positive examples.*

**Example 13** *Following Example 11 and Example 7, we assume that $\mathcal{E}^+ = \{e_1^{LIKES}, e_3^{LIKES}\}$ and $\mathcal{E}^- = \{e_2^{LIKES}\}$. At the end of the first iteration, $h_3^{LIKES}$ is included into $\mathcal{H}^{LIKES}$ since it does not cover negative examples but only one positive example.*

## 4. Final Remarks and Directions of Research

Building rules within ontologies poses several challenges not only to KR researchers investigating suitable hybrid DL-CL formalisms but also to the ML community which has been historically interested in application areas where the knowledge acquisition bottleneck is particularly severe. In particular, ORL may open up new opportunities for KE because it will make systems available to support the knowledge engineer in her most demanding task, *i.e.* defining rules that extend or complement an ontology. Thus, ORL may produce time and cost savings in KE. In this chapter, we have revised the ML literature addressing the problem of learning onto-relational rules. Very few ILP works have been found that propose a solution to this problem [50,28,32]. They adopt CARIN-$\mathcal{ALN}$, $\mathcal{AL}$-LOG and $\mathcal{SHIQ}$+LOG as KR framework, respectively. Note that matching Table 2 against Table 1 one may figure out what is the state-of-the-art and what are the directions of research on onto-relational rules from the ML viewpoint. Also he/she can get suggestions on what is the most appropriate among these ILP frameworks to be implemented for a certain intended application. The specific solution illustrated in Section 3 takes advantage from an augmented expressive power thanks to the chosen $\mathcal{DL}$+LOG$^{\neg\vee}$ instantiation [29]. It supports the evolution of ontologies with the creation of a concept/role, change operations which both boil down to the addition of new rules to the input KB.

From the comparative analysis of the ILP frameworks reviewed in Section 2, a common feature emerges: All proposals resort to Buntine's generalized subsumption and extend it in a non-trivial way. This choice is due to the fact that, among the semantic generality orders in ILP, generalized subsumption applies only to definite clauses, therefore suits well the hypothesis language in all three frameworks. Following these guidelines, new ILP frameworks can be designed to deal with more or differently expressive hybrid DL-CL languages according to the DL chosen (*e.g.*, learning CARIN-$\mathcal{ALCNR}$ rules), or the clausal language chosen (*e.g.*, learning recursive CARIN rules), or the integration scheme (*e.g.*, learning CARIN rules with $\mathcal{DL}$-literals in the head). An important requirement will be the definition of a *semantic* generality relation for hypotheses to take into account the background knowledge. Of course, generalized subsumption may turn out to be not suitable for all cases, *e.g.* for the case of learning $\mathcal{DL}$+LOG$^{\vee}$ rules [29]. Also it would be interesting to investigate how the nature of rules (*i.e.*, the intended context of usage) may impact the learning process as for the scope of induction and other variables in the learning problem statement. For example, the problem of learning $\mathcal{AL}$-LOG rules for classification purposes differ greatly from the apparently similar learning problem faced in [37]. Finally, it is worthy to consider hybrid KR formalisms with loose and full integration scheme.

Besides theoretical issues, most future work will have to be devoted to implementation and application. When moving to practice, issues like efficiency and scalability become of paramount importance. These concerns may drive the attention of ILP research towards less expressive hybrid KR frameworks in order to gain in tractability, *e.g.* instantiations of $\mathcal{DL}$+LOG$^{\neg\vee}$ with DL-Lite [4]. Applications can come out of some of the many use cases for Semantic Web rules specified by the RIF W3C Working Group.

## References


[1] Alexander Borgida. On the relative expressiveness of description logics and predicate logics. *Artificial Intelligence*, 82(1–2):353–367, 1996.

[2] Wray Buntine. Generalized subsumption and its application to induction and redundancy. *Artificial Intelligence*, 36(2):149–176, 1988.

[3] John Cabral, Robert C. Kahlert, Cynthia Matuszek, Michael J. Witbrock, and Brett Summers. Converting semantic meta-knowledge into inductive bias. In Stefan Kramer and Bernhard Pfahringer, editors, *Inductive Logic Programming, 15th International Conference, ILP 2005, Bonn, Germany, August 10-13, 2005, Proceedings*, volume 3625 of *Lecture Notes in Computer Science*, pages 38–50. Springer, 2005.

[4] Diego Calvanese, Maurizio Lenzerini, Riccardo Rosati, and Guido Vetere. DL-Lite: Practical Reasoning for Rich DLs. In Volker Haarslev and Ralph Möller, editors, *Proc. of the 2004 Int. Workshop on Description Logics*, volume 104 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2004.

[5] Stefano Ceri, Georg Gottlob, and Letizia Tanca. What you always wanted to know about datalog (and never dared to ask). *IEEE Transactions on Knowledge and Data Engineering*, 1(1):146–166, 1989.

[6] Stefano Ceri, Georg Gottlob, and Letizia Tanca. *Logic Programming and Databases*. Springer, 1990.

[7] Luc De Raedt. *Logical and Relational Learning*. Springer, 2008.

[8] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf. $\mathcal{AL}$-log: Integrating Datalog and Description Logics. *Journal of Intelligent Information Systems*, 10(3):227–252, 1998.

[9] Wlodzimierz Drabent, Thomas Eiter, Giovambattista Ianni, Thomas Krennwallner, Thomas Lukasiewicz, and Jan Maluszynski. Hybrid Reasoning with Rules and Ontologies. In François Bry and Jan Maluszynski, editors, *Semantic Techniques for the Web, The REWERSE Perspective*, volume 5500 of *Lecture Notes in Computer Science*, pages 1–49. Springer, 2009.

[10] Thomas Eiter, Georg Gottlob, and Heikki Mannila. Disjunctive DATALOG. *ACM Transactions on Database Systems*, 22(3):364–418, 1997.

[11] Thomas Eiter, Giovambattista Ianni, Thomas Lukasiewicz, Roman Schindlauer, and Hans Tompits. Combining answer set programming with description logics for the semantic web. *Artificial Intelligence*, 172(12-13):1495–1539, 2008.

[12] Thomas Eiter, Giovambattista Ianni, Roman Schindlauer, and Hans Tompits. A uniform integration of higher-order reasoning and external evaluations in answer-set programming. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30-August 5, 2005*, pages 90–96, 2005.

[13] Natalya Fridman Noy and Michel C. A. Klein. Ontology evolution: Not the same as schema evolution. *Knowledge and Information Systems*, 6(4):428–440, 2004.

[14] Alan M. Frisch. The substitutional framework for sorted deduction: Fundamental results on hybrid reasoning. *Artificial Intelligence*, 49:161–198, 1991.

[15] Alan M. Frisch. Sorted downward refinement: Building background knowledge into a refinement operator for inductive logic programming. In Saso Džeroski and Peter A. Flach, editors, *Inductive Logic Programming, 9th International Workshop, ILP-99, Bled, Slovenia, June 24-27, 1999, Proceedings*, volume 1634 of *Lecture Notes in Computer Science*, pages 104–115. Springer, 1999.

[16] Alan M. Frisch and Anthony G. Cohn. Thoughts and afterthoughts on the 1988 workshop on principles of hybrid reasoning. *AI Magazine*, 11(5):84–87, 1991.

[17] Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9(3/4):365–386, 1991.

[18] Birte Glimm, Ian Horrocks, Carsten Lutz, and Ulrike Sattler. Conjunctive query answering for the description logic $\mathcal{SHIQ}$. *Journal of Artificial Intelligence Research*, 31:151–198, 2008.

[19] François Goasdoué, Véronique Lattès, and Marie-Christine Rousset. The Use of CARIN Language and Algorithms for Information Integration: The PICSEL System. *International Journal of Cooperative Information Systems*, 9(4):383–401, 2000.

[20] Ian Horrocks, Jürgen Angele, Stefan Decker, Michael Kifer, Benjamin N. Grosof, and Gerd Wagner. Where are the rules? *IEEE Intelligent Systems*, 18:76–83, 2003.

[21] Joanna Józefowska, Agnieszka Lawrynowicz, and Tomasz Lukaszewski. The role of semantics in mining frequent patterns from knowledge bases in description logics with rules. *Theory and Practice of Logic Programming*, 10(3):251–289, 2010.

[22] Jörg-Uwe Kietz. Learnability of description logic programs. In Stan Matwin and Claude Sammut, editors, *Inductive Logic Programming, 12th International Conference, ILP 2002, Sydney, Australia, July 9-*

*11, 2002. Revised Papers*, volume 2583 of *Lecture Notes in Computer Science*, pages 117–132. Springer, 2003.

[23] Markus Krötzsch, Frantisek Simančík, and Ian Horrocks. A description logic primer. In Johanna Völker and Jens Lehmann, editors, *Perspectives of Ontology Learning*, Studies on the Semantic Web. AKA Heidelberg / IOS Press, 2012.

[24] Agnieszka Ławrynowicz and Volker Tresp. Introducing machine learning. In Johanna Völker and Jens Lehmann, editors, *Perspectives of Ontology Learning*, Studies on the Semantic Web. AKA Heidelberg / IOS Press, 2012.

[25] Jens Lehmann, Nicola Fanizzi, and Claudia d'Amato. Concept learning. In Johanna Völker and Jens Lehmann, editors, *Perspectives of Ontology Learning*, Studies on the Semantic Web. AKA Heidelberg / IOS Press, 2012.

[26] Douglas B. Lenat, Ramanathan V. Guha, Karen Pittman, Dexter Pratt, and Mary Shepherd. Cyc: Toward programs with common sense. *Communications of the ACM*, 33(8):30–49, 1990.

[27] Alon Y. Levy and Marie-Christine Rousset. Combining Horn rules and description logics in CARIN. *Artificial Intelligence*, 104:165–209, 1998.

[28] Francesca A. Lisi. Building Rules on Top of Ontologies for the Semantic Web with Inductive Logic Programming. *Theory and Practice of Logic Programming*, 8(03):271–300, 2008.

[29] Francesca A. Lisi. Inductive Logic Programming in Databases: From Datalog to $\mathcal{DL}$+log. *Theory and Practice of Logic Programming*, 10(3):331–359, 2010.

[30] Francesca A. Lisi. $\mathcal{AL}$-QUIN: An Onto-Relational Learning System for Semantic Web Mining. *International Journal on Semantic Web and Information Systems*, 7(3):1–22, 2011.

[31] Francesca A. Lisi and Floriana Esposito. Efficient Evaluation of Candidate Hypotheses in $\mathcal{AL}$-log. In Rui Camacho, Ross D. King, and Ashwin Srinivasan, editors, *Inductive Logic Programming, 14th International Conference, ILP 2004, Porto, Portugal, September 6-8, 2004, Proceedings*, volume 3194 of *Lecture Notes in Computer Science*, pages 216–233. Springer, 2004.

[32] Francesca A. Lisi and Floriana Esposito. Foundations of Onto-Relational Learning. In Filip Železný and Nada Lavrač, editors, *Inductive Logic Programming, 18th International Conference, ILP 2008, Prague, Czech Republic, September 10-12, 2008, Proceedings*, volume 5194 of *Lecture Notes in Computer Science*, pages 158–175. Springer, 2008.

[33] Francesca A. Lisi and Floriana Esposito. Learning $\mathcal{SHIQ}$+log Rules for Ontology Evolution. In A. Gangemi, J. Keizer, V. Presutti, and H. Stoermer, editors, *Semantic Web Applications and Perspectives (SWAP2008)*, volume 426 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.

[34] Francesca A. Lisi and Floriana Esposito. Nonmonotonic Onto-Relational Learning. In Luc De Raedt, editor, *Inductive Logic Programming, 19th International Conference, ILP 2009, Leuven, Belgium, July 02-04, 2009. Revised Papers*, volume 5989 of *Lecture Notes in Computer Science*, pages 88–95, 2010.

[35] Francesca A. Lisi and Donato Malerba. Bridging the Gap between Horn Clausal Logic and Description Logics in Inductive Learning. In Amedeo Cappelli and Franco Turini, editors, *AI*IA 2003: Advances in Artificial Intelligence, 8th Congress of the Italian Association for Artificial Intelligence, Pisa, Italy, September 23-26, 2003, Proceedings*, volume 2829 of *Lecture Notes in Computer Science*, pages 49–60. Springer, 2003.

[36] Francesca A. Lisi and Donato Malerba. Ideal Refinement of Descriptions in $\mathcal{AL}$-log. In Tamás Horváth and Akihiro Yamamoto, editors, *Inductive Logic Programming: 13th International Conference, ILP 2003, Szeged, Hungary, September 29-October 1, 2003, Proceedings*, volume 2835 of *Lecture Notes in Computer Science*, pages 215–232. Springer, 2003.

[37] Francesca A. Lisi and Donato Malerba. Inducing Multi-Level Association Rules from Multiple Relations. *Machine Learning*, 55:175–210, 2004.

[38] John W. Lloyd. *Foundations of Logic Programming*. Springer, 2nd edition, 1987.

[39] Victor W. Marek and Miroslaw Truszczynski. Stable models and an alternative logic programming paradigm. In Krzysztof R. Apt, Victor W. Marek, Miroslaw Truszczynski, and David S. Warren, editors, *The Logic Programming Paradigm: a 25-Year Perspective*, pages 169–181. Springer, 1999.

[40] Tom M. Mitchell. Generalization as search. *Artificial Intelligence*, 18:203–226, 1982.

[41] Boris Motik and Riccardo Rosati. Reconciling description logics and rules. *J. ACM*, 57(5), 2010.

[42] Boris Motik, Ulrike Sattler, and Rudi Studer. Query Answering for OWL-DL with Rules. *Journal on Web Semantics*, 3(1):41–60, 2005.

[43] Stephen H. Muggleton. Inductive logic programming. *New Generation Computing*, 8(4):295–317, 1991.

[44] Shan-Hwei Nienhuys-Cheng and Ronald de Wolf. *Foundations of Inductive Logic Programming*, vol-

ume 1228 of *Lecture Notes in Artificial Intelligence*. Springer, 1997.

[45] J. Ross Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.

[46] Riccardo Rosati. On the decidability and complexity of integrating ontologies and rules. *Journal of Web Semantics*, 3(1):61–73, 2005.

[47] Riccardo Rosati. Semantic and computational advantages of the safe integration of ontologies and rules. In François Fages and Sylvain Soliman, editors, *Principles and Practice of Semantic Web Reasoning, Third International Workshop, PPSWR 2005, Dagstuhl Castle, Germany, September 11-16, 2005, Proceedings*, volume 3703 of *Lecture Notes in Computer Science*, pages 50–64. Springer, 2005.

[48] Riccardo Rosati. $\mathcal{DL}$+log: Tight Integration of Description Logics and Disjunctive Datalog. In P. Doherty, John Mylopoulos, and Christopher A. Welty, editors, *Proc. of Tenth International Conference on Principles of Knowledge Representation and Reasoning*, pages 68–78. AAAI Press, 2006.

[49] Riccardo Rosati. On Combining Description Logic Ontologies and Nonrecursive Datalog Rules. In Diego Calvanese and Georg Lausen, editors, *Web Reasoning and Rule Systems, Second International Conference, RR 2008, Karlsruhe, Germany, October 31-November 1, 2008. Proceedings*, volume 5341 of *Lecture Notes in Computer Science*, pages 13–27. Springer, 2008.

[50] Céline Rouveirol and Véronique Ventos. Towards Learning in CARIN-$\mathcal{ALN}$. In James Cussens and Alan M. Frisch, editors, *Inductive Logic Programming, 10th International Conference, ILP 2000, London, UK, July 24-27, 2000, Proceedings*, volume 1866 of *Lecture Notes in Artificial Intelligence*, pages 191–208. Springer, 2000.

[51] Edna Ruckhaus, Vladimir Kolovski, Bijan Parsia, and Bernardo Cuenca Grau. Integrating Datalog with OWL: Exploring the $\mathcal{AL}$-log Approach. In Sandro Etalle and Miroslaw Truszczynski, editors, *Logic Programming, 22nd International Conference, ICLP 2006, Seattle, WA, USA, August 17-20, 2006, Proceedings*, volume 4079 of *Lecture Notes in Computer Science*, pages 455–456. Springer, 2006.

[52] Chiaki Sakama. Nonmonotonic inductive logic programming. In Thomas Eiter, Wolfgang Faber, and Miroslaw Truszczynski, editors, *Logic Programming and Nonmonotonic Reasoning, 6th International Conference, LPNMR 2001, Vienna, Austria, September 17-19, 2001, Proceedings*, volume 2173 of *Lecture Notes in Computer Science*, pages 62–80. Springer, 2001.

[53] Ljiljana Stojanovic, Alexander Maedche, Boris Motik, and Nenad Stojanovic. User-driven ontology evolution management. In Asunción Gómez-Pérez and V. Richard Benjamins, editors, *Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web, 13th International Conference, EKAW 2002, Siguenza, Spain, October 1-4, 2002, Proceedings*, volume 2473 of *Lecture Notes in Computer Science*, pages 285–300. Springer, 2002.

# Part III

# Lexical Learning

# Learning Domain Ontologies by Corpus-Driven FrameNet Specialization

Bonaventura COPPOLA [a,1], Aldo GANGEMI [b], Alfio GLIOZZO [a], Davide PICCA [c], and Valentina PRESUTTI [b]

[a] *IBM Thomas J. Watson Research Center, Yorktown Heights, NY*
[b] *Semantic Technology Lab - ISTC-CNR, Roma, Italy*
[c] *Center for Computational Learning Systems - Columbia University, New York, NY*

**Abstract.** In this chapter we introduce a knowledge engineering methodology to adapt existing portions of FrameNet to new or specialized domains. Firstly, frame occurrences are detected in domain texts by a FrameNet-based statistical analyzer. Secondly, frame arguments are assigned additional semantic types by using a super-sense tagging tool. Thirdly, the resulting instances are statistically filtered in order to select the most relevant ones for the specific domain. Finally, we represent the newly created frames as OWL2 ontologies. We exploit state-of-the-art Natural Language Processing technology for frame detection and super-sense tagging. The formal semantics behind OWL2 is used overall to back the learning process: the semantics of frames is discussed, and choices are made to maintain the best from the two worlds of lexical and formal semantics, also exploiting the Linguistic Meta Model as a bridge. The proposed methodology is aimed at mostly automatizing the domain adaptation process performed by a domain expert. We retain a human intervention step for final quality assessment of new frames before their inclusion in the specialized domain ontology resulting from the process.

**Keywords.** Frame Semantics, Ontology Learning, FrameNet, Domain Adaptation, Linguistic Meta Model

## Introduction

Ontology learning from text is an important functionality for ontology design, due to the demand for domain-specific knowledge that supports semantic solutions scaling from personal knowledge management to large organizations, without requiring substantial competence from domain experts or expensive assistance from knowledge engineers. This is specially true if we consider the large number of different domains involved in modelling knowledge e.g. within an organization or at a Web-wide scale.

The current state of the art in ontology learning from text deals with taxonomical relations and simple binary domain relations, but research has just started dealing with more complex domain relations that can be able to answer realistic competency questions [25], such as *who is communicating what to whom and for what reason?*. Complex relations are called *semantic frames* in linguistics [19], and show interesting analogies

[1]Corresponding Author: Bonaventura Coppola, IBM Thomas J. Watson Research Center - 19 Skyline Drive, Hawthorne, NY 10532; E-mail: bcoppola@us.ibm.com

with other data structures that are known in Artificial Intelligence and the (Semantic) Web, such as *(AI) frames*, *microformats*, *microdata*, *infoboxes*, *knowledge patterns* [23], etc.

In this chapter we present a methodology to adapt FrameNet [4] to new domains. This is done by detecting semantic frames [19] and situations (the referents of frame-annotated sentences, cf. [20,29]) from textual domain corpora, and by using them to generate domain specific modules of a domain ontology. The methodology aims at specializing and/or instantiating a set of frames from the FrameNet lexicon, by extracting and filtering configurations of data from text corpora. The extracted domain-specific ontologies show desirable characteristics such as cognitive and linguistic relevance, density, right granularity level, and inherent capability of being automatically evaluated based on expert requirements (in the form of competency questions). The *ontology patterns* chapters from this book describes in some detail the challenges of ontology learning also with respect to frame-like structures, and relates them to *ontology design patterns*.

A previous attempt to achieve a similar objective has been reported by [5]. In contrast to that work, we have used different text processing technology and knowledge representation standards. A previous report about the methods described in this chapter is in [13].

The chapter is structured as follows. Section 1 describes the general methodology and shortly introduces the specific functions of the individual components of our workflow. A sample use case is also introduced in order to show the end to end process. The natural language processing technology for detecting occurrences of frames in a text corpus (Frame Detector) is presented in section 2.1. A Knowledge Distiller (Section 2.3) complements the output of the detector by means of a WordNet Super Sense Tagging component (Section 2.2), and learns typical patterns from the occurrences in a corpus, providing initial material to start a domain adaptation process on frames. The specific formal representation chosen for representing such frames (OWL2 [34] with the Linguistic Meta Model (LMM) vocabulary [30,20]) is presented in Section 3, thus connecting the acquired information to a stack of reusable lexical-semantic ontologies: FrameNet-OWL [20][29], WordNet-OWL [22,1], and DBpedia [2]. In Section 4 we show how the LMM representation of learnt frames is eventually converted into a regular ontology by means of *transformation patterns*, and evaluate the effectiveness of the whole process. We motivate how the methodology is sustainable and cost effective, in that it allows to process mid-size specialized corpora in a reasonable time. Also, the accuracy of the data distilled at the end of the acquisition process is good enough to allow for an easy assessment by an expert with reference to competency questions. In other words, the methodology turns into a substantial reduction of the manual effort required for designing domain-specific ontologies that are ready to be evaluated against user requirements.

## 1. Process Workflow and General Architecture

In Figure 1 we display the processing workflow realizing the proposed methodology. A text corpus representing a specific domain is provided in input. It is then analyzed exploiting Natural Language Processing (NLP) technology for frame detection and type

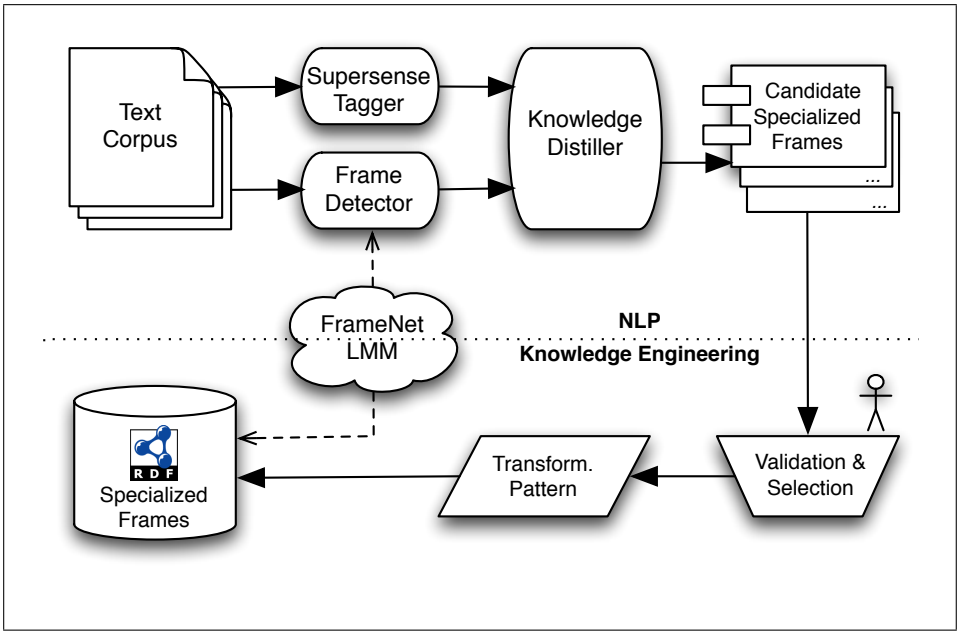



**Figure 1.:** Architecture

recognition. This is done by parallel execution of two advanced tools: a Frame Detection system, which recognizes the predicate argument structures appearing in the sentences and maps them into frame definitions provided by FrameNet. The second system is a Super Sense Tagger (SST), whose goal is to assign Super Senses (i.e. top level categories in WordNet like *person* or *group*) to frame arguments. Therefore, by analyzing a large corpus we collect a big number of frame instances enriched with super-sense types for their frame arguments. Then, we statistically filter them by applying a Knowledge Distiller, whose function is to select the most relevant frames and argument co-occurrence patterns, discarding the most noisy ones. After that, some of the distilled frames are not yet truly significant, requiring a manual validation and selection process performed by a domain expert. Finally, the selected frames are represented according to the Linguistic Meta Model, which allows to link the purely lexical semantics of extracted lexical elements to a formal semantics, eventually encoded in OWL2 [34]. This is done by applying some transformation patterns.

In the remaining part of this section we will show the application of the NLP section of the workflow on a real example. In the remaining sections of the chapter, we will describe each component in detail, present the resulting specialized frame, and eventually show the application of the transformation patterns on its LMM representation.
Let's consider the following sentence:

```
Public opinion and the press nowadays accuse Bush
of being unable to give a response.
```

This sentences describes a situation involving a judgment communication frame, whose definition in FrameNet is reported below:

**JudgmentCommunication**
*Core Elements:* **Communicator**, **Evaluee**, **Expressor**, **Medium**, **Reason**, **Topic**
*non-Core Elements:* **Addressee**, **Frequency**, **Manner**, **Place**, **Time**, **Manner**
*Related Frames:* **Judgment**, **Statement**, **Labeling**, **Bragging**

The sentence is parsed by the Frame Detector component using the above frame definition in order to recognize the exact text spans instantiating the frame's participating arguments (or *Elements*). To do that, the target word "evoking" the frame is first identified (in the example, the verb *accuse* is a candidate target word for the judgment communication frame). Then, the arguments of the frame are associated to the elements of the sentence's predicate argument structure provided by a syntactic parser. Eventually, the output of this process in output from the Frame Detector is the following analysis:

[*Public opinion and the press*]COMMUNICATOR nowadays <u>*accuse*</u> [*Bush*]EVALUEE
[*of being unable to give a response*]REASON

The underlined word *accuse* is the target word (or *lexical unit* or predicate) which plays the role of *evoker*. Slots marked by square brackets are the corresponding frame elements.

In the next step, the role of the SST is to identify concepts expressed in the text fragments and to annotate them by using general categories in order to facilitate the pattern acquisition process. To this aim, the SST module recognizes entity boundaries and provides the critical distinction between *instances* and *concepts*. In addition, it assigns types to each of them. This is practically done by semantically tagging all the relevant tokens by applying the *BOI* annotation scheme[2]. *B* and *I* tags are also associated to semantic types (e.g. *person*, *group*, *artifact*). Occurrences of nouns are then subdivided into instances (denoting proper nouns of actual entities, e.g. Obama, IBM) and concepts (denoting common nouns, e.g. cat, researcher).

The SST output for our sample sentence is shown below:

[2] where *B* stands for Begin, *I* stands for Inside and *O* stands for outside.

```
Public     B-noun.cognition_Concept
opinion    I-noun.cognition_Concept
and        O
the        O
press      I-noun.group_Concept
accuse     B-verb.communication
Bush       B-noun.person_Instance
of         O
being      B-verb.stative
unable     O
to         O
give       B-verb.possession
a          O
response   B-noun.phenomenon_Concept
```

The domain-specific frame patterns are then generated by collecting concepts and types as recognized by the SST[3]. The following instance illustrates the result of the pattern acquisition process after merging the output from the Frame Detector with the one from the SST system.

**Frame: JudgmentCommunication**
**Target:** accuse [communication]
**Communicator:** public opinion [cognition], press [group] [4]
**Evaluee:** [person] [5]
**Reason:** response [phenomenon] [6]

The above process, when applied to the whole text corpus, generates a large quantity of patterns, many of them being noisy due to the inherent difficulty of the NLP steps. In addition, some patterns are simply not meaningful for the target domain because they contain generic frame element instances. For example, in the example above, the slot filler *response* for the argument **Reason** is not domain-specific. Similarly, the type *cognition* for the frame argument **Communicator** is not very relevant as well. Therefore, the goal of the Knowledge Distiller component is to statistically filter a large quantity of those patterns before proposing them to the human validator. Such whole process is executed independently for each different frame. After applying the distiller, the instance above is simplified and the following is generated:

**Frame: JudgmentCommunication**
**Target:** accuse
**Communicator:** public opinion, press, [group]

[3]Occurrences of Verbs and Proper Nouns recognized by SST are not considered, and only their types are used.
[4]When more than one concept is identified in a slot filler, then all them are regarded as separate possible slot fillers, togheter with their types
[5]The instance name "Bush" is not acquired, only its type is.
[6]The full slot filler "of being unable to give a response" is not acquired, only concepts recognized inside are.

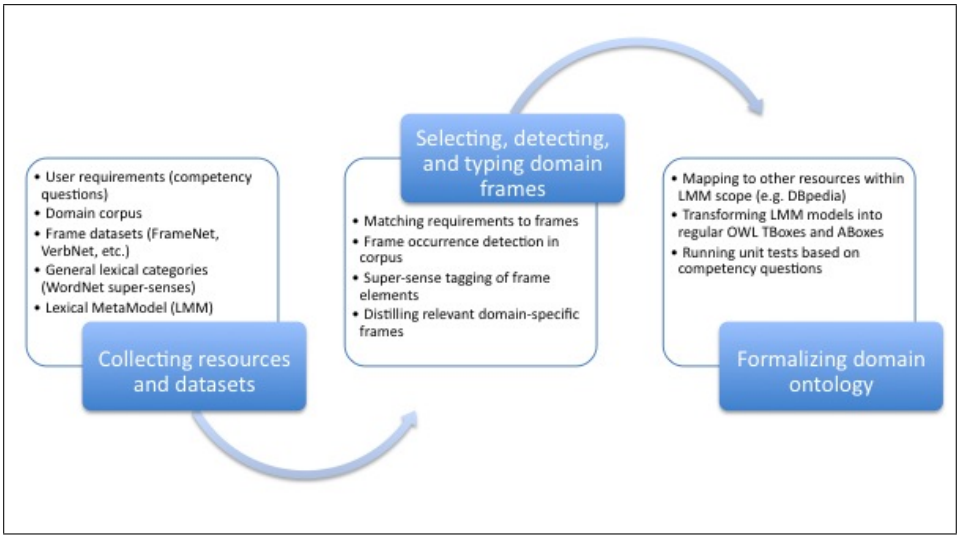



**Figure 2.:** A summary of the datasets, components, functionalities, and phases of the frame-based ontology learning method

**Evaluee:** [person]

Simplified patterns of this very kind are submitted to the domain expert, who can decide wether to accept them –and therefore including them as specific subframes in the final ontology, or to reject them. All the accepted frames are then automatically encoded in OWL2 according to some transformation patterns. Firstly, a newly created frame is automatically converted in OWL2 according to the LMM-based ontology of FrameNet, e.g.:

f:JudgmentCommunication s:hasFrameElement fe:Evaluee.judgmentCommunication .
f:JudgmentCommunication s:hasFrameElement fe:Medium.judgmentCommunication .
f:JudgmentCommunication s:hasFrameElement fe:Place.judgmentCommunication .
f:JudgmentCommunication s:hasFrameElement fe:Reason.judgmentCommunication .
f:Judgment_communication s:hasLexUnit lu:blame.v .
f:JudgmentCommunication s:hasSubframe f:Judgment .
f:JudgmentCommunication s:isInheritedBy f:JudgmentDirectAddress .
f:JudgmentCommunication s:uses f:Statement .

Secondly, the frame is transformed into a full-fledged OWL2 ontology according to a transformation pattern (examples are provided in Section 3, the method and the converted FrameNet in OWL are described in [29]). The result is a domain-specific ontology that can be further used to describe the domain of interest, typically events or situations that comply to the semantics of a frame. As noted before, we emphasize that the only manual intervention required is a selection/filtering procedure over preprocessed and clustered patterns, that can be done efficiently by domain experts to improve the quality of the generated ontology. Figure 2 provides a conceptual summary of such methodology.

## 2. The Frame Learning Components

In this section we present the three key NLP components exploited in the processing workflow described in Figure 1, namely the Frame Detector (Section 2.1), the Super Sense Tagger (Section 2.2), and the Knowledge Distiller (Section 2.3).

Ontology learning from text is usually performed with three components: a text corpus of an arbitrary size, a set of lexical resources, and some algorithms to extract, rank, and reengineer selected invariances from texts. Preliminary text classification may be also performed [6]. In the literature most of the effort has concentrated in learning the terminology for a specific domain, named entities from a text, taxonomies of concepts and their instances, relations between instances, trying to augment and adapt the information contained in e.g. WordNet with domain-specific data. Unfortunately, this is not enough for our purposes, since we are trying to learn structures with multiple related elements in the context of a frame, as occurring in a particular text. Therefore, we need text processing components that are able to identify occurrences of frames in text as in [14], and to recognize the corresponding frame elements and their types, also allowing for the learning of domain-specific specializations and instantiations.

To this aim, we have integrated two advanced text processing components: a Frame Detector and a Super-Sense Tagger, described in the following subsections. As a result, portions of text corresponding to lexical units and frame elements can be collected for each frame, which describe particular *framed situations* within a text.

### 2.1. The Frame Detection System

The Frame Detector component takes care of analyzing plain English input text, locating possible candidate frames invoked along by the terms in a text, and then tagging individual Frame Element instances of such frames. Tagging involves both locating the exact text boundary of each Frame Element, and assigning its correct semantic label. The Berkeley FrameNet Project [4] currently includes the definitions of nearly 800 Frames, 4,000 Frame Elements, and 135,000 annotated English sentences. Recalling the running example introduced in Section 1, an example of sentence annotation for the **Judgment-Communication** would be the following:

[*Public opinion and the press*]COMMUNICATOR nowadays <u>*accuse*</u> [*Bush*]EVALUEE
[*of being unable to give a response*]REASON

where the underlined word *accuse* is the target word (or *lexical unit* or predicate) which plays the role of *evoker* for this particular Frame.

To implement a FrameNet-based parsing system we adopted a multi-stage classification approach over natural language. Foundational studies in this area generally refer to Semantic Role Labeling [24]. In particular, our strategy extends the approaches in [28] to FrameNet and it is thoroughly described in [15,14,16]. Hence, moving from previous pattern-based approaches [33], we exploit Support Vector Machines (SVMs) as a general statistical machine learning framework, in which we plug in Tree Kernels to obtain a similarity measure capable of working over structured objects as syntactic trees. Our strategy includes (1) **Target Word Detection**, where the semantically relevant words bringing predicative information are detected; (2) **Frame Disambiguation**, where

the correct Frame for any target word has to be selected among several candidates; (3) **Boundary Detection**, where the sequences of words constituting the Frame Elements (arguments) are detected; and (4) **Role Classification**, which assigns semantic labels to the Frame Elements detected in the previous stage.

We have tested the Frame Detector Component in the natural setting of the FrameNet Corpus. Version 1.3 of FrameNet was used for both learning and test. After preprocessing and parsing with the Charniak's constituency-based syntactic parser we obtained 135,293 annotated and parsed sentences. We split the data considering the part of speech of predicates, ending up with 782 different frames. The overall dataset was then partitioned into 90% training set and 10% test set. We exploited both the SubSet Tree Kernel and the Polynomial Kernel (degree = 3) respectively over structured features [28] and standard ones [24]. In this way, with the Polynomial Kernel we also took advantage of previous liguistic modelling work done for traditional Semantic Role Labeling. Also, their combination was tested, which in fact brought the best results, see [15] for complete evaluation details. We computed Precision, Recall and $F_1$ measure of our classifiers for the step (3) *Boundary Detection*, and for the sequence of (3+4) *Boundary Detection + Role Classification*. The optimal learning configuration with Tree Kernel + Polynomial Kernel was capable of tagging FrameNet data over noisy syntax with exact boundaries and role labels (3+4) at 0.75 Precision, 0.55 Recall, and 0.63 $F_1$, which is the component-level evaluation for the Frame Detector. A performance drop when cross-testing on off-domain corpora as in the present work is always expected and observed.

### *2.2. The Super-Sense Tagger*

In our methodology, we exploit the Super-Sense Tagger (SST) described in [11] for a deeper semantic interpretation of texts, finalized to the acquisition of structured knowledge in a frame-based representation. Supersenses are lexicographer's categories, used to provide an initial broad classification for the lexicon entries in WordNet [18]. Although simplistic in many ways, the supersense ontology has several attractive features for ontology engineering. In fact, supersenses correspond to the typical high level classes of most ontologies, reflecting distinctions between persons, locations, organizations, acts, etc. In this work we use super-senses to characterize the type of Frame Elements. Exploiting a Super-Sense Tagger, we identify types in frame occurrences by filling the frame slots that are identified by the frame detection component.

SST[7] has been implemented using the SemCor corpus (a fraction of the Brown corpus annotated with WordNet word senses) and can be used for annotating large collections of English texts [10] The tagger implements a Hidden Markov Model, trained with the perceptron algorithm introduced in [12]. The tagset used by the tagger defines 26 supersense labels for nouns and 15 supersense labels for verbs. The tagger outputs named entity information, but also covers other relevant categories and attempts lexical disambiguation at the supersense level. In recent work, we have extended the tagset by subdividing each category into two sub-categories, *Instance* and *Concept* so that now the term "president" is tagged as *noun.person_Concept* and the term "Bill Clinton" as *noun.person_Instance*.

We evaluated the performances of SST by adopting a n-fold cross validation strategy on the SemCor corpus exploited for training. Precision, Recall and $F_1$ for any Su-

[7]The tagger is publicly available at: `http://sourceforge.net/projects/supersensetag/`.

persense were calculated. We obtained a general $F_1$ of 0.76. For some categories the $F_1$ is exceptionally high. Some of those best classified categories are really essential for ontology learning. For example, important labels such as *noun.person*, *noun.group*, or *noun.location* achieve results higher than 0.70. For some categories we even got a $F_1$ over 0.80: e.g. *noun.person_Instance*: $F_1$ 0.90) or *noun.person_Concept*: $F_1$ 0.81.

### *2.3. The Knowledge Distilling Component*

The output of the Frame Detector component provides a shallow semantic parsing of text, where text fragments are associated with frame elements, and the lexical units are identified. After joining these data along with the tagging provided by the SST, the final output of the text processing components is recalled in the example below:

**Frame: JudgmentCommunication**
**Target:** accuse [communication]
**Communicator:** public opinion [cognition], press [group]
**Evaluee:** [person]
**Reason:** response [phenomenon]

Next, we detect the relevant co-occurrence patterns, while avoiding the noise produced by the errors in the NLP components caused by the language variability. To this aim, the Knowledge Distiller recognizes a set of possible types for each frame element using the output of the SST system. Then, it detects the redundant concepts belonging to each type, and finally matches those concepts to patterns involving the elements recognized so far. It is implemented in a three-step approach described below:

**Type Recognition.** Most of the occurring elements that are recognized by the frame detector belong to a typically small number of super-senses (e.g. in a given domain *buyers* are either persons or groups, less likely they could be *animals*). For each frame element, we count the occurrences of all the different super senses associated with the matched arguments, and we filter out those having less that 10 occurrences.

**Domain Specialization** . This step allows to detect specific concepts that can be used for assigning types to the frame elements. For each frame element and for each super-sense detected by the step above, we select a list of (tagged) terms, and sort them by frequency. This step allows to filter out a significant part of the noise by exploiting the high redundancy existing in the corpus. Since the SST distinguishes between concept-tagging and instance-tagging, we select only the terms tagged by concept-tagging senses, having more than 3 occurrences. Those can be regarded as "valid" concepts to be used as semantic types. For example, we have found that occurrences of the frame element COMMUNICATOR belonging to the **person_concept** super-sense include terms like *child, woman* and *civilian*, while occurrences categorized by the **noun.state** super-sense, e.g. *recognition*, have a sensibly lower frequency, and are most likely to be errors.

**Frame Detection** The result of the two steps above is a list of "qualified", domain-oriented semantic types for domain-specific frame elements. Moreover, we are interested in frames involving multiple elements rather than in typing each frame ar-

gument independently. To this aim, we filter out all those terms that have not been identified by the domain specialization process from the frames acquired after the NLP process, and we rank the resulting list of domain frames by frequency. This procedure aims at detecting frames involving many arguments at the same time, in order to acquire statistically relevant tuples.

The result of these three steps above is a ranked list of relevant frames for a domain. This list is then proposed to the domain expert for a manual validation. Let's recall below an example of domain frame acquired using the methodology described so far:

**Frame: JudgmentCommunication**
**Target:** accuse
**Communicator:** public opinion
**Evaluee:** president [person]

The described procedure is fully automatic, and we have adopted the same threshold uniformly for three case study frames under analysis in this chapter (Section 4). The results of this method include a set of domain-specific frames that can be candidate data for answering a competency question, and therefore for implementing those frames as knowledge patterns in an ontology that fits user requirements.

## 3. Representing Frames in LMM

Frame Semantics [19] is a linguistic theory that aims at representing the conceptual interface between linguistic and real-world knowledge in the form of *semantic frames*, which are structures that describe types of situations along with the roles of their participating elements. Based on Frame Semantics, FrameNet [4] is a lexical knowledge base, consisting of a set of *frames*, which have proper *frame elements* (roles) and *lexical units*, also called the textual *target* of a frame. Frame elements are unique to their frame, and can be optional or non-typical ('non-core'). An occurrence of a frame consists in some piece of text, whose words can be normalized as *lexemes*, and which have semantic roles provided by the elements of that frame. A frame usually has only some of its roles actually lexicalized in texts. Types of linguistic realizations of a frame are called *lexical units*. Frames respectively frame elements are related between them, e.g. through the *inheritsFrom* or *hasSubFE* relations.

The intuition underlying Frame Semantics is not very dissimilar from that described in [27], but the latter developed in the AI context, and eventually acquired a formal semantics, as in [9]. The work in [20] is a reconstruction of different semantics of frames, and how they can be reconciled. As an assumption that can be adopted for different approaches to frame semantics, in [20] it is suggested that frames can be represented as (intensional) polymorphic n-ary relations, with typed arguments (either mandatory or optional). For example,

$$\texttt{Desiring(x,y,e)} \rightarrow \texttt{Agent(x)} \wedge \texttt{Agent(y)} \wedge \texttt{Event(e)} \quad (1)$$

An occurrence of a frame can be straightforwardly treated as an instance of an n-ary relation, e.g.:

$$\texttt{Desiring(Susan, Marko, ListeningToHer)} \tag{2}$$

The logical representation of frames as n-ary relations is easily generalizable and clear, but hardly manageable by automated reasoners on large knowledge bases, specially with semantic web languages like RDF or OWL. A hard design problem is constituted by the polymorphism of many frames/n-ary relations, which can vary in number of the arguments that can be taken by the relation. For example, the same frame **Desiring** can be assumed with four arguments:

$$\begin{aligned}\texttt{Desiring(x,y,e,t)} \rightarrow \texttt{Agent(x)} \wedge \texttt{Agent(y)} \wedge \\ \wedge\ \texttt{Event(e)} \wedge \texttt{Time(t)}\end{aligned} \tag{3}$$

This problem was originally evidenced by Davidson with reference to a logic of events [17]. A neo-davidsonian approach is actually taken by [8], where an attempt is made to reconstruct FrameNet in terms of DRT (Discourse Representation Theory), with *events* being the DRT correlate of frames, and *semantic roles* being generic correlates for equivalence classes of frame elements[8]. A limitation found by [8] lies in the fact that DRT semantic primitives are tightly associated with elements having a given syntactic category (e.g. it assumes that an event cannot be expressed by an adverb), while FrameNet semantics only requires that any grammatical *construction* expresses a frame, independently from its syntactic category. The DRT approach has anyway the (computational) advantage of simplifying some assumptions of Frame Semantics, for example the uniqueness of frame elements to their frame. A formal semantics for frames that is also computationally manageable has been provided by Description Logics (DL) [3], such as OWL2(DL). Due to their limited expressive power, description logics represent frames as classes, with *roles* (binary relations) that link a class to the types of the arguments of the original n-ary relation. Those types are classes as well, so that a graph of frames emerges out of this semantics. The example in (3) can be reengineered in DL as follows:

$$\texttt{T} \sqsubseteq \forall \texttt{R}_1\texttt{.Agent},\ \texttt{T} \sqsubseteq \forall \texttt{R}_1^-\texttt{.Desiring} \tag{4}$$

$$\texttt{T} \sqsubseteq \forall \texttt{R}_2\texttt{.Agent},\ \texttt{T} \sqsubseteq \forall \texttt{R}_2^-\texttt{.Desiring} \tag{5}$$

$$\texttt{T} \sqsubseteq \forall \texttt{R}_3\texttt{.Event},\ \texttt{T} \sqsubseteq \forall \texttt{R}_3^-\texttt{.Desiring} \tag{6}$$

$$\texttt{T} \sqsubseteq \forall \texttt{R}_4\texttt{.Time},\ \texttt{T} \sqsubseteq \forall \texttt{R}_4^-\texttt{.Desiring} \tag{7}$$

$$\texttt{Desiring} \sqsubseteq (\exists \texttt{R}_1 \sqcap \exists \texttt{R}_2 \sqcap \exists \texttt{R}_3 \sqcap \exists \texttt{R}_4) \tag{8}$$

$$\texttt{hasKey(Desiring,(R}_1\texttt{,R}_2\texttt{,R}_3\texttt{,R}_4\texttt{))} \tag{9}$$

With $R_i$ being a binary projection of the n-ary relation, $R_i^-$ its inverse relation, and axiom 9 the identification constraint for avoiding duplicates of a same relation tuple after reification.[9] The computational features of description logics make them a reasonable choice to formally represent linguistic frames, and this is the approach followed e.g. by [32].[10] On the other hand, FrameNet contains many meta-level relations that convey

[8] Frame elements are assumed to be *local* to a frame: this is quite complex an assumption for typical semantic parsers

[9] This axiom is known as `hasKey` in OWL2(DL).

[10] F-logic [26] is another alternative that however has no direct correspondence to current standard semantic web languages.

the intended semantics of frames: relations between frames, between frame elements, and between frames and frame elements, lexical units, and lexemes, between sentences and words, valences and lexical units, etc. OWL2 is appropriate to encode those meta-level relation by means of its *punning* mechanism: when e.g. two frames are related, their formal interpretation consists of a relation between *Individuals* and not between *Classes*. Clearly, this is not a real representation of full FrameNet semantics, since the intended reasoning of e.g. `subFrame` relation between frames requires a proper second-order semantics which is not attainable by a simple punning mechanism. Moreover, the alignment between the different lexical data coming from frame detection, super-sense typing, possible alignments to other lexical resources, etc., requires a proper lexical meta-model.

The Lexical Meta-Model (LMM) [30] is a formal semiotic vocabulary compliant with OWL2 punning, which also enables the alignment of heterogeneous lexical resources. LMM is implemented in OWL2[11]. LMM facilitates the integration of heterogeneous lexical knowledge resources, such as WordNet [18] and FrameNet[12], as well as of other semi-formal resources (thesauri, subject directories, etc.), with regular ontologies expressed in a formal language like OWL.

LMM can be considered as a *semiotic façade* enabling the reengineering of a lexically-based resource to an ontology resource [20]. In the methodology reported in this chapter, it is used to make different lexical data interoperable, and to transform the output of the three learning components *frame detector, super-sense tagger*, and *distiller* (see next sections) into an OWL ontology. The core LMM elements are `lmm:Expression`, which can `lmm:express` a `lmm:Meaning` (that is `lmm:conceptualizedBy lmm:Agent`), and which can `lmm:denote` a `lmm:Reference`; in addition, any entity can have a `lmm:Context` and can be `lmm:representedBy` a `lmm:FormalExpression`. Expressions can be any information object, and lexical items in particular; references can be any entity; meanings can be any information or conceptual entity that is used to describe or encode the (informal, natural) semantics of an expression, i.e. other expressions, concepts from a thesaurus, frames, topics, etc. A rich set of relations is defined in LMM to associate meanings between them, as well as to associate expressions, meanings, and references to formal constructs. A complete reengineering of FrameNet as a plugin to LMM can be found in the OWL version of FrameNet[13][20] (Fig. 3). A `ofn:Frame` has some `ofn:FrameElement`, is lexically realized by some `ofn:LexicalUnit`, and all these can be evoked by some `ofn:Lexeme`, and can have some `ofn:SemanticType`. Since semantic types do not cover all frame elements and lexical units, [14] we use the super-sense tagger component (section 2.2) to infer WordNet `wn:SuperSenses` as additional types for frame elements. All mentioned classes and relations are aligned to LMM; an excerpt of alignments can be found here:[15]

[11] `http://www.ontologydesignpatterns.org/ont/lmm/lmm1.owl`.

[12] The OWL ontology that encodes the alignment of the FrameNet OWL schema to LMM can be downloaded at `http://www.ontologydesignpatterns.org/ont/lmm/ofn2lmm.owl`

[13] `http://www.ontologydesignpattern.org/ont/ofn/ofntb.owl`

[14] In FrameNet 1.3 the majority of frames (719 out of 795) has types for one or more of their frame elements (4187 out of 7124), while no lexical unit (about 10000) has semantic types assigned to the application of frame elements in their context

[15] `http://www.ontologydesignpattern.org/ont/lmm/ofn2lmm.owl`

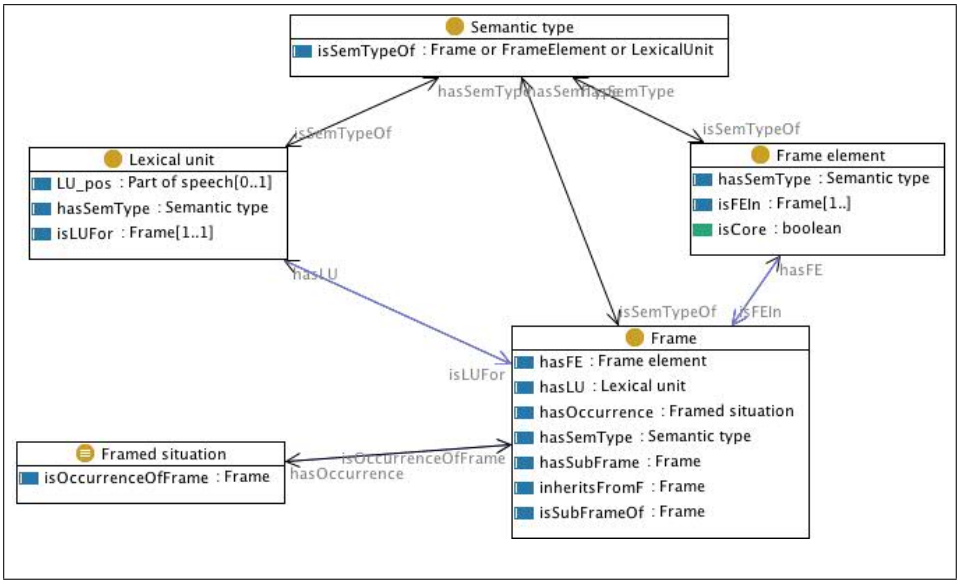



**Figure 3.:** FrameNet OWL: A full-fledged metamodel for FrameNet

```
ofn:Lexeme rdfs:subClassOf lmm:Expression                    (10)
ofn:Frame rdfs:subClassOf lmm:Meaning                        (11)
ofn:FrameElement rdfs:subClassOf lmm:Meaning                 (12)
ofn:evokes rdfs:subClassOf lmm:expresses                     (13)
ofn:FramedSituation rdfs:subClassOf lmm:Reference            (14)
```

A thorough discussion on alternative formal translations of FrameNet is contained in [20]. A complete translation of the new FrameNet 1.5 version to RDF and OWL linked data has been performed [29] has been recently produced, and reported in [29]. It does not change the meta-model substantially however.

## 4. Process Evaluation and Knowledge Engineering

We have tested our methods on the Europarl corpus[16], comprising about 30 million documents, extracted from the proceedings of the European Parliament. It includes 11 official languages of the European Union. For our experiments, we have used the English part, including 1,461,429 sentences and 39,618,240 words.

In step 1, sentences from the text corpus are annotated with super-senses by the SST, and with frames, frame elements, and lexical units by the Frame Detector component. For the sake of this experiment, we focused on the annotation of three frames: **Killing**, **JudgmentCommunication**, and **Commerce**. Results for each frame have been collected in tables reporting, for each frame occurrence recognized in the corpus, all the frame el-

[16] http://homepages.inf.ed.ac.uk/pkoehn/publications/europarl-mtsummit05.pdf

| Communicator | TargetVerb | Evaluee | Reason | *Validity* |
|---|---|---|---|---|
| people | accuse | european union | - | *ok* |
| - | accused | security | violations | *ok* |
| member states | blame | problems | - | *ok* |
| - | blamed | fishermen | negotiations | *ok* |
| companies | charge | price | - | *no* |
| amnesty international | cited | - | torture | *ok* |
| commission | charged | - | failure | *ok* |
| - | charged | authorities | crime | *ok* |
| - | charged | group | report | *ok* |
| employment | charged | - | - | *no* |

**Table 1.** A sample output of the distiller with manual evaluation.

ements and their types, as recognized by SST. Occurrences of core frame elements show meaningful frequency in the corpus, so confirming the hypothesis that some elements are more central than others in the conceptualization of frames. About 1000 occurrences for each frame have been collected after step 1.

In step 2, the raw frame occurrence data have been filtered by applying the procedure described in section 2.3, consisting of identifying the most frequent super-senses for each frame element, in order to learn more specific types (e.g. `Person` or `Group`, and to filter out instances having different types (all super-senses can be assumed as disjoint classes of synsets, therefore multi-typing with super-senses should be avoided). After applying the automatic distiller procedure based on combined frequency thresholds, data from only about 100 frame occurrences have been retained as candidate domain-specific frames for each generic frame. In addition to types, step 2 allows to learn sensible lexemes for the domain specialization of frame elements (e.g. *people*), and sensible lexical units (e.g. `Accuse` for the candidate frame specializations. An example of the outcome of the distiller after evaluation is shown in Table 1. Each distilled domain frame is formalized as an `owl:NamedIndividual rdf:type ofn:Frame`, thus resulting to be also `rdf:type lmm:Meaning` because of the LMM alignment, according to the guidelines provided in section 3. For example, in one filtered occurrence we first specialize the generic frame (15), and then add a new super-sense typing over its frame elements (16).

```
europarl:AccuseOfViolation lmm:specializes
            ofnabox:JudgmentCommunication                    (15)
ofnabox:Communicator ofn:hasSemType europarl:People          (16)
```

After a manual evaluation on the domain-specific frames detected for the three generic frames chosen, 62% of distilled domain-specific frames resulted to be valid, where validity is assumed as respecting the conceptualization of the generic frame, in terms of extensional interpretation of the classes derived from target verbs, and of properties derived from frame elements. In this section we explain the workflow used to make this interpretation viable.

The validation of domain-specific frames is performed by assuming a rapid ontology design process, where frames are converted into *knowledge patterns*,[31,23,29] i.e.

small ontologies that are topologically dense, provide solutions to competency questions (task orientedness), are linguistically grounded, and cognitively appropriate. Linguistic groundedness is naturally provided by the corpus, cognitive appropriateness is here inherited from FrameNet frames that are specialized (section 3). We will describe here density and task orientedness of the learnt patterns. In practice, ontology designers acting as evaluators have been asked to validate the detected frames in terms of the density and task properties. *Topological density* requires that the ontology corresponding to a domain-specific frame is (syntactically) a labeled connected graph with nodes labeled by owl:Class names and edges labeled by owl:ObjectProperty names only.[17] Density is ensured in principle, because the domain-specific frame inherits the conceptual structure of the generic frame that has been specialized, and all FrameNet frames ensure density because of their design after conversion (section 3).

On the contrary, ontologies produced by traditional learning methods typically lack density and do not feature the expected formal structure implicit in experts' conceptualization of their domain. Experts' conceptualization as extracted from text is in fact dependent on the syntactic and semantic dependencies within the text, which are mostly ignored by traditional ontology learning methods [21].

We remark that frame-based ontology learning has also this desirable property. Since frames are formally equivalent to n-ary relations at design time (section 3), and competency questions can be formalized as n-ary relations at query time [7], it is straightforward to match a linguistic competency question like: *what are the reasons for those events?* to the knowledge base of domain-specific frames.[18]

Domain frames can be used for two different design tasks: the first one is producing a traditional TBox domain ontology, e.g. consisting of OWL classes like `AccuseOfViolation`; the second one is to classify named entities that instantiate the domain frame (*framed situations*), which are formalized as OWL individuals.

The generation of a TBox from the LMM-encoded domain frame and framed situations is performed by means of customizable *transformation patterns* [20], which encode rules for translating each instance of an LMM type (Meaning, Expression, etc.) into an OWL element (see also [21]. In other words, a transformation pattern is a recipe to transform a certain class of logical constructs into another.

As a concrete example, we show a diagram depicting a proposed knowledge pattern that has been derived from the domain-specific frame **AccuseOfViolation** (Fig. 4). The transformation patterns adopted are exemplified here as a set of OWL2Full axioms acting over the domain of FrameNet OWL schema, and the OWL metamodel:[19] OWL2 punning is again used here.

`ofn:Frame trans:transformableTo owl:Class` (17)

[17] Taxonomical (rdfs:subClassOf) and owl:DatatypeProperty links do not contribute to the density of an ontology.

[18] Several domain-specific frames can be retrieved which have evidence in the corpus: **AccuseOfViolation**, **BlameNegotiation**, **ChargeForFailure**, **ChargeForCrime**, etc.

[19] Transformation patterns are based on alignments between the FrameNet schema and LMM (cf. section 3), between LMM and the FormalSemantics vocabulary: `http://ontologydesignpatterns.org/ont/dul/FormalSemantics.owl`, and the OWL meta-model.

```
ofn:FrameElement trans:transformableTo owl:ObjectProperty (18)
ofn:SemanticType trans:transformableTo owl:Class (19)
ofn:hasFE trans:transformableTo owl:Restriction (20)
ofn:hasSemType trans:transformableTo rdfs:range (21)
```

For example, given the previous axioms 15 and 16, and the following axioms extending FrameNet OWL ABox,[20]:

```
europarl:AccuseOfViolation ofn:hasFE ofnabox:Communicator (22)
europarl:AccuseOfViolation ofn:hasFE ofnabox:Reason (23)
ofnabox:Reason ofn:hasSemType europarl:Violation (24)
```

we obtain the following TBox axioms, after applying the transformation patterns:

$$\texttt{T} \sqsubseteq \forall \texttt{ofnowl:Communicator.europarl:People} \quad (25)$$

$$\texttt{T} \sqsubseteq \forall \texttt{ofnowl:Communicator}^{-}\texttt{.europarl:AccuseOfViolation} \quad (26)$$

$$\texttt{T} \sqsubseteq \forall \texttt{ofnowl:Reason.europarl:Violation} \quad (27)$$

$$\texttt{T} \sqsubseteq \forall \texttt{ofnowl:Reason}^{-}\texttt{.europarl:AccuseOfViolation} \quad (28)$$

Task-based fitness can be proved by launching a unit-test (cf. [7]) that contains data about the `europarl:AccuseOfViolation` competency question: *what accuses of violation people has made for what reason?*. From the n-ary relation hypothesis, we know that a frame has direct translations into competency questions, queries, and ontologies: if the TBox produced is able to support the proper queries, then the TBox fits the task. This is the case actually, since the classes and properties that have been built (Fig. 4) are able to encode the query derived from the competency question:

```
SELECT DISTINCT ?x ?y ?z WHERE {?x a Accuse . ?y a People .
?z a Violation . ?x :Communicator ?y . ?x :Reason ?z} (29)
```

## 5. Conclusions

In this chapter we have presented and implemented a method for frame-based ontology learning by using components for frame detection and super-sense tagging on a text corpus, the FrameNet and WordNet lexical knowledge bases, and the LMM meta-modelling framework.

The method can be used to generate candidate *domain-specific* frames suggested by qualified corpus evidence, and to populate an ontology with complex facts extracted from the corpus. The method fits a task-based analysis and evaluation of ontologies, as

[20] http://www.ontologydesignpatterns.org/ofnframes/accuseofviolation.owl

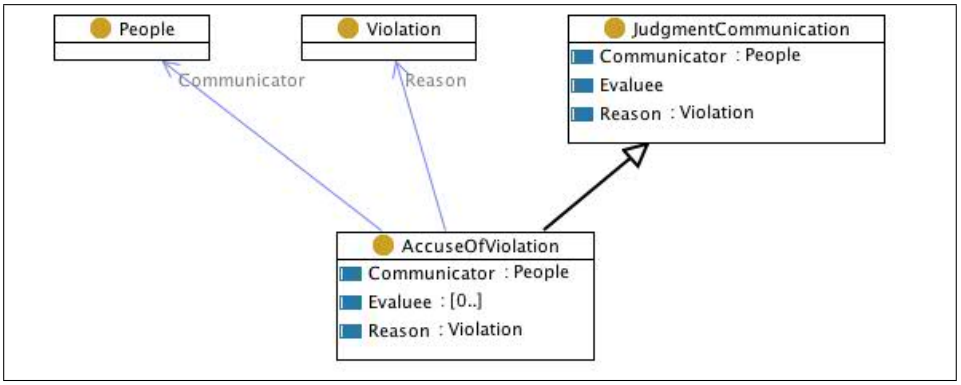



**Figure 4.:** A knowledge pattern derived from the domain-specific frame **AccuseOfViolation**, based on the transformation patterns.

it has been developed in the NeOn EU project,[21] where novel methods for pattern-based design have been implemented [7].

The presented method shows some significant improvements in terms of functionalities and technology with respect to existing ontology learning approaches, not only because we are acquiring more complex knowledge structures (i.e. frames instead of simple taxonomies or binary relations) but since we also provide an easier alignment to existing semantic web resources: the use of LMM makes the results more easily mappable to different kinds of data, such as datasets from Linked Open Data.

Our experiment proves that we can build a resource of reusable knowledge patterns by learning them from domain text corpora. There are challenging research issues in this area of ontology design. An important one is this: since the amount of domain-oriented patterns can be huge, and FrameNet alone provides around 1000 generic frames, are they complete, so that all domain patterns specialize them? Should we consider a procedure to learn patterns without previous knowledge of generic ones? Such questions are being investigated within the research programme outlined in [23].

## References

[1] M. van Assem, A. Gangemi, and G. Schreiber. Conversion of WordNet to a standard RDF/OWL representation. In *Proc. of the Int. Conference on Language Resources and Evaluation (LREC) , Genoa, Italy*. 2006.

[2] S. Auer, C. Bizer, J. Lehmann, G. Kobilarov, R. Cyganiak, and Z. Ives. DBpedia: A Nucleus for a Web of Open Data. In *Aberer et al. (Eds.): 6th International Semantic Web Conference*. Lecture Notes in Computer Science, Springer, 2007.

[3] F. Baader, editor. *The Description Logic Handbook: theory, implementation, and applications*. Cambridge University Press, Cambridge, 2003.

[4] C.F. Baker, C.J. Fillmore, and J.B. Lowe. The Berkeley FrameNet Project. In *Proceedings of the 17th international conference on Computational linguistics*, pages 86–90, Morristown, NJ, USA, 1998.

[5] R. Basili, C. Giannone, and D. De Cao. Learning domain-specific framenets from texts. In *ECAI Workshop on Ontology Learning and Population*, 2008.

[6] R. Basili, A. Moschitti, and M.T. Pazienza. A text classifier based on linguistic processing. In *Proceedings of the IJCAI-99 Workshop on Machine Learning for Information Filtering*, 1999.

[21] `http://www.neon-project.org`

[7] E. Blomqvist, V. Presutti, A. Gangemi, and E. Daga. Experimenting with extreme design. In *In Proceedings of the Conference on Knowledge Engineering and Knowledge Management (EKAW2010)*, Redondo Beach, California, USA, 2010. Springer.

[8] J. Bos and M. Nissim. Combining Discourse Representation Theory with FrameNet. In R. Rossini Favretti, editor, *Frames, Corpora, and Knowledge Representation*, pages 169–183. Bononia University Press, 2008.

[9] R.J. Brachman. A Structural Paradigm for Representing Knowledge. Ph.d. thesis, Harvard University, USA, 1977.

[10] M. Ciaramita and Y. Altun. Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger. In *Proceedings of EMNLP-06*, pages 594–602, Sydney, Australia, 2006.

[11] M. Ciaramita and M. Johnson. Supersense tagging of unknown nouns in wordnet. In *Proceedings of EMNLP-03*, pages 168–175, Sapporo, Japan, 2003.

[12] M. Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP-02*, 2002.

[13] B. Coppola, A. Gangemi, A.M. Gliozzo, D. Picca, and V. Presutti. Frame detection over the semantic web. In Lora Aroyo, Paolo Traverso, Fabio Ciravegna, Philipp Cimiano, Tom Heath, Eero Hyvönen, Riichiro Mizoguchi, Eyal Oren, Marta Sabou, and Elena Paslaru Bontas Simperl, editors, *ESWC*, volume 5554 of *Lecture Notes in Computer Science*, pages 126–142. Springer, 2009.

[14] B. Coppola and A. Moschitti. A general purpose framenet-based shallow semantic parser. In Nicoletta Calzolari (Conference Chair) et al., editor, *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta, may 2010. European Language Resources Association (ELRA).

[15] B. Coppola, A. Moschitti, and D. Pighin. Generalized framework for syntax-based relation mining. In *Proceedings of the IEEE International Conference on Data Mining (ICDM 2008)*, Pisa, Italy, 2008.

[16] B. Coppola, A. Moschitti, S. Tonelli, and G. Riccardi. Automatic framenet-based annotation of conversational speech. In *Proceedings of the IEEE Workshop on Spoken Language Technology (SLT 2008)*, Goa, India, 2008.

[17] D. Davidson. The Logical Form of Action Sentences. In *The Logic of Decision and Action*. University of Pittsburgh Press, Pittsburgh, 2nd edition, 1967.

[18] C. Fellbaum, editor. *WordNet. An Electronic Lexical Database*. MIT Press, 1998.

[19] C.J. Fillmore. The Case for Case. In Emmon Bach and Robert T. Harms, editors, *Universals in Linguistic Theory*, pages 1–210. Holt, Rinehart, and Winston, New York, 1968.

[20] A. Gangemi. *What's in a schema? A formal metamodel for ECG and FrameNet*. Studies in Natural Language Processing. Cambridge University Press, April 2010.

[21] A. Gangemi. Back to the future: Frame representation and semantic technologies. *Cahiers de Lexicologie*, 99(2), 2012.

[22] A. Gangemi, R. Navigli, and P. Velardi. The OntoWordNet Project: Extension and Axiomatization of Conceptual Relations in WordNet. In *ODBASE 2003*, 2003.

[23] A. Gangemi and V. Presutti. Towards a Pattern Science for the Semantic Web. *Semantic Web*, 1(1-2):61–68, 2010.

[24] D. Gildea and D. Jurafsky. Automatic Labeling of Semantic Roles. *Computational Linguistics*, 28(3):245–288, 2002.

[25] M. Gruninger and M. Fox. The role of competency questions in enterprise engineering, 1994.

[26] M. Kifer, G. Lausen, and J. Wu. Logical Foundations of Object-Oriented and Frame-Based Languages. *Journal of Association for Computing Machinery*, 42:4:741–843, 1995.

[27] M. Minsky. A Framework for Representing Knowledge. In P. Winston, editor, *The Psychology of Computer Vision*. McGraw-Hill, 1975.

[28] A. Moschitti, D. Pighin, and R. Basili. Tree kernels for semantic role labeling. *Computational Linguistics*, 34(2):193–224, 2008.

[29] A. G. Nuzzolese, A. Gangemi, and V. Presutti. Gathering Lexical Linked Data and Knowledge Patterns from FrameNet. In *Proc. of the 6th International Conference on Knowledge Capture (K-CAP)*, pages 41–48, Banff, Alberta, Canada, 2011.

[30] D. Picca, A. Gangemi, and A. Gliozzo. LMM: an OWL Metamodel to Represent Heterogeneous Lexical Knowledge. In *Proc. of the International Conference on Language Resources and Evaluation (LREC), Marrakech, Morocco*, 2008.

[31] V. Presutti and A. Gangemi. Content Ontology Design Patterns as Practical Building Blocks for Web

Ontologies. In *Proceedings of the 27th International Conference on Conceptual Modeling (ER 2008)*, Berlin, 2008. Springer.

[32] J. Scheffczyk, C. F. Baker, and S. Narayanan. Reasoning over Natural Language Text by means of FrameNet and Ontologies. In *Ontologies and the Lexicon*. Cambridge University Press, 2010.

[33] H. Tanev, M. Kouylekov, M. Negri, B. Coppola, and B. Magnini. Multilingual pattern libraries for question answering: a case study for definition questions. In *Proceedings of LREC 2004*, Lisbon, Portugal, 2004.

[34] W3C OWL Working Group. OWL 2 Web Ontology Language, W3C Recommendation, 27 October 2009. http://http://www.w3.org/TR/owl2-overview/, 2009.

# Information Extraction for Ontology Learning

Fabian SUCHANEK
*Max Planck Institute for Informatics, Germany*

**Abstract.** In this chapter, we discuss how ontologies can be constructed by extracting information from Web documents. This is a challenging task, because information extraction is usually a noisy endeavor, whereas ontologies usually require clean and crisp data. This means that the extracted information has to be cleaned, disambiguated, and made logically consistent to some degree. We will discuss three approaches that extract an ontology in this spirit from Wikipedia (DBpedia, YAGO, and KOG). We will also present approaches that aim to extract an ontology from natural language documents or, by extension, from the entire Web (OntoUSP, NELL and SOFIE). We will show that information extraction and ontology construction can enter into a fruitful reinforcement loop, where more extracted information leads to a larger ontology, and a larger ontology helps extracting more information.

**Keywords.** Information Extraction, Reasoning, Consistency, Taxonomy, Disambiguation

## 1. Introduction

### *1.1. Motivation*

The World Wide Web provides an enormous source of knowledge. News articles, university home pages, scientific papers, personal blogs, commercial Web sites, catalogs such as the Internet Movie Database, and entire encyclopedias such as Wikipedia are available online. These sites host information on a wealth of topics, from scientific theories to current political events, from digital cameras to endangered wildlife species and from train schedules to cooking recipes. This information is available as digital documents. This means that machines can store, display and index these documents. However, machines do not have a semantic representation of the content of the documents. This means that machines cannot figure out whether a document contains contradictory information. It also means that we cannot ask SPARQL queries against the documents. Thus, there is a gap between the syntactical way in which data is stored on the Web and the semantic way in which information is stored in RDF/OWL ontologies.

Information Extraction (IE) steps in to bridge that gap. IE is the science of extracting structured data from unstructured or semi-structured documents. For example, given a natural language document with the sentence “Elvis Presley was born in Tupelo", the goal of an IE system could be to extract the triple *<Elvis, bornIn, Tupelo>*. This fact could then become part of an ontology. If this process could be applied to all the documents of the Web, then the syntactic information of the Web could become semantically

accessible to machines. This chapter will present recent promising work in this direction. The remainder of this section will first discuss IE in general and for ontologies in particular. Then, Section 2 will present approaches that extract ontological information from Wikipedia. Section 3 discusses approaches that target unstructured Web pages, or the entire Web, before Section 4 concludes.

### *1.2. Information Extraction*

IE is the process of extracting structured information from one or multiple given source documents. There are literally hundreds of different IE techniques. Some of these techniques have become particularly popular. At first, the document is usually preprocessed. This may include character set detection or conversion, the removal of special characters, DOM tree construction for HTML documents and general cleaning, such as the removal of irrelevant portions of the document. Next, the document is usually segmented into words or tokens (cf. Maynard and Bontcheva [13], this volume). At the level of tokens, we can already apply techniques of Named Entity Recognition. NER will identify the tokens that are dates, numbers and proper names. Some NER techniques can even distinguish location names, organization names and person names. All approaches that we will see in this chapter use basic preprocessing of this kind.

After NER, first methods of fact extraction can be applied. If the document is structured (such as HTML tables, lists, or Web sites that follow a template), we can employ techniques such as wrapper induction [9] or table annotation [12]. These techniques can extract tabular information from Web pages. If the document is unstructured (i.e., if it consists of natural language text), we can apply approaches that seek patterns in the sentences of the text [10]. For example, the pattern "X is a Y" indicates that X could be an instance of the concept Y (such as in the sentence "Elvis Presley is a rock singer"). The patterns can either be predefined or learned from examples. Other techniques rely on a co-occurrence analysis of terms. For example, if *Elvis Presley* always appears together with the words *guitar* and *sing*, then this may indicate that he is a musician.

If the document contains natural language text, it can also be linguistically analyzed, as explained by Maynard and Bontcheva [13] (this volume). This may include part-of-speech tagging and syntactic analysis, such as dependency parsing. Then, linguistic techniques can be applied to deduce the meaning of a sentence and distill the facts it expresses. There are many more approaches, which blend, adapt or use the previously mentioned approaches or which take a new perspective altogether. The reader is referred to [18] for a general survey of techniques.

### *1.3. IE for Ontologies*

Traditional IE concentrated on entity extraction, relation extraction, and fact extraction. Given a sentence "Elvis married Priscilla Beaulieu", an IE system aimed to extract that the relation *married* holds between *Elvis* and *Priscilla Beaulieu*. If this fact is to be used in an ontology, however, it is not sufficient to extract just these three tokens. Rather, three more desiderata have to be fulfilled:

1. **Canonicity:** The entity names have to be disambiguated and mapped to existing entities of the ontology. In the example, the system has to determine that "Elvis" refers to the entity *Elvis Presley* (instead of, say, the singer *Elvis Costello*). The

system may also decide to introduce a new entity if the entity does not yet exist in the knowledge base. The same applies to class names and relation names.
2. **Taxonomic organization:** The extracted entities have to be placed in a taxonomy of classes. In the example, the system has to know that Elvis is a singer and that this implies that he is a person.
3. **Consistency:** The extracted facts have to be logically consistent with the ontology. In the example, the system may have to check that Priscilla was born before Elvis died.

These desiderata put an additional burden on the IE system. At the same time, the ontology can also help the IE process. This can happen on multiple levels. First, the data that is already present in the ontology can serve as seed data for the IE process. For example, assume that the ontology already knows that Einstein was married to Mileva Maric. If the system comes across the sentence "Einstein married Mileva Maric", the system can guess the meaning of the pattern "X married Y". This, in turn, will allow it to understand the sentence "Elvis married Priscilla Beaulieu". The ontology can also help in another way: If the ontology has to be logically consistent, then certain interpretations of the source document can be excluded. In our sample sentence, the word "Elvis" cannot refer to Saint Elvis, because this saint lived in the 5th century and cannot have married Priscilla.

Thus, the more knowledge has already been extracted, the better the system will be able to extract new information. The better it extracts new information, the more knowledge will be extracted. Much like humans, who learn faster when they know more, and know more when they learn faster, IE and ontologies could enter a fruitful cycle of knowledge accumulation. We will look at two classes of ontological knowledge extraction systems that go into this direction. The first class extracts information from Wikipedia. The second class ventures beyond Wikipedia and extracts information from arbitrary documents – or targets even the whole Web.

## 2. IE from Wikipedia

Wikipedia is a multilingual, Web-based encyclopedia. It is written collaboratively by volunteers and is available for free under the terms of the Creative Commons Attribution-ShareAlike License[1]. As of February 2011, the English Wikipedia contained more than 3.5 million articles[2]. Wikipedia suggests itself for ontological information for multiple reasons. First, Wikipedia articles follow a number of conventions concerning the formatting and the content. This makes Wikipedia much more homogeneous than a set of random Web pages and facilitates IE. Second, Wikipedia is constantly updated and improved by thousands of volunteers, so that it is widely regarded as credible, correct and authoritative. Wikipedia also covers many different aspects of knowledge, from science to literature or politics, which makes it useful as a source for a general-purpose ontology. Lastly, Wikipedia is a highly structured resource, which contains not just natural language texts, but also standardized tabular information, a category system and meta-

[1] `http://creativecommons.org/`
[2] `http://en.wikipedia.org/wiki/Wikipedia:Statistics`

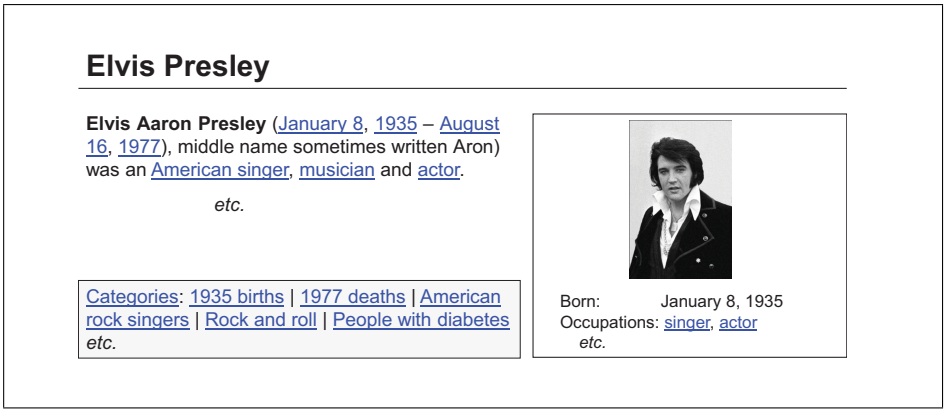



**Figure 1.** A Wikipedia Article

information such as inter-article links and user information. This allows structured IE techniques to work, which often have a higher precision than unstructured techniques.

For these reasons, it is not surprising that several ontological IE approaches have targeted Wikipedia. This section will present three of them. For this purpose, let us first discuss the technical markup of Wikipedia.

### *2.1. Wikipedia*

Each Wikipedia article is a single Web page and usually describes a single topic or entity, the *article entity*. Figure 1 shows an excerpt of the article about Elvis Presley. As an online encyclopedia, Wikipedia has several characteristics: First, each article is highly interlinked to other articles. In the example, a click on the word *singer* leads to the Wikipedia article about singers. Each Wikipedia article has an unstructured, natural language part (on the left hand side in Figure 1). In addition to that, some Wikipedia articles also have an *infobox* (pictured on the right hand side). An infobox is a standardized table with information about the article entity. For example, there is an infobox template for people, which contains the birth date, the profession, and the nationality. Other widely used infobox templates exist for cities, music bands, companies etc. Each row in the infobox contains an attribute and a value. For example, our infobox contains the attribute *Born* with the value *January 8, 1935*. An infobox attribute may also have multiple values, as exemplified by the *Occupations* attribute. The majority of Wikipedia articles have been manually assigned to one or multiple *categories*. Our sample article is in the categories *American rock singers*, *1935 births*, and 34 more. Figure 1 shows an excerpt at the bottom.

Wikipedia is rendered as HTML pages, but is written in a special markup language, the *Wiki markup language*. Figure 2 shows an excerpt of the article on Elvis Presley in this language. The XML dump of Wikipedia (as of 2010) is approximately 27 Gigabytes large and stores the articles in the Wiki markup language.

We will now look at three systems that exploit this structured nature of Wikipedia: DBpedia, YAGO and KOG. Since these systems build an ontology, they can use automated reasoning to check and control the knowledge they are extracting. While DBpedia uses fewer reasoning, YAGO uses simple deductive reasoning and KOG uses the more sophisticated Markov logic [17].

```
{{Infobox musical artist
| Name = Elvis Presley
| Img = Elvis Presley 1970.jpg
| Born = {{birth date|1935|1|8|}}
| Occupation = [[singer]], [[actor]]
etc.
}}

'''Elvis Aaron Presley''' ([[January 8]], [[1935]] - [[August
16]], [[1977]]), middle name sometimes written '''Aron''', was
an [[United States|American]] [[singer]], [[musician]] and
[[actor]]. etc.

[[Category:1935 births]] [[Category:1977 deaths]] etc.
```

**Figure 2.** The Wikipedia Markup Language

### 2.2. DBpedia

DBpedia [2] is a system that extracts an RDF ontology from Wikipedia[3]. The project is a community effort, coordinated by the University of Leipzig, the Free University of Berlin and the OpenLink Software company. DBpedia was among the first projects to harvest Wikipedia at a large scale for ontology construction.

#### 2.2.1. Fact Extraction

The DBpedia system takes as input a dump of Wikipedia. The system creates one resource for each article in Wikipedia. For example, the Wikipedia page named "Elvis Presley" gives rise to the DBpedia resource `http://dbpedia.org/resource/Elvis_Presley`. The DBpedia extractors create basic facts about the resource by exploiting, among other things, the title of the article (which becomes the *rdfs:label* of the resource), the image of the article (which creates a triple linking the resource and the image URL) and the external links of the article (which each generate one triple). DBpedia applies this process to all language pages of Wikipedia, so that each resource in DBpedia has labels and abstracts in multiple languages (if Wikipedia has them).

DBpedia uses two approaches to harvest the infoboxes of Wikipedia. The first approach is *recall-oriented*. This means that it tries to harvest as much information as possible, even if this information may not be correctly extracted or contradictory. In this approach, every value of the infobox gives rise to one triple. The subject of the triple is the article resource. The relation of the triple is the infobox attribute, prefixed by `http://dbpedia.org/property/`. The object of the triple is the value in the infobox. For example, the line "Occupation = [[singer]]" in Elvis' infobox translates into the triple

```
<http://dbpedia.org/resource/Elvis_Presley,
http://dbpedia.org/property/Occupation,
http://dbpedia.org/resource/singer>
```

[3] `http://dbpedia.org`

The extractors can detect and handle lists, dates, numbers, and units as data types. This extraction technique covers all infoboxes and all infobox attributes. Unfortunately, Wikipedia sometimes uses different identifiers for the same relation. For example, some Wikipedia templates contain the attribute *occupation* while others contain *profession*. Therefore, the extraction algorithm creates many synonymous relations without an indication that they are synonymous.

Hence DBpedia also pursues a second approach to infobox harvesting [4]. This approach is *precision-oriented*, meaning that it extracts fewer information, but with higher quality. For this purpose, the DBpedia community has begun to map Wikipedia templates into an ontology. This ontology was created by manually arranging the 350 most commonly used infobox templates within the English edition of Wikipedia into a subsumption hierarchy. This hierarchy consists of 170 classes. The community then mapped the 2350 attributes of these templates to 720 manually defined ontology properties. The property mappings imclude fine-grained rules on how to parse infobox values and define target data types, which help the parsers to process attribute values. For instance, if a mapping defines the target data type to be a list of links, the parser will ignore additional text that might be present in the attribute values. This extraction provides cleaner data, but works only on the templates that have been manually identified. This applies to roughly half of the resources already.

### *2.2.2. Taxonomies*

DBpedia arranges its resources into 4 taxonomies. The first taxonomy is derived from the **Wikipedia Category** system. Every category of Wikipedia becomes a class, structured by *skos:broader* and *skos:narrower* relationships. The main advantage of the category system is that it is collaboratively extended and kept up-to-date by thousands of Wikipedia editors. A disadvantage is that categories do not form a proper topical hierarchy, as there are cycles in the category system and as categories often only represent a rather loose relatedness between articles. The second taxonomy is imported from the **YAGO** ontology. This taxonomy is described in detail in Section 2.3. While YAGO achieves a high accuracy in general, there are a few errors and omissions due to its automatic generation. A third taxonomy has been taken from the Upper Mapping and Binding Exchange Layer **UMBEL**. The UMBEL taxonomy was derived from OpenCyc. The fourth taxonomy is being developed manually by the **DBpedia community** from the most commonly used infobox templates within the English edition of Wikipedia. It consists of 170 classes that form a shallow subsumption hierarchy. It includes 720 properties with domain and range definitions. As all taxonomies use Wikipedia identifiers for the instances, they can all co-exist in parallel in DBpedia. See the PARIS project [19] for a comparison of the class and relation hierarchies of YAGO and DBpedia[4].

### *2.2.3. Summary*

As of February 2011, DBpedia contains 3.5 million resources and 670 million triples. The DBpedia data can be downloaded from the Web, but it can also be accessed through a SPARQL interface and through the Linked Data protocol [3]. This has made DBpedia a hub in the world of linked data. One of the main technical contributions of DBpedia is the systematic harvesting of infoboxes for fact extraction. This techniques give DBpedia

[4] `http://webdam.inria.fr/paris`

large coverage of practically all structured facts of Wikipedia. The entities of DBpedia are canonic, because DBpedia is based on Wikipedia, which contains very few duplicate entities. The community-based definition of extraction patterns ensures that more and more of the properties of DBpedia are also canonic. Through its 4 class hierarchies, DBpedia also gives an extensive taxonomic structure to its data.

DBpedia does not use reasoning to check its data for consistency. The YAGO project makes first steps in this direction.

### *2.3. YAGO*

YAGO[20] is an ontology[5] that has been automatically constructed from Wikipedia, WordNet [8], the Universal WordNet [7] and Geonames[6]. The project is being developed by the Max Planck-Institute for Informatics in Germany. YAGO was one of the first projects that used the extracted knowledge for consistency checks on other extracted knowledge.

#### *2.3.1. Fact Extraction*

Much like DBpedia, YAGO starts out from Wikipedia and makes every article a resource. For every article, YAGO also collects the categories of the article. It determines which of the categories identify a class to which the article entity belongs. For example, the category *American rock singers* identifies a class for the article entity *Elvis Presley*, while the category *Rock music* does not. YAGO creates a class for each of these category names and makes the article entity an instance of these classes. Then, YAGO links the classes to the corresponding classes in the WordNet taxonomy [8]. The class *American rock singers*, e.g., becomes a subclass of the WordNet class *singer*. YAGO harvests the infoboxes by help of a manual mapping from infobox attributes to properties [21]. YAGO also contains the Universal WordNet UWN [7]. UWN was derived from WordNet and translates its class names into up to 200 different languages. The YAGO extractors add in the classes and instances from Geonames, a dataset of geographical entities. Careful matching algorithms make sure that each Geonames class finds its place in the existing taxonomy and that each Geonames instance is mapped to the matching entity from Wikipedia, if it already exists. Thereby, YAGO is virtually free of duplicates and achieves canonicity of entities, classes and relations.

#### *2.3.2. Fact Checking*

In YAGO, every entity is assigned to at least one class. If YAGO cannot find a class for an entity, the entity is purged. This applies also to literals. YAGO contains a manually designed taxonomy for literals (the class *ThreeLetterLanguageCode*, e.g., is a subclass of *String*). Each literal class comes with a regular expression that identifies lexical forms of the data type. Furthermore, the relationships in YAGO are defined manually and have ranges and domains. This forces the extracted data into a rather rigid framework of classes and constraints. Within this framework, the YAGO extractors can perform a limited type of reasoning while extracting the facts:

[5] `http://yago-knowledge.org`
[6] `http://geonames.org`

1. **Functional Dependencies:** If a relation is known to be a function, only one object will be accepted for each subject
2. **Reductive Type Checking:** If a value extracted from an infobox does not meet the domain and range conditions of the relation, it is rejected. The type checking applies to resources (which have a type given by the taxonomy) as well as to literals (which can be type checked by the regular expression from the data type taxonomy).
3. **Type Coherence Checking:** At the top-level, the taxonomy of YAGO is partitioned into 5 different branches (locations, artifacts, people, other physical entities, and abstract entities). If an entity is an instance in multiple branches, a voting procedure is used to determine the only branch that will be accepted for that entity.

These mechanisms ensure that potentially erroneous triples are not accepted into YAGO.

### *2.3.3. Fact Deduction*

The YAGO extractors use reasoning not just to eliminate triples, but also to deduce new triples. YAGO implements a deduction mechanism for simple Horn rules. These rules are manually defined. The following rule, e.g., deduces that if an entity has an academic supervisor, then that entity must be a person:

$$\frac{<X, :hasAcademicSupervisor, Y>}{<X, rdf{:}type, :person>}$$

This mechanism, *Inductive Type Checking*, works for certain relations that identify the type of its argument unambiguously. Other rules deduce temporal and spatial information for facts. For example, YAGO knows that the relation *wasBornOnDate* indicates the temporal starting point of an entity. YAGO also knows that the relation *wasBornInPlace* happened at the time of the birth of the entity. Therefore, YAGO deduces that the time of the *wasBornInPlace* fact must be the date that comes with the *wasBornOnDate* fact. Horn rules are used to propagate time and space information from one fact to the other. As a result, YAGO can attach time intervals and geographic locations to many of its entities and facts, thus giving the data a temporal and a spatial dimension [11].

### *2.3.4. Summary*

YAGO contains 80 million triples about 10 million resources. This includes temporal and spatial information for both facts and entities. YAGO has been evaluated manually and shown to have an accuracy of 95% (with respect to Wikipedia as the ground truth). The main technical contribution of the project is the use of a basic reasoning framework to check extracted facts for consistency and to derive new facts. Through the manual definitions of properties and the matching algorithms, YAGO achieves canonicity of entities, classes, and relations. Through the fact checking mechanisms, YAGO achieves a certain consistency of the extracted data. By mapping Wikipedia categories to the classes of WordNet, YAGO gives its data a taxonomic backbone.

YAGO does not exploit that infobox templates can also yield class information. We are going to discuss next a project that uses the infobox templates to construct a taxonomy.

### 2.4. KOG

Th Kylin Ontology Generator (KOG) [25] is a system that builds an ontology from the infoboxes of Wikipedia and combines it with WordNet. KOG is part of the Machine Reading project at the University of Washington. KOG was the first system to treat Wikipedia infoboxes as classes and to infer their attributes and subsumption relations.

#### 2.4.1. Class Extraction

KOG starts out from the dump of Wikipedia. It treats each infobox template as a class. The attributes of the template are considered attributes of the class. KOG cleans the infobox data in 4 steps:

1. **Recognizing Duplicate Infobox Types**: Sometimes, the same class of things (e.g., US counties) are described by two types of infoboxes (e.g., *uscounty* and *us_county*). KOG uses the Wikipedia redirect pages, the Wikipedia editing history and name similarity heuristics to detect and merge equivalent infobox types.
2. **Assigning Meaningful Names**: Some infobox types have obscure names such as *abl*. KOG uses the Wikipedia redirect links, the Wikipedia categories of the respective articles and Google spell checking to find more meaningful names for the types (such as *AmericanBaseballLeague* instead of *abl*).
3. **Inferring Attribute Ranges**: KOG collects all values of a given attribute (e.g., all people who occur as values of the attribute *directedBy*). For each value, it determines the class of which the value is an instance. KOG uses the YAGO hierarchy and the DBpedia hierarchy [2] for this purpose. It chooses the superclass of the most frequent classes as the range for the attribute. For example, if most people that appear in a *directedBy* attribute are instances of the class *AmericanMovieDirectors* or the class *ItalianMovieDirectors*, KOG chooses the superclass *movieDirectors*.
4. **Attribute Mapping**: Some attributes are similar, even if they appear in different infobox types. For example, both the infoboxes for actors and the infoboxes for scientists have an attribute *spouse*. KOG uses string matching techniques, information from the taxonomy (see below) and data from the edit history of pages to identify such attributes.

#### 2.4.2. Taxonomy Construction

Next, KOG builds a taxonomy in 2 phases: First, KOG establishes which infobox template must be a subclass of which other infobox template. For example, KOG figures out that the template *EnglishPublicSchool* must be a subclass of *PublicSchool*. KOG uses name matching, pattern matching and information from the Wikipedia categories, the edit histories, and WordNet [8] for this purpose. Next, KOG connects the infobox classes to WordNet. For this purpose, KOG uses the mapping that exists already in YAGO [20] and the mapping that exists already in DBpedia [2], and trains a classifier to detect subsumption between a Wikipedia entity and a WordNet entity. KOG uses Markov Logic Networks [17] to take into account rules such as the transitivity of *subClassOf*. This produces a subsumption hierarchy of classes. Each class is canonic and has canonic attributes. The hierarchy is connected to the WordNet hierarchy.

### *2.4.3. Summary*

An evaluation of KOG yields precision rates around and beyond 90%. As of the time of this writing, the KOG ontology is not available online. One of the main technical contributions of the KOG system is the exploitation of infoboxes of Wikipedia as classes. These classes are anchored in WordNet, so that the KOG ontology possesses a strong taxonomic structure. KOG implements many techniques to make sure its classes, entities and relations are canonic. Unlike DBpedia and YAGO, KOG achieves this canonicity without manual rules. KOG also implements reasoning techniques to ensure the consistency of the extracted data.

KOG is highly tailored to the internal structure of Wikipedia. We will next look at systems that go beyond Wikipedia.

## 3. IE from the Web

While Wikipedia contains already much information, it contains only part of the huge amount of data that is available on the Web. This is why several newer approaches have embarked to go beyond Wikipedia, and to extract ontological information from the entire Web. The Web is much more heterogeneous than Wikipedia, with different file formats, different languages, different page layouts, and only creeping standardization. Furthermore, the information on the Web exhibits various degrees of credibility. Data may be faulty, incomplete, contradictory, or wrong. In addition, the Web is one of the largest computer-processable resources at all. This makes IE from the Web particularly challenging.

On the other hand, an IE algorithm can also benefit from the size of the Web: The redundancy of the Web means that the same piece of information is likely to appear multiple times in multiple forms. For example, the fact that Elvis won the Grammy Award is likely to appear several pages in several forms. If the algorithm manages to extract this piece of information from one page, it may be able to discover it subsequently on another page. This may allow it to learn a new textual pattern. This pattern, in turn, can then generate new facts. This mutual reinforcement of pattern discovery and fact discovery is a well-known principle in IE [5,1].

Due to the redundancy, the goal is no longer to extract all information from a given single document. Rather, the goal is to extract all information that is expressed in the documents, no matter from which document. This idea heralds a paradigm shift, which has been called *machine reading* [15] or *macro-reading* [6]. Quite a number of approaches have embarked in this direction. Some of the more prominent ones are the Read-the-Web project NELL [6] of Carnegie-Mellon University, the TextRunner/KnowItAll project[7] and the OntoUSP project at the University of Washington, and the PROSPERA/SOFIE project at the Max Planck Institute for Informatics. In this article, we will focus on OntoUSP, NELL, and SOFIE, as they aim to produce ontological output with canonicity, consistency, and a taxonomy in mind.

[7] `http://www.cs.washington.edu/research/knowitall/`

### 3.1. OntoUSP

OntoUSP[8] is a system that can extract information in an unsupervised way from natural language text[16]. OntoUSP is part of the Machine Reading project [15] at the University of Washington. OntoUSP can build up a knowledge base from the input documents and answer questions on it without using any training data.

#### 3.1.1. Fact Extraction

The goal of OntoUSP is to build a probabilistic knowledge base from a set of natural language documents. In this knowledge base, both entities and relationships will be represented as clusters of synonymous terms. For example, a protein called *IL-4* will be represented as a cluster of synonymous terms, such as *Protein IL-4*, *the cytokin interleukin-4* or *the IL-4 protein*. The relationship *enhance*, which holds between a protein and a gene, is represented as the cluster of the terms *enhances*, *induces* etc.

OntoUSP assumes that the documents have been parsed by a dependency parser (cf. Maynard and Bontcheva [13], this volume). Thus, OntoUSP takes as input a set of dependency trees. The dependency trees are first represented in a so-called *quasi-logical form* (QLF), i.e., in a first order logic conjunction: every node in the dependency tree becomes a constant; every edge becomes a binary predicate applied to the two node constants and every leaf becomes a unary predicate applied to the leaf node constant. For example, the sentence "IL4 induces CD11b" becomes

$$induces(n_1) \wedge subj(n_1, n_2) \wedge IL4(n_2) \wedge obj(n_1, n_3) \wedge CD11b(n_3)$$

Parts of such QLFs can be generalized to *lambda forms*, i.e., to sub-formulas where some constants are replaced by variables. In the example, we can extract the lambda form $\lambda x_2, x_3.induces(n_1) \wedge subj(n_1, x_2) \wedge obj(n_1, x_3)$. The OntoUSP algorithm groups lambda forms to *lambda form clusters*. A lambda form cluster is a set of semantically interchangeable lambda forms. For example, the cluster for "IL4 induces CD11b" will contain a cluster of synonymous lambda forms for *IL4* (such as *the IL-4 protein*), a cluster of synonymous forms for *induces* and a cluster of synonymous forms for *CD11b*. To build these clusters, OntoUSP uses three operators: MERGE (which merges two clusters to a larger cluster), ABSTRACT (which creates a super-cluster that contains two sub-clusters) and COMPOSE (which combines two forms to a sequence of forms, such as *the* with *protein* to *the protein*). OntoUSP uses a Markov Logic [17] to assign a cost to these operations and a hill-climbing algorithm to perform the clustering.

The result of this clustering is a probabilistic knowledge base, in which synonymous linguistic expressions are grouped together. The clusters are arranged in a taxonomic inclusion hierarchy. For example, OntoUSP can figure out that "induce" and "inhibit" are sub-clusters of the more general "regulate". OntoUSP can also abstract away the difference between active voice and passive voice. This allows OntoUSP to answer questions on the knowledge base, even if the question uses other terms than the original text.

[8] Ontological Unsupervised Semantic Parsing

### 3.1.2. Summary

OntoUSP has been run on a corpus of 2000 biomedical documents and evaluated with 2000 natural language questions. The system achieves a precision of 91%. The main technical contribution of OntoUSP is the unsupervised learning of a hierarchy of synonymous linguistic forms. The knowledge that OntoUSP builds up is *implicit* in the sense that it lacks explicit labels, logic constraints and crisp assignments of entities to classes. Still, OntoUSP exhibits a canonicity in the sense that it groups together synonymous names for one entity. OntoUSP also builds a hierarchical structure of the forms, thus inducing a type of taxonomy. However, this structure is not a class hierarchy in the classical sense, because it lacks explicit labels and crisp membership.

Due its implicit nature, OntoUSP cannot use reasoning to ensure the logical consistency of its knowledge base. We will discuss next a system that makes use of logical constraints to extract knowledge.

## 3.2. NELL

NELL[9] is a research project that aims to extract ontological information from the whole Web [6]. The project is driven by the Carnegie Mellon University. The NELL system aims to gather ontological information continuously on a large scale. It was launched in January 2010 and has been running ever since. The idea is that NELL shall not just constantly accumulate new knowledge, but also become better at it.

### 3.2.1. Fact Extraction

The goal of NELL is to create a set of facts (triples) together with a taxonomy. NELL uses as input a text corpus of several million documents. It also uses an initial ontology with predefined categories and binary relations. Each category comes with a predefined set of seed instances and each relation comes with a predefined set of seed pairs. For example, the category of *insects* is filled with 10 initial known insect names. The relation *playsFor* is filled with 10 initial pairs of soccer players and soccer teams. Furthermore, the system is also given a set of mutual-exclusion constraints. For example, the system knows that vehicles and people are disjoint. In addition, domain and range constraints are also supplied. The key idea of NELL is to use not just one extractor, but multiple extractors in parallel. Information gathered by one extractor is used to support the other extractors. This principle is called *coupled training*.

More precisely, NELL uses three learners: The first learner is the Coupled Pattern Learner (CPL), which extracts patterns and facts from natural language text. Given a document and given seed instances for categories, CPL extracts the contexts in which the seeds appear. For example, if it is known that Microsoft is a company, and if Microsoft appears in a sentence like "Bill Gates was the CEO of Microsoft", then the CPL will extract the pattern "CEO of X" as a pattern that identifies companies. If the CPL finds another sentence in which this pattern appears, it will propose the corresponding word as an instance of the category *company*. The CPL uses heuristics based on POS-tagging to determine the size of the patterns. The same idea applies to binary relationships: If a seed pair for a relation appears in a sentence, the CPL extracts the words between the two elements as a pattern. If this pattern appears again with two other elements, the

[9] Never-Ending Language Learner, `http://rtw.ml.cmu.edu/rtw/overview`

CPL proposes this pair of words as an instance of the relationship. The CPL uses some manually defined patterns for bootstrapping, which include the Hearst patterns [10].

The second learner is the Coupled SEAL (CSEAL), which is a modified version of SEAL [24]. CSEAL extracts patterns and facts in a similar way to CPL, but from structured documents. This means that the patterns that CSEAL learns are sequences of characters around the seed, which may include text, punctuation, HTML tags and HTML attributes. Just like CPL, CSEAL will learn new patterns from occurrences of seeds and use these patterns to propose new instances.

The third learner is the Meta-Bootstrap-Learner (MBL). The MBL runs in parallel with CPL and CSEAL and controls these extractors. For every instance that the extractors discover, MBL checks whether it satisfies the domain and range constraints and the exclusion constraints specified by the ontology. If that is not the case, MBL tells the extractor to filter out that instance. As a consequence, the extractor will downgrade the patterns that extracted this instance. Vice versa, MBL can also tell the extractor to promote an instance, if that instance was found by both extractors. As a consequence, the extractor will boost the patterns that extracted this instance. Through this coupling, MBL exploits the synergy of the two extractors and uses the results of one extractor to boost the performance of the other.

The NELL system has been continuously running since January 2010. The results undergo periodic screening and correction by a human. New categories and relations are also being added.

#### *3.2.2. Summary*

As of February 2011, NELL has accumulated a knowledge base of 500,000 asserted instances of 589 different categories and relations, at an estimated average precision of 87%. NELL is a live system and its progress can be followed online at the project homepage. The main technical contribution of the NELL project is the coupling of different extractors, which boost and control each other. Through the mutual exclusion rules and the type constraints, NELL can ensure a certain consistency of its results. Due to its reliance on Hearst Patterns, NELL is particularly strong on extracting *type* and *subclassOf* facts. This allows NELL to build a taxonomic structure.

Canonicity has not yet been a focus of the project. NELL does not work on ontological entities, but on strings, which may be synonymous or polysemous. We will next see a project that targets also this disambiguation of entities.

### *3.3. SOFIE*

SOFIE[10] is an IE system that aims to extend the YAGO ontology [20] with facts extracted from natural language documents [22]. It is part of the YAGO-NAGA project at the Max Planck Institute for Informatics in Germany. SOFIE aims to solve the canonicity problem, the pattern discovery and the consistency problem in one unified framework.

#### *3.3.1. Model*

In previous work, the problems of disambiguation, consistency reasoning and IE have mostly been attacked in isolation. However, these problems are highly intertwined. Con-

[10] Self-Organizing Framework for Information Extraction, `http://mpii.de/yago-naga/sofie`

sider, e.g., the sentence "Elvis died in 460 AD.". If the ontology already knows that Elvis Presley was born in 1935, then the word "Elvis" in the sample sentence cannot refer to Elvis Presley, because people have to be born before they die. This indicates that consistency constraints can guide disambiguation. Likewise, disambiguation can produce hints for pattern discovery and so on. Therefore, SOFIE aims to solve all three problems in one unified framework. This framework is based on the Weighted Maximum Satisfiability problem (*Weighted MAX SAT*). In this setting, all data is expressed as logic formulas.

SOFIE first describes the input documents as logic formulas. For every pair of proper names or numbers that appears in the text within a certain window, SOFIE generates a proposition of the form *patternOcc(Elvis@D1, "died in", 460)*. This proposition means that the word "Elvis" appeared with the string "died in" with the number 460. *Elvis@D1* identifies the word "Elvis" in document *D1*. Even though SOFIE does not yet know which entity is meant by "Elvis", SOFIE assumes that all occurrences of that word in the same document have the same meaning. Therefore, all those words are identified by the same token, a *wic* (word in context). So far, the generated proposition makes a statement only about the occurrence of certain strings. It does not yet make a statement about entities.

SOFIE extracts facts only about entities that are already known to the ontology. Furthermore, SOFIE assumes that the ontology knows all names that are commonly used to refer to an entity. This allows SOFIE to make hypotheses about the meaning of the wics. By taking into account the context of the wic (the bag of words in the document) and the context of a possible meaning of the wic (the names of the entities related to the entity in question in the ontology), SOFIE can assign a weight to these hypotheses. In the example, *Elvis@D1* could give rise to the following propositions (with weights):

*means(Elvis@D1, ElvisPresley)[0.8]*
*means(Elvis@D1, SaintElvis)[0.3]*

SOFIE also maps the ontology to logic formulas. In the case of YAGO, which just contains simple triples, every triple of YAGO becomes one proposition. For example, SOFIE will know *wasBornOnDate(ElvisPresley, 1935)*. SOFIE also makes use of semantic rules. These have to be supplied manually. One example for such a rule is

*bornOnDate(X,Y) and diedOnDate(X,Z)* $\Rightarrow Y < Z$ *(*)*

Other, freely configurable rules can enforce functional dependencies or incorporate domain knowledge. In addition, SOFIE uses rules that glue the framework together:

*patternOcc(WX, P, WY) and expresses(P, R)*
*and means(WX, X) and means(WY, Y)*
$\Rightarrow R(X, Y)$

This rule means: If pattern $P$ appears with wics $WX$ and $WY$ and $P$ expresses the YAGO relation $R$, and if $WX$ has been disambiguated to entity $X$, and $WY$ to $Y$, then $X$ and $Y$ stand in relationship $R$. In the example, assuming that "died in" is known to express *diedOnDate*, we could deduce *diedOnDate(ElvisPresley,460)*. An analogous rule says that if a pattern $P$ occurs with two entities, and if the ontology knows that the entities stand in relationship $R$, then $P$ expresses $R$. For example, if YAGO knows that Einstein died in 1955, then a sentence of the form "Einstein died in 1955" (with appropriate disambiguation) will trigger the deduction of *expresses("died in",diedOnDate)*.

### 3.3.2. Reasoning

All of the above formulas are grounded, i.e., one creates one instance of the formula for every possible instantiation of the variables. All formulas that do not have an explicit weight are given a large weight $W$. This gives SOFIE a huge set of weighted propositional formulas. These formulas can never all be true at the same time. *means*, for example, is declared a function, but already has two images for *Elvis@D1*. Furthermore, the deduction of pattern meanings may be overly hasty and wrong. Lastly, the documents themselves may contain contradictory information. Therefore, SOFIE aims to find an assignment of truth values to the propositions that maximizes the weight of the satisfied formulas. In the example, assigning *true* to the proposition *means(Elvis@D1, ElvisPresley)* will gain a weight of 0.8. At the same time, this assignment will make it impossible to satisfy rule (*). Therefore, SOFIE might decide to make *means(Elvis@D1, SaintElvis)* true instead and to deduce *diedOnDate(SaintElvis, 460)*. Finding the optimal assignment of truth values to the propositions is an NP-hard problem. SOFIE implements a heuristic algorithm that runs in polynomial time and has an approximation guarantee of 1/2.

The optimal solution of the Weighted MAX SAT problem corresponds to the most plausible interpretation of the documents, given the background knowledge. SOFIE can continuously analyze new documents and compute the most plausible interpretation of the evidence at any time. As more evidence accumulates, the most plausible interpretation of it may change over time.

### 3.3.3. Summary

SOFIE has been run on Web documents and has been shown to have a precision of 90%. This includes both the correctness of the facts and the correct disambiguation of entities. The main technical contribution of SOFIE is the modeling of disambiguation, pattern deduction and consistency in one single framework. This allows SOFIE to exploit synergies between these three tasks. Through its automated reasoning, SOFIE achieves both canonicity of the extracted facts and consistency with the ontology. SOFIE does not find new classes. It re-uses the existing taxonomy of YAGO. SOFIE has been extended to the PROSPERA system [14] to run in a parallelized way on large scale.

## 4. Conclusion

In this chapter, we have discussed IE for ontologies. Three main desiderata for ontological IE are the canonicity of entities and relationships, the consistency of the extracted facts and the taxonomic organization of the data. We have seen 3 approaches that extract information from Wikipedia with these goals in mind: DBpedia, YAGO and KOG. These systems can already use different degrees of reasoning to ensure the consistency of the extracted data. We have also discussed ontological IE from the Web. Such approaches can use the redundancy of the Web to mutually reinforce pattern discovery and fact discovery. We have seen 3 such systems, OntoUSP, NELL, and SOFIE. All of them use some degree of reasoning to ensure logical consistency of the extracted facts.

The rapid growth of the Web is drawing more and more attention to the “semantic gap” between syntactic text and semantic knowledge. At the same time, the availability of large scale knowledge bases (such as for example those on the Web of Linked Data

[3]) opens up new ways of using structured information to deal with unstructured text. These two trends have nurtured a strong interest in ontological IE. Today, numerous research groups work on making the Web semantically accessible. Newer IE approaches go beyond fact extraction and extract entire axioms. For example, IE can extract entire OWL axioms from the first sentences of Wikipedia articles [23].

## References


[1] Eugene Agichtein, Luis Gravano, Jeff Pavel, Viktoriya Sokolova, and Aleksandr Voskoboynik. Snowball: a prototype system for extracting relations from large text collections. *SIGMOD Records*, 30(2):612, 2001.

[2] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary G. Ives. DBpedia: A Nucleus for a Web of Open Data. In *Proceedings of the International Semantic Web Conference (ISWC)*, volume 4825 of *Lecture Notes in Computer Science*, pages 722–735, Berlin, Germany, 2007. Springer.

[3] Christian Bizer, Tom Heath, Kingsley Idehen, and Tim Berners-Lee. Linked data on the Web. In *Proceedings of the International Conference on World Wide Web (WWW)*, pages 1265–1266, New York, NY, USA, 2008. ACM.

[4] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. Dbpedia - a crystallization point for the web of data. *Web Semant.*, 7:154–165, September 2009.

[5] Sergey Brin. Extracting patterns and relations from the world wide web. In *Selected papers from the International Workshop on the WWW and Databases*, pages 172–183, London, UK, 1999. Springer.

[6] Andrew Carlson, Justin Betteridge, Richard C. Wang, Estevam R. Hruschka Jr., and Tom M. Mitchell. Coupled semi-supervised learning for information extraction. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining (WSDM 2010)*, 2010.

[7] Gerard de Melo and Gerhard Weikum. Towards a universal wordnet by learning from combined evidence. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM 2009)*, pages 513–522, New York, NY, USA, 2009. ACM.

[8] C. Fellbaum, editor. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

[9] Dayne Freitag and Nicholas Kushmerick. Boosted wrapper induction. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 577–583, Menlo Park, CA, USA, 2000. AAAI Press.

[10] Marti A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the International Conference on Computational Linguistics (ICCL)*, pages 539–545. Association for Computational Linguistics, 1992.

[11] Johannes Hoffart, Fabian M. Suchanek, Klaus Berberich, and Gerhard Weikum. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artif. Intell.*, 194:28–61, 2013.

[12] Girija Limaye, Sunita Sarawagi, and Soumen Chakrabarti. Annotating and searching web tables using entities, types and relationships. *Proceedings of the International Conference on Very Large Databases (VLDB)*, 3:1338–1347, September 2010.

[13] Diana Maynard and Kalina Bontcheva. Natural language processing. In Johanna Völker and Jens Lehmann, editors, *Perspectives of Ontology Learning*, Studies on the Semantic Web. AKA Heidelberg / IOS Press, 2012.

[14] Ndapandula Nakashole, Martin Theobald, and Gerhard Weikum. Scalable knowledge harvesting with high precision and high recall. In *Proceedings of the fourth ACM international conference on Web search and data mining*, WSDM '11, pages 227–236, New York, NY, USA, 2011. ACM.

[15] Hoifung Poon, Janara Christensen, Pedro Domingos, Oren Etzioni, Raphael Hoffmann, Chloe Kiddon, Thomas Lin, Xiao Ling, Mausam, Alan Ritter, Stefan Schoenmackers, Stephen Soderland, Dan Weld, Fei Wu, and Congle Zhang. Machine reading at the university of washington. In *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading*, FAM-LbR '10, pages 87–95, 2010.

[16] Hoifung Poon and Pedro Domingos. Unsupervised ontology induction from text. In *Conference of the Association for Computational Linguistics (ACL)*, pages 296–305, 2010.

[17] Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine Learning*, 62(1-2), 2006.

[18] Sunita Sarawagi. Information Extraction. *Foundations and Trends in Databases*, 2(1), 2008.

[19] Fabian M. Suchanek, Serge Abiteboul, and Pierre Senellart. PARIS: Probabilistic Alignment of Relations, Instances, and Schema. *Proceedings of the International Conference on Very Large Databases (VLDB)*, 5(3):157–168, 2011.

[20] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. YAGO: A core of semantic knowledge - unifying WordNet and Wikipedia. In Carey L. Williamson, Mary Ellen Zurko, and Prashant J. Patel-Schneider, Peter F. Shenoy, editors, *Proceedings of the International Conference on World Wide Web (WWW)*, pages 697–706, Banff, Canada, 2007. ACM.

[21] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. YAGO - A Large Ontology from Wikipedia and WordNet. *Elsevier Journal of Web Semantics*, 6(3):203–217, September 2008.

[22] Fabian M. Suchanek, Mauro Sozio, and Gerhard Weikum. SOFIE: A Self-Organizing Framework for Information Extraction. In *International World Wide Web conference (WWW 2009)*, New York, NY, USA, 2009. ACM Press.

[23] Johanna Völker, Pascal Hitzler, and Philipp Cimiano. Acquisition of OWL DL axioms from lexical resources. In Enrico Franconi, Michael Kifer, and Wolfgang May, editors, *Proceedings of the 4th European Semantic Web Conference (ESWC)*, volume 4519 of *Lecture Notes in Computer Science*, pages 670–685. Springer, JUN 2007.

[24] Richard C. Wang and William W. Cohen. Character-level analysis of semi-structured documents for set expansion. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3 - Volume 3*, EMNLP '09, pages 1503–1512, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[25] Fei Wu and Daniel S. Weld. Automatically refining the Wikipedia infobox ontology. In *Proceedings of the International Conference on World Wide Web (WWW)*, pages 635–644, New York, NY, USA, 2008. ACM.

# Empirically Grounded Emergent Knowledge

Vít NOVÁČEK [a] and Siegfried HANDSCHUH [a]
[a] *Digital Enterprise Research Institute, National University of Ireland Galway*
*IDA Business Park, Lower Dangan, Galway, Ireland*
*e-mail: vit.novacek@deri.org*

**Abstract.** This chapter introduces a framework for representation and processing of emergent knowledge. By emergent knowledge we mean most primarily semantic patterns discovered within simple statements automatically extracted from textual resources. Empirical grounding of the emergent knowledge is achieved by applying the principles of distributional semantics in order to anchor the discovered semantic patterns in the textual data in a well-founded and non-arbitrary manner. We also propose a method for seamless combination of the distributional (bottom-up) and symbolic (top-down) aspects of the semantics of emergent knowledge. Broad applicability of our approach is showcased within several practical scenarios in the domain of life sciences.

**Keywords.** Empirical Knowledge, Emergent Knowledge, Distributional Semantics

## Introduction

The vast realms of the web encompass a substantial portion of the human knowledge nowadays. However, the particular pieces of the knowledge are interleaved with a lot of noise and scattered among many information resources of various relevance. Thus it is often difficult or even infeasible to get what one needs to know from the ballast of largely irrelevant content. Our goal is to contribute to giving more meaning to the content on the web, identifying and interlinking relevant knowledge out there so that machines can help humans to make use of it more efficiently. We are particularly interested in giving a clear empirical grounding to the machine-processable meaning of the web resources. To do so, we have investigated an extension of a general corpus semantics framework [2] and applied it in the scope of continual ontology learning.

Essentially, we show how to represent statements extracted from the web in a corpus-like structure, and how to glean more complex, emergent knowledge (e.g., implicit relationships between similar entities and properties, or rules) from the simple extracted pieces. The proposed corpus-like storage allows for inferring more complex emergent knowledge from the data by means of various statistical and algebraic methods of analysis known from computational linguistics and information retrieval. This way the knowledge we consequently operate with is empirically grounded in the data in a well-founded and non-arbitrary manner.

In addition to the bottom-up knowledge acquisition process, we propose a symbolic layer on the top of the distributional core, which allows for a straightforward application of rule-based reasoning on the top of the emergent knowledge bases. Thus we combine the benefits of bottom-up and top-down approaches to semantics in one framework for continuous knowledge acquisition. To showcase the practical relevance of the introduced framework, we give an overview of its three different applications to various problems related to information overload in life sciences.

The rest of the chapter is organised as follows. Section 1 presents an overview of the related work. The general principles of the proposed framework are elaborated in Section 2. Specific applications are described in Section 3, including summaries of corresponding evaluation experiments. Finally, we sum up the chapter in Section 4.

## 1. Related Work

Most of the ontology learning approaches (see [16] or [4] for an overview) focus on methods for extraction of particular ontological constructs (e.g., sub-class-of, instance-of or disjoint-with relationships) from unstructured natural language texts. Our general approach is more light-weight and modular – we are more interested in a layered representation of various aspects of knowledge emerging from the web/linked data (c.f. [35] in this volume) and/or texts. This involves rather simple information about co-occurrence of entities in various types of inputs, general patterns gleaned from the inputs, and a perspective enabling rule-based reasoning with the learned content. Thus, instead of focusing on the advancement of particular methods for knowledge extraction, we provide a general coherent framework for representation and processing of various emergent knowledge aspects which allows for straightforward incorporation of applicable state of the art methods whenever necessary.

As our approach is focused on bottom-up and emergent knowledge acquisition, it is clearly related to works like [30], [14], [27] or [5]. However, the method [30] focuses more on ontology merging than on knowledge acquisition (although instances extracted from related documents are employed in the merging process). The combination of heterogeneous sources for taxonomy acquisition by means of machine learning presented in [5] is conceptually similar to our incremental and synergetic approach to knowledge acquisition. However, we are not focused on a single semantic phenomenon (i.e., taxonomy) and want rather to provide a general framework for extraction and analysis of emergent semantics. Perspectives of emergent semantics for learned ontologies are outlined in [14], which proposes a general bottom-up methodology for emergent knowledge maintenance with no particular technical contribution, though. Clustering-based emergent construction of ontologies from text managed by a multi-agent self-organising system is investigated in [27]. We complement the approaches to emergent ontology evolution [14,27] by providing a framework for distributional representation and analysis of extracted knowledge augmented by a symbolic inference layer, which allows for more uniform and sophisticated consequent processing and utilisation. The combination of ontology learning and reasoning that is targeted by our approach (among other things) is related to works like [8] and [9]. However, [8] deals much rather with the resolution of inconsistencies in learned knowledge and consequent essentially trivial Description Logics inference, while [9] requires rather lossy translation of ontologies into first-order

formulas, which are, moreover, often relatively simple. We offer more general and customisable rule-based inference operating on the top of the extracted knowledge.

Perhaps the closest inspiration of our approach is the general distributional framework for corpus-based semantics [2]. However, we generalise and extend the tensor-based representation of weighed co-occurrence relations between natural language expressions proposed in [2] to reflect provenance and also possible contexts of the basic statements. Moreover, we provide novel methods of smooth combination of the distributional and symbolic semantic levels in order to allow for automated formal reasoning about the empirically grounded knowledge emerging from the web.

There are additional works related to the particular applications of our general approach. These are beyond the scope of this overview chapter, though, and are dealt with thoroughly in our more technical publications, namely in [22], [26] and [21].

## 2. General Framework

The proposed distributional framework for grounded emergent knowledge has two layers – the bottom-up and top-down one. The former caters for the implicit meaning, while the latter allows for adding more value to the bottom-up analysis by utilising the current Semantic Web resources (e.g., RDF Schema or ontologies). A general way of using the framework follows this pipeline: (1) convert a set of simple RDF documents (e.g., Semantic Web graphs, or triples extracted from resources like natural language texts or databases) into the internal distributional representation; (2) extract interesting patterns from it; (3) make use of extant top-down semantic resources to materialise more implicit knowledge by means of inference (optional); (4) utilise the results to improve the quality of the initial RDF data set. The last step can consist of exporting the distributional patterns as RDF statements to be added to the input data (e.g., as links between the entities or properties found to be similar). Alternatively, one can present the patterns directly to users along the original data set to facilitate its machine-aided augmentation.

### *2.1. Bottom-Up Layer*

This layer serves for grounding complex semantic phenomena in the underlying data. In particular, it provides compact structures for representation of simple statements and their provenance, as well as for analysis of knowledge patterns (e.g., implicit conceptual relationships or rules) emerging from the basic level.

#### *2.1.1. Source Representation*

The basic structure of the bottom-up layer is a so called source (or graph) representation $\mathbf{G}$, which captures the co-occurrence of things (i.e., subjects and objects) within relations (i.e., predicates) across a set of documents (i.e., RDF graphs). Let $A_l, A_r$ be sets representing left and right arguments of binary co-occurrence relationships (i.e., statements), and $L$ the types of the relationships. $A_l, A_r, L$ correspond to sets of RDF subjects, objects and predicates, respectively. Furthermore, let $P$ be a set representing provenances of particular relationships (i.e., graph names). We define the source representation as a 4-ary labeled tensor $\mathbf{G} \in \mathbb{R}^{|A_l| \times |L| \times |A_r| \times |P|}$. It is a four-dimensional array structure indexed by subjects, predicates, objects and provenances, with values reflecting a fre-

quency or weight of statements in the context of particular provenance sources (0 if a statement does not occur in a source). For instance, if a statement $(a_l, l, a_r)$ occurs $k$-times in a data source $d$ (a single graph or a set of graphs in general), then the element $g_{a_l,l,a_r,d}$ of $\mathbf{G}$ will be set to $k$ to reflect it. More details are illustrated in the following.

**Example 1** *Let us consider* 7 *statements (acquired from biomedical texts):*

$(protein\ domain, different, protein)$, $(protein\ domain, type, domain)$, $(gene, different, protein)$, $(internal\ tandem\ duplications, type, mutations)$, $(internal\ tandem\ duplications, in, juxtamembrane)$, $(internal\ tandem\ duplications, in, extracelullar\ domains)$, $(protein\ domain, type, domain)$

*with provenances* $D_1, D_1, D_2, D_3, D_3, D_3, D_4$*, respectively. The source representation (using statement occurrence frequencies as values) is:*

| $s \in A_l$ | $p \in L$ | $o \in A_r$ | $d \in P$ | $g_{s,p,o,d}$ |
|---|---|---|---|---|
| $protein\ domain$ | $different$ | $protein$ | $D_1$ | 1 |
| $protein\ domain$ | $type$ | $domain$ | $D_1$ | 1 |
| $gene$ | $different$ | $protein$ | $D_2$ | 1 |
| $internal\ tandem\ duplications$ | $type$ | $mutations$ | $D_3$ | 1 |
| $internal\ tandem\ duplications$ | $in$ | $juxtamembrane$ | $D_3$ | 1 |
| $internal\ tandem\ duplications$ | $in$ | $extracelullar\ domains$ | $D_3$ | 1 |
| $protein\ domain$ | $type$ | $domain$ | $D_4$ | 1 |

*We omit all zero values and use the tabular notation as a convenient and concise representation of a 4-dimensional tensor, with the three first columns for indices and the fourth one for the corresponding value.*

### *2.1.2. Corpus Representation*

The source tensor is merely a low-level data representation preserving the association of statements with their provenance contexts. Before allowing for actual distributional analysis, the data have to be transformed into a more compact structure $\mathbf{C}$ called corpus representation. $\mathbf{C} \in \mathbb{R}^{|A_l| \times |L| \times |A_r|}$ is a ternary (three-dimensional) labeled tensor, devised according to [2] in order to provide for a universal and compact distributional representation for the proposed bottom-up web semantics framework. A corpus $\mathbf{C}$ can be constructed from a source representation $\mathbf{G}$ using functions $a : \mathbb{R} \times \mathbb{R} \to \mathbb{R}, w : P \to \mathbb{R}, f : A_l \times L \times A_r \to \mathbb{R}$. For each $\mathbf{C}$ element $c_{s,p,o}$, $c_{s,p,o} = a(\sum_{d \in P} w(d) g_{s,p,o,d}, h(s,p,o))$, where $g_{s,p,o,d}$ is an element of the source tensor $\mathbf{G}$ and the $a, f, w$ functions act as follows: (1) $w$ assigns a relevance degree to each source; (2) $f$ reflects the relevance of the statement elements (e.g., a mutual information score of the subject and object within the sources); (3) $a$ aggregates the result of the $w, f$ functions' application. This way of constructing the elements of the corpus tensor from the low-level source representation essentially aggregates the occurrences of statements within the input data, reflecting also two important things – the relevance of particular sources (via the $w$ function), and the relevance of the statements themselves (via the $f$ function). The specific implementation of the functions is left to applications – possible examples include (but are not limited to) ranking (both at the statement and document level) or statistical analysis of the statements within the input data.

**Example 2** *A corpus corresponding to the source tensor from Example 1 can be represented (again in a tabular notation) as given below. The* $w$ *values were* 1 *for all sources and* $a, f$ *aggregated the source values using relative frequency (in a data set containing* 7 *statements).*

| $s \in A_l$ | $p \in L$ | $o \in A_r$ | $c_{s,p,o}$ |
|---|---|---|---|
| $protein\ domain$ | $different$ | $protein$ | $1/7$ |
| $protein\ domain$ | $type$ | $domain$ | $2/7$ |
| $gene$ | $different$ | $protein$ | $1/7$ |
| $internal\ tandem\ duplications$ | $type$ | $mutations$ | $1/7$ |
| $internal\ tandem\ duplications$ | $in$ | $juxtamembrane$ | $1/7$ |
| $internal\ tandem\ duplications$ | $in$ | $extracelullar\ domains$ | $1/7$ |

### *2.1.3. Corpus Perspectives*

The elegance and power of the corpus representation lays in its compactness and universality that, however, yields for many diverse possibilities of the underlying data analysis. The analysis are performed using a process of so called matricisation of the corpus tensor $\mathbf{C}$. Essentially, matricisation is a process of representing a higher-order tensor using a 2-dimensional matrix perspective. This is done by fixing one tensor index as one matrix dimension and generating all possible combinations of the other tensor indices within the remaining matrix dimension. In the following we illustrate the process on the simple corpus tensor from Example 2. Detailed description of matricisation and related tensor algebra references can be found in [2].

**Example 3** *When fixing the subjects ($A_l$ set members) of the corpus tensor from Example 2, one will get the following matricised perspective (the rows and columns with all values equal to zero are omitted here and in the following examples):*

| $s/\langle p,o\rangle$ | $\langle d,p\rangle$ | $\langle t,dm\rangle$ | $\langle t,m\rangle$ | $\langle i,j\rangle$ | $\langle i,e\rangle$ |
|---|---|---|---|---|---|
| $protein\ domain$ | $1/7$ | $2/7$ | $0$ | $0$ | $0$ |
| $gene$ | $1/7$ | $0$ | $0$ | $0$ | $0$ |
| $internal\ tandem\ duplications$ | $0$ | $0$ | $1/7$ | $1/7$ | $1/7$ |

.

*The abbreviations $d, p, t, dm, m, i, j, e$ stand for $different$, $protein$, $type$, $domain$, $mutations$, $in$, $juxtamembrane$, $extracellular\ domains$. One can clearly see that the transformation is lossless, as the original tensor can be easily reconstructed from the matrix by appropriate re-grouping of the indices.*

The corpus tensor matricisations correspond to vector spaces consisting of elements defined by particular rows of the matrix perspectives. Each of the vectors has a name (the corresponding matrix row index) and a set of features (the matrix column indices). The features represent the distributional attributes of the entity associated with the vector's name – the contexts aggregated across the whole corpus. Thus by comparing the vectors, one essentially compares the meaning of the corresponding entities emergently defined by the underlying data. For exploring the matricised perspectives, one can uniformly use the linear algebra methods that have been successfully applied to vector space analysis tasks for the last couple of decades. Large feature spaces can be reliably reduced to a couple of hundreds of the most significant indices by techniques like singular value decomposition or random indexing Vectors can be compared in a well-founded manner by various metrics or by the cosine similarity. See [6] for an overview of vector space semantic models, distances, similarities and singular value decomposition, one of the most used matrix decomposition techniques. Using these techniques, matrix perspectives can be combined with vector space analysis techniques in order to investigate a wide range of semantic phenomena related to synonymy, clustering, ambiguity resolution, taxonomy detection or analogy discovery. The following example illustrates how to perform clustering of similar entities and properties (more examples come later in Section 3).

**Example 4** *Let us add two more matrix perspectives to the $s/\langle p, o\rangle$ one provided in Example 3. The first one represents the distributional features of objects (based on the contexts of predicates and subjects they tend to co-occur with in the corpus):*

| $o/\langle p, s\rangle$ | $\langle d, pd\rangle$ | $\langle t, pd\rangle$ | $\langle d, g\rangle$ | $\langle t, itd\rangle$ | $\langle i, itd\rangle$ |
|---|---|---|---|---|---|
| $protein$ | $1/7$ | $0$ | $1/7$ | $0$ | $0$ |
| $domain$ | $0$ | $2/7$ | $0$ | $0$ | $0$ |
| $mutations$ | $0$ | $0$ | $0$ | $1/7$ | $0$ |
| $juxtamembrane$ | $0$ | $0$ | $0$ | $0$ | $1/7$ |
| $extracellular\ domains$ | $0$ | $0$ | $0$ | $0$ | $1/7$ |

.

*$d, pd, t, g, itd, i$ stand for different, protein domain, type, gene, internal tandem duplications, in. Similarly, the second perspective represents the distributional features of properties:*

| $p/\langle s, o\rangle$ | $\langle pd, p\rangle$ | $\langle pd, d\rangle$ | $\langle g, p\rangle$ | $\langle itd, m\rangle$ | $\langle itd, j\rangle$ | $\langle itd, ed\rangle$ |
|---|---|---|---|---|---|---|
| $different$ | $1/7$ | $0$ | $1/7$ | $0$ | $0$ | $0$ |
| $type$ | $0$ | $2/7$ | $0$ | $1/7$ | $0$ | $0$ |
| $in$ | $0$ | $0$ | $0$ | $0$ | $1/7$ | $1/7$ |

.

*$itd, pd, p, d, g, m, j, ed$ stand for internal tandem duplications, protein domain, protein, domain, gene, mutations, juxtamembrane, extracellular domains.*

*The vector spaces induced by the matrix perspectives $s/\langle p, o\rangle$ and $o/\langle p, s\rangle$ can be used for finding similar entities by comparing their corresponding vectors. Using the cosine vector similarity, one finds that $sim_{s/\langle p,o\rangle}(protein\ domain, gene) = \frac{\frac{1}{7}\frac{1}{7}}{\sqrt{(\frac{1}{7})^2+(\frac{2}{7})^2}\sqrt{(\frac{1}{7})^2}} \doteq 0.2972$ and $sim_{o/\langle p,s\rangle}(juxtamembrane, extracel\text{-}lular\ domains) = \frac{\frac{1}{7}\frac{1}{7}}{\sqrt{(\frac{1}{7})^2}\sqrt{(\frac{1}{7})^2}} = 1$. These are the only non-zero similarities among the subject and object entities present in the corpus. As for the predicates, all of them have a zero similarity. This quite directly corresponds to the intuition a human observer can get from the data represented by the initial statements from Example 1. Protein domains and genes seem to be different from proteins, yet protein domain is a type of domain and gene is not, therefore they share some similarities but are not completely equal according to the data. Juxtamembranes and extracellular domains are both places where internal tandem duplications can occur, and no other information is available, so they can be deemed equal until more data comes. Among the particular predicates, no patterns as clear as for the entities can be observed, therefore they can be considered rather dissimilar given the current data.*

### *2.2. Top-Down Layer*

This layer serves for application of extant semantic resources (like manually designed ontologies or rule bases) to the uncertain representations of the bottom-up layer. A significant portion of the expressive Semantic Web standards (RDFS, OWL) and widely used extensions (such as N3 [1]) can be expressed by conjunctive rules [2]. To allow for a seamless combination of this top-down layer of the Semantic Web with the bottom-up principles introduced in the previous section, we propose a straightforward adaptation of state of the art rule-based reasoning methods.

Conjunctive rules can be described as follows in the 'language' of the bottom-up semantics. Let $\mathcal{S} = \mathbb{R}^{|A_l \cup V| \times |L \cup V| \times |A_r \cup V|}$ be a set of corpus tensors with their index domains $(A_l, L, A_r)$ augmented by a set of variables $V$. Then $(\mathbf{L}, \mathbf{R}, w)$, where $\mathbf{L}, \mathbf{R} \in$

[1] See `http://www.w3.org/DesignIssues/Notation3.html` for details.

[2] See `http://www.w3.org/TR/rdf-mt/`, `http://www.w3.org/TR/owl2-profiles/` or [31].

$\mathcal{S}, w \in \mathbb{R}$, is a rule with an antecedent $\mathbf{L}$, a consequent $\mathbf{R}$ and a weight $w$. The values of the rule tensors are intended to represent the structure of the rule statements – a non-zero value reflects the presence of a statement consisting of the corresponding indices in the rule. However, the antecedent tensor values can also specify the weights of the relationship instances to be matched and thus facilitate uncertain rule pattern matching. The weights can be used to set relative importance of rules. This is especially useful when combining rules from rule sets of variable relevance – one can assign higher weights to rule coming from more reliable resources and the other way around. We assume the weights to be set externally – if this is not the case, they are assumed to be 1 by default.

**Example 5** *An RDFS entailment rule for transitivity can be stated in N3 as:* `{?x rdfs:subClassOf ?y . ?y rdfs:subClassOf ?z } => {?x rdfs:subClassOf ?z }.` *The rule is transformed to the tensor form as:*

$$\left( \begin{array}{|c|c|c||c|} \hline s \in A_l \cup V & p \in L \cup V & o \in A_r \cup V & l_{s,p,o} \\ \hline ?x & rdfs:subClassOf & ?y & 1 \\ \hline ?y & rdfs:subClassOf & ?z & 1 \\ \hline \end{array}, \begin{array}{|c|c|c||c|} \hline s \in A_l \cup V & p \in L \cup V & o \in A_r \cup V & r_{s,p,o} \\ \hline ?x & rdfs:subClassOf & ?z & 1 \\ \hline \end{array}, 1 \right).$$

Rules can be applied to a corpus by means of Algorithm 1. The particular rule-based

**Algorithm 1** Rule Evaluation

1: $RESULTS \leftarrow \emptyset$
2: $FOREST \leftarrow conditionTrees(\mathcal{R})$
3: **for** $T \in FOREST$ **do**
4: **for** $(I, \mathbf{R}, w) \in matches(T)$ **do**
5: $\mathbf{R}' \leftarrow w \cdot materialise(I, \mathbf{R})$
6: $RESULTS \leftarrow RESULTS \cup \mathbf{R}'$
7: **end for**
8: **end for**
9: **return** $\sum_{\mathbf{X} \in RESULTS} \mathbf{X}$

reasoning method we currently use is a modified version of the efficient RETE algorithm for binary predicates [7]. The $conditionTrees()$ function in Algorithm 1 generates a set of trees of antecedent conditions from a rule set $\mathcal{R}$.

**Example 6** *For instance, let us imagine the following rule set (described in N3 again):* $R_1$ : `{?x rdfs:subClassOf ?y . ?y rdfs:subClassOf ?z } => {?x rdfs:subClassOf ?z}.` $R_2$ `:{?x rdfs:subClassOf ?y . ?z rdf:type ?x} => {?z rdf:type ?y}.` *For simplicity, we assume the weights of the rules* $R_1, R_2$ *to be* 1.0. *Given this rule set, the* $conditionTrees()$ *function returns a single tree with a root condition* `?x rdfs:subClassOf ?y` *and the* `?y rdfs:subClassOf ?z, ?z rdf:type ?x` *conditions as the root's children. The tree leafs (i.e., children of the root's children) then point to the consequents and weights of the rules* $R_1, R_2$*, respectively.*

The rule condition forest allows for optimised incremental generation of all possible corpus instance assignments to the variables in the rule conditions – each condition is being evaluated only once even if it occurs in multiple rules. The generation of instance assignments for particular condition variables is realised by the function $matches()$ in

Algorithm 1. It produces tuples $(I, \mathbf{R}, w)$, where $I$ is an assignment of instances to the antecedent variables along a particular root-leaf path in the given tree $T$. $\mathbf{R}, w$ are then the rule consequent and weight in the leaf of the corresponding instance assignment path.

The function $materialise()$ takes the computed instance assignment $I$ and applies it to the consequent $\mathbf{R}$. The values of the materialised consequent tensor $\mathbf{R}'$ are computed as $r_{s,p,o} = \top\{c_{i_1,i_2,i_3} | (i_1, i_2, i_3) \in I\}$ for each $(s, p, o)$ element of the consequent that has a non-zero value in the original $\mathbf{R}$ tensor. The $c_{i_1,i_2,i_3}$ elements of the tensor $\mathbf{C}$ (the corpus representation, i.e., knowledge base) correspond to all statements in the instantiated rule conditions along the assignment path $I$. Finally, the $\top$ operation is an application of a fuzzy conjunction (t-norm) to a set of values[3]. The result of Algorithm 1 is a sum of all materialised tensors weighted by the corresponding rule weights.

**Example 7** *To exemplify an iterative rule materialisation (knowledge base closure), let us add two more elements to the corpus from Example 2 (the weights are purely illustrative):*

| $A_l$ | $L$ | $A_r$ | *value* |
|---|---|---|---|
| $domain$ | $rdfs:subClassOf$ | $molecular\ structure$ | $2/9$ |
| $molecular\ structure$ | $rdfs:subClassOf$ | $building\ block$ | $1/9$ |

*If we assume that the* `type` *relation from the previous examples is equivalent to the* `rdf:type` *relation from the rule $R_2$ in Example 6, we can apply the $R_1, R_2$ rules to the extended corpus representation with the following results. After assigning instances to the antecedent variables, the only instance path leading towards $R_1$ in the condition tree consists of the statements* `domain rdfs:subClassOf molecular structure` *and* `molecular structure rdfs:subClassOf building block`*. The $R_2$ branch generates four possible instance paths. The root can have two values:* `domain rdfs:subClassOf molecular structure`*,* `molecular structure rdfs:subClassOf building block`*. Similarly for the child – there are two statements in the corpus that fit the corresponding condition:* `protein domain rdf:type domain` *and* `internal tandem duplications rdf:type mutations`*. When using the minimum t-norm we can enrich the knowledge base by the following materialised consequents:*

| $s \in A_l$ | $p \in L$ | $o \in A_r$ | $r_{s,p,o}$ |
|---|---|---|---|
| $domain$ | $rdfs:subClassOf$ | $building\ block$ | $1/9$ |
| $protein\ domain$ | $rdf:type$ | $molecular\ structure$ | $2/9$ |

*If we apply Algorithm 1 again, we get one more new statement:*

| $s \in A_l$ | $p \in L$ | $o \in A_r$ | $r_{s,p,o}$ |
|---|---|---|---|
| $protein\ domain$ | $rdf:type$ | $building\ block$ | $2/9$ |

*After that the corpus representation already remains stable (its closure has been computed), as no further application of the rules produces new results.*

## 3. Specific Applications

We have utilised the general framework introduced in the previous section within several practical tasks that show the broad applicability of the presented research. The ma-

[3] See [10]. Note that although the minimum t-norm, $t(a, b) = min(a, b)$, can be applied to any positive values in the corpus representation tensors with the intuitively expected (fuzzy-conjunctive) semantics, any other t-norm, such as the product one, $t(a, b) = ab$, would lead to rather meaningless results if the tensor values were not normalised to the [0, 1] interval first.

jor recent applications are described in more detail in the following sections. Note that despite of the differences in the applications, they are all deployed on life science data. This is due to the fact that the field presents practically motivated challenges related to our research [32]. Life sciences also offer abundance of relevant experimental data in both structured and unstructured form, often including large gold standard data sets for evaluation. However, any of the applications can be straightforwardly deployed in other domains, too, as the only strict requirement in order for the tools to work is availability of any kind of textual and/or RDF input data.

### *3.1. EUREEKA and CORAAL*

EUREEKA is a back-end implementing the emergent knowledge representation and processing framework introduced in Section 2, while CORAAL is a front-end application that utilises EUREEKA to offer intelligent search&browse services on the top of life science publication repositories. In the rest of this section, we provide a brief overview of the EUREEKA and CORAAL applications, and summarise their evaluation. Readers interested in details are referred to [25,24], or to [22] for the most comprehensive information[4].

#### *3.1.1. Method and Data*

We applied the EUREEKA back-end to: (i) automated extraction of machine-readable knowledge bases from the texts of life science articles (using various state of the art Natural Language Processing methods; see [19] in this volume for details on this topic); (ii) integration, refinement and extension of the extracted knowledge within one large emergent knowledge base; (iii) exposure of the processed knowledge via a query-answering and faceted browsing interface, tracking the article provenance of particular statements.

For the initial knowledge extraction, we used a NLP-based heuristics stemming from [15,34] in order to process chunk-parsed texts into subject-predicate-object-score quads. The scores were derived from aggregated absolute and document frequencies of subject/object and predicate terms. The extracted quads encoded three major types of ontological relations between concepts: (I) taxonomical—*type*—relationships; (II) concept difference (i.e., negative *type* relationships); (III) "facet" relations derived from verb frames in the input texts (e.g., *has part*, *involves* or *occurs in*). We imposed taxonomy on the latter, considering the head verb of the respective phrase as a more generic relation (e.g., *involves expression of* was assumed to be a type of *involves*). Also, several artificial relation types were introduced to restrict the semantics of some most frequent relations. Namely, (positive) *type* was considered transitive and anti-symmetric, and *same as* was set transitive and symmetric. Similarly, *part of* was assumed transitive and being inverse of *has part*. Note that the *has part* relation has rather general semantics within the extracted knowledge, i.e., its meaning is not strictly physically mereological, it can refer also to, e.g., conceptual parts or possession of entities.

[4] An instance of CORAAL deployed on cancer research publications from Elsevier is available at `http://coraal.deri.ie`. As of March 2013, we have also developed a related light-weight tool for exploration of text and data collections both in biomedical and general domains. The tool is called SKIMMR and its detailed description is in [23]. SKIMMR is available as a Python package that can be easily installed locally and deployed on arbitrary data – see `https://pypi.python.org/pypi/skimmr_bm/` (biomedical version) or `https://pypi.python.org/pypi/skimmr_gt/` (general version).

The extracted quads were processed as follows (details of the particular steps and the underlying principles are described in [22]): (I) *addition* – The extracted quads were incrementally added into an emergent knowledge base $K$, using a fuzzy aggregation of the respective conceptual matrices. As a seed defining the basic domain semantics (i.e., synonymy and core taxonomy of $K$), we used the EMTREE and NCI thesauri. (II) *closure* – After the addition of new facts into $K$, we computed its materialisation according to RDFS entailment rules [3] ported to the format specified in [22]. (III) *extension* – the extracted concepts were analogically extended using similar stored knowledge.

We exposed the content of the eventual knowledge base via a query-answering module. It returns answer statements sorted according to their relevance scores and similarity to the query (details of the algorithm are again provided in [22]). Answers are provided by an intersection of publication provenance sets corresponding to the respective statements' subject and object terms. The module currently supports queries in the following form: $t \mid s : (NOT\ )?p : o(\ AND\ s : (NOT\ )?p : o)^*$, where $NOT$ and $AND$ stands for negation and conjunction, respectively. $s, o, p$ may be either variable—anything starting with the ? character or even the ? character alone—or a lexical expression. $t$ may be lexical expressions only. The ? and * wildcards mean zero or one and zero or more occurrences of the preceding symbols, respectively, | stands for or. Only one variable name is currently allowed to appear within a single query statement and across a statement conjunction.

For the particular deployment of CORAAL we have processed 11,761 articles from biomedical journals published by Elsevier that were related to cancer research. The access to the articles was provided within the Elsevier Grand Challenge[5]. The domain was selected so due to the expertise of our sample users and testers from Masaryk Oncology Institute in Brno, Czech Republic. From the article repository, we extracted the knowledge and publication meta-data for further processing by CORAAL. Besides the publications themselves, we employed legacy machine-readable vocabularies for the refinement and extension of the extracted knowledge (in particular, the NCI and EMTREE thesauri[6].

CORAAL exposes two data-sets as an output of the publication processing: First, we used a triple store containing publication meta-data (citations, their contexts, structural annotations, titles, authors and affiliations) associated with respective full-text indices. The resulting store contained $7,608,532$ of RDF subject-predicate-object statements [18] describing the input articles. This included $247,392$ publication titles and $374,553$ authors (both from full-texts and references processed).

Apart of the triple store, we employed a custom EUREEKA knowledge base with facts of various certainty extracted and inferred from the article texts and the seed life science thesauri. Directly from the articles, $215,645$ concepts were extracted (and analogically extended later on). Together with the data from the initial thesauri, the domain lexicon contained $622,611$ terms, referring to $347,613$ unique concepts. The size of the emergent knowledge base was $4,715,992$ weighed statements (ca. $99$ and $334$ extracted and inferred statements per publication in average, respectively). The contextual meta-knowledge related to the statements, namely provenance information, amounted to more than $10,000,000$ additional statements (should it be expressed in RDF triples).

[5] See `http://www.elseviergrandchallenge.com`.

[6] See `http://www.cancer.gov/cancertopics/terminologyresources` and `http://www.embase.com/emtree/`, respectively.

### *3.1.2. Evaluation Summary*

We evaluated both EUREEKA and CORAAL tools via the CORAAL interface. A sample of actual users (experts in the domain of cancer research and clinical care) helped us to assess various quantitative and qualitative aspects of the emergent content processed within our framework. The results are summarised in Table 1. The *P, R, F* stand for preci-

| Type | Quantitative | | | Qualitative | | | |
|---|---|---|---|---|---|---|---|
| Attr. | *P* | *R* | *F* | *KR rel.* | *QA rel.* *ans.* | *QA rel.* *doc.* | *Effic.* |
| Val. | 0.532 | 0.305 | 0.310 | 0.289 | 0.348 | 0.425 | 0.657 |
| $\Delta$ | 189% | 374% | 279% | N/A | N/A | N/A | 247% |

**Table 1.** Summary of the evaluation

sion, recall and F-score, respectively, and reflect the quantitative assessment of the automatically generated CORAAL knowledge base content in comparison to a gold standard created by domain experts. As a base-line, we used a standard RDF store for processing the knowledge.

Within the qualitative evaluation, we measured not only factual correctness, but also practical relevance of the knowledge extracted and processed by CORAAL. In particular, we assessed the relevance of statements related to important concepts (*KR rel.*), answers to typical queries (*QA rel. – ans.*) and documents retrieved with the answers (*QA rel. – doc.*). In addition, we measured the efficiency of users when accomplishing typical knowledge-based literature search tasks with our tool in comparison with a base-line (Google, PubMed and ScienceDirect services). Where applicable, the improvement of CORAAL over the base-line is indicated in the $\Delta$ line of Table 1, while the Attr. and Val. lines present the evaluated attribute and its value, respectively. One can clearly see that our framework outperforms the base-lines by rather large margins. Note that what we have provided here is but a brief summary. Full account on the evaluation of the use case deployment is given in [22] (Chapter 9).

## *3.2. Knowledge Consolidation*

In another practical deployment of our research, we chose to showcase the alternative approach to the very definition of meaning of the (Semantic) Web by applying it to knowledge consolidation. The Semantic Web has been designed for asserting meaning of things mostly in a top-down manner (via explicit specifications of RDF descriptions or ontologies). The framework introduced in Section 2 suggests another, bottom-up way of looking at meaning of things on the web, where the semantics consist of implicit patterns that emerge from a simple language of countless triple statements.

Such an alternative way of looking at the Semantic Web can bring better solutions to problems in areas like knowledge consolidation (by which we basically mean clustering of related entities and properties). For instance, in our previously mentioned CORAAL prototype, users can search for properties linking particular life science entities (like genes or diseases). CORAAL extracts all the underlying statements automatically from text, which leads to thousands of properties occurring only in very small number of triples. This may result in too specific query answers and user frustration, as they have to struggle to figure out how to get more general information. Imagine one wants to know

more about organs involved in the production of a hormone H. A query for that can look like *H secreted_in ?x*. However, such a query may retrieve only a single result. More results could be retrieved via related properties like *excreted_in* or *produced_in*, but it is rather tedious to try all such possibilities without knowing precisely how exactly one should ask. A solution grouping extracted content into more general inter-related clusters would significantly improve the user satisfaction and efficiency, as hitting a single property would also reveal all the related ones. Yet for achieving such a consolidation, one needs to know not (only) what is meant by the statements at the level of the particular documents (which is covered by the current approaches). What is more important (and less explored) are the minuscule contextual features distributed across the whole data set (e.g., properties and $\langle$ subject, object $\rangle$ tuples that tend to co-occur at a larger scale with sufficient significance). This is what constitutes the global evidence of what is actually *meant* by the data set at large (and not just *asserted* at the level of local semantic descriptions). By capturing these aspects, one can consolidate the little scattered chunks of related knowledge in an empirically valid manner.

In the following we provide an outline of our approach to knowledge consolidation using the general framework introduced in Section 2, and summarise its evaluation. Readers interested in more detailed description of this experiment are referred to [26].

### *3.2.1. Method and Data*

The first type of data we used in the knowledge consolidation experiment were four RDF documents (parts of the Linked Open Data cloud) that were converted into RDF from manually curated life science databases and served on the D2R web site[7]. To keep the data set focused, we chose resources dealing with drugs and diseases: Dailymed, Diseasome, Drugbank and Sider[8]. This data set is referred to by the LD identifier in the rest.

The second data set we used was generated from the textual content of the LD documents, which contain many properties with string literal objects representing natural language (English) definitions and detailed descriptions of the entries (e.g., `drugbank:pharmacology` describes the biochemical mechanism of drug functions). We extracted the text from all such properties, cleaned it up (removing spurious HTML mark-up and irregularities in the sentence segmentation) and applied a simple NLP relation extraction pipeline on it (a slightly modified version of the pipeline described in Section 3.1, only without negative relationships). This produced a data set of extracted statements (XD in the following text)[9].

Concerning the size of the experimental data, the linked data sets contained ca. $630$ thousand triples, $126$ properties and around $270$ thousands of simple entities (i.e., either subjects or objects) corresponding to almost $150$ thousands of unique identifiers (i.e., preferred labels). The size of the extracted data set was around $3/4$ of the linked data one, however, the number of extracted properties was much higher – almost $35$ thousand. Apart of the LD, XD data sets, we also prepared their LD$^-$, XD$^-$ alternatives, where we

[7] See `http://www4.wiwiss.fu-berlin.de/`.

[8] See `http://dailymed.nlm.nih.gov`, `http://diseasome.eu/`, `http://www.drugbank.ca/` and `http://sideeffects.embl.de/`, respectively.

[9] Details and examples of a typical application of the extraction pipeline are again available as a part of the data package provided at `http://140.203.154.177/resources/2011/iswc2011_data.zip` (an archive containing all the data, source code and additional descriptions relevant to this part of the chapter).

just 'flattened' all the different properties to uniform links. We did so to investigate the influence the multiple property types have on the distributional web semantics features within the experiments.

Before performing the knowledge consolidation, we had to incorporate the RDF data (the LD, XD sets) into the framework introduced in Section 2, i.e., to populate the graph and source representation tensors $\mathbf{G}, \mathbf{C}$ (separate tensors for each of the LD, XD, LD$^-$, XD$^-$ data sets). The $\mathbf{G}$ indices were filled by the lexical elements of triples and by the corresponding source graph identifiers (there were five provenance graphs – one for each of the four linked data documents and one for the big graph of extracted statements). The $\mathbf{G}$ values were set to 1 for all elements $g_{s,p,o,d}$ such that the statement $(s, p, o)$ occurred in the graph $d$; all other values were 0. To get the $\mathbf{C}$ tensor values $c_{s,p,o}$, we multiplied the frequency of the $(s, p, o)$ triples (i.e., $\sum_{d \in P} g_{s,p,o,d}$) by the point-wise mutual information score [17] of the $(s, o)$ tuple. This method is widely used for assigning empirical weights to distributional semantics representations [2], we only slightly adapted it to the case of our "triple corpora" by using the frequencies of triple elements and triples themselves. As we were incorporating triples from documents with equal relevance, we did not use any specific provenance weights in the $\mathbf{C}$ tensor computation. After the population of the corpus tensor, we used its $s/\langle p, o\rangle$, $o/\langle p, s\rangle$ perspectives for generating similar entities and the $p/\langle s, o\rangle$ perspective for similar properties, proceeding exactly as described in Example 4. A cluster of size $x$ related to a vector $\mathbf{u}$ in a perspective $\pi$ was generated as a set of up to $x$ most similar vectors $\mathbf{v}$ such that $sim_\pi(\mathbf{u}, \mathbf{v}) > 0$.

To evaluate the entity consolidation, we employed MeSH[10] as a gold standard. MeSH is manually designed and covers a lot of disease, gene and drug terms, therefore the groups of related things within its taxonomical structure are a good reference comparison for artificially generated clusters of entities from the same domain. To the best of our knowledge, no similar applicable gold standard that would cover our property consolidation data sets exists, thus we had to resort to manual assessment of the corresponding results. As a baseline, we used randomly generated clusters of entities and properties.

### *3.2.2. Evaluation Summary*

The quality of entity clusters was based on their overlap with the MeSH gold standard. The overlap itself was computed as an arithmetic mean of the MeSH-based similarities between all entities from a cluster that were present in MeSH. The MeSH-based similarities were inversely proportional to the distance of the particular entities in the MeSH tree structure (essentially an edge-based approach motivated by the similarity measures commonly used in the context of life science knowledge bases [28]). For evaluating the property clusters, we employed two human evaluators (one bioinformatician and one clinical researcher). They assessed two factors – an adequacy (*aq*) and accuracy (*ac*) of property clusters. Given a property cluster $C$, $ac = \frac{|C|-|N|}{|C|}$, $aq = \frac{|R|}{|C|-|N|}$, where $N$ is a set of properties deemed as noise by a human evaluator, and $R$ is a set of properties considered to be relevant to the seed property the cluster was generated from.

Firstly we will discuss the results of property clustering. The adequacy of clustering was best (0.875) for small, crisp LD clusters (with a rather strict similarity threshold of 0.75), while for bigger clusters without a similarity threshold restriction, it was

[10] A freely available controlled vocabulary and thesaurus for life sciences, see `http://www.nlm.nih.gov/mesh/`.

decreasing, yet still significantly better than the baseline. For the extracted data set (XD), roughly one half of extracted properties was deemed to be accurate. Out of these, around $46.4\%$ in average were adequate members of the analysed property clusters, which is quite promising, as it would allow for reduction of the space of about $35,000$ extracted properties to several hundreds with an error rate around $50\%$. This may not be enough for a truly industry-strength solution, but it could already be useful in prototype applications if extended by result filtering (e.g., ranking) or faceted browsing like in our CORAAL tool. Examples of interesting property clusters we generated are: $C_1 = \{$*secreted_in, excreted_in, appear_in, detected_in, accounted_for_in, produced_in, eliminated_in*$\}$, $C_2 = \{$*from, following_from, gathered_from*$\}$, $C_3 = \{$*increase, increased_by, diminish*$\}$. $C_1$ appears to be related to production/consumption of substances in organs, $C_2$ to origin of substances in location and $C_3$ to quantity change.

To discuss the entity clustering results, let us have a look at Figure 1, which shows the dependency of the cluster quality on the cluster sizes for all the evaluated data sets. The dotted green line (circle markers) represents the baseline (EC-BL), while the red

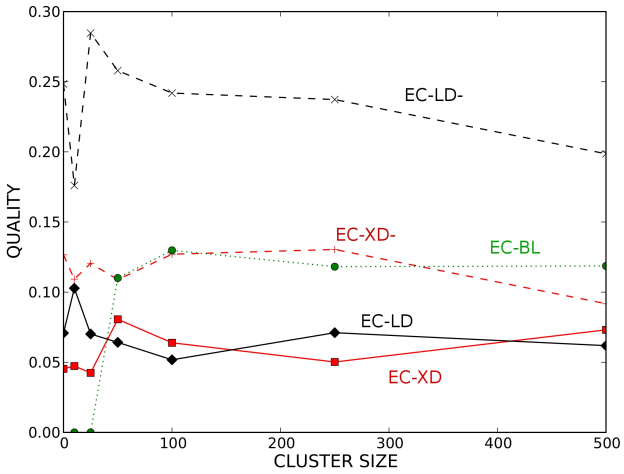



**Figure 1.** Dependency of the cluster quality on their sizes

and black lines (square/plus and diamond/cross markers) are for the extracted and linked open data sets, respectively (EC-XD/EX-XD$^-$, EC-LD/EC-LD$^-$). The solid lines are for original data sets (with properties), whereas the dashed lines indicate resources with properties reduced to mere links. One can immediately see that the results of entity consolidation are significantly better in terms of quality than the baseline for clusters of size up to 25. This holds for all evaluated data sets and thus demonstrates a clear contribution of our approach. This is, we believe, not the most interesting thing, though. Quite surprisingly, the data sets flattened to mere links between entities (the dotted lines) produce much better results than the original resources with multiple property types. This is especially the case of flattened linked data resources (the dashed black line), which perform better than the baseline for all cluster sizes. Another counterintuitive finding is that the flattened automatically extracted resources (red dashes) perform better than the manually

created linked data sets (without flattened properties). For clusters of size $50$ and bigger, the flattened extracted batch oscillates around the random baseline, while both batches with actual properties are consistently worse.

We believe that these surprising results may be caused by the actual difference between the empirical similarities induced by the full and flattened data. The gold standard similarity imposed by the MeSH thesaurus may be directly related to its taxonomy structure. The flattened resources may produce a related, rather simple 'subsumption' type of distributional similarity, while the resources with multiple property types can give rise to a more complex 'structural' similarity. This could be the reason for a better fit of the flattened data to the gold standard. Also, it could explain the poor (i.e., worse than random) performance of the full-fledged data sets for larger cluster sizes. In these cases, the flattened resources may be producing bigger clusters of more general and more specific (but still related) terms, whereas the other type of similarity just increases the noise by adding more and more specific and mutually unrelated structural sub-clusters. As these are all mere assumptions now, though, we are going to further investigate the interesting by-products of our knowledge consolidation experiments in near future.

### *3.3. Rule Mining and Automated Document Annotation*

This particular application of our general framework for emergent knowledge processing is again motivated by the information overload in the context of life science literature. As can be seen for instance in [12], a popular way of tackling this problem is annotation of articles by terms from standardised biomedical vocabularies. Such annotations can in turn make the retrieval of relevant documents much more efficient. However, as providing the necessary annotations manually is very expensive, automated methods are desired [12], which is what the application described in this section addresses. Note that similarly to the previous sections, we provide only an outline of our approach and its evaluation here. Those interested in more details are referred to [21].

#### *3.3.1. Method and Data*

To facilitate the automated document annotation, we utilised another type of semantic features that can be discovered using the framework introduced in Section 2 – IF-THEN rules mined from the emergent knowledge bases. Rules are useful for our motivating use case due to their applicability to extension of the basic article annotations – once we know that an article has annotations that conform to a rule's antecedent, we can also add annotations present in the corresponding rule consequent.

To simplify the presentation, let us consider conjunctive IF-THEN rules of type $(?x, l_1, r_1) \wedge (?x, l_2, r_2) \wedge \cdots \wedge (?x, l_k, r_k) \rightarrow (?x, l_{k+1}, r_{k+1}) \wedge (?x, l_n, r_n)$ in the following, where $?x$ is a variable and $l_i, r_i, i \in \{1, \ldots, n\}$ are concrete relation and (right) argument terms. An example of such rule is $(?x, \text{type of}, domain) \rightarrow (?x, \text{different from}, protein)$, which says that everything that is a type of domain is not a protein. The rule mining consists of two steps: (1) using the matrix perspective $\langle p, o\rangle/s$ for finding candidate sets of $\langle l_i, r_i\rangle$) tuples that can form rules; (2) using the matrix perspective $s/\langle p, o\rangle$ for pruning the generated rules based on their confidence.

The first step corresponds to finding all frequent itemsets in a database as described in the associative rule mining classics [1]. The row vectors of the $\langle p, o\rangle/s$ matrix are essentially the 'items' – features of the rules, i.e., the concrete $(l_i, r_i), i \in \{1, \ldots, n\}$ tu-

ples. By grouping close vectors, we can discover related features that may possibly form rules. Perhaps a simplest way of doing this is $k$-means clustering based on Euclidean distance [11] applied to the $\langle p, o\rangle/s$ matrix. The $k$ parameter is set so that the sizes of the generated clusters correspond to the desired maximum number of statements present in a rule. In practice, we recommend to apply dimensionality reduction to the columns of the matrix. This makes the clustering faster, while also leading to noise reduction and better representation of the features' meaning in the sense of [6]. The described approach effectively replaces the process of finding frequent itemsets in [1]. Using our distributional representation, we find promising 'itemsets' not via support in discrete data transactions, but by exploiting their continuous latent semantics.

The second step involves pruning of the previously generated rules using measures of *support* ($supp$) and *confidence* ($conf$). Only rules with sufficiently high confidence are kept as a result of the mining process. The measures are computed on a matrix that is a transpose of the one used for generating the rules ($s/\langle p, o\rangle$ in case of the discussed type of rules). We keep the original dimensions of the matrix this time, so that we can check for the confidence of the rules using the actual data without any transformations. We base the rule pruning on the definitions of support and confidence provided in [1], however, we generalise the support so that we can fully exploit the power of our distributional representation (see [21] for details).

After giving an overview of the rule mining method we developed on the top of the general framework introduced in Section 2, we can go back to the actual document annotation experiment. As a corpus of documents for annotation, we employed $2,003$ articles from the PubMed repository that had their fulltext content available from PubMed Central[11]. The articles were selected so that for each article present, the corpus also contained corresponding related articles as offered by the PubMed's related articles service [13]. This fact was important for the evaluation later on. For the article annotation, we used the MeSH 2011 version.

We processed the data using the following high-level pipeline: (1) extraction of statements from the articles and from MeSH; (2) incorporation of the extracted statements into two separate knowledge bases for PubMed articles and for MeSH thesaurus; (3) construction of basic MeSH annotation sets for each article; (4) mining of rules from the MeSH knowledge base; (5) rule-based extension of the basic annotation sets; (6) evaluation of the initial and extended sets of annotations.

In the extraction step, we were focusing on simple binary co-occurrence statements. We tokenized the article text into sentences, then applied part of speech tagging and shallow parsing in order to determine noun phrases. Any two noun phrases $NP_1, NP_2$ occurring in the same sentence formed a statement $(NP_1, R, NP_2)$, where $R$ stands (here and in the following) for a $related\_to$ relationship expressing a general relatedness between the left and right arguments. Any synonyms of MeSH terms in the statements were converted to the corresponding preferred MeSH headings in order to lexically unify the data. $1,379,235$ statements were generated from the $2,003$ articles this way. From the MeSH data set, we generated $(T_1, R, T_2)$ statements for all terms (i.e., headings) $T_1, T_2$ such that they were parent, child or sibling of each other in the MeSH hierarchy, which led to $41,632$ statements. Note that for both data sets, we considered the $R$ relation

[11] See `http://www.ncbi.nlm.nih.gov/pubmed/` and `http://www.ncbi.nlm.nih.gov/pmc/`, respectively.

symmetric, which effectively made the $s/\langle p,o\rangle$ and $o/\langle s,p\rangle$ perspectives equivalent in the consequent steps.

The adopted model of co-occurrence limited to a single general relationship $R$ may seem to be restrictive, however, we chose to do so to be able to link the semantics of the data extracted from articles with the semantics of MeSH in the most general sense applicable. Apart of that, the experiment we reported on in Section 3.2 suggests that in this context, such 'flattened' semantics may actually perform better than a model with multiple relations.

The second step in the experimental pipeline was incorporation of the extracted statements into knowledge bases (i.e., the source, corpus and perspective structures described in Section 2). The incorporation was done in the same way for both PubMed and MeSH data, following the process described already in Section 3.2.1.

The annotations for each article $d$ were computed using the article knowledge base as follows. First we constructed a set $TF = \{(t, f_d(t))|t \in d\}$, where $t$ are all terms extracted from $d$ and $f_d(t)$ is the absolute frequency of the term $t$ in $d$. For each $(t, f_d(t))$ tuple from $TF$, we computed a set $REL_t = \{(t', f_d(t) \cdot sim_{s/\langle p,o\rangle}(t,t'))|sim_{s/\langle p,o\rangle}(t,t') > 0\}$. Rephrased in prose, the $REL_t$ sets contained tuples of all terms similar to $t$ and the actual similarities multiplied by the $f_d(t)$ frequency (more frequent terms should generally produce terms with higher relatedness value). The $sim_{s/\langle p,o\rangle}$ similarity function was defined as in Example 4. Eventually, we collated the particular term relatedness values across the whole document $d$ into an overall relatedness $rel(t') = \frac{1}{W}\sum_{w\in W_{t'}} w$, where $W_{t'} = \{r|(t',r) \in \bigcup_{t\in d} REL_t\}$ and $W$ is a sum of all the relatedness values occurring in the $\bigcup_{t\in d} REL_t$ union. The final output of this step for each document $d$ was a set of all related terms $t'$ such that $t'$ is in MeSH. The $rel(t')$ values were used for ranking the set of MeSH annotations and taking only the top ones if necessary.

The rule mining part of the experimental pipeline was executed iteratively with different random initialisations of the clusters until no new rules were added in at least 10 most recent iterations. We obtained $33,384$ rules with confidence at least $0.5$ this way. The rules were then used for extending the basic article annotation sets as follows. Let us assume an article $d$ has annotations $\{t_1, t_2, \ldots, t_n\}$. Then for any rule $(?x, R, e_1) \wedge (?x, R, e_2) \wedge \cdots \wedge (?x, R, e_k) \rightarrow (?x, R, e_{k+1}) \wedge (?x, R, e_{k+2}) \wedge \cdots \wedge (?x, R, e_m)$ such that $\{t_1, t_2, \ldots, t_n\} \subseteq \{e_1, e_2, \ldots, e_k\}$, we used the consequent set $\{e_{k+1}, e_{k+2}, \ldots, e_m\}$ as extended annotations for the article $d$. The relatedness measure of the extensions $e$ was computed as $\frac{1}{W}\sum_{w\in C_e} w$, where $C_e$ is a set of confidences of all rules that contributed with the extension $e$, and $W$ is a sum of all such confidences across all extensions computed. Similarly to the basic annotation sets, the relatedness of the extensions was used for their ranking and possible restriction to top-scoring ones.

### *3.3.2. Evaluation Summary*

To evaluate the annotation sets produced in the experimental pipeline, we used two methods. Firstly, we measured precision and recall of the basic and extended annotation sets based on their comparison with manually provided MeSH annotations of the corresponding articles (available through the PubMed's Entrez API). For each article, we computed average precision, precision and recall [17] of all computed annotations and also of top $h$ ones, where $h$ is the number of human annotations for the given article.

The second evaluation method focused on the utility of the computed annotations, namely in the task of finding related articles. We used a standard vector space model [29] for determining the relatedness of documents, where features were formed by the sets of computed or manually assigned article annotations. For each document, we computed different sets of related documents (based on the human annotations and on the basic/extended ones generated by our framework). To determine their precision and recall, the computed sets were compared to corresponding sets of related articles provided by the dedicated PubMed service. Similarly to the evaluation of annotations themselves, we measured average precision, precision and recall of all and of top $h$ related articles computed, where $h$ was the number of related articles in the gold standard.

The comparison with the manually curated MeSH annotations was not particularly impressive, with highest precision and recall values of $16.4\%$ and $12.7\%$, respectively. On the other hand, the automatically computed annotations performed much better than the 'manual' ones when using them as features for finding related articles. There was a substantial improvement namely regarding precision and overall F-score. The only measure where the manually curated annotations performed slightly (ca. $1.1$-times) better than the next-best automated method was recall. Especially notable was the difference in precision – the extended annotations achieved more than $91\%$ when finding related articles, which was about two-times better than when using the manual annotations.

The results we obtained may have several interpretations. We believe that one of the more plausible ones is related to the nature of the manually provided MeSH annotations. As mentioned for instance in [20], the goal of PubMed annotators is to provide best MeSH 'tags' for the purpose of indexing in digital library collections. Thus they are motivated to select annotations that better discriminate papers from each other. This may, however, be rather detrimental when the task is to identify related papers using the annotations, as features used for identifying relatedness (i.e., similarity) are often incompatible with the features used for discrimination of entities [33]. This reasoning can in turn explain why our automatically computed article annotations, apparently very different from the manually curated ones, perform significantly better when used as features for finding related articles. The better performance (especially in case of the precision of extended annotations) may indicate that the automatically computed annotations are selected in a more fine-grained manner and from a more varied 'vocabulary' than the ones provided by human annotators, who can hardly grasp the scale of all the hypothetically available annotations (in addition to having different motivations as mentioned before). This is not to say that either kind of annotations is worse than the other, it much rather means that they simply serve slightly different purposes.

To conclude the discussion, we believe that despite of the low performance of our approach in terms of comparison with manually curated MeSH annotations, we can still offer potentially very beneficial results (especially in case of annotations augmented by emergent rules). This holds particularly for use cases where the annotations are supposed to be produced in a scalable and economical way in order to determine similarities between articles. Examples of such use cases include not only identification of related documents, but also question answering or automated linking of publications and supplementary data (e.g., pharmaceutical, disease or protein datasets available as Linked Open Data).

## 4. Conclusions and Future Work

We have introduced a layered framework for discovery of emergent semantic phenomena empirically grounded in simple statements extracted from the web data and/or texts. Further augmentation of the content processed within our framework is made possible by a symbolic layer that involves deductive (rule-based) reasoning on the top of the essentially inductively inferred emergent knowledge. This allows for seamless combination of the more traditional, top-down approaches to web semantics with our bottom-up framework.

Apart of the general framework, we gave an overview of three distinct applications that were motivated by various challenges related to information overload in life sciences. The presented results clearly show a practical applicability of our framework in diverse scenarios and demonstrate a significant potential for future development.

In future, we want to investigate inference of more complex and varied semantic phenomena (e.g., analogies or named hierarchical clusters). Consequently, we intend to develop practical applications involving continuous materialisation of emergent knowledge bases according to relevant rule sets. Last but not least, we want to work more intensively with potential users of the introduced technologies in order to deliver truly industry-strength solutions that would successfully cope with the present challenges in information overload.

*Acknowledgements* This work has been supported by the 'Líon II' project funded by SFI under Grant No. SFI/08/CE/I1380. We also thank the book editors and anonymous chapter reviewers for remarks that provided us with much guidance when preparing the final text.

## References

[1] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. *SIGMOD Rec.*, 22(2):207–216, 1993.

[2] Marco Baroni and Alessandro Lenci. Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics*, 36(4):673–721, 2010.

[3] Dan Brickley and R. V. Guha. *RDF Vocabulary Description Language 1.0: RDF Schema*, 2004. Available at (Feb 2006): `http://www.w3.org/TR/rdf-schema/`.

[4] Paul Buitelaar and Philipp Cimiano. *Ontology Learning and Population: Bridging the Gap between Text and Knowledge*. IOS Press, 2008.

[5] Philipp Cimiano, Alexander Pivk, Lars Schmidt-Thieme, and Steffen Staab. Learning taxonomic relations from heterogenous sources of evidence. In Paul Buitelaar, Philipp Cimiano, and Bernardo Magnini, editors, *Ontology Learning from Text: Methods, Evaluation and Applications*, pages 59–73. IOS Press, 2005.

[6] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[7] Robert B. Doorenbos. *Production Matching for Large Learning Systems*. PhD thesis, 1995.

[8] Peter Haase and Johanna Völker. Ontology learning and reasoning - dealing with uncertainty and inconsistency. In *Proceedings of URSW2005*, pages 45–55, NOV 2005.

[9] Hele-Mai Haav. An ontology learning and reasoning framework. In Yasushi Kiyoki, Jaak Henno, Hannu Jaakkola, and Hannu Kangassalo, editors, *Information Modelling and Knowledge Bases XVII*, volume 136 of *Frontiers in Artificial Intelligence and Applications*, pages 302–309. IOS Press, 2006.

[10] Petr Hájek. *Metamathematics of Fuzzy Logic*. Dordrecht: Kluwer, 1998.

[11] J. A. Hartigan and M. A. Wong. Algorithm AS 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[12] Minlie Huang, Aurélie Névéol, and Zhyiong Lu. Recommending MeSH terms for annotating biomedical articles. *Journal of the American Medical Informatics Association*, 18(5):660–667, 2011.

[13] Jimmy Lin and W. John Wilbur. PubMed related articles: a probabilistic topic-based model for content similarity. *BMC Bioinformatics*, 8(1), 2007.

[14] Alexander Maedche. Emergent semantics for ontologies. In *Emergent Semantics*, IEEE Intelligent Systems, pages 85–86. IEEE Press, 2002.

[15] Alexander Maedche and Steffen Staab. Discovering conceptual relations from text. In *Proceedings of ECAI 2000*. IOS Press, 2000.

[16] Alexander Maedche and Steffen Staab. Ontology learning. In S. Staab and R. Studer, editors, *Handbook on Ontologies*, chapter 9, pages 173–190. Springer, 2004.

[17] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schuetze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[18] Frank Manola and Eric Miller. *RDF Primer*, 2004. Available at (November 2008): `http://www.w3.org/TR/rdf-primer/`.

[19] Diana Maynard and Kalina Bontcheva. Natural language processing. In Johanna Völker and Jens Lehmann, editors, *Perspectives of Ontology Learning*, Studies on the Semantic Web. AKA Heidelberg / IOS Press, 2013.

[20] Aurélie Névéol, Sonya E. Shooshan, Susanne M. Humphrey, James G. Mork, and Alan R. Aronson. A recent advance in the automatic indexing of the biomedical literature. *Journal of Biomedical Informatics*, 42(5), 2009.

[21] Vít Novácek. Distributional framework for emergent knowledge acquisition and its application to automated document annotation. *CoRR*, abs/1210.3241, 2012.

[22] Vít Nováček. *EUREEKA! Towards a Practical Emergent Knowledge Processing*. PhD thesis, Digital Enterprise Research Institute (DERI), National University of Ireland Galway, 2011. Available at (January 2011): `http://dl.dropbox.com/u/21379226/thesis/phd_inf.pdf`.

[23] Vít Nováček and Gully Burns. SKIMMR: Machine-aided skim-reading. In *IUI13 Companion*. ACM, 2013.

[24] Vít Nováček and Stefan Decker. Towards lightweight and robust large scale emergent knowledge processing. In *Proceedings of ISWC'09*. Springer, 2009.

[25] Vít Nováček, Tudor Groza, Siegfried Handschuh, and Stefan Decker. CORAAL - dive into publications, bathe in the knowledge. *Web Semantics: Science, Services and Agents on the World Wide Web*, 8(2), 2010.

[26] Vít Nováček, Siegfried Handschuh, and Stefan Decker. Getting the meaning right: A complementary distributional layer for the web semantics. In *Proceedings of ISWC'11*. Springer, 2011.

[27] Kevin Ottens, Nathalie Aussenac-Gilles, Marie-Pierre Gleizes, and Valerie Camps. Dynamic ontology co-evolution from texts: Principles and case study. In *Proceedings of ESOE 2007 Workshop*, pages 70–83. CEUR-WS, 2007.

[28] Catia Pesquita, Daniel Faria, AndrÃl' O. FalcÃčo, Phillip Lord, and Francisco M. Couto. Semantic similarity in biomedical ontologies. *PLoS Computational Biololgy*, 5(7), 2009.

[29] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, 1975.

[30] Gerd Stumme and Alexander Maedche. Fca-merge: Bottom-up merging of ontologies. In *IJCAI'01*, 2001.

[31] Herman J. ter Horst. Completeness, decidability and complexity of entailment for RDF schema and a semantic extension involving the OWL vocabulary. *Journal of Web Semantics*, pages 79–115, 2005.

[32] Junichi Tsujii. Refine and pathtext, which combines text mining with pathways. Keynote at Semantic Enrichment of the Scientific Literature 2009 (SESL 2009), March 2009.

[33] Amos Tversky. Features of similarity. *Psychological Review*, 84(2):327–352, 1977.

[34] Johanna Voelker, Denny Vrandecic, York Sure, and Andreas Hotho. Learning disjointness. In *Proceedings of ESWC'07*. Springer, 2007.

[35] Denny Vrandecic and Anja Jentzsch. Linked data and the semantic web. In Johanna Völker and Jens Lehmann, editors, *Perspectives of Ontology Learning*, Studies on the Semantic Web. AKA Heidelberg / IOS Press, 2013.

# Part IV

# Learning from Web Data

# Capturing Emergent Semantics from Social Tagging Systems

Dominik BENZ [a] and Andreas HOTHO [b]
[a] *Knowledge and Data Engineering Group, University of Kassel, Germany*
*benz@cs.uni-kassel.de*
[b] *Data Mining and Information Retrieval Group, University of Würzburg, Germany*
*hotho@informatik.uni-wuerzburg.de*

**Abstract.** With the advent of the so-called "Web 2.0", the participatory nature of many of its applications has nurtured the vision of many researchers to finally overcome the knowledge acquisition bottleneck inherent to many Semantic Web applications. Despite the uncontrolledness and openness of platforms like collaborative tagging systems, blogs, social networks or wikis, evidence for the presence of emergent semantics within the resulting large bodies of human-annotated content was found. The strengths and weaknesses of traditional top-down approaches (like expert-defined ontologies) and this "wisdom of the crowds" approaches are obviously inverse. Therefore the idea to bridge the gap between the social and the semantic web has motivated a large number of approaches coming from different research areas.

In this chapter, we will give a systematic overview of the recent trends and developments within the research branch targeted towards capturing emergent semantics from social tagging systems. Starting with an introduction to the data characteristics, we give an overview of approaches which analyze evidences and factors of emergent semantics. The following main part of this chapter is concerned with various methods geared towards making the implicit semantics within social tagging data explicit. The presentation is structured along a set of comparison dimensions, including the different levels of the ontology learning layer cake. We conclude the chapter by giving an outlook on promising future research directions.

**Keywords.** Folksonomies, Social Tagging, Emergent Semantics

## 1. Ontology Learning to bridge the gap between Web 2.0 and the Semantic Web

In contrast to the "Web 1.0", where information consumers and producers are clearly distinct, the participatory nature of many so-called "Web 2.0" applications allows anyone to easily annotate digital information resources (e.g. websites or multimedia content) someone else authored. Despite the fact that no central authority or expert team exercises control, some of these systems successfully tackle problems that Semantic Web applications were initially designed for, especially in the fields of knowledge sharing, reuse and organization. This phenomenon has often been attributed to *emergent semantics*, describing the formation of an implicit consensus of shared understandings. It constitutes a bottom-up alternative[1] to the top-down Semantic Web paradigm inherent e.g. in expert-created

[1] sometimes called "grassroots semantics"

ontologies.

Although many approaches which (semi-)automatically support the creation and maintenance of ontologies have reached a decent degree of maturity, the fact that large user populations are effectively involved in the creation of *metadata* opened up new possibilities and challenges to the field of ontology learning. The vision of *bridging the gap* [3,36] between Web 2.0 and the Semantic Web has motivated a large number of approaches. An early and inspiring insight was that the characteristics of the "traditional" way of capturing semantics e.g. within ontologies and the user-created metadata from social tagging systems are inverse: While ontologies provide a precise and expressive knowledge representation formalism, their creation and maintenance suffers from the knowledge acquisition bottleneck. Metadata from the Social Web, on the other hand, is available in abundance and adapts fast to changes, but suffers from problems like ambiguity or lack of precision. Clearly, combining "the best of both worlds" seems to be a promising direction to follow towards a next generation World Wide Web.

In this chapter, we give a systematic overview of the research area of capturing emergent semantics from social tagging systems. Our starting point are works which provide evidence for the presence of emergent semantics within user-contributed tagging data; as a next step, we analyze the question which factors influence the quality of the learned semantic structures. The following main part of this chapter summarizes approaches of making the implicit semantics explicit, structured along a set of comparison dimensions including the different layers of the ontology learning layer cake. We conclude with discussing promising future research directions.

## 2. Web 2.0 data

In general, it is difficult to draw clear boundaries between "traditional" Web 1.0 data and Web 2.0 data. But despite the fact that the Web 1.0 was also intended to be a "web of people", a commonality of most Web 2.0 applications is that they greatly alleviate the process of contributing and sharing content on the Web for end-users. Their inherent ease of use and their immediate usefulness has in fact engaged millions of human users in editing and annotating web resources. For the context of this work, we will mainly consider *collaborative tagging* and leave out all the other Web 2.0 phenomena.

Tagging systems like Delicious, Flickr or BibSonomy[2] allow the sharing of different kinds of content (e.g. bookmarks, pictures or scientific papers). Users can assign freely chosen keywords or *tags* to the uploaded items to ease organization and retrieval. Despite their uncontrolled nature, [25] have provided evidence for *emergent semantics* within the resulting so called *folksonomies*. More formally, we stick to [29] and refer to a *folksonomy* as a tuple $\mathbb{F} := (U, T, R, Y)$ where $U$, $T$, and $R$ are finite sets, whose elements are called *users*, *tags* and *resources*, respectively, and $Y$ is a ternary relation between them, i.e. $Y \subseteq U \times T \times R$.

In the context of ontology learning, the question is now to which extent the data resulting from tagging systems differs from "traditional" input data of ontology learning algorithms. A fundamental difference lies hereby within the way *how* and *by whom* the data is created. This difference can be best explained along the comparison dimensions (i)

[2] `http://www.delicious.com`, `http://www.flickr.com`, `http://www.bibsonomy.org`

| | | Social tagging data | | "traditional" OL input |
|---|---|---|---|---|
| *Motivation* | user goal | solution of specific (personal) task, primarily for own benefit | ↔ | capturing and conveying information for a specific audience |
| | community goal | aggregate user prefrences | ↔ | more content in higher quality |
| | medium publicity | side-effect | ↔ | core aspect |
| | user types | producers, consumers | ↔ | contributors, lurkers |
| *Communic.* | richness | low | ↔ | high |
| | type | many-to-many, many-to-one | ↔ | one-to-many |
| | cost | low | ↔ | high |
| | influence on users | by actions, implicit | ↔ | by messages, explicit |
| *Req.-ments* | amount of domain knowledge | small | ↔ | large |
| | cognitive cost | low (lightweight conceptualization) | ↔ | high (structuring of content) |
| | nr. of contributors | high | ↔ | low |

**Table 1.** Dimensions of comparision between "traditional" input of ontology learning algorithms and input from social tagging systems: Motivation, communiction and requirements.

*motivation* of contributors, (ii) *communication* among contributors and (iii) *requirements* for contributions. The following sections explain each of these in detail.

**Motivation of contributors** For many ontology learning datasets (e.g., medical journals in [18]), it is justified to presume that an important motivation of the authors is to *capture and convey information such that it can be used by others*. This is not necessarily the case for social tagging data: [25] presented evidence that "users bookmark primarily for their own benefit, not for the collective good". The aspect that an intrinsically private information management task (like the maintenance of a personal bookmark collection) is performed in a public space is a novel characteristic of the resulting data. To use a more pointed formulation, the main motivation for users to "produce" data in a Web 2.0 system is not necessarily to make the data available to the public (as it is when writing articles or books), but rather to solve a specific personal task.

**Communication among contributors** Compared to, e.g., a group of people writing a book, the connection among all users involved e.g. in the creation of a tag cloud describing a certain resource is much less explicit. [56] refers to these communication features as *ballot box communication*. Compared to "traditional" computer-mediated communication, one of its distinguishing characteristics is the many-to-one nature of information flows. It describes the paradigm that the many individual contributions are aggregated and presented to each user as a single "voice of the crowd". This exposure leads to the fact that an individual is implicitly influenced by local or global trends within a certain community - without necessarily being able to pin down this influence to one or more particular users.

**Requirements for contribution** Compared to the high requirements of, e.g., writing a book, we can find probably the most distinguishing feature among the characteristics of social tagging systems: A central aspect of them is their very low entry barrier and immediate usability for a large number of users. [56] noted that this effect is partially a result of the limited interaction options which lower the participation costs compared to e.g. reading and writing messages. Sinha[3] reported on lower cognitive costs of tagging. In addition, most systems impose no special requirements of domain knowledge,

[3] http://rashmisinha.com/2005/09/27/a-cognitive-analysis-of-tagging/

which leads to the situation that the knowledge is not concentrated within a small circle of experts, but more fragmented and distributed among a broad community. Table 1 summarizes the main dimensions of comparison discussed before.

## 3. Evidence and Factors of Emergent Semantics in Social Tagging Data

According to [61], emergent semantics *"refers to a set of principles and techniques analyzing the evolution of decentralized semantic structures in large scale distributed information systems"*[4]. In this article, collaborative tagging is also mentioned as a key application to be analyzed using an emergent semantic paradigm.

While the phenomenon of collaborative tagging was discussed in its early stages mainly in newsgroups or mailing lists (e.g. [41,50]), a first systematic analysis was performed by [25]. One core finding was that the openness and uncontrolledness of these systems did not give rise to a "tag chaos", but led on the contrary to the development of stable patterns in tag proportions assigned to a given resource. [15] reported similar results and denoted the emerging patterns as *"semantic fingerprints"* of resources. [17] analyzed statistical properties of tag co-occurrence networks; by using a shuffling approach, they discovered local patterns of co-occurrence indicating a possible underlying semantic hierarchical organization. A consensus of these works is that the large bodies of human-annotated content resulting from collaborative tagging systems contain evidences for emergent semantics.

Following the reported evidences for the existence of latent conceptual structures, a natural next question is which *factors* are having an impact on their emergence and quality. This question has been addressed by a much lower number of researchers, because the necessary data (e.g. which recommendation or spam prevention techniques are used) for such analyses is often only available to the system operators themselves. Despite this fact, we suggest to categorize the influencing factors into *system properties*, *tagging pragmatics* and *system abuse*.

**System properties:** A first intuitive assumption is that the emergent semantics from a tagging system depends on intrinsic properties of the system itself. Among those, the sheer *size* of the system already seems to play a crucial role; an early empirical observation by [25] was that "*after the first 100 or so*" tagging activities on a given resource, its semantic fingerprints tended to stabilize. A more systematic analysis was done by [34], who used a measure of semantic distance as an indicator of emergent semantics based on various folksonomy partitions. The solid black line in Figure 1 shows an almost continuous decrease in semantic distance (indicating higher quality of emergent semantics) when subsequently adding users in random order. This suggests the rule "more data, better semantics". However, further studies considering pragmatic aspects of tagging relativized this claim (see below). However, to the best of our knowledge it is still an open an interesting question at which point in time a folksonomy (or certain parts of it) are "mature" enough for harvesting its implicit semantics.

Another aspect is the *domain* of the system under consideration. Using again folksonomy-derived measures of tag relatedness, [5] computed similar tags based on Delicious (known for its technophile user basis) and Flickr (used by a broader population to share pictures). A finding was a different kind of semantics: While, e.g., "bug" and "net"

[4] retrieved from `http://people.csail.mit.edu/pcm/papers/EmergentSemantics.pdf`

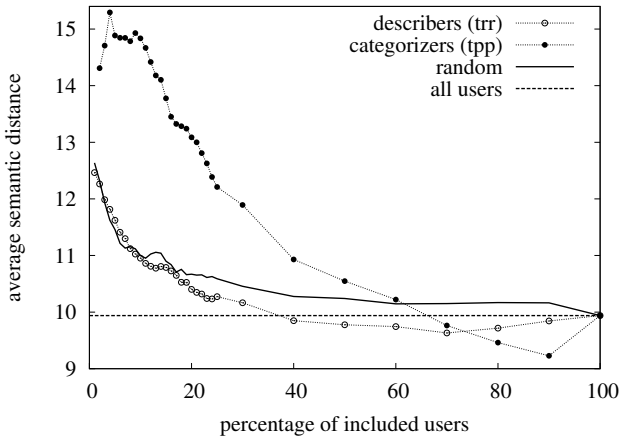



**Figure 1.** Influencing factors on Emergent Semantics.

were semantically related to browser and Internet tags within Delicious, their use within Flickr covered different (nature-related) aspects.

**Tagging pragmatics:** Besides the question of tag semantics (i.e. *what* tags mean), researchers started to observe different usage patterns within the user populations and gained interest in tagging pragmatics (i.e. *how* tags are used). [34] proposed a broad distinction of taggers into *categorizers*, who typically use a small and controlled set of tags, and *describers*, using a more verbose style of tagging. In their work, they systematically analyzed sub-folksonomies containing different ratios of categorizers and describers with regard to the quality of the contained semantics. The latter was measured again using a notion of semantic distance as introduced by [16]. Figure 1 shows some exemplary results. When subsequently adding describers (dotted line with blank circle, from left ro right), an interesting observation is that roughly 40% of all users suffice to reach the semantic precision of the whole dataset (dashed line). When adding categorizers (dotted line with filled circle), most sub-folksonomies contain a smaller amount of implicit semantics compared to random addition (solid line). The global optimum is found for 90% of categorizers; the authors hypothesized that the missing 10% of extreme describers were probably spammers, whose removal has beneficial effects. These results provide empirical evidence for a causal link between tagging pragmatics and tag semantics.

**System abuse and spam:** A common problem of tagging systems and many other popular application on the web is that growing popularity often leads to an increase of unintended and spam-related activities. As an example, [33] reported that among the 20,000 users of BibSonomy, 18,500 were identified manually as spammers, responsible for 90% of all posted bookmarks. These issues obviously have to be taken into account when analyzing emergent semantics in such systems. [17] identified anomalies in tag co-occurrence networks which could be traced back to spamming activity. [34] made a similar observation of a detrimental effect of spammers on global emergent semantics. In both cases, very simple "spam detection" techniques (namely filtering out posts containing an excessive number of tags) led to a disappearance of the anomalies and to an improvement of the semantic precision (see also Figure 1). The discussion of more sophisticated spammer identification algorithms is beyond the scope of this chapter; the

| *Dimension* | *Values* |
|---|---|
| Data Sources | folksonomy structure, resource content, tag content, user-defined tag relations, additional metadata, external sources |
| Data Filtering | by tag / resource / user properties, by external source, manual |
| Learning Technique | statistical data mining & machine learning (clustering, association rules, generative models, latent semantic analysis), SNA measures, NLP techniques, custom algorithms |
| Learning Task | ontology construction (terms, synonyms, concepts, concept hierarchy, relations & axioms), semantic measures (relatedness, generality), tag sense disambiguation, ontology maintenance, ontology population |
| Evaluation | human assessment, gold-standard based, application-centered (folksonomy), application-centered (external application) |

**Table 2.** Comparison dimensions and possible values to compare methods of making semantics in folksonomies explicit.

interested reader is referred to [27]. However, we would like to emphasize that proper spam handling is an essential step in any methodology of exploiting tag semantics.

## 4. Methods to make emergent semantics from social tagging data explicit

The early insights of emergent semantics in social tagging systems and the availability of large test data sets quickly motivated a large number of approaches coming from different disciplines targeted towards making the implicit semantic structures explicit. A complete coverage is beyond the scope of this chapter; so our approach is to first propose a set of comparison dimensions and values in Section 4.1 which are intended to cover as exhaustively as possible all approaches. The following sections are mainly structured along the core dimension of "Learning Tasks" (see Table 2). The selection of the presented approaches was guided by the goals of (i) preferring early works on a specific direction (ii) covering a possibly broad range of relevant authors and (iii) providing of at least one example for each of the values defined in our comparison model.

### *4.1. Comparison dimensions*

The main dimensions and values used for comparison are summarized in Table 2. [23] also provided an extensive review of approaches to discover tag semantics, including a suggestion for a unified process model. Complementing their work, our main focus is not to compare existing approaches based on the different process steps, but rather using dimensions similar to prior comparisons of approaches of general ontology learning like [44,49,10]. As we consider the "Learning Tasks" to be the core dimension of comparison, we will cover each of its values in a subsequent subsection; the other dimensions will now be briefly explained.

First of all, a core question is of course which **Data Source** is exactly used to derive semantics from. A large number of approaches (e.g. [26]) is solely based on the *folksonomy structure*. By this we mean the tripartite structure itself as defined in Section 2, as well as derived structures like tag-resource [43] or tag co-occurrence [8] networks. The advantage of this kind of approaches is their independence from tag language and content type of the shared resources (bookmarks, videos, pictures, ...). When taking into account the *tag content* (i.e. the lexical representation of a tag itself [52]) or the *resource*

*content* [12], the advantage of more information is to be traded off against a restricted applicability to different languages and content types. If available, *additional metadata* like time and location information can also be exploited [32]. Some systems also allow their users to define *tag relations*, which can also be used for taxonomy induction [46]. Finally, approaches which consider *external sources* like existing ontologies or thesauri [39] benefit from rich prior knowledge, but of course naturally depend on the availability of such.

Having chosen a specific data source, most approaches perform an a-priori **Data Filtering** in order to exclude inappropriate content or optimize the input for a specific learning procedure. Typically, data items are hereby disregarded based on certain *tag, user* or *resource properties*. Very common are minimum frequency thresholds [55], or top-k selections [16] which restrict the analysis to popular folksonomy partitions. A method to restrict the networks to their denser parts is to use $p$*-cores* [7], which iteratively removes nodes with less than $p$ connections. But also more sophisticated strategies like including only resources annotated with a least one verb [37] or users with certain tag usage patterns [34] can be found. The latter is also often related to spam removal. Another possible data restriction is to only keep items which are present in external repositories like tags in Tagpedia[5] [53]. In addition, the fine-tuning of the resulting dataset is also often done manually, e.g. by removing system tags like "system:unfiled" [6].

The variety of disciplines interested in learning tag semantics also led to a variety of applied **Learning Techniques**. Clustering is an obvious candidate of *statistical data mining and machine learning* techniques to form groups of (semantically) related tags [4]. Closely related are association rule mining methods [28] which are able to detect semantic relations among items. A more specialized example from this area are statistical models of subsumption [48], targeting the discovery of is-a relations among tags. From a different perspective, generative approaches like the separable mixture model (SMM, [59]) are modeling the users' behaviour in assigning tags to resources. Starting from an observed tag co-occurrence distribution, a conditional distribution of tags over a fixed number of topics is computed. Another theoretically well-founded approach is to apply dimensionality reduction techniques like latent semantic analysis (LSA, [21]) to the high-dimensional tag vector space, resulting in a mapping of tags to "topics" or "concepts". Because some folksonomy-induced networks exhibit suitable properties, also measures stemming from *social network analysis* (SNA) like centrality or clustering coefficient were applied [43], mostly in order to distinguish between general and specific tags. Despite the fact that the assignment of tags to resources does not follow any kind of syntactical pattern, researchers from the natural language processing community (NLP) have used part-of-speech taggers to gain a deeper syntactic understanding of a tag and finally to discover equality and synonymy relations among tags [52]. A last family of approaches are custom algorithms specifically tailored for the task of capturing tag semantics, like the incremental tag taxonomy induction algorithm proposed by [26].

Besides the specific learning tasks (which will be covered in the subsequent subsections), the last comparison dimension is which **Evaluation** paradigm was used to assess the quality of the learned semantics. Because this dimension differs at least from general ontology learning, we just recapitulate briefly the three main classes mentioned by [19] of (i) human assessment (e.g. [48]), (ii) gold-standard based (e.g. [8]) and (iii) application-

[5] `http://www.tagpedia.org`

centered approaches. The latter is often performed within the folksonomy system itself, e.g. by using the learned semantics to improve tag recommendations [54] or information retrieval [39]. Examples of integration into external systems are less frequent, but found for example in the context of e-learning applications [20].

### *4.2. Semantic measures*

When a sufficient amount of knowledge is captured within an ontology like e.g. WordNet, then one can also derive useful semantic measures from it. Of special interest for many knowledge-based applications are hereby measures of **Semantic Relatedness** among terms, which assign a score to a given term pair $(t_1, t_2)$ reflecting the likeness of their semantic meaning (cf. [13]). For this reason and because the notion of relatedness also plays an important role for many ontology induction algorithms, an early research question was to which extent such measures could be derived from a *folksonomy*. While a number of approaches successfully applied several kinds of such measures (e.g. an adapted version of Jaccard similarity coefficient in [42], a systematic analysis of the characteristics of different relatedness measures was done by [16]. Among their findings was that the so-called *tag context relatedness* (representing tags in the co-occurrence vector space with other tags, and computing cosine similarity therein) yields semantically more closely related tags, compared to e.g. plain co-occurrence. [40] provided a generalization including further aggregation and projection variants, pointing out that measures based on mutual information also score high in terms of semantic precision.

Apart from semantic relatedness, some knowledge extraction algorithms are also based on a notion of **Semantic Generality** of tags, which is used e.g. to distinguish between broader and narrower terms. Again, several measures were used in the literature (e.g. network centrality by [26]). A systematic comparison was done by [9], who compared each measure with a gold-standard measure derived from several taxonomies and found that relatively simple measures (like e.g. degree centrality or even frequency) already encode an acceptable degree of generality information.

### *4.3. Ontology construction*

Despite the fact that e.g. measures of semantic tag relatedness do have a value on their own, the goal of formalizing the implicit knowledge in folksonomies by constructing ontologies remains desirable. For the coverage of such approaches, we will stick to the tasks contained in the "ontology learning layer cake" [18] and detail on relevant approaches for each layer.

**Terms:** A main advantage of folksonomies compared to text documents as input for ontology learning algorithms is that the process of term extraction is much simpler – in fact, many approaches skip this step completely and regard tags directly as terms. However, observing a great variety of spelling and abbreviation variants among tags, some works perform *tag normalization* at different levels of complexity ([14,51,53]).

**Synonyms & Concepts:** Many researchers came up with various methods to form groups of tags with a similar meaning, often referred to as *concepts*. In a strict sense, the latter does not conform to the definition given by [18], according to which the process of concept formation should also provide an intensional definition (e.g. a natural language description or a set of typical attributes) of each concept. Despite that, we will

treat for simplicity reasons both variants (i.e. semantic groupings of tags with and without[6] an additional intensional definition) uniformly as "concepts". [4] proved the applicability of clustering to discover concepts by using an algorithm based on spectral bisection. Their approach requires an predefinition of the number of clusters. Examples of approaches which produce a variable number of clusters are [60,47]. Because these clustering techniques are usually based on a notion of tag similarity, another relevant result is given by [16], who identified distributional measures of tag similarity which yield preferably synonym tags (as found in WordNet). One of them was applied by [8] for the process of "synsetizing" a folksonomy by merging tag pairs whose similarity was above a similarity threshold. [30] explores the possibility to match tags with the same meaning across different languages. [31] applied FCA techniques to discover so-called frequent tri-concepts (i.e. sets of users, tags and resources belonging to an implicit concept) within folksonomies. [54] apply a translation approach to compute mappings between tags used by different users to describe similar concepts.

Another class of approaches is to take into account external sources containing prior knowledge about semantic relations. [2] proposed a method to enrich folksonomies by mapping tags to concepts defined in WordNet; [53] described a similar approach using categories and concepts derived from Wikipedia. Though coming from a different direction, approaches like [1] are exploiting WordNet for query expansion, geared towards enhancing retrieval quality by querying with "concepts on the fly".

**Concept Hierarchy:** While synonym resolution is mainly targeted towards improving retrieval tasks, the reconstruction of hierarchical relationships among tags (or learned concepts) is often mentioned in the context of enhanced browsing facilities. Because the maintenance of larger hierarchies is a difficult task and hence the idea of self-organizing structures is very appealing, many researchers have focussed on this aspect. [43] pioneered in deriving broader / narrower tag relations from a user-tag graph (called "actor-concept network"), which are effectively based on subcommunity relationships. [26] suggested a custom algorithm to induce a "tree of tags", based on a measure of tag generality and a measure of tag similarity. This algoritm was later extended by [8] by (i) using computationally more lightweight measures and (ii) handling of synonymy and ambiguity. [48] applied a statistical model of subsumption (originally stemming from work on deriving concept hierarchies from text) for a similar purpose. [21] used probabilistic latent semantic indexing (PLSI) to induce a taxonomy of tags. [60] used a divisive clustering technique based on deterministic annealing to iteratively split the set of tags into semantically coherent subsets. The approach of [42] consists in building first a directed weighted tag graph using a notion of generality, and then removing edges until a maximum spanning tree is found. Based on a different data source, [46] suggested to integrate user-specified tag relations[7] into a global concensus structure.

**Relations & Axioms:** Apart from learning taxonomic relations among concepts as described in the previous paragraph, literature on learning other kinds of relations from social tagging data is still sparse. A possible reason for this is that a large portion of relation learning techniques based on text (see [18] for an overview) comprise the exploitation of syntactic dependencies or lexico-syntactic patterns, which do not exist in

[6] Of course one could also argue that the set of tags themselves can always be interpreted as a lightweight intensional description.

[7] Some social tagging systems like BibSonomy or Flickr allow users to create explicit directed tag relationships.

folksonomies. However, when reviewing the results of taxonomy learning techniques, it turns out that in some cases the learned taxonomic relations do not always convey a sharp and precise "is-a" semantics. In the examples given by the authors, one can also find occurrences of e.g. "part-of"-relationships or purpose-related connections. But the "disambiguation" of these relations remains so far an open and interesting research problem, as well as the extraction of more complex constructs like axioms.

### *4.4. Tag Sense Disambiguation*

Besides the aforementioned problem of synonymy, another major and often mentioned weakness of tagging systems is ambiguity of tags. Of course this it not an intrinsic problem of social tagging, but in a way "inherited" from the fact that tags can mostly be considerd as natural language entities. However, the openness and uncontrolledness of these systems makes this issue more visible. In principle, the task of disambiguating tag meanings belongs to the process of concept identification (see Section 4.3); but in order to clarify the different aspects and approaches, it is treated separately in this chapter. Statistical natural language processing distinguishes between supervised, dictionary-based and unsupervised disambiguation [38]. In all cases, information taken from the *context* of a term forms the basis for its assignment to a certain sense. In the process model of discovering tag semantics by [23], "context identification" is also included as a major step.

Supervised approaches are based on labelled training data, and learn usually a classifier based on context features of a given word. Such approaches have rarely been applied to social tagging sytems. Dictionary-based approaches rely on sense definitions defined in dictionaries or thesauri. [22] uses cosine similarity between tag co-occurrence vectors and a bag-of-words representation of Wikipedia pages to identify the most suitable sense definition within DBPedia[8]. [35] also computes a relevance score between tags and Wikipedia articles for the same purpose.

Unsupervised approaches are trying to partition the context of a given term into clusters corresponding to its different senses. [58] analyzed several folksonomy-derived networks with regard their suitability to derive senses by graph clustering algorithms. [59] proposed an entropy-based metric to capture the level of ambiguity of a given tag. [8] applied hierarchical agglomerative clustering of co-occuring tags to identify different senses.

As a last class of approaches, methods like the one proposed by [45] require the user to define the intended meaning during the tagging process by choosing among a set of possible senses.

### *4.5. Other Tasks*

Besides the above-mentioned major tasks, some other approaches of exploiting folksonomy semantics are thinkable and can be found in the literature. Instead of constructing an ontology from scratch, *ontology maintenance* and *refinement* are concerned with the insertion, update and removal of concepts in order to keep the captured knowledge up to date. Though folksonomies could possibly be useful to this end, existing approaches are still scarcely found; [24] discusses some ideas. The same holds for *ontology population*,

[8] `http://www.dbpedia.org`

which is focussed on finding instances of given concepts. [11] describes the SOBOLEO system for semantic annotation, which is intended to enhance tagging and foster ontology evolution. Another applications is to employ folksonomies for disambiguating web search queries [57].

## 5. Conclusions & Future directions

The main focus of this chapter was to give a detailed overview of the development and the state of the art in capturing emergent semantics from social tagging systems. Special properties of social tagging data compared to "traditional" input of ontology learning algorithms were discussed, along with a review of works analyzing emergent semantics and its influencing factors within these systems. In the sequel, relevant approaches on making these implicit semantic structures explicit were discussed, structured mainly along the different learning tasks defined in a small comparison framework.

In summary, there is solid evidence for emergent semantics, and existing as well as novel methods have been successfully adapted for knowledge extraction from tagging data. When a sufficient amount of tagging data for a given domain is available, it is very worth to be considered for the process of ontology engineering. Despite these promising results, folksonomy-based approaches alone can not be seen as the silver bullet of knowledge acquisition: Especially in the context of learning different kinds of relations (e.g. meronymy or even typed relations), current approaches still need to be further developed to reach the performance of other approaches based on e.g. the rich syntactic structure of natural language texts. Ultimately, a promising research direction is to analyze which data (coming from different social and traditional sources) is most suitable as an input to different kinds of knowledge acquisition tasks and applications.

## References

[1] Rabeeh Abbasi and Steffen Staab. RichVSM: enRiched Vector Space Models for Folksonomies. In *HyperText'09: Proc. of 20th ACM Conf. on Hypertext and Hypermedia*, 2009.

[2] Sofia Angeletou. Semantic enrichment of folksonomy tagspaces. In *Int'l Semantic Web Conference*, volume 5318 of *LNCS*, pages 889–894. Springer, 2008.

[3] Anupriya Ankolekar, Markus Krötzsch, Thanh Tran, and Denny Vrandecic. The two cultures: mashing up web 2.0 and the semantic web. In *WWW '07: Proc. of the 16th Int'l Conf. on World Wide Web*, pages 825–834, New York, NY, USA, 2007.

[4] Grigory Begelman, Philipp Keller, and Frank Smadja. Automated tag clustering: Improving search and exploration in the tag space. In *Proc. of the Collaborative Web Tagging Workshop at the WWW 2006*, Edinburgh, Scotland, May 2006.

[5] Dominik Benz, Marko Grobelnik, Andreas Hotho, Robert Jäschke, Dunja Mladenic, Vito D. P. Servedio, Sergej Sizov, and Martin Szomszor. Analyzing tag semantics across collaborative tagging systems. In *Proc. of the Dagstuhl Seminar on Social Web Communities*, number 08391, 2008.

[6] Dominik Benz and Andreas Hotho. Position paper: Ontology learning from folksonomies. In *Workshop Proc. of Lernen - Wissensentdeckung - Adaptivität (LWA 2007)*, pages 109–112. Martin-Luther-Universität Halle-Wittenberg, sep 2007.

[7] Dominik Benz, Andreas Hotho, Robert Jäschke, Beate Krause, Folke Mitzlaff, Christoph Schmitz, and Gerd Stumme. The social bookmark and publication management system bibsonomy. *The VLDB Journal*, 19(6):849–875, December 2010.

[8] Dominik Benz, Andreas Hotho, and Gerd Stumme. Semantics made by you and me: Self-emerging ontologies can capture the diversity of shared knowledge. In *Proc. of the 2nd Web Science Conference (WebSci10)*, Raleigh, NC, USA, 2010.

[9] Dominik Benz, Christian Körner, Andreas Hotho, Gerd Stumme, and Markus Strohmaier. One tag to bind them all : Measuring term abstractness in social metadata. In *Proc. of the 8th Extended Semantic Web Conference (ESWC 2011)*, Heraklion, Crete, May 2011.

[10] Chris Biemann. Ontology learning from text: A survey of methods. *LDV Forum*, 20(2):75–93, 2005.

[11] Simone Braun, Andreas Schmidt, and Valentin Zacharias. Soboleo: vom kollaborativen tagging zur leichtgewichtigen ontologie. In *Mensch & Computer - 7. Fachübergreifende Konferenz - M&C 2007*, pages 209–218, München, 2007.

[12] Christopher H. Brooks and Nancy Montanez. Improved annotation of the blogosphere via autotagging and hierarchical clustering. In *WWW '06: Proc. of the 15th Int'l Conf. on World Wide Web*, pages 625–632, New York, NY, USA, 2006. ACM Press.

[13] Alexander Budanitsky and Graeme Hirst. Evaluating wordnet-based measures of lexical semantic relatedness. *Computational Linguists*, 32(1):13–47, 2006.

[14] Ivan Cantador, Martin Szomszor, Harith Alani, Miriam Fernandez, and Pablo Castells. Enriching ontological user profiles with tagging history for multi-domain recommendations. In *1st Int'l Workshop on Collective Semantics: Collective Intelligence & the Semantic Web (CISWeb 2008)*, June 2008.

[15] Ciro Cattuto. Semiotic dynamics in online social communities. *The European Physical Journal C - Particles and Fields*, 46:33–37, August 2006.

[16] Ciro Cattuto, Dominik Benz, Andreas Hotho, and Gerd Stumme. Semantic grounding of tag relatedness in social bookmarking systems. In *The Semantic Web – ISWC 2008, Proc.Intl. Semantic Web Conference 2008*, volume 5318 of *LNAI*, pages 615–631, Heidelberg, 2008. Springer.

[17] Ciro Cattuto, Christoph Schmitz, Andrea Baldassarri, Vito D. P. Servedio, Vittorio Loreto, Andreas Hotho, Miranda Grahl, and Gerd Stumme. Network properties of folksonomies. *AI Communications*, 20(4):245–262, December 2007.

[18] Philipp Cimiano. *Ontology Learning and Population from Text: Algorithms, Evaluation and Applications*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[19] Klaas Dellschaft and Steffen Staab. On how to perform a gold standard based evaluation of ontology learning. In *Proc. of ISWC-2006 Int'l Semantic Web Conference*, Athens, GA, USA, November 2006. Springer, LNCS.

[20] Iyad Abu Doush and Enrico Pontelli. Integrating semantic web and folksonomies to improve e-learning accessibility. In *Computers Helping People with Special Needs*, volume 6179 of *LNCS*, pages 376–383. Springer Berlin / Heidelberg, 2010.

[21] Takeharu Eda, Masatoshi Yoshikawa, Toshio Uchiyama, and Tadasu Uchiyama. The effectiveness of latent semantic analysis for building up a bottom-up taxonomy from folksonomy tags. *World Wide Web*, 12(4):421–440, 2009.

[22] Andres Garcia, Martin Szomszor, Harith Alani, and Oscar Corcho. Preliminary results in tag disambiguation using dbpedia. In *Proc. of the First Int'l Workshop on Collective Knowledge Capturing and Representation*, September 2009.

[23] Andres Garcia-Silva, Oscar Corcho, Harith Alani, and Asuncion Gomez-Perez. Review of the state of the art: Discovering and associating semantics to tags in folksonomies. *Knowledge Engineering Review*, 26(4), December 2011.

[24] Domenico Gendarmi and Filippo Lanubile. Community-driven ontology evolution based on folksonomies. In *LNCS: On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops*, pages 181–188. Springer, 2006.

[25] Scott Golder and Bernardo A. Huberman. The structure of collaborative tagging systems. *Journal of Information Sciences*, 32(2):198–208, April 2006.

[26] Paul Heymann and Hector Garcia-Molina. Collaborative creation of communal hierarchical taxonomies in social tagging systems. Technical report, Computer Science Department, Standford University, April 2006.

[27] Andreas Hotho, Dominik Benz, Robert Jäschke, and Beate Krause, editors. *ECML PKDD Discovery Challenge 2008 (RSDC'08)*. Workshop at 18th Europ. Conf. on Machine Learning (ECML'08) / 11th Europ. Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD'08), 2008.

[28] Andreas Hotho, Robert Jäschke, Christoph Schmitz, and Gerd Stumme. Emergent semantics in bibsonomy. In *Informatik 2006 – Informatik für Menschen. Band 2*, volume P-94 of *Lecture Notes in Informatics*, Bonn, October 2006.

[29] Andreas Hotho, Robert Jäschke, Christoph Schmitz, and Gerd Stumme. Information retrieval in folksonomies: Search and ranking. In *The Semantic Web: Research and Applications*, volume 4011 of *LNCS*,

pages 411–426, Heidelberg, June 2006. Springer.

[30] Jason Jung. Matching multilingual tags based on community of lingual practice from multiple folksonomy: A preliminary result. In *Trends in Applied Intelligent Systems*, volume 6097 of *LNCS*, pages 39–46. Springer, Berlin / Heidelberg, 2010.

[31] Robert Jäschke, Andreas Hotho, Christoph Schmitz, Bernhard Ganter, and Gerd Stumme. Discovering shared conceptualizations in folksonomies. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(1):38–53, feb 2008.

[32] Lyndon Kennedy, Mor Naaman, Shane Ahern, Rahul Nair, and Tye Rattenbury. How flickr helps us make sense of the world: context and content in community-contributed media collections. In *MULTIMEDIA '07: Proc. of the 15th Int'l Conf. on Multimedia*, pages 631–640, New York, NY, USA, 2007. ACM.

[33] Beate Krause, Christoph Schmitz, Andreas Hotho, and Gerd Stumme. The anti-social tagger - detecting spam in social bookmarking systems. In *Proc. of the Fourth Int'l Workshop on Adversarial Information Retrieval on the Web*, 2008.

[34] Christian Körner, Dominik Benz, Markus Strohmaier, Andreas Hotho, and Gerd Stumme. Stop thinking, start tagging - tag semantics emerge from collaborative verbosity. In *Proc. of the 19th Int'l World Wide Web Conference (WWW 2010)*, Raleigh, NC, USA, apr 2010. ACM.

[35] Kangpyo Lee, Hyunwoo Kim, Hyopil Shin, and Hyoung-Joo Kim. Tag sense disambiguation for clarifying the vocabulary of social tags. In *CSE*, pages 729–734. IEEE Computer Society, 2009.

[36] Freddy Limpens, Fabien Gandon, and Michel Buffa. Bridging ontologies and folksonomies to leverage knowledge sharing on the social web: A brief survey. *Automated Software Engineering - Workshops, 2008. ASE Workshops 2008. 23rd IEEE/ACM Int'l Conference on*, pages 13–18, Sept. 2008.

[37] Mohamed Zied Maala, Alexandre Delteil, and Ahmed Azough. A conversion process from Flickr tags to RDF descriptions. *IADIS Int'l Journal on WWW/Internet*, 6(1), 2008.

[38] C. Manning and H. Schütze. *Foundations of statistical natural language processing*. MIT Press, Cambridge, MA, 1999.

[39] Leandro Balby Marinho, Krisztian Buza, and Lars Schmidt-Thieme. Folksonomy-based collabulary learning. In *Int'l Semantic Web Conference*, volume 5318 of *LNCS*, pages 261–276. Springer, 2008.

[40] Benjamin Markines, Ciro Cattuto, Filippo Menczer, Dominik Benz, Andreas Hotho, and Gerd Stumme. Evaluating similarity measures for emergent semantics of social tagging. In *18th Int'l World Wide Web Conference*, pages 641–641, April 2009.

[41] Adam Mathes. Folksonomies - cooperative classification and communication through shared metadata, December 2004.

[42] Pasquale De Meo, Giovanni Quattrone, and Domenico Ursino. Exploitation of semantic relationships and hierarchical data structures to support a user in his annotation and browsing activities in folksonomies. *Inf. Syst.*, 34(6):511–535, 2009.

[43] Peter Mika. Ontologies are us: A unified model of social networks and semantics. In *The Semantic Web - ISWC 2005, Proc. of the 4th Int'l Semantic Web Conference, ISWC 2005, Galway, Ireland, November 6-10*, volume 3729 of *LNCS*, pages 522–536. Springer, 2005.

[44] Borys Omelayenko. Learning of ontologies for the web: the analysis of existent approaches. In *Proc. of the Int'l Workshop on Web Dynamics, held in conj. with the 8th Int'l Conference on Database Theory (ICDT'01), London, UK*, 2001.

[45] A. Passant and P. Laublet. Meaning of a tag: A collaborative approach to bridge the gap between tagging and linked data. *Proc. of the WWW 2008 Workshop Linked Data on the Web (LDOW2008), Beijing, China, Apr*, 2008.

[46] Anon Plangprasopchok and Kristina Lerman. Constructing folksonomies from user-specified relations on flickr. In *WWW*, pages 781–790. ACM, 2009.

[47] Joni Radelaar, Aart-Jan Boor, Damir Vandic, Jan-Willem van Dam, Frederik Hogenboom, and Flavius Frasincar. Improving the exploration of tag spaces using automated tag clustering. In *Web Engineering*, volume 6757 of *LNCS*, pages 274–288. Springer Berlin / Heidelberg, 2011.

[48] Patrick Schmitz. Inducing ontology from flickr tags. In *Proc. of the Workshop on Collaborative Tagging at WWW2006*, Edinburgh, Scotland, May 2006.

[49] M. Shamsfard and A. Abdollahzadeh Barforoush. The state of the art in ontology learning: a framework for comparison. *The Knowledge Engineering Review*, 18(04):293–316, 2003.

[50] Clay Shirky. Ontology is overrated: Categories, links and tags, May 2005.

[51] Lucia Specia and Enrico Motta. Integrating folksonomies with the semantic web. In *Proc. of the 4th European conference on The Semantic Web: Research and Applications*, volume 4519/2007 of *LNCS*,

pages 624–639. Springer Berlin / Heidelberg, 2007.

[52] Marta Tatu and Dan I. Moldovan. Inducing ontologies from folksonomies using natural language understanding. In *LREC*. European Language Resources Association, 2010.

[53] Maurizio Tesconi, Francesco Ronzano, Andrea Marchetti, and Salvatore Minutoli. Semantify del.icio.us: Automatically turn your tags into senses. In *Proc. of the Workshop Social Data on the Web (SDoW2008)*, 2008.

[54] Robert Wetzker, Carsten Zimmermann, Christian Bauckhage, and Sahin Albayrak. I tag, you tag: translating tags for advanced user models. In *WSDM*, pages 71–80. ACM, 2010.

[55] Xian Wu, Lei Zhang, and Yong Yu. Exploring social annotations for the semantic web. In *WWW '06: Proc. of the 15th Int'l Conf. on World Wide Web*, pages 417–426, New York, NY, USA, 2006. ACM Press.

[56] Mu Xia, Yun Huang, Wenjing Duan, and Andrew B. Whinston. Ballot box communication in online communities. *Commun. ACM*, 52(9):138–142, 2009.

[57] Ching Man Au Yeung, Nicholas Gibbins, and Nigel Shadbolt. Web search disambiguation by collaborative tagging. In *Proc. of the Workshop on Exploiting Semantic Annotations in Information Retrieval (ESAIR 2008), co-located with ECIR 2008, Glasgow, United Kingdom, 31 March, 2008*, pages 48–61, 2008.

[58] Ching Man Au Yeung, Nicholas Gibbins, and Nigel Shadbolt. Contextualising tags in collaborative tagging systems. In *HT '09: Proc. of the 20th ACM Conf. on Hypertext and hypermedia*, pages 251–260, New York, NY, USA, 2009. ACM.

[59] Lei Zhang, Xian Wu, and Yong Yu. Emergent semantics from folksonomies: A quantitative study. *Journal on Data Semantics VI*, 2006.

[60] Mianwei Zhou, Shenghua Bao, Xian Wu, and Yong Yu. An unsupervised model for exploring hierarchical semantics from social annotations. In *Proc. of the 6th international The semantic web and 2nd Asian conference on Asian semantic web conference*, pages 680–693, 2008.

[61] M. Tame Özsu and Lin Liu. Encyclopedia of database systems, 2009.

# Semantic Enrichment of Places: From Public Places Descriptions to Linked Data

Ana Oliveira ALVES [a,b] and Francisco Câmara PEREIRA [a],
[a] *CISUC, Centre for Informatics and Systems of University of Coimbra, Portugal*
[b] *ISEC, Coimbra Institute of Engineering, Portugal*

**Abstract.** We present a methodology for extracting Lightweight Ontologies from textual descriptions about Public Places available on the Web. In this context, a place is a "Point of Interest" (POI), composed generally of a latitude/longitude pair and a name. In our approach, we enrich this information using the KUSCO system, which builds a Semantic Index for a given POI through Natural Language Processing techniques and Statistical computing over collected information on the Web. This process is called "Semantic Enrichment of POIs". Semantic Indexes are then contextualized and mapped to the Linked Web Data [1]. This mapping is the focus of this chapter. Thus, the POI can be represented not by a bag of words but instead by an interlinked cloud of concepts that enable us to infer more knowledge about a place.

**Keywords.** Semantics of Place, Lightweight Ontologies, Information Extraction

## 1. Introduction

From the human perspective, places are often associated with meaning, and different people relate to places in different ways. The meaning of place is derived from social conventions: its private or public nature, possibilities for communication, and many other factors. By definition, a POI is a place with meaning to someone, and the perception of this meaning is the main objective of our research.

It is noticeable that a place can be described or referenced according to a range of perspectives, depending on what is intended to be communicated (e.g. its function or its physical properties or its content or its relationship with the subject). We use different online resources from where we extract information related to POIs: the Yahoo!Local POI directory, Wikipedia, Boston Calendar and the open World Wide Web (using Yahoo Search API). From the point of view of each resource, a different perspective on each place can be found. In Yahoo!Local, some places have rich descriptions and their official website is indicated; in Wikipedia, places are described with historical and geographical viewpoints; in the Boston Calendar, the dynamic life of places becomes apparent through the flow of events that happen in the city; using regular web search, the possibilities are immense, as is the resultant ambiguity, but often places are described thoroughly in their specific home pages. Our system, KUSCO, applies NLP tools (cf. Maynard and Bontcheva [2] in this volume), such as POS tagging, Noun Phrase Chunking and Named Entity Recognition, to extract relevant terms from these textual descriptions about places.

Some filtering heuristics are applied such as stop-list and geographic word removal, and the relevance of each term is computed by TF-IDF. After selecting N top relevant terms, we can see each POI as a *bag of words* representing its main attributes, products and services.

Nowadays we have the Linked Open Data (LOD) project [1] that connects different Ontologies and provides Web APIs to consult interlinked data. This project includes WordNet [3], Yago [4] and DBpedia [5] covering multi-domain knowledge that is used in our system to contextualize terms related to public places. These places are classified in SUMO [6], a Upper Ontology also present in LOD. Thus, the information extracted is completely interlinked in the cloud from the bottom (concepts) to the bottom (public places).

This chapter is organized in the following manner: the second section introduces the reader to the main subjects present in this work and to the related state of the art, in order that the reader may become familiar with the concepts and algorithms used; the third section presents our model of Semantic Enrichment of Place materialized in a implemented system named KUSCO; the fourth section illustrates the KUSCO performance through examples and experimental evaluation; and finally the last section summarizes the conclusions and points the next steps in this research.

## 2. Background

### *2.1. Semantics of Place*

As argued in [7], absolute position such as the pair latitude/longitude is a precise representation of place but with no meaning associated. Beyond position, the name and labels associated to location are also of great importance in acquisition of places[8]. While this is true to *personal* places, we think this is also true to public places. According to [9], place consists of three components: physical setting, thus the locale of a place, activities performed at a place and the meaning of a place to the *public* and the individual. In an urban view of a city, people generally create POIs referring to *buildings* than to other categories of places like parts inside buildings, regions, junctions, and others[10].

The possibility to automatically associate labels has been investigated in the literature. These labels are generally limited to generic and personal ones like work, home, friends, etc. However, our approach is not centered on the user. Instead, it focuses on the place itself and this representation should include public aspects and the functionality of places, since the relation between a specific individual and the place itself is not of great importance. We think that a richer representation of a public Place with more meaningful commonsense concepts associated will complement works such as those previously described.

In [11], the authors propose a semi-automatic process of tag assignment, which integrates knowledge from Semantic Web ontologies and the collection of Web2.0 tags. This approach should be the, theoretically, correct one: it shares the formal soundness of Ontologies with the informal perspective of social networks. However it is a bit hard to implement: for each new POI/category the main points have to be chosen manually. The dynamics of this kind of information, particularly when depending on Web 2.0 social networks, would demand enormous resources to keep the information up to date, and

the compliance with semantic standards already seems unlikely to be widely accepted by individual users.

Working in a different direction, Rattenbury et al. [12] identify places and events from tags that are assigned to photos on Flickr. They exploit the regularities of tags which regard to time and space at several levels, so when "bursts" (sudden high quantities of a given tag in space or time) are found, they become an indicator of an event of meaningful place. Accordingly, the reverse process is possible, the search for the tag clouds that correlate with that specific time and space. They do not, however, make use of any enrichment from external sources, which could add more objective information, and their approach is limited to the specific scenarios of Web 2.0 platforms that carry significant geographical reference information.

Other attempts have also been made towards analyzing Flickr tags [13,14] by applying ad-hoc approaches to determine "important" tags within a given region of time [13] or space [14] based on inter-tag frequencies, or visualizing them over areas of the World [15]. However, no determination of the properties or semantics of specific tags has been provided [12].

In the Web-a-Where project, Amitay et al. [16] associate web pages to geographical locations to which they are related, also identifying the main "geographical focus". The "tag enrichment" process thus consists on finding words (normally Named Entities) that show potential for geo-referencing, and then applying a disambiguation taxonomy (e.g. "MA" with "Massachusetts" or "Haifa" with "Haifa/Israel/Asia"). The results are very convincing, but the authors do not explore the other idea beyond using explicit geographical references. An extension could be to detect and associate patterns such as those referred above in [12] without the need for explicit location referencing.

Our work focuses on the semantic aspect of location representation. Furthermore, we also take advantage of information available on the Web about public places. With the rapid growth of the World Wide Web, a continuously increasing number of commercial and non-commercial entities are acquiring a presence online, whether through the deployment of proper web sites or by referral from related institutions. This presents an opportunity for identifying the information which describes how different people and communities relate to places, and thereby enrich the representation of POIs. Nowadays, this information found on the Web is rarely structured or tagged with semantic meaning. Indeed, it is widely known that the majority of online information contains unrestricted user-written text. Hence, we become dependent primarily on Information Extraction (IE) techniques (cf. Suchanek [17] in this volume) for collecting and composing information from textual descriptions.

### *2.2. Location-based Web Search*

Location-based web search (or *Local Search*) is one of the most popular tasks expected from search engines. A location-based query consists of a topic and a reference location. Unlike general web search, in location-based search, a search engine is expected to find and rank documents which are not only related to the query topic but also geographically related to the location to which the query is associated. There are several issues concerning developing effective geographic search engines and, until now no global location-based search engine has been reported to achieve them [18]. Amongst the most noticeable difficulties are location ambiguity, lack of geographic information on web pages,

language-based and country-dependent variation in addressing styles, and multiple locations related to a single web resource.

Search engine companies have started to develop and offer location-based services. However, they are still geographically limited, mostly to the United States, such as Yahoo!Local, Google Maps and MSN Live Local, and have not become as successful and popular as general search engines. Also the results generally presented are related to their business directories and not to Web documents. Despite this, a lot of work has been done in improving the capabilities of location-based search engines [19,16], but this is beyond of the scope of this chapter. Instead, we make use of generally available search engines and formulate queries using the geographical reference to retrieve information about places. The work in this context is more focused on contributing to the indexing capabilities of such engines in terms of local search (finding an inspiration in [20]) than on becoming any alternative form of search.

### *2.3. Semantic Web*

Although a decade has passed since its definition by Tim Berners-Lee [21], the *Semantic Web* is a visionary architecture of the Web where all on-line information can be processed by machines. This can only happen if data is structured probably as in Ontologies, by axioms defining entities, their properties and the relations between them.

Since it is an ambitious task to represent all knowledge about the World, even for a restricted subset such as Places, the focus of this research is not to create a complete general Ontology about places from scratch or even Domain Ontologies about different types of places. Since one of the first motivations to build ontologies is for knowledge sharing, here the intention is to reuse structured commonsense knowledge and instantiate this knowledge, mainly concepts, related to Public Places. If we want to use ontology technology to increase interoperability between multiple representations or increase access to existing data, we need to build ontologies that are linked to the existing knowledge organization systems (KOS) [22].

Technically, *Linked Data* refers to data published on the Web in such a way that it is machine-readable, its meaning is explicitly defined, it is linked to other external data sets, and it can in turn be linked to and from external data sets [1]. The concepts inferred from textual descriptions of places are interlinked to the following commonsense ontologies present in the Open Linked Data cloud: WordNet, Wikipedia, YAGO and DBpedia. The details of this mapping are presented in sections 4 and 5. Two other resources interlinked in the Open Linked Data project are used in a new perspective: NAICS and SUMO. The first is a business standard coding system and the latter is a generic upper ontology. Both resources are detailed in the following subsections in order to introduce the reader to the mapping process.

#### *2.3.1. North American Industry Classification System (NAICS)*

In industry, classification systems serve to communicate important facts about a company. These codes are generally controlled by a governmental, a professional, trade or by an international standards organization. They often serve as shorthand for users interested in material in a particular area

The North American Industry Classification System(NAICS)[23] is an example of an official POI classification system which is the standard system used by Federal statis-

tical agencies in classifying business establishments for the purpose of collecting, analyzing, and publishing statistical data related to the U.S. business economy. The NAICS was developed under the auspices of the Office of Management and Budget (OMB), and was adopted in 1997 to replace the old Standard Industrial Classification (SIC) system.

The NAICS is a two through six-digit hierarchical classification code system, offering five levels of detail. Each digit in the code is part of a series of progressively narrower categories, and the more digits in the code, the greater the classification detail. The first two digits designate the economic sector, the third digit designates the sub-sector, the fourth digit designates the industry group, the fifth digit designates the NAICS industry, and the sixth digit designates the national industry. A complete and valid NAICS code contains six digits. Figure 1 shows part of the NAICS hierarchy regarding the “Museum" category of Public Place.

71-Arts, Entertainment, and Recreation
711-Performing Arts, Spectator Sports, and Related Industries
7111-Performing Arts Companies
712-Museums, Historical Sites, and Similar Institutions
7121-Museums, Historical Sites, and Similar Institutions
**71211-Museums**
712110-Art galleries (except retail)
712110-Art museums

**Figure 1.** Example of the NAICS hierarchy

### *2.3.2. SUMO*

The Suggested Upper Merged Ontology (SUMO) is an upper ontology developed by IEEE’s Standard Upper Ontology (SUO) Working Group as an open source initiative with the purpose of creating a public standard, accessible through the Web [24].

The ontology is being progressively created through the integration of public content. For instance, the YAGO-SUMO integration incorporates millions of entities from YAGO into SUMO. With the combined force of the two ontologies, an enormous, unprecedented corpus of formalized world knowledge is available for automated processing and reasoning, providing information about millions of entities. From these entities, we are concerned with public geo-referenced entities such as *points-of-interest*, *organizations*, and *companies*[25]. Compared to the original YAGO, more advanced reasoning is possible due to the axiomatic knowledge delivered by SUMO. For example, a reasoner can conclude that a soup dish holds soup while the soup is being eaten, or that a service elevator is a transportation device consisting of a car that moves up and down in a vertical shaft for carrying freight so that objects can move from one floor to another in a building. Another instance of such interlinked data is the integration of NAICS-SUMO achieved by SUMO authors [26] [1]. Despite the fact that the NAICS taxonomy is barely mapped to SUMO, the main economy sectors are uniquely indicated. This is illustrated by comparing the excerpt from the NAICS hierarchy about the “Museum” category in figure 1 with the same information inferred as RDF triples in SUMO, shown in figure 2.

[1] Available through the SUMO ontology portal: http://sigmakee.cvs.sourceforge.net/viewvc/sigmakee/KBs/naics.kif

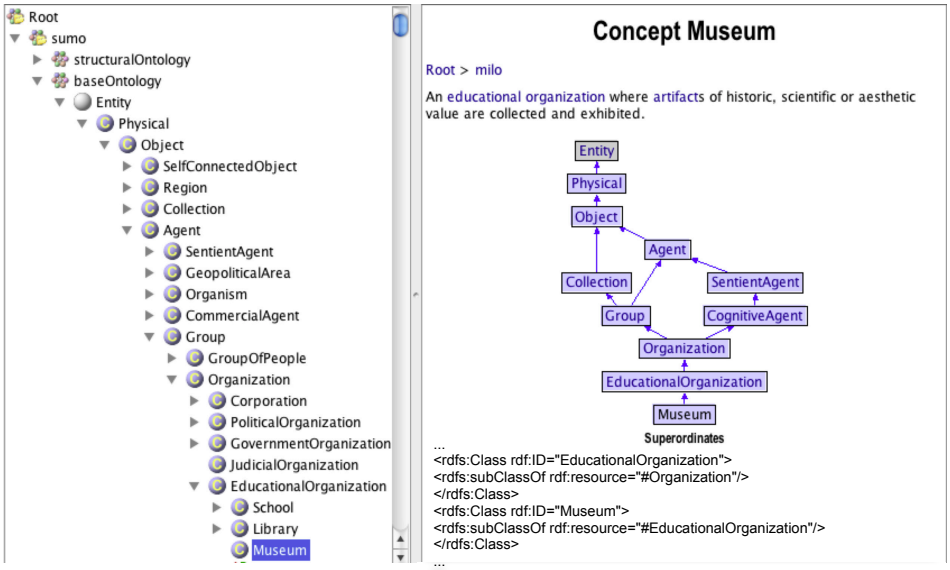



**Figure 2.** Excerpt of the root NAICS hierarchy mapped to SUMO.

## 3. The KUSCO System

A system that is able to extract relevant semantics from public places can be useful for any context-aware system that behaves according to position. The level of information considered in this work adds another layer to other sensors (GPS, accelerometer, compass, communications, etc.), eventually pushing forward the potential for intelligent behavior. This section presents an approach to such a system and its implementation, resulting in an architecture called KUSCO: *Knowledge Unsupervised Search for instantiating Concepts on lightweight Ontologies*. The figure 3 shows the overall KUSCO's architecture. Each module in this architecture is briefly presented next. The firs two modules were introduced in previously published work [27,28,29], while the latter two modules, the focus of this chapter, will be detailed in the next section. The expected processing and data flow in this system is as follows:

- *A POI* Source is specified as input for the system. In the present architecture, the structure of this source (e.g. a collection of documents, a directory Web site, a POI-finding API) is extracted in order to automatically populate a database of POIs. This intensive extraction may be accomplished by simply invoking API (when available) or by Web scraping [27]. KUSCO uses regular expressions to extract POIs from different POI websites and in different formats and integrates them in a conceptual schema that accommodates different levels of information. This choice was made instead of starting from POIs that are already disambiguated in LOD such as those from DBpedia. Apart from the well defined structure of information that we could use from DBpedia for some of the most important public places, we think that using an external POI source is a more intensive approach in order to get exhaustively the most of POIs from a given city

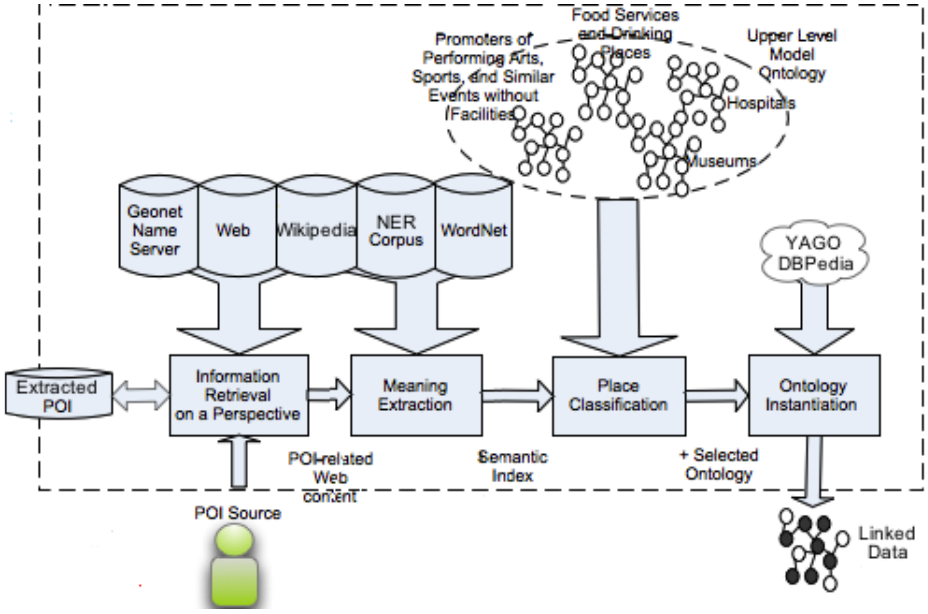



**Figure 3.** The KUSCO system architecture.

of study (including also less popular public places). As multiple POI sources are explored, it is important to have a way to identify similar POIs, so that we don't end up with redundant information and also to collect as much information as possible about a given place. This requires a way to identify similarities based, not only in proximity, but also in name likeness. Our approach makes use of the JaroWinklerTF-IDF class from the SecondString project [30] to identify close names, ignoring misspelling errors and some abbreviations.

- *Information Retrieval on a Perspective* consists of finding documents about each POI from a given background collection. The World Wide Web and Wikipedia were explored to retrieve such information applying three overall different approaches [28]. The system explores the Web in a focused search. The Wikipedia knowledge source is used through two distinct ways to locate generic and specific information about places. For these three combinations of selecting the subset of documents in the source collection, we name each one as a different *Perspective* of the semantic enrichment:

  * The About Perspective crawls the web using a search engine restricting the universe of search to the Official Website of a POI when it is available from POI sources. KUSCO tries to find in the POI Official Website the information which is focused on the purpose, services offered or mission, or in other words, what the POI itself is. After some observations, we concluded that all the information KUSCO needs is quite often found on the "About Web Page" or "Info Page" which concisely presents the company/POI to visitors. With the availability of general search engines (e.g. Google) to restrict a web search to

a given Website, the approach to retrieve the About Page of a given POI basically consists of using this restricted search on the POI Official Website with different combinations of queries around "About us", "About this company", etc.

* The Wikipedia Perspective takes advantage of plenty of relevant information about places which is obtainable from Wikipedia. Both by searching directly for the actual Wikipedia page of a POI (e.g. Starbucks), and indirectly by finding information related to its category (e.g. Restaurant), KUSCO currently applies two variations on searching Wikipedia:
in the ***Red Wiki perspective*** retrieves the Wikipedia page corresponding to the identified set of categories of a POI. Since some POI sources are well structured, that is, they have a hierarchical taxonomy of disjoint categories, we believe that the automatic tagging of places, initially considering the category they belong to, is a first step in labeling that place using encyclopedic knowledge (Wikipedia). Since in most cases no API is currently available from those local directories studied, a wrapper was created based on regular expressions, in order to automatically extract the category taxonomy from each local directory. Through time, these taxonomies grow with new types of services and places. In this way, by using specific wrappers to each POI provider, it is possible to run them periodically to integrate new categories in the respective stored taxonomy. Contextualizing category names in their corresponding Wikipedia articles is based mainly on string similarity between them. We have opted for a top-down approach, beginning with the main categories and continuing until the taxonomy leaves are reached. To increase the confidence in this process, we chose to disambiguate the main categories manually at first and to make sure that at least a more specific category would be connected to the wikipages of its hypernym. When a POI has many categories, KUSCO obtains the articles for each one and considers the union of all the resulting articles as the source of analysis. Since there are many different combinations of categories, it is possible to guarantee that each POI is subjected to its own specific flavor of category;

in the ***Yellow Wiki perspective***, KUSCO tries to find the Wikipedia page corresponding to the POI itself, when its available. While the above approach is centered on place category, in this approach the focus is on place name. Once more, KUSCO uses string similarity to match Place name to a Wikipage title in order to find the Wikipedia description for a given place. At first glance, this method is efficient in mapping compound and rare place names such as 'Beth Israel Deaconess Medical Center' or 'Institute of Real Estate Management'. However, it can naively induce some wrong mappings for those places with very common names (e.g. Registry - a recruitment company in Boston, Energy Source - a battery store in New York). This problem was tackled by determining the specificity of place names, and considering only those with high Information Content (IC) [31]. The Information Content of a concept is defined as the negative log likelihood, $-logp(c)$, where $p(c)$ is the probability of encountering such concept. For example, 'money' has less information content than

'nickel' as the probability of encountering the concept of 'money', $p(Money)$, is larger than the probability of encountering the concept 'nickel', $p(Nickel)$, in a given corpus. For those names present in WordNet (e.g. Registry), the IC is already calculated [32], whereas for those not present, it is heuristically assumed that they are only considered by KUSCO if they are not a node in the Wikipedia taxonomy, i.e. a Wikipage not representing a Wikipedia category (as in the case of Energy Source), but only being a Wikipedia article.

- *Meaning Extraction* finds the bag of relevant concepts in a document retrieved from a given source (Web or Wikipedia). This process includes Noun Phrase chunking and Named Entity Recognition (NER) (cf. Maynard and Bontcheva [2] in this volume) using available Natural Language Processing (NLP) tools [33,34]. On completion of these subtasks, for each textual description, KUSCO ranks their within concepts with TF-IDF [35] (Term Frequency $\times$ Inverse Document Frequency) value in order to extract the most relevant terms that will represent a given place [29]. Instead of a common bag of words, this set contains terms with meaning, since the disambiguation of each term is performed in this module (see the next section for details).

- *Place Classification* performs the task of classifying a given POI within an Upper Level Ontology, allowing the next module to specify the concepts, relations and attributes in this new instance of the suggested ontology.

- *Ontology Instantiation* is the final and proposed module for reusing shared and linked knowledge in order to find and induce already known and new concepts, relations and attributes about a given POI. At the Meaning Extraction Module, only concepts are extracted from Web pages describing places. But here, once the right generic class into a Upper Level Ontology has been found for a given POI, those concepts will be used to instantiate this ontology. As the Ontology is composed not only of concepts but also of relations, the original context where concepts appear inside the Web page will be used to instantiate relations between them. For instance, considering a POI semantic index which contains concepts like: 'facility', 'vegetarian', 'Portuguese', 'terrace', and 'bar', this POI is correctly categorized as a Restaurant by respectively matching some classes in the corresponding ontology: Facility, VegetarianCuisine, PortugueseCousine, OutdoorSeatingOnTerrace and WineBarCuisine. An ontology typically does not have all possible instantiations of generic concepts in a given domain. This instantiation process is possible by WordNet semantic relations, which are implicit by contextualizing concepts from Web pages, and non-taxonomic relations that connect semantically related concepts. So, although the concept 'Dim Sum' from a POI semantic index does not appear explicitly in the Ontology, the WordNet semantic relation in 'Dim Sum' as *a type of* 'Cousine' is useful to capture which classes and their types occur in a given POI. Another example comes from the concept "private dining room facility", although it does not appear in WordNet, we are able to identify its connection with the 'Facility' class by Lexico-syntactic patterns [36] (cf. Maynard and Bontcheva [2] in this volume).

In the next sections, the term disambiguation inside the Meaning Extraction module and automatic Place Classification are detailed since this is the linking, at one hand, of semantic indexes produced by KUSCO, and on the other hand, of POIs to the Linked Data.

## 4. Term Disambiguation

The intermediate result of the KUSCO meaning extracting module (fig. 3) is a Semantic Index composed mainly of concepts (words with meaning) instead of terms. Although many approaches have been studied in Word Sense Disambiguation, none of them has achieved a satisfactory level of accuracy. Moreover, the heuristics of taking the most common sense of a given word shows better results that computationally intensive and complex techniques [37]. KUSCO takes advantage of WordNet and Wikipedia in order to map terms to linked concepts. A Semantic Index of a place in a given perspective is generally made up of both types of elements: generic concepts (noun phrases) and entities (named entities representing places, locations and organizations). Our approach only considers the first group to term disambiguation. This mapping is done in the following way: simple noun phrases (one term) is often contextualized in WordNet, while the compound noun phrases (two or more terms) sometimes representing instances of generic concepts, is quite often contextualized in Wikipedia[2] when there is unambiguously a related article about each instance. This approach for contextualizing noun phrases is formalized by algorithm 1.

WordNet is used here to map words to concepts choosing in each case the most probable sense through corpus occurrence. For example, the term “library" has (at least) two meanings in WordNet: “a room where books are kept" or “a collection of standard programs and subroutines that are stored and available for immediate use"; the first meaning is the most frequently used if we consider statistics from a corpus. The corpus used is the SemCor corpus which was especially developed for WordNet and which is manually annotated with synset occurrences [32].

WordNet is not intended to be a complete resource. We can still check if a given noun phrase corresponds to an instance in WordNet, but even for compound nouns the lack of information is broadly recognized. To solve this problem, we use Wikipedia. This is, in our view, the best open common-sense resource suited for compound noun disambiguation, particularly for named entities. To disambiguate proper or compound nouns not found in WordNet, the same approach is used here. The most common use given to a word is queried in Wikipedia. If it returns a disambiguation page, then the system simply continues with the term decontextualized. This approach is highly dependent on the availability of article entries related with a given noun phrase and the discernment of authors on indicating the right place of a given article. For instance, the term *wheelchair access* being only existent in Wikipedia, is redirected to the article *wheelchair* instead of the article more lexically and semantically closer, *wheelchair accessible*. Another concrete example of ambiguity, due to the redirection of related Wikipedia titles but not exactly meaning the same concept, is the the term *drug stores*. Despite the name is correctly found on Wikipedia it is redirected to *pharmacy* and this article is more related to the

[2]Besides the fact that WordNet is also explored, but as discussed later, this resource is not so rich as Wikipedia in terms of instances and compound terms

profession and not the place where medicaments are dispensed, being the article titled *community pharmacy* the more precise choice.

**Algorithm 1** Noun Phrase Disambiguation: map a noun phrase (simple or compound) to a lexical resource entry

**Require:** $np$ : `noun phrase`
**Ensure:** $map$ : `mapped entry in lexical resource with a term count associated` $\vee$ $SemCor$ : `Anotated WordNet Corpus`
  $WNmap \Leftarrow$ `WordNetDisambiguation`$(np, SemCor)$
  $WKmap \Leftarrow$ `WikipediaDisambiguation`$(np)$
  **if** (`NumberOfWords`$(WKmap) >$ `NumberOfWords`$(WNmap)$) **then**
    $map \Leftarrow WKmap$
  **else**
    $map \Leftarrow WNmap$
  **end if**
  **return** $map$

  **function** WordNetDisambiguation($np$, $SemCor$): $WNmap$
  $candidateSynsets \Leftarrow$ `FindSynsetsInWordNet`$(np)$
  $selectedSynset \Leftarrow MostFrequentNoInstanceSynset(candidateSynsets, SemCor)$
  {each mapping contains also the hit count, every count is initialized with 1}
  $WNmap \Leftarrow (selectedSynset, 1)$
  **return** $WNmap$
  **end function**

  **function** WikipediaDisambiguation($np$): $WKmap$
  $candidatePage \Leftarrow$ `SearchWikipediaAPI`$(np)$
  **if** (IsDisambiguationPage(candidatePage) is False ) **then**
    $selectedPage \Leftarrow candidatePage$
    $WKmap \Leftarrow (selectedPage, 1)$
  **end if**
  **return** $WKmap$
  **end function**

## 5. Automatic Place Classification

Different approaches were implemented and tested to classify POIs into a common taxonomy, which would allow, for instance, to connect directly a place to the SUMO Upper Level Ontology. Given a source of POIs generally there is a proper taxonomy of POI categories, each one classifying them by related terms (e.g. Entertainment Venues or Live Theaters), it is necessary to elect a common classification to use. The aim is to be able to classify POIs from different sources to a common and more widespread taxonomy like NAICS. Doing so is essential in order to perform a proper analysis of the extracted POIs. If the POIs are not mapped to a common taxonomy we will not be able to determine, for instance, how many POIs of restaurants exist in a given area because one POI source may classify them as “Restaurants” and the another as “Eating places”. The approaches here proposed to automatically classify POIs in a given taxonomy rely mainly on category information from POI sources which are also organized in taxonomies. Because

| POI name | Location | Categories | POI source url |
|---|---|---|---|
| **Au Bon Pain** | 1 State St Plz New York, NY | Carry Out & Take Out, Bakeries, Restaurants | http://local.yahoo.com /info-11039898-au-bon-pain-new-york |
| **Boston Athenaeum** | 10 Beacon St, Boston, MA | Tourist Attractions, All Entertainers, Art Museums & Galleries, Libraries | http://local.yahoo.com /info-10150557-boston-athenaeum-boston |
| **Erbaluce** | 69 Church St Boston, MA | Italian Restaurants, Restaurants | http://local.yahoo.com /info-46631862-erbaluce-boston |
| **Petco** | 1210 Providence Hwy, Norwood, MA | Pet Supplies, All Animal Services | http://local.yahoo.com /info-10144601-petco-norwood |
| **Radio Shack** | 925 Lexington Ave, New York, NY | Computer Software, Electronics Retailers, Cellular Phones | http://local.yahoo.com /info-11112320-radio-shack-new-york |

**Table 1.** Some example of POIs from the greater metropolitan area of Boston and New York considering the Yahoo! Local API.

the same POI is sometimes assigned to more than one category in some POI sources, the number of possible combination can be huge. As a consequence, finding mappings between the source taxonomy and the target taxonomy is not always a trivial task. Consider the following mappings:
"Newspaper Publishers" -> "Newspaper Publishers"
"Newspapers Printing" -> "Newspaper Publishers"
"Laboratories" -> "Research & Development in Biotechnology"
Different approaches were studied: Ontology Matching and Lexical/Semantic Similarity, but the most profitable was the application of Machine Learning (ML) algorithms to automatically classify the NAICS code of a given POI [27]. More precisely, the most successful algorithm was a K-nearest neighbor classifier, named IBk [38] (with $k = 1$), which essentially finds the similar test case and assigns the same NAICS code. This algorithm obtained approximately an accuracy of 85%, 76.8% and 73.1% for the automatic classification of POIs in the two, four and six-digit NAICS code respectively.

## 6. Illustrative Examples

In this section, a list of some semantic indexes is presented as the retrieval of information made previously by KUSCO for each perspective. Thus, for the same set of POIs, different views are presented using the Web or Wikipedia as source for the semantic enrichment process. Considering the states of Massachusetts and New York in the U.S., we chose the Yahoo! Local API as the POI source for this examples as shown in table 1.

For each POI under consideration, different perspectives are applied and the top-5 most relevant terms has been found by KUSCO for each of them as shown in table 2. In the About perspective, it is not always possible to find a candidate for the *About* page due either to the completely nonexistence of such page in the POI website or to the organization of the About section with other submenus making it hard to find the best one

to elect as the About page (e.g. Erbaluce's website). In the Red Wiki perspective, when there is no direct mapping in Wikipedia for a given category name or when this name is ambiguous it is used the immediately upper category mapping in the POI taxonomy. In table 3, these categories are highlighted in **bold** (e.g. "All Animals Services" and "Pet Supplies" are mapped to Wikipedia through its subsumer category "Animals & Pets").

Also, in this perspective, all terms that are already known as category names of a given POI are filtered out in its semantic index in order to avoid duplicate information. For instance, in the case of the POIs *Au Bon Pain* and *Boston Athenaeum*, the terms *restaurant* and *Library* were removed from their respective semantic indexes. Furthermore, in this last POI the term *Museums* was maintained by KUSCO as it is a generalization of *Art Museum* and not meaning the same of one of the categories already known from the POI *Boston Athenaeum*.

These extracted terms are then mapped to WordNet and Wikipedia following the algorithm 1. Thus, they can be considered as *concepts* linked to the cloud, as shown in table 3. This mapping is done selecting the most used sense of a given term in the respective lexical resource. While in Wikipedia the only restriction is to guarantee that a given term is not ambiguous (i.e. the obtained sense is not a disambiguation page), for WordNet, the sense chosen has to be the most frequently used in the SemCor corpus and it is only considered if it does not refer to an instance of something. As an example, consider the terms *kiosk*, *soup*, *table service*, *Pizza delivery* and *Tandy*. The first term is unambiguous and is found in WordNet. The second one is ambiguous, since it has more than one meaning in WordNet, being its most used sense picked up. The third exists in WordNet and Wikipedia, but as it is not an instance in WordNet, this meaning is preferred over the Wikipedia definition. The fourth only exists in Wikipedia and it is not ambiguous since its article is not a disambiguation page. Finally, the fifth term is present in both resources, but while in WordNet it is a instance of the concept *actress*, in Wikipedia it is connected to more than one article (disambiguation is needed). Thus, this last term remains decontextualized.

Table 3.: Disambiguation of most relevant terms in each perspective. For each definition the gloss (in WordNet) or the first sentence of the article abstract (in Wikipedia) is shown.

| POI | WordNet/Wikipedia Mapping for Most Relevant Terms |
|---|---|
| **Au Bon Pain** | Soup (WN) liquid food especially of meat or fish or vegetable stock... |
| | breads (WN) food made from dough of flour or meal and usually raised with yeast or baking powder and then baked |
| | plate (WN) a sheet of metal or wood or glass or plastic |
| | kiosk (WN) small area set off by walls for special use |
| | restaurant (WN) a building where people go to eat |
| | Pizza delivery (WK) ...is a service in which a pizzeria delivers a pizza to a customer... |
| | food (WN) any substance that can be metabolized by an animal to give energy and build tissue |
| | fast food (WN) inexpensive food prepared and served quickly |
| | table service (WN) tableware consisting of a complete set of articles for use at table |
| | Good (WN) articles of commerce |
| | locations (WN) a point or extent in space |
| | cafes (WN) a small restaurant where drinks and snacks are sold |
| | associate (WN) a person who joins with others in some activity or endeavor |
| | Membership (WN) the body of members of an organization or group |
| | gathering (WN) a group of persons together in one place |

Continued on Next Page...

Table 3 – Continued

| POI | WordNet/Wikipedia Mapping for Most Relevant Terms |
|---|---|
| **Boston Athenaeum** | opportunities (WN) a possibility due to a favorable combination of circumstances<br>dance (WN) taking a series of rhythmical steps (and movements) in time to music<br>Library (WN) a collection of literary documents or records kept for reference or borrowing<br>Museum (WN) a depository for collecting and displaying objects having scientific or historical or artistic value<br>collections (WN) several things grouped together or considered as a whole<br>landmark building (WK) ...a geographic feature used by explorers and others to find their way back or through an area...<br>galleries (WN) a porch along the outside of a building (sometimes partly enclosed) |
| **Erbaluce** | cuisines (WN) the practice or manner of preparing food or the food so prepared<br>restaurateur (WN) the proprietor of a restaurant<br>Premises (WN) statement that is assumed to be true and from which a conclusion can be drawn<br>delivery service (WK) *A service delivery framework (SDF) is a set of principles, standards, policies and constraints used to guide the design, development, deployment, operation and retirement of services delivered by a service provider...*<br>table wines (WN) wine containing not more than 14% alcohol usually served with a meal<br>wine grape (WN) A grape is a non-climacteric fruit, specifically a berry, and from the deciduous woody vines of the genus Vitis<br>sweet wines (WN) The subjective sweetness of a wine is determined by the interaction of several factors: the amount of sugar, the relative levels of alcohol, acids, and tannins, etc. |
| **Petco** | sheet (WN) a flat artifact that is thin relative to its length and width<br>Release (WN) the act of liberating someone or something<br>discus (WN) *an athletic competition in which a disk-shaped object is thrown as far as possible*<br>sponsorship (WN) the act of sponsoring (either officially or financially)<br>companion (WN) a friend who is frequently in the company of another<br>animals (WN) a living organism characterized by voluntary movement<br>pets (WN) a domesticated animal kept for companionship or amusement<br>dog (WN) a member of the genus Canis (probably descended from the common wolf) that has been domesticated by man since prehistoric times<br>group (WN) any number of entities (members) considered as a unit<br>owners (WN) a person who owns something<br>pet foods (WN) food prepared for animal pets |
| **Radioshack** | Realist (WN) *a philosopher who believes that universals are real and exist independently of anyone thinking of them*<br>Album (WN) a book of blank pages with pockets or envelopes; for organizing photographs or stamp collections, etc<br>Catalog (WN) a complete list of things; usually arranged systematically<br>computer store (WN) a store that sells computers to the small businessperson or personal user<br>hardware (WN) (computer science) the mechanical, magnetic, electronic, and electrical components making up a computer system<br>home (WN) housing that someone is living in<br>mobile phone (WK) ...a device that can make and receive telephone calls over a radio link whilst moving around a wide geographic area...<br>data (WN) an item of factual information derived from measurement or research<br>Application (WN) *the act of bringing something to bear; using it for a particular purpose*<br>Sponsor (WN) someone who supports or champions something<br>brands (WN) a name given to a product or service<br>Electronics (WN) the branch of physics that deals with the emission and effects of electrons and with the use of electronic devices<br>telephones (WN) electronic equipment that converts sound into electrical signals that can be transmitted over distances and then converts received signals back into sounds<br>Contacts (WN) *the physical coming together of two or more things* |

| POI | Perspective | Information Retrieval | Meaning Extraction |
|---|---|---|---|
| **Au Bon Pain** | About | http://www.aubonpain.com/aboutus/ | Soups, breads, plate, kiosks, Louis Kane,... |
| | Red Wiki | http://en.wikipedia.org/wiki/Carry-out | restaurant, Pizza delivery, |
| | | http://en.wikipedia.org/wiki/Bakeries | food, fast food, |
| | | http://en.wikipedia.org/wiki/Restaurants | table service,... |
| | Yellow Wiki | http://en.wikipedia.org/wiki/Au_Bon_Pain | Good, restaurant, locations, cafes, Thomas John,... |
| **Boston Athenaeum** | About | http://bostonathenaeum.org/ | Athen, associate, Meet-ups, Membership, gathering,... |
| | Red Wiki | http://en.wikipedia.org/wiki/Tourist_attractions | opportunities, dance, |
| | | http://en.wikipedia.org/wiki/Entertainers | Frankish, Library, |
| | | http://en.wikipedia.org/wiki/Museums | Museum,... |
| | | http://en.wikipedia.org/wiki/Libraries | |
| | Yellow Wiki | http://en.wikipedia.org/wiki/Boston_Athenaeum | Library, collections, landmark building, galleries, Edward Clarke Cabot,... |
| **Erbaluce** | About | *not found* | |
| | Red Wiki | http://en.wikipedia.org/wiki/Italian_restaurant | cuisines, restaurateur, Sein- |
| | | http://en.wikipedia.org/wiki/Restaurants | feld, Premises, delivery service,... |
| | Yellow Wiki | http://en.wikipedia.org/wiki/Erbaluce | Clarke Encyclopedia, table wines, Caluso, wine grape, sweet wines,... |
| **Petco** | About | http://about.petco.com/ | sheet, Release, discus, sponsorship, companion,... |
| | Red Wiki | **http://en.wikipedia.org/wiki/Animals** | animals, pets, dog, group, owners,... |
| | Red Wiki | **http://en.wikipedia.org/wiki/Pets** | |
| | Yellow Wiki | http://en.wikipedia.org/wiki/Petco | pet foods, Pets, Cesar Millan, San Diego, Halo Brand,... |
| **Radioshack** | About | *http://radioshack.com/* | Tandy, Realist, Album, Catalog, computer store,... |
| | Red Wiki | http://en.wikipedia.org/wiki/Computer_software | hardware, home, |
| | | **http://en.wikipedia.org/wiki/Home_electronics** | mobile phone, data, |
| | | http://en.wikipedia.org/wiki/Cellular_phones | Application,... |
| | Yellow Wiki | http://en.wikipedia.org/wiki/Radio_Shack | Sponsor, brands, Electronics, telephones, Contacts,... |

**Table 2.** The output of which module for the POIs given as example.

The automatic place classification of POIs employed allowed POIs to be mapped to linked data through the mapping NAICS-SUMO upper level ontology currently available in the Linked Open Data project. Table 4 presents the correspondent mapping to the POIs in study.

## 7. Conclusions and Future Work

In this chapter, we have briefly presented an approach for the *Semantic Enrichment of Places*, thereby developing a system, KUSCO, which builds a semantic index associated

| POI | NAICS | SUMO RDF triples |
|---|---|---|
| | 722110 | <rdfs:Class rdf:ID="FoodServicesAndDrinkingPlaces"> |
| **Au Bon** | Full- | <rdfs:subClassOf rdf:resource="#AccommodationAndFoodServices"/> |
| **Pain &** | Service | </rdfs:Class> |
| **Erbaluce** | Restaurants | <rdfs:Class rdf:ID="**FullServiceRestaurants**"> |
| | | <rdfs:subClassOf rdf:resource="#FoodServicesAndDrinkingPlaces"/> |
| | | </rdfs:Class> |
| | 519120, | <rdfs:Class rdf:ID="InformationServices"> |
| | Libraries & | <rdfs:subClassOf rdf:resource="#InformationServicesAndDataProcessingServices"/> |
| **Boston** | Archives | </rdfs:Class> |
| **Athenaeum** | | <rdfs:Class rdf:ID="**LibrariesAndArchives**"> |
| | | <rdfs:subClassOf rdf:resource="#InformationServices"/> |
| | | </rdfs:Class> |
| | 453910 | <rdfs:Class rdf:ID="OtherMiscellaneousStoreRetailers"> |
| | Pet & Pet | <rdfs:subClassOf rdf:resource="#MiscellaneousStoreRetailers"/> |
| **Petco** | Supplies | </rdfs:Class> |
| | Stores | <rdfs:Class rdf:ID="**PetAndPetSuppliesStores**"> |
| | | <rdfs:subClassOf rdf:resource="#OtherMiscellaneousStoreRetailers"/> |
| | | </rdfs:Class> |
| | 443112, | <rdfs:Class rdf:ID="ApplianceTelevisionAndOtherElectronicsStores"> |
| | Radio | <rdfs:subClassOf rdf:resource="#ElectronicsAndApplianceStores"/> |
| **Radioshack** | Tv & Other | </rdfs:Class> |
| | Electronics | <rdfs:Class rdf:ID="**RadioTelevisionAndOtherElectronicsStores**"> |
| | Stores | <rdfs:subClassOf rdf:resource="#ApplianceTelevisionAndOtherElectronicsStores"/> |
| | | </rdfs:Class> |

**Table 4.** POI classification in NAICS-SUMO.

with a given Point of Interest. For each POI, KUSCO finds related information on the Web and executes a sequence of Information Extraction and Natural Language Processing steps to automatically extract the relevant related terms. Each term is contextualized in lexical resources (WordNet and Wikipedia) which guide the extraction process by validating common-sense terms and which are also used to infer the meaning of each term. Once these terms are contextualized, they are called concepts. Their relevance is computed through an extended version of TF-IDF, which considers the semantics of each term. The main contribution in this work includes a clear and well defined methodology for gathering a massive amount of POIs and analyze a considerable proportion of these, creating semantic information about each POI from web pages.

The focus of this chapter was the contextualization of terms and the POI itself in the Linked Open Data cloud. The first mapping was done choosing the most used sense of a given term in knowledge resources to infer its meaning, while the latter was done using automatic classification through an instance based machine learning algorithm.

We have applied this methodology of semantic enrichment to another type of georeferenced entity: social events[39]. For this type of entity new online sources, this time for social events, were selected (specifically Zvents and Boston Calendar). These sources were harvested in order to extract and enrich 148303 events from August 25th 2009 to September 20th 2010 and hosted in 11197 different venues in Boston. This represented

a new perspective for us in extending our approach to a Generic Semantic Enrichment Model[40].

As a future step, with different perspectives of a given place being available, we think the use of this context can help the disambiguation task. This is true mainly for generic and ambiguous concepts that are related in similar perspectives for a given place (e.g. bank and account, each one present in different perspectives), which can benefit of the domain in question (e.g. Finance Services) and other unambiguous concepts found for the same place (e.g. tax, monetary fund). Another improvement is to take advantage of the meaning inferred for each concept in order to relate each of them through semantic relations from Yago, DBPedia and WordNet. Thus, it will be possible to locate the cluster of related concepts and instantiate other concepts that would be discarded without semantic knowledge.

## References


[1] Bizer, C.: The emerging web of linked data. IEEE Intelligent Systems **24** (2009) 87–92

[2] Maynard, D., Bontcheva, K.: Natural language processing. In Lehmann, J., Völker, J., eds.: Perspectives on Ontology Learning. Studies on the Semantic Web. AKA Heidelberg / IOS Press (2014)

[3] Miller, G., Beckwith, R., Fellbaum, C., Gross, D., Miller, K.: Introduction to wordnet: An on-line lexical database. Int J Lexicography **3**(4) (1990) 235–244

[4] Suchanek, F., Kasneci, G., Weikum, G.: YAGO - a large ontology from Wikipedia and WordNet. Elsevier Journal of Web Semantics **6**(3) (September 2008) 203–217

[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: Science, Services and Agents on the World Wide Web **7**(3) (2009) 154 – 165 The Web of Data.

[6] Suggested Upper Merged Ontology: http://www.ontologyportal.org/ Last visited: December, 2011.

[7] Hightower, J.: From position to place. In: Proc. of LOCA. (2003) 10–12 Ubicomp.

[8] Zhou, C., Frankowski, D., Ludford, P., Shekhar, S., Terveen, L.: Discovering personally meaningful places: An interactive clustering approach. Volume 25., New York, NY, USA, ACM (July 2007)

[9] Relph, E.C.: Place and placelessness / [by] E. Relph. Pion, London : (1976)

[10] Falko Schmid, C.K.: In-situ communication and labeling of places. In: 6th International Symposium on LBS & TeleCartography, Springer (9 2009)

[11] Lemmens, R., Deng, D.: Web 2.0 and semantic web: Clarifying the meaning of spatial features. In: Semantic Web meets Geopatial Applications. AGILE'08

[12] Rattenbury, T., Good, N., Naaman, M.: Towards automatic extraction of event and place semantics from flickr tags. In: SIGIR '07, New York, USA, ACM 103–110

[13] Dubinko, M., Kumar, R., Magnani, J., Novak, J., Raghavan, P., Tomkins, A.: Visualizing tags over time. In: WWW '06, New York, USA, ACM 193–202

[14] Jaffe, A., Naaman, M., Tassa, T., Davis, M.: Generating summaries and visualization for large collections of geo-referenced photographs. In: MIR '06. 89–98

[15] Ahern, S.N.M.N.R.Y.J.: World explorer: Visualizing aggregate data from unstructured text in geo-referenced collections. International Conference on Digital Libraries, Vancouver, BC, Canada (2007)

[16] Amitay, E., Har'El, N., Sivan, R., Soffer, A.: Web-a-where: geotagging web content. In: SIGIR '04

[17] Suchanek, F.: Information extraction for ontology learning. In Lehmann, J., Völker, J., eds.: Perspectives on Ontology Learning. Studies on the Semantic Web. AKA Heidelberg / IOS Press (2014)

[18] Asadi, S., Zhou, X., Yang, G.: Using local popularity of web resources for geo-ranking of search engine results. World Wide Web **12** (2009) 149–170

[19] Ahlers, D., Boll, S.: Location-based web search. In Scharl, A., Tochtermann, K., eds.: The Geospatial Web. Springer, London (2007)

[20] Tanasescu, V., Domingue, J.: A differential notion of place for local search. In: LOCWEB '08, New York, USA, ACM 9–16

[21] Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Scientific American **284** (2001) 34–43

[22] Hepp, M., Leenheer, P.D., de Moor, A., Sure, Y., eds.: Ontology Management, Semantic Web, Semantic Web Services, and Business Applications. Volume 7 of Semantic Web And Beyond Computing for Human Experience. Springer (2008)
[23] NAICS: North American Industry Classification System, Mexico's Instituto Nacional de Estadística e Geografía Informática (INEG) and Statistics Canada and the United States Office of Management *and* Budget (OMB) (2011) Last visited: December, 2011.
[24] Breitman, K.K., Casanova, M.A., Truszkowski, W.: Ontology sources. In: Semantic Web: Concepts, Technologies and Applications. NASA Monographs in Systems and Software Engineering. Springer London (2007) 175–199
[25] de Melo, G., Suchanek, F., Pease, A.: Integrating YAGO into the Suggested Upper Merged Ontology. In: Proc. of the ICTAI 2008), IEEE Computer Society, Los Alamitos, CA, USA (2008)
[26] Niles, I., Pease, A.: Towards a standard upper ontology. In: Proc. of the International Conference on Formal Ontology in Information Systems - Vol. 2001. FOIS '01, New York, NY, USA, ACM (2001) 2–9
[27] Rodrigues, F., Alves, A.O., Pereira, F.C., Jiang, S., Ferreira, J.: Automatic classification of Points-of-Interest for land-use analysis. In: Proceedings of Geoprocessing'2012. (2012)
[28] Alves, A.O., Pereira, F., Rodrigues, F., Oliveirinha, J.: Place in perspective: Extracting online information about points of interest. In: Proc. of AmI'10. Springer Berlin / Heidelberg (2010) 61–72
[29] Alves, A.O., Pereira, F.C., Biderman, A., Ratti, C.: Place enrichment by mining the web. In: Proc. of AmI'09, Berlin, Heidelberg, Springer-Verlag (2009) 66–77
[30] Cohen, W., Ravikumar, P., Fienberg, S.: A comparison of string distance metrics for name-matching tasks. In: IJCAI-03 Works. on Information Integration. 73–78
[31] Resnik, P.: Using information content to evaluate semantic similarity in a taxonomy. In: IJCAI. (1995)
[32] Mihalcea, R.: Semcor semantically tagged corpus. Technical report, University of North Texas (1998)
[33] Ramshaw, L., Marcus, M.: Text Chunking using Transformation-Based Learning. In: Proc. of WVLC-1995, Cambridge, USA
[34] Finkel, J.R., Grenager, T., Manning, C.: Incorporating non-local information into information extraction systems by gibbs sampling. In: ACL '05. 363–370
[35] Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval. Information Processing and Management **24**(5) (1988) 513–523
[36] Schutz, A., Buitelaar, P.: Relext: A tool for relation extraction from text in ontology extension. In: The Semantic Web - ISWC 2005, Proceedings. Volume 3729 of Lecture Notes in Computer Science., Galway, Ireland, Springer (2005) 593–606
[37] Navigli, R.: Word sense disambiguation: A survey. ACM Comput. Surv. **41**(2) (2009) 1–69
[38] Aha, D.W., Kibler, D., Albert, M.K.: Instance-based learning algorithms. Mach. Learn. **6** (Jan. 1991) 37–66
[39] Oliveirinha, J.a., Pereira, F., Alves, A.: Acquiring semantic context for events from online resources. In: Proceedings of the 3rd International Workshop on Location and the Web. LocWeb '10, New York, NY, USA, ACM (2010) 8:1–8:8
[40] Alves, A., Pereira, F.: Making sense of location context. In: Proceeding of the International Workshop on Context Discovery and Data Mining - The 18th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, New York, NY, USA, ACM (2012)

# Ontology Learning from Databases: Some Efficient Methods to Discover Semantic Patterns in Data

Farid CERBAH [a], Nadira LAMMARI [b]
[a] *Scientific Studies Department, Dassault Aviation*
[b] *CEDRIC, Cnam*

**Abstract.** Databases are often considered as the most reliable sources for knowledge extraction. Methods and tools have been proposed to build ontologies from databases, mainly by exploiting relational schemas. However, the resulting ontologies are often far below expectations, especially in terms of expressivity. A significant part of the domain-specific semantics needed to build more expressive ontologies can only be recovered from the stored data. This chapter describes some semantic patterns that are frequently encountered in databases and provides a systematic description of data-driven methods to exploit these patterns for ontology learning.

**Keywords.** Ontology Learning, Knowledge Extraction, Relational Databases, Data Mining, Natural Language Processing, Terminology Extraction

## Introduction

To construct the accurate semantic resources required by future knowledge-intensive applications, existing databases are undoubtedly among the most reliable sources to be exploited. However, finding ways to significantly ease the task of building highly expressive ontologies from such structured information sources is far from being a straightforward issue. Early methods exclusively based on the transformation of database schemas often result in incomplete ontologies that need to be further refined at the cost of huge manual post-editing efforts. Such manual tasks might be deemed too tedious and costly by many practitioners. To provide an extended automated support to facilitate the production of high quality ontologies from databases, adequate ontology learning methods should be elaborated.

When considering databases as information sources, ontology learning should be distinguished from the related issue of database-to-ontology mapping which has also motivated substantial work those past few years [2,6,11,4]. The main goal of the mapping technology is to provide declarative means in order to link relational models to pre-existing ontologies and to automatically generate instances from the data. Conversely, the basic issue in ontology learning from databases is to automatically build the ontology models, and more importantly to dig out the hidden semantics which is not directly available from the metadata.

Ontology learning from relational databases is not a new research issue. Several approaches and tools have been developed to deal with such structured input. Past contributions showed how ontologies can be learned and be fruitfully exploited to solve practical problems, such as ensuring integration and interoperation of heterogeneous databases. However, a major limitation of the existing methods is the derivation of ontologies with flat structure that simply mirror the schema of the source databases. Such results do not fully meet the expectations of users that are primarily attracted by the rich expressive power of semantic web formalisms and that could hardly be satisfied with target knowledge repositories that look likes their legacy relational databases. A natural expectation is to get at the end of the learning process ontologies that better capture the underlying conceptual structure of the stored data.
Ontologies with flat structure is the typical result of learning techniques that exclusively exploit information from the database schema without (or just marginally) considering the data. A careful analysis of existing databases shows that additional definition patterns can be learned from the data to significantly enrich the base structure. More particularly, class hierarchies can be induced from the data to refine classes derived from the relational schema.
We define in this chapter a comprehensive approach to ontology learning from relational databases that combines two complementary information sources: the schema definition and the stored data. The transformation of database schemas to ontologies has been extensively studied in previous work. Hence, we will mainly focus here on the exploitation of the stored data. Our primary contribution is a systematic description of three robust methods dedicated to identification of meaningful categorization patterns in the content of the databases. These methods have been implemented and integrated in the RDBToOnto platform[1].

The remainder of the chapter is organized as follows. We introduce in section 1 some motivating examples illustrating the panel of categorization patterns addressed in this work. Then, we give an overview of the typical learning process and of the classical schema-based transformation rules. The next sections are the core of this contribution and are dedicated to the hierarchy mining methods. We give an extensive description of the three formalized methods. Finally, last section offers some concluding remarks and directions for further research.

## 1. Diversity of Semantic Patterns

We give in this section an overview through selected examples of the structuring patterns that will be further explored in this chapter. We introduce here some interesting structuring patterns without paying to much attention to how they can be automatically identified and used to generate appropriate ontology fragments. Some of the most relevant content-driven transformation techniques will be addressed in next sections and we show how to exploit them in order to complement ontologies derived from database schemas.

[1] `http://www.sourceforge.net/projects/rdbtoonto`, see also [8]

**– Categorizing Attributes**

In many relational databases, some attributes are specifically included in tables to categorize the tuples. Figure 1 shows a typical example of such categorizing attributes. The table includes a set of access parts of an aircraft. A schema-based method will simply derive a class from this table and populate it by turning each tuple into a class instance. This straightforward transformation is effective in many situations. However, it is worth noting that additional structuring patterns can be exploited to semantically refine the target model. This example shows how the class derived from the table definition can be refined into subclasses derived from the values of the Type column. Sometimes, two *categorizing attributes* can be involved. For example, from a table of suppliers including the City and Country attributes, a Supplier class can be constructed and extended with a two-level hierarchy by interpreting the values of both columns (resulting in subclasses Sweden Supplier $\longrightarrow$ Stockholm Supplier, Göteborg Supplier, etc).

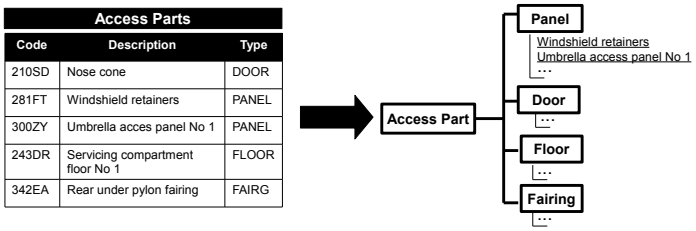



**Figure 1.** A categorization pattern where the categories to be exploited for hierarchy generation are defined in a specific column

To exploit these structuring patterns in ontology learning, two operations need to be performed: Identification of the categorizing attributes and exploitation of the attribute values to build hierarchies. A fully formalized method is presented in section 5.

**– Terminology-based Hierarchies**

Another content-based pattern somehow related with categorizing attributes is based on the semantic relationships that may hold between the textual designations of items. In the example given in figure 2, the source table includes a large list of tools to be used in aircraft maintenance. We can notice that by analyzing the terms in the description column, intermediate structuring classes can be introduced in the resulting ontology. Such patterns are very frequent in technical databases and they appear to be highly productive for building deeper taxonomies. The lexico-syntactic structure of technical terms can be exploited to include in the resulting ontologies some semantic relationships of the underlying specialized domain. Relevant *terminology-based patterns* can be mined from various data sources often encountered in databases, such as large product catalogs, structured descriptions of complex manufactured artifacts or biomedical repositories to name just a few.

We show in section 6 how such patterns relying on the lexical relationships between the terms can be reliably identified using terminology extraction techniques.

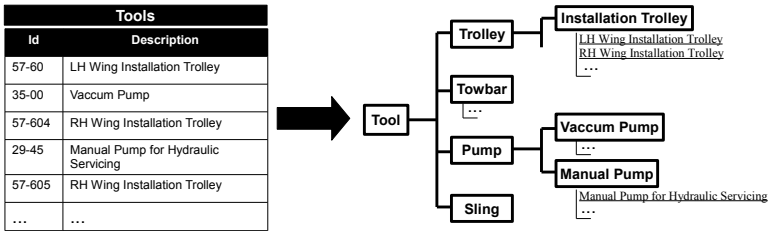



**Figure 2.** Extraction of a class hierarchy from a flat list of technical terms stored in a table column

**– Null Values**

Null or missing values can also reveal part of the hidden semantics. When all instances (i.e tuples) of a generic concept and its sub-concepts are gathered into a single table, some attributes may only be relevant for some subconcepts, and thus filled for instances of these sub-concepts but left empty for the others. For example, in an Employees relation that includes all employees of a flight company, attributes FlightHours and LicenceNumber would be filled with *null values* in entries corresponding to non-members of the flight staff. Partitioning of a relation on the basis of null values and their cooccurrences may reveal the underlying concept hierarchy. Figure 3 shows another example of subclass derivation from multiple null value attributes. In section 7, we define a method dedicated to the identification of these categorization patterns that are also to be discovered in data.

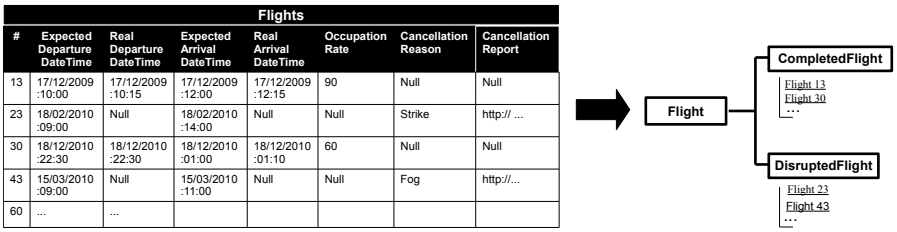



**Figure 3.** Derivation of subclasses from table columns with null values

## 2. The Overall Ontology Learning Process

The learning process is conceived as a combination of the most robust rules for exploiting relational schemas and data mining methods focused on concept hierarchy identification.

The transformation is defined as a composition of automated steps. The main steps of this process are: database normalization, class and property learning and ontology population.

In early approaches, data normalization is not included in the process. It is quite common to consider as input relational databases that are in some normal form, often 2NF or 3NF (e.g. [1]). It is assumed that the transformation process can be easily extended to cope with ill-formed input by incorporating at the early stages of the process

a normalization step based on existing algorithms. Though theoretically acceptable, this assumption has some drawbacks in practice, especially in a data mining perspective. Many legacy databases suffer from redundancy problems and substantial normalization efforts are often required to build up acceptable input for ontology construction. The normalization step aims at eliminating data redundancy through information sharing based on foreign key relations. This may result in an improved schema in which additional key-based constraints between the relationships are made explicit.

The core of the transformation process in ontology learning combines a classical database schema analysis with hierarchy mining in the stored data to identify semantically rich ontology models. The methods for class and property identification are discussed in the next sections.

The final step is ontology population that aims at generating instances of classes and properties from the database content. For a given class, an instance is derived from each tuple of the source relation. Moreover, if the refinement into subclasses has been successfully applied on the class, the instances need to be further dispatched into subclasses.

## 3. Preliminary Definitions

Before providing a detailed description of the methods, we start by introducing some basic notations and definitions.

A *relational database schema* $D$ is defined as a finite set of relation schemas $D = \{R_1, \ldots, R_n\}$ where each *relation schema* $R_i$ is characterized by its finite set of attributes $\{A_{i1}, \ldots, A_{im}\}$. A function $pkey$ associates to each relation its primary key which is a set of attributes $K \subseteq R$.

A relation $r$ on a relation schema $R$ (i.e. an *instance* of $R$) is a set of tuples which are sequences of $|R|$ values. Similarly, a database $d$ on $D$ is defined as a set of relations $d = \{r_1, \ldots, r_n\}$. By convention, if a relation schema is represented by a capital letter, the corresponding lower-case letter denotes an instance of the relation schema.

A *projection* of a tuple $t$ on a set of attributes $X \subseteq R$, denoted $t[X]$, is a restriction on $t$, resulting in the subsequence with values corresponding to attributes of $X$. The projection of a relation $r$ on $X$, denoted $\pi_X(r)$, is defined by $\pi_X(r) = \{t[X] \mid t \in r\}$.

The concept of *inclusion dependency* (e.g. [1]) is used to account for correlations between relations. An inclusion dependency is an expression $R[X] \subseteq S[Y]$ where $X$ and $Y$ are respectively attribute sequences of $R$ and $S$ relation schemas, with the restriction $|X| = |Y|$. The dependency holds between two instances $r$ and $s$ of the relation schemas if for each tuple $u$ in $r$ there is a tuple $v$ in $s$ such that $u[X] = v[Y]$. Informally, an inclusion dependency is a convenient way to state that data items are just copied from another relation.

*Foreign key relationships* can be defined as inclusion dependencies satisfying the additional property: $Y = pkey(S)$. The notation $R[X] \subseteq S[pkey(S)]$ will be used for these specific dependencies.

Formal descriptions of ontology fragments are expressed in OWL abstract syntax.

**Relation to Class**

| Source | Preconditions | Target |
|---|---|---|
| $R \in D$ | $\neg \exists\, C \mid C = targetClassOf(R)$ | class($\boldsymbol{C_R}$) |

**Foreign key Relationship to Functional Object Property**

| Source | Preconditions | Target |
|---|---|---|
| $R_0[A] \subseteq R_1[pkey(R_1)]$ | $C_0 = targetClassOf(R_0)$<br>$C_1 = targetClassOf(R_1)$ | ObjectProperty($\boldsymbol{P_A}$<br>domain($C_0$)<br>range($C_1$)<br>Functional) |

**Composite Key Relation to Object Property**

| Source | Preconditions | Target |
|---|---|---|
| $R_0 \in D$<br>$\lvert R_0 \rvert = 2$<br>$pkey(R_0) = \{K_1, K_2\}$<br>$R_0[K_1] \subseteq R_1[pkey(R_1)]$<br>$R_0[K_2] \subseteq R_2[pkey(R_2)]$ | $C_1 = targetClassOf(R_1)$<br>$C_2 = targetClassOf(R_2)$ | ObjectProperty($\boldsymbol{P_R}$<br>domain($C_1$)<br>range($C_2$)) |

**Table 1.** Three reliable rules that match patterns in the schema. In the **Target** part, the variable in bold holds the Uri of the generated ontology fragment. *targetClassOf* assertions provide traceability to control the process.

## 4. Schema-based Transformations

Ontology learning can benefit from early work in the domain of database reverse engineering where the primary concern was to extract object-oriented models from relational models [5,22,23] and most approaches in this domain are based on an analysis of the relational schemas. The core of the schema-based transformation rules for reverse engineering are still relevant in the context of ontology learning. The most reliable rules have been reused as a starting point and extended in several approaches that have ontologies as target models [26,17].

The transformation can be based on prioritized rules that define typical mappings between schema patterns and ontology elements, namely classes, datatype and object properties. We give in table 1 three of the most reliable rules which are also employed in several existing approaches. The first trivial rule states that every relation can potentially be translated as a class though relations can be consumed by more specific rules with higher priority, such as the third rule. The second rule is also a simple mapping from a foreign key relationship to a functional object property. The third rule is intended to match a relation with a composite primary key and two key-based attributes. Such bridging relations are only introduced in the database to link two other relations through key associations. They are turned into many-to-many object properties. An extensive set of formalized rules is proposed in [25].

As mentioned in section 2, a preliminary normalization task is often required to improve the schema before starting the ontology learning process. More particularly, data duplication between relations is a recurring problem that might have a negative impact on the resulting ontologies. Such data duplications can be formalized as inclusion dependencies where the set of attributes from the right-hand relation are not restricted to the primary key (i.e inclusion dependencies that are not foreign key relationships). To

eliminate these duplications, the database need to be transformed by turning all these inclusion dependencies into foreign key relationships. More formally, each attested dependency $R[X] \subseteq S[Y]$ with $Y \neq pkey(S)$ is replaced with the foreign key relationship $R[A] \subseteq S[pkey(S)]$ where $A$ is a newly introduced foreign key attribute, and all non key attributes in $X$ together with corresponding data in $r$ are deleted from the relation.

Content of the relations is an additional information source allowing to refine with subclasses some of the classes obtained by applying schema-based mapping rules. The exploitation of this second source is central in the data-driven approach we are advocating.

## 5. Categorizing Attributes

In section 1, we introduced through examples the issue of hierarchy mining from database content, showing how classes derived from the schema can be refined with subclasses extracted from the stored data. The motivating example given in figure 1 is an illustration of some modelling patterns attested in many databases where specific attributes are used to assign categories to tuples. These frequently-used patterns are highly useful for hierarchy mining as values of categorizing attributes can be exploited to derive subclasses. One of the methods we propose for hierarchy mining is focused on exploiting the patterns based on such categorizing attributes.

We describe below the pattern identification procedure. Then, we discuss the generation of the subclasses from the identified patterns.

Two sources are involved in the identification of categorizing attributes: the names of attributes and the redundancy in attribute extensions (i.e. in column data). These two sources are indicators that allow to find attribute candidates and select the most plausible one.

### *5.1. Identification of lexical clues in attribute names*

Categorizing attributes are often lexically marked. When used for the purpose of categorization, the attributes may bear names that reveal their specific role in the relation (i.e. classifying the tuples). The categorizing attribute can be clearly identified by its name (e.g. Category, type). The lexical clue that indicates the role of the attribute can just be a part of a compound noun or of an abbreviated form, as in the attribute names CategoryId or CatId. Our candidate filtering method relies on a simple segmentation procedure that aims at identifying clues from a predefined list of frequently used lexical items.

The efficiency of this filtering step heavily depends on this predefined list of lexical clues. More specifically, finding the right balance between precision and recall amounts to find the proportion of word stems (*vs* full clue words) to be included in this clue list. Because of the large proportion of attributes with abbreviated names, good recall cannot be obtained without exploiting clue stems. For example, the stem cat is required to identify the categorizing attributes Catname, CatId and SubcatItemId, all encountered in our test set. The counterpart is that such short clues have a negative impact on the precision. Our extensive experiments suggest to only include the stems of the most frequently used

clue words. For instance, the clue word Family, which is identified as a relevant but not very frequent indicator, is included in the list without its stem (fam).

With an extensive list of lexical clues, the filtering step based on lexical clues can be effective. However, experiments on complex databases showed that this step often ends up with several candidates. Furthermore, attributes that can play a categorization role are not necessarily defined with lexically marked names. We gave in section 1 an example where attributes named Country and City can be seen in some application contexts as good categorization sources even though no lexical clues can be found in the attribute names. These facts motivate the need for complementary ways to characterize the potentially relevant categorizing attributes. Additional filtering mechanisms can help to make a decision even when no lexical clues can be found or to choose between lexically pre-filtered attributes. Information diversity in the attribute extension appears to be a good complementary source in this selection process.

**- Filtering though entropy-based estimation of data diversity**

We make the assumption that a good candidate for tuple categorization might exhibit some typical degree of diversity that can be formally characterized using the concept of entropy from information theory. Entropy is a measure of the uncertainty of a data source. In our context, attributes with highly repetitive values will be characterized by a low entropy. Conversely, among attributes of a given relation, the primary key attribute will have the highest entropy since all values in its extension are distinct.

Informally, the rationale behind this selection step is to favor the candidate that would provide *the most balanced distribution of instances within the subclasses*. We give in what follows a formal definition of this step.

If $A$ is an attribute of a relation schema $R$ instantiated with relation $r$, the diversity in $A$ is estimated by:

$$H(A) = -\sum_{v \in \pi_A(r)} P_A(v)\ .\ \log P_A(v) \tag{1}$$

$$P_A(v) = \frac{|\sigma_{A=v}(r)|}{|r|} \tag{2}$$

- $\pi_A(r)$ is the projection of $r$ on $A$ defined as $\pi_A(r) = \{t[A] \mid t \in r\}$. This set is the *active domain* of $A$. In other words, $\pi_A(r)$ is the set of values attested in the extension of $A$. Each value $v$ of the set $\pi_A(r)$ is a potential category (to be mapped to a subclass in the ontology).
- $\sigma_{A=v}(r)$ is a selection on $r$ defined as $\sigma_{A=v}(r) = \{t \in r \mid t[A] = v\}$. This selection extracts from the relation $r$ the subset of tuples with $A$ attribute equal to $v$. In this specific context, the selection extracts from the relation all entries with (potential) category $v$.
- $P_A(v)$ is the probability of having a tuple with $A$ attribute equal to $v$. This parameter accounts for the weight of $v$ in $A$. It can be estimated by the relative frequency of $v$ (i.e. maximum-likelihood estimation).

Let now $C \in R$ denote the subset of preselected attributes using lexical clues. A first pruning operation is applied to rule out candidates with entropy at marginal values:

```
Method: catAttBasedHierarchy(R, A)
– R is a relationship (and r is its corresponding instance in the database)
– A is a categorizing attribute from R identified through the procedure defined in section 5.1
C_R = targetClassOf(R)
A' = getExternalFullCatList(R, A)
if A' = NULL then A' = A

forall the v ∈ π_A'(r) do
    add class(C_v partial C_R)
end

Method: getExternalFullCatList(R, A)
if R[A] ⊆ S[pkey(S)]  and
   pkey(S) = {B_0}    and
   S = {B_0, B_1}     and
   |π_B0(r)| = |π_B1(r)| then
    return B_1
else
    return NULL
end
```

**Algorithm 1:** A method for hierarchy generation from a categorizing attribute

$$C' = \{\ A \in C \mid H(A) \in [\ \alpha,\ H_{max}(R)\ .\ (1-\beta)\ ]\ \} \tag{3}$$

- $\alpha$ and $\beta$ are parameters such that $\alpha, \beta \in [0, 1]$.
- $H_{max}(R)$ is the highest entropy found among attributes of the relations ($H_{max}(R) = \max_{A \in R} H(A)$). As said earlier, $H_{max}(R)$ is often the entropy of the primary key attribute.

If, after this pruning step, several candidates still remain[2], we ultimately select the attribute that would provide the most balanced organization of the instances. This amounts to look for the attribute whose entropy is the closest to the maximum entropy for the number of potential categories involved. This maximum entropy is given by :

$$\widetilde{H}_{max}(A) = \log |\pi_A(r)| \tag{4}$$

This reference value, derived from the entropy expression (1), is representative of a *perfectly* balanced structure of $|\pi_A(r)|$ categories with the same number of tuples in each category. Note that this value is independent of the total number of tuples ($|r|$).

The final decision aims at selecting the attribute $A^*$ whose entropy is the closest to this reference value:

$$A^* = arg \min_{A \in C'} \frac{|H(A) - \widetilde{H}_{max}(A)|}{\widetilde{H}_{max}(A)} \tag{5}$$

### *5.2. Generation and population of the subclasses*

As shown with algorithm 1, the generation of subclasses from an identified categorizing attribute can be straightforward. A subclass is derived from each value type of the at-

[2]Note that all candidates can be eliminated. In this case, the first candidate is arbitrarily chosen.

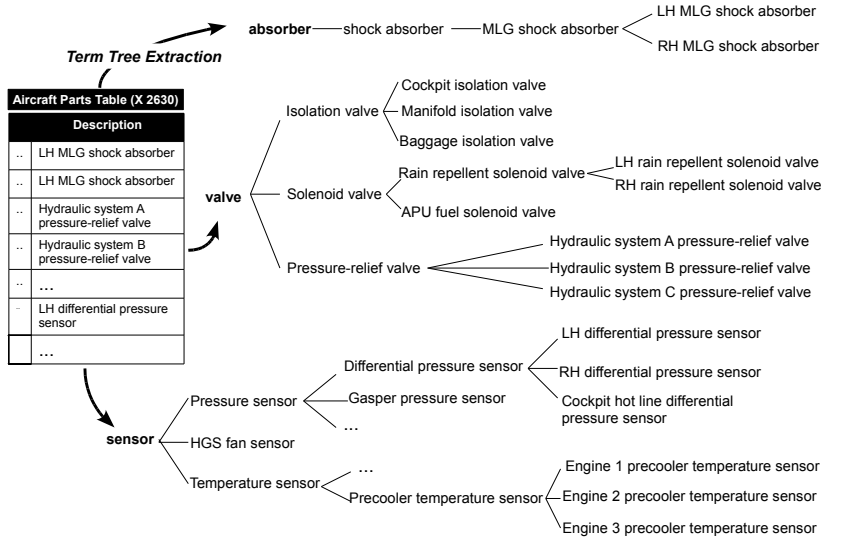



**Figure 4.** The terminology analysis process applied to a table column that includes an extended list of 2630 technical terms referring to aircraft components. The figure gives a partial view of three extracted *term trees*, respectively rooted at the absorber,valve and sensor single word terms.

tribute extension (i.e. for each element of the attribute active domain). However, proper handling of the categorization source may require more complex mappings (*getExternalFullCatList* method). This part of the transformation method accounts for the fact that values to be used for subclass generation can be issued from another relation. In the underlying pattern, the categorizing attribute only includes identifiers to refer through a foreign key relationship to a relation in which all allowed categories are defined.

Classes of the resulting hierarchy are populated by exploiting the tuples from the same source relation. An instance is generated from each tuple. The extra task of dispatching the instances into subclasses is based on a partitioning of the tuples according to values of the categorizing attribute. Formally, for each value $v$ of $A^*$, the corresponding class is populated with the instances derived from the tuples of the set $\sigma_{A^*=v}(r) = \{t \in r \mid t[A] = v\}$.

## 6. Terminology-based Patterns

Those past few years, a lot of attention has been devoted to ontology learning from text, resulting in a wide range of techniques for generating knowledge structures that are often considered as relevant intermediate productions in the process of constructing formal ontologies (cf. [16] in this volume, [3]). However, few methods have been focused on exploiting the textual data stored in heterogeneous databases.

The most commonly extracted knowledge structures extracted from text corpora in ontology learning are domain-specific vocabularies (terminologies) defined as hierarchies of technical terms. The simple method we define in this section aims at mining such terminological resources in databases to build ontologies with deeper taxonomies. As discussed in section 1, many complex technical databases to be exploited for ontol-

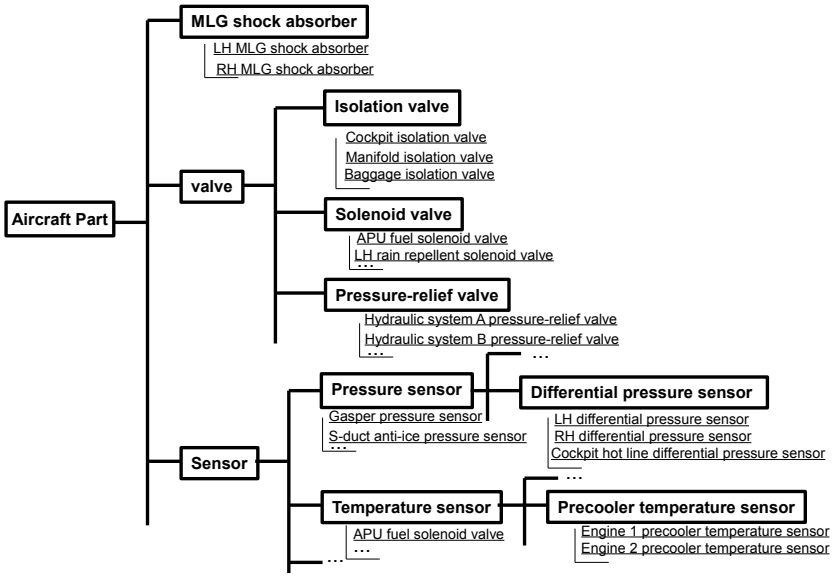



**Figure 5.** Class hierarchy derived from the terminological resources described in figure 4. This structure is obtained with the threshold vector $\delta = [1\ 2\ 3]$

ogy learning may incorporate large poorly structured lists of items (e.g. product catalogs, components of complex manufactured artifacts, biomedical data, etc) from which valuable semantic patterns can be extracted. The method relies on two key assumptions:

- In many situations, the underlying structure of the flat lists of technical terms included in table columns can be recovered through Natural Language Processing (see [19] in this volume for details on commonly exploited NLP techniques). In the *term trees* resulting from the linguistic analysis, the leaves are labeled with the column values (i.e. the full terms) and the intermediate nodes with more generic terms obtained through lexical decomposition of the full terms (see the example of figure 4 further discussed below).
- In the ontology generation process, one the key decisions to be made is the determination of the items from this lexical structure to be turned into classes in order to provide a suitable hierarchical organization of instances in the resulting ontology. Depending on the process settings, some nodes (either intermediate or terminal) will be selected from the term tree to build a class hierarchy while instances are to be derived from some of the leaves (which actually correspond to the source column values). The figure 5 shows a part of the ontology derived from the term trees of figure 4.

### *6.1. Extraction of the term trees*

To build the term trees from the flat input term lists extracted from the table columns, NLP techniques for terminology extraction can be exploited [19,14,7]. However, it is worth noting that the problem addressed here differs on some aspects from the classical terminology extraction problem. In this specific context, a significant set of the terms is

already given and the goal is to complement this set with more generic terms that would enable the construction of meaningful term hierarchies. Conversely, the basic goal of terminology extraction is to extract terms from corpora without necessarily presupposing any initial set of terms.

The main task performed in order to build the term trees is the lexical decomposition of the input terms by applying lexico-syntactic patterns commonly used in term extraction methods. This task relies on the result of a preliminary parsing step where part-of-speech tags are assigned to the components of the terms. For example, the term LH differential pressure sensor will be assigned the tag sequence Abbr/Adj/Noun/Noun. From this sequence, the decomposition task extracts the generic terms corresponding to the eligible tag subsequences Noun, Noun/Noun and Adj/Noun/Noun resulting in the hyperonym chain composed of the terms sensor, pressure sensor and differential pressure sensor.

We have experimented this process on significant datasets using the extraction methods implemented in the *HyperTerm* platform which is dedicated to terminology acquisition [9]. Moreover, a terminology analyzer specifically developed for this purpose is made available with RDBToOnto. This analyzer which is based on the GATE platform [19] takes as input a flat term list (which might correspond to a table column extension) and returns a set of term trees encoded in SKOS [21]. These term trees can be interpreted by RDBToOnto to build part of the ontology. We illustrate in figure 4 the terminology analysis process applied to a table column that includes an extended list of 2630 technical terms referring to aircraft components. The output of this process is a set of 150 term trees resulting from the lexical decomposition of the stored terms. The figure gives a partial view of three extracted term trees, respectively rooted at the absorber,valve and sensor single word terms. The goal is to automatically extract generic terms in order to encompass the input column items in a hierarchical structuring scheme. Consequently, the extraction procedure only retains term trees whose leaves are labeled with the column items (see example in figure 4).

### *6.2. From term trees to class hierarchies*

All the terms generated in the previous step are not necessarily turned into classes. Selecting the terms that will still appear at the ontology level is a difficult issue that may require a semantic interpretation of some kind and need, in most cases, including in the process an additional step for manual tuning. Our ambition here is limited to the definition of a simple generation method based on structural considerations that allows the user to shape the resulting class hierarchy by iteratively adjusting some global parameters. Informally, the method relies on two principles:

- Leaves of the trees (i.e the input column items) can either be converted as classes or instances. The choice between these two global options is application-dependent.Typically, the “all as classes” option can be more suitable when dealing with thesaurus-like input whereas the “leaves as instances” option would be preferred when transitioning a structural descriptions of a complex product (e.g. an aircraft or car system breakdown from a product management system where basic units of the decomposition refer to the concrete components of the vehicle).
- The potential of an intermediate term to be turned into a class is based on both its depth in the tree and the number of leaves of the subtree rooted at this term (i.e. the number of instances that would be gathered by the resulting class). Roughly

1 **Method:** $termBasedHierarchy(R, A, \mathcal{T})$
2 – $R$ is a relationship
3 – $A$ is the attribute from $R$ relying on a terminology-based pattern
4 – $\mathcal{T}$ is the set of term trees extracted from $\pi_A(r)$
5 – $\delta$ is a threshold vector providing for each tree depth the number of leaves that need to be gathered by a term to be considered as a good class candidate
6 $C_R = targetClassOf(R)$
7 **forall the** $T \in \mathcal{T}$ **do**
8 | $\{compactTree(T)\}$ `// Optional removal of all single child nodes`
9 | $t_0 = root(T)$
10 | $deriveClasses(t_0, C_R, T)$
11 **end**

12 **Method:** $deriveClasses(t, C_p, T)$
13 **if** $|leaves(t)| \geq \delta[depth(t)]$ **then**
14 | *add* class($\boldsymbol{C_t}$ partial $C_p$) `// New class` $\boldsymbol{C_t}$ `is derived from` $t$
15 | **forall the** $t' \in children(t)$ **do**
16 | | $deriveClasses(t', C_t, T)$
17 | **end**
18 **end**

**Algorithm 2:** Derivation of a class hierarchy from term trees

speaking, the terms closer to the roots are more likely to be relevant candidates whereas specific terms closer to the leaves could only be selected if they encompass more terminal nodes.

This approach is formally described in algorithm 2. In this algorithm, the class derivation process is iteratively applied to each element of the term tree set $\mathcal{T}$(in *termBasedHierarchy* method). The *deriveClasses* method performs a recursive depth-first exploration of a given term tree. A class $C_t$ is derived from a term $t$ (line 14) if the number of leaves under this term is equal or above the threshold defined in the $\delta$ vector for the corresponding depth in the tree (line 13). The class hierarchy depicted in figure 5 is obtained with the threshold vector $\delta = [1\ 2\ 3]$. In other words, only one leaf is required for the root term to be considered as a relevant class candidate while two leaves are required for terms of depth 2 which are directly linked to the root. Three leaves are required for terms of depth 3, and no classes can be derived from the deeper terms. In the implementation of this lightweight approach included in RDBToOnto, the process can be further constrained through user-defined term exception lists to enforce the exclusion or inclusion of some terms in the resulting ontology.

## 7. Null Values

Another categorization pattern widely adopted in relational databases can be identified through a careful analysis of missing information. As illustrated in section 1, attributes with null or default values may reveal an implicit semantic partitioning of the potentially similar items gathered in a single table. Some of the attributes may appear to be only applicable for a subset of the items leading to the introduction of null values. For example, a MaidenName attribute in a Person relation would be set to null in a significant

1 **Method:** $nullValueBasedHierarchy(R, X)$
2 – $R$ is a relationship (and $r$ its instance in the database)
3 – $X$ is the set of null value attributes in $R$
4 $C_R = targetClassOf(R)$
5 **for** $i = 1$ **to** $|X|$ **do** $c_i = \{x_i\}$
6
7 $C = \{c_1, \ldots, c_{|X|}\}$
8 **while** $\exists (c_i, c_j) \mid dom_{c_i}(r) = dom_{c_j}(r)$ **do** // Group compatible attributes into clusters
9 $C = C \setminus \{c_i, c_j\} \cup \{c_i \cup c_j\}$
10 **end**
11 **forall the** $c \in C$ **do**
12 *add* class($\boldsymbol{C_c}$ partial $C_R$) // class $\boldsymbol{C_c}$ is derived from attribute cluster $c$
13 **forall the** $a \in c$ **do** // A datatype property is derived from each attribute of $c$
14 *add* DatatypeProperty($\boldsymbol{P_a}$ domain($C_c$) range($datatype(a)$))
15 **end**
16 **end**

**Algorithm 3:** Derivation of a class hierarchy from null value attributes

proportion of the tuples since the applicability domain of the attribute is limited to tuples representing women.

We define here a data-driven method allowing the generation of class hierarchies from null values attributes. We start with additional definitions to provide a formal account of null value semantics and of the related method.

The *applicability domain* of an attribute $A$ in a relation instance $r$, denoted by $dom_A(r)$, is the set of tuples where this attribute is applicable (i.e. the set of tuples with a non-null value assigned to this attribute). We also define the applicability domain of a set of attributes $X$, denoted by $dom_X(r)$, the set of tuples where attributes of $X$ are jointly applicable.

A *mutual existence dependency* between two attribute sets $X$ and $Y$ holds if $dom_X(r) = dom_Y(r)$. Conversely, an *exclusive existence dependency* between two attribute sets $X$ and $Y$ holds if $dom_X(r) \cap dom_Y(r) = \emptyset$.

The basic idea behind this method (cf. algorithm 3) is to divide an initial set of relevant null value attributes into clusters of attributes linked through mutual existence dependencies (line 5 to 8) . Then, a subclass with specific attributes is defined from each of the resulting clusters of null value attributes. The process can be outlined on the example of figure 3. The initial set of all null value attributes is {Real Departure DateTime, Real arrival DateTime, Occupation Rate, Cancellation Reason, Cancellation Report}. Partitioning of this input set produces the two clusters {Real Departure DateTime, Real arrival DateTime, Occupation Rate} and {Cancellation Reason, Cancellation Report} which respectively correspond to the subclasses CompletedFlight and DisruptedFlight.

To ensure the consistency of the resulting clustering and estimate the plausibility of the input null value attributes, an additional step could be included to check that exclusive mutual dependencies actually hold between each couple of the attribute clusters.

It should be noted that this formalized process starts with an identified set of relevant null value attributes. For sake of robustness, the identification of these attributes is not included in the process. A major difficulty of this selection task is the need to consider the status or lifecycle of the database to properly interpret the semantics of the null values. Data sparseness can be interpreted as either inapplicability or temporary absence of the

values. However, this method can be extended to provide some support for this step, more particularly through a more precise analysis of the existence dependencies[3].

## 8. Conclusion and Further Work

In this chapter, we have discussed ontology learning for relational databases with a focus on methods for the identification of semantic patterns in the stored data. We strongly support the idea that expressive ontologies can rarely be generated by exclusively exploring the database schema without looking at the data. The data-driven methods we proposed have been experimented on significant datasets and demonstrated very good performance (more details on evaluation are provided in [10] and [15]).

One of the main extensions to be addressed is the combination of these methods. It is highly instructive to revisit the examples discussed in section 1 and see that in most of them, different types of categorization patterns can be found. In a suitable combination of these methods, null value attributes and categorizing attributes could be the main sources for classes in upper levels of the target ontologies whereas terminology-based patterns could be exploited to further refine the class hierarchies.

The extraction of terminology-based patterns from databases for the purpose of ontology learning is a first step in the exploitation of the unstructured content which is prominent in many databases. Incorporating information extraction techniques to discover complex axioms from these heterogeneous sources is a major challenge for future ontology learning systems (cf. [27,13] in this volume). When analyzing unstructured text from database records, the use of the related categorical attributes as background knowledge may contribute to improve the performance of information extraction [18,20].

Moreover, ontology learning in this context should not be conceived as an endogenous process that only aims at discovering domain semantics from the data and metadata of some source legacy databases. Interoperability of the resulting ontologies can significantly be increased by mapping the application-specific concepts extracted from these databases to equivalent or closely related concepts from widely shared reference ontologies[4]. Solutions for identifying such semantic bridges between ontologies are being investigated in the related area of ontology matching [12,24].

## References

[1] S. Abiteboul, R. Hull, and V. Vianu, editors. *Foundations of databases*. John Benjamins, 1995.

[2] Sören Auer, Sebastian Dietzold, Jens Lehmann, Sebastian Hellmann, and David Aumueller. Triplify: light-weight linked data publication from relational databases. In *Proc. of the 18th International Conference on World Wide Web, WWW 2009, Madrid, Spain*. ACM, 2009.

[3] N. Aussenac-Gilles, S. Despres, and S. Szulman. The TERMINAE method and platform for ontology engineering from texts. In P. Buitelaar and P. Cimiano, editors, *Bridging the Gap between Text and Knowledge - Selected Contributions to Ontology Learning and Population from Text*. IOS Press, 2008.

[3] Cf. Lammari et al. [15] for further details on this type of attribute dependencies and on how null value patterns can be combined with various information sources (schemas, queries, ...).

[4] RDBToOnto allows the reuse of external reference ontologies. However, the provided support is limited to the processing of mappings defined manually. For example, the generation process can be manually constrained so that a *firstname* column will be mapped to *foaf:firstName* property from the FOAF ontology.

[4] J. Barrasa, O. Corcho, and A. Gómez-Pérez. R2O, an extensible and semantically based database-to-ontology mapping language. In *Second Workshop on Semantic Web and Databases (SWDB2004)*, Toronto, Canada, 2004.

[5] Andreas Behm, Andreas Geppert, and Klaus R. Dittrich. On the migration of relational schemas and data to object-oriented database systems. In *Proc. 5th International Conference on Re-Technologies for Information Systems*, Klagenfurt, Austria, 1997. Oesterreichische Computer Gesellschaft.

[6] C. Bizer. D2R MAP - a database to rdf mapping language. In *Proceedings of WWW03*, Budapest, 2003.

[7] D. Bourigault, C. Jacquemin, and M.-C. L'Homme, editors. *Recent Advances in Computational Terminology*. John Benjamins, 2001.

[8] F. Cerbah. Learning highly structured semantic repositories from relational databases – The RDBToOnto tool. In *The Semantic Web: Research and Applications – Proceedings of the 5th European Semantic Web Conference (ESWC 2008)*. Springer, 2008.

[9] F. Cerbah and B. Daille. A Service Oriented Architecture for Adaptable Terminology Acquisition. In *NLDB 2007*. Springer, 2007.

[10] Farid Cerbah. Learning ontologies with deep class hierarchies by mining the content of relational databases. In F. Guillet, G. Ritschard, D. A. Zighed, and H. Briand, editors, *Advances in Knowledge Discovery and Management*, volume 292 of *Studies in Computational Intelligence*. Springer, 2010.

[11] Cristian Pérez de Laborda and Stefan Conrad. Relational.OWL: a data and schema representation format based on owl. In *Proc. of the 2nd Asia-Pacific conference on Conceptual modelling*, Australia, 2005.

[12] Jérôme Euzenat and Pavel Shvaiko. *Ontology Matching*. Springer-Verlag Berlin Heidelberg, 2007.

[13] Sebastian Hellmann, Jens Lehmann, and Johanna Völker. Learning owl class expressions from wikipedia. In Jens Lehmann and Johanna Völker, editors, *Perspectives on Ontology Learning*, Studies on the Semantic Web. AKA Heidelberg / IOS Press, 2014.

[14] K. Kageura, B. Daille, H. Nakagawa, and L.F. Chien. Recent trends in computational terminology. *Terminology*, 10(2):1–25, 2004.

[15] N. Lammari, I. Comyn-Wattiau, and J. Akoka. Extracting generalization hierarchies from relational databases. a reverse engineering approach. *Data and Knowledge Engineering*, 63:568–589, 2007.

[16] Jens Lehmann and Johanna Völker. Introduction. In Jens Lehmann and Johanna Völker, editors, *Perspectives on Ontology Learning*, Studies on the Semantic Web. AKA Heidelberg / IOS Press, 2014.

[17] M. Li, X. Du, and S. Wang. Learning ontology from relational database. In *Proc. of International Conference on Machine Learning and Cybernetics*, volume 6, pages 3410 – 3415. IEEE, 2005.

[18] I. R. Mansuri and S. Sarawagi. Integrating unstructured data into relational databases. In *Proceedings of the 22nd International Conference on Data Engineering (ICDE'06)*, 2006.

[19] Diana Maynard and Kalina Bontcheva. Natural language processing. In Jens Lehmann and Johanna Völker, editors, *Perspectives on Ontology Learning*, Studies on the Semantic Web. AKA Heidelberg / IOS Press, 2014.

[20] M. Michelson and C. A. Knoblock. Semantic annotation of unstructured and ungrammatical text. In *In IJCAI*, 2005.

[21] A. Miles and D. Brickley. Skos core guide – w3c working draft. Technical report, W3C, 2005.

[22] W.J. Premerlani and M.R. Blaha. An approach for reverse engineering of relational databases. *Communications of the ACM*, 37(5), 1994.

[23] S. Ramanathan and J. Hodges. Extraction of object-oriented structures from existing relational databases. *ACM SIGMOD*, 26(1), 1997.

[24] Marta Sabou, Mathieu d'Aquin, and Enrico Motta. Using the semantic web as background knowledge for ontology mapping. In *Proc. of the Int. Workshop on Ontology Matching (OM-2006)*, 2006.

[25] J. F. Sequeda, M. Arenas, and D. P. Miranker. On directly mapping relational databases to RDF and OWL. In *Proc. of the 18th Int. Conference on World Wide Web, WWW 2012, Lyon, France*. ACM, 2012.

[26] L. Stojanovic, N. Stojanovic, and R. Volz. Migrating data-intensive web sites into the semantic web. In *Proceedings of the ACM Symposium on Applied Computing (SAC 02)*, Madrid, 2002.

[27] Fabian Suchanek. Information extraction for ontology learning. In Jens Lehmann and Johanna Völker, editors, *Perspectives on Ontology Learning*, Studies on the Semantic Web. AKA Heidelberg / IOS Press, 2014.

# Part V

# Methodology and User Interaction

# Learning Ontologies via Games with a Purpose

Elena SIMPERL [a], Stephan WÖLGER [b], Stefan THALER [b], Katharina SIORPAES [b]
[a] *University of Southampton, United Kingdom*
[b] *STI Innsbruck, University of Innsbruck*

**Abstract.** In this chapter, we analyse the process of ontology learning, and we give an overview of typical activities and tasks that rely on human contributions. We present a survey of existing games concerned with the learning of ontologies and other related knowledge structures, and compare and discuss their features against the results of the process-oriented analysis. By describing in detail the Generic Gaming Toolkit delivered by the European research project INSEMTIVES, which facilitates the development of games for semantic content authoring, as well as a number of guidelines and best practices derived from our own experiences in the context of the OntoGame framework, we hope to provide useful information to game designers. We conclude the paper with a short summary and an outline of potential future lines of research in the area.

**Keywords.** ontologies, games with a purpose, GWAP, incentives, user interaction, crowdsourcing, human computation

## 1. Introduction

Many aspects of semantic content authoring are demonstrably reliant on human computation [24,20]. This includes knowledge modeling and ontology development, but also semantic annotation - the description of digital artifacts such as Web pages, images, audio and video content using ontological concepts - and, most recently, data publication and interlinking according to Linked Data principles [25] (cf. also Vrandecic and Jentzsch [35] in this volume). In all of these areas, human abilities is indispensable for the resolution of those particular tasks that are acknowledged to be hardly approachable in a systematic, engineering-driven fashion; and also, though to a lesser extent, to the wide array of methods and techniques that have been proposed as an attempt to perform others automatically. In this second category, despite constant progress in improving the performance of the corresponding algorithms and the quality of their results, experiences show that human assistance is nevertheless required, even if it just for the validation of algorithm outputs. Ontology learning falls in the same category; as we will discuss in Section 2, putting aside the configuration of the overall learning environment, the selection of applicable techniques and the generation of training data, many ontology learning algorithms combine per design human and computational intelligence, where human skills are required to evaluate intermediary results and decide upon the further steps in the learning process.

One novel approach which proved successful to solve technical tasks via human computation is based on 'games with a purpose' [32]. The idea behind games with a purpose is simple, but effective; tasks which remain difficult to handle by machines, but which humans seem to accomplish easily are hidden behind entertaining, collaborative games targeting not experts, but casual Internet users. By playing a game with a purpose, users are indirectly generating data that can be capitalized to build knowledge corpora required for the training of algorithms, and to validate the results of such algorithms, thus providing a powerful example of how human and computational intelligence can be interwoven to address important, challenging problems. Since the original proposal by Van Ahn in 2006 games with a purpose have been applied to tasks as diverse as image and video annotation,[1] genetics,[2] natural language processing,[3], conceptual modeling,[4] and through our own work in the context of the OntoGame gaming framework, to ontology engineering and semantic annotation.[5]

Games with a purpose is only one of the most popular instances of socially-inspired approaches to knowledge acquisition proposed in the last years. According to recent surveys people spend a substantial amount of time every day on playing games [1,2,3]. Games in general create an environment which is in itself intrinsically motivating, offering through their very nature many motivational features such as fun, challenge, fantasy, competition and collaboration with others, and recognition, to name only a few [17]. Through games with a purpose, one leverages these features to turn the significant number of hours willingly spent playing by Internet users through sophisticated algorithms into meaningful results that lead to qualitative improvements of information management technology. The concept is particularly useful for those types of problems that so far have not been the classical subject of popular science and engineering, but have targeted highly specialized audiences, because of the expert-driven and knowledge-intensive nature of the underlying tasks to be performed, the novelty of the questions raised, or both.

The area of semantic technologies is a good example therefor, with the tasks discussed above, including ontology learning, as typical scenarios which require approaches combining both human and computational intelligence to yield optimal results. The multitude of methodologies, techniques and tools developed over the past decade for the purpose of ontology development, alignment or learning offer many sophisticated features, but they are hardly accessible to a lay audience due to the fact that they assume in-depth expertise not only with respect to the task at hand, but also with respect to the underlying processes and procedures according to which the tasks are executed. These limitations laid the foundations for the emergence of a new field of research in semantic technologies, inspired by the success of the Web 2.0 phenomenon in encouraging user participation and capitalizing on the power of collective intelligence [5,14,19,22,31]. Despite early success stories and promising prospects, the application of these crowdsourcing approaches comes with a trade-off: putting aside the (important) question of motivators and incentives, addressing a priorly unknown user base that reaches far beyond the scope for which expert tools have been created constrains the complexity of the tasks that can be feasibly undertaken, and the domains for which knowledge can be reliably collected

[1] http://www.gwap.com/
[2] http://fold.it
[3] http://galoap.codeplex.com/
[4] http://apps.facebook.com/conceptgame/
[5] http://ontogame.sti2.at/

from a mass audience, and introduces the need for specific evaluation and quality assurance mechanisms. The collaborative nature of these approaches imposes further restrictions with respect to the types of tasks to be tackled, and may require mechanisms for decomposing the work into smaller chunks to be carried out (to a certain degree) independently, and for combining the partial results. In the particular case of games with a purpose, an additional challenge is the design of the games, which have to fulfill highest usability expectations, be enjoyable and engaging to ensure players retention, and, as a side effect, serve the purpose for which they have been actually created. Turning back to the area of semantic technologies, the types of semantic-content authoring problems that can be realistically solved via games with a purpose are greatly confined. The ontologies and semantic annotations that result are lightweight (with respect to expressivity) and cover mainstream topics that a large share of Internet users feel related to. In addition, a great majority of the tasks that have been an active subject of research and development over the past decade are too complex, too unstructured, or both to be turned into entertaining game experiences. In Section 4 we will elaborate on these aspects in more detail by means of several examples.

To conclude this introductory section, the concept of games with a purpose has rapidly achieved great resonance in many research communities confronted with the problem of attracting user involvement for the resolution of various tasks of technical nature, which had been for a long time the realm of expert audiences. This chapter will provide a selection of the games that have been developed in the last years to support the learning of ontologies, as a proof for the increasing popularity of the overall idea. However, as already discussed above, as in every emerging field of research, there are still a number of open issues of theoretical and practical nature that require closer investigation in order to gain a thorough understanding of the new challenges arising by applying a game-based approach to semantic content authoring tasks, and to optimize the outcomes of existing games and their exploitation.

The remainder of this chapter is structured as follows. We will start with an analysis of the process of ontology learning in order to provide an overview of typical activities and tasks that rely on human contributions (Section 2). In Section 3 we will survey existing games concerned with the learning of ontologies and other related knowledge structures, and compare and discuss their features against the results of the process-oriented analysis. In Section 4 we will concentrate on the realization of new games serving similar purposes; we will present the Generic Gaming Toolkit delivered by the European research project INSEMTIVES which facilitates the development of games for semantic content authoring, and a number of guidelines and best practices derived from our own experiences in the context of the OntoGame framework, which may prove useful for game designers. We conclude the paper with a short summary and an outline of potential future lines of research in the area (Section 5.

## 2. Combining Human and Computational Intelligence in Ontology Learning

In this section we will analyze the role of human contributions in ontology learning projects, and identify related activities and tasks which rely on these contributions. The analysis is based on a larger survey conducted in 2009 of some of the most prominent methodologies, methods and tools in the broad area of semantic content authoring, in-

cluding ontology development, evaluation, learning, alignment and semantic annotation, whose findings can be found in [24].

Ontology learning approaches can be categorized according to the types of resources that are used as input to derive ontological knowledge: unstructured resources such as textual documents, and semi-structured resources such as folksonomies and UML diagrams. Cimiano [7] presents the ontology learning layer cake, which covers the most relevant types of methods and techniques, as well as core activities and tasks in ontology learning. In our analysis, we considered a selection of ontology learning approaches published in the Semantic Web literature over the past ten years to identify commonalities and characteristics regarding the types of human input that they require. We focused on methodological aspects rather than on the features and functionality of specific tools, as our aim was to gain a better understanding of the actual tasks which are typically undertaken to learn an ontology, and not to evaluate or compare existing tools, methods and techniques. Consequently, the analysis is based on a study of the relevant publications, and on the information and experience reports of these publications, and not on actual experiments with the corresponding technology.

Maedche and Staab [16] introduce a framework for building ontologies supported by ontology learning, which combines machine learning and knowledge acquisition features. They propose the following five steps for the execution of the learning process:

**Import and reuse** In the first step, relevant sources are collected. This can include heterogeneous documents, regardless of their degree of structure. Additionally, a core ontology is used that contains generic and domain-specific terms. This ontology is later extended.

**Ontology extraction** The second step is about learning new concepts from the document corpus with the help of natural language processing techniques.

**Ontology pruning** The aim of this step is to focus the ontology on the target domain by removing irrelevant concepts.

**Ontology refinement** The ontology is further extended into relations between concepts and other types of ontological primitives.

**Ontology evaluation** In the final step, the ontology is evaluated in its target application setting. In case it needs to be improved, the process is repeated starting from the first step.

Aussenac-Gilles and colleagues [4] propose the Terminae method for ontology engineering from texts. The approach combines linguistic and modeling techniques. It is semi-automatic in the sense that it depends on the input of a human "supervisor". The method consists of three main steps as follows:

**Domain resources** In the first step domain resources are gathered. This includes a corpus of text documents, as well as existing ontologies, terminologies and other potentially useful knowledge structures. The authors recommend the assistance of a domain expert to choose representative texts providing good coverage of the domain, and define guidelines for the selection of such corpora.

**Linguistic analysis** In the second step a linguistic analysis is performed by applying different natural language processing techniques to the corpus put together in the previous step. This is an iterative process that eventually results into lexical data which forms the basis for the development of the conceptual model.

**Conceptual model** The third step is about building this conceptual model, including the definition of concepts and their hierarchical organization. Furthermore, normalization can be carried out using methods such as OntoClean [12] in order to improve the conceptual model .

Simperl and colleagues [21] propose a methodology for ontology learning that is embedded in the ontology engineering process by Gomez-Perez and colleagues [11]. They introduce a process model with eight phases and finer-grained activities:

**Feasibility study** The first phase assesses whether a learning approach to support ontology development is feasible, starting from the requirements specification that was created in the course of the ontology engineering project. Ontology engineers, domain experts, and ontology learning experts collaborate in order to come up with a risk analysis document. The feasibility study is then split into a number of sub-tasks to define the main dimensions of the prospected ontology learning project, including the specification of the type of ontology to be created and the information sources from which the ontology will be learned.

**Requirements specification** This phase is concerned with the specification of the requirements for the ontology learning process based on the ontology requirements specification document (ORSD) and the risk analysis document produced earlier. The methodology further refines the operation of the phase by identify additional sub-tasks referring to, for instance, the characteristics of the corpus to be used as input for learning, and the features that should be provided by the software to be used.

**Selection of information sources, learning methods and tools** At the end of this phase the ontology learning team has made a decision on the actual methods and tools to be used and built a corpus of information sources that will be processed to learn the ontology.

**Learning preparation** Here ontology learning experts configure the learning tools and prepare the information sources for carrying out the learning process. The result is a tool environment, a documentation, and an execution plan, which specify which tools will process which resources, how they will be configured and executed, and when and what type of user intervention is required to run the overall process.

**Learning execution** In this phase, the actual knowledge is acquired through the operation of the ontology learning tools on the information sources previously collected. This involves human input for the completion of semi-automatic techniques or for the evaluation of intermediary results, which might lead to adjustments and reiterations of specific parts of the tool pipeline.

**Ontology evaluation and integration** The team analyzes the resulting ontologies against the requirements specification, and integrates them into the final ontology following methods of ontology integration [10,18].

*The Role of Human Input* Ontology engineering can be heavily supported by computer programs. Yet, in most of the cases the results of automatic ontology learning methods and techniques are not sufficient in terms of completeness and accuracy, and human processing skills are required at various stages in the process. In terms of the methodology presented in [21], which splits the overall ontology learning process in atomic tasks, the feasibility study, the requirements specification, and the selection of information sources

and ontology learning methods and tools are clearly human-driven. In addition, human input is helpful for the configuration of the learning environment and for resuming the execution of those tools that follow a semi-automatic approach, or that require the user to evaluate intermediary results in order to decide upon the further steps in the learning process. The question of how to optimally acquire or collect the human input required at all these levels does not have a unique or simple answer. Studies in organization theory have identified a plethora of factors that affect the success of a crowdsourcing approach in a specific setting, and proposed means to measure the performance of groups jointly accomplishing work. The type of task to be performed (combinable vs unitary, divisible, optimizable, see [27]), as well as the social structure and the nature of the task outcomes (public vs private) are key factors in this context [8,9,20]. While the organizational context of an ontology learning project needs to be studied on a case-to-case basis to understand its characteristics and their impact on the intrinsic and extrinsic motivations of the individuals and the team as a whole, the activities and tasks that are typically part of the process can be studied with respect to their potential to be meaningfully crowdsourced, be that as part of a game with a purpose (or in fact any other related approach to harness human computation in the large). From the tasks identified in [21] the ones which can be hidden behind a gamelike interface are related to the validate the intermediary results of an ontology learning algorithm, which consist in suggestions for terms, synonyms, concepts, and relationships between concepts [7]. Following design guidelines published in the literature [34,17], covering the full range of expressivity of ontology learning algorithms within a single game is likely to yield suboptimal results, mainly because of the complexity of the task involved. Instead, one could imagine a series of different (selection-agreement) games solving inter-related sub-tasks such as identifying synonyms, building a taxonomy of concepts, describing the relationship between concepts, translating concept labels into different languages, or documenting ontological concepts. In the next section we give an overview of some of the games with a purpose addressing these aspects.

## 3. Games for Learning Ontologies

The selection is based on a survey of relevant literature in the area of games with a purpose, complemented by submissions to the INSEMTIVES Game Challenge.[6] For each of the games, we provide a summary of its most important characteristics, resulting from trials of the games - when available online - and information available in publications and documentation:

**Aim** Short paragraph including the name, the purpose and some general information about a game.

**Knowledge corpus** Some games are tied to a specific domain, others are domain-independent. They use a specific collection of digital resources to generate challenges to be solved during gameplay.

**Human contribution** The types of inputs collected, and the modes of interaction with the players.

**Output** Data generated by the game.

**Link** Link at which readers can try out the game.

[6] `http://challenge.insemtives.eu/`

### 3.1. OntoPronto

**Aim** OntoPronto [23] is a two-player real-time quiz game for ontology development, specifically for the classification of concepts according to a pre-defined upper-level ontology. As a corpus, OntoGame uses Wikipedia articles, which players have to map consensually to the most specific class of the PROTON ontology [28]. The game is designed for two players, but also supports a single-player mode on pre-recorded game challenges. Points are earned for consensual answers to the challenges included in each game round. Through an integration with Facebook, achieved results can also be made public within the social network of players.

**Knowledge corpus** The game uses a random selection of Wikipedia articles and the PROTON ontology, but could be easily adjusted to other scenarios.

**Human contribution** The users have to read the first paragraph of a randomly chosen Wikipedia article and first agree whether the article represents a concepts or an instance (Figure 1). In the next step, they have to select the class of the PROTON ontology that best describes the corresponding article.

**Output** Ontology consisting of classes and instances classified according to the PROTON ontology, represented in SKOS.

**Link** `http://ontogame.sti2.at/`


**Figure 1.** OntoPronto: Extending an existing ontology with classes and instances

### 3.2. Concept Game

**Aim** Concept Game [13] focuses on the creation of a commonsense knowledge base consisting of assertions created via text mining that are validated by players. It is a single-player game, timed, implemented as Facebook application. Players are rewarded for choosing the correct answers - correctness being derived via majority voting - and progress and unlocking of a new game level are posted publicly on the Facebook wall of the user.

**Knowledge corpus** The triple-like assertions are obtained from automatic text mining algorithms applied on a seed set of concepts from the ConceptNet[7] semantic network and Wikipedia articles. The concepts have been annotated by experts and used as basis for building BagPack models. The resulting confidence values were used to rate assertions from the dependency parsed Wikipedia corpus made available by the WaCky project.[8] Top-rated assertions build the corpus of Concept Game.

**Human contribution** The users have to validate previously collected and randomly chosen candidate assertions (Figure 2), indicating whether the corresponding statement is meaningful or meaningless.

**Output** A commonsense knowledge based, validated by humans.

**Link** `http://apps.facebook.com/conceptgame/`

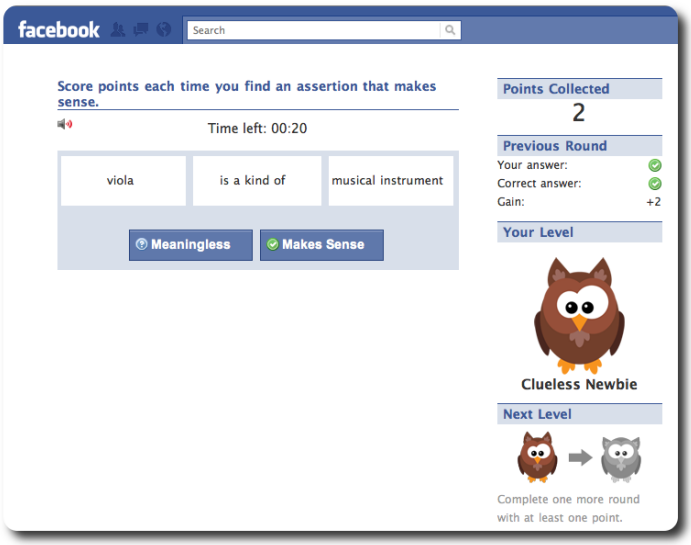



**Figure 2.** Concept Game: Validating commonsense assertions

[7] (`http://conceptnet.media.mit.edu/`)
[8] `http://wacky.sslmit.unibo.it`

### 3.3. FACTory Game

**Aim** FACTory Game is a single-player, turn-based online quiz game that develops a knowledge base of commonsense facts validated by humans. Scoring is designed in the same way as for Concept Game, correct answers being identified based on multiple answers on the same questions from different players.

**Knowledge corpus** The game randomly chooses facts from the Cyc knowledge base and presents them to the players. The assertions from Cyc consist of facts, rules of thumb and heuristics for reasoning about the objects and events of everyday life. The content is represented in a formalized language describing terms and assertions on these terms (ground assertions as well as rules).

**Human contribution** The users are provided with knowledge from the Cyc knowledge base and decide whether the given information is true, false, does not make sense or whether it is not possible to say whether the information makes sense (Figure 3).

**Output** Validated assertions extracted from the Cyc knowledge base.

**Link** `http://game.cyc.com`

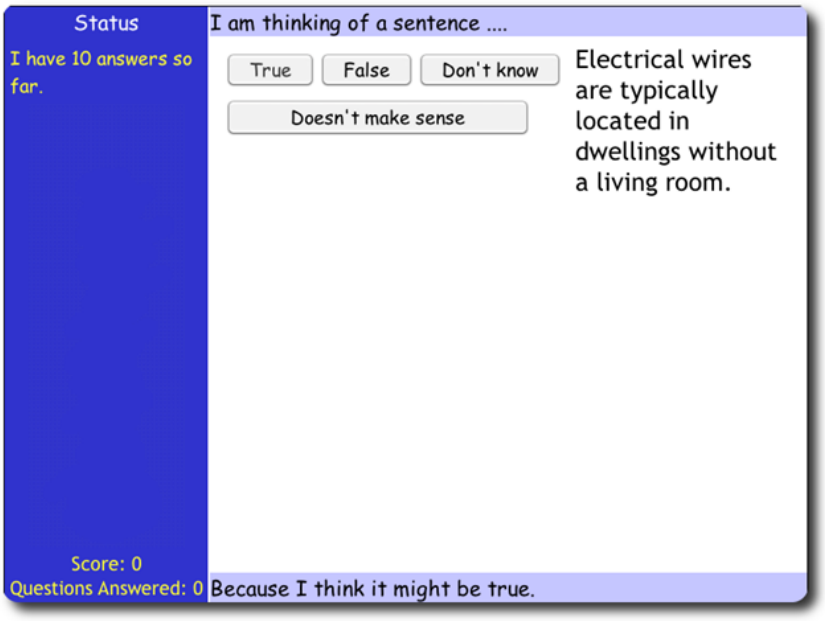



**Figure 3.** FACTory Game: Validating Cyc knowledge

### 3.4. Phrase Detectives

**Aim** Phrase Detectives is about indicating relationships between words and phrases [6]. It is a multi-player online game tackling the challenging problem of anaphora resolution. Anaphora resolution systems are useful for many applications of com-

putational linguistics, for instance, in information extraction, text summarization and search. In its newest release the game is also available as Facebook application, where people can compete against experts or in a team with their friends (in selection-agreement mode).

**Knowledge corpus** Phrase Detectives uses texts from Wikipedia, as well as fairy tales and other types of texts. From this knowledge corpus, users are assigned text fragments they have to annotate according to specific rules and instructions.

**Human contribution** The players annotate text fragments and make specific annotation decisions. They have to complete different tasks as for example to make decisions regarding whether or not a certain phrase has appeared already before in the game, determining whether a phrase is referring to something else, or whether it is a property of another phrase (Figure 4). Contributions are rewarded with points and upgrades to the so-called 'graduate' level. Leader and top scores boards advertise the achievements of the most successful players. Input validation is realized via comparisons with existing gold-standard texts and with answers provided by other users on the same text.

**Output** Text corpus annotated with anaphoric references.

**Link** `http://anawiki.essex.ac.uk/phrasedetectives/`

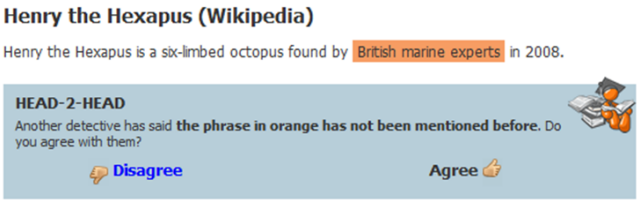



**Figure 4.** Phrase Detectives: Identifying relationships between words and phrases for anaphora resolution

### *3.5. Rapport Game*

**Aim** Rapport Game [15] is a game for building a semantic network encoding commonsense knowledge. To do so players ask other players questions related to interesting topics. The answers they receive to their questions are then evaluated by majority voting and taken as basis for building the network.

**Knowledge corpus** Any topic that is of interest for the players as they start interacting with each other by asking/answering questions related to a chosen topic.

**Human contribution** Players choose other players from a list and start interacting with each other in one of four possible ways. They start discussing by asking/answering questions related to topics of their interest and voting each other's answers (Figure 5).

**Output** Semantic network.

**Link** `http://apps.facebook.com/conceptnet/`


**Figure 5.** Rapport Game: Question-answering to build a semantic network

### *3.6. Verbosity*

**Aim** Verbosity [33]) is a two-player online game for gathering commonsense facts. In each round, one player is assigned the role of the 'describer' and the other the 'guesser'. The describer is producing hints for the guesser to guess the input suggested by the game designer. The goal is that the guesser reproduces the original input of the game based on the information supplied by the describer. In this way, the game collects commonsense facts about virtually any topic. The game can also be played in a single-player mode, simulating the describer.

**Knowledge corpus** Arbitrary topics.

**Human contribution** The describer gives hints to the guesser about the topic suggested by the game designer by filling in triple-like sentence templates with pre-defined predicates 'is-kind-of', 'is-used-for', and 'is-the-opposite-of', while the guesser enters terms that may fit the description (Figure 6). As the describer sees the answers of the guesser, she can also indicate whether the guesser gets closer to the solution or not. Points are assigned cooperatively, whenever the pair of players successfully completes a game round.

**Output** Knowledge base of commonsense facts expressed as triples using pre-defined predicates.

**Link** `http://www.gwap.com/`


**Figure 6.** Verbosity: Collecting commonsense facts

### *3.7. WhoKnows?*

**Aim** In WhoKnows? (Figure 7) players evaluate the accuracy of DBpedia knowledge answering questions automatically extracted from DBpedia triples.[9] Questions which are marked as meaningless by a certain number of users indicate data quality problems in DBpedia and could be subsequently used for ranking and curation purposes.

**Knowledge corpus** WhoKnows? uses questions of the form 'object is-property-of subject' which are generated automatically from DBpedia triples. Each challenge is conceived as a multi-choice question consisting of correct and incorrect answers. The latter are created automatically from DBpedia triples that hold the same property and a different object.

**Human contribution** The player is provided with questions that he should answer correctly and as fast as possible. Each question has a closed set of answers, including correct and incorrect choices.

**Output** Validated DBpedia assertions, ranking of facts.

### *3.8. Comparison*

As discussed in Section 2 many aspects of the execution of ontology learning algorithms can be turned into games with a purpose following different game models. This includes computations linguistics aspects, as illustrated in the Phrase Detectives game, but also conceptual modeling and fact validation aspects as shown in OntoPronto and the multitude of commonsense fact checking instances of which this chapter can offer only a small selection. While many of the games surveyed here share common aims and ideas, their underlying game mechanics is very different at various levels: the actual game play,

[9] http://dbpedia.org/

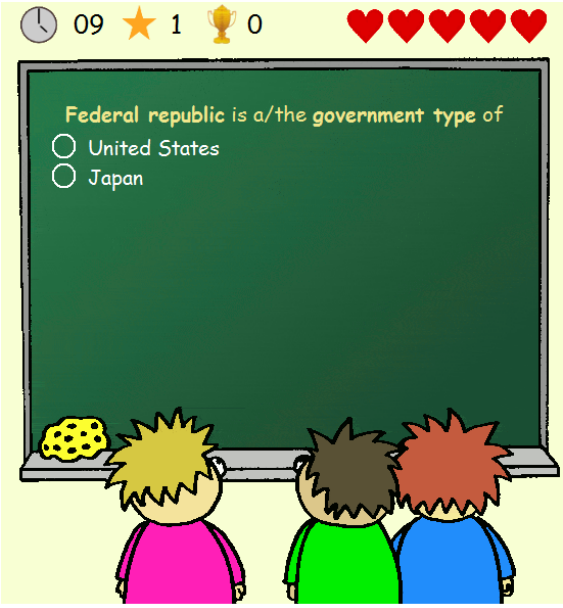



**Figure 7.** WhoKnows?: Curating DBpedia

the quality assurance measures by which players inputs are accepted, and the incentives that are put in place to reward players contributions and encourage players retention.

Table 1 summarizes the most important features of the games surveyed in this section.

The evaluation of players' inputs is done either by comparison or cooperatively when pairs of players agree upon the solution to a challenge. In either cases it is essential that the game attracts a critical mass of players. For comparative metrics, an answer can be reliably validated only if it has been confirmed in multiple game rounds. In addition, cooperative metrics rely on a gameplay in which pairs of players are assigned randomly in each round, which implies that at each moment in time there has to be a certain number of players logged in to the game; otherwise, the second player can be emulated (as it is done, for instance, in Verbosity and OntoPronto) using pre-recorded challenges.

The types of knowledge that is processed through the games vary, however, all games we surveyed[10] choose topics of general interest and openly-available knowledge corpora as foundation for the selection of challenges. As this selection is performed automatically, most of them also offer means for users to filter potential meaningless challenges, thus constantly improving their background algorithms. The data generated through the games consists of knowledge assertions that can be used to create, extend or curate a knowledge base - or in the case of Phrase Detectives, train anaphora resolution systems.

In terms of the motivation for users to contribute, most tools rely on simple game mechanics related to what Malone referred to as 'challenge' in [17]: timed game rounds, rewards implemented as scores, leader boards and different game levels. Multi-player games such as OntoPronto and Verbosity encourage cooperation in the sense that players can win points only if they agree with their partners on the solution to a given problem.

[10] See also `http://www.insemtives.eu/games` for a more comprehensive overview.

**Table 1.** Comparison table

| | **Aim** | **Knowledge corpus** | **Human contribution** | **Output** |
|---|---|---|---|---|
| OntoPronto | ontology development, classification of concepts | random selection of Wikipedia articles, PROTON ontology | read a paragraph from Wikipedia, agree whether the article represents a concepts or an instance, select the class of the PROTON ontology that best describes the corresponding article | Ontology consisting of classes and instances represented in SKOS |
| Concept Game | develop knowledge base of commonsense knowledge | seed concepts from ConceptNet, assertions from Wikipedia corpus | decide about meaningfulness of assertion | validated assertions, forming commonsense knowledge base |
| FACTory Game | curate Cyc | assertions from Cyc facts, rules of thumb and heuristics for reasoning about the objects and events of everyday life | decide whether assertions are meaningful | validated assertions |
| Phrase Detectives | anaphora resolution | texts from Wikipedia, fairy tales and other types of texts | indicate relationships between words and phrases | annotated data set of anaphoric references |
| Rapport Game | building commonsense semantic network | any topic that is of interest for the gamers | discuss by asking and answering questions related to topics of interest and voting other players' answers | semantic network |
| Verbosity | gathering commonsense facts on given topic | any | describe given topic using predefined sentence templates, guess topic | database of commonsense facts |
| Who Knows? | curate DBpedia, rank DBpedia facts | set of questions and possible answers created based on DBpedia entities, predicates and categories | answer multi-choice triple-like questions | validated assertions |

For such game models it is assumed that player enjoyment is triggered primarily by social aspects, in this case the challenge being to achieve consensus with a game partner who is not known in advance using only the communication channels that are provided by the game designer; such selection-agreement games also promote fantasy and creativity, which are known as important factors to make a game more interesting for its audience [17].

## 4. Building New Games

This section is dedicated to the development of new games with a purpose for ontology learning, and any other type of scenario involving the creation and management of Semantic Web data. We introduce the Generic Gaming Toolkit, a extensible framework which provides the main building blocks for the design and implementation of semantic games, and conclude with a series of general guidelines derived from relevant literature on games and game mechanics and from our own experiences in the context of OntoGame and the INSEMTIVES project.

### *4.1. Enabling Technology*

The Generic Gaming Toolkit facilitates the development of semantic games (see Figure 8). It consists of a programming API covering the most common functional components which are required to turn a specific technical task into a game with a purpose, and a technology platform to run such games [26,29]. Both the source code and the associated code documentation are publicly available at SourceForge.[11]

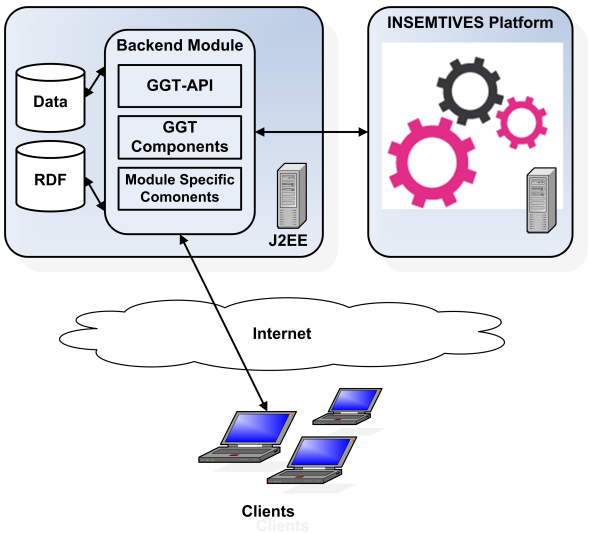



**Figure 8.** Generic Gaming Toolkit: Overall architecture

[11] http://insemtives.svn.sourceforge.net/viewvc/insemtives/

Conceptually Generic Gaming Toolkit differentiates among challenges, inputs, resources, and play records. A 'challenge' abstracts from the actual task players carry out while playing the game. Each challenge can be split into several sub-challenges, which are structured hierarchically. In OntoPronto, for instance, the main task is to classify the topic of a Wikipedia article in the PROTON ontology, and is sub-divided into two atomic tasks: deciding whether an article refers to a concept or an instance, and identifying the most representative class in PROTON the class or instance from the previous step can be classified as. A 'resource' abstracts from the topic a challenge is about, for example, a YouTube video, an ontology concept or a Wikipedia article. A resource may also refer to the answer provided by a player, which in the OntoPronto case, are classes in the PROTON ontology. An 'input' represents an answer of a player to a certain challenge. It is always related to a resource, deals with a specific challenge and may contain a resource as answer. A 'play record' behaves like a container for player inputs about a specific resource. It is used for all subsequent background computations by which player inputs are translated into semantic content, including the evaluation of the inputs, and the actual encoding using Semantic Web standards.

*Agreement Negotiation* Agreement negotiation denominates the process by which a pair of gamers decides whether to take a given challenge or skip. The process starts with collecting inputs from of all players. These can be: 'Disagreed' in case it is not clear whether or not to skip a given challenge, 'Skipped' if both players decided to skip, and 'Agreed' if they declared that they want to provide an answer to the question raised in the game round at hand. Once the inputs of all players have been collected, the game is resumed according to a pre-defined negotiation strategy. In the games implemented so far as part of the OntoGame series, we ask players to reach an agreement on whether to skip or accept a challenge; however, the toolkit can be easily extended to support variations of this simple model, for instance, for multi-player scenarios in which the majority of players decide upon the questions to be answered.

*Partner Matching* Another important feature of the Generic Gaming Toolkit is the matching of the players. When a users logs to a game the system selects a game partner according to criteria such as IP-addresses, location, age, and gender, all of which may influence the quality of the inputs collected. The selection of players within the game ensures that players remain anonymous while consensually resolving challenges, thus increasing the probability that they behave as intended by the game designer. In other games of the OntoGame family we have also used recorded game rounds, for which answers are already available from previous players, in order to realize a single-player mode in addition to the two-players mode deployed in, for instance, OntoPronto.

*Players' Reliability* Reliability is a rough indicator of the trustworthiness of the answers provided by a player; having such an indicator is a very important aspect of the process of semantic content creation in the Generic Gaming Toolkit, since the game mechanics are always about questions where the answer is actually unknown to the game designer. It has to be assumed that there are at least some players that want to trick the system and behave in an irrational and unfavorable way. As the very purpose of the games is to derive useful, structured knowledge from game inputs, each game needs to implement functionality to verify whether an input comes from a trustworthy player or a cheater. Permitting cheating would decrease the gaming fun as well as the quality of the generated data.

*Consensus Finding* One of the most important features of the Generic Gaming Toolkit is the abstract projection of the process to find consensus for the set of answers provided for each question. Finding consensus can be achieved in many different ways such as plainly by a simple majority. However, there are also more sophisticated ways of identifying a consensual set of answers, e.g., by weighting each answer by their player's reliability, considering only answers that have been answered similarly by all players of a certain game round. Additionally restrictions such as a minimum number of different players or a minimal number of answers on a challenge can be imposed on the process of consensus finding.

*Ranking* Ranking is used to give an indication of the importance of a play record. The importance defines which record is most suited to be played next. Ranking metrics can be implemented in various ways, depending on the needs of the game, i.e., one could rank the records that are closest to produce output highest, or records that have not yet been played very frequently. In the Generic Gaming Toolkit we have implemented two different strategies: one that does not rank at all, which is required when ranking is not to be considered during the record selecting procedure; and a second one considering the number of required answers, the number of different players, and the number of questions that have been answered.

*Matching* The last relevant conceptual feature of the Generic Gaming Toolkit is the process of defining whether or not a set of answers coincide. Note that this is different from assessing consensus, as matching answers is used to compute an equality between answers. An example of doing so is comparing two players' answers and evaluating their similarity. Another implemented matching algorithm checks whether a player's answer equals the majority of all previous collected answers about this resource.

### *4.2. Design Principles and Open Issues*

In this section, we summarize some of the most important lessons learned over the last three years of continuously developing the OntoGame framework. The ultimate aim of this line of research is to provide comprehensive decision support in matters related to the execution of human-aided semantic content authoring tasks. This can be achieved by identifying those tasks that can be effectively addressed through games, crowdsourcing platforms such as Amazon Mechanical Turk,[12] or social platforms.

*Task selection.* The identification of those semantic-content-authoring tasks that are suitable for the casual-game paradigm is of paramount importance for the overall success of the approach. Hiding tasks behind games is not trivial, and cannot work for every aspect of semantic-content-authoring, no matter how highly human-driven it might be. Candidate tasks cannot be too difficult, or too easy; they have to be divisible or combinable, so that they can be broken down into smaller chunks that can be independently solved across a potentially large group of contributors [27]. They have to be suitable for a broad audience of players and, in the context of games, be mappable to a series of consensual-decision-making challenges. Our experiences show that the resulting workflows have to be not just constrained in their structure - basically sequences of atomic

[12] `http://www.mturk.com/`

tasks which are approached and solved consensually by players - but also in size. As a rule of thumb, a sequence of more than 3 to 4 interrelated tasks is likely to lead to challenges that are too complicated for players to learn and keep track of at the fast pace the game is expected to be played. This makes the game less appealing for players, thus diminishing their willingness to play and reducing the amount of data produced from their inputs. Each atomic task results in a question - to which the correct answer is not known in advance to the game designer - which needs to be answered by the pair of players competing against each other at a certain point in time in the game. This includes open-scale questions, whose answers are typed in by users in a dedicated field, or closed-scale ones, for which the system provides a set of possible answers from which the players have to choose one viable option. In the first scenario one needs to make use of specific matching algorithms to cope with potential variations in the form of the answers - for instance, different spellings - in order to ensure that consensus finding is feasible in most cases. Nevertheless, if the set of potentially correct answers is too broad for a consensus to be likely to emerge in most cases, the task is probably less appropriate for a game-based approach and additional knowledge has to be taken into account to reduce the space of possible solutions. An interesting research question in this context would be the extent to which one could combine a games-based approach, which can obviously solve only very specific types of semantic-content-authoring tasks, with other human-computation paradigms and incentive mechanisms. For example, one could imagine refining the results of casual games in Amazon's Mechanical Turk, or different combinations of games similarly to the GWAP framework.

In ontology learning projects, a game-based approach is feasible as a means to validate the results of automatic algorithms carrying out a specific task either at the level of text corpus from which the ontology is eventually learned, or at the level of the conceptual structures that can be detected. Each of these aspects, however, have to be treated separately, as individual games or game-like experiences, and as such it would be interesting to investigate how related crowdsourcing approaches might fit in the picture, as mentioned earlier. This might prove particularly beneficial especially in the context of ontology learning, given the increasing popularity of Mechanical Turk in computational linguistics and the emerging portfolio of methods, techniques and tools assisting researchers in validating the outcomes of their algorithms and creating training corpora therefor. In addition, the question of combining the results of different crowdsourcing platforms - games or others - from a data and execution flow point of view remains to be solved. The current state of the art in the area - at least as far as the Semantic Web community is concerned - is that more and more games with a purpose, most recently as Facebook applications, are being developed, but it is unclear how their results could be exploited beyond their original scope. This problem is to some extent solved by openly publishing the generated data according to Linked Data principles; however, in order to further optimize the usability and usefulness of the data, in particular in the context of so-called 'crowdsourcing pipelines' that are essential for tackling almost any activity on the ontology life cycle - as these activities are too complex to be crowdsourced as they are - one would need special-purpose metadata schemas capturing the most important parameters of the crowdsourcing project the data resulted from.

*Knowledge Corpora* Games with a purpose usually need a corpus of knowledge to start with, which is an integral part of the game challenges. Challenges cannot be shown too often to the same players without damaging the game experience [34]. This requires a

large repository of knowledge in the background that can be used as input, whereas online collections of resources such as YouTube, Flickr, Wikipedia or WordNet are surely useful. In the context of SpotTheLink [30], a game on ontology alignment, we experimented with various ontologies in domains such as eCommerce and eTourism as knowledge corpus for the game. None of them received a positive resonance in initial trials - this was due to the structure and size of the ontologies (many inheritance levels, large number of concept siblings per level in the eCommerce setting), or the domain itself (perceived as less interesting by interviewees, or simply too far away from their daily life). Ontologies suitable for an ontology alignment game should be of manageable size (several hundreds of concepts), and in a domain that a broad audience of users can relate to (e.g., media, entertainment, sports, but also ontologies capturing general knowledge such as DBPedia and PROTON, as in OntoPronto or Who Knows?). These limitations may affect the acceptance of a game whose purpose is the learning of an ontology in a domain which is accessible to an expert audience. A second problem game designers have to solve is the quality of the knowledge corpus when the corpus is generated automatically; some of the challenges presented to the users might not be meaningful, thus hampering the game experience.

*Game fun* The challenge here is to reach a balance between an appealing design and the purposefulness of the game with respect to the task to be solved. Games for semantic-content authoring are in many cases on the edge of being too difficult for a non-expert audience. The positive side of this is that such games provide an intellectual challenge, which is important to keep the games sufficiently interesting. The negative side is that creating an attractive and easy-to-grasp interface for such technical tasks is not trivial; user interfaces studies for semantic technologies are still in their infancy even when it comes to expert-oriented environments such as ontology editors. In selection-agreement games, the navigation must force users to move along existing knowledge structures, and make a consensual selection. This requires appropriate visualization techniques, while remaining game-like and playful in terms of the colors and metaphors used. Massive user participation and generation of output is crucial for the games and the methods they incorporate: they require a critical mass of contributions. Even when the task is intellectually challenging, and the interface is perceived as usable and pleasant, we cannot expect a massive user involvement per se. Additional incentives schemes and motivations are needed to make users start and continue playing. This can include measures such as competition, reputation and sociability. Keeping scores is an important feature of every game. Players want to improve their ranking and standing within the player community. Measures which should be applied are making score lists prominent - and rewarding progress through badges and alike - making users aware when they are about to lose a rank or when they improved, and to provide means for users to be able to brag about the results on social platforms. With respect to sociability, knowing that they are playing against a real partner is motivating for many players. As the games are cooperative by design, players might want to know more about their partner. Allowing communication after the gaming session - if they achieve a certain amount of points - could be such a measure. Moreover, players should be able to indicate preferences for the choice of their partners, be able to invite people from their social network to play the game, and report extensively on their achievements, for instance, through frequent status updates.

Recent studies on the massive success of social games deployed via Facebook and other platforms, which are similar to games with a purpose through their relatively sim-

ple, casual plot and the incentive mechanisms they incorporate, have identified a number of essential game-design features which ensure growth in number of players and encourage player retention. Many of these features are available at best only in a very basic form in the games with a purpose we surveyed: multiple difficulty levels, changing game goals, virtual gifts, to name only a few. It is yet unclear how these features could be implemented in games aiming at solving a Semantic Web-related problem, as the underlying tasks are often repetitive, and distinguishing variable levels of difficulty and goals automatically is not straightforward. In most cases, the games have been evaluated in controlled experiments and there is scanted evidence on their acceptance by the large community of casual gamers, and on their ability to viral growth and players retention.

## 5. Conclusions

In this chapter we have analyzed the process of ontology learning and discuss the feasibility of a game-based approach to solving those problems which computer programs still find difficult despite many years of research, increasing volumes of data available, and progress with respect to the accuracy of the results. We have presented a selection of semantic games that try to address the shortcomings of automatic ontology development by describing their purpose, the domain of interest, the required human contribution and the results of each game. To foster the further development of this new and exciting field of research we have presented the Generic Gaming Toolkit as enabling technology for the implementation and deployment of semantic games, devised general design guidelines, and identified several open research questions.

## Acknowledgements

The work presented has been funded by the FP7 project INSEMTIVES under EU Objective 4.3 (grant number FP7-231181).

## References

[1] Casual Games Association. Casual Games Market Report. `http://www.org.id.tue.nl/IFIP-TC14/documents/CasualGamesMarketReport-2007.pdf`, 2007.

[2] Entertainment Software Association. Essential Facts About the Computer and Video Game Industry. `http://www.theesa.com/facts/pdfs/ESA_Essential_Facts_2010.pdf`, 2010.

[3] International Game Developers Association. 2008-2009 Casual Games White Paper. `http://www.igda.org/casual/`, 2009.

[4] N. Aussenac-Gilles, Sylvie Despres, and Sylvie Szulman. *Ontology learning and population: bridging the gap between text and knowledge*, chapter The TERMINAE method and platform for ontology engineering from texts, pages 199–223. IOS Press, 2008.

[5] S. Braun, A. Schmidt, A. Walter, G. Nagypal, and V. Zacharias. Ontology maturing: A collaborative web 2.0 approach to ontology engineering, May 8 2007.

[6] Jon Chamberlain, Massimo Poesio, and Udo Kruschwitz. A demonstration of human computation using the phrase detectives annotation game. In *Proceedings of the ACM SIGKDD Workshop on Human Computation*, pages 23–24, 2009.

[7] P. Cimiano. *Ontology Learning and Population from Text: Algorithms, Evaluation and Applications*. Springer, 2006.

[8] R. Cuel, O. Morozova, M. Rohde, E. Simperl, K. Siorpaes, O. Tokarchuk, T. Widenhoefer, F. Yetim, and M. Zamarian. Motivation mechanisms for participation in human-driven semantic content creation. *International Journal of Knowledge Engineering and Data Mining*, 1(4), 2011.

[9] R. Cuel, O. Tokarchuk, and M. Zamarian. Mechanism Design for Designing Annotation Tools. In *Proceedings of the the Sixth International Conference on Internet and Web Applications and Services (ICIW 2011)*, 2011.

[10] K. Dellschaft and S. Staab. *Ontology Learning and Population: Briding the Gap between Text and Knowledge*, chapter Strategies for the evaluation of ontology learning, pages 253–272. IOS Press, 2008.

[11] A. Gomez-Perez, M. Fernandez-Lopez, and O. Corcho. *Ontological Engineering*. Advanced Information and Knowledge Processing. Springer, 2004.

[12] N. Guarino and C. A. Welty. Evaluating ontological decisions with ontoclean. *Communications of the ACM*, 45(2):61–65, 2002.

[13] Amac Herdagdelen and Marco Baroni. The Concept Game: Better commonsense knowledge extraction by combining text mining and a game with a purpose. In *Proceedings of the AAAI Fall Symposium on Commonsense Knowledge*, pages 52–57, 2010.

[14] Markus Krötzsch, Denny Vrandecic, Max Völkel, Heiko Haller, and Rudi Studer. Semantic wikipedia. *Journal of Web Semantics*, 5(4):251–261, 2007.

[15] Y. Kuo, J. Lee, K. Chiang, R. Wang, E. Shen, C. Chan, and J. Hsu. Community-based game design: experiments on social games for commonsense data collection. In *Proceedings of the ACM SIGKDD Workshop on Human Computation, HCOMP '09*, pages 15–22, 2009.

[16] A. Maedche and S. Staab. Ontology learning for the semantic web. *IEEE Intelligent Systems*, 16(2):72–79, 2001.

[17] T. W. Malone. What makes things fun to learn? Heuristics for designing instructional computer games. In *Proceedings of the 3rd ACM SIGSMALL Symposium and the first SIGPC Symposium on Small Systems*, SIGSMALL '80, pages 162–169, 1980.

[18] H.S. Pinto and J.P. Martins. A methodology for ontology integration. In *International Conference on Knowledge Capture (K-CAP)*, pages 131–138. ACM Press, 2001.

[19] Sebastian Schaffert, François Bry, Joachim Baumeister, and Malte Kiesel. Semantic wikis. *IEEE Software*, 25(4):8–11, 2008.

[20] E. Simperl, R. Cuel, and M. Stein. *Incentive-Centric Semantic Web Application Engineering*. Synthesis Lectures on the Semantic Web: Theory and Technology. Morgan & Claypool, 2013.

[21] E. Simperl, C. Tempich, and D. Vrandecic. *Ontology learning and population: bridging the gap between text and knowledge*, chapter A Methodology for Ontology Learning, pages 225–249. IOS Press, 2008.

[22] K. Siorpaes and M. Hepp. Games with a Purpose for the Semantic Web. *IEEE Intelligent Systems*, 23(3):50–60, 2008.

[23] K. Siorpaes and M. Hepp. OntoGame: Weaving the Semantic Web by Online Games. In *Proceedings of the European Semantic Web Conference ESWC2008*, pages 751–766, 2008.

[24] K. Siorpaes and E. Simperl. Human intelligence in the process of semantic content creation. *World Wide Web Journal*, 13(1), 2010.

[25] K. Siorpaes and E. Simperl. Incentives, Motivation, Participation, Games: Human Computation for Linked Data. Linked Data in the Future Internet (Future Internet Assembly), 2010.

[26] K. Siorpaes and S. Thaler. Requirements and design of a generic gaming toolkit and api. Deliverable D4.1.1, INSEMTIVES, March 2010.

[27] I. D. Steiner. *Group Process and Productivity (Social Psychological Monograph)*. Academic Press Inc, 1972.

[28] I. Terziev, A. Kiryakov, and D. Manov. Base upper-level ontology (bulo) guidance. Technical report, 2005.

[29] S. Thaler. Generic gaming toolkit and api implementation. Deliverable D4.1.2, INSEMTIVES, September 2010.

[30] S. Thaler, E. Simperl, and K. Siorpaes. SpotTheLink: A Game for Ontology Alignment. In *Proceedings of the 6th Conference for Professional Knowledge Management Innsbruck (WM2011)*, 2011.

[31] Tania Tudorache, Sean M. Falconer, Natalya Fridman Noy, Csongor Nyulas, Tevfik Bedirhan Üstün, Margaret-Anne D. Storey, and Mark A. Musen. Ontology development for the masses: Creating icd-11 in webprotégé. In *Knowledge Engineering and Management by the Masses - Proceedings of the 17th International Conference (EKAW 2010)*, pages 74–89, 2010.

[32] L. Van Ahn. Games with a purpose. *IEEE Computer*, 29(6):92–94, 2006.

[33] L. Van Ahn, M. Kedia, and M. Blum. Verbosity: a game for collecting common-sense facts, 2006.
[34] L. von Ahn and L. Dabbish. Designing games with a purpose. *Communications of the ACM*, 51(8):58–67, 2008.
[35] Denny Vrandecic and Anja Jentzsch. Linked data and the semantic web. In Johanna Völker and Jens Lehmann, editors, *Perspectives of Ontology Learning*, Studies on the Semantic Web. AKA Heidelberg / IOS Press, 2012.

# Formal Concept Analysis Methods for Interactive Ontology Learning

Sebastian RUDOLPH [a], Barış SERTKAYA [b]
[a] *Karlsruhe Institute of Technology*
*rudolph@kit.edu*
[b] *SAP Research Center Dresden*
*baris.sertkaya@sap.com*

**Abstract.** In this chapter we give a thorough overview over ontology learning approaches based on Formal Concept Analysis. By using information provided by a domain expert, our approaches learn the relationships between classes of the application domain and extend the ontology accordingly. We present basis for efficient and formally well-founded techniques and tools that can be used for interactive ontology learning. The use of techniques from Formal Concept Analysis ensures that, on the one hand, the interaction with the expert is kept to a minimum, and, on the other hand, enables us to show that the knowledge in the resulting ontology is complete in a certain, well-defined sense.

**Keywords.** interactive ontology learning, formal concept analysis, ontology exploration

## 1. Introduction

Designing and maintaining ontologies is a cumbersome and error-prone task. In most of the cases, during the design phase, the ontology engineer cannot fully specify the relationships between the classes of the application domain either because there are too many of them or because he does not have a complete list of these relationships ready at hand. In such cases, automated tools are not of much use either, since this task requires comprehensive knowledge of the application domain, which means that interaction with a domain expert is required. Besides designing an ontology, maintaining an existing one has similar problems. Here the ontology engineer faces the problem of checking whether the ontology already has all of the relevant relationships between the classes or not. Again this problem cannot be solved by an automated tool alone because such questions can only be answered by a domain expert. What one can use to solve these problems is a semi-automated tool that *learns the application domain* from a domain expert. More precisely, this method should be able to interactively acquire domain knowledge from an expert in order to support the design or maintenance of the ontology.

An interactive knowledge acquisition algorithm developed in Formal Concept Analysis can be used to solve these problems. Formal Concept Analysis (FCA) [14] is a field of applied mathematics that is based on a lattice-theoretic formalization of the notions of concept and conceptual hierarchy. It has been successfully used in many areas of com-

puter science including data analysis, data mining and machine learning. FCA provides efficient algorithms for analyzing data and discovering hidden dependencies in the data. It also allows the user to visualize the data in an easily understandable way. In addition to these, FCA provides a knowledge acquisition method that efficiently acquires knowledge in a specific application domain by asking questions to a domain expert. Moreover, this method guarantees completeness of the resulting knowledge in a well-defined way.

FCA has been used in the literature to create or complete ontologies in an interactive process involving both reasoning engines and human experts. As a general rationale behind these techniques, the amount of interaction with the experts – and hence their workload – shall be minimized. Consequently, the overall process can be seen as a kind of user-assisted ontology learning. Although this sounds fairly straightforward, one cannot directly use classical FCA methods for this purpose. They have to be adapted to meet the requirements of ontological knowledge representation such as the open-world assumption or the potential infinity of class descriptions.

In this chapter we will give a thorough overview over the field of FCA-based ontology learning. Section 2 will introduce the foundational notions used in FCA. Section 3 provides the connections between notions of ontological knowledge representation and notions of FCA and sketches a basic algorithm how ontologies can be generated in an FCA-style way. Section 4 presents substantial refinements to this algorithm necessary to employ these techniques in an open-world setting with already present prior ontological knowledge. In Section 5, this approach is further extended in order to account for the potentially infinite amount of class descriptions.

## 2. Formal Concept Analysis

In FCA, data is represented in the form of so-called *formal contexts*, which in their simplest form just specify for a given set of objects and attributes, which attributes are satisfied by which objects.

**Definition 1** (Formal context)**.** A *formal context* is a triple $\mathbb{K} = (G, M, I)$, where $G$ is a set of objects, $M$ is a set of attributes, and $I \subseteq G \times M$ is a relation that associates each object $g$ with the attributes satisfied by $g$. In order to express that an object $g$ is in relation $I$ with an attribute $m$, we write $gIm$. $\diamond$

A formal context is usually visualized as a cross table, where the rows represent the objects, and the columns represent the attributes of the context. A cross in column $m$ of row $g$ means that object $g$ has attribute $m$, and the absence of a cross means that $g$ does not have attribute $m$.

**Definition 2** (Derivation operator)**.** Let $\mathbb{K} = (G, M, I)$ be a formal context. For a set of objects $A \subseteq G$, we define the set of attributes that are satisfied by all objects in $A$ as follows: $A' := \{m \in M \mid \forall g \in A.\ gIm\}$. Similarly, for a set of attributes $B \subseteq M$, we define the set of objects that satisfy all attributes in $B$ as follows: $B' := \{g \in G \mid \forall m \in B.\ gIm\}$. $\diamond$

Given a formal context, the most common method to analyze it is to find (a canonical base of) the implications between the attributes of this context. Implications between attributes are constraints that hold in a given context. They are statements of the form

"Every object that satisfies the attributes $m_{i1}, \ldots, m_{ik}$ also satisfies the attributes $m_{j1}, \ldots, m_{j\ell}$."

Formally, an implication between attributes is defined as follows:

**Definition 3** (Implication between attributes)**.** Let $\mathbb{K} = (G, M, I)$ be a formal context. An *implication between the attributes* in $M$ is a pair of sets $L, R \subseteq M$, usually written as $L \to R$. An implication $L \to R$ *holds* in $\mathbb{K}$ if every object of $\mathbb{K}$ that has all of the attributes in $L$ also has all of the attributes in $R$, i.e., if $L' \subseteq R'$. We denote the set of all implications that hold in $\mathbb{K}$ by $Imp(\mathbb{K})$, and call it the *implicational theory* of $\mathbb{K}$. $\diamond$

The implicational theory $Imp(\mathbb{K})$ of a formal context $\mathbb{K}$ can be large. Thus, one is interested in small bases generating $Imp(\mathbb{K})$. There may exist different bases of $Imp(\mathbb{K})$, and not all of them need to be of minimum cardinality. A base $\mathcal{J}$ of $Imp(\mathbb{K})$ is called *minimum base* iff no base of $Imp(\mathbb{K})$ has a cardinality smaller than the cardinality of $\mathcal{J}$. Duquenne and Guigues have given a description of such a minimum base [15] for formal contexts with a finite set of attributes. It is called the *Duquenne-Guigues Base* or the *stem base* of a formal context.

In some applications where one wants to compute the Duquenne-Guigues Base, the formal context is not given explicitly as a cross table, but it is only implicitly "known" to a domain expert. In such cases, Ganter's interactive *attribute exploration* algorithm [11,13] has proved to be a useful method to compute the Duquenne-Guigues Base and efficiently capture the expert's knowledge. Consider the following setting: There is an application domain which can be represented as a formal context $\mathbb{K}$, but $\mathbb{K}$ is not explicitly known. However, due to his expertise, a domain expert is able to answer if an implication holds in $\mathbb{K}$ and, in case it does not hold he is able to give a counterexample. By asking implication questions to the domain expert, the attribute exploration method computes a base for $Imp(\mathbb{K})$ and a subcontext $\mathbb{K}'$ of $\mathbb{K}$ such that $Imp(\mathbb{K}') = Imp(\mathbb{K})$. For each implication question, the expert either says that it holds in $\mathbb{K}$, in which case the implication is added to the base, or he gives a counterexample from $\mathbb{K}$, which is then added to $\mathbb{K}'$. What makes attribute exploration an attractive method for capturing expert knowledge is that it guarantees to make the best use of the expert's answers, and to ask the minimum possible number of questions that suffice to acquire complete knowledge about the application domain [11,13].

## 3. Ontology Exploration

Provided with this machinery of attribute exploration, the basic idea to employ it for ontological knowledge acquisition is as follows: We assume that the domain which we want to describe can be represented by one description logic [24] (DL) interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$. Picking a set $M$ of concept names we can extract a formal context from $\mathcal{I}$ in the straightforward way by letting the objects be the domain elements of $\mathcal{I}$ and defining the relation $I$ as concept membership.

**Definition 4** (interpretation context)**.** Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be a DL interpretation and let $M$ be a subset of the concept names of $\mathcal{I}$. Then the corresponding *interpretation context* $\mathbb{K}_{\mathcal{I}}^{M}$ is the formal context $(G, M, I)$ with $G = \Delta^{\mathcal{I}}$ and $I = \{(\delta, A) \mid \delta \in A^{\mathcal{I}}\}$. $\diamond$

The key insight now is that there is a tight correspondence between implications holding in $\mathbb{K}_{\mathcal{I}}^{M}$ and a certain type of GCIs: it can be easily proven that an implication $A_1, \ldots, A_n \to B_1, \ldots B_m$ holds in $\mathbb{K}_{\mathcal{I}}^{M}$ exactly if the GCI $A_1 \sqcap \ldots \sqcap A_n \sqsubseteq B_1 \sqcap \ldots \sqcap B_m$ is satisfied by $\mathcal{I}$. Hence, performing attribute exploration w.r.t. $\mathbb{K}_{\mathcal{I}}^{M}$ allows for acquiring (a base of) all GCIs of that type. Note that, although $\mathbb{K}_{\mathcal{I}}^{M}$ may be infinite, the exploration will terminate as long as $M$ is finite. The next common insight is that already present ontological background knowledge can be exploited by presenting every implication coming up in the course of the exploration first to an automated reasoning engine which might or might not confirm or deny it based on already present information. This eases the workload of the expert who will then be only confronted with an implication if the reasoning engine is not able to confirm or deny it on the basis of the present information. Moreover, GCIs and counterexamples acquired during the exploration process can be directly fed back into the background knowledge.

This general encoding and work flow underlies most of the work of applying attribute exploration to ontology learning tasks. It was first sketched in [27], yet it needed to be substantially refined and extended to meet the requirements of ontological modeling for the Semantic Web. In the sequel we will present these extensions and also provide more technical details on (an adapted version of) the exploration algorithm itself.

## 4. Ontology Completion

The standardization of OWL [19] as the ontology language for the semantic web [7] led to the fact that several ontology editors like Protégé [23], and Swoop [22] now support OWL, and ontologies written in OWL are employed in more and more applications. As the size of these ontologies grows, tools that support improving their quality become more important. The tools available until now use DL reasoning to detect inconsistencies and to infer consequences, i.e., implicit knowledge that can be deduced from the explicitly represented knowledge. There are also promising approaches that allow to pinpoint the reasons for inconsistencies and for certain consequences, and that help the ontology engineer to resolve inconsistencies and to remove unwanted consequences [30,21,20,18,5,26]. These approaches address the quality dimension of *soundness* of an ontology, both within itself (consistency) and w.r.t. the intended application domain (no unwanted consequences). In [4,3] Baader et. al. have considered a different quality dimension: *completeness*. They have provided a basis for formally well-founded techniques and tools that support the ontology engineer in checking whether an ontology contains all the relevant information about the application domain, and to extend the ontology appropriately if this is not the case.

Given an application domain and a DL ontology describing it, it is interesting to know whether the ontology contains all the relevant information about the domain:

- Are all the relevant constraints that hold between concepts in the domain captured by the TBox?
- Are all the relevant individuals existing in the domain represented in the ABox?

Such questions cannot be answered by an automated tool alone. Clearly, to check whether a given relationship between concepts—which does not follow from the TBox—holds in the domain, one needs to ask a domain expert, and the same is true for questions

regarding the existence of individuals not described in the ABox. The rôle of the automated tool is to ensure that the expert is asked as few questions as possible; in particular, she should not be asked trivial questions, i.e., questions that could actually be answered based on the represented knowledge. Answering a non-trivial question may require the expert to study the relevant literature in the application domain or even to discover new knowledge.

At the first glance, this task seems to be easily solvable by using the attribute exploration method of FCA. However for this setting one cannot use attribute exploration as it is. The main reason is the open-world semantics of DL ontologies in contrast to the closed-world semantics of formal contexts. In DLs, if we cannot deduce from a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$ that an individual $i$ is an instance of $C$, then we do not assume that $i$ is an instance of $\neg C$. Thus, our knowledge about the relationships between individuals and concepts is incomplete. In contrast, classical FCA and attribute exploration assume that the knowledge about objects is complete: a cross in row $g$ and column $m$ of a formal context means that object $g$ has attribute $m$, and the absence of a cross means that $g$ does not have $m$.

There has already been some work on how to extend FCA and attribute exploration from complete knowledge to the case of partial knowledge [12,8,16,17,9,28]. However, these works are based on assumptions that are different from the ones in the ontology completion setting. In particular, they assume that the expert cannot answer all queries and, as a consequence, the knowledge obtained after the exploration process may still be incomplete. In contrast, in the ontology completion setting the intention is to complete the ontology, i.e., in the end one wants to have complete knowledge about the relationships between concepts of the ontology. What may be incomplete is the description of individuals used during the exploration process.

### *4.1. Partial contexts*

In [4] Baader et. al. have introduced an extension of attribute exploration that can deal with partial knowledge, and have shown how it can be used to complete OWL ontologies. This extension is based on the notion of a partial object description and partial context.

**Definition 5** (Partial object description)**.** A *partial object description (pod)* is a tuple $(A, S)$ where $A, S \subseteq M$ are such that $A \cap S = \emptyset$. We call such a pod a *full object description (fod)* if $A \cup S = M$. A set of pods is called a *partial context* and a set of fods a *full context* . ◇

Note that in this definition, $A$ is the set of attributes that a partial object certainly has, and $S$ is the set of attributes that it certainly does not have. A partial context can be extended by either adding new pods or by extending existing pods.

**Definition 6** (Realizer)**.** We say that the pod $(A', S')$ *extends* the pod $(A, S)$, and write this as $(A, S) \leq (A', S')$, if $A \subseteq A'$ and $S \subseteq S'$. Similarly, we say that the partial context $\mathcal{K}'$ *extends* the partial context $\mathcal{K}$, and write this as $\mathcal{K} \leq \mathcal{K}'$, if every pod in $\mathcal{K}$ is extended by some pod in $\mathcal{K}'$. If $\overline{\mathcal{K}}$ is a full context and $\mathcal{K} \leq \overline{\mathcal{K}}$, then $\overline{\mathcal{K}}$ is called a *realizer* of $\mathcal{K}$. ◇

The notion of implications in formal contexts is extended to partial contexts as follows:

**Definition 7** (Implication in partial contexts)**.** Let $L, R \subseteq M$. The implication $L \to R$ is *refuted* by the pod $(A, S)$ if $L \subseteq A$ and $R \cap S \neq \emptyset$. It is *refuted* by the partial context $\mathcal{K}$ if it is refuted by at least one element of $\mathcal{K}$. The set of implications that are not refuted by a given partial context $\mathcal{K}$ is denoted by $Imp(\mathcal{K})$. The set of all fods that do not refute a given set of implications $\mathcal{L}$ is denoted by $Mod(\mathcal{L})$. $\diamond$

In the ontology completion setting it is assumed that the domain expert has access to a full context $\overline{\mathcal{K}}$ and thus can answer all implication questions w.r.t. $\overline{\mathcal{K}}$, though finding these answers may involve extensive literature study, or even proving new mathematical theorems or carrying out new experiments, etc. What is partial is the subcontext that the attribute exploration algorithm works with. The reason is that the initial context may be partial, and the same is true for the counterexamples that the expert provides for implications that do not hold in $\overline{\mathcal{K}}$.

The setting can be described in more detail as follows: We are given an initial (possibly empty) partial context $\mathcal{K}$, an initially empty set of implications $\mathcal{L}$, and a full context $\overline{\mathcal{K}}$ that is a realizer of $\mathcal{K}$. The expert answers implication questions "$L \to R$" w.r.t. the full context $\overline{\mathcal{K}}$. If she says "yes," then $\overline{\mathcal{K}}$ does not refute $L \to R$ (and thus $L \to R$ holds in the corresponding formal context $\overline{\mathcal{K}}$). The implication $L \to R$ is then added to $\mathcal{L}$. Otherwise, the expert extends the current context $\mathcal{K}$ such that the extended context refutes $L \to R$ and still has $\overline{\mathcal{K}}$ as a realizer. Consequently, the invariant $\mathcal{K} \leq \overline{\mathcal{K}} \subseteq Mod(\mathcal{L})$ will be satisfied. The aim is to enrich $\mathcal{K}$ and $\mathcal{L}$ such that

- eventually $\mathcal{L}$ is not only sound, but also complete for $Imp(\overline{\mathcal{K}})$,
- and $\mathcal{K}$ refutes all other implications (i.e., all the implications refuted by $\overline{\mathcal{K}}$).

As in the classical case, this should be achieved by asking as few questions as possible to the expert. Our approach is based on the notion of undecided implications.

**Definition 8** (Undecided implication)**.** Let $\mathcal{L}$ be a set of implications and $\mathcal{K}$ a partial context. An implication is called *undecided* w.r.t. $\mathcal{K}$ and $\mathcal{L}$ if it neither follows from $\mathcal{L}$ nor is refuted by $\mathcal{K}$. It is *decided* w.r.t. $\mathcal{K}$ and $\mathcal{L}$ if it is not undecided w.r.t. $\mathcal{K}$ and $\mathcal{L}$. $\diamond$

In principle, the attribute exploration algorithm on partial contexts tries to decide all undecided implications by either adding the implication to $\mathcal{L}$ or extending $\mathcal{K}$ such that it refutes the implication. If all implications are decided, then the goal is achieved.

**Proposition 1.** Assume that $\mathcal{K} \leq \overline{\mathcal{K}} \subseteq Mod(\mathcal{L})$ and that all implications are decided w.r.t. $\mathcal{K}$ and $\mathcal{L}$. Then $\mathcal{L}$ is complete for $Imp(\overline{\mathcal{K}})$ and $\mathcal{K}$ refutes all implications not belonging to $Imp(\overline{\mathcal{K}})$.

How can one find all undecided implications? Generating all possible implications and checking which ones are undecided is infeasible due to its computational cost. Instead it suffices to consider implications whose left-hand sides are $\mathcal{L}$-closed.

**Proposition 2.** Let $\mathcal{L}$ be a set of implications and $L \to R$ an implication. Then, $L \to R$ follows from $\mathcal{L}$ iff $\mathcal{L}(L) \to R$ follows from $\mathcal{L}$.

Given an $\mathcal{L}$-closed set $L$ as left-hand side, what kind of right-hand sides should be considered? Obviously, we need not consider right-hand sides $R$ for which the implication $L \to R$ is refuted by $\mathcal{K}$: such implications are already decided. The largest right-hand side $R$ such that $L \to R$ is not refuted by $\mathcal{K}$ can be computed as follows:

**Proposition 3.** For a given left-hand side $L$ and a partial context $\mathcal{K}$, the largest right-hand side is $\mathcal{K}(L) := M \setminus \bigcup\{S \mid (A,S) \in \mathcal{K}, L \subseteq A\}$ such that $L \to \mathcal{K}(L)$ is not refuted by $\mathcal{K}$.

In order to enumerate all left-hand sides, we can use the well-known approach from FCA for enumerating closed sets in the lectic order [11,13], which is defined as follows:

**Definition 9.** Assume that $M = \{m_1, \ldots, m_n\}$ and fix some linear order $m_1 < m_2 < \cdots m_n$ on $M$. The *lectic order* $<$ is defined as follows: for $m_i \in M$ and $A, B \subseteq M$ we define $A <_i B$ iff $m_i \in B \setminus A$ and $A \cap \{m_1, \ldots, m_{i-1}\} = B \cap \{m_1, \ldots, m_{i-1}\}$. The order $<$ is the union of the orders $<_i$.

Obviously, $<$ extends the strict subset order, and thus $\emptyset$ is the smallest and $M$ the largest set w.r.t. $<$.

**Proposition 4.** Given a set of implications $\mathcal{L}$ and an $\mathcal{L}$-closed set $A \subsetneq M$, the next $\mathcal{L}$-closed set following $A$ in the lectic order is $\mathcal{L}((A \cap \{m_1, \ldots, m_{j-1}\}) \cup \{m_j\})$ where $j$ is maximal such that $A <_j \mathcal{L}((A \cap \{m_1, \ldots, m_{j-1}\}) \cup \{m_j\}$.

If an implication is added because the expert has stated that it holds in $\overline{\mathcal{K}}$, then we can extend the current context $\mathcal{K}$ by closing the first component of every pod in $\mathcal{K}$ w.r.t. the new set of implications $\mathcal{L}$. In fact, $\mathcal{L} \subseteq Imp(\overline{\mathcal{K}})$ makes sure that the extended context is still realized by $\overline{\mathcal{K}}$. To allow for this and possible other ways of extending the partial context, the formulation of the algorithm just says that, in case an implication is added, the partial context can also be extended. Whenever an implication is not accepted by the expert, $\mathcal{K}$ will be extended to a context that refutes the implication and still has $\overline{\mathcal{K}}$ as a realizer. Based on these considerations, the attribute exploration algorithm for partial contexts is described in Algorithm 1.

### *4.2. Ontologies and partial contexts*

Given a consistent DL ontology $(\mathcal{T}, \mathcal{A})$, it is not difficult to see that any individual in $\mathcal{A}$ induces a partial object description. To be more precise, let $M$ be a finite set of concept descriptions. Any individual name $a$ occurring in $\mathcal{A}$ gives rise to the partial object description

$$pod_{\mathcal{T},\mathcal{A}}(a, M) := (A, S) \text{ where } \begin{array}{l} A := \{C \in M \mid \mathcal{T}, \mathcal{A} \models C(a)\} \text{ and} \\ S := \{C \in M \mid \mathcal{T}, \mathcal{A} \models \neg C(a)\}, \end{array}$$

and the whole ABox induces the partial context $\mathcal{K}_{\mathcal{T},\mathcal{A}}(M) := \{pod_{\mathcal{T},\mathcal{A}}(a, M) \mid a \text{ is an individual name occurring in } \mathcal{A}\}$. Similarly, any element $d \in \Delta^{\mathcal{I}}$ of an interpretation $\mathcal{I}$ gives rise to the full example

$$fod_{\mathcal{I}}(d, M) := (\overline{A}, \overline{S}) \text{ where } \begin{array}{l} \overline{A} := \{C \in M \mid d \in C^{\mathcal{I}}\} \text{ and} \\ \overline{S} := \{C \in M \mid d \in (\neg C)^{\mathcal{I}}\}, \end{array}$$

and the whole interpretation induces the full context $\mathcal{K}_{\mathcal{I}}(M) := \{fod_{\mathcal{I}}(d, M) \mid d \in \Delta^{\mathcal{I}}\}$. Note that $fod_{\mathcal{I}}(d, M)$ is indeed a fod since every $d \in \Delta^{\mathcal{I}}$ satisfies either $d \in C^{\mathcal{I}}$ or $d \in \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} = (\neg C)^{\mathcal{I}}$.

**Algorithm 1** Attribute exploration for partial contexts

**Initialization**
$\mathcal{K}$ {initial partial context, realized by the underlying full context $\overline{\mathcal{K}}$}
$\mathcal{L} := \emptyset$ {initial empty set of implications}
$P := \emptyset$ {lectically smallest $\mathcal{L}$-closed subset of $M$}
**while** $P \neq M$ **do**
  Compute $\mathcal{K}(P)$
  **if** $P \neq \mathcal{K}(P)$ **then** {$P \to \mathcal{K}(P)$ is undecided}
    Ask the expert if the undecided implication $P \to \mathcal{K}(P)$ is refuted by $\overline{\mathcal{K}}$
    **if** no **then** {$P \to \mathcal{K}(P)$ not refuted}
      $\mathcal{K} := \mathcal{K}'$ where $\mathcal{K}'$ is a partial context such that $\mathcal{K} \leq \mathcal{K}' \leq \overline{\mathcal{K}}$
      $\mathcal{L} := \mathcal{L} \cup \{P \to \mathcal{K}(P) \setminus P\}$
      $P :=$ next $\mathcal{L}$-closed left-handside (Definition 4)
    **else** {$P \to \mathcal{K}(P)$ refuted}
      Get a partial context $\mathcal{K}'$ from the expert such that $\mathcal{K} \leq \mathcal{K}' \leq \overline{\mathcal{K}}$ and $P \to \mathcal{K}(P)$ is refuted by $\mathcal{K}'$
      $\mathcal{K} := \mathcal{K}'$
    **end if**
  **else** {trivial implication}
    $P :=$ next $\mathcal{L}$-closed left-handside (Definition 4)
  **end if**
**end while**

The notion of refutation of an implication is transferred from partial (full) contexts to knowledge bases (interpretations) in the obvious way.

**Definition 10.** The implication $L \to R$ over the attributes $M$ is *refuted* by the knowledge base $(\mathcal{T}, \mathcal{A})$ if it is refuted by $\mathcal{K}_{\mathcal{T},\mathcal{A}}(M)$, and it is *refuted* by the interpretation $\mathcal{I}$ if it is refuted by $\mathcal{K}_{\mathcal{I}}(M)$. If an implication is not refuted by $\mathcal{I}$, then we say that it *holds in* $\mathcal{I}$. The set of implications over $M$ that hold in $\mathcal{I}$ is denoted by $Imp_M(\mathcal{I})$. In addition, we say that $L \to R$ *follows from* $\mathcal{T}$ if $\sqcap L \sqsubseteq_{\mathcal{T}} \sqcap R$, where $\sqcap L$ and $\sqcap R$ respectively stand for the conjunctions $\bigsqcap_{C \in L} C$ and $\bigsqcap_{D \in R} D$. $\diamond$

Obviously, $L \to R$ is refuted by $(\mathcal{T}, \mathcal{A})$ iff there is an individual name $a$ occurring in $\mathcal{A}$ such that $\mathcal{T}, \mathcal{A} \models C(a)$ for all $C \in L$ and $\mathcal{T}, \mathcal{A} \models \neg D(a)$ for some $D \in R$. Similarly, $L \to R$ is refuted by $\mathcal{I}$ iff there is an element $d \in \Delta^{\mathcal{I}}$ such that $d \in C^{\mathcal{I}}$ for all $C \in L$ and $d \notin D^{\mathcal{I}}$ for some $D \in R$. In addition, the implication $L \to R$ holds in $\mathcal{I}$ iff $(\sqcap L)^{\mathcal{I}} \subseteq (\sqcap R)^{\mathcal{I}}$. Given these, we are now ready to define what we mean by a completion of a DL knowledge base. Intuitively, the knowledge base is supposed to describe an intended model. For a fixed set $M$ of "interesting" concepts, the knowledge base is complete if it contains all the relevant knowledge about implications between these concepts. To be more precise, if an implication holds in the intended interpretation, then it should follow from the TBox, and if it does not hold in the intended interpretation, then the ABox should contain a counterexample. Based on the notions introduced in the previous subsection, this can formally be defined as follows.

**Definition 11.** Let $(\mathcal{T}, \mathcal{A})$ be a DL knowledge base, $M$ a finite set of concept descriptions, and $\mathcal{I}$ a model of $(\mathcal{T}, \mathcal{A})$. Then $(\mathcal{T}, \mathcal{A})$ is *$M$-complete* (or simply *complete* if $M$

is clear from the context) *w.r.t.* $\mathcal{I}$ if the following three statements are equivalent for all implications $L \to R$ over $M$:

1. $L \to R$ holds in $\mathcal{I}$;
2. $L \to R$ follows from $\mathcal{T}$;
3. $L \to R$ is not refuted by $(\mathcal{T}, \mathcal{A})$.

Let $(\mathcal{T}_0, \mathcal{A}_0)$ be a DL knowledge base that also has $\mathcal{I}$ as a model. Then $(\mathcal{T}, \mathcal{A})$ is a *completion* of $(\mathcal{T}_0, \mathcal{A}_0)$ if it is complete and extends $(\mathcal{T}_0, \mathcal{A}_0)$, i.e., $\mathcal{T}_0 \subseteq \mathcal{T}$ and $\mathcal{A}_0 \subseteq \mathcal{A}$. $\diamond$

In order to rephrase the definition of completeness, let us say that the element $d \in \Delta^{\mathcal{I}}$ of an interpretation $\mathcal{I}$ *satisfies* the subsumption statement $C \sqsubseteq D$ if $d \notin C^{\mathcal{I}}$ or $d \in D^{\mathcal{I}}$, and that $\mathcal{I}$ *satisfies* this statement if every element of $\Delta^{\mathcal{I}}$ satisfies it. In addition, let us call the individual name $a$ a *counterexample in* $(\mathcal{T}, \mathcal{A})$ to the subsumption statement $C \sqsubseteq D$ if $\mathcal{T}, \mathcal{A} \models C(a)$ and $\mathcal{T}, \mathcal{A} \models \neg D(a)$.

**Lemma 1.** The knowledge base $(\mathcal{T}, \mathcal{A})$ is complete w.r.t. its model $\mathcal{I}$ iff the following statements are equivalent for all subsets $L, R$ of $M$:

1. $\sqcap L \sqsubseteq \sqcap R$ is satisfied by $\mathcal{I}$;
2. $\sqcap L \sqsubseteq_{\mathcal{T}} \sqcap R$ holds;
3. $(\mathcal{T}, \mathcal{A})$ does not contain a counterexample to $\sqcap L \sqsubseteq \sqcap R$.

Using these, Algorithm 1 that works on partial contexts can be adapted for computing a completion of a given ontology $(\mathcal{T}_0, \mathcal{A}_0)$ w.r.t. a fixed model $\mathcal{I}$ of this ontology. It is assumed that the *expert* has enough information about this model to be able to answer questions of the form "Is $L \to R$ refuted by $\mathcal{I}$?". If the answer is "no," then $L \to R$ is added to the implication base computed by the algorithm. In addition, the GCI $\sqcap L \sqsubseteq \sqcap R$ is added to the TBox. Since $L \to R$ is not refuted by $\mathcal{I}$, the interpretation $\mathcal{I}$ is still a model of the new TBox obtained this way. If the answer is "yes," then the expert must extend the current ABox (by adding or refining assertions) such that the extended ABox refutes $L \to R$ and $\mathcal{I}$ is still a model of this ABox. In order to optimize this, before actually asking the expert whether the implication $L \to R$ is refuted by $\mathcal{I}$, we can first check whether $\sqcap L \sqsubseteq \sqcap R$ already follows from the current TBox. If this is the case, then we know that $L \to R$ cannot be refuted by $\mathcal{I}$.

The complexity of this algorithm is the same as the complexity of the classical attribute exploration algorithm [11,13]: in the worst case it is exponential in the number of attributes. Regarding the number of questions asked to the expert, as in the classical case the extended algorithm asks the minimum number of questions with positive answers. Our approach for completing ontologies applies to ontologies written in arbitrary DLs, provided that the language allows for at least conjunction and negation, the TBox formalism allows for GCIs, the ABox formalism allows for concept assertions, and the subsumption and the instance problem are decidable.

In [6] Baader and Sertkaya have addressed usability issues of an implementation of this method. They have improved the method in such a way that at any time during completion the expert can pause the process, see all of her previous answers or changes to the knowledge base, undo some of those changes, and continue completion. This improvement takes into account that the expert does not have to answer the same questions she has answered before pausing the process. The improved approach saves previous an-

swers, and uses them as background knowledge when the expert continues completion. Another wish of ontology engineers, namely postponing questions was solved by pausing completion, changing the order of attributes, and restarting the completion with previous answers as background knowledge. In theory, this method might not postpone a question, thus the expert might be asked the last question again. However, in [6] it was reported that in practice the method turned out to be useful in many cases when the expert was not able to answer a particular question and wanted to get another one. An implementation of the ontology completion method together with these usability issues is available as an open-source plugin for the Protégé ontology editor under the name *OntoComP*[1] [32].

## 5. Relational Exploration

The approach of Relational Exploration [28] is concerned with the aim of acquiring *all* GCIs valid in the domain of interest which are expressible within a certain DL. Typically the chosen DL is sub-Boolean, such as $\mathcal{EL}$, $\mathcal{FLE}$, or $\mathcal{ALE}$.[2] This approach goes beyond the procedure described in the preceding section where a finite set of "interesting" concepts is fixed a-priori. The problem that needs to be faced when trying this is that virtually all such DLs allow for infinitely many semantically different concept descriptions. Hence, applying the aforementioned techniques in a naive way would require to handle a context with infinitely many attributes, which is clearly not possible. Therefore, the idea is to perform exploration (as introduced in the previous section) repeatedly in several subsequent steps (denoted by $i = 0, 1, \ldots$) including more and more (but always finitely many) concepts. Thereby the information acquired in previous steps is taken into account not just by feeding it into the subsequent step but also by reducing the to-be-explored set of concepts through the removal of semantically equivalent concepts.

In relational exploration, this iteratively organized process will successively increment the considered concepts' maximal role depth.[3] Every single step of this procedure is subdivided into three phases: attribute generation, background knowledge explication, and (semi-)interactive exploration. In the sequel, we give a detailed account of these steps for $\mathcal{FLE}$ as the underlying description logic. We use $\mathcal{FLE}_i$ to denote the set of all $\mathcal{FLE}$ concepts with a role depth of at most $i$.

### 5.1. Attribute Generation

In this phase, we stipulate the attribute set $M_i \subseteq \mathcal{FLE}_i$ for exploration step $i$ based on the information collected in the previous exploration steps. If $i = 0$, we simply let $M_0$ be the set of all concept names plus the concept $\bot$. Otherwise, we use the implicational base $\mathcal{L}_{i-1}$ explored in the previous step in order to generate an empirically reduced set of attributes. The new set of attributes then comprises:

- all concept names as well as the $\bot$-concept,

[1] `http://ontocomp.googlecode.com`

[2] The reason for this is twofold: First, full Boolean DLs (like $\mathcal{ALC}$) would lead to exponentially larger attribute sets making the exploration process infeasible. Second, the hypothetical axioms generated by the exploration algorithm would be significantly more difficult to comprehend by the human expert.

[3] The *maximal role depth* of a concept is the maximal nesting of role restrictions occurring in that concept. E.g., the maximal role depth of $\exists marriedTo.\top \sqcap \exists hasChild.\forall hasToy.Pink$ is 2.

- for every concept $C \in M_{i-1}$, the all-quantified versions $\forall r.C$ for every role name $r$, and
- for every $\mathcal{L}_{i-1}$-closed set of concepts $\mathfrak{C} \subseteq M_{i-1}$ that does not contain $\bot$, the existentially quantified conjunction $\exists r.\bigsqcap \mathfrak{C}$ for every role name $r$.

It can be shown that this set is roughly spoken still sufficient to comprehensively "talk about" the considered domain in terms of $\mathcal{FLE}_i$.

### 5.2. Background Knowledge Explication

After having stipulated the attribute set for the current exploration step, we can determine the implications that can be added as a-priori knowledge. In order to do that, we exploit the previous implicational base $\mathcal{L}_{i-1}$, and determine the set of implications which can already be guaranteed to be valid before starting the exploration. Thereby, we use the functions $\overline{\mathsf{norm}}_i : \mathcal{FLE}_i \to 2^{\mathcal{FLE}_i}$ and $\mathsf{norm}_i : \mathcal{FLE}_i \to 2^{\mathcal{FLE}_i}$ recursively defined as follows:[4]

$$
\begin{aligned}
\mathsf{norm}_i(C) &= \mathcal{L}_i(\overline{\mathsf{norm}}_i(C)) \\
\overline{\mathsf{norm}}_i(C) &= \{C\} \text{ for } C \in M_0 \\
\overline{\mathsf{norm}}_i(\textstyle\bigsqcap \mathfrak{C}) &= \bigcup\{\overline{\mathsf{norm}}_i(A) \mid A \in \mathfrak{C}\} \\
\overline{\mathsf{norm}}_i(\forall r.A) &= \{\forall r.\tilde{A} \mid \tilde{A} \in \mathsf{norm}_{i-1}(A)\} \\
\overline{\mathsf{norm}}_i(\exists r.A) &= \begin{cases} \{\bot\} & \text{if } \bot \in \mathsf{norm}_{i-1}(A), \\ \{\exists r.\textstyle\bigsqcap \mathsf{norm}_{i-1}(A)\} & \text{otherwise.} \end{cases}
\end{aligned}
$$

The set of a-priori implications of step $i$ then contains the implications:

- $\{\bot\} \to M_i$,
- $\{\tilde{A} \mid \overline{\mathsf{norm}}_i(A) = \{\tilde{A}\}, A \in \mathfrak{A}\} \to \{\tilde{B} \mid \overline{\mathsf{norm}}_i(B) = \{\tilde{B}\}, B \in \mathfrak{B}\}$ for every implication $\mathfrak{A} \to \mathfrak{B}$ from $\mathcal{L}_{i-1}$,
- $\{\forall r.A \mid A \in \mathfrak{A}\} \to \{\forall r.B \mid B \in \mathfrak{B}\}$ for every implication $\mathfrak{A} \to \mathfrak{B}$ from $\mathcal{L}_{i-1}$,
- $\{\exists r.\bigsqcap \mathfrak{A}\} \to \{\exists r.\bigsqcap \mathfrak{B}\}$ for all $\mathcal{L}_{i-1}$-closed sets $\mathfrak{A}, \mathfrak{B} \subseteq M_{i-1}$ with $\mathfrak{A} \subsetneq \mathfrak{B}$ where there is no $\mathcal{L}_{i-1}$-closed set $\mathfrak{C}$ with $\mathfrak{A} \subsetneq \mathfrak{C} \subsetneq \mathfrak{B}$, and
- $\{\exists r.\bigsqcap \mathfrak{A}, \forall r.A\} \to \{\exists r.\bigsqcap \mathcal{L}_{i-1}(\mathfrak{A} \cup \{A\})\}$ for every concept $A \in M_{i-1}$ and every $\mathcal{L}_{i-1}$-closed set $\mathfrak{A} \subseteq M_{i-1} \setminus \{A\}$Baris.

In doing this, we deliver implicational knowledge that trivially follows from former exploration steps prior to engaging in the next interactive exploration phase. The "observable behavior" of the system (i.e., the questions asked to the human expert) would be the same without that preparation, since the used decision procedure would automatically answer questions concerning this kind of knowledge. However, providing this knowledge in advance obviously reduces the number of calls to the decision procedure, which are assumed to be costly.

### 5.3. Interactive Exploration

After all these preparations, the actual exploration process as described in Section 4 takes place on the attribute set $M_i$.

[4] Essentially these functions take an arbitrary $\mathcal{FLE}_i$ concept $C$ and convert it into a set of concepts from $M_i$ such that the conjunction over these and $C$ have the same extension in $\mathcal{I}$. This allows to express everything that can be expressed in via $\mathcal{FLE}_i$ by means of implications on the reduced concept set $M_i$.

At the end of this phase, we have an implicational base $\mathcal{L}_i$ for $M_i$ and thus a means to decide any GCI on $\mathcal{FLE}_i$ due to the following proposition:

**Proposition 5.** Let $A \in \mathcal{FLE}_i$ and let $\mathcal{I}$ be the explored interpretation. Then $A^{\mathcal{I}} = \left(\bigsqcap \overline{\mathsf{norm}}_i(A)\right)^{\mathcal{I}} = \left(\bigsqcap \mathsf{norm}_i(A)\right)^{\mathcal{I}}$. Consequently for any $\mathcal{FLE}_i$ GCI $C \sqsubseteq D$ we obtain $\mathcal{I} \models C \sqsubseteq D$ exactly if $\bigsqcap \overline{\mathsf{norm}}_i(C) \sqsubseteq \bigsqcap \overline{\mathsf{norm}}_i(D)$.

### *5.4. Termination*

Although the exploration process just described will have to be stopped after few steps in most practical cases due to the drastic increase of time costs, at least from the theoretical point of view the question emerges, whether and under which circumstances the proposed algorithm terminates, i.e., all information necessary to decide any $\mathcal{I}$-subsumption statement on $\mathcal{FLE}$ (of arbitrary role depth) has been acquired. It is clear, that termination can only be achieved, if the considered interpretation allows for a complete description by finitely many $\mathcal{FLE}$ statements as characterized in the following definition.

**Definition 12.** An interpretation $\mathcal{I}$ will be called *finitely $\mathcal{FLE}$-characterizable* if there is a finite set $\mathcal{F}$ of $\mathcal{FLE}$ GCIs such that for every $\mathcal{FLE}$ GCI $C \sqsubseteq D$ we have $\mathcal{I} \models C \sqsubseteq D$ if and only if $\mathcal{F} \models C \sqsubseteq D$.

It turns out that this property can indeed be efficiently checked and thus gives rise to a termination criterion.

**Proposition 6.** Let $\mathcal{I}$ be an interpretation and let $(\mathcal{L}_i)$ be the sequence of implicational bases iteratively explored as described before.

$\mathcal{I}$ is finitely $\mathcal{FLE}$-characterizable if there is an $n \in \mathbb{N}$ such that the mapping $F_n : \{\mathfrak{A}^{\mathcal{L}_n} \mid \mathfrak{A} \subseteq M_n\} \to \{\mathfrak{B}^{\mathcal{L}_{n+1}} \mid \mathfrak{B} \subseteq M_{n+1}\}$ with $F_n(\mathfrak{A}) := \mathsf{norm}_{n+1}(\bigsqcap \mathfrak{A})$ is a bijection between the $\mathcal{L}_n$-closed subsets from $M_n$ and the $\mathcal{L}_{n+1}$-closed subsets from $M_{n+1}$.

The reason why this criterion is valid is that it provides a way to "shrink" an $\mathcal{FLE}_{n+1}$ concept to maximal role depth $n$ preserving its semantics with respect to $\mathcal{I}$. Then – exploiting this fact – one can do even more: for any concept $C \in \mathcal{FLE}$ (i.e., of arbitrary role depth), we find an "empirically equivalent" concept $\tilde{C} \in \mathcal{FLE}_n$ by applying the function $\pi : \mathcal{FLE} \to \mathcal{FLE}_n$ with:

$$
\begin{aligned}
C &\mapsto C \text{ for all concept names } C \text{ or if } C = \bot \\
\exists r.C &\mapsto \begin{cases} \bigsqcap [\exists r](\mathsf{norm}_{n-1}(C)) \text{ if } \exists r.C \in \mathcal{FLE}_n, \\ \bigsqcap F_n^{-1}([\exists r](\mathsf{norm}_n(\pi(C))))^{\mathcal{L}_{n+1}} \text{ otherwise.} \end{cases} \\
\forall r.C &\mapsto \begin{cases} \bigsqcap [\forall r]\mathsf{norm}_{n-1}(C) \text{ if } \forall r.C \in \mathcal{FLE}_n, \\ \bigsqcap F_n^{-1}([\forall r]\mathsf{norm}_n(\pi(C)))^{\mathcal{L}_{n+1}} \text{ otherwise.} \end{cases} \\
\bigsqcap \mathfrak{C} &\mapsto \bigsqcap \{\pi(C) \mid C \in \mathfrak{C}\}.
\end{aligned}
$$

Thereby, for a set $\mathfrak{C}$ of concepts, we let $[\exists r]\mathfrak{C} = \exists r.\bigsqcap_{C \in \mathfrak{C}} C$ and $[\forall r]\mathfrak{C} = \bigsqcap_{C \in \mathfrak{C}} \forall r.C$.

In words, the $\pi$ function just realizes the following transformation: beginning from "inside" the concept expression $C$, subformulae having maximal role depth of $n+1$ are substituted by $\mathcal{I}$-equivalent ones with smaller role depth. When applied iteratively, this results in a concept $\tilde{C}$ from $\mathcal{FLE}_n$ that is $\mathcal{I}$-equivalent to the original one. The validity

of this concept can now be checked by the method described in the preceding section. Hence, we obtain the following proposition.

**Proposition 7.** Let $\mathcal{I}$ be a finitely $\mathcal{FLE}$-characterizable interpretation. Then for any $C \in \mathcal{FLE}$ we have and $\pi(C)^{\mathcal{I}} = C^{\mathcal{I}}$ and consequently for every $\mathcal{FLE}$ GCI $C \sqsubseteq D$ holds $\mathcal{I} \models C \sqsubseteq D$ exactly if $\mathcal{I} \models \pi(C) \sqsubseteq \pi(D)$.

It is easy to show that $\mathcal{I}$ is finitely $\mathcal{FLE}$-characterizable if $\Delta^{\mathcal{I}}$ is finite. That means, termination of the described algorithm is guaranteed whenever the described domain is finite. This seems to be a reasonable assumption for most practical scenarios.

## 6. Conclusion and Outlook

In the present chapter we have presented approaches to interactive ontology learning based on Formal Concept Analysis. By using information provided by a domain expert, our approaches learn the relationships between classes of the application domain and extend both the terminological and assertional parts of a Description Logic ontology accordingly. Our approaches provide basis for efficient and formally well-founded techniques and tools for interactive ontology learning. The use of techniques from Formal Concept Analysis ensures that, on the one hand, the interaction with the expert is kept to a minimum, and, on the other hand, it enables us to show that the knowledge in the resulting ontology is complete in a certain, well-defined sense.

Our approaches are based on the so-called attribute exploration method, which is an interactive knowledge acquisition method developed in Formal Concept Analysis. Starting from an approach called Ontology Exploration where attribute exploration was rather directly applied to generate ontological knowledge, here we have identified weaknesses of this approach and presented adaptations of the method more suitable in a Semantic Web setting. We have thus come up with a method capable of dealing with partial information, prior ontological knowledge and potentially infinite sets of concept descriptions. Due to space restrictions we were not able to give all details of these methods and proofs of our formal arguments. The proofs of our arguments in Section 4 can be found in [31], and the proofs of arguments in Section 5 can be found in [28].

It should be noted that the basic interpretation-to-context encoding presented in Section 3 is not the only possible one. Alternative encodings lead to methods accomplishing completeness with respect to other logical fragments such as first-order Horn rules [35] or an extended version of domain-range-restrictions [29]. In [1,2] an extension to the DL $\mathcal{EL}_{gfp}$, which extends $\mathcal{EL}$ by cyclic concept definitions with greatest fixpoint semantics, has been presented. Moreover it seems worthwhile to couple the presented techniques with complementary approaches to ontology learning such as NLP-based methods [34,33].

On a more general level, the knowledge acquisition approaches described here share their principled motivation – eliciting expert knowledge while minimizing the imposed workload – with other approaches to ontology refinement [10,25].

## References

[1] Franz Baader and Felix Distel. A finite basis for the set of EL-implications holding in a finite model. In Raoul Medina and Sergei Obiedkov, editors, *Proceedings of the 6th International Conference on Formal*

*Concept Analysis, (ICFCA 2008)*, volume 4933 of *Lecture Notes in Artificial Intelligence*, pages 46–61. Springer-Verlag, 2008.

[2] Franz Baader and Felix Distel. Exploring finite models in the description logic ELgfp. In Sébastien Ferré and Sebastian Rudolph, editors, *Proceedings of the 7th International Conference on Formal Concept Analysis, (ICFCA 2009)*, volume 5548 of *Lecture Notes in Artificial Intelligence*, pages 146–161. Springer-Verlag, 2009.

[3] Franz Baader, Bernhard Ganter, Ulrike Sattler, and Barış Sertkaya. Completing description logic knowledge bases using formal concept analysis. In *Proceedings of the Third International Workshop OWL: Experiences and Directions (OWLED 2007)*. CEUR-WS, 2007.

[4] Franz Baader, Bernhard Ganter, Barış Sertkaya, and Ulrike Sattler. Completing description logic knowledge bases using formal concept analysis. In Manuela M. Veloso, editor, *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI'07)*, pages 230–235. AAAI Press, 2007.

[5] Franz Baader and Rafael Peñaloza. Axiom pinpointing in general tableaux. *Journal of Logic and Computation*, 2010. To appear.

[6] Franz Baader and Barış Sertkaya. Usability issues in description logic knowledge base completion. In Sébastien Ferré and Sebastian Rudolph, editors, *Proceedings of the 7th International Conference on Formal Concept Analysis, (ICFCA 2009)*, volume 5548 of *Lecture Notes in Artificial Intelligence*, pages 1–21. Springer-Verlag, 2009.

[7] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American*, 284(5):34–43, 2001.

[8] Peter Burmeister and Richard Holzer. On the treatment of incomplete knowledge in formal concept analysis. In Bernhard Ganter and Guy W. Mineau, editors, *Proceedings of the 8th International Conference on Conceptual Structures, (ICCS 2000)*, volume 1867 of *Lecture Notes in Computer Science*, pages 385–398. Springer-Verlag, 2000.

[9] Peter Burmeister and Richard Holzer. Treating incomplete knowledge in formal concept analysis. In *Formal Concept Analysis*, volume 3626 of *Lecture Notes in Computer Science*, pages 114–126. Springer-Verlag, 2005.

[10] Philipp Cimiano, Sebastian Rudolph, and Helena Hartfiel. Computing intensional answers to questions - an inductive logic programming approach. *Data & Knowledge Engineering*, 69(3):261–278, 2010.

[11] Bernhard Ganter. Two basic algorithms in concept analysis. Technical Report Preprint-Nr. 831, Technische Hochschule Darmstadt, Darmstadt, Germany, 1984.

[12] Bernhard Ganter. Attribute exploration with background knowledge. *Theoretical Computer Science*, 217(2):215–233, 1999.

[13] Bernhard Ganter. Two basic algorithms in concept analysis. In Léonard Kwuida and Barış Sertkaya, editors, *Proceedings of the 8th International Conference on Formal Concept Analysis, (ICFCA 2010)*, volume 5986 of *Lecture Notes in Artificial Intelligence*, pages 329–359. Springer-Verlag, 2010. Reprint of [11].

[14] Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer-Verlag, Berlin, Germany, 1999.

[15] Jean-Louis Guigues and Vincent Duquenne. Familles minimales d'implications informatives resultant d'un tableau de données binaries. *Mathématiques, Informatique et Sciences Humaines*, 95:5–18, 1986.

[16] Richard Holzer. Knowledge acquisition under incomplete knowledge using methods from formal concept analysis: Part I. *Fundamenta Informaticae*, 63(1):17–39, 2004.

[17] Richard Holzer. Knowledge acquisition under incomplete knowledge using methods from formal concept analysis: Part II. *Fundamenta Informaticae*, 63(1):41–63, 2004.

[18] Matthew Horridge, Bijan Parsia, and Ulrike Sattler. Laconic and precise justifications in owl. In Amit P. Sheth, Steffen Staab, Mike Dean, Massimo Paolucci, Diana Maynard, Timothy W. Finin, and Krishnaprasad Thirunarayan, editors, *Proceedings of the 7th International Semantic Web Conference, (ISWC 2008)*, volume 5318 of *Lecture Notes in Computer Science*, pages 323–338. Springer-Verlag, 2008.

[19] Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From SHIQ and RDF to OWL: the making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26, 2003.

[20] Aditya Kalyanpur, Bijan Parsia, Matthew Horridge, and Evren Sirin. Finding all justifications of OWL DL entailments. In *Proceedings of the 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, (ISWC 2007 + ASWC 2007)*, volume 4825 of *Lecture Notes in Computer Science*, pages 267–280. Springer-Verlag, 2007.

[21] Aditya Kalyanpur, Bijan Parsia, Evren Sirin, and Bernardo Cuenca Grau. Repairing unsatisfiable con-

cepts in OWL ontologies. In York Sure and John Domingue, editors, *The Semantic Web: Research and Applications. Proceedings of the 3rd European Semantic Web Conference (ESWC 2006)*, volume 4011 of *Lecture Notes in Computer Science*, pages 170–184. Springer-Verlag, 2006.

[22] Aditya Kalyanpur, Bijan Parsia, Evren Sirin, Bernardo Cuenca Grau, and James A. Hendler. Swoop: A web ontology editing browser. *Journal of Web Semantics*, 4(2):144–153, 2006.

[23] Holger Knublauch, Ray W. Fergerson, Natalya Fridman Noy, and Mark A. Musen. The protégé OWL plugin: An open development environment for semantic web applications. In Sheila A. McIlraith, Dimitris Plexousakis, and Frank van Harmelen, editors, *Proceedings of the 3rd International Semantic Web Conference, (ISWC 2004)*, volume 3298 of *Lecture Notes in Computer Science*, pages 229–243. Springer-Verlag, 2004.

[24] Markus Krötzsch, Frantisek Simančík, and Ian Horrocks. A description logic primer. In Johanna Völker and Jens Lehmann, editors, *Perspectives of Ontology Learning*, Studies on the Semantic Web. AKA Heidelberg / IOS Press, 2012.

[25] Nadeschda Nikitina, Sebastian Rudolph, and Birte Glimm. Interactive ontology revision. *Web Semantics: Science, Services and Agents on the World Wide Web*, 12(0), 2012. in press.

[26] Rafael Peñaloza and Barış Sertkaya. On the complexity of axiom pinpointing in the $\mathcal{EL}$ family of Description Logics. In Fangzhen Lin, Ulrike Sattler, and Miroslaw Truszczynski, editors, *Proceedings of the Twelfth International Conference on Principles of Knowledge Representation and Reasoning, (KR 2010)*, pages 280,289. AAAI Press, 2010.

[27] Sebastian Rudolph. Exploring relational structures via $\mathcal{FLE}$. In Karl Erich Wolff, Heather D. Pfeiffer, and Harry S. Delugach, editors, *Proceedings of the 12th International Conference on Conceptual Structures (ICCS 2004)*, volume 3127 of *Lecture Notes in Computer Science*, pages 196–212. Springer-Verlag, 2004.

[28] Sebastian Rudolph. *Relational exploration: Combining Description Logics and Formal Concept Analysis for knowledge specification*. Ph.D. dissertation, Fakultät Mathematik und Naturwissenschaften, TU Dresden, Germany, 2006.

[29] Sebastian Rudolph. Acquiring generalized domain-range restrictions. In Raoul Medina and Sergei Obiedkov, editors, *Proceedings of the 6th International Conference on Formal Concept Analysis, (ICFCA 2008)*, volume 4933 of *Lecture Notes in Artificial Intelligence*, pages 32–45, 2008.

[30] Stefan Schlobach and Ronald Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In Georg Gottlob and Toby Walsh, editors, *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI'03)*, pages 355–362. Morgan Kaufmann, 2003.

[31] Barış Sertkaya. *Formal Concept Analysis Methods for Description Logics*. Ph.D. dissertation, Institute of Theoretical Computer Science, TU Dresden, Germany, 2007.

[32] Barış Sertkaya. Ontocomp: A protege plugin for completing owl ontologies. In Lora Aroyo and Paolo Traverso, editors, *Proceedings of the 6th European Semantic Web Conference, (ESWC 2009)*, volume 5554 of *Lecture Notes in Computer Science*, pages 898–902. Springer-Verlag, 2009.

[33] Johanna Völker and Sebastian Rudolph. Fostering web intelligence by semi-automatic owl ontology refinement. In *Proceedings of the 7th International Conference on Web Intelligence (WI)*, 2008.

[34] Johanna Völker and Sebastian Rudolph. Lexico-logical acquisition of owl dl axioms. In Raoul Medina and Sergei A. Obiedkov, editors, *Proceedings of the 6th International Conference on Formal Concept Analysis, (ICFCA 2008)*, volume 4933 of *Lecture Notes in Computer Science*, pages 62–77. Springer, 2008.

[35] Monika Zickwolff. *Rule Exploration: First Order Logic in Formal Concept Analysis*. Ph.D. dissertation, TH Darmstadt, Germany, 1991.

# Ontology Design Patterns in Ontology Learning

Eva BLOMQVIST [a,b,1], Aldo GANGEMI [b], and Francesco DRAICCHIO [b]
[a] *Dept. of Computer and Information Science, Linköping University*
[b] *STLab, ISTC-CNR*

**Abstract.** Ontology Learning (OL), either from text or other resources, suffers from problems related to implicit knowledge being incompletely extracted, knowledge that is too specific for creating a general domain model, and data or text structures that are not easily transformed into a high-quality ontology. Many types of background knowledge can be used to remedy problems of implicit or missing knowledge, however applying an appropriate modeling solution and style is a harder problem. Ontology Design Patterns (ODPs) are encodings of general best practices in ontology design. There are many kinds of patterns that can contribute to OL systems in different ways. In this chapter we first present the kinds of ODPs that exist and the kinds of problems they address, we then discuss how ODPs could potentially contribute to OL. Finally, we take two examples of research approaches where ODPs have been used for improving OL results or perform OL in novel ways. An overall conclusion is that ODPs can both provide guidance for OL, in terms of task-focus and making design choices available in a formalized manner, as well as being reusable building blocks that could potentially raise the focus of OL from single elements to detection and composition of larger complex components.

**Keywords.** Ontology Design Patterns, Best Practices, Ontology Refinement

## Introduction

In the era of the Semantic Web, ontologies need to be fast and easy to construct, understand and reuse. Ontology Learning (OL) plays a crucial role in facilitating ontology construction from existing resources. However, OL suffers from some inherent problems. The sources used for OL are rarely intended for this purpose, rather they are intended for human consumption, e.g., natural language texts, or for use by other types of software, e.g., databases and other data sources. When extracting ontologies, this introduces problems related to knowledge being implicitly expressed, knowledge that is too specific for creating a general domain model, and data or text structures that are not easily transformed into a high-quality ontology. Many types of background knowledge can be used to remedy problems of implicit or missing knowledge, however applying an appropriate modeling solution and style is a harder problem that has not been equally well researched.

[1]Corresponding Author: Eva Blomqvist, Linköping University, SE-581 83 Linköping, Sweden; E-mail: eva.blomqvist@liu.se or evabl444@gmail.com

Computational ontologies as they are used on the Semantic Web have a (primarily logical) structure, and must match both the *domain of interest* and the *task* of the application; they allow the description of entities whose classes, attributes and relations are of concern because of their relevance in a domain for some purpose, e.g. query, search, integration, matching, explanation, etc. For a long time, the main focus of OL has been on extracting domain entities with their classes (mainly) and relations, rather than establishing the structure of the ontology that would actually enable the solution of the intended task. This is a major distinction in evaluating ontologies; Gangemi et al. [24] presents the reason why existing ontologies tend to distribute across the dimensions of *coverage* and *task-fitness*. The task of a coverage-oriented ontology is mostly to cover as much of the domain as possible in order to annotate and retrieve domain documents, task-oriented ontologies have more specific tasks, e.g. answering specific queries, or performing a certain type of reasoning. Early methods for OL were mostly targeting coverage-oriented ontologies, while later approaches also try to address task-oriented ones. However, this introduces new problems into OL, since the logical structure of task-oriented ontologies is a major concern, and such ontologies have different types of requirements that need to be addressed, i.e. specific usage scenarios.

In this chapter, we focus on patterns for ontology construction [7,20,29], and in particular we explore some ideas of how they can be exploited in the context of OL. Under the assumption that there exist classes of problems that can be solved by applying common solutions (similarly to software engineering practices), Ontology Design Patterns (ODPs) support reusability on the design side specifically. Thereby, ODPs are encodings of general best practices in ontology design (ODPs are further described and defined in Sect. 2). ODPs can be analogous to software libraries when they have the form of small (or cleverly modularized) ontologies with explicit documentation of design rationales; they can be used as *building blocks* in ontology design. ODPs can also be analogous to software patterns, when they describe best practices for ontology engineering. There are several types of ODPs [29], and some are useful for solving OL problems, both when used as concrete building blocks or more abstract 'best practices'. They have the potential to improve OL result quality and help raise the focus of OL from single elements, to more complex composite structures solving certain tasks.

The chapter is organized as follows: In Section 1 we first discuss some challenges for state of art Ontology Learning; in Section 2 ODPs are discussed in more detail, and their usefulness to OL is addressed in Section 3. Finally, Section 4 describes two sample methods that reuse ODPs to improve OL results, or perform OL in new ways.

## 1. Ontology Learning Challenges

Blomqvist previously analyzed current approaches to OL [4], and singled out some challenges. One problem concerns how to determine the requirements (the task specification) of an ontology, to represent them in a form that can be automatically processed by an OL system, in order to select the appropriate input for the OL process. Many OL approaches work with whatever input is given by the user in a non-discriminative manner, without taking any particular requirement into account. Specially when constructing task-oriented ontologies, this is a clear shortcoming, since we do not know what task the ontology will be able to perform in the end, and what requirements it realizes.

OL from text also suffers from several problems related to the nature of text documents and natural language (NL) (cf. Maynard and Bontcheva [40]), e.g. inherent ambiguity of NL, and implicit knowledge. Interpretation of text documents usually requires a lot of knowledge from the reader, e.g. as Brewster et al. [9] notes writing a text could be seen as aiming towards 'knowledge evolution and maintenance' rather than specifying and defining knowledge from scratch.

In response to these challenges, some approaches try to use special resources instead of arbitrary text documents, e.g. dictionary entries [57]; however that is only feasible in presence of those special resources. The most common way to address the problem is by using some form of background knowledge to 'fill the gaps', e.g. general-purpose lexical resources such as WordNet [1] or FrameNet [2], or reusing existing general-purpose (foundational, upper-level or core) or domain-specific ontologies. However, reuse of background knowledge sometimes adds noise to the process. For example, we cannot safely assume that lexical resources have a direct formal semantics [27,21], or that existing ontologies are appropriate to the requirements either.

Extracting ontological elements, such as terms and relations, have been the subject of research for a long time and is described in previous chapters of this book (e.g. Suchanek [51], this volume), nevertheless, result quality is still a challenge. Forming appropriate logical definitions, i.e. axiomatizing those elements, is an even more open issue. In particular, the composition of the elements – how the pieces fit together – is a challenge, and often ontologies resulting from OL are quite shallow and 'light-weight'. So far, few approaches include semi-automatic ways to refine, enrich, or improve a draft ontology [4,3].

On the other hand, an ontology built from OL methods can be used as input to a manual ontology engineering process, so entering a post-processing method. Additionally, we can also see OL as semi-automatic methods within an overall design methodology. Since design methodologies include users, the output of OL methods should then be highly understandable and comprehensive from a user perspective, and facilitate their design work. Blomqvist [4] argued that some early OL techniques actually might not improve on the corresponding manual ontology engineering tasks: reading text documents, collecting and defining ontological elements by hand, etc. Unfortunately, most of the ontologies built through those early OL systems had no indication of the semantics of the extracted elements, making the post-processing step very challenging for the user.

Another important problem in OL is the low complexity of formal structures that can be extracted, typically named entities, type assertions, disjointness axioms, some domain relations, some constraints on binary relations, etc. As described by Coppola et al. [13] (this volume), relations with arities >2 are very difficult to extract from natural language, but those relations are at the core of cognitive understanding [12,21,22]. Indeed good examples of task-oriented ontologies contain a lot of reified n-ary relations, e.g. for dealing with events and more abstract entities such as plans, diagnoses, norms, etc.

In summary, currently OL suffers from lack of reliable methods for producing high-quality and highly axiomatized ontologies from existing sources, as opposed to simply extracting isolated elements and producing light-weight representations of them. Background knowledge is needed, but to avoid the reuse problems that are experienced when reusing large monolithic ontologies, we propose to use small, task-centered ODPs instead. ODPs can address challenges both on the side of logical axiomatization and lack

of explicit mentions of common sense structures in input sources, at the same time addressing some quality problems experienced in OL (see Section 4).

## 2. Ontology Design Patterns

Throughout experiences in ontology engineering projects[2] as well as in other ongoing international projects that have experimented with these ideas, typical conceptual patterns have emerged out of different domains, for different tasks, and while working with experts having heterogeneous backgrounds. For example, a simple pattern we can call *participation* (including objects taking part in events) emerges in domain ontologies as different as describing enterprise models [32], legal norms [28], software management [42], biochemical pathways [23], and fishery techniques [26]. Other, more complex patterns have also emerged in the same disparate domains. Similarly, the success of very simple and small ontologies like FOAF [10] and SKOS [43] shows the potential of really portable, or 'sustainable' ontologies. These lessons learnt support a new approach to ontology design. However, ODPs are not limited to being reusable *building blocks*, there exist many different types of ODPs, and one of their main aims is to encode modeling best practices.

In principle, ODPs do not depend on any specific representation language[3]. In the example section of this chapter we will focus mainly on Content ODPs (CPs, aka *knowledge patterns*), as defined towards the end of Section 2.1 and described further in Section 2.2, and in order to provide the reader with concrete examples and a closer view on their exploitation on the Semantic Web, and in OL, we have decided to refer to OWL CPs throughout the chapter (for details on OWL and its relation to DLs, see Krötzsch et al. [37] in this volume).

### 2.1. Types of Ontology Design Patterns

An ODP is a modeling solution to solve a recurrent ontology design problem, and ideally it encodes some best practice in the field. Several types of ODPs have been identified, and in [29] they are grouped into six families: *Structural* ODPs, *Correspondence* ODPs, *Content* ODPs (CPs), *Reasoning* ODPs, *Presentation* ODPs, and *Lexico-Syntactic* ODPs. In this section we give an overview of the ODP families, with some examples. For more details, the reader is referred to work by Gangemi and Presutti [29]. Ideas on how these relate to challenges in OL are given in Section 3.

***Structural ODPs*** Structural ODPs include Logical ODPs and Architectural ODPs. Logical ODPs are compositions of logical constructs that solve a problem of expressivity, while Architectural ODPs affect the overall shape of the ontology either internally or externally.

*Logical ODPs* are only expressed in terms of a logical vocabulary, because their signature (the set of predicate names, e.g. the set of classes and properties in an OWL ontology, c.f. Krötzsch et al [37] in this volume, is empty (with minor exceptions, e.g. the

[2]For example, in the projects *FOS*: http://www.fao.org/agris/aos/, *WonderWeb*: http://wonderweb.semanticweb.org, *Metokis*: http://metokis.salzburgresearch.at, *NeOn*: http://www.neon-project.org, and *IKS*: http://www.iks-project.eu/

[3]With the exception of Logical ODPs.

default inclusion of `owl:Thing` in OWL). Logical ODPs help to solve design problems where the primitives of the representation language do not directly support certain logical constructs. For example, if the representation language is OWL, and a designer needs to represent a relation between more than two elements, a Logical ODP is needed in order to express an n-ary relation semantics by using classes and binary relation primitives. Logical ODPs were one of the first kind of patterns to be discussed for the Semantic Web, e.g. by the W3C OEP task force [19]. More examples can be found in various online portals [19,49,53].

*Architectural ODPs* affect the overall shape of the ontology. They can be of two types: (i) *internal*, defined in terms of collections of Logical ODPs that have to be exclusively employed when designing an ontology e.g., an OWL profile or the varieties of description logics; (ii) *external*, defined in terms of meta-level constructs e.g., the *modular architecture* consists of an ontology network, where the involved ontologies play the role of modules.

***Reasoning ODPs*** Reasoning ODPs are applications of Logical ODPs oriented to obtain certain reasoning results, based on the behavior implemented in a reasoning engine. Examples of Reasoning ODPs include: *classification*, *subsumption*, *inheritance*, *materialization*, *de-anonymizing*, etc. Reasoning ODPs, when declared on top of an ontology, inform about the state of that ontology with respect to carrying out queries, evaluation, etc. Examples of more complex Reasoning ODPs have been described by van Harmelen et al. [54], including patterns for semantic search, personalization etc.

***Correspondence ODPs*** Correspondence ODPs include Reengineering ODPs and Alignment ODPs. Reengineering ODPs provide designers with solutions to the problem of transforming a conceptual model, which can even be a non-ontological resource, into a new ontology, hence they are closely related to the OL problem. Alignment ODPs are patterns for creating semantic associations between two existing ontologies.

*Reengineering ODPs* are transformation rules applied in order to create a new ontology (*target* model) starting from elements of a *source* model. The target model is an ontology, while the source model can be either an ontology, or a non-ontological resource e.g., a thesaurus, a data model, a UML model, a linguistic structure, etc. Reengineering ODPs are described in terms of metamodel transformation rules. Such transformation rules are commonly used in OL for identifying ontological elements in linguistic structures, and transforming them into ontological elements (see also Lexico-syntactic ODPs below), examples can be found in the ODP Portal [50]. Other patterns have been proposed for, for instance, transforming XML or relational databases into OWL/RDF, i.e. so-called triplification, but so far only a few triplifiers express their transformation rules as reusable patterns (e.g. Semion works in this direction [46]). Finally, transformation ODPs are also defined for refactoring ontologies, e.g. transformations between alternate modelling styles or solutions, as exemplified by Šváb-Zamazal et al. [58]. *Alignment ODPs* refer to semantic relations between elements in two or more ontologies. There are three basic semantic relations that are commonly used for mapping assertions: *equivalence*, *containment*, and *overlap*. Alignment ODPs provide designers with solutions to relate two ontologies without changing the logical types (e.g. `owl:Class`) of the ontology elements involved.

***Presentation ODPs*** Presentation ODPs deal with usability and readability of ontologies from a user perspective. They are meant as best practices that support the reuse of ontologies by facilitating their evaluation and selection. Examples are Naming ODPs and Annotation ODPs . The former are conventions about how to create names for namespaces, files, and ontology elements in general (classes, properties, etc.) [52]. Annotation ODPs provide annotation properties or annotation property schemas that can be used in order to improve the understandability of ontologies and their elements.

***Lexico-syntactic ODPs*** Lexico-syntactic ODPs are linguistic structures or schemas that consist of certain types of words following a specific order, and that permit to generalize and extract some conclusions about the meaning they express. They are also useful for associating simple Logical and Content ODPs with natural language sentences. A simple example is the Hearst patterns [34] that are commonly used in OL for detecting hints of ontological elements and axioms in natural language. An extensive discussion of lexico-syntactic patterns in linguistics can be found in Maynard and Bontcheva [40], Section 5, and a catalogue with examples is available in the ODP Portal [48].

***Content Ontology Design Patterns (CPs)*** CPs encode *conceptual*, rather than *logical* design patterns. In other words, while Logical ODPs solve design problems independently of a particular conceptualization, CPs propose patterns for solving design problems for the domain classes and properties that populate an ontology, therefore addressing *content* problems [20]. CPs are instantiations of Logical ODPs (or of compositions of Logical ODPs), featuring a non-empty signature. In principle, CPs do not depend on any specific language, however in order to reuse them as *building blocks*, they have to be implemented in some way. In the context of this chapter, we deal with CPs in a Semantic Web context. Hence, we use OWL as a reference formalism for representation.

### *2.2. Content ODPs*

CPs are useful because they provide solutions to domain-oriented problems, and are directly reusable. On one hand, CPs are comparable to software engineering (SE) design patterns for what concerns the way they are documented and communicated. On the other hand, the intuition behind their usage is analogous to that of software engineering (object oriented) reusable libraries, e.g. Java libraries. Section 2.2.1 lists the characteristics that differentiate CPs as special ontologies, and in Section 2.2.2 we describe two CP examples (further examples can be found in the ODP Portal [47]).

In order to document ODPs we use a similar approach to Software Engineering (SE) patterns. The mainstream approach for describing SE patterns is to use a template, although there is no standard format. Each CP is associated with a *catalogue entry* including a number of information fields (full list of fields can be seen in the ODP Portal [47]): **Name** provides a name for the pattern; **Intent** describes the problem (use case) addressed by the pattern; **Competency questions** contains examples of competency questions that the knowledge base associated with the CP can address; **Diagram** depicts a diagram representing the pattern; **Consequences** provides a description of the benefits and/or possible trade-offs when using the pattern; **Known uses** gives examples of realistic ontologies where the pattern is used; **Building block** provides references to implementations of the pattern. In the case of CPs for Semantic Web ontologies, the latter provides the URI of an OWL file (containing an implementation of the pattern).

### 2.2.1. Characteristics of CPs

CPs are reusable solutions to recurrent modeling problems. As known since a long time in conceptual modeling and knowledge engineering, these problems have two components; a domain and a use case (i.e. task). The same domain can have many use cases, and the same use case can be found in different domains. Ontologies are usually considered as models of a domain, but their use case is usually not explicitly described. As reusable solutions, CPs must explicitly encode a 'Generic Use Case' (GUC), i.e. a generalization of use cases that can be provided as examples of tasks that are supported by a particular CP. GUC can in many cases be captured by means of *competency questions* [32]. A competency question is a typical query that an expert might want to submit to a knowledge base of its target domain, for a certain task. In principle, an accurate domain ontology should specify *all and only* the conceptualizations required in order to answer all the competency questions formulated by, or acquired from, experts. Also note that competency questions may range from simple 'look-up queries', to complex questions requiring several inference steps.

Theoretically speaking, CPs are language-independent, and could be encoded in a higher-order representation language. Nevertheless, pragmatically their (sample) representation in OWL is essential in order to (re)use them as building blocks on the Semantic Web. Regardless of the particular way a CP has been created, it is a *small, autonomous* ontology. Smallness (typically two to ten classes with relations defined between them) and autonomy facilitate composing CPs, and enable them to govern the complexity of the whole ontology. There are combinations of ontology elements that do not allow any useful inference, e.g., a taxonomy with only two sibling classes does not allow for any inferences until the taxonomy is extended with super- or subclasses, or other axioms, similarly an object property alone does not allow for any useful inferences until it is used in some axiom, etc. Hence, a CP should be an *inference-enabling component*, in the sense that it allows some form of inference, e.g. a taxonomy with two sibling disjoint classes, a property with explicit domain and range set, a property and a class with a universal restriction on that property, etc. This can be a particularly important feature for ontologies resulting from OL, which are commonly 'light-weight' and sometimes lacking axiomatizations. Furthermore, a CP can be an element in a partial order, i.e. a hierarchy of CPs, where the ordering relation requires that at least one of the classes or properties in the pattern is specialized. Many CPs nicely match linguistic patterns called *frames*. A frame can be described as a lexically founded ODP, i.e. CPs are *linguistically relevant components*. The richest repository of frames is FrameNet [2]. Frames can be used for validating CPs with respect to lexical coverage, for lexicalizing them, and can be reengineered as CPs (as described by Coppola et al. [13] in this volume, and in [45], where Cps are generalized to the notion of *knowledge patterns* for a complete porting of FrameNet to RDF-OWL). Finally, a CP should be used to describe a 'best practice' of modelling, i.e. they are *formalizations of best practices*. Empirical evidence of being a best practice, in terms of objective and subjective improvement of ontology quality, has been described in [5,6].

### 2.2.2. Sample CPs

In this section we show two CPs taken from the ODP Portal[4], Each CP is presented in a catalogue-like way, and with reference to the OWL language. For space reasons, we describe each CP with a simplified catalogue entry composed of: the *name*, the *intent*, *competency questions*, some *examples* of its application, the *diagram* describing its structure, the *elements* and the role they play in the pattern. We have used TopBraid Composer[5] in order to produce the OWL encoding. With the same tool, we automatically generated a diagrammatical visualization based on a UML profile for OWL. UML classes (boxes) are used in order to depict OWL classes. When a class name is preceded by a prefix, e.g. `sit:`, it is interpreted as a class imported (e.g. by `owl:imports`) from another (typically more general) CP that is indexed by means of that prefix.

***The information realization CP*** The *information realization* CP[6] is extracted from the Dolce+DnS Ultra Lite ontology[7], and represents the relations between information objects, such as poems, songs, computer programs, etc., and their physical realizations, such as printed books, recorded tracks, physical files, etc.

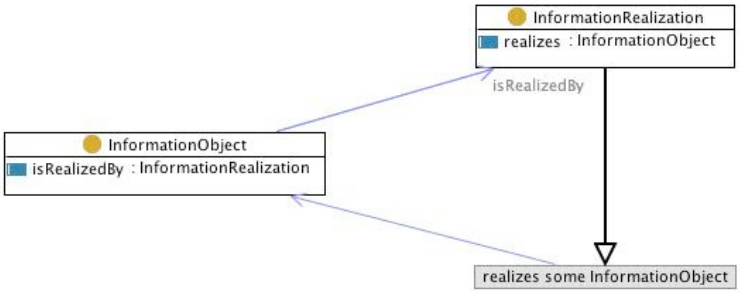



**Figure 1.** The information realization CP's graphical representation in UML.

The *information realization CP* is associated with information according to the catalogue entry fields reported below:
**Intent:** To represent relations between information objects and their physical realization.
**Competency questions:** Which physical object realizes a certain information object? Which information object is realized by a certain physical object?
**Diagram:** Figure 1 shows a UML diagram of the information realization CP.
**Elements:**

- `InformationObject`: A piece of information, such as a musical composition or a text, independently from how it is concretely realized.
- `InformationRealization`: A concrete realization of an InformationObject, e.g. the written document containing the text of a law.
- `realizes/isRealizedBy`: The relation between an information realization and an information object, e.g. the paper copy of the Italian Constitution realizes the text of the Constitution.

[4]http://www.ontologydesignpatterns.org
[5]http://www.topbraidcomposer.com/
[6]http://www.ontologydesignpatterns.org/cp/owl/informationrealization.owl
[7]http://www.ontologydesignpatterns.org/ont/dul//DUL.owl

***The Time Indexed Person Role CP*** The *time indexed person role*[8] is a CP that represents time indexing for the relation between persons and roles they play, e.g., George W. Bush was the president of the United States in 2007. This CP is also extracted from the Dolce+DnS Ultra Lite ontology. According to its associated catalogue entry, the main information associated with this CP is the following:
**Intent:** To represent time indexing for the relation between persons and roles they play.
**Competency questions:** Who was playing a certain role during a given time interval? When did a certain person play a specific role?
**Diagram:** An illustration of the most specific part of the CP can be seen in Figure 2.
**Elements:** (Selection of most specific elements realizing pattern functionality.)

- `Person`: The commonsense notion, i.e. either physical or social persons.
- `Role`: A concept that classifies a person
- `TimeInterval`: Any region in a dimensional space representing time.
- `TimeIndexedPersonRole`: A situation that expresses time indexing for the relation between persons and roles they play.
- `hasRole/isRoleOf`: A relation between a Role and an Entity, e.g. 'John is considered a typical rude man'; your last concert constitutes the achievement of a lifetime; '20-year-old means she's mature enough'. The two properties are the inverse of each other.
- `isSettingFor/hasSetting`: A relation between time indexed role situations and related entities, e.g. 'I was the director between 2000 and 2005 ', i.e.: the situation in which I was a director is the setting for the role of director, me, and the time interval. The two properties are the inverse of each other.

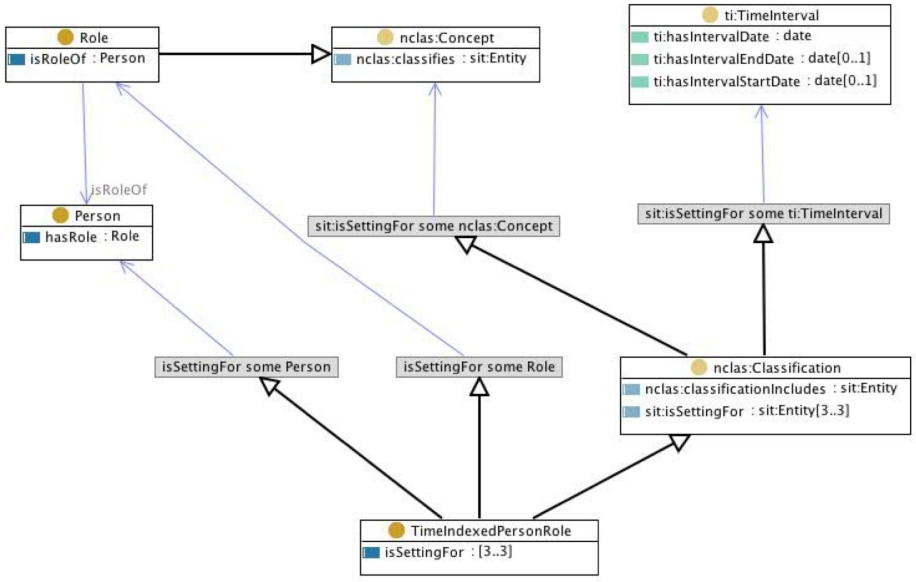



**Figure 2.** The time indexed person role CP's graphical representation in UML.

[8] http://ontologydesignpatterns.org/cp/owl/timeindexedpersonrole.owl

## 3. Applications of ODPs in OL

Different types of ODPs have been used to support OL, as already mentioned in connection with the ODP types. In this section we exemplify some of those usages, and in other cases describe ideas towards new usage areas of ODPs in OL.

Lexico-syntactic ODPs (LOPs) are probably the most well-known in the OL community, and have been used for OL since the early times. Maedche used Hearst-type LOPs in order to extract taxonomic relations from text corpora [39], which has also been picked up in subsequent work, e.g. by Cimiano [11]. Völker et al. introduce LOPs specific to dictionary entries [56], in order to extract more expressive axioms. However, none of these LOPs were at the time described and collected in catalogues, for others to reuse. Such catalogues are currently emerging, e.g., as in the ODP Portal [48].

When LOPs are explicitly described, and they are linked to Logical or Content ODPs, they can be considered as Reengineering ODPs, since they provide methods and rules to transform linguistic structures into formal ones, e.g. a natural language sentence into an OWL model. An example of using LOPs to enrich CPs, and applying them to OL, was given by Nikitina et al. [44]. Other examples include work by Mainard et al. [41] and Aguado de Cea et al. [16]. We also envision future developments of using more complex LOPs, e.g. based on lexical frames, where frame detection (as described by Coppola et al. [12]) can then be used together with Content ODPs corresponding to the lexical frames, in order to construct ontologies from text. More details on using lexical frames for OL have been discussed by Coppola et al. [13] (this volume).

When moving to non-linguistic resources, we find that RDF triplifiers and other reengineering tools have gained a lot of momentum, therefore Reengineering ODPs are going to play a central role in OL. For example, with tools like Semion [46] an ontology designer can select some input and a set of Reengineering ODPs (so-called *recipes*, written in a dedicated SPARQL-based rule language), and receive an ontology as output. Semion has been used also to transform FrameNet into RDF linked data, and to investigate appropriate semantics for using semantic frames as ontologies [45]. Ontology evolution and refinement based on Reengineering ODPs for model refactoring has also been explored [58,59]. The process is semi-automatic, in the sense that the user has to select the input and ODPs, but detection and transformation are performed automatically.

One of the main shortcomings of ontologies resulting from OL is their low usability from a user perspective, e.g. their lack of documentation and axiomatization that best represent the intended meaning of the learnt concepts and relations. For providing more user-friendly ontologies, Presentation ODPs could be applied. So far we are not aware of any attempt to explicitly use Presentation ODPs in OL, but as a minimum some form of naming conventions is usually applied to the generated concepts and relations. Actually, some early OL approaches used to produce so-called *glosses* as natural language explanations of the concepts extracted [55]. A prerequisite for pattern-based OL developing this direction further is to collect and formally describe a larger set of Presentation ODPs.

Logical ODPs, and their corresponding *anti-patterns*, i.e. common mistakes that can be resolved by applying a (positive) Logical ODP, have been used in approaches to error detection and refactoring. These are not OL-specific, but since many ontologies are highly error-prone, it could be a fruitful direction to apply automatic error detection as a post-processing step in OL. Anti-pattern detection has been proposed, and applied for manual debugging of inconsistent ontologies [14,17], where so called inconsistency

patterns represent the anti-patterns that need to be avoided; solutions are provided as alternative patterns (i.e. alternative Logical ODPs that represent possible solutions).

So far, CPs have not been extensively utilized in OL, despite the fact that they are available as actual building blocks implemented in OWL. To the best of our knowledge, one of the few approach attempting to use CPs for OL is the OntoCase framework (described more in depth in the following section). In this approach CPs are applied as a post-processing step for improving ontology quality, rather than directly using the CPs in the OL process. Current developments improve on this by directly trying to detect or discover CPs in input sources, which can be facilitated by connecting CPs to lexical resources, e.g. lexical frames to be used in frame detection as mentioned above. A detection approach and experience is presented by Coppola et al. [13] (this volume). Another very recent one (described in section 4.2) is the FRED API, which evolves an implementation [8] of DRT (Discourse Representation Theory [35]) for deep parsing of text, by adding frame detection and transformation of DRT structures (DRS) into OWL-RDF ontologies, by making use of ODPs. Another unexplored direction addresses the problem of taking task-related requirements into account when generating ontologies. CPs are annotated with information about the tasks they support, i.e. through competency questions. Assuming that similar, but more specific, task-requirements exist for the ontology to be built, it still remains to be explored how those requirements can be automatically matched for selecting and specializing the *right* CPs for the task.

As a *vision level* recap of this overview, our impression is that we are moving to a broader, integrating scenario where knowledge derived from multiple heterogeneous sources aggregates around 'frameworks of reference', close to CPs in scope. The tendency to build more sense out of the huge Linked Data Cloud is the most visible manifestation of that. As an example, in their visionary article Gangemi and Presutti [30] present *knowledge patterns* as hubs of formal knowledge, linguistic data, structured data, rules and reasoning recipes, and other associated content. Knowledge patterns can act as an abstraction that facilitates projects aiming at a 'global accelerator of knowledge',[9] where knowledge analytics and business intelligence meet the results of heterogeneous information integration.

## 4. Using ODPs in OL - Sample Methods

To illustrate how ODPs in general, and CPs in particular might be used for OL, in this section we describe two sample methods. One, the OntoCase method in Section 4.1, contains initial work on applying CPs in a post-processing step, to improve OL results, and the other (see Section 4.2) describes how ODPs can be used together with linguistic resources to 'translate' from NL to complex ontological structures in OWL. In Section 1 some challenges in OL were discussed, e.g. that typically, learnt ontologies are quite light-weight and lack axiomatization, due to knowledge being implicit in text, and other problems. One of the main objectives of both these sample methods was to add some of this background knowledge by means of ODPs and produce a higher degree of axiomatization. Such patterns can also assist in structuring the existing learnt knowledge, by introducing best practice modeling solutions, so that the resulting ontologies become more intuitive and easily understandable.

[9] Cf. http://www.futurict.ethz.ch/FuturICTFlagship

### *4.1. The OntoCase Method*

The OntoCase framework, as introduced by Blomqvist [4,3], proposed a particular method for enriching learnt ontologies, by means of CPs. Experiments showed the two main features of this method: 1) The ability to add a (taxonomical) top structure to the ontology, representing background knowledge possibly implicit in the texts that were the basis of the input ontology. 2) The ability to add more structure to the learnt ontology, i.e., to increase the taxonomical depth and the relation-to-concept ratio.

As input to the method an initial draft ontology extracted by some OL tool (below referred to as the input ontology), and a catalogue of CPs are assumed. The OWL building block itself is used for retrieval and reuse of CPs. For the matching procedure OntoCase draws on ontology matching research. Primarily, terms of the learnt ontology and the CPs respectively are used, i.e., local names or labels. When matching properties, the domains and ranges of the properties are also used (if present). One important feature of CPs are their generality, i.e., they are general and reusable in many contexts. OntoCase attempts to bridge the abstraction gap by extending the direct matching of concept and relation names to generalization/specialization. This is done by using some background knowledge, e.g., WordNet in the current implementation, to find specialization/generalization relations based on hyponym/hypernym relations of the involved terms. Finally, OntoCase uses a ranking scheme to weight all this evidence of matches between the learnt ontology and a certain CP, in order to assess not only the correctness of the match but also the usefulness of incorporating the CP in the ontology (in terms of increased axiomatization etc.). Details on the method have been described previously by Blomqvist [4].

After selecting the CPs to incorporate, the reuse phase is concerned with cloning and adapting, i.e., automatically specializing, and composing the selected CPs, and integrating them into an enriched version of the input ontology. Different sets of heuristics can be used, to either include the complete CP and input ontology, or to prune those parts that did not have any match. For the experiments below, a set of heuristics [4] for maximizing the axiomatization of the resulting ontology was used.

#### *4.1.1. OntoCase Evaluation Results*

Ontology evaluation methods were used for studying the quality and characteristics of the output of OntoCase. Gangemi et al. describe an overall framework for ontology evaluation [25], consisting of three levels; structural, functional, and usability evaluations. Structural evaluations analyze the quality of the syntax and semantics of the ontology as it is represented. Functional evaluations analyze how well the ontology conforms to the intended conceptualization, i.e. the requirements. Usability evaluations concern the understandability and reusability of the ontology, as well as user satisfaction. Since these are aspects and measures for ontology evaluation, clearly they can also be used to evaluate the output of OL.

The OntoCase method was evaluated in three independent settings; the SEMCO project's requirements engineering ontology, the JIBSNet university intranet ontology, and a set of agricultural ontologies of the FAO. The structural level of the ontologies was analyzed within all experiments, based on measures such as number of concepts, number of concepts at the top level (i.e. root concepts, with no other superclass but `owl:Thing`), number of subclass axioms and properties, and average depth of the taxonomy, as suggested by both Gangemi et al. [25] and Yao et al. [60]. Two separate ap-

proaches for taxonomic evaluation [31,33], were used when feasible. For the last two experiments, a sample of the elements were evaluated due to the size of the ontologies.

To evaluate functional and usability characteristics, i.e. the content of the ontologies, a subset of the OntoMetric framework [38] (only the dimension *content*) was used in the SEMCO experiment. In the JIBSNet and FAO cases the evaluation was performed using a random sample of classes and properties, whereby the same factors were not applicable. Instead individual assessment of the concept and property sample was done by domain experts (or ontology engineers in the FAO case). Through a graphical illustration of the concepts, their placement in the taxonomy, and the properties associated to them through axioms, the experts were asked to classify them into one of five categories ranging from *essential* (i.e. highly relevant for inclusion in the ontology) to *incorrect* (i.e. not to be included). Their individual opinions were weighted together, resulting in an overall category for each element as either *correct*, *unsure*, or *incorrect*.

For a more thorough discussion of the results, the reader is referred to previous work by Blomqvist [4], here we only summarize the main findings. From the SEMCO experiment we see a clear difference between OntoCase and naive methods for CP reuse. Primarily related to the ability to abstract, hence the ontology is given a new top structure. Although it is not inherently a positive feature to include more abstract concepts, in the specific case of enriching very shallow ontologies this turned out to be an improvement. When considering the 'correctness' (according to the scale presented above), in the JIBSNet case the fraction of 'correct' concepts on the topmost levels of the taxonomy increased from 33% in the input ontology, to 58% after executing OntoCase. The reason for making this distinction (i.e. 'top concepts') was to show that even the top structure, where most of the pattern classes and properties were added, is reasonable. In total, the amount of 'correct' concepts increased from 27% to 53%, and the amount of correct properties from 51% to 61%. When considering the agricultural ontologies of the third experiment, we noted an average increase in property 'correctness' of about 11%. It was concluded that it is in particular with respect to the ontology structure (e.g. properties, axiomatization) OntoCase improves the input ontology. When compared to manual methods the automatic approach of course does not perform as well, but it has some merits, especially compared to manually constructing an ontology only based on similar (textual) sources. For example, more properties were included in the ontologies constructed by OntoCase, even compared to manual construction. Evaluation of the taxonomy, using several methods [31,33], showed comparable levels of correctness between all ontologies.

To summarize these results we can conclude that even a simple method such as OntoCase, i.e., applying mainly state-of-the-art ontology matching techniques, is able to improve learnt ontologies, based on CPs. The main merit is the added structure. A general top structure is introduced, adding some of the missing general background knowledge not found explicitly in a text corpus. These improvements are achieved without increasing the error-rate of the ontology elements. These experiments can be viewed as an indication of the potential of utilizing CPs in OL, however more research is still needed, e.g., concerning how to take task-related ontology requirements into account when automatically reusing CPs.

### 4.2. The FRED Method

The FRED API [18] builds on Boxer [8], an implementation of DRT (Discourse Representation Theory [35]) for deep parsing of text, by enhancing frame detection and adding transformation of DRT structures (DRS) into OWL-RDF ontologies. FRED is based on an highly configurable, rule-based mapping mechanism, which links Boxer meta-model primitives, VerbNet [36] and FrameNet roles [2], domain-lexicon-derived concepts, extracted types, and named entities, to Semantic Web vocabularies such as the OWL realizations of CPs. As a consequence, FRED performs frame-driven extraction of 'situations' from NL, without using a model trained over domain-specific data. The productive set of Situation-based CPs[10] can therefore be easily specialized by using it.

When requested, FRED attempts to detect FrameNet frames based on thematic roles assigned to event relations in DRS. The detection technique largely relies on existing Boxer functionalities, but frame detection function has been enhanced, with results that are comparable[11] to the results of the state-of-the-art frame detection tool, Semafor [15] on the shared Task 19 of SemEval07 (cf. [18] for details). However, the efficiency of FRED is superior, which makes frame detection practicable for real-world applications. Frames can also be viewed as Description-Situation-based CPs[12].

FRED is a modular semantic RESTful framework that, given an input text, produces OWL and RDF formal representations based on heuristics that map the DRS output into OWL based on Logical ODPs. On one hand, the heuristics exploit the similarity between DRS first-order logic-like syntax and semantics, as well as between DRS *events* and neo-Davidsonian reification of n-ary relations, which is also recommended as a Logical ODP in description logics (and OWL) design. On the other hand, heuristics are used to solve several issues resulting from typical assumptions of DRT or hardly resolvable ambiguities in NL, e.g.:

- DRS pervasively creates variables as *discourse referents*, which create redundant anonymous individuals from the viewpoint of OWL design practices,
- the default (implicit) quantification in DRT is existential, except when explicit universal quantification is lexicalized; but this conflicts with Lexico-syntactic ODPs, e.g., for definitional constructions,
- OWL *restrictions* need to be 'reconstructed' from sparse DRS constructs,
- subsumption must be derived from different constructs, e.g., co-reference axioms,
- terminology extraction is mildly implemented in Boxer, therefore other (e.g., co-referential) rules need to be enforced.

Generally speaking, FRED deals with issues such as: *What is the difference between similar sentences in terms of DRT vs. DL? When does a sentence express a concepts vs. a fact on the grounds of DRT? What logical form should DRS boxing correspond to in a DL? What are the boundaries of a (domain) term?* A sample FRED output graph can be viewed in Figure 3. As the figure illustrates, given the example input sentence, the output

[10] The Situation CP is a CP instantiation of the n-ary relation Logical ODP for OWL, its description and OWL building block can be found at: http://ontologydesignpatterns.org/wiki/Submissions:Situation

[11] Precision when accepting partially correct detection (partially correct means that more specific or more generic frames are accepted) is for both around .75, recall is .58 for FRED vs. .75 for Semafor. The difference is partly due to that Semafor has been trained on the same FrameNet corpus as the one in the SemEval task.

[12] The Description-Situation CP enables a dual representation of situations, and their parametrized descriptions, cf. http://www.ontologydesignpatterns.org/cp/owl/descriptionsituation.owl

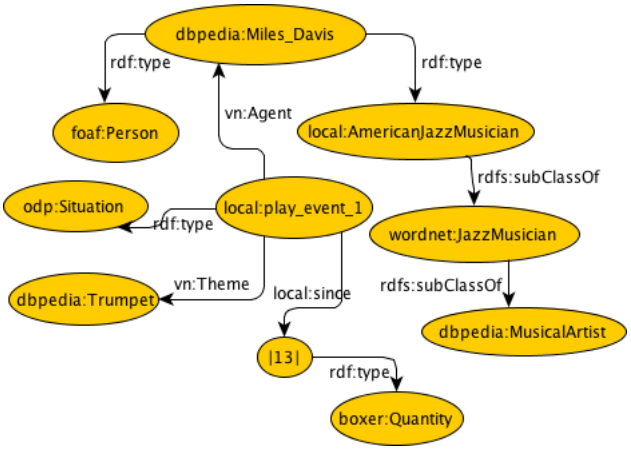



**Figure 3.** A graph of FRED RDF output for the sentence: *Miles Davis was an American jazz musician who played the trumpet since he was 13.*

.

is a 'situation'-instance (i.e. an instance of the Situation-class belonging to the Situation CP), connected to a set of elements that defines the situation, e.g., the agent involved, the theme, and the time duration. In summary, the FRED method shows the potential of combining ODPs (both Logical ODPs and CPs) with linguistic resources and deep parsing, in order to directly detect complex logical structures in NL texts, and transform them into OWL ontologies.

## 5. Concluding Remarks

In this chapter we have introduced the notion of Ontology Design Patterns, and presented the different types of ODPs available. In addition, we have discussed some challenges in OL that could be addressed through the application of ODPs. Lexico-syntactic ODPs have been used in OL for a long time, although it is only now that they are being described and shared in catalogues. In combination with Content or Logical ODPs, they can also support reengineering from textual resources into ontologies, which is one of the core problems for OL. Currently, such explicitly described Reengineering ODPs mainly exist for structured sources, but most likely this will change in the future. The last section exemplifies the application of ODPs to improve OL methods, an interesting future direction of OL. We have also pointed to a number of repositories of different patterns that could be discussed, reused and extended based on specific requirements from the OL community. So far we did not see any attempt to apply more complex or abstract ODPs, e.g. Architecture or Reasoning ODPs, in OL. Nevertheless, the requirements for applying such ODPs are there also for OL; the ontologies produced need not only to cover the domain but also to support certain tasks, e.g. reasoning tasks, hence they need to provide a certain logical structure, both on the micro and macro level. We conclude that ODPs have the potential to both provide guidance for OL, in terms of task-focus and making design choices available in a formalized manner, and act as reusable building blocks that could raise the focus of OL from single elements to detection and composition of larger and more complex components.

## Acknowledgements

This work has been partly funded by the European Commission under grant agreement FP7-ICT-2007-3/ No. 231527 (IKS - Interactive Knowledge Stack).

## References

[1] Reem Al-Halimi, Robert C. Berwick, J. F. M. Burg, Martin Chodorow, Christiane Fellbaum, Joachim Grabowski, Sanda Harabagiu, Marti A. Hearst, Graeme Hirst, Douglas A. Jones, Rick Kazman, Karen T. Kohl, Shari Landes, Claudia Leacock, George A. Miller, Katherine J. Miller, Dan Moldovan, Naoyuki Nomura, Uta Priss, Philip Resnik, David St-Onge, Randee Tengi, Reind P. van de Riet, and Ellen Voorhees. *WordNet - An Electronic Lexical Database*. MIT Press, 1998.

[2] Collin F. Baker, Charles J. Fillmore, and John B. Lowe. The berkeley FrameNet project. In *Proceedings of the 17th International Conference on Computational Linguistics*, volume 1, pages 86–90, Montreal, Quebec, Canada, 1998. Association for Computational Linguistics.

[3] Eva Blomqvist. OntoCase-Automatic Ontology Enrichment Based on Ontology Design Patterns. In *ISWC 2009, 8th International Semantic Web Conference, Chantilly, VA, USA, October 25-29, 2009. Proceedings*, volume 5823 of *Lecture Notes in Computer Science*, pages 65–80. Springer, 2009.

[4] Eva Blomqvist. *Semi-automatic Ontology Construction based on Patterns*. PhD thesis, Linköping University, Department of Computer and Information Science at the Institute of Technology, 2009.

[5] Eva Blomqvist, Valentina Presutti, and Aldo Gangemi. Experiments on Pattern-Based Ontology Design. In *In Proceeding of K-CAP 2009*, pages 41–48, Redondo Beach, California, USA, 2009. ACM.

[6] Eva Blomqvist, Valentina Presutti, Aldo Gangemi, and Enrico Daga. Experimenting with eXtreme Design. In *In Proceedings of the Conference on Knowledge Engineering and Knowledge Management (EKAW2010)*, Redondo Beach, California, USA, 2010. Springer.

[7] Eva Blomqvist and Kurt Sandkuhl. Patterns in Ontology Engineering: Classification of Ontology Patterns. In *Proceedings of the International Conference on Enterprise Information Systems 2005*, Miami Beach, Florida, May 24-28 2005.

[8] Johan Bos. Wide-coverage Semantic Analysis with Boxer. In *In STEP 2008 Conference Proceedings, Research in Computational Semantics*, page 277Ð286. College Publications, 2008.

[9] Christopher Brewster, Fabio Ciravegna, and Yorick Wilks. Background and foreground knowledge in dynamic ontology construction. In *Proc. Semantic Web Workshop, SIGIR*, 2003.

[10] Dan Brickley and Libby Miller. The Friend Of A Friend (FOAF) vocabulary specification, July 2005.

[11] Philipp Cimiano. *Ontology Learning and Population from Text - Algorithms, Evaluation and Applications*. Springer, 2006.

[12] Bonaventura Coppola, Aldo Gangemi, Alfio Gliozzo, Davide Picca, and Valentina Presutti. Frame Detection over the Semantic Web. In *Proceedings of the Fifth European Semantic Web Conference*. Springer, 2009.

[13] Bonaventura Coppola, Aldo Gangemi, Alfio Gliozzo, Davide Picca, and Valentina Presutti. Learning domain ontologies by corpus-driven framenet specialization. In Johanna Völker and Jens Lehmann, editors, *Perspectives of Ontology Learning*, Studies on the Semantic Web. AKA Heidelberg / IOS Press, 2012.

[14] Oscar Corcho, Catherine Roussey, Luis Manuel Vilches Blazquez, and Ivan Perez. Pattern-based owl ontology debugging guidelines. In *Proceedings of WOP2009 collocated with ISWC2009*, volume 516. CEUR-WS.org, November 2009.

[15] Dipanjan Das, Nathan Schneider, Desai Chen, and Noah A. Smith. Probabilistic Frame-semantic Parsing. In *Proceedings of HLT '10 - Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2009.

[16] Guadalupe Aguado de Cea, Asunción Gómez-Pérez, Elena Montiel-Ponsoda, and Mari Carmen Suárez-Figueroa. Using linguistic patterns to enhance ontology development. In Conference on Knowledge Engineering and Ontology Development (KEOD 2009), editors, *Proceedings of the International Conference on Knowledge Engineering and Ontology Development (KEOD 2009).*, 2009.

[17] Rim Djedidi and Marie-Aude Aufaure. Onto-evoal an ontology evolution approach guided by pattern modeling and quality evaluation. In Sebastian Link and Henri Prade, editors, *Foundations of Information*

*and Knowledge Systems*, volume 5956 of *Lecture Notes in Computer Science*, pages 286–305. Springer Berlin / Heidelberg, 2010.

[18] Francesco Draicchio. *Frame-driven Extraction of Linked Data and Ontologies from Text*. Master's Thesis. University of Bologna Electronic Press, University of Bologna, Dept. of Computer Science, 2012.

[19] Ontology Engineering and Patterns Task Force (OEP). http://www.w3.org/2001/sw/BestPractices/OEP/ (Accessed: 2011-03-30).

[20] Aldo Gangemi. Ontology Design Patterns for Semantic Web Content. In *The Semantic Web ISWC 2005*, volume 3729 of *Lecture Notes in Computer Science*. Springer, 2005.

[21] Aldo Gangemi. *What's in a schema? A formal metamodel for ECG and FrameNet*. Studies in Natural Language Processing. Cambridge University Press, 2010.

[22] Aldo Gangemi. Back to the Future: Frame Representation and Semantic Technologies. *Cahiers de Lexicologie*, 99(2), 2012.

[23] Aldo Gangemi, Carola Catenacci, and Massimo Battaglia. Inflammation ontology design pattern: an exercise in building a core biomedical ontology with descriptions and situations. In Domenico Maria Pisanelli, editor, *Ontologies in Medicine*. IOS Press, Amsterdam, 2004.

[24] Aldo Gangemi, Carola Catenacci, Massimiliano Ciaramita, and Jos Lehmann. Modelling ontology evaluation and validation. In *ESWC*, pages 140–154, 2006.

[25] Aldo Gangemi, Carola Catenacci, Massimiliano Ciaramita, and Jos Lehmann. Qood grid: A metaontology-based framework for ontology evaluation and selection. In *Proc. of the 4th International EON Workshop*, Located at WWW2006, 2006.

[26] Aldo Gangemi, Frehiwot Fisseha, Johannes Keizer, Jos Lehmann, Anita Liang, Ian Pettman, Margherita Sini, and Marc Taconet. A core ontology of fishery and its use in the fishery ontology service project. In Aldo Gangemi and Stefano Borgo, editors, *EKAW04 Workshop on Core Ontologies in Ontology Engineering*, Northamptonshire, UK, 2004. CEUR Proceedings Vol. 118.

[27] Aldo Gangemi, Roberto Navigli, and Paola Velardi. The ontowordnet project: Extension and axiomatization of conceptual relations in wordnet. In *CoopIS/DOA/ODBASE*, pages 820–838, 2003.

[28] Aldo Gangemi, Domenico M. Pisanelli, and Geri Steve. A formal ontology framework to represent norm dynamics. In *In Proceedings of the Second International Workshop on Legal Ontologies (LEGONT*, 2001.

[29] Aldo Gangemi and Valentina Presutti. Ontology Design Patterns. In *Handbook on Ontologies, 2nd Ed.*, International Handbooks on Information Systems. Springer, 2009.

[30] Aldo Gangemi and Valentina Presutti. Towards a pattern science for the semantic web. *Semantic Web*, 1(1-2):61–68, 2010.

[31] Asunción Gómez-Pérez. Evaluation of Taxonomic Knowledge in Ontologies and Knowledge Bases. In *Proc. of KAW'99*, volume 2, Banff, 1999.

[32] Michael Gruninger and Mark S. Fox. The role of competency questions in enterprise engineering. In *Proceedings of the IFIP WG5.7 Workshop on Benchmarking - Theory and Practice*, 1994.

[33] Nicola Guarino and Chris Welty. Evaluating Ontological Decisions with OntoClean. *Communications of the ACM*, 45(2):61–65, February 2002.

[34] Marti A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the Fourteenth International Conference on Computational Linguistics*, pages 539–545, Nantes, France, July 1992.

[35] Hans Kamp and Uwe Reyle. From Discourse to Logic: Introduction to Model-theoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory. In Sebastian Link and Henri Prade, editors, *Foundations of Information and Knowledge Systems*, volume 42 of *Studies in Linguistics and Philosophy*. Kluwer / Dordrecht, 1993.

[36] Karin Kipper, Anna Korhonen, Neville Ryant, and Martha Palmer. Extending verbnet with novel verb classes. In *Proc. of the Fifth International Conference on Language Resources and Evaluation (LREC 2006)*, Genoa, Italy, 2006.

[37] Markus Krötzsch and Ian Horrocks Frantisek Simančík. A description logic primer. In Johanna Völker and Jens Lehmann, editors, *Perspectives of Ontology Learning*, Studies on the Semantic Web. AKA Heidelberg / IOS Press, 2012.

[38] Adolfo Lozano-Tello and Asunción Gómez-Pérez. ONTOMETRIC: A Method to Choose the Appropriate Ontology. *Journal of Database Management*, 15(2), April-June 2004.

[39] Alexander Maedche. *Ontology Learning for the Semantic Web*. Kluwer Academic Publishers, 2002.

[40] Diana Maynard and Kalina Bontcheva. Natural language processing. In Johanna Völker and Jens Lehmann, editors, *Perspectives of Ontology Learning*, Studies on the Semantic Web. AKA Heidelberg / IOS Press, 2012.

[41] Diana Maynard, Adam Funk, and Wim Peters. Using lexico-syntactic ontology design patterns for ontology creation and population. In *Proceedings of WOP2009 collocated with ISWC2009*, volume 516. CEUR-WS.org, November 2009.

[42] Peter Mika, Daniel Oberle, Aldo Gangemi, and Marta Sabou. Foundations for service ontologies: Aligning owl-s to dolce. In S. Staab and P. Patel-Schneider, editors, *Proceedings of the World Wide Web Conference (WWW2004), Semantic Web Track*, 2004.

[43] Alistair Miles and Dan Brickley. SKOS core guide. W3C working draft, W3C, November 2005. Published online on November 2nd, 2005 at http://www.w3.org/TR/2005/WD-swbp-skos-core-guide-20051102/.

[44] Nadejda Nikitina, Sebastian Rudolph, and Sebastian Blohm. Refining ontologies by pattern-based completion. In Eva Blomqvist, Kurt Sandkuhl, Francois Scharffe, and Vojtech Svatek, editors, *Proceedings of the Workshop on Ontology Patterns (WOP 2009), collocated with the 8th International Semantic Web Conference (ISWC-2009), Washington D.C., USA, 25 October, 2009*, volume 516. CEUR Workshop Proceedings, 2009.

[45] Andrea G. Nuzzolese, Aldo Gangemi, and Valentina Presutti. Gathering Lexical Linked Data and Knowledge Patterns from FrameNet. In *K-CAP*, 2011.

[46] Andrea G. Nuzzolese, Aldo Gangemi, Valentina Presutti, and Paolo Ciancarini. Fine-tuning triplication with semion. In V. Presutti, V. Svatek, and F. Share, editors, *EKAW workshop on Knowledge Injection into and Extraction from Linked Data (KIELD2010)*, volume 631. CEUR Workshop Proceedings, 2010.

[47] ODP Portal. Content odp submissions. http://ontologydesignpatterns.org/wiki/Submissions:ContentOPs (Accessed: 2011-03-30).

[48] ODP Portal. Lexico-syntactic odp submissions. http://ontologydesignpatterns.org/wiki/Submissions:LexicoSyntacticODPs (Accessed: 2011-03-30).

[49] ODP Portal. Logical odp submissions. http://ontologydesignpatterns.org/wiki/Submissions:LogicalODPs (Accessed: 2011-03-30).

[50] ODP Portal. Reengineering odp submissions. http://ontologydesignpatterns.org/wiki/Submissions:ReengineeringODPs (Accessed: 2011-03-30).

[51] Fabian Suchanek. Information extraction for ontology learning. In Johanna Völker and Jens Lehmann, editors, *Perspectives of Ontology Learning*, Studies on the Semantic Web. AKA Heidelberg / IOS Press, 2012.

[52] Vojtěch Svátek, Ondrej Šváb Zamazal, and Valentina Presutti. Ontology naming pattern sauce for (human and computer) gourmets. In *Proceedings of the Workshop on Ontology Patterns (WOP 2009), collocated with the 8th International Semantic Web Conference (ISWC-2009), Washington D.C., USA, 25 October, 2009*, volume 516. CEUR Workshop Proceedings, 2009.

[53] Manchester University. Ontology design patterns (odps) public catalog. http://www.gong.manchester.ac.uk/odp/html/index.html (Accessed: 2011-03-30).

[54] Frank van Harmelen, Annette ten Teije, and Holger Wache. Knowledge engineering rediscovered: Towards reasoning patterns for the semantic web. In N. Noy, editor, *Proceedings of The Fifth International Conference on Knowledge Capture*, pages 81–88. ACM, september 2009.

[55] Paola Velardi, Roberto Navigli, Alessandro Cucchiarelli, and Francesca Neri. Evaluation of OntoLearn, a methodology for automatic population of domain ontologies. In Paul Buitelaar, Philipp Cimiano, and Bernardo Magnini, editors, *Ontology Learning from Text: Methods, Applications and Evaluation*. IOS Press, 2006.

[56] Johanna Völker, Peter Haase, and Pascal Hitzler. Learning expressive ontologies. In *Ontology Learning and Population: Bridging the Gap between Text and Knowledge*, Frontiers in Artificial Intelligence and Applications. IOS Press, 2008.

[57] Johanna Völker, Pascal Hitzler, and Philipp Cimiano. Acquisition of OWL DL Axioms from Lexical Resources. In Enrico Franconi, Michael Kifer, and Wolfgang May, editors, *Proceedings of the 4th European Semantic Web Conference (ESWC'07)*, Lecture Notes in Computer Science. Springer, 2007.

[58] Ondřej Šváb Zamazal, Vojtěch Svátek, and Luigi Iannone. Pattern-based ontology transformation service exploiting oppl and owl-api. In Philipp Cimiano and H. Pinto, editors, *Knowledge Engineering and Management by the Masses*, volume 6317 of *Lecture Notes in Computer Science*, pages 105–119. Springer Berlin / Heidelberg, 2010.

[59] Ondřej Šváb Zamazal, Vojtěch Svátek, François Scharffe, and Jérôme David. Detection and transformation of ontology patterns. In Ana Fred, Jan L. G. Dietz, Kecheng Liu, and Joaquim Filipe, editors, *Knowledge Discovery, Knowlege Engineering and Knowledge Management*, volume 128 of *Communications in Computer and Information Science*, pages 210–223. Springer Berlin Heidelberg, 2011.

[60] Haining Yao, Anthony M. Orme, and Letha Etzkorn. Cohesion Metrics for Ontology Design and Application. *Journal of Computer Science*, 1(1):107–113, 2005.