Towards explainable artificial intelligence (XAI) based on contextualizing data with knowledge graphs

by

Md Kamruzzaman Sarker

B.S., Khulna University of Engineering & Technology, 2013

M.S., Wright State University, 2019

AN ABSTRACT OF A DISSERTATION

submitted in partial fulfillment of the
requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Computer Science
Carl R. Ice College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2020

Abstract

Artificial intelligence (AI)—including the sub-fields of machine learning and deep learning has advanced considerably in recent years. In tandem with these performance improvements, understanding how AI systems make decisions has become increasingly difficult due to many nonlinear transformations of input data and the complex nature of the algorithms involved.

Explainable AI (XAI) are the techniques to examine these decision processes. A main desideratum of XAI is user understandability, while explanations should take into account the context and domain knowledge of the problem. Humans understand and reason mostly in terms of concepts and combinations thereof. A knowledge graph (KG) embodies such understanding in links between concepts; such a natural conceptual network creates a pathway to use knowledge graphs in XAI applications to improve overall understandability of complex AI algorithms.

Over the course of this dissertation, we outline a number of contributions towards explaining the AI decision in a human friendly way. We show a proof-of-concept on how domain knowledge can be used to analyze the input and output data of AI algorithms. We materialize the domain knowledge into knowledge graph (more technically ontology) and by using concept induction algorithm find the pattern between input and output. After demonstrating this, we start to experiment on a large scale, as we found that the current state of the art concept induction algorithm does not scale well with large amounts of data. To solve this runtime issue, we develop a new algorithm efficient concept induction (ECII), which improves the runtime significantly.

During this process, we also find that current tools are not adequate to create and edit the knowledge graphs, as well as that there is scarcity to quality knowledge graph. We make the creation and editing process easier, by creating OWLAx and ROWLTab plugin for the

industry-standard ontology editor, Protégé. We also develop a large knowledge graph from the Wikipedia category hierarchy.

Overall, these research contributions improved the software support to create knowledge graph, developed a better knowledge graph, and showed a new direction on how AI decision making can be explained by using a contextual knowledge graph.

Towards explainable artificial intelligence (XAI) based on contextualizing data with knowledge graphs

by

Md Kamruzzaman Sarker

B.S., Khulna University of Engineering & Technology, 2013

M.S., Wright State University, 2019

A DISSERTATION

submitted in partial fulfillment of the
requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Computer Science
Carl R. Ice College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2020

Approved by:

Major Professor
Pascal Hitzler

Copyright

Abstract

Artificial intelligence (AI)—including the sub-fields of machine learning and deep learning has advanced considerably in recent years. In tandem with these performance improvements, understanding how AI systems make decisions has become increasingly difficult due to many nonlinear transformations of input data and the complex nature of the algorithms involved.

Explainable AI (XAI) are the techniques to examine these decision processes. A main desideratum of XAI is user understandability, while explanations should take into account the context and domain knowledge of the problem. Humans understand and reason mostly in terms of concepts and combinations thereof. A knowledge graph (KG) embodies such understanding in links between concepts; such a natural conceptual network creates a pathway to use knowledge graphs in XAI applications to improve overall understandability of complex AI algorithms.

Over the course of this dissertation, we outline a number of contributions towards explaining the AI decision in a human friendly way. We show a proof-of-concept on how domain knowledge can be used to analyze the input and output data of AI algorithms. We materialize the domain knowledge into knowledge graph (more technically ontology) and by using concept induction algorithm find the pattern between input and output. After demonstrating this, we start to experiment on a large scale, as we found that the current state of the art concept induction algorithm does not scale well with large amounts of data. To solve this runtime issue, we develop a new algorithm efficient concept induction (ECII), which improves the runtime significantly.

During this process, we also find that current tools are not adequate to create and edit the knowledge graphs, as well as that there is scarcity to quality knowledge graph. We make the creation and editing process easier, by creating OWLAx and ROWLTab plugin for the

industry-standard ontology editor, Protégé. We also develop a large knowledge graph from the Wikipedia category hierarchy.

Overall, these research contributions improved the software support to create knowledge graph, developed a better knowledge graph, and showed a new direction on how AI decision making can be explained by using a contextual knowledge graph.

Table of Contents

List of Figures x

Acknowledgements xi

Dedication xii

1 Introduction 1

1.1 Concepts, Semantics, Ontology, Reasoning and Knowledge Graph 3

1.2 Outline 5

2 Towards Knowledge Graph Based XAI 7

2.1 Overview 7

2.2 Related Work 8

2.3 Proposed Framework 9

2.4 Contribution 11

3 Concept Induction 13

3.1 Overview 13

3.2 Contribution 15

4 Making Ontology Editing Easier 17

4.1 Overview 17

4.2 Contribution 20

5 Knowledge Graph 22
5.1 Motivation 22
5.2 Contribution 23
6 Conclusion 26
6.1 Summary 26
6.2 Future Work 27
A Contributions 42

List of Figures

1.1 Ontology example 4

4.1 A screenshot of ROWLTab user interface 18

4.2 A screenshot of OWLAx user interface 19

Acknowledgments

First and foremost, I want to thank Almighty ALLAH, the Compassionate, the Most Merciful who created us.

A special thanks to my advisor, Pascal Hitzler, inspired me to look at the problem from multiple broad perspectives, which immensely helped me be a better researcher.

Then I would love to thank my parents, Md Mazidur Rahman and Jahanara Ferdousi, for their enormous love, my lovely wife Lubna Suba for being patient and enduring all the pain throughout the Ph.D. time, my sisters Kamrunnaher Munni and Maysha Fahmida Iffa, brother Abdul Quaium, parents-in-law Abu Jafor and Anjumanara Begum for their unlimited support. To all my other relatives who kept faith in me during this long journey. To all my friends who live in Bangladesh, in Dayton, Ohio, in Hillsboro, Oregon, in Manhattan, Kansas, and all other places of the world for your support, Thank you.

I also express my gratitude to all the collaborators, especially at Data Semantics (DaSe) Lab, Wright State University, Kansas State University, Accenture Technology Lab, Intel Corporation.

The author acknowledges the support from the National Science Foundation Grant NO. 1017225 *III: Small: TROn – Tractable Reasoning with Ontologies*, Ohio Federal Research Network project *Human-Centered Big Data*, Defense Advanced Research Projects Agency (DARPA) under Agreement No. HR00111890019 *Recovering the Sources of Individual Differences Unduly-named Errors (ReSIDUE).*

Dedication

To my mother, Jahanara Ferdousi, who held me in her womb, gave birth, feed me, and raised me.

Chapter 1

Introduction

Artificial intelligence (AI) is creating a revolution by improving the performance of automated decisions in nearly all scientific fields. This revolutionary performance is coming through the advancement of its sub-fields, particularly machine learning and deep learning. These successes are more concretely using neural networks [47, 82, 26], where billions and billions of parameter hold the weight to make an automated decision. Besides this exceptionally high number of parameters, and those algorithms do many non-linear transformations of the input data. However, as a result of these complex transformations with a tremendous amount of parameters, the decision process cannot be interpreted by a human user [33, 57].

It is important to be able to interpret the AI decision or obtain an explanation from the AI model for a decision, as sometimes a very accurate AI system fails embarrassingly. In image classification, a best-performing state of the art AI model may make a wrong decision just by changing a single pixel on the input image; this happens in almost all domains. The fragility of these high performing systems [81, 60, 3, 17] make it justified to seek the explanation, interpretation from these systems. This is especially important in many safety-critical domains, such as medical, law, military, and autonomous vehicles.

There exists a significant amount of work to interpret/explain the AI decision. Those work can be categorized in many ways, such as: in terms of explanation representation, in

terms of explanation generation, in terms of explaining the decision, in terms of explaining the internal structure, and others. An overview of the recent work can be found here [1, 32, 89, 68, 29, 4].

Although there is no consensus on which attributes an explanation should have, being humanly understandable is one of the essential attributes [24, 19, 38, 39, 29]. Humans use concepts to explain their decision [40]. These concepts come from the knowledge human have. It is easier for a human to understand the interpretation if that is represented in terms of concepts. Structured knowledge such as knowledge graph or ontology represents these concepts by connecting one concept with another [10, 54].

In this thesis, I propose a novel way to show explanations in terms of concepts, in addition to the direct input feature. My explanation model maps the input data with a description logic knowledge base: to show explanation in human understandable concepts. This is especially important when the AI model's input is numeric, however, we want to get explanation in terms of concept. To give an example, when music is being used to heal autistic child [44, 14], the system needs to be interpretable. If the explanation system consists of just numeric value of the music, the user (doctor, parents, etc.) would not understand it easily; however, if the explanation consists of concepts like pitch, mfcc, then the system would be easy to understand. My proposed explanation framework is similar with, propositional rule extraction from trained neural network [65, 67] in characteristics. In my work, I go beyond the propositional rule extraction to show the explanation using Description Logic, which is more expressive than propositional logic.

The proposed explanation framework uses a knowledge graph as an input and concept induction (also known as concept learning) as the engine. I find its application in many fields including Semantic Web [53], Psycholog [64, 42, 9] and in many others. Large scale empirical evaluation using the proposed framework proved the necessity to improve the runtime of concept induction and use of comprehensive knowledge graph. As such, it becomes important to concentrate on exploring ways to improve the concept induction and developing a better knowledge graph. I propose a novel algorithm that exceeds the existing runtime in 2–3 orders of magnitude.

Developing a knowledge graph requires significant effort from domain experts and ontologists and time-consuming. To create a knowledge graph, domain experts discuss with the ontologists to generate the schema. First, the domain expert draws the schema with feedback from ontologists. Then, ontologists convert the schema into axioms to be machine-processable. This is somewhat a complicated procedure. In this case, one of the most used software tools is Protégé [63, 85], which supports the building process from 2000. To make the development process faster, I develop methods and also software tools. After that work, I contribute to creating a large scale knowledge graph. And perform an empirical evaluation showing how different knowledge graph impacts the generated explanation from the same AI model.

1.1 Concepts, Semantics, Ontology, Reasoning and Knowledge Graph

Concepts [59, 16] are the abstract ideas or general notions that occur in the mind, in speech, or in thought[1]. They are the fundamental building blocks of thoughts, beliefs and communication.

The semantics of concepts [8] is represented using an ontology. I focus on Description Logic (DL) [6] as a formal knowledge representation language to define ontologies since it offers reasoning support for most of its expressive families and compatibility to W3C standards e.g.,OWL 2.

A DL ontology $\mathcal{O} \doteq \langle \mathcal{T}, \mathcal{A} \rangle$ is composed of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. A TBox is a set of concept and role axioms while an ABox is a set of concept assertion axioms, e.g., $C(a)$, role assertion axioms, e.g., $R(a, b)$, and individual in equality axioms e.g., $a \neq b$, $a = b$. Triple $(\mathcal{N}_C, \mathcal{N}_R, \mathcal{N}_I)$, as signature for $\mathcal{O}$, consists of 3 disjoint sets of (i) atomic concepts $\mathcal{N}_C$, (ii) atomic roles $\mathcal{N}_R$, and (iii) individuals $\mathcal{N}_I$.

Example 1. ***(TBox and ABox Concept Assertion Axioms)***

[1] https://en.wikipedia.org/wiki/Concept

Figure 1.1 presents (i) a TBox $\mathcal{T}$ where (1.1) denotes the concept of "ComputerMemory is part of Computer", (ii) concept assertions (1.5-1.6) states that "Individual Macbook is computer and YankeeStadium is BaseballStadium ".

$$\exists partsOf.ComputerMemory \sqsubseteq Computer \quad (1.1)$$

$$\exists partsOf.ComputerCases \sqsubseteq Computer \quad (1.2)$$

$$\exists LCDMonitor \sqcap Computer \sqsubseteq ComputerMonitor \quad (1.3)$$

$$\exists USBMouse \sqcap Computer \sqsubseteq ComputerMouse \quad (1.4)$$

$$Computer(MacBook) \quad (1.5)$$

$$BaseballStadium(YankeeStadium) \quad (1.6)$$

Figure 1.1: $\mathcal{O} \doteq \langle \mathcal{T}, \mathcal{A} \rangle$. Sample of TBox $\mathcal{T}$ and ABox $\mathcal{A}$ related with Computer and Baseball.

DL supports reasoning such as (i) subsumption, denoted $\sqsubseteq$, for elaborating sub/super-concept (is-a) relationships, (ii) instance for determining instance relationships or (iii) consistency to identify contradictory knowledge.

Example 2. ***(Ontology Reasoning)***

Subsumption X and instance Y can be inferred from axioms in figure 1.1 using relevant DL completion rules e.g., [5] for $\mathcal{EL}^{++}$. For instance, from the axioms in figure 1.1 we can reason that, $LCDMonitor \sqsubseteq ComputerMonitor$.

Knowledge Graph, the term first coined by Google in 2012 [2] and from then, many researchers begin to use the term interchangeably with ontology. Technically, knowledge graph is a way of organizing information using a graph structure, by means of the node-edge-node relation. Currently, knowledge graphs are being used in many domains to improve the performance of many downstream tasks of natural language processing [62]. A detailed description of knowledge graph, completeness, embedding, etc., can be found in [25, 66, 87]. Use of ontology or knowledge graph in the context of explanation is pretty recent and an overview of some current discussion is provided in [46, 36].

[2] https://blog.google/products/search/introducing-knowledge-graph-things-not/

1.2 Outline

This thesis is a cumulative dissertation that shows a new direction to explain the AI system in a human friendly way. This work can be divided into four concrete research topics that incrementally build towards this goal. The remainder of this dissertation is outlined as follows:

Chapter 2 presents the first research topic: explaining the black-box artificial intelligence decision. It put forward the key research question, how we can explain black-box AI decision in a human friendly way. The primary contribution rerefenced in this section are

- *Explaining Trained Neural Networks with Semantic Web Technologies: First Steps* [75]

Chapter 3 discusses the concept induction algorithm, how they play a key role on the proposed explainable model. Then it shows, the state of the art, their lackings and the contribution I made. Contributions regarding the concept induction is as follows:

- *Efficient Concept Induction for Description Logics* [70]

Chapter 4 bring forth the need of software tools to build knowledge graph efficiently, then ask the research question, how we can visually create knowledge graph, and use of rules to create the knowledge graph. Contributions, referenced in this chapter to address these questions:

- *OWLAx: A Protege Plugin to Support Ontology Axiomatization through Diagramming* [73]
- *Modeling OWL with Rules: The ROWL Protege Plugin* [69]
- *Rule-Based OWL Modeling with ROWLTab Protégé* [72]

Chapter 5 discusses the importance of comprehensive knowledge graph for explainability task. Then it shows a way to break cycle from the Wikipedia Knowledge graph to make it usable for the explainability. It also provides an evaluation between multiple knowledge graphs to show the impact of knowledge graph on explainability.

- *Wikipedia Knowledge Graph for Explainable AI* [70]

Chapter 6 presents concluding remarks through a brief summary that highlights the overall contributions and how they fit together, especially with respect to the state of the art. Additionally, I provide an outlook on future work.

Chapter 2

Towards Knowledge Graph Based XAI

2.1 Overview

One of the main goals of this thesis is to produce human-understandable explanations. Human frequently understands an idea in terms of the concept better than the numeric value. When a human is a consumer, explaining the deep learning system in terms of concepts is important for properly understanding the system.

Concepts and the connection between the concepts thereof make the knowledge graph. These knowledge graphs capture the real-world information, and we can run deductive reasoner over the knowledge graph. The benefit of running deductive reasoner is that we can find inferred concepts that were not asserted, along with being able to trace back all the steps, making it fully explainable.

Authors [43, 90] proposed methods to generate human understandable concepts from the deep learning model. Author [12] demonstrates the added value of using semantic annotations to label objects in the hidden layers of popular CNN architectures. However, this utilizes only very shallow annotations and is by design, not able to produce more in-depth

explanations by using a more expressive knowledge graph. Nevertheless, these lines of research indicate that even some shallow domain knowledge is already helpful. This motivates me to use domain knowledge in the form of knowldge graph to enhance the explanation of AI system. Following these, I formulate the research question.

Q1 . *How we can generate human friendly explanation from artificial intelligence system by relating the explanation with human understandable concepts, such as knowledge graph?*

2.2 Related Work

Recently researchers are investigating ways to incorporate knowledge graph in the form of domain knowledge or background knowledge to enhance the explanation. Using shallow background knowledge, authors in [23] show; how using simple knowledge graphs (as RDF triples) helped to explain the decisions of a stock market prediction system. The authors showed that an abruptly changing stock price could be explained more human-friendly, using background information from Freebase and WikiData.One of the limitations of [23] is the limited expressiveness of RDF, which prevents using more complex background knowledge. Authors in [77] proposed to map human-understandable concepts to the hidden neurons for zero-shot learning. However, the human-understandable concepts used are attributes of the input, and there is no direct relation between the concepts, i.e., background information is shallow in expressiveness.

There is a small amount of work, to the best of my knowledge, which uses a knowledge graph with high expressiveness (such as description logic). In [2], authors indicate the use of more expressive domain knowledge or ontologies to explain the decision of image classification. Author in [20] extended the TREPAN [21] algorithm, where a decision tree is generated from a trained neural network, to explain the decision. When building the tree, it assigns more weight to those more general features in the ontology, so more generic concepts/features are formed at the top of the tree, and more specific concepts/features are

used on the bottom (lower/leaves) of the tree. One of the limitations of this approach is that the generated decision tree may become huge and hard to understand. For visual question answering, [86] proposes methods to explain why an answer is being selected. Authors in [18] showed the use of expressive background knowledge for transfer learning. Along similar lines, authors in [28] used expressive background knowledge to explain feature transferability from one model to another. These publications demonstrate the added value of rich background information for providing explanations in a human-friendly way.

Some related work along the lines of propositional rule extraction from trained AI systems are [7, 22, 50].

2.3 Proposed Framework

Before describing the framework, an example of propositional rule extraction which is taken from [45] is presented below, to get a clearer idea of the proposed framework. Assume that the input-output mapping P of the neural network without background knowledge could be extracted as

$$p_1 \wedge q \rightarrow r \qquad p_2 \wedge q \rightarrow r.$$

Now assume furthermore that we also have background knowledge K in form of the rules

$$p_1 \rightarrow p \qquad p_2 \rightarrow p.$$

The background knowledge then makes it possible to obtain the simplified input-output mapping P_K, as

$$p \wedge q \rightarrow r.$$

The simplification through the background knowledge is caused by p acting as a "generalization" of both p_1 and p_2. We can think of p, p_1 and p_2 as classes or concepts, which are hierarchically related, e.g., p_1 being "oak," p_2 being "maple," and p being "tree." This

example is confined to propositional logic.[1]

The way, I design the explanation framework (EF) is for the global interpretation of the network, instead of a local or individual explanation. We train the network with the training data. After that, to generate a global explanation, the EF takes a couple of training instances as positive individuals and a couple of other training instances as negative individuals. EF also requires a knowledge graph as input, where the training instances need to be mapped. After that, the EF runs the inductive logic programming (ILP) tools (such as DL-Learner[48], ECII3) on the input knowledge graph and the positive and negative instances to generate an explanation. An illustrative example is shown below.

Let us assume we are training an image classifier to classify objects. We train the network, and then, we want to know why the network identifies an image as a mountain. To get an explanation from this framework, we need to map the training objects with a knowledge graph (such as DBpedia, Wikipedia knowledge graph5). Suppose we mapped the training images using some mapping algorithm, where mountain objects contain *Largelandform*, *Peakofland* etc. After mapping those, we take some mountain images and some other random images from the training set and run the explanation model. The explanation framework may come up with explanations such as:

$$\exists imageContains.((GeographicArea) \sqcap (Object \sqcap \neg Device))$$

This means, the system is classifying any images as mountain, which has *GeographicArea* and not any *device*, based on this training data. Here, *GeographicArea* is a generalization of the concept *Largelandform*. We see that, the explanation not only contains the objects of the images but also generalizes the objects description according to the knowledge graph.

[1]How to go beyond the propositional paradigm in neural-symbolic integration is one of the major challenges in the field [27].

2.4 Contribution

[75] Md. Kamruzzaman Sarker, Ning Xie, Derek Doran, Michael Raymer, and Pascal Hitzler. Explaining trained neural networks with semantic web technologies: First steps. In Tarek R. Besold, Artur S. d'Avila Garcez, and Isaac Noble, editors, *Proceedings of the Twelfth International Workshop on Neural-Symbolic Learning and Reasoning, NeSy 2017, London, UK, July 17-18, 2017*, volume 2003 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2017

In this publication [75], I showed a conceptual proof to extend the propositional rule extraction using the Semantic Web technologies (knowledge graph, concept induction) to make the explanation more human friendly. This fully addresses the research question Q1. Although this shows a new direction to explain the AI system, this approach also has limitations. It does not take into account what is going on inside the model. So it is susceptible to adversarial attack[31] and sometimes may generate non-sensical explanation. However, this limitation can be used to explore the internal activity of the model too. For example, for trained deep neural network, we can try to identify what a single neuron or a group of neurons is representing. For a set of neuron/s, we can take all the instances of the training data, which activates the neuron/s, as positive instances and some other instances which do not activate that neuron as negative and run the explanation framework. It will come up with some concepts (possibly complex), which may indicate what that neuron is representing. To get insights of these types, we need to perform large scale experiments (in the range of millions if we just want to consider 20 different subsets of neurons).

For evaluating the proposed explanation framework, I experimented on ADE20K [91] dataset, which has around 20,000 images of different scene categories. This dataset also provides attributes of the images, which I have mapped with the SUMO [61] ontology. At the time of empirical evaluation of the proposed framework, I found DL-Learner is the most mature ILP system, which I used to perform the evaluation. DL-Learner took, on average more than an hour to carry out a single experiment. Taking an hour for a single experiment

is a direct hindrance to perform large scale experiments. As such, we need to use efficient ILP system that can scale well with large scale knowledge graphs and execute within minutes rather than in hours. We also need to use more complex and comprehensive background knowledge. In the next chapters, I show how to improve ILP system runtime, tools to support the building of knowledge graph, and building a large scale knowledge graph.

Chapter 3

Concept Induction

3.1 Overview

Since the rise of the Semantic Web, ontology or knowledge graph becomes the main ingredient to structure, publish, integrate, and share the data. There exist many commercial and openly available ontology, containing billions of facts. Even though, there exists many and also increasing exponentially, it is nearly impossible to make the ontology complete, as the real world information is also increasing constantly. Adding new facts to the knowledge graph is an ongoing process. As such, acquiring new facts from external data and the existing ontology itself is an ongoing research and commonly termed as ontology learning. Among the many existing methods to solve this knowledge acquisition problem, some notables are [58, 55, 88, 30].

One of the sub-task of ontology learning is concept learning or concept induction. Concept induction can be described as one of generating complex description logic class expressions S from a given description logic knowledge base (or ontology) $\mathcal{O}$ and sets P and N of instances, understood as positive and negative examples, such that $\mathcal{O} \models S(a)$ for all $a \in P$, and $\mathcal{O} \not\models S(b)$ for all $b \in N$. In a practical ontology engineering process, solutions sought are often approximate, i.e., they will not satisfy $\mathcal{O} \models S(a)$ for all $a \in P$, but for as many as

possible, and will not satisfy $\mathcal{O} \not\models S(b)$ for all $b \in N$, but for as many as possible.

As discussed in the previous chapter 2, concept induction is found to be useful for generating concept based explanation from neural networks. Along with this use case, concept induction has been employed for ontology engineering, in particular in the context of ontology and knowledge graph refinement, see e.g. [51, 66] and the use case descriptions in [15].

Concept Induction is traditionally studied with methods derived from Inductive Logic Programming (ILP). One of the preliminary work on concept induction in the context of Semantic Web isOWL Class Expression Learner (OCEL) and Class Expression Learning for Ontology Engineering (CELOE) [53]. These algorithms use a refinement operator to find concepts. Another concept induction algorithm is EL Tree Learner (ELTL) [35], which creates tree hierarchy at first and then find the concepts. Based on these algorithms, the Agile Knowledge Engineering and Semantic Web (AKSW) group produced a tool called DL-Learner [15]. Researchers later developed some improvements to these algorithms. Parallel Class Expression Learning for Ontology Engineering (PCELOE) [83], and Two way Parallel Class Expression Learning (PARCEL) [84] are the parallel execution of the original CELOE algorithm. Another algorithm is Restricted CELOE [41], where only individuals can reference concepts, and each individual can reference only one concept. Algorithm, Fast Instance Check (FIC) [35], do the materialization once and find concepts subsequently. An overview of the concept induction system can be found in [52].

The most mature and recent system for this type of Concept Induction we found is DL-Learner, as presented in [15] and based on the algorithms from [53, 49]. However, while DL-Learner is an excellent and very useful system, its algorithm – which comes with theoretical correctness results [53] – has significant performance issues in some scenarios, such as the one described in previous chapter 2. In fact, the application need for scalability was exposed through our experiments in the explainability scenario, which prompted us to investigate the runtime issue more thoroughly. The key problem is that in a single execution, the DL-Learner system will make a significant number of external calls to a description logic reasoner. While the latter has become rather efficient in recent years, the accumulated time needed, particularly if the input ontologies are large, can be prohibitive for the use of the

approach. For example, for the scenario described in [76], a single run of DL-Learner can easily take over two hours, while the scenario easily necessitates thousands of such runs. To overcome this problem, we formulate the research question.

Q2 . *How can we improve the runtime of concept induction?*

3.2 Contribution

[70] Md. Kamruzzaman Sarker and Pascal Hitzler. Efficient concept induction for description logics. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 3036–3043. AAAI Press, 2019

In the publication, *Efficient Concept Induction for Description Logics* [70], we developed a concept induction algorithm named ECII, which improved the runtime at least 2 to 3 orders of magnitude while reducing some accuracy. ECII algorithm does not use refinement operators; rather, it combines the atomic concepts to create complex concepts. We experimented with all the dataset mentioned on the original DL-Learner paper and also used the ADE20K [91] dataset to create a large scale ontology. On the small, medium, and large scale ontology (large scale ontology has around 50,000 axioms), ECII performed better in terms of runtime while producing comparable results in terms of accuracy.

This improvement of runtime comes at the cost of some accuracy. We calculated two types of accuracy, approximate accuracy and exact accuracy (accuracy by reasoner). We use the closed world assumption for approximate accuracy instead of the open-world assumption of OWL 2 ontology. For OWL 2 semantics, so this accuracy diverges from the correct accuracy. ECII performed similarly for approximate accuracy and decreased performance for exact accuracy compared to DL-Learner. This approximate calculation is necessary for some domains such as machine learning, knowledge graph instance checking. Especially for

explainability, as we described earlier, it is crucial to run the experiment within 1 minute rather than 2 hours. This system is not a better choice when accuracy is needed in terms of open-world assumption.

Further, experiments with the ECII system, using multiple large scale data (axioms size is more than 500,00), such as Wikipedia ontology5, proved ECII is highly efficient in terms of runtime. Empirical evaluation shows that ECII consistently takes less than 2 minutes to produce the results while using the large Wikipedia knowledge graph. Overall this publication addresses the research question Q1. ECII software is open source and publicly available[1] with MIT License, promoting reproducible research.

[1]https://github.com/md-k-sarker/ecii

Chapter 4

Making Ontology Editing Easier

4.1 Overview

Developing a knowledge graph or ontology requires a conceptualization of the ideas, and once the conceptualization is done, it needs to be converted into ontology axioms. When converting the conceptualizations into ontology axioms, developers and ontologists frequently find it easier to express the conceptualizations in terms of rules rather than the direct ontology axioms. To give just a simple example: The exact semantics behind a logical axiom such as

$$\mathsf{Journal} \sqsubseteq \forall \mathsf{publishedBy}.\mathsf{Organization}$$

in our experience often remains somewhat unclear even for people with significant exposure to ontologies and ontology modeling. On the other hand, a rule such as

$$\mathsf{Journal}(x) \wedge \mathsf{publishedBy}(x, y) \rightarrow \mathsf{Organization}(y)$$

is rather intuitive for most in its meaning and can be both produced and processed much more readily.

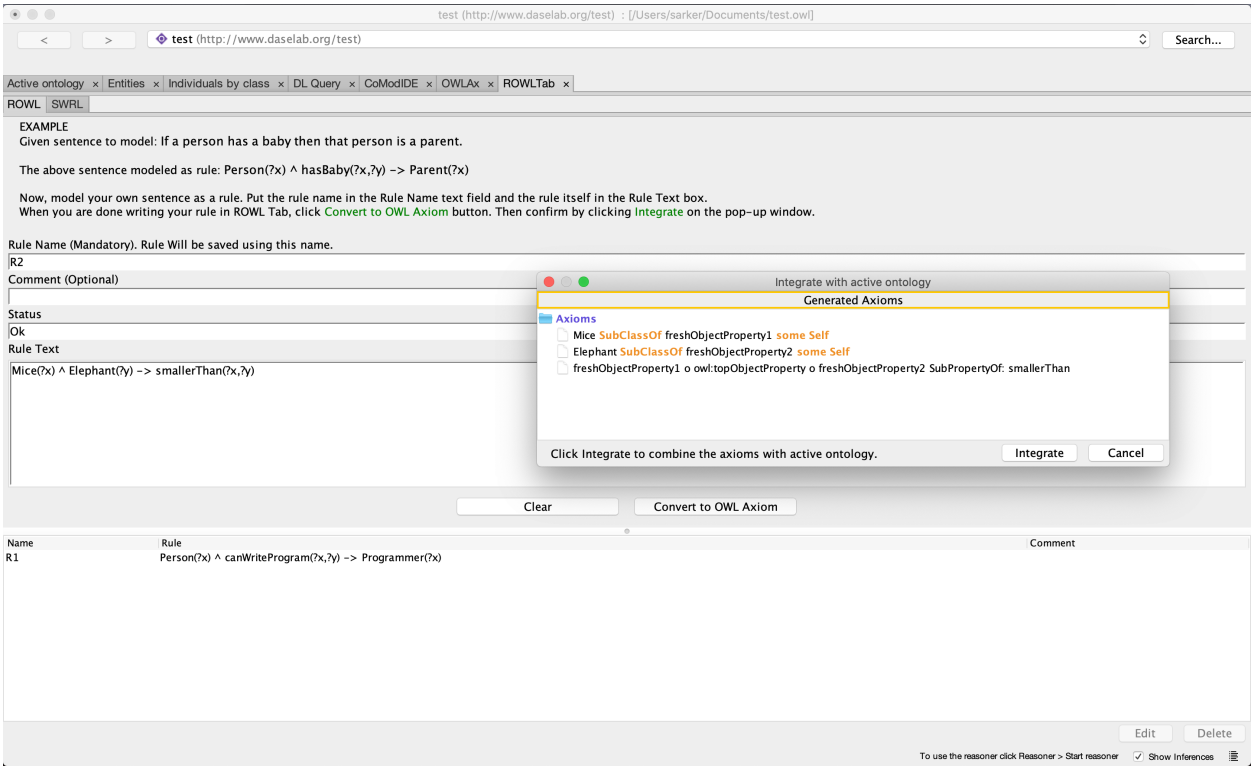

Figure 4.1: ROWLTab UI: SWRL rule and the corresponding axioms, bottom parts showing previously written rules, which are editable.

The axiom and the rule just given are, of course, logically equivalent.[1] In fact, manyOWL axioms can be expressed equivalently as rules, which are, arguably, easier to understand and to produce. Because of this, we ask the following research question.

Q3 . *Can we make automated tools to convert SWRL rule into OWL axioms?*

As a consequence of this, we have developed a Protégé plugin ROWLTab, which accepts rules as input, and adds them as OWL axioms to an ontology, provided the rule is expressible by an equivalent set of such axioms. Figure 4.1 shows the user interface of the ROWLTab plugin.

After developing the tools, we perform an user evaluation to check, whether creating ontology by inserting rule boosts user experience or not. For this, we ask the following research question.

Q4 . *Does the ROWLTab plugin improve user experience for developing ontology?*

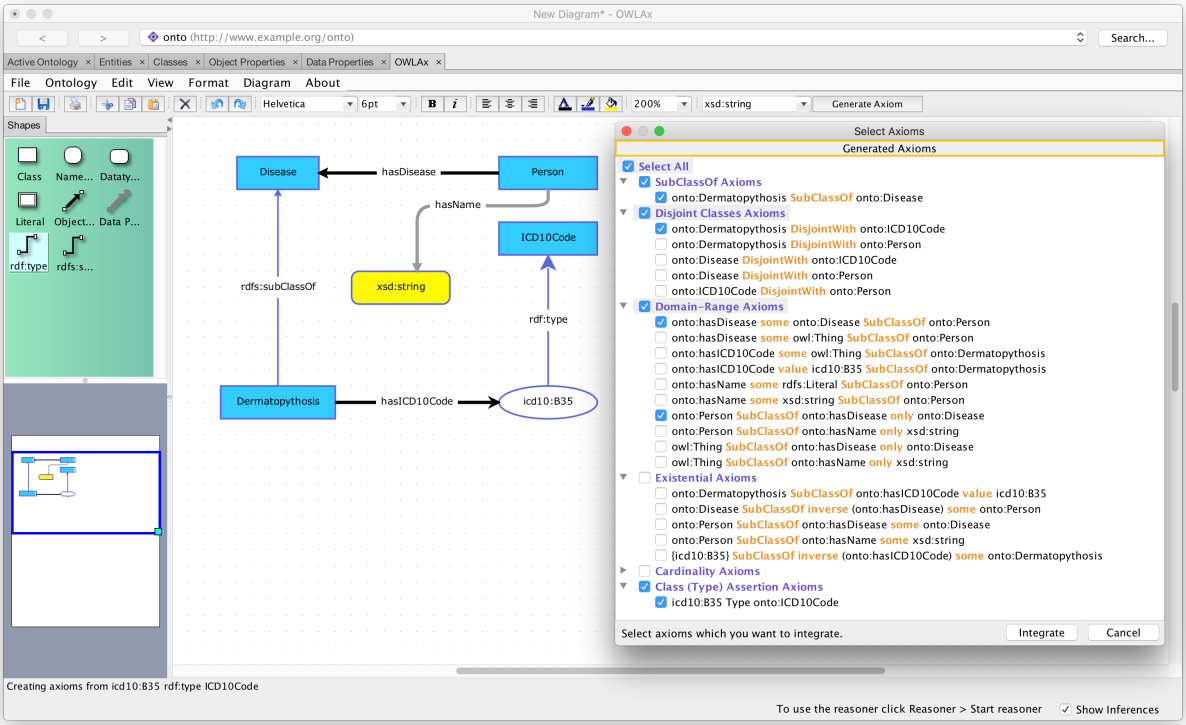

Figure 4.2: OWLAx UI: class diagram and corresponding axioms

In addition to writing OWL axioms in the form of SWRL rule, after drawing the class diagram, it is more convenient to directly draw the diagram and let the software tools take care of creating the axioms directly. Having such kind of tools would significantly decrease the time to convert the manual figure into OWL axioms, furthermore reduce the human error during the conversion process. Because of this, we ask the following research question.

Q5 . *Can we develop visual editor which will generate OWL axioms directly from drawing?*

This research effort led to the development of a tool OWLAx, which is a plugin for Protégé . Figure 4.2 shows the screenshot version 1.2.0 of the tool. Features and implementation details of it were presented in [73].

[1]When we interpret the rule in the sense of first-order predicate logic

4.2 Contribution

[69] Md. Kamruzzaman Sarker, David Carral, Adila Alfa Krisnadhi, and Pascal Hitzler. Modeling OWL with rules: The ROWL protege plugin. In Takahiro Kawamura and Heiko Paulheim, editors, *Proceedings of the ISWC 2016 Posters & Demonstrations Track co-located with 15th International Semantic Web Conference (ISWC 2016), Kobe, Japan, October 19, 2016*, volume 1690 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2016

[72] Md. Kamruzzaman Sarker, Adila Krisnadhi, David Carral, and Pascal Hitzler. Rule-based OWL modeling with rowltab protégé plugin. In Eva Blomqvist, Diana Maynard, Aldo Gangemi, Rinke Hoekstra, Pascal Hitzler, and Olaf Hartig, editors, *The Semantic Web - 14th International Conference, ESWC 2017, Portorož, Slovenia, May 28 - June 1, 2017, Proceedings, Part I*, volume 10249 of *Lecture Notes in Computer Science*, pages 419–433, 2017

[73] Md. Kamruzzaman Sarker, Adila Alfa Krisnadhi, and Pascal Hitzler. Owlax: A protege plugin to support ontology axiomatization through diagramming. In Takahiro Kawamura and Heiko Paulheim, editors, *Proceedings of the ISWC 2016 Posters & Demonstrations Track co-located with 15th International Semantic Web Conference (ISWC 2016), Kobe, Japan, October 19, 2016*, volume 1690 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2016

In the first publication *Modeling OWL with Rules: The ROWL Protégé Plugin* [69], Md Kamruzzaman Sarker and David Carral developed the ROWTab plugin to convert the SWRL rule into OWL DL axioms. In case such a conversion is not possible, the plugin notifies the user. Previously entered rules can be edited later, giving the user options to modify the rules in case of error. David Carral developed the conversion engine, and Md Kamruzzaman Sarker developed the user interface and did the integration of ROWLTab with Protégé . The development of this plugin and the publication addresses the research question Q3.

Second publication *Rule-Based OWL Modeling with ROWLTab Protégé Plugin* [72], describes how the ROWLTab plugin improves user experience to develop ontology through an

user evaluation. Adila Krisnadhi and Pascal Hitzler provided significant feedback towards the evaluation. Expert ontologists and graduate students without any prior knowledge of ontology developments took part in the user evaluation. Evaluators were asked to create ontology with and without the ROWLTab plugin. There were three different types of questions in terms of difficulty: *easy, medium, and hard*. Results were analyzed with respect to time, correctness, and the number of clicks (mouse click+keyboard click) required to answer the questions. The evaluation showed that ROWLTab required fewer clicks and less time for all types of questions. For correctness: with and without using ROWLTab, evaluators performed the same for easy questions but produced better results for medium and hard questions using ROWLTab. This evaluation and the publication addresses the research question Q4.

In the third publication *OWLAx: A Protege Plugin to Support Ontology Axiomatization through Diagramming* [73] Md Kamruzzaman Sarker, with significant feedback from Adila Krisnadhi, showed the initial way of creating ontology visually. This software tool allows users to draw the conceptual diagram of the ontology and converts to OWL axioms (OWL DL). It supports the drawing in terms of nodes and edges reminiscent of the triple of an axiom. It does not support advanced axioms. However, it shows a research direction to follow. Further publications from other authors [37, 78], shows it is being used in their research and has been found to be adaptable, extensible, and reusable. This contribution addresses research question Q5.

Chapter 5

Knowledge Graph

In this chapter, which is based on [74], I provide an evaluation of the knowledge graphs in the context of explainable AI and also introduce the cycle free Wikipedia category hierarchy knowledge graph (WKG).

5.1 Motivation

One frequently explored approach to explaining the AI decision regarding domain knowledge involves using human understandable concepts. These concepts and their relation in the real-world make the knowledge graph. There exists many knowledge graphs, some prominent are: DBpedia [11], Suggested Upper Merged Ontology (SUMO) [61], Freebase [13], ConceptNet [79], Yago [80]. As there are many knowledge graphs, it is important to know which knowledge graph will be better suited to enhance the explainability. This problem creates the following research question.

Q6 . *How can we evaluate the performance of different knowledge graphs in the context of XAI?*

When explaining the decision using a knowledge graph, the AI model's input data are mapped with the knowledge graph using some mapping algorithm (such as string similar-

ity [56] or deep learning-based entity mapping). After the mapping, concept induction takes positive and negative instances (optional), which comes from the AI model. Concept induction also takes the knowledge graph as input and tries to explain the model AI by keeping the positive instances' attributes while discarding the negative instances. The explanation comes in the form of an atomic concept or complex concept. The concepts' hierarchy helps to describe the explanation by generalizing the concept or specifying the concept. For this generalization and narrowing-down purpose, concept hierarchy and the relation (binary predication in First-Order Logic, role in Description Logic) hierarchy is used. As we need to generalize or specifies the concept, for this purpose cycle from the knowledge graph needs to be removed. In many knowledge graphs, this concept hierarchy is often cyclic and this hinders the use of knowledge graph in the context of XAI.

5.2 Contribution

[74] Md Kamruzzaman Sarker, Joshua Schwartz, Pascal Hitzler, Lu Zhou, Srikanth Nadella, Brandon Minnery, Ion Juvina, Michael L. Raymer, and William R. Aue. Wikipedia knowledge graph for explainable ai. In Boris Villazon-Terrazas, Fernando Ortiz-Rodriguez, Sanju Mishra Tiwari, and Shishir K Shandilya, editors, *Knowledge Graph and Semantic Web Conference (KGSWC)*, 2020

In the publication, *Wikipedia Knowledge Graph for Explainable AI* [74], I show a direction to evaluate the explanation performance based on different knowledge graphs and also developed the non-cyclic Wikipedia category hierarchy Knowledge Graph (WKG). For WKG's development, I use only the Wikipedia article title, article category, and hierarchy of the categories to keep it simple and easily usable in the context of XAI. In the knowledge graph, articles title become the entities (named individuals in OWL 2), their categories become the concepts (classes in OWL 2), and the concepts hierarchy form the hyponym-hypernym relation. To encode these informations I use *rdf:type* and *rdfs:subClassOf* relation. Developing this non-cyclic knowledge graph makes it easy to use in the XAI applications.

Non-cyclic version was developed by breaking cycle (details is in [74]). By breaking the

cycles, we lost some subclass-superclass relation, consequently some concepts and individuals also. As such, it becomes necessary to evaluate how much information is lost. It's easy to measure how much information is lost overall; however, it is hard to measure how this will affect the explanation's quality extracted from the AI system. As the XAI application use the hyponym-hypernym relation from the knowledge graph, it's worth to measure how many subclass-superclass relations we missed during this process. We lost substantial amount (total 1,988,618 out of 5,962,463, which is 0.33%) of subclass-superclass relation. Following that, we do a qualitative evaluation on how the semantics of the overall hierarchy is lost. We take the matching concepts from SUMO, Wikipedia, and DBpedia knowledge graphs and manually compare the subclass-superclass relation, finding the semantics remain almost the same. To give an example, axioms: $Aircraft \sqsubseteq Vehicles$ in SUMO, $Aircraft \sqsubseteq MeansOfTransportation$ in DBpedia, $Aircraft \sqsubseteq Vehicles\_by\_types$ in Wikipedia Knowledge graph shows the semantics are almost similar. One reason may be behind this, that the Wikipedia category hierarchy contains much redundant information.

Evaluation of the knowledge graph in the context of XAI is essential. When we have multiple knowledge graphs, we need to know which knowledge graph will produce a better explanation. Qualitative evaluation for explanation requires human participants, and explanation varies from human to human. To make it bias-free, I provide a quantitative evaluation to compare the performance of different knowledge graphs. For the quantitative evaluation, following [76], I compare the Wikipedia Knowledge graph with SUMO ontology on the ADE20K dataset [91]. I convert the ADE20k images annotations into an OWL ontology. After that, I aligned it with SUMO and with WKG, as in [76]. I use all five experiments mentioned in [76], but expanded the scope of the experiments. Previously because of the concepts induction's slow execution, it was impossible to do a large-scale experiment. Development of ECII [70], make it possible to do a large scale experiment. Previously I used only 3–10 images for each experiment; now, I took all the training images (around 100) of the relevant categories from the ADE20K dataset. Explanation was sought from pre-trained ResNet-50 [34]. The quantitative evaluation uses the average coverage score, and experiments show that WKG has a better average coverage score than the SUMO ontology. This

evaluation address the research question Q6.

Chapter 6

Conclusion

6.1 Summary

In this thesis, I showed a new direction to explain the AI decision; in terms of concepts, so humans can easily understand the explanation. This technique maps the training data with a contextual knowledge graph. It goes beyond the propositional rule extraction and extends this technique to allow description logic inside the rule. The concept induction algorithm is the primary tool to extract the explanation (in the forms of rules) from the trained AI model. The concept induction algorithm uses the trained AI model and contextual knowledge graph as input. Concepts that people understand form the domain knowledge, which needs to be converted into a standard knowledge base (e.g., description logic based ontology or knowledge graph). Training instances are mapped (any mapping algorithm will work, such as string similarity and others) with the knowledge base.

Following the proposed directions, empirical evaluation with larges knowledge bases proved that existing concept induction is slower for this task; thus, I had to investigate new ways to perform the rigorous experiment. Subsequently, I have developed ECII [71], which improved the concept induction algorithm. This algorithm trades little amount of accuracy to gain significant runtime improvement. ECII achieves several orders of magnitude

improvement in runtime than existing ones.

Explanation generated from this method also depends on the knowledge base, which should comprehensively cover the problem's domain. Developing such a knowledge base is a complex and time-consuming task. I have developed tools to facilitate the building process of the knowledge base. This was accomplished through the development of relevant tools [73, 69, 72]. I have also developed a large scale knowledge graph to facilitate the explanation research by creating a cycle free Wikipedia category hierarchy knowledge graph [74] and also performed an empirical evaluation to show how the performance of explanation is changing based on the Knowledge bases.

Altogether, these research contributions showed a new direction to generate explanations in a human-friendly way by mapping the training instances with the knowledge base. It improved concept induction runtime and also improved tooling support to build a large scale knowledge graph.

6.2 Future Work

The direction to explain AI decision, introduced in this thesis, provides a novel ground for further exploration. The models presented in this thesis, while delivering encouraging results, are not yet deployable in the real world, and more work is needed to advance this promising direction. I envision investigating thoroughly to find a trustable, fair, easily understandable explanation from the AI model.

Further large scale experiment should include different domains. It should use different types of AI models, such as multimodal, deep and wide neural networks, and many variants of the neural network, including but not limited to RNN, CNN to validate the proposed method. Alongside finding the input-output pattern, finding the internal node's representation in a human-understandable way is an important direction to pursue.

Besides the explainability, improving concept induction is worth investigating. Not only for its use in explainability, but there are also many tasks the concept induction algorithm is indeed essential, such as finding a semantic cluster of the data, learning new concepts from

the knowledge base, identifying whether training data is biased or not, and many others.

Acquiring and converting real-world knowledge into machine-processable form to support automated reasoning while still maintaining the actions' transparency is of great importance. Developing such methods and software tools are one of the critical components to support the explainability research.

Bibliography

[1] Amina Adadi and Mohammed Berrada. Peeking inside the black-box: A survey on Explainable Artificial Intelligence (XAI). *IEEE Access*, 6:52138–52160, 2018.

[2] Marjan Alirezaie, Martin Längkvist, Michael Sioutis, and Amy Loutfi. A symbolic approach for explaining errors in image classification tasks. In *Working Papers and Documents of the IJCAI-ECAI-2018 Workshop on*, 2018.

[3] Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani B. Srivastava, and Kai-Wei Chang. Generating natural language adversarial examples. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2890–2896. Association for Computational Linguistics, 2018.

[4] Alejandro Barredo Arrieta, Natalia Díaz Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. Explainable artificial intelligence (XAI): concepts, taxonomies, opportunities and challenges toward responsible AI. *Inf. Fusion*, 58:82–115, 2020.

[5] Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the EL envelope. In *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30 - August 5, 2005*, pages 364–369, 2005.

[6] Franz Baader and Werner Nutt. Basic description logics. In *Description logic handbook*, pages 43–95, 2003.

[7] Sebastian Bader and Pascal Hitzler. Dimensions of neural-symbolic integration – A structured survey. In Sergei N. Artëmov, Howard Barringer, Artur S. d'Avila Garcez, Luís C. Lamb, and John Woods, editors, *We Will Show Them! Essays in Honour of Dov Gabbay, Volume One*, pages 167–194. College Publications, 2005.

[8] Tim Berners-Lee, James Hendler, Ora Lassila, et al. The semantic web. *Scientific american*, 284(5):28–37, 2001.

[9] Jeffrey R Binder, Chris F Westbury, Kristen A McKiernan, Edward T Possing, and David A Medler. Distinct brain systems for processing concrete and abstract concepts. *Journal of cognitive neuroscience*, 17(6):905–917, 2005.

[10] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data-the story so far. *Semantic services, interoperability and web applications: emerging concepts*, pages 205–227, 2009.

[11] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. DBpedia—A crystallization point for the Web of Data. *Journal of Web Semantics*, 7(3):154–165, 2009.

[12] Aude Oliva Bolei Zhou, David Bau and Antonio Torralba. Interpreting deep visual representations via network dissection. *arXiv preprint arXiv:1711.05611*, 2018.

[13] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *In SIGMOD Conference*, pages 1247–1250, 2008.

[14] Jean-Pierre Briot and François Pachet. Deep learning for music generation: challenges and directions. *Neural Computing and Applications*, 32(4):981–993, 2020.

[15] Lorenz Bühmann, Jens Lehmann, and Patrick Westphal. DL-Learner – A framework for inductive learning on the semantic web. *J. Web Sem.*, 39:15–24, 2016.

[16] Susan Carey. Knowledge acquisition: Enrichment or conceptual change. *Concepts: core readings*, pages 459–487, 1999.

[17] Davide Castelvecchi. Can we open the black box of ai? *Nat.*, 538(7623):20–23, 2016.

[18] Jiaoyan Chen, Freddy Lécué, Jeff Z Pan, Ian Horrocks, and Huajun Chen. Knowledge-based transfer learning explanation. In *Sixteenth International Conference on Principles of Knowledge Representation and Reasoning*, 2018.

[19] Vladimir Cherkassky and Sauptik Dhar. Interpretation of black-box predictive models. In *Measures of Complexity*, pages 267–286. Springer, 2015.

[20] Roberto Confalonieri, Fermín Moscoso del Prado, Sebastia Agramunt, Daniel Malagarriga, Daniele Faggion, Tillman Weyde, and Tarek R. Besold. An ontology-based approach to explaining artificial neural networks, 2019.

[21] Mark Craven and Jude W Shavlik. Extracting tree-structured representations of trained networks. In *Advances in neural information processing systems*, pages 24–30, 1996.

[22] Artur S. d'Avila Garcez and Gerson Zaverucha. The connectionist inductive lerarning and logic programming system. *Applied Intelligence*, 11(1):59–77, 1999.

[23] Shumin Deng, Ningyu Zhang, Wen Zhang, Jiaoyan Chen, Jeff Z. Pan, and Huajun Chen. Knowledge-driven stock trend prediction and explanation via temporal convolutional network. In *Companion Proceedings of The 2019 World Wide Web Conference*, WWW '19, pages 678–685, New York, NY, USA, 2019. ACM.

[24] Derek Doran, Sarah Schulz, and Tarek R. Besold. What does explainable AI really mean? A new conceptualization of perspectives. In Tarek R. Besold and Oliver Kutz, editors, *Proceedings of the First International Workshop on Comprehensibility and Ex-*

planation in AI and ML 2017, Bari, Italy, 2017., volume 2071 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2017.

[25] Lisa Ehrlinger and Wolfram Wöß. Towards a definition of knowledge graphs. In Michael Martin, Martí Cuquet, and Erwin Folmer, editors, *Joint Proceedings of the Posters and Demos Track of the 12th International Conference on Semantic Systems - SEMANTiCS2016 and the 1st International Workshop on Semantic Change & Evolving Semantics (SuCCESS'16) co-located with the 12th International Conference on Semantic Systems (SEMANTiCS 2016), Leipzig, Germany, September 12-15, 2016*, volume 1695 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2016.

[26] Ronen Eldan and Ohad Shamir. The power of depth for feedforward neural networks. In Vitaly Feldman, Alexander Rakhlin, and Ohad Shamir, editors, *Proceedings of the 29th Conference on Learning Theory, COLT 2016, New York, USA, June 23-26, 2016*, volume 49 of *JMLR Workshop and Conference Proceedings*, pages 907–940. JMLR.org, 2016.

[27] Artur Garcez, Tarek Besold, Luc de Raedt, Peter Földiak, Pascal Hitzler, Thomas Icard, Kai-Uwe Kühnberger, Luis Lamb, Risto Miikkulainen, and Daniel Silver. Neural-symbolic learning and reasoning: Contributions and challenges. In Evgeniy Gabrilovich, Ramanathan Guha, Anrew McCallum, and Kevin Murphy, editors, *Proceedings of the AAAI 2015 Spring Symposium on Knowledge Representation and Reasoning: Integrating Symbolic and Neural Approaches*, volume SS-15-03 of *Technical Report*. AAAI Press, Palo Alto, CA, 2015.

[28] Yuxia Geng, Jiaoyan Chen, Ernesto Jimenez-Ruiz, and Huajun Chen. Human-centric transfer learning explanation via knowledge graph. *arXiv preprint arXiv:1901.08547*, 2019.

[29] Randy Goebel, Ajay Chander, Katharina Holzinger, Freddy Lecue, Zeynep Akata, Simone Stumpf, Peter Kieseberg, and Andreas Holzinger. Explainable AI: the new 42?

In *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, pages 295–303. Springer, 2018.

[30] Asunción Gómez-Pérez and David Manzano-Macho. An overview of methods and tools for ontology learning from texts. *Knowl. Eng. Rev.*, 19(3):187–212, 2004.

[31] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2672–2680, 2014.

[32] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)*, 51(5):93, 2018.

[33] David Gunning. Explainable artificial intelligence (XAI). *Defense Advanced Research Projects Agency (DARPA)*, 2017.

[34] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society, 2016.

[35] Sebastian Hellmann, Jens Lehmann, and Sören Auer. Learning of OWL class descriptions on very large knowledge bases. *Int. J. Semantic Web Inf. Syst.*, 5(2):25–48, 2009.

[36] Pascal Hitzler, Federico Bianchi, Monireh Ebrahimi, and Md. Kamruzzaman Sarker. Neural-symbolic integration and the semantic web. *Semantic Web*, 11(1):3–11, 2020.

[37] Pascal Hitzler and Cogan Shimizu. Modular ontologies as a bridge between human conceptualization and data. In Peter Chapman, Dominik Endres, and Nathalie Pernelle,

editors, *Graph-Based Representation and Reasoning - 23rd International Conference on Conceptual Structures, ICCS 2018, Edinburgh, UK, June 20-22, 2018, Proceedings*, volume 10872 of *Lecture Notes in Computer Science*, pages 3–6. Springer, 2018.

[38] Andreas Holzinger, Chris Biemann, Constantinos S Pattichis, and Douglas B Kell. What do we need to build explainable AI systems for the medical domain? *arXiv preprint arXiv:1712.09923*, 2017.

[39] Andreas Holzinger, Peter Kieseberg, Edgar Weippl, and A Min Tjoa. Current advances, trends and challenges of machine learning and knowledge extraction: From machine learning to explainable AI. In *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, pages 1–8. Springer, 2018.

[40] Ming Hsu, Meghana Bhatt, Ralph Adolphs, Daniel Tranel, and Colin F Camerer. Neural systems responding to degrees of uncertainty in human decision-making. *Science*, 310(5754):1680–1683, 2005.

[41] Yingbing Hua and Björn Hein. Concept learning in engineering based on refinement operator. In Fabrizio Riguzzi, Elena Bellodi, and Riccardo Zese, editors, *Up-and-Coming and Short Papers of the 28th International Conference on Inductive Logic Programming (ILP 2018), Ferrara, Italy, September 2-4, 2018*, volume 2206 of *CEUR Workshop Proceedings*, pages 76–83. CEUR-WS.org, 2018.

[42] Katja Katja Wiemer-Hastings and Xu Xu. Content differences for abstract and concrete concepts. *Cognitive science*, 29(5):719–736, 2005.

[43] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie J. Cai, James Wexler, Fernanda B. Viégas, and Rory Sayres. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV). In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 2673–2682. PMLR, 2018.

[44] Jinah Kim, Tony Wigram, and Christian Gold. The effects of improvisational music therapy on joint attention behaviors in autistic children: a randomized controlled study. *Journal of autism and developmental disorders*, 38(9):1758, 2008.

[45] Maryam Labaf, Pascal Hitzler, and Anthony B. Evans. Propositional rule extraction from neural networks under background knowledge. In Tarek R. Besold, Artur S. d'Avila Garcez, and Isaac Noble, editors, *Proceedings of the Twelfth International Workshop on Neural-Symbolic Learning and Reasoning, NeSy 2017, London, UK, July 17-18, 2017*, volume 2003 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2017.

[46] Freddy Lecue. On the role of knowledge graphs in explainable AI. *Semantic Web journal*, 2019. http://www.semantic-web-journal.net/system/files/swj2198.pdf, retrieved on July 26, 2019.

[47] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436, 2015.

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

[49] Jens Lehmann, Sören Auer, Lorenz Bühmann, and Sebastian Tramp. Class expression learning for ontology engineering. *Web Semantics: Science, Services and Agents on the World Wide Web*, 9(1):71–81, 2011.

[50] Jens Lehmann, Sebastian Bader, and Pascal Hitzler. Extracting reduced logic programs from artificial neural networks. *Applied Intelligence*, 32(3):249–266, 2010.

[51] Jens Lehmann and Lorenz Bühmann. ORE – A tool for repairing and enriching knowledge bases. In Peter F. Patel-Schneider, Yue Pan, Pascal Hitzler, Peter Mika, Lei Zhang, Jeff Z. Pan, Ian Horrocks, and Birte Glimm, editors, *The Semantic Web – ISWC 2010 – 9th International Semantic Web Conference, ISWC 2010, Shanghai, China, November 7-11, 2010, Revised Selected Papers, Part II*, volume 6497 of *Lecture Notes in Computer Science*, pages 177–193. Springer, 2010.

[52] Jens Lehmann, Nicola Fanizzi, Lorenz Bühmann, and Claudia d'Amato. Concept learning. In Jens Lehmann and Johanna Völker, editors, *Perspectives on Ontology Learning*, volume 18 of *Studies on the Semantic Web*, pages 71–91. IOS Press, Amsterdam, 2014.

[53] Jens Lehmann and Pascal Hitzler. Concept learning in description logics using refinement operators. *Machine Learning*, 78(1-2):203–250, 2010.

[54] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. Dbpedia – A large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195, 2015.

[55] Jens Lehmann and Johanna Voelker. An introduction to ontology learning.

[56] Vladimir I. Levenshtein. On the minimal redundancy of binary error-correcting codes. *Inf. Control.*, 28(4):268–291, 1975.

[57] Zachary C Lipton. The mythos of model interpretability. *arXiv preprint arXiv:1606.03490*, 2016.

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

[59] Eric Margolis, Stephen Laurence, et al. *Concepts: core readings*. Mit Press, 1999.

[60] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1765–1773, 2017.

[61] I. Niles and A. Pease. Towards a Standard Upper Ontology. In *Proceedings of the International Conference on Formal Ontology in Information Systems – Volume 2001*, pages 2–9, 2001.

[62] Natalya Fridman Noy, Yuqing Gao, Anshu Jain, Anant Narayanan, Alan Patterson, and Jamie Taylor. Industry-scale knowledge graphs: lessons and challenges. *Commun. ACM*, 62(8):36–43, 2019.

[63] Natalya Fridman Noy, Michael Sintek, Stefan Decker, Monica Crubézy, Ray W. Fergerson, and Mark A. Musen. Creating semantic web contents with protégé-2000. *IEEE Intell. Syst.*, 16(2):60–71, 2001.

[64] Allan Paivio. *Mind and Its Evolution: A Dual Coding Theoretical Approach*. Psychology Press, 2014.

[65] Vasile Palade, Ciprian-Daniel Neagu, and Ronald J. Patton. Interpretation of trained neural networks by rule extraction. In Bernd Reusch, editor, *Computational Intelligence, Theory and Applications, International Conference, 7th Fuzzy Days, Dortmund, Germany, October 1-3, 2001, Proceedings*, volume 2206 of *Lecture Notes in Computer Science*, pages 152–161. Springer, 2001.

[66] Heiko Paulheim. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic Web*, 8(3):489–508, 2017.

[67] Marko Robnik-Sikonja and Igor Kononenko. Explaining classifications for individual instances. *IEEE Trans. Knowl. Data Eng.*, 20(5):589–600, 2008.

[68] Wojciech Samek, Thomas Wiegand, and Klaus-Robert Müller. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. *arXiv preprint arXiv:1708.08296*, 2017.

[69] Md. Kamruzzaman Sarker, David Carral, Adila Alfa Krisnadhi, and Pascal Hitzler. Modeling OWL with rules: The ROWL protege plugin. In Takahiro Kawamura and Heiko Paulheim, editors, *Proceedings of the ISWC 2016 Posters & Demonstrations Track co-located with 15th International Semantic Web Conference (ISWC 2016), Kobe, Japan, October 19, 2016*, volume 1690 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2016.

[70] Md. Kamruzzaman Sarker and Pascal Hitzler. Efficient concept induction for description logics. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 3036–3043. AAAI Press, 2019.

[71] Md. Kamruzzaman Sarker and Pascal Hitzler. Efficient concept induction for description logics. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019.*, pages 3036–3043. AAAI Press, 2019.

[72] Md. Kamruzzaman Sarker, Adila Krisnadhi, David Carral, and Pascal Hitzler. Rule-based OWL modeling with rowltab protégé plugin. In Eva Blomqvist, Diana Maynard, Aldo Gangemi, Rinke Hoekstra, Pascal Hitzler, and Olaf Hartig, editors, *The Semantic Web - 14th International Conference, ESWC 2017, Portorož, Slovenia, May 28 - June 1, 2017, Proceedings, Part I*, volume 10249 of *Lecture Notes in Computer Science*, pages 419–433, 2017.

[73] Md. Kamruzzaman Sarker, Adila Alfa Krisnadhi, and Pascal Hitzler. Owlax: A protege plugin to support ontology axiomatization through diagramming. In Takahiro Kawamura and Heiko Paulheim, editors, *Proceedings of the ISWC 2016 Posters & Demonstrations Track co-located with 15th International Semantic Web Conference (ISWC 2016), Kobe, Japan, October 19, 2016*, volume 1690 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2016.

[74] Md Kamruzzaman Sarker, Joshua Schwartz, Pascal Hitzler, Lu Zhou, Srikanth Nadella, Brandon Minnery, Ion Juvina, Michael L. Raymer, and William R. Aue. Wikipedia knowledge graph for explainable ai. In Boris Villazon-Terrazas, Fernando Ortiz-

Rodriguez, Sanju Mishra Tiwari, and Shishir K Shandilya, editors, *Knowledge Graph and Semantic Web Conference (KGSWC)*, 2020.

[75] Md. Kamruzzaman Sarker, Ning Xie, Derek Doran, Michael Raymer, and Pascal Hitzler. Explaining trained neural networks with semantic web technologies: First steps. In Tarek R. Besold, Artur S. d'Avila Garcez, and Isaac Noble, editors, *Proceedings of the Twelfth International Workshop on Neural-Symbolic Learning and Reasoning, NeSy 2017, London, UK, July 17-18, 2017*, volume 2003 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2017.

[76] Md. Kamruzzaman Sarker, Ning Xie, Derek Doran, Michael Raymer, and Pascal Hitzler. Explaining trained neural networks with semantic web technologies: First steps. In Tarek R. Besold, Artur S. d'Avila Garcez, and Isaac Noble, editors, *Proceedings of the Twelfth International Workshop on Neural-Symbolic Learning and Reasoning, NeSy 2017, London, UK, July 17-18, 2017.*, volume 2003 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2017.

[77] Ramprasaath R Selvaraju, Prithvijit Chattopadhyay, Mohamed Elhoseiny, Tilak Sharma, Dhruv Batra, Devi Parikh, and Stefan Lee. Choose your neuron: Incorporating domain knowledge through neuron-importance. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 526–541, 2018.

[78] Cogan Shimizu. Towards a comprehensive modular ontology IDE and tool suite. In Sabrina Kirrane and Lalana Kagal, editors, *Proceedings of the Doctoral Consortium at ISWC 2018 co-located with 17th International Semantic Web Conference (ISWC 2018), Monterey, USA, October 8th - to - 12th, 2018*, volume 2181 of *CEUR Workshop Proceedings*, pages 65–72. CEUR-WS.org, 2018.

[79] Robyn Speer, Joshua Chin, and Catherine Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge. In Satinder P. Singh and Shaul Markovitch, editors, *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 4444–4451. AAAI Press, 2017.

[80] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A Core of Semantic Knowledge. In Carey L. Williamson, Mary Ellen Zurko, Peter F. Patel-Schneider, and Prashant J. Shenoy, editors, *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, New York, NY, USA, 2007. ACM Press.

[81] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[82] Matus Telgarsky. benefits of depth in neural networks. In Vitaly Feldman, Alexander Rakhlin, and Ohad Shamir, editors, *Proceedings of the 29th Conference on Learning Theory, COLT 2016, New York, USA, June 23-26, 2016*, volume 49 of *JMLR Workshop and Conference Proceedings*, pages 1517–1539. JMLR.org, 2016.

[83] An C. Tran, Jens Dietrich, Hans W. Guesgen, and Stephen Marsland. An approach to parallel class expression learning. In Antonis Bikakis and Adrian Giurca, editors, *Rules on the Web: Research and Applications - 6th International Symposium, RuleML 2012, Montpellier, France, August 27-29, 2012. Proceedings*, volume 7438 of *Lecture Notes in Computer Science*, pages 302–316. Springer, 2012.

[84] An C. Tran, Jens Dietrich, Hans W. Guesgen, and Stephen Marsland. Two-way parallel class expression learning. In Steven C. H. Hoi and Wray L. Buntine, editors, *Proceedings of the 4th Asian Conference on Machine Learning, ACML 2012, Singapore, Singapore, November 4-6, 2012*, volume 25 of *JMLR Proceedings*, pages 443–458. JMLR.org, 2012.

[85] Tania Tudorache, Jennifer Vendetti, and Natalya Fridman Noy. Web-protege: A lightweight OWL ontology editor for the web. In Catherine Dolbear, Alan Ruttenberg, and Ulrike Sattler, editors, *Proceedings of the Fifth OWLED Workshop on OWL:*

Experiences and Directions, collocated with the 7th International Semantic Web Conference (ISWC-2008), Karlsruhe, Germany, October 26-27, 2008, volume 432 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.

[86] Peng Wang, Qi Wu, Chunhua Shen, Anton van den Hengel, and Anthony Dick. Explicit knowledge-based reasoning for visual question answering. *arXiv preprint arXiv:1511.02570*, 2015.

[87] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743, 2017.

[88] Wilson Wong, Wei Liu, and Mohammed Bennamoun. Ontology learning from text: A look back and into the future. *ACM Comput. Surv.*, 44(4):20:1–20:36, 2012.

[89] Quan-shi Zhang and Song-Chun Zhu. Visual interpretability for deep learning: a survey. *Frontiers of Information Technology & Electronic Engineering*, 19(1):27–39, 2018.

[90] Bolei Zhou, Yiyou Sun, David Bau, and Antonio Torralba. Interpretable basis decomposition for visual explanation. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part VIII*, volume 11212 of *Lecture Notes in Computer Science*, pages 122–138. Springer, 2018.

[91] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ADE20K dataset. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 5122–5130. IEEE Computer Society, 2017.

Appendix A

Contributions

In the following pages, all contributions made for this dissertation are listed in order of appearance in this document.

Explaining Trained Neural Networks with Semantic Web Technologies: First Steps

Md Kamruzzaman Sarker, Ning Xie, Derek Doran, Michael Raymer, and Pascal Hitzler

Data Science and Security Cluster, Wright State University, Dayton, OH, USA

Abstract. The ever increasing prevalence of publicly available structured data on the World Wide Web enables new applications in a variety of domains. In this paper, we provide a conceptual approach that leverages such data in order to explain the input-output behavior of trained artificial neural networks. We apply existing Semantic Web technologies in order to provide an experimental proof of concept.

1 Introduction

Trained neural networks are usually imagined as black boxes, in that they do not give any direct indications why an output (e.g., a prediction) was made by the network. The reason for this lies in the distributed nature of the information encoded in the weighted connections of the network. Of course, for applications, e.g., safety-critical ones, this is an unsatisfactory situation. Methods are therefore sought to explain how the output of trained neural networks are reached.

This topic of explaining trained neural networks is not a new one, in fact there is already quite a bit of tradition and literature on the topic of rule extraction from such networks (see, e.g., [2,9,16]), which pursued very similar goals. Rule extraction, however, utilized propositional rules as target logic for generating explanations, and as such remained very limited in terms of explanations which are human-understandable. Novel deep learning architectures attempt to retrieve explanations as well, but often the use-case is only for computer vision tasks like object or scene recognition. Moreover, explanations in this context actually encode greater details about the images provided as input, rather than explaining why or how the neural network was able to recognize a particular object or scene.

Semantic Web [4,12] is concerned with data sharing, discovery, integration, and reuse. As field, it does not only target data on the World Wide Web, but its methods are also applicable to knowledge management and other tasks off the Web. Central to the field is the use of knowledge graphs (usually expressed using the W3C standard Resource Description Framework RDF [3]) and type logics attached to these graphs, which are called *ontologies* and are usually expressed using the W3C standard Web Ontology Language OWL [11].

This paper introduces a new paradigm for explaining neural network behavior. It goes beyond the limited propositional paradigm, and directly targets the problem of explaining neural network activity rather than the qualities of

the input. The paradigm leverages advances in knowledge representation on the World Wide Web, more precisely from the field of Semantic Web technologies. It in particular utilizes the fact that methods, tool, and structured data in the mentioned formats are now widely available, and that the amount of such structured data on the Web is in fact constantly growing [5,18]. Prominent examples of large-scale datasets include Wikidata [22] and data coming from the schema.org [10] effort which is driven by major Web search engine providers. We will utilize this available data as background knowledge, on the hypothesis that background knowledge will make it possible to obtain more concise explanations. This addresses the issue in propositional rule extraction that extracted rulesets are often large and complex, and due to their sizes difficult to understand for humans. While the paper only attempts to explain input-output behavior, the authors are actively exploring ways to also explain internal node activations.

An illustrative example

Let us consider the following very simple example which is taken from [14]. Assume that the input-output mapping P of the neural network without background knowledge could be extracted as

$$p_1 \wedge q \rightarrow r \qquad p_2 \wedge q \rightarrow r.$$

Now assume furthermore that we also have background knowledge K in form of the rules

$$p_1 \rightarrow p \qquad p_2 \rightarrow p.$$

The background knowledge then makes it possible to obtain the simplified input-output mapping P_K, as

$$p \wedge q \rightarrow r.$$

The simplification through the background knowledge is caused by p acting as a "generalization" of both p_1 and p_2. For the rest of the paper it may be beneficial to think of p, p_1 and p_2 as classes or concepts, which are hierarchically related, e.g., p_1 being "oak," p_2 being "maple," and p being "tree."

Yet this example is confined to propositional logic.[1] In the following, we show how we can bring structured (non-propositional) Semantic Web background knowledge to bear on the problem of explanation generation for trained neural networks, and how we can utilize Semantic Web technologies in order to generate non-propositional explanations. This work is at a very early stage, i.e., we will only present the conceptual architecture of the approach and minimal experimental results which are encouraging for continuing the effort.

The rest of the paper is structured as follows. In Section 2 we introduce notation as needed, in particular regarding description logics which underly the OWL standard, and briefly introduce the DL-Learner tool which features prominently in our approach. In Section 3 we present the conceptual and experimental setup

[1] How to go beyond the propositional paradigm in neural-symbolic integration is one of the major challenges in the field [8].

for our approach, and report on some first experiments. In Section 4 we conclude and discuss avenues for future work.

2 Preliminaries

We describe a minimum of preliminary notions and information needed in order to keep this paper relatively self-contained. *Description logics* [1,12] are a major paradigm in knowledge representation as a subfield of artificial intelligence. At the same time, they play a very prominent role in the Semantic Web field since they are the foundation for one of the central Semantic Web standards, namely the W3C Web Ontology Language OWL [11,12].

Technically speaking, a description logic is a decidable fragment of first-order predicate logic (sometimes with equality or other extensions) using only unary and binary predicates. The unary predicates are called *atomic classes*,[2] while the binary ones are refered to as *roles*,[3] and constants are refered to as *individuals*. In the following, we formally define the fundamental description logic known as $\mathcal{ALC}$, which will suffice for this paper. OWL is a proper superset of $\mathcal{ALC}$.

Desciption logics allow for a simplified syntax (compared to first-order predicate logic), and we will introduce $\mathcal{ALC}$ in this simplified syntax. A translation into first-order predicate logic will be provided further below.

Let $\mathcal{C}$ be a finite set of atomic classes, $\mathcal{R}$ be a finite set of roles, and $\mathcal{N}$ be a finite set of individuals. Then *class expressions* (or simply, *classes*) are defined recursively using the following grammar, where A denotes atomic classes from $\mathcal{A}$ and R denotes roles from $\mathcal{R}$. The symbols $\sqcap$ and $\sqcup$ denote conjunction and disjunction, respectively.

$$C, D ::= A \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \forall R.C \mid \exists R.C$$

A *TBox* is a set of statements, called (*general class inclusion*) *axioms*, of the form $C \sqsubseteq D$, where C and D are class expressions – the symbol $\sqsubseteq$ can be understood as a type of subset inclusion, or alternatively, as a logical implication. An *ABox* is a set of statements of the forms $A(a)$ or $R(a, b)$, where A is an atomic class, R is a role, and a, b are individuals. A description logic *knowledge base* consists of a TBox and an ABox. The notion of *ontology* is used in different ways in the literature; sometimes it is used as equivalent to TBox, sometimes as equivalent to knowledge base. We will adopt the latter usage.

We characterize the semantics of $\mathcal{ALC}$ knowledge bases by giving a translation into first-order predicate logic. If α is a TBox axiom of the form $C \sqsubseteq D$, then $\pi(\alpha)$ is defined inductively as in Figure 1, where A is a class name. ABox axioms remain unchanged.

DL-Learner [6,17] is a machine learning system inspired by inductive logic programming [20]. Given a knowledge base and two sets of individuals from the knowledge base – called positive respectively negative examples – DL-Learner

[2] or *atomic concepts*

[3] or *properties*

$$\begin{aligned}
\pi(C \sqsubseteq D) &= (\forall x_0)(\pi_{x_0}(C) \rightarrow \pi_{x_0}(D)) \\
\pi_{x_i}(A) &= A(x_i) \\
\pi_{x_i}(\neg C) &= \neg \pi_{x_i}(C) \\
\pi_{x_i}(C \sqcap D) &= \pi_{x_i}(C) \wedge \pi_{x_i}(D) \\
\pi_{x_i}(C \sqcup D) &= \pi_{x_i}(C) \vee \pi_{x_i}(D) \\
\pi_{x_i}(\forall R.C) &= (\forall x_{i+1})(R(x_i, x_{i+1}) \rightarrow \pi_{x_{i+1}}(C)) \\
\pi_{x_i}(\exists R.C) &= (\exists x_{i+1})(R(x_i, x_{i+1}) \wedge \pi_{x_{i+1}}(C))
\end{aligned}$$

Fig. 1. Translating TBox axioms into first-order predicate logic. We use auxiliary functions π_{x_i}, where the x_i are variables. The axiom $A \sqsubseteq \exists R.\exists S.B$, for example, would be translated to $(\forall x_0)((A(x_0)) \rightarrow (\exists x_1)(R(x_0, x_1) \wedge (\exists x_2)(S(x_1, x_2) \wedge B(x_2))))$.

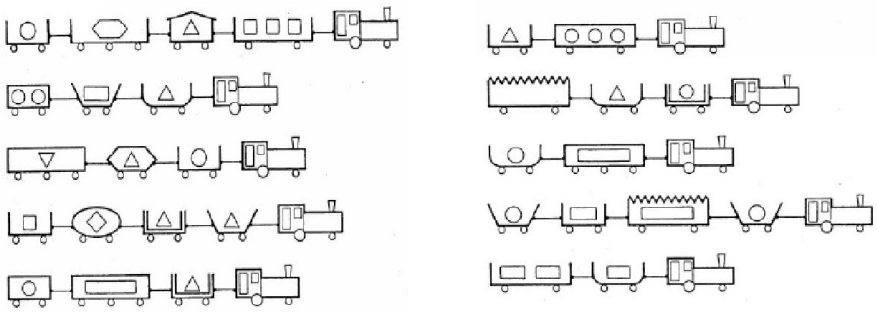

Fig. 2. Michalski's trains, picture from [15]. Positive examples on the left, negative ones on the right.

attempts to construct class expressions such that all the positive examples are contained in each of the class expressions, while none of the negative examples is. DL-Learner gives preference to shorter solutions, and in the standard setting returns approximate solutions if no fully correct solution is found. The inner workings of DL-Learner will not matter for this paper, and we refer to [6,17] for details. However, we exemplify its functionality by looking at Michalski's trains as an example, which is a symbolic machine learning task from [15], and which was presented also in [17].

For purposes of illustrating DL-Learner, Figure 2 shows two sets of trains, the positive examples are on the left, the negative ones are on the right. Following [17], we use a simple encoding of the trains as a knowledge base: Each train is an individual, and has cars attached to it using the hasCar property, and each car then falls into different categories, e.g., the top leftmost car would fall into the classes Open, Rectangular and Short, and would also have information attached to it regarding symbol carried (in this case, square), and how many of them (in this case, one). Given these examples and knowledge base, DL-Learner comes

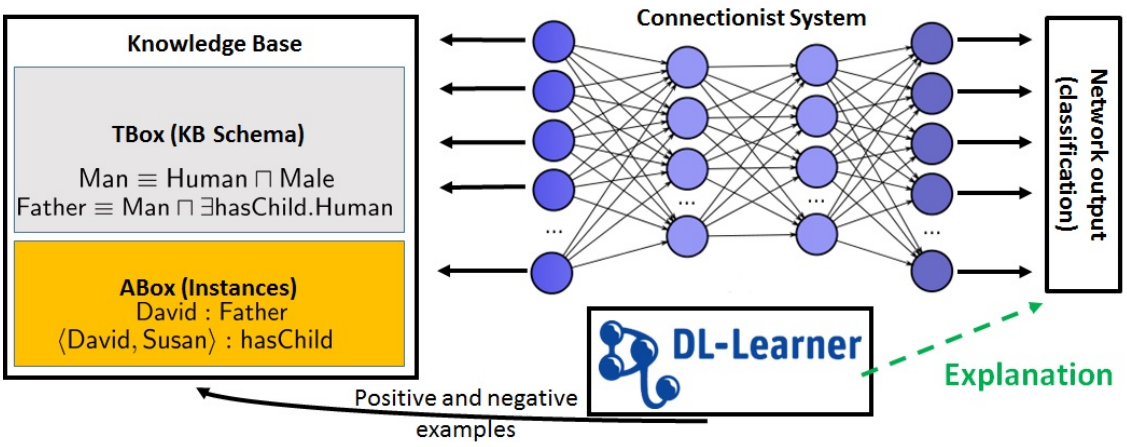

Fig. 3. Conceptual architecture – see text for explanations.

up with the class

$$\exists \text{hasCar}.(\text{Closed} \sqcap \text{Short})$$

which indeed is a simple class expression such that all positive examples fall under it, while no negative example does.

3 Approach and Experiments

In this paper, we follow the lead of the propositional rule extraction work mentioned in the introduction, with the intent of improving on it in several ways.

1. We generalize the approach by going significantly beyond the propositional rule paradigm, by utilizing description logics.
2. We include significantly sized and publicly available background knowledge in our approach in order to arrive at explanations which are more concise.

More concretely, we use DL-Learner as the key tool to arrive at the explanations. Figure 3 depicts our conceptual architecture: The trained artificial neural network (connectionist system) acts as a classifier. Its inputs are mapped to a background knowledge base and according to the networks' classification, positive and negative examples are distinguished. DL-Learner is then run on the example sets and provides explanations for the classifications based on the background knowledge.

In the following, we report on preliminary experiments we have conducted using our approach. Their sole purpose is to provide first and very preliminary insights into the feasibility of the proposed method. All experimental data is available from `http://daselab.org/projects/human-centered-big-data`.

We utilize the ADE20K dataset [23,24]. It contains 20,000 images of scenes which have been pre-classified regarding scenes depicted, i.e., we assume that the

Fig. 4. Test images. Positive examples p_1, p_2, p_3 on the left (from top), negative examples n_1, n_2, n_3 on the right (from top).

classification is done by a trained neural network.[4] For our initial test, we used six images, three of which have been classified as "outdoor warehouse" scenes (our positive examples), and three of which have not been classified as such (our negative examples). In fact, for simplicity, we took the negative examples from among the images which had been classified as "indoor warehouse" scenes. The images are shown in Figure 4.

The ADE20K dataset furthermore provides annotations for each image which identify information about objects which have been identified in the image. The annotations are in fact richer than that and also talk about the number of objects, whether they are occluded, and some more, but for our initial experiment we only used presence or absence of an object. To keep the initial experiment simple, we furthermore only used those detected objects which could easily be mapped to our chosen background knowledge, the Suggested Upper Merged Ontology (SUMO).[5] Table 1 shows, for each image, the objects we kept. The Suggested Upper Merged Ontology was chosen because it contains many, namely about 25,000 common terms which cover a wide range of domains. At the same

[4] Strictly speaking, this is not true for the training subset of the ADE20K dataset, but that doesn't really matter for our demonstration.

[5] `http://www.adampease.org/OP/`

image p_1: road, window, door, wheel, sidewalk, truck, box, building
image p_2: tree, road, window, timber, building, lumber
image p_3: hand, sidewalk, clock, steps, door, face, building, window, road
image n_1: shelf, ceiling, floor
image n_2: box, floor, wall, ceiling, product
image n_3: ceiling, wall, shelf, floor, product

Table 1. Objects recorded for each image.

time, the ontology arguably structures the terms in a relatively straightforward manner which seemed to simplify matters for our initial experiment.

In order to connect the annotations to SUMO, we used a single role called "contains." Each image was made an individual in the knowledge base. Furthermore, for each of the object identifying terms in Table 1, we either identified a corresponding matching SUMO class, or created one and added it to SUMO by inserting it at an appropriate place within SUMO's class hierarchy. We furthermore created individuals for each of the object identifying terms, including duplicates, in Table 1, and added them to the knowledge base by typing them with the corresponding class. Finally, we related each image individual to each corresponding object individual via the "contains" role.

To exemplify – for the image p_1 we added individuals road1, window1, door1, wheel1, sidewalk1, truck1, box1, building1, declared Road(road1), Window(window1), etc., and finally added the ABox statements contains(p_1, road1), contains(p_1, window1), etc., to the knowledge base. For the image p_2, we added contains(p_2, tree2), contains(p_2, road2), etc. as well as the corresponding type declarations Tree(tree2), Road(road2), etc.

The mapping of the image annotations to SUMO is of course very simple, and this was done deliberately in order to show that a straightforward approach already yields interesting results. As our work progresses, we do of course anticipate that we will utilize more complex knowledge bases and will need to generate more complex mappings from picture annotations (or features) to the background knowledge.

Finally, we ran DL-Learner on the knowledge base, with the positive and negative examples as indicated. DL-Learner returns 10 solutions, which are listed in Figure 5. Of these, some are straightforward from the image annotations, such as (1), (5), (8, (9) and (10). Others, such as (2), (4), (6), (7) are much more interesting as they provide solutions in terms of the background knowledge without using any of the terms from the original annotation. Solution (3) looks odd at first sight, but is meaningful in the context of the SUMO ontology: SelfConnectedObject is an abstract class which is a direct child of the class Object in SUMO's class hierarchy. Its natural language definition is given as "A SelfConnectedObject is any Object that does not consist of two or more disconnected parts." As such, the class is a superclass of the class Road, which explains why (3) is indeed a solution in terms of the SUMO ontology.

∃contains.Window	(1)	∃contains.LandTransitway	(6)
∃contains.Transitway	(2)	∃contains.LandArea	(7)
∃contains.SelfConnectedObject	(3)	∃contains.Building	(8)
∃contains.Roadway	(4)	∀contains.¬Floor	(9)
∃contains.Road	(5)	∀contains.¬Ceiling	(10)

Fig. 5. Solutions produced by DL-Learner for the warehouse test.

We have conducted four additional experiments along the same lines as described above. We briefly describe them below – the full raw data and results are available from `http://daselab.org/projects/human-centered-big-data`.

In the second experiment, we chose four workroom pictures as positive examples, and eight warehouse pictures (indoors and outdoors) as negative examples. An example explanation DL-Learner came up with is

$$\exists \text{contains}.(\text{DurableGood} \sqcap \neg \text{ForestProduct}).$$

On of the outdoor warehouse pictures indeed shows timber. DurableGoods in SUMO include furniture, machinery, and appliances.

In the third experiment, we chose the same four workroom pictures as negative examples, and the same eight warehouse pictures (indoors and outdoors) as positve examples. An example explanation DL-Learner came up with is

$$\forall \text{contains}.(\neg \text{Furniture} \sqcap \neg \text{IndustrialSupply}),$$

i.e., "contains neither furniture nor industrial supply". IndustrialSupply in SUMO includes machinery. Indeed it turns out that furniture alone is insufficient for distingushing between the positive and negative exaples, because "shelf" is not classified as funiture in SUMO. This shows the dependency of the explanations on the conceptualizations encoded in the background knowledge.

In the fourth experiment, we chose eight market pictures (indoors and outdoors) as positive examples, and eight warehouse pictures (indoors and outdoors) as well as four workroom pictures as negative examples. An example explanation DL-Learner came up with is

$$\exists \text{contains}.\text{SentientAgent},$$

And indeed it turns out that people are shown on all the market pictures. There is actually also a man shown on one of the warehouse pictures, driving a forklift, however "man" or "person" was not among the annotations used for the picture. This example indicates how our approach could be utilized: A human monitor inquiring with an interactive system about the reasons for a certain classification may notice that the man was missed by the software on that particular picture, and can opt to interfere with the decision and attempt to correct it.

In the fifth experiment, we chose four mountain pictures as positive examples, and eight warehouse pictures (indoors and outdoors) as well as four workroom

pictures as negative examples. An example explanation DL-Learner came up with is

$$\exists \text{contains.BodyOfWater}.$$

Indeed, it turns out that all mountain pictures in the example set show either a river or a lake. Similar to the previous example, a human monitor may be able to catch that some misclassifications may occur because presence of a body of water is not always indicative of presence of a mountain.

4 Conclusions and Further Work

We have laid out a conceptual sketch how to approach the issue of explaining artificial neural networks' classification behaviour using Semantic Web background knowledge and technologies, in a non-propositional setting. We have also reported on some very preliminary experiments to support our concepts.

The sketch already indicates where to go from here: We will need to incorporate more complex and more comprehensive background knowledge, and if readily available structured knowledge turns out to be insufficient, then we foresee using state of the art knowledge graph generation and ontology learning methods [13,19] to obtain suitable background knowledge. We will need to use automatic methods for mapping network input features to the background knowledge [7,21], while the features to be mapped may have to be generated from the input in the first place, e.g. using object recognition software in the case of images. And finally, we also intend to apply the approach to sets of hidden neurons in order to understand what their activations indicate.

Acknowledgements. This work was supported by the Ohio Federal Research Network project *Human-Centered Big Data*.

References

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, 2nd edn. (2010)
2. Bader, S., Hitzler, P.: Dimensions of neural-symbolic integration – A structured survey. In: Artëmov, S.N., Barringer, H., d'Avila Garcez, A.S., Lamb, L.C., Woods, J. (eds.) We Will Show Them! Essays in Honour of Dov Gabbay, Volume One. pp. 167–194. College Publications (2005)
3. Beckett, D., Berners-Lee, T., Prud'hommeaux, E., Carothers, G.: RDF 1.1. Turtle – Terse RDF Triple Language. W3C Recommendation (25 February 2014), available at http://www.w3.org/TR/turtle/
4. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American 284(5), 34–43 (May 2001)
5. Bizer, C., Heath, T., Berners-Lee, T.: Linked Data – The Story So Far. International Journal on Semantic Web and Information Systems 5(3), 1–22 (2009)
6. Bühmann, L., Lehmann, J., Westphal, P.: DL-Learner – A framework for inductive learning on the semantic web. Journal of Web Semantics 39, 15–24 (2016)

7. Euzenat, J., Shvaiko, P.: Ontology Matching, Second Edition. Springer (2013)
8. Garcez, A., Besold, T., de Raedt, L., Földiak, P., Hitzler, P., Icard, T., Kühnberger, K.U., Lamb, L., Miikkulainen, R., Silver, D.: Neural-symbolic learning and reasoning: Contributions and challenges. In: Gabrilovich, E., Guha, R., McCallum, A., Murphy, K. (eds.) Proceedings of the AAAI 2015 Spring Symposium on Knowledge Representation and Reasoning: Integrating Symbolic and Neural Approaches. Technical Report, vol. SS-15-03. AAAI Press, Palo Alto, CA (2015)
9. d'Avila Garcez, A.S., Zaverucha, G.: The connectionist inductive lerarning and logic programming system. Applied Intelligence 11(1), 59–77 (1999)
10. Guha, R.V., Brickley, D., Macbeth, S.: Schema.org: evolution of structured data on the web. Commun. ACM 59(2), 44–51 (2016)
11. Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S. (eds.): OWL 2 Web Ontology Language Primer (Second Edition). W3C Recommendation (11 December 2012), http://www.w3.org/TR/owl2-primer/
12. Hitzler, P., Krötzsch, M., Rudolph, S.: Foundations of Semantic Web Technologies. CRC Press/Chapman & Hall (2010)
13. Ji, H., Grishman, R.: Knowledge base population: Successful approaches and challenges. In: Lin, D., Matsumoto, Y., Mihalcea, R. (eds.) The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA. pp. 1148–1158. The Association for Computer Linguistics (2011)
14. Labaf, M., Hitzler, P., Evans, A.B.: Propositional rule extraction from neural networks under background knowledge. In: Proceedings of the Twelfth International Workshop on Neural-Symbolic Learning and Reasoning, NeSy'17, London, UK, July 2017 (2017), to appear
15. Larson, J., Michalski, R.S.: Inductive inference of VL decision rules. SIGART Newsletter 63, 38–44 (1977)
16. Lehmann, J., Bader, S., Hitzler, P.: Extracting reduced logic programs from artificial neural networks. Applied Intelligence 32(3), 249–266 (2010)
17. Lehmann, J., Hitzler, P.: Concept learning in description logics using refinement operators. Machine Learning 78(1-2), 203–250 (2010)
18. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., Bizer, C.: DBpedia – A large-scale, multilingual knowledge base extracted from Wikipedia. Semantic Web 6(2), 167–195 (2015)
19. Lehmann, J., Völker, J.: Perspectives on Ontology Learning, Studies on the Semantic Web, vol. 18. IOS Press (2014)
20. Muggleton, S., Raedt, L.D.: Inductive logic programming: Theory and methods. Journal of Logic Programming 19/20, 629–679 (1994)
21. Uren, V.S., Cimiano, P., Iria, J., Handschuh, S., Vargas-Vera, M., Motta, E., Ciravegna, F.: Semantic annotation for knowledge management: Requirements and a survey of the state of the art. J. Web Sem. 4(1), 14–28 (2006)
22. Vrandecic, D., Krötzsch, M.: Wikidata: a free collaborative knowledgebase. Commun. ACM 57(10), 78–85 (2014)
23. Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., Torralba, A.: Semantic understanding of scenes through the ADE20K dataset. arXiv preprint arXiv:1608.05442 (2016)
24. Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., Torralba, A.: Scene parsing through ADE20K dataset. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2017)

Efficient Concept Induction for Description Logics

Md Kamruzzaman Sarker and **Pascal Hitzler**
Data Semantics (DaSe) Laboratory, Dept. of Computer Science and Engineering, Wright State University
{sarker.3, pascal.hitzler}@wright.edu

Abstract

Concept Induction refers to the problem of creating complex Description Logic class descriptions (i.e., TBox axioms) from instance examples (i.e., ABox data). In this paper we look particularly at the case where both a set of positive and a set of negative instances are given, and complex class expressions are sought under which the positive but not the negative examples fall. Concept induction has found applications in ontology engineering, but existing algorithms have fundamental performance issues in some scenarios, mainly because a high number of invokations of an external Description Logic reasoner is usually required. In this paper we present a new algorithm for this problem which drastically reduces the number of reasoner invokations needed. While this comes at the expense of a more limited traversal of the search space, we show that our approach improves execution times by up to several orders of magnitude, while output correctness, measured in the amount of correct coverage of the input instances, remains reasonably high in many cases. Our approach thus should provide a strong alternative to existing systems, in particular in settings where other systems are prohibitively slow.

Introduction

With the rise of the Web Ontology Language OWL (Hitzler et al. 11 December 2012), description logics have become the leading paradigm for the representation of ontologies (Hitzler, Krötzsch, and Rudolph 2010). The knowledge acquisition bottleneck, in the form of acquisition of description logic knowledge bases, thus becomes an issue also for the field of ontology engineering and applications. Ontology Learning is a term often used for this in the Semantic Web context, and a relatively recent overview of the many facets of this subfield can be found in (Lehmann and Völker 2014).

In this paper, we study one of the subproblems of ontology learning, commonly known as Concept Induction or Concept Learning. Generally speaking, this problem can be described as one of generating complex description logic class expressions S from a given description logic knowledge base (or ontology) $\mathcal{O}$ and sets P and N of instances, understood as positive and negative examples, such that $\mathcal{O} \models S(a)$ for all $a \in P$, and $\mathcal{O} \not\models S(b)$ for all $b \in N$. In a practical ontology engineering process, solutions sought are often approximate, i.e., they will not satisfy $\mathcal{O} \models S(a)$ for all $a \in P$, but for as many as possible, and will not satisfy $\mathcal{O} \not\models S(b)$ for all $b \in N$, but for as many as possible.

Concept Induction is traditionally studied with methods derived from Inductive Logic Programming, and an overview of corresponding results and systems can be found in (Lehmann et al. 2014). The most mature and recent system for this type of Concept Induction we are aware of is DL-Learner, as presented in (Bühmann, Lehmann, and Westphal 2016) and based on the algorithms from (Lehmann and Hitzler 2010; Lehmann et al. 2011). Concept Induction has been employed for ontology engineering, in particular in the context of ontology and knowledge graph refinement, see e.g. (Lehmann and Bühmann 2010; Paulheim 2017) and the use case descriptions in (Bühmann, Lehmann, and Westphal 2016). Another recent use case in the context of explaining deep learning systems was described in (Sarker et al. 2017).

However, while DL-Learner is an excellent and very useful system, its algorithm – which comes with theoretical correctness results (Lehmann and Hitzler 2010) – has major performance issues in some scenarios, such as the one described in (Sarker et al. 2017). In fact, it was the application need for scalability exposed through our experiments in the scenario from (Sarker et al. 2017) which primarily prompted the investigations which we report herein. The key problem is that in a single execution, the DL-Learner system will make a significant number of external calls to a description logic reasoner. While the latter have become rather efficient in recent years, the accumulated time needed, in particular if the input ontologies are large, can be prohibitive for the use of the approach. For example, as we will report in the evaluation section, for the scenario described in (Sarker et al. 2017) a single run of DL-Learner can easily take over two hours, while the scenario easily necessitates thousands of such runs.

In this paper, we follow the idea that in some scenarios such as the one mentioned, it may be prudent to give up some completeness guarantees, and to instead focus on execution speed. In the approach which we describe herein, we do in fact invoke a description logic reasoner only once for each run of our algorithm. The reasoner computes a materialization of facts in term of class memberships of individuals to atomic classes, and this materialization is used through-

out the rest of the algorithm. Furthermore, we depart from the established tradition in Concept Induction in that we do not use a refinement-operator-based approach to produce candidate solutions. As a consequence, our approach saves up to several orders of magnitude on large input ontologies, as compared to DL-Learner. At the same time, accuracy of the solutions provided by our approach remains reasonably high, which indicates a favorable trade-off between accuracy and efficiency, for application scenarios where such a trade-off is desired.

The rest of the paper is organized as follows. We first describe a high-level perspective of our approach and algorithm.Then we describe the concrete algorithm which we have implemented, together with some formal results. After that, we present our experimental evaluation in terms of a comparison with DL-Learner, and finally we conclude.

The General Algorithm

In terms of terminology, we generally follow (Lehmann and Hitzler 2010) where appropriate. The general learning problem we address is *concept induction* as described in the following, where we assume that some description logic $\mathcal{L}$ has been fixed: Given an ontology $\mathcal{O}$ (consisting of an ABox and a TBox) over $\mathcal{L}$ and two sets P and N of individuals called *positive* respectively *negative examples*, we seek a (possibly complex) class expression $\mathcal{S}$ over $\mathcal{L}$ such that $\mathcal{O} \models \mathcal{S}(a)$ for all $a \in P$ (we say that C *covers* a), and $\mathcal{O} \not\models \mathcal{S}(b)$ for all $b \in N$. We call $\mathcal{S}$ a *solution* for this.

The most prominent refinement-operator based approach to concept induction, as provided in the DL-Learner system (Bühmann, Lehmann, and Westphal 2016), essentially follows a generate-and-test paradigm: Based on a so-called refinement operator, an idea borrowed from Inductive Logic Programming (Nienhuys-Cheng and de Wolf 1997), subsequent candidate concepts are generated and tested as to the degree to which they come close to a solution, and following some strategy the next candidate concepts are generated until eventually, a full solution is found or some approximate solution has been created. The testing and assessment steps involve the calling of an external description logic reasoner, while the number of generated potential candidates can be very large. Since description logic reasoners generally have to run rather complex algorithms,[1] the repeated testing necessary amounts for the majority of the runtime used.

In some application scenarios for DL-Learner such as the one presented in (Sarker et al. 2017), however, it turns out that due to the sheer amount of tests required this approach takes a very long time so that the data space cannot be sufficiently explored – we provide runtime data in the evaluation section below. Some aspects of the scenario are as follows.

(a) It requires many runs with changing example sets, while the ontology remains unchanged.

(b) The positive and negative examples do not include entities with complex property relationships. Rather, for each example a the corresponding ABox statements are only of the form $R(a, b)$, with R some role and b some (other) individual, together with statements of the form $A(a)$ and $B(b)$, where A and B are atomic classes. We call such examples *star-shaped*.

(c) It is more important to quickly arrive at relatively simple solutions, if such simple solutions exist, rather than to comprehensively explore the space of possible complex solutions. I.e., it seems favorable to trade some completeness for higher efficiency.

In this paper, we are going to show how the performance issue can be addressed in such scenarios. We will now first describe our approach in a general way, and in the next section we will describe the concrete algorithm.

Our new approach consists of the following three steps:

1. Select a finite set $\{C_1, \ldots, C_n\}$ of complex class expressions over $\mathcal{L}$, and set $\mathcal{O}' = \mathcal{O} \cup \{A_i \equiv C_i \mid i = 1, \ldots, n\}$, where the A_i are atomic classes not yet occurring in $\mathcal{O}$.
2. Use a reasoner to compute membership in atomic classes from $\mathcal{O}'$ of all individuals occuring in examples. Note that this includes the newly added atomic classes from step 1.
3. Generate candidate class expressions (possibly, iteratively) using only the constructors $\sqcap$, $\sqcup$ and $\neg$ and atomic classes, and test using the results from step 2 to what extent they constitute approximate solutions, using the assessment to guide iterative generation of candidates.

Let us briefly consider the pros and cons of this approach.

- Steps 1 and 2 need to be performed only once for each set of examples, provided $\mathcal{O}$ does not change. I.e., they can be considered pre-processing steps.
- Depending on the underlying logic $\mathcal{L}$, step 2 can take considerable time, but this preprocessing overhead would be outweighed by time saved in step 3, as it will not be necessary to invoke a reasoner for each candidate solution.
- Testing in step 3 is in general not equivalent to using a full reasoner. Our algorithm is *approximate one* in the sense that we trade some completeness for improved efficiency.
- We could also allow some use of existential and universal quantification, but for ease of comprehensibility of solutions we chose not to, currently. Quantifiers can of course be included in the complex classes generated in step 1.
- Our approach will necessarily miss solutions also because there are infinitely many possible complex class expressions involving quantifiers, while in step 1 we generate only a finite number of class expressions involving quantifiers. We implemented our approach based on the hypothesis that the benefit of significantly improved runtime will outweigh this drawback in many relevant scenarios.
- In our approach, preprocessing runtime can effectively be controlled by selecting more or fewer complex class expressions in step 1. Selecting more will mean that we include the exploration of more possible solutions but increase pre-processing time, while selecting fewer will have the opposite effect.

The scenarios we have in mind and which prompted this work are those regarding data exploration under background knowledge: The ontology $\mathcal{O}$ constitutes the background knowledge, and the exploration mechanism suggests clusters of data points for which it would be desirable to obtain a meaningful label or an explanation. E.g., data clusters could be explored visually, while labels are generated based

[1] Worst-case computational complexities for these algorithms are typically rather high (Hitzler, Krötzsch, and Rudolph 2010).

on fixed background knowledge, and displayed next to clusters. Another scenario in the context of explainable artificial intelligence is about exploring the space of activation patterns of hidden layer neurons in deep learning systems as in the already mentioned (Sarker et al. 2017), and we will include corresponding evaluation data below.

Let us now revisit the properties (a) through (c) of our setting mentioned above.

Regarding property (a), a key difference between our approach and the algorithms employed in DL-Learner is that we only have to use a reasoner once for computing Step 2, while the DL-Learner algorithm would have to invoke a reasoner for every candidate solution.

Regarding property (b), we understand that this seems to be a severe restriction. However, some settings can be compiled into the star-shaped format which we require, by making some minor additions to the ontology, e.g. by means of role chain expressions. Practically speaking, many of the known scenarios where concept learning has been applied seem to fit our requirement, and we will further elucidate this in the evaluation section.

Regarding property (c), this trade-off will not be a good one for all scenarios, but it will when time is of the essence.

Efficient Concept Induction from Instances

We now provide a concrete instance of the algorithm described in generic terms in the previous section. We call it the "Efficient Concept Induction from Instances" algorithm, ECII (pronounced like "easy"). We consider again the same three steps, and we assume $\mathcal{O}$ to be an OWL DL ontology (Hitzler et al. 11 December 2012), i.e., essentially a knowledge base expressed in the description logic $\mathcal{SROIQ}$.[2] In the description we will use some parameters, which are natural numbers n_i and k_i, and we will refer to them in our exhibition.

In step 1 of the algorithm, we select as additional complex class expressions all class expressions C which are formed by the grammar $C ::= B \mid C_1 \sqcap C_2 \mid \exists R.C$ where B is atomic, R is a role, and which contain at most n_1 occurrences of the $\sqcap$ symbol and at most n_2 occurrences of the $\exists$ symbol (in our system, both n_1 and n_2 default to 3, but can be set differently). The rationale behind this choice is simply that we wanted to remain within the $\mathcal{EL}^{++}$ language which allows for very efficient (and polynomial time) reasoning.(Hitzler et al. 11 December 2012) However, if $\mathcal{O}$ is not in OWL EL, which is the case for most of our evaluation data on which we will report later, then this does not necessarily lead to any advantage, and we could also allow other complex classes from OWL DL which are not in OWL EL – that does not fundamentally modify the approach.

In step 2 of the algorithm we compute the materialization for all relevant individuals, i.e., the membership of all individuals from the examples in all atomic classes from $\mathcal{O}'$. There are several good algorithms and systems for this, which can be used off-the-shelf. If the ontology $\mathcal{O}$ is already in OWL EL, then this reasoning task is in fact worst-case polynomial in the size of the input.

In order to explain what we do in step 3 of the algorithm, we need a bit of preparation. First, some definitions.

A *negated disjunct* is a class expression of the form $\neg(D_1 \sqcup \cdots \sqcup D_k)$, where the D_i are atomic classes from $\mathcal{O}$. A *conjunctive Horn clause* is a class expression of the form $B \sqcap \mathcal{D}$, where B is an atomic class from $\mathcal{O}'$ and $\mathcal{D}$ is a negated disjunct. A *candidate class* is a class expression of the form $\bigsqcup_{i=1}^{m} \mathcal{H}_i$, with conjunctive Horn clauses $\mathcal{H}_i$.

As the term suggests, step 3 of the algorithm will generate solution candidates which involve candidate classes, and will check whether they are solutions. The candidate classes are not the solution candidates; the latter will be defined below. We restrict ourselves to candidate classes of the mentioned form because we think that conjunctive Horn clauses can easily be understood by humans, and our intention was to provide solutions in such an easily ingestible form. We will return to this issue in the evaluation section.

We narrow the concept induction problem to the following, which fits our scenarios of interest, as described earlier.

Given an ontology $\mathcal{O}$ (consisting of an ABox and a TBox) over $\mathcal{L}$, an *example* is an individual a together with a set $\mathcal{A}(a)$ of ABox statements (not necessarily contained in $\mathcal{O}$) of the forms $A(a)$, $R(a, b)$ or $B(b)$, for any roles R, individuals b and atomic classes B. We call two sets P and N of examples the set of *positive* respectively *negative examples* and set $\mathcal{A} = \bigcup_{a \in P \cup N} \mathcal{A}(a)$. We now seek to find a (possibly complex) class expression $\mathcal{S}$ over $\mathcal{L}$ such that $\mathcal{O} \models \mathcal{S}(a)$ for all $a \in P$ (and in this case we say that C *covers* a), and also that $\mathcal{O} \not\models \mathcal{S}(b)$ for all $b \in N$. We call $\mathcal{S}$ a *solution* for this learning problem.

Now let $R_1, \ldots, R_l$ be all roles occurring in all (positive and negative) examples. A *solution candidate* is a class expression of the form $A \sqcap \prod_{i=1}^{l} \exists R_i.\mathcal{C}_i$ or

$$A \sqcap \prod_{i=1}^{l} \exists R_i. \left(\bigsqcup_{j=1}^{m} (B_{j_i} \sqcap \neg(D_1 \sqcup \ldots \sqcup D_{j_{i_k}})) \right) \quad (1)$$

where the $\mathcal{C}_i$ are candidate classes and where A and all the B_j and D_j are atomic classes in $\mathcal{O}'$.

In order to now spell out step 3 of the algorithm, we have to present how the solution candidates are generated, and how they are checked whether they are solutions.

Let us first turn to the algorithm for checking whether a solution candidate is a solution. What we in fact do, is to determine the individuals which fall under the candidate classes which are part of a solution candidate.

Let

$$\mathcal{C} = \bigsqcup_{j=1}^{m} (B_j \sqcap \neg(D_1 \sqcup \ldots \sqcup D_{j_k}))$$

be such a candidate class. We now require some definitions. If R is a role occuring in $\mathcal{A}(a)$ for a (positive or negative) example a, then we define *the set of all R-fillers of a* to be the set $R(a) = \{b \mid R(a, b) \in \mathcal{A}(a)\}$, and if X is a set of individuals then we define *the set of all inverse R-fillers of X* to be the set $R^-(X) = \{a \mid$ there is some $b \in$

[2] For background on description logics, see (Baader, Brandt, and Lutz 2005; Hitzler, Krötzsch, and Rudolph 2010)

X with $R(a,b) \in \mathcal{A}(a)\}$. We also set

$$\overline{R}^+ = \bigcup_{a \in P} R(a) \qquad \overline{R}^- = \bigcup_{a \in N} R(a) \qquad \overline{R} = \overline{R}^+ \cup \overline{R}^-$$

for each role R. The *extension* $\downarrow B$ of an atomic class B is defined as $\downarrow B = \{b \mid B(b) \in \mathcal{A}\}$. The extension of the class candidate $\mathcal{C}$ given above is then defined as

$$\downarrow \mathcal{C} = \bigcup_{j=1}^{m} (\downarrow B_j \cap \setminus (\downarrow D_1 \cup \ldots \cup \downarrow D_{j_k})).$$

Now if

$$\mathcal{S} = A \sqcap \prod_{i=1}^{l} \exists R_i.\mathcal{C}_i$$

is a solution candidate, then let

$$\downarrow \mathcal{S} = \downarrow A \cap \bigcap_{i=1}^{l} R_i^- (\downarrow \mathcal{C}_i).$$

We now call a solution candidate $\mathcal{S}$ an *approximate solution* if and only if both of the following hold.

1. $a \in \downarrow \mathcal{S}$ for all positive examples a in P.
2. $a \notin \downarrow \mathcal{S}$ for all negative examples a in N.

Note that checking whether a solution candidate is an approximate solution can be done simply by means of a number of straightforward set-theoretic operations, which is easily implemented. In the general case, approximate solutions will of course not be (full) solutions, and whether or not this is a reasonable thing to do depends on the use case, and in particular on the question whether the runtime improvments for the use case outweigh the severity of the reasoning mistakes we get in return. We will better understand this trade-off when we discuss our evaluation results.

Let us now turn to the generation of solution candidates. Essentially, the set of all possible solution candiates can be understood as a search space, within which we want to locate solutions or at least approximate solutions which are highly accurate in terms of coverage or not of the examples.

The DL-Learner system, which is our primary comparison, is based on traversing the search space by means of a so-called *refinement operator*, an idea borrowed from Inductive Logic Programming (Nienhuys-Cheng and de Wolf 1997): Given a solution candidate, the refinement operator produces a set of new solution candidates. These are all assessed as to their accuracy in terms of coverage or not of the examples, and the best are kept and the process is iterated. The DL-Learner system calls an external reasoner each time the accuracy of a candidate solution is to be assessed.

Our approach, however, dispenses of the need to call an external reasoner as discussed above. We could now of course use a refinement-operator approach to iteratively create solution candidates and check them. However since accuracy assessment in our setting is very quick, and since our solutions are of very specific forms, we instead opted for a direct assembly of solution candidates from its parts, as described in the following.

Recall that our solution candidates are of the form (1). Our algorithm for constructing solution candidates consists of three consecutive steps: (I) Select a set of conjunctive Horn clauses. (II) Select a set of candidate classes constructed from the selection in step (I). (III) Select a set of candidate solutions constructed from the selection in step (II).

These three steps, each of which we are going to describe in detail shortly, depend on five user-defined parameters $k_1, \ldots, k_5$ which are natural numbers. Their default values are $k_1 = k_2 = k_3 = 3$ and $k_4 = k_5 = 50$, but they can be changed by the user. There is another user-defined parameters $keepCommonTypes$, which defines whether to keep or delete the $commonTypes$ (which are the set of atomic concepts which appear both in positive and negative instances). Default value for it is $false$.

(I) For every role R occuring in the examples, set $N_R = \{B \mid B$ is an atomic class in $\mathcal{O}'$ and there is $b \in \overline{R}$ with $b \in \downarrow B\}$. Then, for each role R, construct the set $H_{0,R}$ of all conjunctive Horn clauses, each of which contains only atomic classes from N_R, and at most k_1 of such classes each.

Then, for each $H \in H_{0,R}$, calculate the accuracy of H as

$$\alpha_1(H) = \frac{|\overline{R}^+ \cap \downarrow H| + |\overline{R}^- \setminus \downarrow H|}{|\overline{R}|}.$$

Finally, let H_R be the set of the k_4 conjunctive Horn clauses from $H_{0,R}$ which have the highest accuracy; if two clauses are of the same accuracy, then we use those of shorter length, where length is measured in the number of atomic classes occuring in the conjunctive Horn clause. H_R is the set of conjunctive Horn clauses selected in this step.

(II) For every role R occuring in the examples, set $C_{0,R}$ to be the set of candidate classes assembled as disjunctions of maximally k_2 conjunctive Horn clauses from H_R each.

Then, as before, we select the set C_R of the k_6 candidate classes from $C_{0,R}$ which have the highest accuracy, and if two candidate classes are of the same accuracy, then we use those of shorter length.

(III) Construct the set of candidate solutions $S_0 = \left\{ A \sqcap \prod_{i=1}^{k_3} \exists R_i.\mathcal{C}_i \middle| \mathcal{C}_i \in C_{R_i}, A \text{ an atomic class in } \mathcal{O}' \right\}$. Our output set S consists of the best candidate solutions from S_0 again selected by means of highest accuracy and shortest length, where the accuracy for each $C \in S$ is

$$\alpha_2(C) = \frac{|P \cap \downarrow C| + |N \setminus \downarrow C|}{|P \cup N|}.$$

Let us make some remarks. In (I), we essentially brute-force the search for good conjunctive Horn clauses for each set of R-fillers, and N_R are all the classes relevant for these R-fillers. Accuracy is simply measured by coverage of R-fillers from the positive examples and non-coverage of the R-fillers from the negative examples, as a quotient with the number of all R-fillers. Length as a tie-breaker makes sense because we retain our intention to produce human-interpretable solutions, i.e., simple solutions are preferred. The size of $H_{0,R}$ is of course exponential in k_1, but this simply reflects the nature of the search space. Likewise, step (II) is exponential in k_2, and step (III) is exponential in k_3. The accuracy measure in step (III) is essentially the same as the one in steps (I) and (II), just that now we are looking at the examples, instead of the R-fillers. The required

computations in all steps are straightforward arithmetic or set operations.

Due to the already mentioned deliberate decision to trade completeness for runtime improvements, our approach will of course not necessarily find all solutions or best solutions. It is to be understood as a heuristics which delivers a favorable trade-off between accuracy and speed, as we argue in the evaluation section below.

Experimental Evaluation

The goal of our experimental evaluation was to test the hypothesis that the ECII algorithm leads to a favorable trade-off between runtime improvements and loss in accuracy, compared to DL-Learner. We expected to see runtime improvements of 2 or more orders of magnitude for large input ontologies, while we expected that accuracy would only moderately decrease in many test cases.

To evaluate our approach, we implemented the ECII system in Java (version 1.8) which makes it platform independent. We made use of the OWL API (Horridge and Bechhofer 2011) (version 4.5), which is an open source implementation for manipulating OWL 2 ontologies. As external reasoner for step 1 of the algorithm we used Pellet (Sirin et al.) In principle, ECII can also use other reasoners. We used Apache Maven as build system. The ECII system and all experimental data and results, including ontologies and configuration files are available online.[3]

All experiments were conducted on a 2.2. GHz core I7 machine with 16GB RAM.

For DL-Learner we used the CELOE (Lehmann et al. 2011) algorithm and the fast instance check (DL FIC) variation of it, which is another approximation approach which trades time for correctness. We terminated DL-Learner at the first occurance of a solution with accuracy 1.0, making use of the *stopOnFirstDefinition* parameter. For some large ontologies DL-Learner could not produce a solution with accuracy 1.0 within 4,500 seconds (i.e., 75 minutes); in these cases we terminated the algorithm after 4,500 seconds, using the *maxExecutionTimeInSeconds* parameter. For the FIC mode of DL-Learner, the time limit was set to be the execution time of the ECII system. For ECII we used the default settings, for K_s i.e., $k_1 = k_2 = k_3 = 3$ and $k_4 = k_5 = 50$ and varied the $keepCommonTypes$ as true and false, as mentioned earlier. We use ECII to denote the ECII system with default parameters, and DL-Learner to denote DL-Learner with full reasoner (CELOE setting) unless otherwise mentioned. Note that a runtime comparison for the cases where DL-Learner does not produce a solution with accuracy 1.0 is difficult to do, since DL-Learner would simply keep producing candidate solutions for a very long time, while ECII by design tests only a limited number of candidate solutions. Our 75-minute cap on DL-Learner runtime is unavoidably somewhat arbitrary.

As evaluation scenarios we used all evaluation scenarios (except Carcinogesis) from the original DL-Learner algorithm paper (Lehmann and Hitzler 2010), as well as the scenario from (Sarker et al. 2017) which makes use of the ADE20k (Zhou et al. 2017) dataset. We cannot describe all these scenarios in detail, and the reader is asked to refer to (Lehmann and Hitzler 2010; Sarker et al. 2017) for details. The evaluation scenarios from (Lehmann and Hitzler 2010) were Michalski's trains (Michalski 1980), Forte Family (Richards and Mooney 1995), Poker (`http://www.ics.uci.edu/~mlearn/MLRepository.html`), Moral Reasoner (`http://mlearn.ics.uci.edu/databases/moral-reasoner/`), and Yinyang family relationship (Iannone, Palmisano, and Fanizzi 2007), which are benchmark scenarios from Inductive Logic Programming. Carcinogesis was excluded because according to (Lehmann and Hitzler 2010) it did not work with a full reasoner under DL-Learner, and we encountered the same problem.

We will now briefly discuss each of the scenarios in turn, before we summarize; Table 1 provides an overview of the results. Runtimes were averaged over 3 runs. Accuracy of a solution S was assessed as

$$\alpha_3(S) = \frac{|P_S| + |N_S|}{|P \cup N|}, \text{ where}$$
$$P_S = \{a \in P \mid \mathcal{O} \cup \mathcal{A} \models S(a)\} \text{ and}$$
$$N_S = \{b \in N \mid \mathcal{O} \cup \mathcal{A} \not\models S(b)\},$$

using a full reasoner. DL-Learner fast instance check (DL FIC) is compared with the α_2 accuracy score (see above) of ECII system with default (ECII DF) and keeping common types (ECII KCT). In Table 1, the accuracy value for DL-Learner is always the one with the best solution. The α_3 accuracy score for DL-Learner is the score of the result returned by DL-Learner with the highest such score. The α_2 score for ECII is the average score over three runs, where for each run the solution with the best α_2 score is used for the average. The α_3 accuracy score for ECII has been computed by taking all (or if more than five, a random section of five) of the results returned by ECII which score maximally in terms of α_2. For each of these, the α_3 score has been calculated using a full reasoner (in fact, Pellet), and averaging over these results.

The Yinyang family relationship problem is about creating descriptions for family relationship types from instance data. This dataset includes a small ontology with 157 logical axioms. We notice that ECII with default parametes performs worse than DL-Learner in this task both in terms of runtime and in terms of α_3 accuracy, but ECII KCT obtains same α_2 score as DL FIC when keeping the common types.

The Michalski's train dataset ontology has 273 axioms. On this task, ECII is significantly outperformed by DL-Learner in terms of runtime, while both find perfect solutions. The reason that DL-Learner is quicker lies in the fact that it quickly comes up with solution by making a good choice in the refinement operator steps, thus leading to a quick termination. This is not necessarily an indication that DL-Learner runtime will always be significantly quicker for problems of this input size – see e.g. the next paragraph.

The Forte family dataset is also of small size, the ontology has 341 axioms. The problem defined for this was to describe the uncle (Richards and Mooney 1995; Lehmann and

[3] `https://github.com/md-k-sarker/ecii-expr`.

Experiment Name	Number of Logical Axioms	Runtime (sec)					Accuracy (α_3)		Accuracy α_2			
		DL[a]	DL FIC(1)[b]	DL FIC(2)[c]	ECII DF[d]	ECII KCT[e]	DL[a]	ECII DF[d]	DL FIC(1)[b]	DL FIC(2)[c]	ECII DF[d]	ECII KCT[e]
Yinyang_examples	157	0.065	0.0131	0.019	0.089	0.143	1.000	0.610	1.000	1.000	0.799	1.000
Trains	273	0.01	0.020	0.047	0.05	0.095	1.000	1.000	1.000	1.000	1.000	1.000
Forte	341	2.5	1.169	6.145	0.95	0.331	0.965	0.642	0.875	0.875	0.733	1.000
Poker	1,368	0.066	0.714	0.817	1	0.281	1.000	1.000	0.981	0.984	1.000	1.000
Moral Reasoner	4,666	0.1	3.106	4.154	5.47	6.873	1.000	0.785	1.000	1.000	1.000	1.000
ADE20k I	4,714	577.3[f]	4.268	31.887	1.966	23.775	0.926	0.416	0.263	0.814	0.744	1.000
ADE20k II	7,300	983.4[f]	16.187	307.65	20.8	293.44	1.000	0.673	0.413	0.413	0.846	0.900
ADE20k III	12,193	4,500[g]	13.202	263.217	51	238.8	0.375	0.937	0.375	0.375	0.930	0.937
ADE20k IV	47,468	4,500[g]	93.658	523.673	116	423.349	0.375	NA	0.608	0.608	0.660	0.608

[a] DL : DL-Learner
[b] DL FIC (1) : DL-Learner fast instance check with runtime capped at execution time of ECII DF
[c] DL FIC (2) : DL-Learner fast instance check with runtime capped at execution time of ECII KCT
[d] ECII DF : ECII default parameters
[e] ECII KCT : ECII keep common types and other default parameters
[f] Runtimes for DL-Learner were capped at 600 seconds.
[g] Runtimes for DL-Learner were capped at 4,500 seconds.

Table 1: Runtime and accuracy comparison of DL-Learner and ECII. Some figures are averages as described in the text.

Hitzler 2010) relationship. Two problems (large and small) were taken from this dataset by varying the numbers of positive and negative individuals, and the score is averaged. ECII outperforms DL-Learner in terms of runtime but dips in accuracy. For approximation accuracy ECII ($\alpha_2 = 1.00$) outperforms DL-Learner fast instance check ($\alpha_2 = 0.87$) version by setting $keepCommonTypes$ to $true$.

The Poker dataset has 1,368 axioms and thus a slightly larger ontology. (Lehmann and Hitzler 2010) defined two problems using this dataset namely learning the definition of a pair and of a straight. With the same (1.0) accuracy, DL-Learner is significantly quicker, presumably for similar reasons as for the Trains dataset.

The Moral Reasoner dataset has 4,666 axioms and is of medium size as part of our set of scenarios. In (Lehmann and Hitzler 2010), several versions of this scenario were explored, and we chose the one using all instances and which did not modify the original set of atomic classes which can be used to construct solutions. For this problem one of the solutions with accuracy 1.0 found by DL-Learner consists of the single atomic class "guilty," while ECII average accuracy is only 0.785. One of the solutions found by ECII, with accuracy 0.91, is ¬plan_known⊔¬careful⊔guilty. DL-Learner was significantly quicker. We see that DL-Learner outperforms ECII on this task, and the runtime performance of DL-Learner can be explained as in the Trains example. Regarding accuracy, ECII did not come up with "guilty" as candidate solution, presumably because its α_2 score is not high enough. Indeed, the ontology for this dataset contains a significant numer of class disjunctions, which causes difficulties for our approximate reasoning procedure. This seems to explain both why ECII does not produce "guilty" as a solution and also why it would construct a complicated solution using disjunctions which scores well with respect to α_2 but not well with respect to α_3-accuracy.

The ADE20k dataset contains over 20,000 images of scenes which are preclassified in terms of scene type. Each image may have several annotations of objects which have been provided by humans. For our evaluation we used the dataset from (Sarker et al. 2017) where the ADE20k dataset, with annotations, was cast into an OWL ontology and aligned with the Standard Upper Merged Ontology SUMO[4]. In (Sarker et al. 2017), sets of positive and negative exmples were selected from the images, and DL-Learner was used to generate corresponding class expressions. Only a handful of such experiments was reported in (Sarker et al. 2017): As we see below and in Table 1, the scenario yields to prohibitively long runtimes for DL-Learner, which makes a thorough investigation along the lines of (Sarker et al. 2017) impossible.

In order to compare performances, we used this scenario to create four problem classes with varying input sizes, as listed in Table 1.

For the smallest problem class, ontology size 4,714 axioms, ECII took on average 1.966 seconds to terminate and come up with solutions having average accuracy of 0.416. DL-Learner took 300 seconds to produce solutions with average accuracy of 0.926. As DL-Learner did not find solutions with accuracy 1.0 within a reasonable time span we capped execution at 10 minutes or 600 seconds. DL-Learner then took on average 577.3 seconds and the average accuracy obtained within this time was 0.926 which is the same as of 300 seconds. For one representative problem, a solution found by ECII had an accuracy of 1.0. The highest accuracy obtained by DL-Learner within this time limit on that same problem was 0.88. We notice that accuracy of ECII lags behind that of DL-Learner on average, but with this input size we already see significant runtime improvements, and sometimes even higher accuracy. ECII outperforms DL FIC interms of α_2 score.

For the medium size setting – ontology with 7,300 axioms, we used exactly the settings reported in (Sarker et al. 2017). DL-Learner was able to come up with solutions with accuracy 1.0, but the time required to produce these solutions was 983 seconds average. The runtime for ECII was 20.8 seconds on average, and the average accuracy was 0.673. Accuracy of ECII on these is less than DL-Learner but runtime is very significantly improved. Approximate accuracy is better for ECII compared to DL FIC.

Our largest test ontologies were created using attributes from all images, from the types with names starting with the

[4] http://www.adampease.org/OP/

letter "R" from the ADE20k dataset. In this case we created 2 different ontologies, in one version from the validation data – ontology size 12,193 axioms – and in the other version from the training data – ontology size 47,468 axioms.

For the validation data ontology, DL-Learner was not able to find a solution with accuracy 1.0, and we terminated DL-Learner after 4,500 seconds (i.e., 75 minutes). DL-Learner produced a solution with accuracy of 0.375. ECII took 51 seconds to run and produced soultions with an average accuracy of 0.937. In fact, it turns out that DL-Learner did produce $\exists$imageContains.$\top$ as solution, which essentially means that it did not manage to take even a few refinement steps. One solution found by ECII had an accuracy of 0.90. ECII system also outperforms DL Fast instance check (DL FIC), in terms of α_2 score. DL FIC was able to achive α_2 score as 0.375 while ECII system achives significantly higher score.

For the training data ontology, DL-Learner was also terminated after 4,500 seconds. It procuded the same solution as before with accuracy of 0.375. ECII took 116 seconds producing an average α_2-score of 0.66. Due to the size of the ontology, we were not able to run a reasoner to compute the α_3 accuracy value for the solutions provided by ECII.

From the last two tasks, we see that ECII provides a very significant runtime improvement over DL-Learner, and is in fact able to produce approximate solutions in cases where DL-Learner can only return a trivial guess such as $\top$.

Let us summarize the results using some charts. Figure 1 displays a runtime comparison over all experiments; the experiments are sorted from left to right in increasing input size. The DL-Learner curve has much higher variance, which presumably is because runtime is capped whenever a 1.0 accuracy solution is found, while runtime is significantly higher for comparable sizes if this is not the case. ECII has an algorithm which is quicker in those test cases where DL-Learner did not find a 1.0 accuracy solution, i.e., for Forte and all ADE20k experiments, and is several orders of magnitudes quicker for the large input ontologies.

Figure 2 visualizes the accuracy (α_3) comparison results. While ECII achieves 1.0 accuracy, as does DL-Learner, in some cases (Train, Poker), it usually achieves only a lower accuracy for all cases where DL-Learner is able to produce non-trivial solutions. When DL-Learner produces only a trivial result, as in ADE20k III, ECII in fact is able to do better than DL-Learner. It is reasonable to assume that this would indeed happen in many cases where DL-Learner is simply not able to process a large input size.

Figure 3 shows the approximate accuracy (α_2) comparison results. We can see that ECII always outperforms DL FIC system.

Conclusions and Future Work

When we initially set out to develop the ECII algorithm and system, our goal was to provide an alternative to DL-Learner which trades some accuracy for speed. We anticipated that our approach would only be slightly less accurate but with one or two orders of magnitude in runtime improvements.

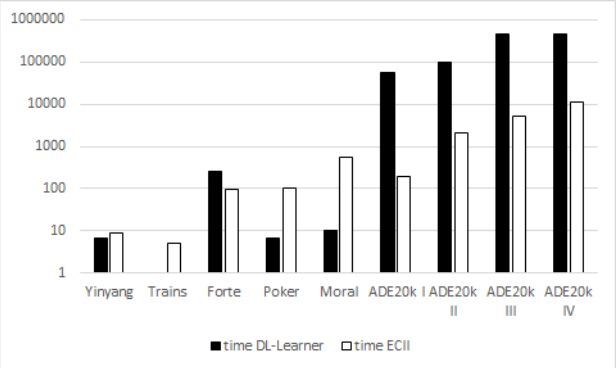

Figure 1: Runtime comparison between DL-Learner and ECII. The vertical scale is logarithmic in hundredths of seconds, and note that DL-Learner runtime has been capped at 4,500 seconds for ADE20k III and IV. For ADE20k I it was capped at each run at 600 seconds.

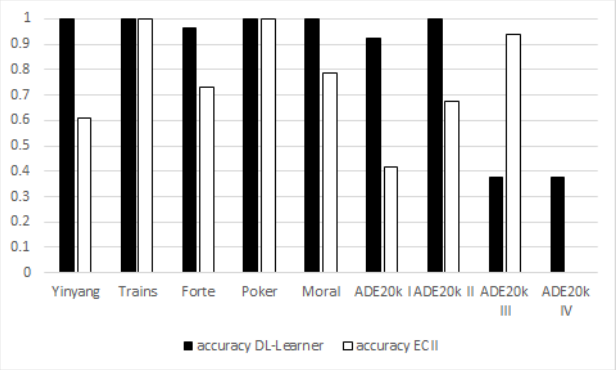

Figure 2: Accuracy (α_3) comparison between DL-Learner and ECII. For ADE20k IV it was not possible to compute an accuracy score within 3 hours for ECII as the input ontology was too large.

In a sense, our experiments show that we were not bold enough in our assumptions regarding runtime improvmements for large input ontologies, while for smaller ones there aren't any, unless in the cases where DL-Learner cannot provide an accuracy 1.0 solution. At the same time, our experiments also show that we were too optimistic regarding accuracy results for the smaller ontologies, while at the same time we see much better results in the cases where DL-Learner has to resort to trivial solution attempts.

So overall, based on the evaluation, ECII indeed seems to provide a reasonable alternative to DL-Learner in some cases, and in fact provides reasonable solutions even in cases where DL-Learner is unable to do so. From this perspective, we have achieved what we set out to do.

Our analysis of course also raises rather obvious points for further investigations and improvements. Further experiments in fact should shed further light on the strengths and

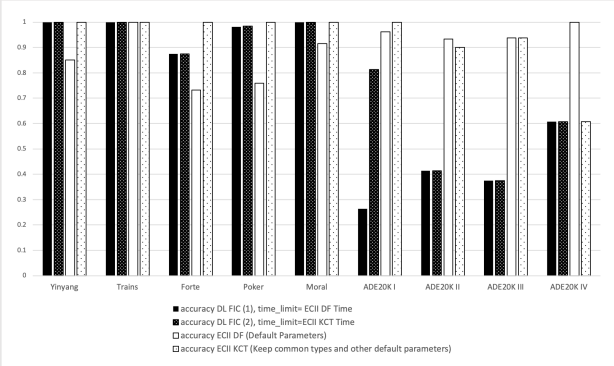

Figure 3: Approximation accuracy (α_2) comparison between DL-Learner fast instance check and ECII.

weaknesses of ECII, and this needs to be explored. We have not varied the ECII default parameters, but we conjecture that we can move the time-accuracy trade-off in both directions by changing them. We have also not yet made full use of the theoretical results for OWL EL, but only looked at the obvious evaluation datasets for a fair comparison.

As discussed in the evaluation section, it also turns out that the solutions with the highest α_2-score are not always the best ones with respect to the correct α_3-accuracy. Of course, calculating the α_3-accuracy requires a full reasoner, and thus significant time, but by doing a few such checks one may be able to improve accuracy at the expense of some of the runtime gains. As a further alternative, a post-processing step could be added to the ECII algorithm which takes a somewhat larger number of the solution candidates which perform high with respect to α_2-score, and returns only those among them which also score high on α_3-accuracy.

References

Baader, F.; Brandt, S.; and Lutz, C. 2005. Pushing the EL envelope. In Kaelbling, L. P., and Saffiotti, A., eds., *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30 – August 5, 2005*, 364–369. Professional Book Center.

Bühmann, L.; Lehmann, J.; and Westphal, P. 2016. DL-Learner – A framework for inductive learning on the semantic web. *J. Web Sem.* 39:15–24.

Hitzler, P.; Krötzsch, M.; Parsia, B.; Patel-Schneider, P.; and Rudolph, S., eds. 11 December 2012. *OWL 2 Web Ontology Language: Primer (Second Edition)*. W3C Recommendation. Available at http://www.w3.org/TR/owl2-primer/.

Hitzler, P.; Krötzsch, M.; and Rudolph, S. 2010. *Foundations of Semantic Web Technologies*. Chapman & Hall/CRC.

Horridge, M., and Bechhofer, S. 2011. The OWL API: A java API for OWL ontologies. *Semantic Web* 2(1):11–21.

Iannone, L.; Palmisano, I.; and Fanizzi, N. 2007. An algorithm based on counterfactuals for concept learning in the semantic web. *Applied Intelligence* 26(2):139–159.

Lehmann, J., and Bühmann, L. 2010. ORE – A tool for repairing and enriching knowledge bases. In Patel-Schneider, P. F.; Pan, Y.; Hitzler, P.; Mika, P.; Zhang, L.; Pan, J. Z.; Horrocks, I.; and Glimm, B., eds., *The Semantic Web – ISWC 2010 – 9th International Semantic Web Conference, ISWC 2010, Shanghai, China, November 7-11, 2010, Revised Selected Papers, Part II*, volume 6497 of *Lecture Notes in Computer Science*, 177–193. Springer.

Lehmann, J., and Hitzler, P. 2010. Concept learning in description logics using refinement operators. *Machine Learning* 78(1-2):203–250.

Lehmann, J., and Völker, J. 2014. *Perspectives on Ontology Learning*, volume 18 of *Studies on the Semantic Web*. IOS Press.

Lehmann, J.; Auer, S.; Bühmann, L.; and Tramp, S. 2011. Class expression learning for ontology engineering. *Web Semantics: Science, Services and Agents on the World Wide Web* 9(1):71–81.

Lehmann, J.; Fanizzi, N.; Bühmann, L.; and d'Amato, C. 2014. Concept learning. In Lehmann, J., and Völker, J., eds., *Perspectives on Ontology Learning*, volume 18 of *Studies on the Semantic Web*. IOS Press, Amsterdam. 71–91.

Michalski, R. S. 1980. Pattern recognition as rule-guided inductive inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (4):349–361.

Nienhuys-Cheng, S., and de Wolf, R., eds. 1997. *Foundations of Inductive Logic Programming*, volume 1228 of *Lecture Notes in Computer Science*. Springer.

Paulheim, H. 2017. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic Web* 8(3):489–508.

Richards, B. L., and Mooney, R. J. 1995. Automated refinement of first-order horn-clause domain theories. *Machine Learning* 19(2):95–131.

Sarker, M. K.; Xie, N.; Doran, D.; Raymer, M.; and Hitzler, P. 2017. Explaining trained neural networks with semantic web technologies: First steps. In Besold, T. R.; d'Avila Garcez, A. S.; and Noble, I., eds., *Proceedings of the Twelfth International Workshop on Neural-Symbolic Learning and Reasoning, NeSy 2017, London, UK, July 17-18, 2017.*, volume 2003 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Sirin, E.; Parsia, B.; Grau, B. C.; Kalyanpur, A.; and Katz, Y. Pellet: A practical OWL-DL reasoner. *J. Web Sem.* 5(2):51–53.

Zhou, B.; Zhao, H.; Puig, X.; Fidler, S.; Barriuso, A.; and Torralba, A. 2017. Scene parsing through ADE20K dataset. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, 5122–5130. IEEE Computer Society.

Modeling OWL with Rules: The ROWL Protégé Plugin

Md. Kamruzzaman Sarker[1], David Carral[1],
Adila A. Krisnadhi[1,2], and Pascal Hitzler[1]

[1] Wright State University, OH, USA
[2] Universitas Indonesia, Depok, Indonesia

Abstract. In our experience, some ontology users find it much easier to convey logical statements using rules rather than OWL (or description logic) axioms. Based on recent theoretical developments on transformations between rules and description logics, we develop ROWL, a Protégé plugin that allows users to enter OWL axioms by way of rules; the plugin then automatically converts these rules into OWL DL axioms if possible, and prompts the user in case such a conversion is not possible without weakening the semantics of the rule.

1 Motivation

It has long been argued, that rules are much more intuitive and easier to master than description logics, in terms of what their intended meaning is. We find this substantiated throughout our experiences as teachers and as ontology modelers which frequently work with domain experts.

To give just a simple example: The exact semantics behind a logical axiom such as

$$\mathsf{Journal} \sqsubseteq \forall \mathsf{publishedBy}.\mathsf{Organization}$$

in our experience often remains somewhat unclear even for people with significant exposure to ontologies and ontology modeling. On the other hand, a rule such as

$$\mathsf{Journal}(x) \wedge \mathsf{publishedBy}(x,y) \rightarrow \mathsf{Organization}(y)$$

is rather intuitive for most in its meaning, and can be both produced and processed much more readily.

The axiom and the rule just given are of course logically equivalent.[3] In fact many OWL axioms can be expressed equivalently as rules, which are, arguably, easier to understand and to produce.

As a consequence of these observations, we have produced a Protégé plugin which accepts rules as input, and adds them as OWL axioms to a given ontology, provided the rule is expressible by an equivalent set of such axioms. In case the rule is not readily transferable, the user is prompted and asked how to translate the rule, as there are different options how to do it in such cases. More information about the plugin can be found at `http://daselab.org/content/modeling-owl-rules`.

[3] When we interpret the rule in the sense of first-order predicate logic, i.e., according to the open world semantics.

2 Rules-to-OWL Transformation

In this section, we provide some examples of translations of rules into OWL axioms in an attempt to convey an intuitive understanding of our transformation. For a formal and complete of such procedure we refer the reader to [2]. Note that, as opposed to [2], we do not consider rules in our implementation that would require the use of role conjunction, as this is a logical constructor not currently allowed in OWL.

The following rule can be used to characterize all individuals taking courses and working for a department as student workers.

$$\mathsf{attends}(x,y) \wedge \mathsf{Course}(y) \wedge \mathsf{worksFor}(x,z) \wedge \mathsf{Dept}(z) \rightarrow \mathsf{StudentWorker}(x) \quad (1)$$

We transform this rule into a DL axiom via a series of equivalence preserving transformations. First, we detect that both y and z are variables that can be "rolled up," as they only occur in a single object property. Thus, these variables can be sequentially removed from the rule, resulting in the following:

$$\exists \mathsf{attends.Course}(x) \wedge \mathsf{worksFor}(x,z) \wedge \mathsf{Dept}(z) \rightarrow \mathsf{StudentWorker}(x)$$
$$\exists \mathsf{attends.Course}(x) \wedge \exists \mathsf{worksFor.Dept}(x) \rightarrow \mathsf{StudentWorker}(x)$$

Furthermore, we can unify all unary atoms of the form $C(x)$, i.e., sharing the same variable x, yielding:

$$(\exists \mathsf{attends.Course} \sqcap \exists \mathsf{worksFor.Dept})(x) \rightarrow \mathsf{StudentWorker}(x)$$

The previous rule can then be directly translated into OWL as the following axiom:

$$\exists \mathsf{attends.Course} \sqcap \exists \mathsf{worksFor.Dept} \sqsubseteq \mathsf{StudentWorker}$$

For the next example, we have the following rule, which specifies that "all mice are smaller than all elephants."

$$\mathsf{Mouse}(x) \wedge \mathsf{Elephant}(y) \rightarrow \mathsf{smallerThan}(x,y)$$

Translating such a rule into OWL requires us to first connect the variables in the body. We do so by adding atoms of the form $U(t,u)$ with U the universal property, i.e., owl:topObjectProperty.

$$\mathsf{Mouse}(x) \wedge U(x,y) \wedge \mathsf{Elephant}(y) \rightarrow \mathsf{smallerThan}(x,y)$$

The previous rule can be directly translated into OWL resulting in the following three axioms where R_{Mouse} and R_{Elephant} are fresh object properties not previously occurring in the ontology:

$$\mathsf{Mouse} \sqsubseteq \exists R_{\mathsf{Mouse}}.\mathsf{Self}$$
$$\mathsf{Elephant} \sqsubseteq \exists R_{\mathsf{Elephant}}.\mathsf{Self}$$
$$R_{\mathsf{Mouse}} \circ U \circ R_{\mathsf{Elephant}} \sqsubseteq \mathsf{smallerThan}$$

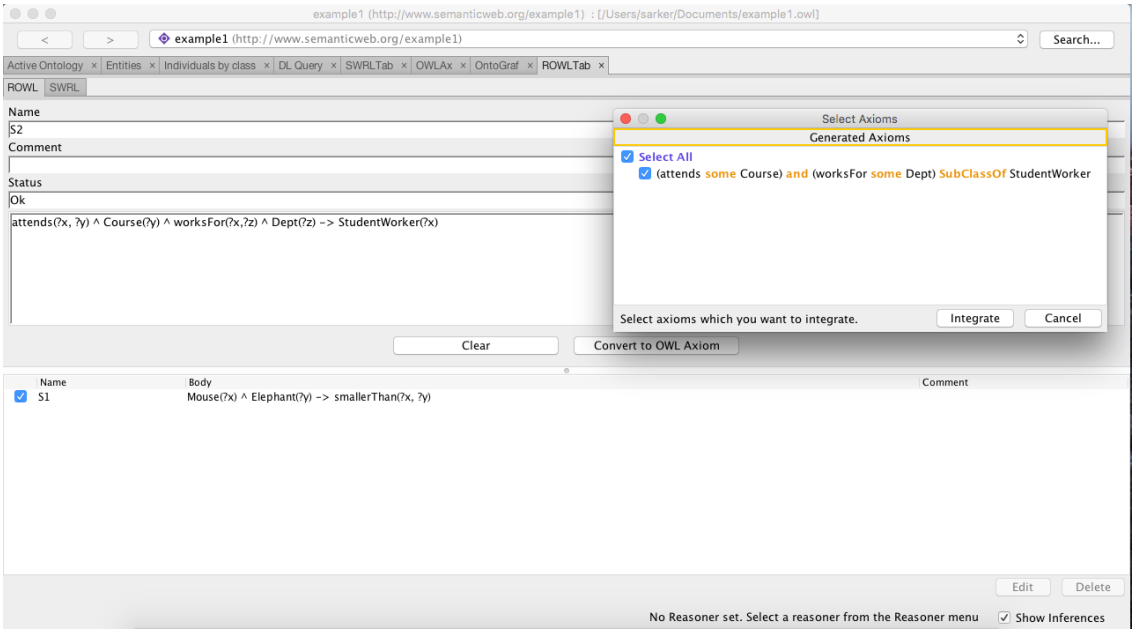

Fig. 1. The ROWL interface. The pop-up window appears after clicking "Convert to OWL Axiom" button and the transformation is successful.

Certain rules cannot be expressed in OWL employing our approach. For example, the following rule, which characterizes the set of individuals taught by their own uncle, cannot be translated by our approach.

$$\mathsf{hasFather}(x,y) \wedge \mathsf{hasBrother}(y,z) \wedge \mathsf{taughtBy}(x,z) \rightarrow \mathsf{TaughtByUncle}(x)$$

Note that, such a rule cannot be reduced in the same way as rule , since every variable occurs in at least two atoms with object properties as predicates.

In cases, such as the previous one, in which a rule cannot be translated into OWL using a set of DL axioms, our implementation will suggest several options to translate such rule using nominal schemas [1]. The chosen option by the user will be recorded in an annotation which will be added to the rule. As of right now, there is no syntax for nominal schemas in OWL and thus, we have decided that an annotation is the best way to convey such information.

3 Plugin Description and Features

Figure 1 depicts the user interface of the ROWL plugin. This plugin is implemented on top of Protégé's SWRLTab plugin implementation and thus, it borrows the pretty much SWRLTab user interface for entering rules as input. As seen in the figure, the plugin consists of two tabs: ROWL and SWRL. The latter is really SWRLTab input interface, while the former is a modification of the latter where we add "Convert to OWL Axioms" button. A user can enter a rule in ROWL tab using the standard SWRL syntax, e.g.:

$$\mathsf{attends(?x,\ ?y)\ \hat{}\ Course(?y)\ \hat{}\ worksFor(?x,\ ?z)\ \hat{}\ Dept(?z)\ \text{->}\ StudentWorker(?x)}$$

When the "Convert to OWL Axiom" button is clicked, ROWL will attempt to apply the rules-to-OWL transformation described in the previous section to the given rule. If successful, a pop-up will appear displaying one or more OWL axioms resulted from the transformation, presented in Manchester syntax. These axioms can then be integrated into the active ontology.

If the given rule cannot be transformed into OWL axiom, ROWL will prompt the user if they still want to insert the rule into the ontology as an SWRL rule with annotation. If the user agrees, ROWL will switch to its SWRL tab and proceeds in the same way as adding a rule via the original SWRLTab. Note that ROWL is separate from the original SWRLTab, hence any SWRL rule added via ROWL will not affect rules added through the original SWRLTab.

Note that once the axioms generated from rules are added into the ontology, the plugin does not provide a way to undo it and recover the original rule from which the axioms were generated. That is, when a user enters a rule through this plugin and converts it to OWL axioms, the active ontology is either augmented with the generated OWL axioms or SWRL rules. To have this feature, we need a way to record which axioms were generated from which rules. This will be considered as part of future development of this plugin.

Finally, a feature of ROWL not found in SWRLTab is the possibility to automatically add declarations for classes and properties if the inserted rule contain classes or properties not yet defined in the ontology. For example, in the rule above, the original SWRLTab requires that attends and worksFor to be already defined as object property, while Course, Dept, and StudentWorker as class in the ontology. This would add a little bit more freedom for the user to enter any rule (s)he wishes during modeling because (s)he does not need to first exit the plugin and declare the classes and properties directly in Protégé.

Acknowledgements. This work was supported by the National Science Foundation under award 1017225 *III: Small: TROn – Tractable Reasoning with Ontologies*.

References

1. Krötzsch, M., Maier, F., Krisnadhi, A., Hitzler, P.: A better uncle for OWL: nominal schemas for integrating rules and ontologies. In: Srinivasan, S., Ramamritham, K., Kumar, A., Ravindra, M.P., Bertino, E., Kumar, R. (eds.) Proceedings of the 20th International Conference on World Wide Web, WWW 2011, Hyderabad, India, March 28 – April 1, 2011. pp. 645–654. ACM (2011)
2. Martínez, D.C., Hitzler, P.: Extending description logic rules. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, Ó., Presutti, V. (eds.) The Semantic Web: Research and Applications - 9th Extended Semantic Web Conference, ESWC 2012, Heraklion, Crete, Greece, May 27-31, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7295, pp. 345–359. Springer (2012), http://dx.doi.org/10.1007/978-3-642-30284-8_30

Rule-based OWL Modeling with ROWLTab Protégé Plugin

Md. Kamruzzaman Sarker[1], Adila Krisnadhi[1,2], David Carral[3], and Pascal Hitzler[1]

[1] Data Semantics (DaSe) Laboratory, Wright State University, OH, USA
[2] Faculty of Computer Science, Universitas Indonesia, Depok, Indonesia
[3] Center for Advancing Electronics Dresden (cfaed), TU Dresden, Germany

Abstract. It has been argued that it is much easier to convey logical statements using rules rather than OWL (or description logic (DL)) axioms. Based on recent theoretical developments on transformations between rules and DLs, we have developed ROWLTab, a Protégé plugin that allows users to enter OWL axioms by way of rules; the plugin then automatically converts these rules into OWL 2 DL axioms if possible, and prompts the user in case such a conversion is not possible without weakening the semantics of the rule. In this paper, we present ROWLTab, together with a user evaluation of its effectiveness compared to entering axioms using the standard Protégé interface. Our evaluation shows that modeling with ROWLTab is much quicker than the standard interface, while at the same time, also less prone to errors for hard modeling tasks.

1 Introduction

About a decade ago, not long after description logics [1] had been chosen as the basis for the then-forthcoming W3C Recommmendation for the Web Ontology Language OWL [6], a rather agressively voiced discussion as to whether a rule-based paradigm might have been a better choice emerged in the Semantic Web community [7,19,23]. On the one hand, this eventually led to a new W3C Recommendation on the Rule Interchange Format RIF [9], based on the rules paradigm, while an alternative approach which layered rules on top of the existing OWL standard, known as SWRL [8], remained a mere W3C member submission. However, SWRL has proven significantly more popular than RIF. To see this, it may suffice to compare the Google Scholar citation numbers for SWRL – over 2500 since 2004 – and RIF – just over 50 since 2009.

At the same time, researchers kept investigating more elaborate ways to bridge between the two paradigms [10,12,13,16,20], and in particular how to convert rules into OWL [2,11,14,15]. These results regarding conversion now make it possible to express axioms first as rules, and only then to convert them into OWL. We have consistently used this approach to model complex OWL axioms throughout the last few years, as rules are, arguably, easier to understand and produce than OWL (or description logic) axioms in whichever syntax.

Consider the sentence: "If a person has a parent who is female, then this parent is a mother", which we consider to be of medium difficulty in terms of modeling it in OWL. As a first-order logic rule, this can be expressed as

$$\text{Person}(x) \wedge \text{hasParent}(x, y) \wedge \text{Female}(y) \rightarrow \text{Mother}(y).$$

This can be expressed in OWL using description logic syntax as follows:

$$\text{Female} \sqcap \exists \text{hasParent}^{-}.\text{Person} \sqsubseteq \text{Mother}$$

Based on anecdotal evidence, many people find it easier to come up with the rule than directly with the OWL axiom. Following this lead, we have produced a Protégé [18] plugin, called ROWLTab, which accepts rules as input, and adds them as OWL axioms to a given ontology, provided the rule is expressible by an equivalent set of such axioms. In case the rule is not readily convertible, the user is prompted and asked whether the rule shall be saved as SWRL rule.

In order to assess the usefulness of the ROWLTab, we have furthermore conducted a user experiment in which we compare the ROWLTab interface for adding axioms to the standard Protégé interface. Our hypotheses for the user evaluation were that given complex relationships expressed as natural language sentences as above, users will be quicker to add them to an ontology using the ROWLTab than with the standard Protégé interface, and that they will also make less mistakes in doing so. The first hypothesis has been fully confirmed by our experiment, the second has been partially confirmed.

The rest of the paper will be structured as follows. In Section 2 we explain in more detail the rule-to-OWL conversion algorithm used. In Section 3 we present the ROWLTab Protégé plugin. In Section 4 we present our user evaluation and results, and in Section 5 we conclude. More information about the plugin can be found at `http://daselab.org/content/modeling-owl-rules`. A preliminary report on the plugin, without evaluation and with much fewer details, was presented as a software demonstration at the ISWC2016 conference [21].

2 SWRL Rules to OWL Axioms Transformation

In this section we introduce theoretical notions employed across the paper. Note that, due to space constraints, some of the definitions below are simplified and may not exactly correspond with existing definitions from different sources.

Let $\mathbf{C}$, $\mathbf{R}$, $\mathbf{I}$ and $\mathbf{V}$ be pairwise disjoint, countably infinite sets of *classes*, *properties*, *individuals* and *variables*, respectively, where $\top, \bot \in \mathbf{C}$, the *universal property* $U \in \mathbf{R}$ (i.e., `owl:topObjectProperty`) and, for every $R \in \mathbf{R}$, $R^{-} \in \mathbf{R}$ and $R^{--} = R$. A *class expression* is an element of the grammar $\mathbf{E} ::= (\mathbf{E} \sqcap \mathbf{E}) \mid \exists R.\mathbf{E} \mid \exists R.\mathsf{Self} \mid C \mid \{a\}$ where $C \in \mathbf{C}$, $R \in \mathbf{R}$ and $a \in \mathbf{I}$. Furthermore, let $\mathbf{T} = \mathbf{I} \cup \mathbf{V}$ be the set of *terms*. An *atom* is a formula of the form $C(t)$ or $R(t, u)$ where $C \in \mathbf{E}$, $R \in \mathbf{R}$ and $t, u \in \mathbf{T}$. For the remainder of the paper, we identify pairs of atoms of the form $R(t, u)$ and $R^{-}(u, t)$. Furthermore, we identify a conjunction of formulas with the set containing all the formulas in the conjunction and vice-versa.

An *axiom* is a formula of the form $C \sqsubseteq D$ or $R_1 \circ \ldots \circ R_n \sqsubseteq R$ with $C, D \in \mathbf{E}$ and $R_{(i)} \in \mathbf{R}$. A *rule* is a first-order logic formula of the form $\forall \boldsymbol{x}(\beta(\boldsymbol{x}) \rightarrow \eta(\boldsymbol{z}))$ with β and η are conjunctions of atoms, $\boldsymbol{x}$ and $\boldsymbol{z}$ are non-empty sets of terms where $\boldsymbol{z} \subseteq \boldsymbol{x}$. As customary, we often omit the universal quantifier from rules. Axioms and rules are also referred to as *logical formulas*. Axioms as defined above essentially correspond to OWL 2 EL axioms [4] plus inverse property expression, while rules correspond to SWRL rules minus (in)equality and built-in atoms.

Consider some terms t and u and a conjunction of atoms β. We say that t *and* u *are directly connected in* β if both terms occur in the same atom in β. We say t *and* u *are connected in* β if there is some sequence of terms $t_1, \ldots, t_n$ with $t_1 = t$, $t_n = u$, and t_{i-1} and t_i are directly connected in β for every $i = 2, \ldots, n$.

The notions of *interpretation* and of an interpretation *entailing an axiom* follow the standard definitions for description logics [5]. For rules ρ of the form $\beta \rightarrow \eta$ we say that an interpretation $\mathcal{I}$ *entails* ρ if, for every substitution σ we have that $\mathcal{I}, \sigma \models \beta$ implies $\mathcal{I}, \sigma \models \eta$, i.e., the semantics of rules follows the standard semantics of first-order predicate logic. We say that two sets of logical formulas $\mathcal{S}$ and $\mathcal{S}'$ are *equivalent* if and only if every interpretation $\mathcal{I}$ that entails $\mathcal{S}$ also entails $\mathcal{S}'$ and vice-versa. Furthermore, we say that $\mathcal{S}'$ is a *conservative extension* of $\mathcal{S}$ if and only if (i) every interpretation that entails $\mathcal{S}'$ also entails $\mathcal{S}$ and (ii) every interpretation that entails $\mathcal{S}$ and is only defined for the symbols in $\mathcal{S}$ can be extended to an interpretation entailing $\mathcal{S}'$ by adding suitable interpretations for additional signature symbols. It is well-known that a set of logical formulas can be replaced by another set without affecting the outcome of reasoning tasks if the latter set is a conservative extension of the former.

We now formally discuss the transformation of rules into axioms. We do not include a comprehensive description of this transformation, which was introduced in [15], but only present a simplified version in an attempt to make this publication more self-contained. Specifically, our presentation makes the following assumptions about rules, all without loss of generality.

1. Rules do not contain constants. Note that, an atom of the form $R(a, b)$ (resp. $A(a)$) with $a, b \in \mathbf{I}$ in the body of a rule may be replaced by an equivalent atom $\exists U.(\{a\} \sqcap \exists R.\{b\})(x)$ (resp. $\exists U.(\{a\} \sqcap A)(x)$) where x is any arbitrarily chosen variable occurring in the body of the rule. Furthermore, atoms of the form $R(x, a)$ with $x \in \mathbf{V}$ and $a \in \mathbf{I}$ occurring in the body may be replaced by $\exists R.\{a\}(x)$. Similar transformations may be applied to the head of a rule in order to remove all occurrences of constants.
2. The head of a rule is of the form $C(x)$ or $S(x, y)$ with $C \in \mathbf{E}$, $S \in \mathbf{R}$ and $x, y \in \mathbf{V}$. Note that, a rule of the form $\beta \rightarrow \eta$ is equivalent to a set of rules $\{\beta \rightarrow \eta_1, \ldots, \beta \rightarrow \eta_n\}$ provided that $\eta = \eta_1 \cup \ldots \cup \eta_n$.
3. All of the variables in the body of a rule are connected. If two variables x and y are not connected in the body of a rule, we may simply add the atom $U(x, y)$ to the body of the rule resulting in a semantically equivalent rule.

The preprocessing implied by the aforementioned assumptions has been implemented in the ROWLTab plugin and thus, such constraints need not be con-

sidered by the end users. Moreover, such preprocessing is carefully implemented in an attempt to minimize the number of necessary modifications for a rule to satisfy assumptions (1-3). We now proceed with the definition of our translation.

Definition 1. *Given some rule $\rho = \beta \to \eta$, let $\delta(\beta \to \eta)$ be the rule that results from exhaustively applying transformations (1-3) where (1) and (2) should be applied with higher priority than (3).*

1. *Replace every atom of the form $R(x, x)$ in β or in η with $\exists R.\mathsf{Self}(x)$.*
2. *Replace every maximal subset of the form $\{C_1(y) \ldots, C_n(y)\} \subseteq \beta$ with the atom $C_1 \sqcap \ldots \sqcap C_n(y)$.*
3. *For every variable y not occurring in η that occurs in exactly one binary atom in β of the form $R(z, y)$, do the following:*
   - *If there is some atom of the form $C(y) \in \beta$, then replace the atoms $R(z, y)$ and $C(y)$ in β with the atom $\exists R.C(z)$.*
   - *Otherwise, replace the atom $R(z, y)$ in β with $\exists R.\top(z)$.*

Example 1. Consider the rule $\rho = \text{Person}(x) \wedge \text{hasParent}(x, y) \wedge \text{Female}(y) \to \text{Mother}(y)$. Then, the transformation presented in the previous definition would sequentially produce the following sequence of rules

$$(\exists \text{hasParent}^{-}.\text{Person})(y) \wedge \text{Female}(y) \to \text{Mother}(y)$$
$$(\exists \text{hasParent}^{-}.\text{Person} \sqcap \text{Female})(y) \to \text{Mother}(y)$$

Rule $\delta(\rho)$ from the previous example can be directly transformed into an axiom as indicated in the following lemma.

Lemma 1. *Consider some rule ρ. If $\delta(\rho)$ is of the form $C(x) \to D(x)$, then ρ is equivalent to the axiom $C \sqsubseteq D$.*

Proof. Let υ and υ' be some rules such that υ' results by applying some of the transformations (1-3) introduced in Definition 1 to υ. Note that, by definition, we can conclude equivalency between υ and υ'. Thus, we can show via induction that ρ is equivalent to $\delta(\rho)$. Furthermore, if $\delta(\beta \to \rho)$ is of the form $C(x) \to D(x)$, then, by the definition of the semantics of rules and axioms, $C \sqsubseteq D$ is equivalent to $\delta(\beta \to \rho)$. Since the equivalence relation is transitive, we can conclude that ρ is equivalent to $C \sqsubseteq D$.

As indicated by Lemma 1, rule ρ from Example 1 is equivalent to the axiom $\exists \text{hasParent}^{-}.\text{Person} \sqcap \text{Female} \sqsubseteq \text{Mother}$.

Lemma 2. *Consider some rule ρ. If the rule $\delta(\rho)$ is of the form $\bigwedge_{i=2}^{n}(C_i(x_{i-1}) \wedge R_i(x_{i-1}, x_i)) \wedge C_n(x_n) \to S(x_1, x_n)$, then the set of axioms $\{C_i \sqsubseteq \exists R_{C_i}.\mathsf{Self} \mid i = 1, \ldots, n\} \cup \{R_{C_1} \circ R_1 \circ \ldots \circ R_{C_{n-1}} \circ R_n \circ R_{C_n} \sqsubseteq S\}$ where all R_{C_i} are fresh properties unique for every class C_i is a conservative extension of the rule ρ.*

Proof. As shown in proof of Lemma 1, rules ρ and $\delta(\rho)$ are indeed equivalent. Thus, the lemma follows from the fact that the set of rules presented in the statement of the lemma is a conservative extension of $\delta(\rho)$.

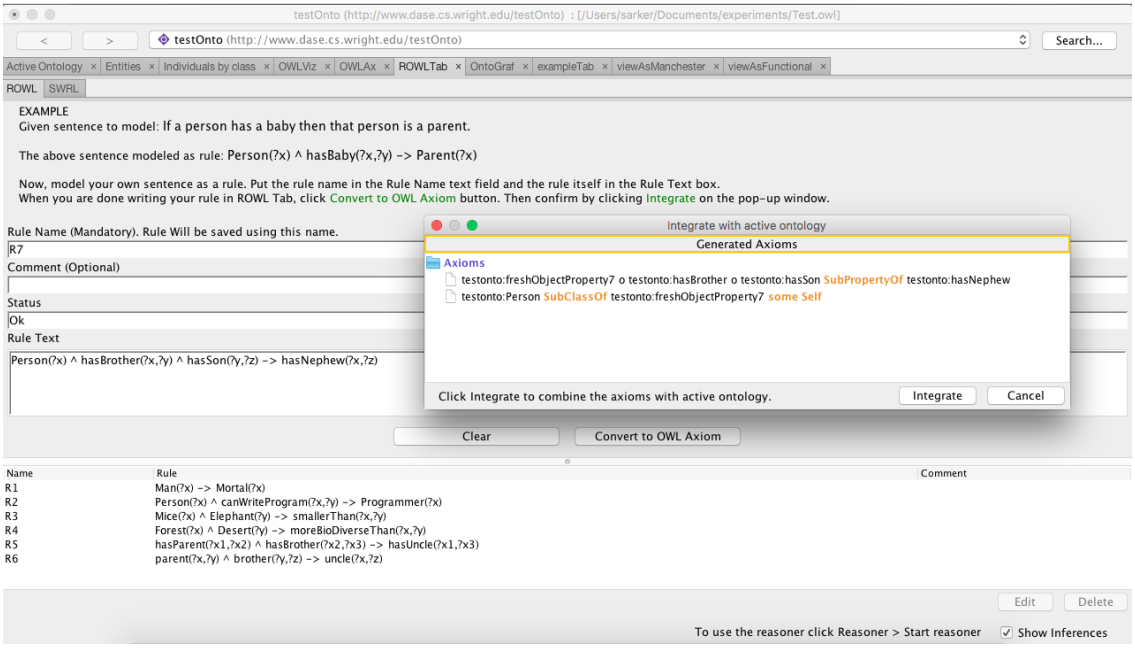

Fig. 1. The ROWLTab interface with generated axioms.

We finalize the section with some brief comments about the presented transformation. First, the transformation is sound, but has not been proven to be complete. That is, there may be some rules which are actually expressible as axioms, but cannot be handled by our translation algorithm. Moreover, the transformation of a given rule may in some cases produce axioms that violate some of the global syntactic restrictions of OWL 2 DL, such as regularity restriction on property inclusion/hierarchy, when added to an ontology.

3 Plugin Description and Features

Figure 1 shows the user interface of the ROWLTab plugin with generated axioms from a rule and also shows previously saved rules in the bottom part of the user interface. As seen in the figure, the plugin consists of two tabs: ROWL and SWRL. The latter is really the SWRLTab input interface, while the former is our implementation of rule-to-OWL conversion functionality. We have in fact reused the source code of the SWRLTab, kept its functionality intact and added extra functionality by means of the ROWL tab. The upper part of the interface is for rule insertion and the bottom part is for rule modification. At the top a modeling example is also shown.

A user can enter a rule in the "Rule Text" input box using the standard SWRL syntax, e.g.

```
Person(?x) ^ hasChild(?x,?y) ^ Female(?y) -> hasDaughter(?x,?y)
```

Every rule needs a distinct name to be eligible to be saved for later modification. A suggested rule name is automatically generated but the user also has the option

to change the rule name; this can be done in the "Rule Name" input box. The user can also give annotations to a rule in the "Comment" input box. The plugin does syntax checking of the rule, but nothing more sophisticated, e.g. tautologies can also be entered, and checks for global constraints like RBox regularity, which are required for the ontology to stay within OWL 2 DL, are not performed. Since rule-to-OWL conversion often results in the use of property chains, extra care is needed by the modeler to ensure compliance, if compliance is desired; this can also be checked by using a reasoner such as HermiT [17] from within Protégé. When the user is writing a rule it checks whether a predicate is already declared or not. If not declared it will show that the predicate is invalid and the user needs to declare it before the rule can be converted to OWL. Auto completion of predicate names is also supported. The user can use tab-key for auto-completion to existing class, object properties, data properties and individual names.

Figure 1 also shows a button "Convert to OWL Axiom" below the "Rule Text" input box. This button is initially deactivated and if the inserted rule is syntactically correct then this button becomes clickable. When the "Convert to OWL Axiom" button is clicked, ROWLTab will attempt to apply the rule-to-OWL transformation described in the previous section to the given rule. If successful, a pop-up will appear displaying one or more OWL axioms resulting from the transformation, presented in Manchester syntax. These axioms can then be integrated into the active ontology by clicking the "Integrate" button of the pop-up interface. If the given rule cannot be transformed into OWL axioms, ROWLTab will prompt the user if (s)he still want to insert the rule into the ontology as a SWRL rule. If the user agrees, ROWLTab will switch to its SWRL tab and proceed in the same way as adding a rule via the original SWRLTab.

As described in Section 2, the translation of a rule into OWL axioms may sometimes require the introduction of fresh object properties, which will be automatically created by the plugin when necessary. The namespace for these fresh object properties is taken from the default namespace. To create a unique fresh object property, the plugin counts the number of existing fresh object properties in the active ontology (including imports) then increments the counter by 1 and creates the new object property with the incremented counter as part of its identifier.

Once the axioms generated from a rule are added to the ontology, the rule will be saved with the ontology for later modification; the rule is in fact added as an annotation to every OWL axiom generated by the rule. Figure 1 shows saved rules displayed on the bottom left of the ROWLTab plugin. A user can modify or delete rules at any time. If a rule is modified or deleted, the axioms generated by that rule will be affected. That means if a rule is deleted then the axioms generated by the rule will also be deleted. To edit a rule which was previously used to generate axioms the user needs to select the rule first and then click the "Edit" button at the bottom right part of the interface. The rule will then appear in the "Rule Text" input box for modification. The user can also double click on the rule to edit that rule. To delete a rule, the user needs to select that rule and then click the "Delete" button on the bottom right of the interface.

A feature of ROWLTab not found in the SWRLTab is the possibility to automatically add declarations for classes and properties if the inserted rule contains classes or properties not yet defined in the ontology. For example, in the rule above, the original SWRLTab requires that hasChild and hasDaughter be already defined as object properties, and Person and Female as classes in the ontology. This means that the user does not need to first exit the plugin and declare the classes and properties outside the ROWLTab.

Another feature of the ROWLTab plugin is that it actually works as a superset of SWRLTab plugin. So if a user need to work with the SWRLTab plugin, the user does not need to install the SWRLTab plugin separately, as the ROWLTab contains a full instance of the SWRLTab plugin. If the ROWLTab plugin is installed the user only need to switch the tab from ROWL to SWRL to get the full SWRLTab functionality. This also creates a limitation that when a new version of SWRLTab is available the developer of ROWLTab has to embed the newer version of SWRLTab explicitly.

To manage the source code and to be able to modify the source code efficiently in the future we have separated the view module from the control module. The view module consist of the user interface and the control module implements the rule-to-OWL transformation. Besides those two modules we have a separate listener module, which acts as a bridge between the view and controller module. We have used Maven as our build system to easily manage the dependency of various APIs. This plugin is open source and the source code is available at the DaseLab website (`http://dase.cs.wright.edu/content/modeling-owl-rules`).

4 Evaluation

For the evaluation of the ROWLTab plugin, we conducted a user evaluation to answer the following three questions:

- Is writing OWL axioms into Protégé via the ROWLTab plugin quicker than writing them directly through the standard Protégé interface?
- Is writing OWL axioms into Protégé via ROWLTab plugin less error-prone than writing them directly through the standard Protégé interface?
- Do users view modeling OWL axioms via ROWLTab plugin to be an easier task than directly through the standard Protégé interface?

The evaluation was conducted by asking the participants to model a set of natural language sentences as rules using the ROWLTab or as OWL axioms using the standard Protégé interface. We recorded time and number of keyboard and mouse clicks required for each question (see Section 4.1) and also recorded the responses which we subsequently assessed for correctness (see Section 4.2). Finally, the participants answered a brief questionnaire (see Section 4.3).

Before describing the experiment in detail, we would like to encourage the readers to do it themselves; it should take less than an hour, the software can be obtained from the ROWLTab website already indicated.

Table 1. Evaluation Questions

Group A	Group B	Difficulty
1. Every father is a parent.	7. Every parent is a human.	easy
2. Every university is an educational institution.	8. Every educational institution is an organization.	
3. If a person has a mother then that mother is a parent.	9. If a person has a parent who is female, then this parent is a mother.	medium
4. Any educational institution that awards a medical degree is a medical school.	10. Any university that is funded by a state government is a public university.	
5. If a person's brother has a son, then that son is the first person's nephew.	11. If a person has a female child, then that person would have that female child as her daughter.	hard
6. All forests are more biodiverse than any desert.	12. All teenagers are younger than all twens.	

For the experiment we recruited 12 volunteers from among the graduate students at Wright State University. Our sole selection criterion was that the participants had at least some basic knowledge of OWL, and had at least minimal exposure to Protégé. All participants were then given a half-hour briefing in which we explained, by means of examples, how to model natural language sentences with and without the ROWLTab in Protégé.

Each participant was given the same twelve natural language sentences to model. The sentences are listed in Table 1 where group A consists of sentence 1 to 6 and group B consists of sentence 7 to 12. As indicated in the table, each group contains two easy, two medium, and two hard sentences to model. Each participant modeled one of the sets of sentences using the ROWLTab and the other group without using the ROWLTab, and we randomly assigned whether the participant will model Group A using the ROWLTab or Group B using the ROWLTab. In order to minimize learning effects which may come from different sentences, we made sure that for each sentence in Group A there is a very similar sentence in Group B, and vice-versa: Each sentence number n in Group A corresponds to sentence number $n+6$ in Group B. We furthermore randomized whether the participant will first model using ROWLTab, and then without the ROWLTab, or vice-versa, also to control for a possible learning effect during the course of the experiment. There was no time limit for the modeling; participants were informed that it should usually take no longer than an hour to model all twelve sentences. Participants were also informed that they cannot go back to earlier sentences during the course of the experiment.

Our categorization into easy, medium, and hard sentences was done as follows: Easy sentences expressed simple subclass relationships. Medium sentences required the use of property restrictions to model them in OWL; the medium sentence 9 was discussed in Section 1. Hard sentences could only be expressed using two or three OWL axioms, together with a technique called *rolification* [11,14].

Table 2. Average and standard deviation of time (in seconds), number of clicks (keyboard and mouse), and correctness score per difficulty category of sentences.

Sentence Category	Time (in secs)		# clicks		Correctness	
	Protégé avg/std	ROWL avg/std	Protégé avg/std	ROWL avg/std	Protégé avg/std	ROWL avg/std
easy	79/ 41	47/ 9	44/ 38	59/ 19	2.9/0.3	2.9/0.3
medium	312/181	116/61	216/131	141/ 91	2.2/0.5	2.5/0.8
hard	346/218	160/66	351/318	228/168	0.9/0.7	2.5/0.7

For example, sentence 5 when expressed as a rule becomes

$$\text{Person}(x) \wedge \text{hasBrother}(x, y) \wedge \text{hasSon}(y, z) \rightarrow \text{hasNephew}(x, z).$$

In order to express this sentence as OWL axioms, one first has to *rolify* the class Person by adding the axiom $\text{Person} \sqsubseteq \exists R_{\text{Person}}.\text{Self}$, where R_{Person} is a fresh property name, and to then add the property chain axiom

$$R_{\text{Person}} \circ \text{hasBrother} \circ \text{hasSon} \sqsubseteq \text{hasNephew}.$$

We informed all the participants regarding the total number of easy, medium, and hard sentences the participants would face. With each sentence, we also displayed the suitable class and property-names which had been pre-defined by us, i.e. the participants did not have to declare them in Protégé, and directed participants to use the displayed class and property names to the maximum extent possible. For example, the pre-defined classes and properties for sentence 9 were Person, hasParent, Female, Mother, while for sentence 5 they were Person, hasBrother, hasSon, hasNephew. An exception to this is when modeling the hard sentences (5, 6, 11, and 12) via standard Protégé. Those sentences contain class names that need to be rolified, which necessitates one to declare one or more fresh object properties. In this case, we informed the participants that the hard sentences may require them to declare additional object properties without disclosing that this is due to rolification.

4.1 Time Used For Modeling

Our hypothesis was that, on medium and hard sentences, participants would be able to model quicker with the ROWLTab than without it. Cumulated data is given in Table 2.

For the statistical analysis, our null hypothesis was that there is no difference between the time taken with ROWLTab versus Protégé. Since each participant had modeled sentences from each difficulty class, we could perform a paired (two-tailed) t-test – note that assuming normal distributions appears to be perfectly reasonable for this data. For the medium sentences the null hypothesis was rejected with $p \approx 0.002 < 0.01$. For the hard questions the null hypothesis was rejected with $p \approx 0.020 < 0.05$. Both results are statistically significant with $p < 0.05$, thus confirming our hypotheses.

Interestingly, if we run the same t-test also on the easy sentences, the same null hypothesis is also rejected with $p \approx 0.019 < 0.05$. We will reflect on this further below.

In order to aid us in interpreting the results, we also recorded the number of clicks (keyboard plus mouse) required for modeling each sentence; cumulative data is provided also in Table 2. If we run the number of clicks through the same t-test as before, the null hypothesis being that there is no difference between the two interfaces. The corresponding p-values are 0.092 (easy), 0.030 (medium) and 0.173 (hard), i.e. we have $p < 0.05$ only for the medium sentences. The click analysis may provide us with a partial answer to the better performance of the ROWLTab regarding time used: Fewer clicks may in this case simply translate into less time required. However, this observation does not explain the data for easy and hard questions. We will return to this discussion at the end of Section 4.2, after we have looked at answer correctness.

4.2 Correctness of Modeled Axioms

Our hypothesis was that for medium and hard questions, participants would provide more correct answers with the ROWLTab than without it. To see if we can confirm this hypothesis, we verify the correctness of the axioms in the OWL files obtained from the participants. The correct set of axioms for each modeling question is given in Table 3. Since the sentences are short and the resulting OWL axioms are relatively simple, the verification was done manually. Also, for modeling tasks where the participants were asked to model the sentence via ROWLTab, we check the correctness of the rules by examining the OWL axioms obtained after translation, which are annotated with information regarding the actual rule input given to ROWLTab. We then assign a score of 0, 1, 2, or 3 to each answer from the participants as follows:

- Modeling a sentence via ROWLTab, the score is:
  - 3 if the participant's rule is fully correct (equivalent to the answer key),
  - 2 if the participant's mistakes are only in the incorrect use of variables (wrong placement, missing/spurious variables), i.e., the rule still employs the correct predicates in the rule body and head and no spurious predicates are used,
  - 1 if there's a missing predicate in the participant's rule or spurious predicates are used that makes the rule not equivalent to the correct answer,
  - 0 if the participant provides no answer.
- Modeling a sentence via the standard Protégé interface, the score is:
  - 3 if the participant's OWL axioms are fully correct,
  - 2 if the participant's OWL axioms employ the correct set of class and property names, but there is a mistake in the use of logical constructs,
  - 1 if there is a missing or spurious class names or property names, or even missing some necessary OWL axioms,
  - 0 if the participant provides no answer.

Table 3. Answers to Evaluation Questions – Rui and Axi are answers for question i in the form of rule and OWL axioms, resp. where $R_1, \ldots, R_7$ are fresh (object) properties generated due to rolification

Ru1: $\text{Father}(x) \rightarrow \text{Parent}(x)$ Ax1: $\text{Father} \sqsubseteq \text{Parent}$
Ru2: $\text{University}(x) \rightarrow \text{EducationalInstitution}(x)$
Ax2: $\text{University} \sqsubseteq \text{EducationalInstitution}$
Ru3: $\text{Person}(x) \wedge \text{hasMother}(x, y) \rightarrow \text{Parent}(y)$
Ax3: $\exists \text{hasMother}^{-}.\text{Person} \sqsubseteq \text{Parent}$
Ru4: $\text{EducationalInstitution}(x) \wedge \text{awards}(x, y) \wedge \text{MedicalDegree}(y) \rightarrow \text{MedicalSchool}(x)$
Ax4: $\text{EducationalInstitution} \sqcap \exists \text{awards}.\text{MedicalDegree} \sqsubseteq \text{MedicalSchool}$
Ru5: $\text{Person}(x) \wedge \text{hasBrother}(x, y) \wedge \text{hasSon}(y, z) \rightarrow \text{hasNephew}(x, z)$
Ax5: $\text{Person} \sqsubseteq \exists R_1.\mathsf{Self}$, $\text{R}_1 \circ \text{hasBrother} \circ \text{hasSon} \sqsubseteq \text{hasNephew}$
Ru6: $\text{Forest}(x) \wedge \text{Desert}(y) \rightarrow \text{moreBiodiverseThan}(x, y)$
Ax6: $\text{Forest} \sqsubseteq \exists R_2.\mathsf{Self}$, $\text{Desert} \sqsubseteq \exists R_3.\mathsf{Self}$, $R_2 \circ U \circ R_3 \sqsubseteq \text{moreBiodiverseThan}$
Ru7: $\text{Parent}(x) \rightarrow \text{Human}(x)$ Ax7: $\text{Parent} \sqsubseteq \text{Human}$
Ru8: $\text{EducationalInstitution}(x) \rightarrow \text{Organization}(x)$
Ax8: $\text{EducationalInstitution} \sqsubseteq \text{Organization}$
Ru9: $\text{Person}(x) \wedge \text{hasParent}(x, y) \wedge \text{Female}(y) \rightarrow \text{Mother}(x)$
Ax9: $\text{Person} \sqcap \exists \text{hasParent}.\text{Female} \sqsubseteq \text{Mother}$
Ru10: $\text{University}(x) \wedge \text{fundedBy}(x, y) \wedge \text{StateGovernment}(y) \rightarrow \text{PublicUniversity}(x)$
Ax10: $\text{University} \sqcap \exists \text{fundedBy}.\text{StateGovernment} \sqsubseteq \text{PublicUniversity}$
Ru11: $\text{Person}(x) \wedge \text{hasChild}(x, y) \wedge \text{Female}(y) \rightarrow \text{hasDaughter}(x, y)$
Ax11: $\text{Person} \sqsubseteq \exists R_4.\mathsf{Self}$, $\text{Female} \sqsubseteq \exists R_5.\mathsf{Self}$, $R_4 \circ \text{hasChild} \circ R_5 \sqsubseteq \text{hasDaughter}$
Ru12: $\text{Teenager}(x) \wedge \text{Twen}(y) \rightarrow \text{youngerThan}(x, y)$
Ax12: $\text{Teenager} \sqsubseteq \exists R_6.\mathsf{Self}$, $\text{Twen} \sqsubseteq \exists R_7.\mathsf{Self}$, $R_6 \circ U \circ R_7 \sqsubseteq \text{youngerThan}$

The average and standard deviation of the correctness score for easy, medium, and hard questions can be found in Table 2. Here, our null hypothesis was that there is no difference in correctness of the answers given with ROWLTab versus Protégé. For the same reasons as before, we thus performed a paired (two-tailed) t-test, the null hypothesis being that there be no difference whether using ROWLTab or not. For the medium sentences we obtained $p \approx 0.18 > 0.05$, so the null hypothesis could not be rejected. But for the hard questions the null hypothesis was rejected with $p \approx 0.0001 < 0.01$. The latter result is statistically significant with $p < 0.01$. If we run the same t-test also on the easy sentences, the same null hypothesis cannot be rejected; we in fact obtain $p \approx 1.0000$.

We thus confirm our hypothesis that ROWLTab helps users in modeling hard sentences correctly; however we could not confirm this for medium sentences on this population of participants. It could be hypothesized that the participants

were sufficiently familiar with Protégé to perform well on medium difficulty, thus use of the ROWLTab only had an effect on time used, as shown in Section 4.1.

At the same time, the correctness analysis also sheds further light on the hard questions: While participants used less time for these on the ROWLTab, they did not use significantly fewer clicks; however answer correctness was much higher on the ROWLTab. This seems to indicate that the additional time using Protégé was spent thinking (and indeed, rather unsuccessfully) about the problem, while this additional thinking was not required when using the ROWLTab.

4.3 Participant Survey

We finally used a questionnaire with four questions to assess the subjective value which the use of the ROWLTab had to the participants. For this, we asked all participants to indicate to what extent they agree with each of the following statements.

1. ROWLTab is a useful tool to help with ontology modeling.
2. Modeling rules with ROWLTab was easier for me than modeling without it.
3. Given some practice, I think I will find modeling rules with the ROWLTab easier than modeling without it.
4. The ROWLTab is better for ontology modeling than the SWRLTab.

Participants were asked to click, on screen whether they agree with each statement, on a scale from -3 (strongly disagree) to +3 (strongly agree). It turns out that participants agreed highly with all three statements:

Question Number	mean	standard deviation
1 (ROWL is a useful tool.)	2.83	0.39
2 (ROWL makes modeling easier.)	3.00	0.00
3 (Modeling with ROWL easier with some practice.)	2.75	0.45
4 (ROWLTab better than SWRLTab)	1.75	1.22

In assessing these responses, we need to be aware that the pool of participants came from the investigators' institution, and many of them were either associated with the investigators' lab or had attended classes by one of the investigators. Hence the scores should be interpreted with caution. Nevertheless, the scores for the first three questions indicate strong agreement with the usefulness of the ROWLTab.

Regarding the fourth question, it should be noted that our briefing did not include a briefing on the SWRLTab. As discussed in Section 3, the user interaction of the ROWLTab is very similar to that of the SWRLTab, so the only substantial difference would be in the fact that the ROWLTab produces OWL axioms, while the SWRLTab produces SWRL axioms with a different semantics. We do not know to what extent the participants were aware of this difference. A quarter of the participants answered this question with "0" (neutral). Results of the experiment can be found at `http://dase.cs.wright.edu/content/rowl` and raw result can be found at `https://github.com/md-k-sarker/ROWLPluginEvaluation/tree/master/results` .

Table 4. Summary of evaluation results. Entries indicate whether the difference between using the ROWLTab and not using it were statistically significant.

category	time	clicks	correctness
easy	significant ($p < 0.05$)	not significant	not significant
medium	significant ($p < 0.01$)	significant ($p < 0.05$)	not significant
hard	significant ($p < 0.05$)	not significant	significant ($p < 0.01$)

5 Conclusions and Further Work

We have presented the Protégé ROWLTab plugin for rule-based OWL modeling in Protégé, and its underlying algorithms. We have furthermore reported on a user evaluation for assessing the improvements arising from the use of ROWLTab.

The evaluation results are summarized in Table 4: We have a significant time improvement in all three categories (it was hypothesized by us only for medium and hard sentences). In the medium category, where answer correctness was not significantly different, the ROWLTab required significantly less clicks. In the hard category, the difference in answer correctness was also significant.

The evaluation results are rather encouraging, and we also already received direct feedback from users that the ROWLTab is considered very useful. But while basic functionality is already in place, we already see further improvements that can be made to the plugin:

- When rolification is used for the transformation of a rule to OWL, the ROWLTab currently invents an artificial property name for the fresh object property. It may be helpful to more directly support a renaming of these properties, or to come up with a standard naming scheme for properties arising out of rolification. Note, however, that it is not sufficient to have one fresh property for each defined atomic class, as in some cases complex classes need to be rolified [11].
- The translation of rules into OWL often leads to the use of property chains, which may result in a non-regular property hierarchy, thus violating a global syntactic restriction of OWL 2 DL. While standard tools such as reasoners, which can be called from within Protégé, can detect this issue, it may be helpful to catch this earlier, e.g. directly at the time when a rule is translated.
- Currently, if a rule is input which cannot be translated to OWL, it is simply saved as a SWRL rule, i.e., with a significantly modified (and, in a sense, restricted) semantics. However, through the use of so-called *nominal schemas* [14] it is possible to recover more of the first-order semantics of the input rules, and it has even been shown that the use of such nominal schemas can lead to performance improvements of reasoners compared to SWRL [22].

More substantial possible future work would carry the ROWLTab theme beyond the basic rule paradigm currently supported:

- The rule syntax could be extended to allow for capturing OWL features which cannot be expressed by means of the basic rules currently supported. In particular, these would be right-hand side (head) disjunctions and existentials as well as cardinality restrictions, as well as left-hand side universal quantifiers. It would even be conceivable to add additonal shortcut notation, e.g. for witnessed universals [3], or for nominal schemas [14].

– The development of a full-blown rule syntax for all of OWL 2 DL would then also make it possible to perform all ontology modeling using rules, i.e., to establish an interface where the user would get a pure rules view on the ontology, if desired.

We are looking forward to feedback by ontology modelers on the route which we should take with the plugin in the future.

Acknowledgements. This work was supported by the National Science Foundation under award 1017225 *III: Small: TROn – Tractable Reasoning with Ontologies* and the German Research Foundation (DFG) within the Cluster of Excellence "Center for Advancing Electronics Dresden" (cfaed). We would also like to thank Tanvi Banerjee and Derek Doran for some advise on statistics.

References

1. Baader, F., et al. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, 2nd edn. (2010)
2. Carral, D., Hitzler, P.: Extending description logic rules. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, Ó., Presutti, V. (eds.) The Semantic Web: Research and Applications - 9th Extended Semantic Web Conference, ESWC 2012, Heraklion, Crete, Greece, May 27-31, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7295, pp. 345–359. Springer (2012), http://dx.doi.org/10.1007/978-3-642-30284-8_30
3. Carral, D., Krisnadhi, A., Rudolph, S., Hitzler, P.: All but not nothing: Left-hand side universals for tractable OWL profiles. In: Keet, C.M., Tamma, V.A.M. (eds.) Proceedings of the 11th International Workshop on OWL: Experiences and Directions (OWLED 2014), Riva del Garda, Italy, October 17-18, 2014. CEUR Workshop Proceedings, vol. 1265, pp. 97–108. CEUR-WS.org (2014)
4. Cuenca Grau, B., Motik, B., Wu, Z., Fokoue, A., Lutz, C.: OWL 2 Web Ontology Language Profiles (Second Edition). W3C Recommendation (11 December 2012), http://www.w3.org/TR/owl2-profiles/
5. Hitzler, P., Krötzsch, M., Rudolph, S.: Foundations of Semantic Web Technologies. CRC Press/Chapman & Hall (2010)
6. Hitzler, P., et al. (eds.): OWL 2 Web Ontology Language Primer (Second Edition). W3C Recommendation (11 December 2012), http://www.w3.org/TR/owl2-primer/
7. Horrocks, I., Parsia, B., Patel-Schneider, P.F., Hendler, J.A.: Semantic web architecture: Stack or two towers? In: Fages, F., Soliman, S. (eds.) Principles and Practice of Semantic Web Reasoning, Third International Workshop, PPSWR 2005, Dagstuhl Castle, Germany, September 11-16, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3703, pp. 37–41. Springer (2005)
8. Horrocks, I., et al.: SWRL: A Semantic Web Rule Language Combining OWL and RuleML. W3C Member Submission (21 May 2004), http://www.w3.org/Submission/SWRL/
9. Kifer, M., Boley, H. (eds.): RIF Overview (Second Edition). W3C Working Group Note (5 February 2013), https://www.w3.org/TR/rif-overview/
10. Knorr, M., Hitzler, P., Maier, F.: Reconciling OWL and non-monotonic rules for the semantic web. In: Raedt, L.D., et al. (eds.) ECAI 2012 – 20th European Conference on Artificial Intelligence, Montpellier, France, August 27-31 , 2012. pp. 474–479. IOS Press (2012)

11. Krisnadhi, A., Maier, F., Hitzler, P.: OWL and rules. In: Polleres, A., et al. (eds.) Reasoning Web. Semantic Technologies for the Web of Data - 7th International Summer School 2011, Galway, Ireland, August 23-27, 2011, Tutorial Lectures. Lecture Notes in Computer Science, vol. 6848, pp. 382–415. Springer (2011)
12. Krötzsch, M.: Description Logic Rules, Studies on the Semantic Web, vol. 8. IOS Press (2010)
13. Krötzsch, M., Hitzler, P., Vrandecic, D., Sintek, M.: How to reason with OWL in a logic programming system. In: Eiter, T., et al. (eds.) Proceedings of the Second International Conference on Rules and Rule Markup Languages for the Semantic Web, RuleML2006. pp. 17–26. IEEE Computer Society, Athens, Georgia (2006)
14. Krötzsch, M., Maier, F., Krisnadhi, A., Hitzler, P.: A better uncle for OWL: nominal schemas for integrating rules and ontologies. In: Srinivasan, S., et al. (eds.) Proceedings of the 20th International Conference on World Wide Web, WWW 2011, Hyderabad, India, March 28 - April 1, 2011. pp. 645–654. ACM (2011)
15. Krötzsch, M., Rudolph, S., Hitzler, P.: Description Logic Rules. In: Ghallab, M., et al. (eds.) Proceeding of the 18th European Conference on Artificial Intelligence, Patras, Greece, July 21-25, 2008. vol. 178, pp. 80–84. IOS Press, Amsterdam, The Netherlands (2008)
16. Krötzsch, M., Rudolph, S., Hitzler, P.: ELP: Tractable Rules for OWL 2. In: Sheth, A.P., et al. (eds.) Proceedings of the 7th International Semantic Web Conference, ISWC 2008, Karlsruhe, Germany, October 26-30, 2008. Lecture Notes in Computer Science, vol. 5318, pp. 649–664. Springer (2008)
17. Motik, B., Shearer, R., Horrocks, I.: Hypertableau Reasoning for Description Logics. Journal of Artificial Intelligence Research 36, 165–228 (2009)
18. Musen, M.A.: The Protégé project: A look back and a look forward. AI Matters 1(4), 4–12 (2015)
19. Patel-Schneider, P.F., Horrocks, I.: A comparison of two modelling paradigms in the semantic web. In: Proc. of the Fifteenth International World Wide Web Conference (WWW 2006). pp. 3–12. ACM (2006)
20. Rudolph, S., et al.: Efficient OWL Reasoning with Logic Programs – Evaluations. In: Marchiori, M., et al. (eds.) Proceedings of The First International Conference on Web Reasoning and Rule Systems 2007 (RR2007), Innsbruck, Austria, 7-8 June 2007. Lecture Notes in Computer Science, vol. 4524, pp. 370–373. Springer (2007)
21. Sarker, M.K., Carral, D., Krisnadhi, A.A., Hitzler, P.: Modeling OWL with rules: The ROWL protege plugin. In: Kawamura, T., Paulheim, H. (eds.) Proceedings of the ISWC 2016 Posters & Demonstrations Track, Kobe, Japan, October 19, 2016. CEUR Workshop Proceedings, vol. 1690. CEUR-WS.org (2016)
22. Steigmiller, A., Glimm, B., Liebig, T.: Reasoning with nominal schemas through absorption. Journal of Automated Reasoning 53(4), 351–405 (2014)
23. W3C Workshop on Rule Languages for Interoperability, 27-28 April 2005, Washington, DC, USA. W3C (2005), https://www.w3.org/2004/12/rules-ws/accepted

OWLAx: A Protégé Plugin to Support Ontology Axiomatization through Diagramming

Md. Kamruzzaman Sarker[1], Adila A. Krisnadhi[1,2], and Pascal Hitzler[1]

[1] Wright State University, OH, USA
[2] Universitas Indonesia, Depok, Indonesia

Abstract. Once the conceptual overview, in terms of a somewhat informal class diagram, has been designed in the course of engineering an ontology, the process of adding many of the appropriate logical axioms is mostly a routine task. We provide a Protégé[3] plugin which supports this task, together with a visual user interface, based on established methods for ontology design pattern modeling.

1 Motivation

When modeling with domain experts, particularly when they do not possess intimate knowledge about ontology engineering, it is in our experience best to use a visual approach to first design a conceptual overview of ontology modules (or corresponding content ontology design patterns), in the form of class diagrams [4]. We have found it most effective to use non-electronic means for this, such as whiteboards and flipcharts, as they readily support a natural flow of discussion without assuming any prior knowledge of particular software tools.

The ontology engineers in the modeling team will of course keep track of the precise meaning of each part of the diagram, so that they can convert their insights into exact specifications, i.e., axioms for the ontology. This conversion can then, based on the class diagram and the discussions during the modeling sessions, in our experience mostly be done by the ontology engineers without a lot of required further interaction with the domain experts. For documentation (or publication) purposes, the class diagram will usually also be redrawn using appropriate software tools.

In our experience, based on the class diagram and the discussions with domain experts during its design, it is mostly a routine, albeit somewhat tedious, task to write down appropriate axioms for an ontology module in an ontology editing tool. Most axioms in fact arise out of a systematic exploration of the class diagram. In order to simplify this part of the work, we have cast this systematic exploration into a Protégé plugin, which we describe herein. Of course, some axioms – arguably the more interesting and more critical ones – will not come up as candidates during our systematic exploration, and so will have to be added manually. Nevertheless, our plugin is helpful in making the task of adding many routine axioms much quicker and less error prone.

[3] http://protege.stanford.edu/

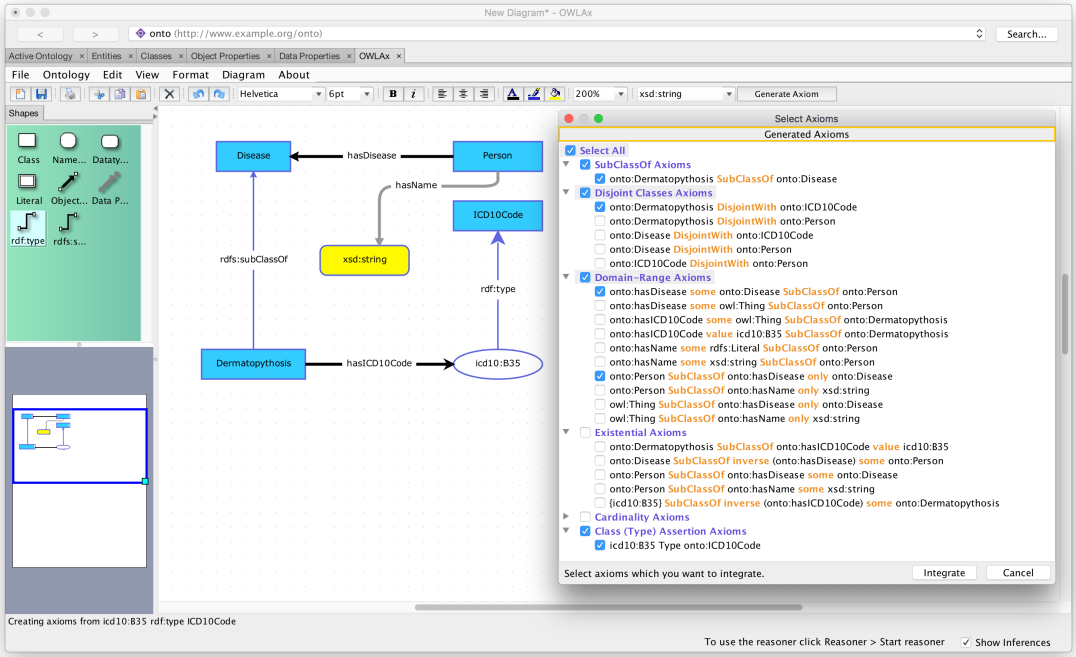

Fig. 1. OWLAx UI when "Generate Axioms" command is executed.

More information about the plugin is located at `http://daselab.org/content/ontology-axiomatization-support`.

2 OWLAx: Description and Main Functionalities

The plugin provides an interface for drawing a *class diagram*, and a command (accessible through an item in the menu or a button in the toolbar) to generate axioms from the given diagram, which are to be *added* into the currently active ontology. As seen in Fig. 1, the class diagram itself is composed of *nodes* and *edges*. A node in the class diagram represents either a *class*, *datatype*, *individual name*, or *literal*. Meanwhile, an edge represents either an *object property*, *data property*, the *typing* relation (i.e., rdf:type), or the *subclass* relation (i.e., rdfs:subClassOf). A pallette on the left side of the interface provides the user with those nodes and edges, which can be dragged and dropped onto the canvas.

In the following, $X \xrightarrow{P} Y$ means that there is a directed edge P from a node X to another node Y in the class diagram. Also, A and B denote class names, M a datatype, c a named individual, ℓ a literal, R an object property, and Q a data property. The plugin enforces the class diagram to contain at least one node, and if there is an edge, it only allows the following node-edge-node configurations: $A \xrightarrow{R} B$, $A \xrightarrow{R} c$, $A \xrightarrow{Q} M$, $A \xrightarrow{Q} \ell$, $c \xrightarrow{\text{rdf:type}} A$, and $A \xrightarrow{\text{rdfs:subClassOf}} B$. We do not aim to represent all possible relationships between components of the class diagram above because in our experience when modeling, the class diagram is usually considered informal, and the aforementioned node-edge-node

configurations are those typically used to describe a conceptual overview when we conduct modeling [4]. In fact, we do not intend to represent all possible OWL 2 constructs in the diagram unlike, e.g., Graphol [2], Graffoo [3], or Ontodia [5].

From the class diagram, a user can generate several types of candidate axioms based on the relationships depicted in the class diagram. They are only *candidates* since the class diagram is informal; each candidate axiom captures one way to read a relationship, and the actual intent should typically be inquired to the domain experts while conducting the modeling. Note that from one type relationship, more than one actual intents need to be formalized, i.e., the candidate axioms are not mutually exclusive. On the other hand, the list of candidate axioms is not exhaustive to keep it sufficiently simple: there are obviously axioms that will not be directly generated from the class diagram, especially if it is too complex. For such axioms, one has to simply directly input them in Protégé.

The plugin facilitates the creation of candidate axioms through a dialog box (accessible through "Generate Axiom" command from the menu or toolbar) that contains a checkbox of the candidate axioms presented in the Manchester syntax. After clicking "Integrate", the plugin will integrate the axioms with a check-mark to the ontology. We explain some of the candidate axioms below, though we use mainly description logic notation [1].

Every $c \xrightarrow{\text{rdf:type}} A$ leads to a *class assertion* $A(c)$, and $A \xrightarrow{\text{rdfs:subClassOf}} B$ to a *subclass axiom* $A \sqsubseteq B$. Next, for every $A \xrightarrow{R} B$, the plugin generates several types of candidate axioms. First, it generates *(unscoped) domain restriction* $\exists R.\top \sqsubseteq A$ — equivalent to R rdfs:domain A — and *scoped domain restriction* $\exists R.B \sqsubseteq A$. The former would be later integrated if the domain experts involved in modeling agrees that for every pair of instances x, y, if $x\ R\ y$ holds, then x belongs to A (regardless whether or not y belongs to B), while the latter is chosen if the domain experts agrees that if $x\ R\ y$ holds and y is known to belong to B, then x belongs to A. Such agreements will be solicited from domain experts involved in the modeling for every candidate axiom. Besides domain restrictions, the plugin also generates *scoped* and *unscoped range restrictions* $A \sqsubseteq \forall R.B$, $\top \sqsubseteq \forall R.B$ — equivalent to R rdfs:range B; several *existential axioms*, e.g., $A \sqsubseteq \exists R.B$, etc.; and several *functionality restrictions*, e.g., $A \sqsubseteq (\leq 1\ R.B)$, etc.

Similar types of candidate axioms are generated for every $A \xrightarrow{Q} M$, $A \xrightarrow{R} c$, and $A \xrightarrow{Q} \ell$ relationships. Finally, *class disjointness axioms* are generated as candidate axioms for every pair of different classes, unless there is a path of rdfs:subClassOf edges in the diagram connecting one class to the other.

3 Implementation Information and Other Features

The plugin is implemented on top of the OWL API provided by Protégé. The visual components are built using mxGraph.[4] The plugin allows users to save the diagram as XML-annotated PNG, which can then be loaded again. This plugin

[4] `http://jgraph.github.io/mxgraph/`

is *not* for visualizing an ontology for which there are a number of Protégé plugins already existing, but rather, it facilitates creating graphical class diagrams *inside* Protégé and provides a way to generate axioms from it. It eliminates the need to use separate tools for creating the class diagram and writing down the axioms. In addition, the user can customize various aspects of the class diagram, e.g., coloring, size of nodes and edges, text formatting, etc., through the provided menu or by right-clicking the corresponding graphical components.

One could use this plugin for modeling from scratch, or starting from an already created ontology. In the latter case, the plugin will not attempt create a class diagram from the ontology, and rather, start with an empty canvas. Nevertheless, when the user wishes to generate axioms through the plugin, existing axioms that are already in the ontology will be included as part of the list of candidate axioms, and the user can confirm whether to keep them. Finally, we hope to continue improving this plugin, particularly to support quick modeling of modular ontologies and ontology design patterns, and furthermore, evaluate the usability of our plugin via a comprehensive user study.

Acknowledgements. This work was supported by the National Science Foundation award 1017225 *III: Small: TROn – Tractable Reasoning with Ontologies.*

References

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2007)
2. Console, M., Lembo, D., Santarelli, V., Savo, D.F.: Graphical representation of OWL 2 ontologies through Graphol. In: Horridge, M., Rospocher, M., van Ossenbruggen, J. (eds.) Proceedings of the ISWC 2014 Posters & Demonstrations Track a track within the 13th International Semantic Web Conference, ISWC 2014, Riva del Garda, Italy, October 21, 2014. CEUR Workshop Proceedings, vol. 1272, pp. 73–76. CEUR-WS.org (2014)
3. Falco, R., Gangemi, A., Peroni, S., Shotton, D., Vitali, F.: Modelling OWL ontologies with Graffoo. In: Presutti, V., Blomqvist, E., Troncy, R., Sack, H., Papadakis, I., Tordai, A. (eds.) The Semantic Web: ESWC 2014 Satellite Events - ESWC 2014 Satellite Events, Anissaras, Crete, Greece, May 25-29, 2014, Revised Selected Papers. Lecture Notes in Computer Science, vol. 8798, pp. 320–325. Springer (2014)
4. Hitzler, P., Janowicz, K., Krisnadhi, A.A.: Ontology modeling with domain experts: The GeoVocamp experience. In: d'Amato, C., Lécué, F., Mutharaju, R., Narock, T., Wirth, F. (eds.) Proceedings of the 1st International Diversity++ Workshop co-located with the 14th International Semantic Web Conference (ISWC 2015), Bethlehem, Pennsylvania, USA, October 12, 2015. CEUR Workshop Proceedings, vol. 1501, pp. 31–36. CEUR-WS.org (2015)
5. Mouromtsev, D., Pavlov, D., Emelyanov, Y., Morozov, A., Razdyakonov, D., Galkin, M.: The simple web-based tool for visualization and sharing of semantic data and ontologies. In: Villata, S., Pan, J.Z., Dragoni, M. (eds.) Proceedings of the ISWC 2015 Posters & Demonstrations Track co-located with the 14th International Semantic Web Conference (ISWC-2015), Bethlehem, PA, USA, October 11, 2015. CEUR Workshop Proceedings, vol. 1486. CEUR-WS.org (2015)

Wikipedia Knowledge Graph for Explainable AI*

Md Kamruzzaman Sarker[1], Joshua Schwartz[1], Pascal Hitzler[1], Lu Zhou[1], Srikanth Nadella[3], Brandon Minnery[3], Ion Juvina[2], Michael L. Raymer[2,3], and William R. Aue[2]

[1] Kansas State University, Manhattan, KS 66506, USA
[2] Wright State University, Dayton, OH 45435, USA
[3] Kairos Research, Dayton, OH 45458, USA

Abstract. Explainable artificial intelligence (XAI) requires domain information to explain a system's decisions, for which structured forms of domain information like Knowledge Graphs (KGs) or ontologies are best suited. As such, readily available KGs are important to accelerate progress in XAI. To facilitate the advancement of XAI, we present the cycle-free Wikipedia Knowledge Graph (WKG) based on information from English Wikipedia. Each Wikipedia article title, its corresponding category, and the category hierarchy are transformed into different entities in the knowledge graph. Along with cycle-free version we also provide the original knowledge graph as it is. We evaluate whether the WKG is helpful to improve XAI compared with existing KGs, finding that WKG is better suited than the current state of the art. We also compare the cycle-free WKG with the Suggested Upper Merged Ontology (SUMO) and DBpedia schema KGs, finding minimal to no information loss.

Keywords: Knowledge Graph · Wikipedia · Ontology · XAI

1 Introduction

Artificial intelligence (AI)—including the subfields of machine learning and deep learning—has advanced considerably in recent years. In tandem with these performance improvements, understanding how AI systems make decisions has become increasingly difficult due to many nonlinear transformations of input data and the complex nature of the algorithms involved. The research area explainable AI (XAI) [8,7,16] investigates techniques to examine these decision processes.

A main desideratum of XAI is user understandability [6,5], while explanations should take into account the context of the problem and relevant domain knowledge [10]. Humans understand and reason mostly in terms of concepts and combinations thereof. A knowledge graph (KG) embodies such understanding in links between concepts; such a natural conceptual network creates a pathway to use knowledge graphs in XAI applications to improve overall understandability of complex AI algorithms. For an overview of some of the current discussion on

* This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Agreement No. HR00111890019

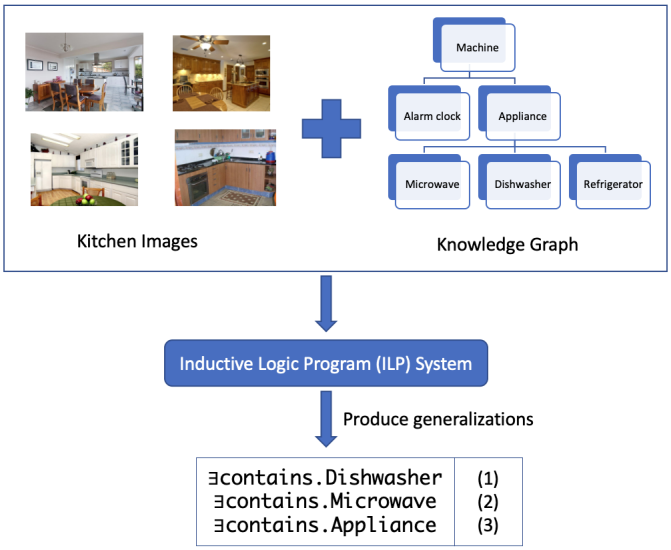

Fig. 1: Example of using knowledge graph to enhance explainability

utilizing knowledge graphs to enhance explanations, and possible limitations of existing approaches, see [12,9].

One of the primary elements of knowledge graphs to use in the XAI context is the notion of a concept hierarchy [4,18]. As illustrated in Figure 1, consider a system trying to explain the decisions of an image classifier. It may determine that an image should be given the label "Kitchen" because it contains a dishwasher, refrigerator, and microwave, and with the help of a KG concept hierarchy, it may produce the more general explanation that the image contains items in the "Appliance" class. These kinds of explanation generation systems are based on inductive logic programming (ILP) [14], and rich concept hierarchies play an important role in the generation of satisfactory explanations. To advance the state of XAI research, we provide a readily available knowledge graph with a rich concept hierarchy.

Wikipedia is perhaps the largest high-quality free source of information on the web. Wikipedia articles are classified into human-managed categories, which form a hierarchy (albeit with cycles). These concepts embody humans' natural ways of thinking and are easily understood, providing a greater benefit in an XAI context.

DBpedia [1], Suggested Upper Merged Ontology (SUMO) [15], Freebase [2], and Yago [19] are among the many high-quality, publicly available knowledge graphs providing domain information. These KGs use information from many sources, including Wikipedia. The hierarchical category information of Wikipedia, in which we are interested, is available in SUMO[1] but not in Freebase. It also exists in DBpedia and is accessible through SPARQL queries. Problematically, though, the Wikipedia parts of SUMO and the DBpedia KG contain cycles. For example, consider the following two axioms from DBpedia.

[1] `http://www.adampease.org/OP/`

I. *1949_establishments_in_Asia skos:broader 1949_establishments_in_India*
II. *1949_establishments_in_India skos:broader 1949_establishments_in_Asia*

These axioms form a cycle in the Wikipedia category hierarchy and hence also in DBpedia. The Wikipedia category hierarchy contains many such cycles, which complicates its use in XAI applications, as choosing parent concepts from the KG becomes nondeterministic.

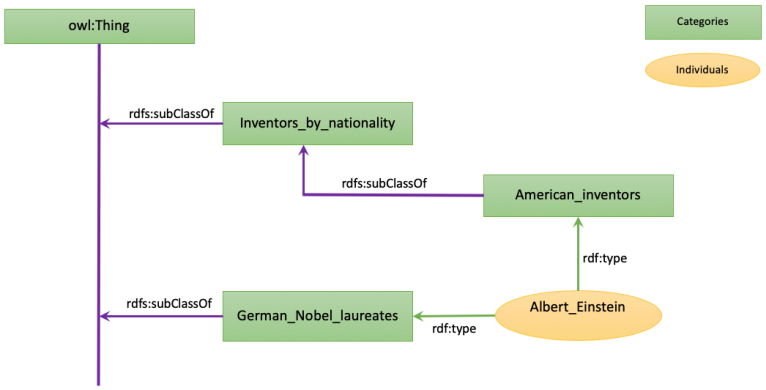

Fig. 2: Example architecture of the Wikipedia knowledge graph

To solve this problem, we provide a noncyclic version of the Wikipedia category hierarchy knowledge graph. We also empirically evaluate how the noncyclic knowledge graph performs in an XAI context and whether breaking cycles degrades its quality, finding that the Wikipedia knowledge graph performs better in both scenarios than other existing knowledge graphs.

The rest of the paper is organized as follows. First, we describe the high level architecture of the knowledge graph in section 2. Next, we describe the steps involved in building the knowledge graph. Then, in section 4, we evaluate the knowledge graph before concluding.

2 Knowledge Graph Architecture

We want to make the knowledge graph as simple as possible to enable use within XAI applications with minimal preprocessing. In the knowledge graph, we will have entities (named individuals in OWL 2), their types (classes in OWL 2), and the types' hierarchy. Many relations can be extracted from Wikipedia, but for simplicity we will use only two: *rdf:type* and *rdfs:subClassOf*. The relation *rdf:type* will be used to assign the individuals to their corresponding types, and the *rdfs:subClassOf* relation will be used to create the hierarchy. The title of a Wikipedia article (a.k.a. page) becomes an entity in our KG. Categories of a page become the types of the corresponding individual. A subcategory relationship becomes a *rdfs:subClassOf* relationship.

Figure 2 shows the architecture of our knowledge graph with an example. We can see that the article *Albert_Einstein* is mapped into the knowledge graph as an individual. This article belongs to many categories, including *German_Nobel_laureates* and *American_inventors*, which are converted into instances of *rdfs:Class*. The category *American_inventors* is a subcategory of *Inventors_by_nationality*, among others, resulting in the relation

American_inventors rdfs:subClassOf Inventors_by_nationality

in the KG.

3 Generating the Knowledge Graph

We now briefly describe a procedure for generating a knowledge graph like the one discussed above from the version of Wikipedia for a particular language; full details are in Appendix A. To construct the Wikipedia category hierarchy knowledge graph from scratch, we explored two alternative approaches: traversing and parsing the hierarchy page by page, and using a Wikipedia data dump.[2] To get all page and category information from Wikipedia through a traversal, we

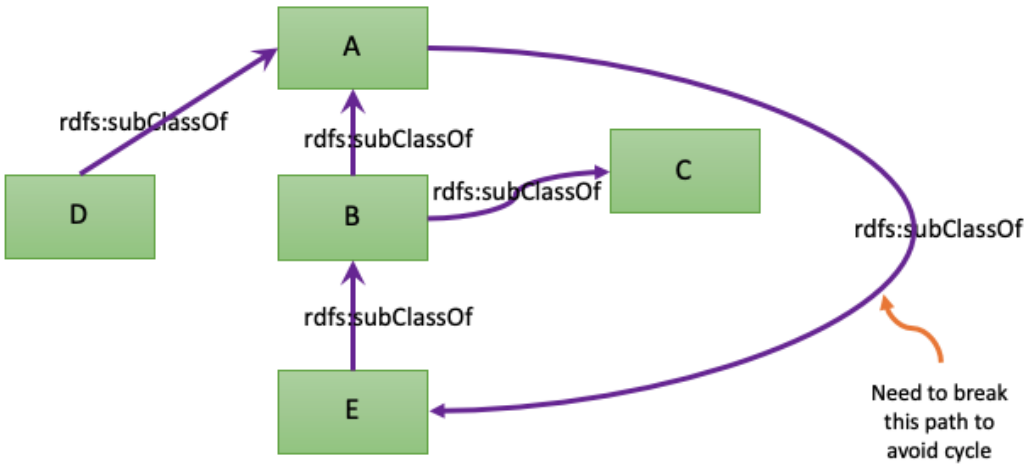

Fig. 3: Example of how cycles are broken

start at the top category[3] and exhaustively look through its subcategories and pages recursively, a time-consuming process complicated by the need to parse each page to find the proper links to visit the next categories or pages. To determine how long this process takes in practice, we used Python to implement the

[2] `http://dumps.wikimedia.org/enwiki/latest`
[3] `https://en.wikipedia.org/wiki/Category:Main_topic_classifications`

visiting and scraping program and found that it took roughly five days on a 2.2 GHz Intel Core i5 machine with 32 GB memory. As taking five days to produce a knowledge graph is not reasonable, we will focus on the Wikipedia data dump option.

A Wikipedia data dump contains all the information for each article: full text, editor list, category, etc. As stated in Section 2, our knowledge graph includes article title, category name, and the hierarchy of categories. These data are stored in the *page* and *categorylinks* tables. Using the Wikipedia data dump is straightforward: we just need to download the dump, import it into a database, and access it through SQL queries. After importing it, producing the full knowledge graph took only one hour, on the order of 1% of the time of the previous approach.

3.1 Concrete Implementation

Following the steps mentioned in Appendix A, we can create a concrete Wikipedia knowledge graph, ensuring compliance with W3C standards to make it maintainable, reusable, and non-proprietary. Many tools are available for this; among the most popular are the OWL API [11], the Apache Jena[4] library, and Owlready2,[5], all of which are compliant with W3C's standards.

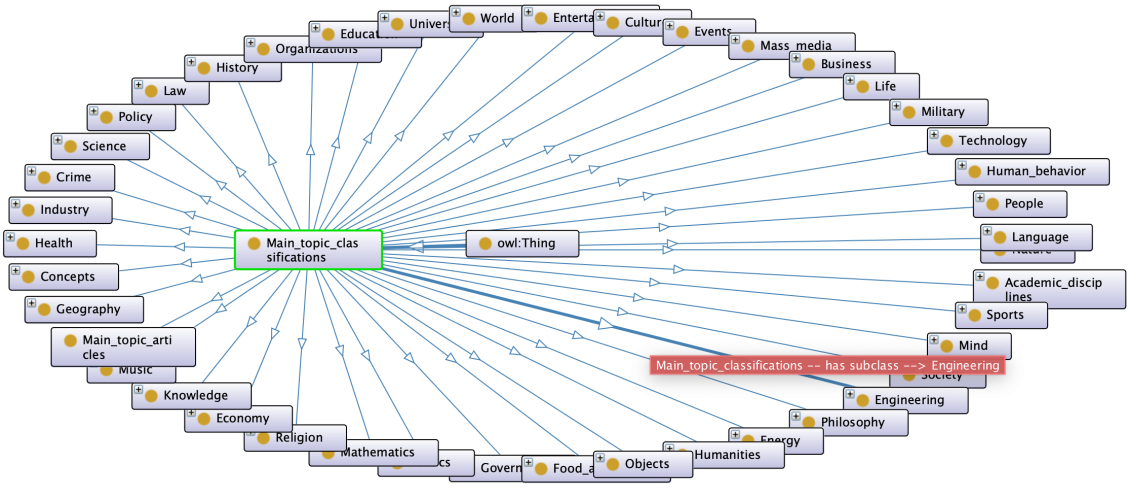

Fig. 4: Wikipedia Knowledge Graph

As discussed in Section 1, the raw Wikipedia hierarchy has cycles, resulting in cyclic relations in the knowledge graph. The Owlready2 library treats concepts as Python classes, representing subclass relationships through inheritance; since Python only supports inheritance without cycles, Owlready2 cannot handle these

[4] `https://jena.apache.org/`
[5] `https://pythonhosted.org/Owlready2/`

Table 1: Entity counts for Wikipedia, SUMO, and DBpedia knowledge graphs

Number of entities/facts	SUMO	DBpedia	Wikipedia cyclic	Wikipedia noncyclic
Concepts	4558	1183	1,901,708	1,860,342
Individuals	86,475	1	6,145,050	6,079,748
Object property	778	1144	2	2
Data property	0	1769	0	0
Axioms	175,208	7228	71,344,252	39,905,216
Class assertion axioms	167381	1	57,335,031	27,991,282
Subclass axioms	5330	769	5,962,463	3,973,845

cycles in relations. In contrast, the OWL API and Jena can support these cyclic relations; we use the former.[6]

While making the KG we face some practical issues, one being that many page titles on Wikipedia have non-ASCII characters, multiple spaces, and other peculiarities. For example, the article `https://en.wikipedia.org/wiki/Polish_People%27s_Party_%22Piast%22_(1913%E2%80%931931)` has title Polish_People%27s_Party_%22Piast%22_(1913%E2%80%931931). From an ontological perspective, this title as an entity name seems bad. We decide to replace spaces and characters in the set

`‘~!@#$%^&*()-+={}[]|\;’"<>,.?/`

with underscores (_) and then trim leading and trailing underscores from the resulting string. Another technical issue consists in the fact that if proper Unicode rendering is not selected, some article names will be saved as non–Unicode-compliant names. For example, as of 20 January 2020, the article title *Fabian's Lizard* contains the additional character *0x92* just before the *s*. This character only exists in windows encoding cp1252 and not in Unicode.[7]

3.2 Breaking Cycles

As stated above, the Wikipedia category hierarchy contains cycles, which we break by visiting the categories using breadth-first search (BFS). Starting from the root—*Main topic classifications*—we go level by level. An example of breaking a cycle is shown in Figure 3. In the example, if we start from A using BFS, we will get B and D as subclasses of A. On the next level, starting from B, we see that E is a subclass of B and store that information. On the next level, starting at E, we see that A is subclass of E; this results in a cycle, so we discard this information. Breaking cycles in this way results in some missing information in the final graph; however, it simplifies the knowledge graph considerably, allowing for efficient parent category determination, which is especially helpful in the XAI context.

[6] Our code is available at `https://github.com/md-k-sarker/Wiki-KG`.

[7] `https://stackoverflow.com/q/29419322/1054358`

Entity counts for both the cyclic and noncyclic versions of the WKG are shown in table 1. We see that breaking cycles results in losing 41,366 concepts (0.02% of the total 1,901,708 concepts) and 65,302 individuals (0.01% of the total 6,145,050 individuals). We further see that we lose a substantial number of class assertion axioms—29,341,749, or 0.5% of the total noncyclic axioms. Figure 4 shows a top-level view of the complete knowledge graph.[8]

4 Evaluation

The goal of our experimental evaluation was to test the hypothesis that the Wikipedia Knowledge graph produces XAI results comparable to or better than existing knowledge graphs. As to the best of our knowledge only SUMO has been used previously in a comparable context [18], to test this we compared the performance of our newly created WKG with that of the SUMO KG. We further hypothesized that breaking cycles in the Wikipedia knowledge graph results in minimal information loss and evaluated WKG relative to SUMO and the DBpedia schema.[9]

4.1 WKG's Effectiveness in XAI

To the best of our knowledge, there is no previously established quantitative measure of XAI quality, so we decided to use the accuracy metric of inductive logic programming (ILP)—the backbone of XAI [18]—to explain a supervised machine learning algorithm's decisions in terms of a KG. ILP provides many alternative solutions by using a KG. To measure a solution's performance, we used coverage score, described in equation (1), as the objective function. To measure the overall performance of a KG, we calculated the average of all scores of the produced solution for an experiment with equation (2).

$$Coverage(S) = \frac{P_S + N_{NS}}{P_S + P_{NS} + N_S + N_{NS}} \tag{1}$$

where

P_S = Number of positive individuals subsumed by the solution
P_{NS} = Number of positive individuals not subsumed by the solution
N_S = Number of negative individuals subsumed by the solution
N_{NS} = Number of negative individuals not subsumed by the solution

$$Average\ coverage = \sum_{i=1}^{n} Coverage(S_i) \tag{2}$$

[8] Available for download at `https://osf.io/3wbyr/`.
[9] `http://downloads.dbpedia.org/2014/dbpedia_2014.owl.bz2`

Following [18], we used the ADE20K dataset [20], which contains over 20,000 images classified by scene type and annotated with contained objects, to compare the results. We cast the ADE20k dataset, with annotations, into an OWL ontology and aligned it with SUMO, as in [18]; in the present context, we also aligned the ontology with WKG. We use all five experiments mentioned in [18], but expand the range of the experiments. While the previous paper used only 3–10 images for each experiment, we took all the training images (around 100) of the relevant categories from the ADE20K dataset. To get the explanation, we use ECII [17] instead of DL-Learner [3] to avoid the latter's considerable time complexity.

Table 2: Comparison of average coverage for WKG and SUMO in XAI context

Experiment name	#Images	#Positive images	Wikipedia		SUMO	
			#Solution	Coverage	#Solution	Coverage
Market vs. WorkRoom and WareHouse	96	37	286	.72	240	.72
Mountain vs. Market and WorkRoom	181	85	195	.61	190	.53
OutdoorWarehouse vs. IndoorWarehouse	55	3	128	.94	102	.89
Warehouse vs. Workroom	59	55	268	.56	84	.24
Workroom vs. Warehouse	59	4	128	.93	93	.84

We will now briefly discuss each of the scenarios in turn, before we summarize; Table 2 Figure 5 provide an overview of the results.

The first experiment involved finding a generalization of market images from the market vs. workroom and warehouse images. The ADE20K training dataset has, for those three categories, a total of 96 images, all of which we used. The objective was to cover as *many* as possible of the 37 images of market scenes and as *few* as possible of the images of workroom and warehouse scenes. When using the Wikipedia knowledge graph, the explanation framework (ECII) produced 286 alternative rules to generalize the market images, while using the SUMO knowledge graph results in 240 alternative rules. Average coverage score for both Wikipedia and SUMO was 0.72, i.e. in this case the simple Wikipedia category hierarchy knowledge graph performs as well as SUMO.

To produce a generalized rule of mountain scenes was the objective of the second experiment. All 181 images from the ADE20K training set were taken in this mountain vs. market and workroom experiment, where 85 images were of mountain scenes. The average coverage for Wikipedia was 0.61, representing slightly better performance than the 0.53 coverage we obtained for SUMO.

In the ADE20K training data, only three images are of outdoor warehouse scenes, while 52 are of indoor warehouse scenes. We wanted to compare the performances of the WKG and SUMO given such skewed sizes of sets of positive and negative individuals, so we took the three images of outdoor warehouses and 52 images of indoor warehouses, aiming to produce a generalized rule to describe the outdoor warehouse scenes. As there are fewer images to describe, both SUMO and Wikipedia performed well: ECII produced average coverages

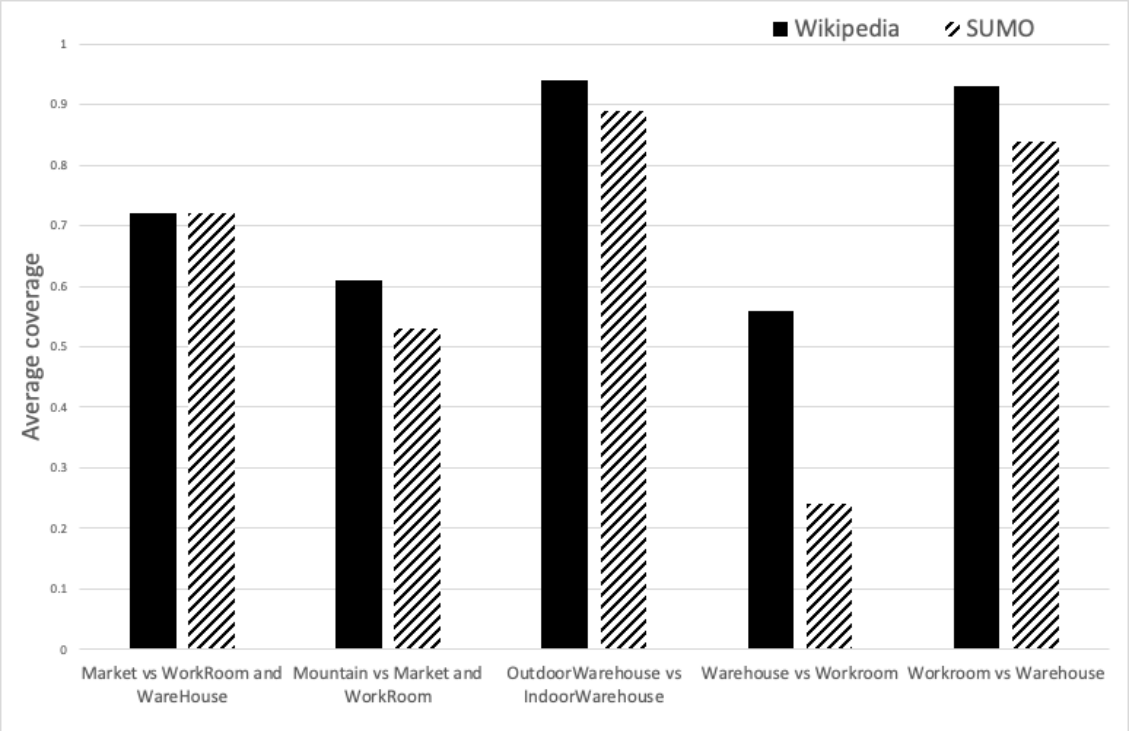

Fig. 5: Comparison of average coverage score between Wikipedia and SUMO knowledge graphs

of 0.89 from SUMO and 0.94 from Wikipedia, leading us to conclude that the Wikipedia KG again resulted in similar performance to the SUMO KG.

In the fourth and fifth experiments, we considered the case of warehouse vs. workroom. The ADE20K training set has 55 warehouse images and four workroom images. To produce a generalized rule to explain warehouse images SUMO returned average coverage of 0.24, while Wikipedia returned 0.56, a significantly larger difference than in previous cases. A large number of positive images compared to that of negative images (55 to 4) may explain the improved coverage score for the Wikipedia KG, as its depth and breadth of concepts exceeds those of SUMO. In the converse experiment (experiment 5)—describing the workroom scenes compared to the warehouse scenes—Wikipedia returned an average coverage score of 0.93 and SUMO returned 0.84. In this case, only four images were used to describe the workroom class, with 55 images on the negative side. Here Wikipedia and SUMO produced comparable average coverage scores.

The results are visualized in Figure 5, showing the simple Wikipedia category hierarchy's superior performance in all experiments compared to the SUMO ontology.

4.2 Noncyclic WKG Information Loss

For the second type of experiment, we evaluated the noncyclic WKG class hierarchy with respect to the DBpedia schema and SUMO knowledge graph to see

what proportion of subclass-superclass axioms remain in the WKG compared to the SUMO and DBpedia after breaking cycles. We expected that some subclass-superclass relations would be lost in the cycle-breaking process and hence not exist in our noncyclic WKG despite being present in other KGs. However, our experimental results show little to no information loss, with a substantial majority of the subclass-superclass relations in SUMO and DBpedia preserved in the noncyclic WKG.

The experiment involved first finding matching concepts in the WKG, SUMO, and DBpedia schema. To match the concepts we used a string similarity measurement algorithm (specifically Levenshtein [13] distance=0), finding 22 matching concepts, shown in Table 3. We extracted the asserted superclasses of those concepts from all three KGs. Details of the parents are shown on table 3. In the WKG, the number of asserted parents for some categories are quite large. For example, the category *Fish* has 114 asserted parent categories in the noncyclic WKG. As such, here we show only some of the parent concepts for each category.[10]

Table 3: Parents of all matching concepts in SUMO, DBpedia and noncyclic Wikipedia knowledge graph

Concept	Parent concepts			#Wikipedia parent concepts
	SUMO	DBpedia	Wikipedia	
Aircraft	Vehicle	MeanOfTransportation	Vehicles_by_type, Technology	5
Beer	AlcoholicBeverage	Beverage, Food	Food_and_drink	5
Birth	OrganismProcess	PersonalEvent, LifeCycleEvent, Event	Life	3
Boxing	Sport, ViolentContest	Sport, Activity	Sports	5
Brain	AnimalAnatomicalStructure, Organ	AnatomicalStructure	Human_anatomy, Physical_objects	15
Building	StationaryArtifact	ArchitecturalStructure, Place	Construction, Engineering	12
Cheese	PreparedFood, DairyProduct	Food	Foods	7
City	LandArea, GeopoliticalArea	Settlement, PopulatedPlace, Place	Human_habitats	42
Currency	FinancialInstrument	Thing	International_trade	60
Death	OrganismProcess	PersonalEvent, LifeCycleEvent, Event	Life	3
Fish	ColdBloodedVertebrate	Animal, Eukaryote, Species	Aquatic_organisms	114
Grape	Fruit	FloweringPlant, Plant, Eukaryote, Species	Edible_fruits	20
Language	LinguisticExpression	Thing	Culture	3
Medicine	BiologicallyActiveSubstance	Thing	Health_care, Health	4
Opera	DramaticPlay	MusicalWork, Work	Performing_arts, Entertainment	7
Painting	Coloring, Covering	Artwork, Work	Arts	7
Sales	Working	Activity	Marketing, Business	5
Sculpture	ArtWork	Artwork, Work	Visual_arts, Culture	7
Sound	BodyOfWater	Document, Work	Consciousness, Mind	5
Spacecraft	Vehicle	MeanOfTransportation	Spaceflight	13
Tax	CharginAFee	TopicalConcept	Governmet_finances	4
Wine	AlcoholicBeverage, PlantAgriculturalProduct	Beverage, Food	Fermented_drinks	32

Due to space constraints, we discuss only a subset of the 22 concepts that matched across the three KGs. We can divide the 22 concepts into twelve subsets by using the first letter of those concepts; among these, the letter B has the largest subset, with five elements: *Beer*, *Birth*, *Boxing*, *Brain*, and *Building*.

The concept *Beer* is available in SUMO, DBpedia and WKG. The only SUMO axiom related to the concept *Beer* is $Beer \sqsubseteq AlcoholicBeverage$, while in DBpedia we have $Beer \sqsubseteq Beverage$ and $Beer \sqsubseteq Food$; finally, in the noncyclic WKG we have the related axioms $Beer \sqsubseteq Food\_and\_drink$. We see that all three KGs have semantically similar parents of varying specificity.

[10] See `https://github.com/md-k-sarker/Wiki-KG` for full results.

Axioms related to the concept *Birth* in DBpedia are $Birth \sqsubseteq LifeCycleEvent$, $Birth \sqsubseteq PersonalEvent$ and $Birth \sqsubseteq Event$; in SUMO we have $Birth \sqsubseteq OrganismProcess$; and in the WKG, $Birth \sqsubseteq Life$. We can see that these parent concepts are again similar in meaning.

In SUMO, axioms related to the concept *Boxing* are $Boxing \sqsubseteq Sport$ and $Boxing \sqsubseteq ViolentContest$; DBpedia has $Boxing \sqsubseteq Sport$ and $Boxing \sqsubseteq Activity$; WKG has $Boxing \sqsubseteq Sports$, among others. The parent concepts of *Boxing* are *Sport*, *Sport*, and *Sports* in SUMO, DBpedia, and WKG, respectively; all of these clearly have the same meaning. Minor changes like the pluralization of the category name in Wikipedia are to be expected, as the SUMO and DBpedia schema are manually curated by domain experts and ontologists, while Wikipedia categories are editable by the general public.

Brain is another concept common to all three KGs. In SUMO we have $Brain \sqsubseteq AnimalAnatomicalStructure$ and $Brain \sqsubseteq Organ$, and in DBpedia, $Brain \sqsubseteq AnatomicalStructure$. Some related axioms in WKG are $Brain \sqsubseteq Human\_anatomy$ and $Brain \sqsubseteq Physical\_objects$. We see that ontologically, there exist some differences between *Human_anatomy* and *AnatomicalStructure*, but similar differences also exist between SUMO and DBpedia.

Finally, axioms related to the *Building* concept are: in SUMO, $Building \sqsubseteq StationaryArtifact$; in DBpedia, $Building \sqsubseteq ArchitecturalStructure$ and $Building \sqsubseteq Place$; and in WKG, ten axioms dealing with direct parents of the concept, including $Building \sqsubseteq Construction$ and $Building \sqsubseteq Society$. We again see that the parents are similar in semantics, though slight differences exist among the three ontologies.

Based on the above, we conclude that there is minimal information loss in the noncyclic Wikipedia KG with respect to DBpedia and SUMO. There exist some minor differences in an ontological sense with the WKG axioms, but such minor differences exist between SUMO and DBpedia as well.

5 Conclusion

The readily available Wikipedia category hierarchy and its corresponding named entities has great importance in artificial intelligence and its subfields. We make the Wikipedia Knowledge Graph (WKG), break its cycles, and make available both the original and cycle-free versions for public use. We evaluate the WKG in the context of XAI and compare it with the DBpedia and SUMO KGs, finding WKG to be highly effective compared to the other two. We also evalute the noncyclic WKG relative to SUMO and the DBpedia schema, finding minimal information loss. Here we evaluate the WKG in a specific XAI application; further work should focus on evaluating it in other such applications and in different domains of artificial intelligence.

References

1. Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S.: DBpedia—A crystallization point for the Web of Data. Journal of Web

Semantics **7**(3), 154–165 (2009)
2. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: In SIGMOD Conference. pp. 1247–1250 (2008)
3. Bühmann, L., Lehmann, J., Westphal, P.: DL-Learner – A framework for inductive learning on the semantic web. J. Web Sem. **39**, 15–24 (2016)
4. Confalonieri, R., del Prado, F.M., Agramunt, S., Malagarriga, D., Faggion, D., Weyde, T., Besold, T.R.: An ontology-based approach to explaining artificial neural networks (2019)
5. Doran, D., Schulz, S., Besold, T.R.: What does explainable AI really mean? A new conceptualization of perspectives. In: Besold, T.R., Kutz, O. (eds.) Proceedings of the First International Workshop on Comprehensibility and Explanation in AI and ML 2017, Bari, Italy, 2017. CEUR Workshop Proceedings, vol. 2071. CEUR-WS.org (2017)
6. Doshi-Velez, F., Kim, B.: Towards a rigorous science of interpretable machine learning. arXiv preprint arXiv:1702.08608 (2017)
7. Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., Pedreschi, D.: A survey of methods for explaining black box models. ACM computing surveys (CSUR) **51**(5), 93 (2018)
8. Gunning, D.: Explainable artificial intelligence (XAI). Defense Advanced Research Projects Agency (DARPA) (2017)
9. Hitzler, P., Bianchi, F., Ebrahimi, M., Sarker, M.K.: Neural-symbolic integration and the semantic web. Semantic Web (2020), accepted for publication
10. Holzinger, A., Biemann, C., Pattichis, C.S., Kell, D.B.: What do we need to build explainable AI systems for the medical domain? (2017)
11. Horridge, M., Bechhofer, S.: The owl api: A java api for owl ontologies. Semantic web **2**(1), 11–21 (2011)
12. Lecue, F.: On the role of knowledge graphs in explainable AI. Semantic Web journal (2019), http://www.semantic-web-journal.net/system/files/swj2198.pdf, retrieved on July 26, 2019
13. Levenshtein, V.I.: On the minimal redundancy of binary error-correcting codes. Inf. Control. **28**(4), 268–291 (1975). https://doi.org/10.1016/S0019-9958(75)90300-9, `https://doi.org/10.1016/S0019-9958(75)90300-9`
14. Muggleton, S., de Raedt, L.: Inductive logic programming: Theory and methods. The Journal of Logic Programming **19-20**, 629 – 679 (1994). https://doi.org/https://doi.org/10.1016/0743-1066(94)90035-3, `http://www.sciencedirect.com/science/article/pii/0743106694900353`, special Issue: Ten Years of Logic Programming
15. Niles, I., Pease, A.: Towards a Standard Upper Ontology. In: Proceedings of the International Conference on Formal Ontology in Information Systems – Volume 2001. pp. 2–9 (2001)
16. Samek, W., Wiegand, T., Müller, K.: Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. CoRR **abs/1708.08296** (2017), `http://arxiv.org/abs/1708.08296`
17. Sarker, M.K., Hitzler, P.: Efficient concept induction for description logics. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 3036–3043 (2019)
18. Sarker, M.K., Xie, N., Doran, D., Raymer, M., Hitzler, P.: Explaining trained neural networks with semantic web technologies: First steps. In: Besold, T.R., d'Avila Garcez, A.S., Noble, I. (eds.) Proceedings of the Twelfth International

Workshop on Neural-Symbolic Learning and Reasoning, NeSy 2017, London, UK, July 17-18, 2017. CEUR Workshop Proceedings, vol. 2003. CEUR-WS.org (2017)
19. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: A Core of Semantic Knowledge. In: Williamson, C.L., Zurko, M.E., Patel-Schneider, P.F., Shenoy, P.J. (eds.) Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007. ACM Press, New York, NY, USA (2007)
20. Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., Torralba, A.: Scene parsing through ADE20K dataset. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017. pp. 5122–5130. IEEE Computer Society (2017)

A Steps for Building the Wikipedia Knowledge Graph

As of 20 January 2020, the *page* table[11] (containing article information) has around 49 million entries, while the *categorylinks* table[12] (containing category information) has around 140 million entries.

As these files are large (the larger is 24GB), proper settings must be applied to the database before importing them to keep the import process from taking a prohibitively long time. In particular, we must disable foreign key checking and increase the buffer length.

There are different types of pages on Wikipedia: some pages are articles, some pages are categories, and some pages are for administrative use. Administrative pages are not of interest for the knowledge graph, so we omit them. Using the information from the table *categorylinks*, we can identify which pages are articles, which are categories, and so on. The column *page_namespace* holds the page type information; for categories, *page_namespace=14*, while for articles, *page_namespace=0*. This table also provides the category hierarchical information, in its columns *cl_from* and *cl_to*. The column *cl_from* is the article name or subcategory name, and column *cl_to* is the category or parent category name (depending on whether the page is an article or category). Each page has a unique ID and title. The table *page* gives us the needed information like ID of the page, title, etc.

The steps to create the knowledge graph are shown in Algorithm 1. By way of example, we demonstrate part of the execution of Algorithm 1 on the article Albert_Einstein.[13] Initially, we need to get the page_id for Albert_Einstein from the *page* table downloaded from the dump by executing the following query.

```
SELECT page_id, page_title, page_namespace FROM page
WHERE page_title = 'Albert_Einstein' and page_namespace = 0;
```

[11] Available for download at `http://dumps.wikimedia.org/enwiki/latest/enwiki-latest-page.sql.gz`, with and described in detail at `https://www.mediawiki.org/wiki/Manual:Page_table`.

[12] Available for download at `http://dumps.wikimedia.org/enwiki/latest/enwiki-latest-categorylinks.sql.gz`, and described in detail at `https://www.mediawiki.org/wiki/Manual:Categorylinks_table`.

[13] `https://en.wikipedia.org/wiki/Albert_Einstein`

Algorithm 1: Wikipedia knowledge graph construction algorithm

```
Function Iterate(A) :
    Find page_id pd, title t, page_namespace pn of page A;
    if pn == 0 then
        Declare title t as an entity e;
        Find categories (c ∈ C) of entity e;
        foreach c ∈ C do
            Declare category c as a rdf:type (class);
            Create facts: e rdf:type c;
            Find the pages (p ∈ P) which are entity of category c;
            foreach p ∈ P do
                Iterate(p) ;
            end
        end
    end
    else if pn == 14 then
        Declare title t a category (class) c;
        Find all sub-categories (sc ⊑ c) of category c;
        foreach sc ∈ C do
            Create relation: sc subClassOf c;
            Iterate(sc);
        end
    end
end
Iterate(Main_topic_classifications)        /* start the process from root */
```

The result of this query is in figure 6, and we can see that the page_id of article Albert_Einstein is 736.

After getting the page_id, we need to get the page's category, which we can get using the following query.

```
SELECT cl_from, cl_to FROM categorylinks WHERE cl_from = 736;
```

As of 20 January 2020, this page belongs to 148 different categories, a subset of which is shown in Figure 7.

Using the results of these queries, we can create axioms like *Albert_Einstein rdf:type German_inventors* and incorporate them into our knowledge graph. To continue creating the full hierarchy, we must continue with the parent categories of each the article's categories.

To get the parent category of a category, we must find the page_id of that category and use that to find its parent. For example, if we want to find the parent category of German_inventors, we need to determine the page_id of the German_inventors page as follows.

```
SELECT page_id, page_title, page_namespace FROM page
WHERE page_title = 'German_inventors' and page_namespace = 14;
```

page_id	page_title	page_namespace
736	Albert_Einstein	0

Fig. 6: Page_id of the article Albert_Einstein

cl_from	cl_to
736	German_Jews
736	German_Nobel_laureates
736	German_agnostics
736	German_emigrants_to_Switzerland
736	German_inventors
736	German_socialists

Fig. 7: Categories for the article Albert_Einstein

page_id	page_title	page_namespace
1033282	German_inventors	14

Fig. 8: Page_id of category *German_inventors*

cl_from	cl_to
1033282	Commons_category_link_is_on_Wikidata
1033282	German_businesspeople
1033282	German_inventions
1033282	Inventors_by_nationality
1033282	Science_and_technology_in_Germany

Fig. 9: Parent categories of the category German_inventors

This will return the result shown in Figure 8, where we see that the page_id of German_inventors is 1033282.

After getting this page_id, we can consult the *categorylinks* table for the parent category:

```
SELECT cl_from, cl_to FROM categorylinks WHERE cl_from = 1033282;
```

This will provide the parent results as shown in Figure 9, where we see that the parent categories of *German_inventors* are *Inventors_by_nationality* and *Science_and_technology_in_Germany*, among others.[14] This kind of relationship creates cycles in the category hierarchy, as discussed in Section 3.2.

We now see the complete process of creating an entity and adding axioms for its types and supertypes. The example above is but one fragment of the knowledge graph creation adventure; to complete the knowledge graph, we need to start from the root of the category hierarchy and continue with Algorithm 1 until all pages have been processed to yield article titles with their categories, along with the resulting category hierarchy.

[14] It may seem odd to have *Science_and_technology_in_Germany* and similar as parent categories of *German_inventors* in an ontology; this reflects the somewhat messy nature of Wikipedia.