Web Semantics: Science, Services and Agents on the World Wide Web 35 (2015) 124–139

Contents lists available at ScienceDirect

Web Semantics: Science, Services and Agents on the World Wide Web

journal homepage: www.elsevier.com/locate/websem

Automatic acquisition of class disjointness

Johanna Völker, Daniel Fleischhacker*, Heiner Stuckenschmidt

Data & Web Science Group, University of Mannheim, Germany

ARTICLE INFO

Article history:
Received 15 December 2014
Received in revised form
17 June 2015
Accepted 6 July 2015
Available online 23 July 2015

Keywords:
Ontology learning
Disjointness
Machine learning
Association rule mining
Semantic web
Linked data

ABSTRACT

Although it is widely acknowledged that adding class disjointness to ontologies enables a wide range of interesting applications, this type of axiom is rarely used on today's Semantic Web. This is due to the enormous skill and effort required to make the necessary modeling decisions. Automatically generating disjointness axioms could lower the barrier of entry and lead to a wider spread adoption. Different methods have been proposed for this automatic generation. These include supervised, top-down approaches which base their results on heterogeneous types of evidence and unsupervised, bottom-up approaches which rely solely on the instance data available for the ontology. However, current literature is missing a thorough comparison of these approaches. In this article, we provide this comparison by presenting two fundamentally different state-of-the-art approaches and evaluating their relative ability to enrich a well-known, multi-purpose ontology with class disjointness. To do so, we introduce a high-quality gold standard for class disjointness. We describe the creation of this standard in detail and provide a thorough analysis. Finally, we also present improvements to both approaches, based in part on discoveries made during our analysis and evaluation.

1. Introduction

One of the most fundamental ideas of Semantic Web technologies is the use of ontologies as background knowledge for information processing. The web ontology language has been created as a rich logical language for representing this knowledge. Ever since the invention of OWL, there has been a discussion about the degree of formalization of ontologies on the Web. Supporters of complex formal ontologies point to the benefits of rigorous formal underpinnings which enforce a certain degree of agreement that can be tested using logical reasoning. The ability to check the consistency of ontological definitions has been cited as one of the biggest benefits of formal ontologies [1]. This benefit, however, comes at a price. Being able to check consistency, usually requires rich logical specifications of classes, which have to be created before the model can be verified. In particular, the descriptions to be checked have to use some kind of negation, either in terms of the logical negation operator or hidden in other operators such a number restrictions. Building such rich axiomatizations is an error-prone and cumbersome task. Avoiding modeling errors, requires not only a deep understanding of the domain to be modeled, but also of the logic used to formalize it [2]. This problem, also referred to as the knowledge acquisition bottleneck is often cited as a main argument against the use of rich formal ontologies. There have been attempts to tackle the knowledge acquisition bottleneck by automatically generating ontologies from texts. So far however, methods for learning ontologies from texts, focus on the creation of so-called lightweight ontologies, which mainly consist of a concept hierarchy and relations between concepts [3]. Although there are first attempts at creating expressive ontologies from text [4] and data [5], it will not be possible to fully automate the creation of richly axiomatized ontologies for some time.

However, as we will show in the remainder of this article, we can support the acquisition of class (or property) disjointness, a limited form of negation. Conceptually, disjointness is a semantic relation between concepts, indicating that the concepts cannot have common instantiations, i.e., their extensions must always be disjoint. In combination with other semantic relations, in particular subsumption, disjointness provides a basis for checking the consistency of conceptual structures. In the following, we briefly discuss the role of disjointness in conceptual modeling, present a logical formalization of disjointness and discuss the use of disjointness as an operator in Semantic Web languages.

Disjointness has been an essential ingredient of ontological modeling from the very beginning on. In fact, *ontology*, in the original sense of the word tries to capture basic distinctions of reality and sort objects into (disjoint) categories. Aristotle's taxonomy of substances from the third century is a good example for the use of

* Corresponding author.
E-mail address: daniel@informatik.uni-mannheim.de (D. Fleischhacker).

http://dx.doi.org/10.1016/j.websem.2015.07.001
1570-8268/© 2015 Elsevier B.V. All rights reserved.

disjointness as a basic principle [6]. Aristotle uses inherent properties – so-called differentiae – to sort objects into disjoint classes. In particular, substances are distinguished into material and immaterial substances constituting the concepts 'body' and 'spirit'. Bodies are distinguished into animate and inanimate bodies leading to the disjoint concepts 'living' and 'mineral' and so forth. This means, that the whole taxonomy for categorizing objects is based on the notion of disjointness, as categories located at the same level of the taxonomy are disjoint by definition.

Ontologies, as they are used in the context of the Semantic Web today, although motivated by philosophical ontology, are heavily influenced by more pragmatic approaches to conceptual modeling. As a result, it is *not* the case that sibling classes in an ontology can automatically be assumed to be disjoint. Classes in an ontology do not always represent categories in the sense of ontological analysis, but often represent roles that certain objects play in a domain. A person, for example can be an author of a publication and at the same time the reviewer of a different one. Thus, the two classes 'author' and 'reviewer', which that are likely to appear on the same level of a class hierarchy, are by no means disjoint. The example of authors and reviewers also shows that the notion of disjointness is often context-dependent: We can say that nobody can be the author and the reviewer *of the same paper* or be married and unmarried *at the same time*.

Unfortunately, the definition of disjointness and the way it is perceived by humans is not always consistent. For example, when being asked about the disjointness of the two pairs of concepts 'author'/'reviewer' and 'married'/'unmarried', most people will say that the latter concepts are disjoint and the former are not, although in both cases we can argue that the disjointness is relative to a certain context. An in-depth investigation of this problem is out of the scope of this paper. Instead, our investigation uses the opinions of domain experts to determine whether classes should be considered disjoint or not.

However, such intricacies make manually modeling disjointness a non-trivial task which might also be the reason why it is not yet widely used in real-world datasets. This limited deployment is also visible in the Linked Open Data cloud, a network of interlinked datasets, many of which are accompanied by some kind of schema. According to the LODStats project,[1] only 6 out of 365 properly crawled datasets contain class disjointness statements, which means that only 1.7% of the datasets contain disjointness. Moreover, even these datasets only feature 49 disjointness statements in total. Glimm et al. [7] performed a survey on the Billion Triple Challenge 2011 dataset which discovers class disjointness as one of the top-20 OWL primitives employed in the dataset but nevertheless its usage in absolute numbers is low.

Since the Web of Data is driven by user-created content, it is not an option to force users to create more expressive ontologies, as this would pose a high entry barrier for contributors. Instead, the usage of automatic methods to enrich ontologies could help to close to gap between more expressivity without placing the burden of its full complexity on the casual contributor. In particular, employing statistical methods to generate more complex ontologies from the vast amount of data already available could help to ease the transition from the current Web of Data to the Semantic Web.

In the next section, we formally introduce disjointness as a modeling primitive (cf. Section 2). We then present two different methods for the automated acquisition of disjointness axioms from Semantic Web data:

In Section 3, we report on recent experiments in the field of statistical schema induction [8] and the discovery of class disjointness axioms from facts, i.e., class membership assertions, in a knowledge base. In Section 4, we describe a supervised machine learning approach, which does not presume the existence of facts, but only requires schema-level descriptions of the classes.

The remainder of this article is dedicated to answering the following research questions:

1. Which problems do human ontology engineers face when modeling class disjointness?
2. Which of these two approaches works better? Does the inductive, bottom-up discovery of disjointness axioms from given facts (cf. Section 3) outperform the supervised approach to learning disjointness, which relies on schema-level information (cf. Section 4)?
3. Does a supervised approach to learning disjointness work across datasets, so that we can train a classifier once and use it to enrich any other ontology with disjointness axioms?

In order to answer these questions and to enable a systematic evaluation of both types of approaches, we built a gold standard of manually created disjointness axioms (cf. Section 5).

Section 6 describes the experimental setup and the results of our comparative evaluation. We take a closer look at the generated axioms to identify the strengths and weaknesses of both approaches, and we suggest an extension to our previously developed framework [9], which helps to overcome a major problem that we identified during the evaluation (cf. Section 6.2.1).

In the light of our findings, we discuss some related work (cf. Section 7), before we conclude with a summary and an outlook to future work (cf. Section 8).

2. Class disjointness

Terminological knowledge usually groups objects of the world that have certain properties in common. A description of the shared properties is called a class definition. Classes can be arranged into a subclass–superclass hierarchy. Classes can be defined in two ways, by enumeration of its members or by stating that it is a refinement of a complex logical expressions. The specific logical operators to express such logical definitions can vary between ontology languages; the general definitions we give here abstract from these specific operators. Further relations can be specified in order to establish structures between classes. Terminological knowledge considers binary relations that can either be defined by restricting their domain and range or by declaring them to be a sub-relation of an existing one. In order to capture the actual information content of a knowledge base, we allow to specify single objects, also called instances. In our view on terminological knowledge, instances can be defined by stating their membership in a class. Further, we can define instances of binary relations by stating that two objects form such a pair.

We can define semantics and logical consequence of a terminological knowledge base using an interpretation mapping $.^{\mathfrak{I}}$ into an abstract domain Δ such that:

- $c^{\mathfrak{I}} \subseteq \Delta$ for all class definitions c in the way defined above
- $r^{\mathfrak{I}} \subseteq \Delta \times \Delta$ for all relation definitions r
- $o^{\mathfrak{I}} \in \Delta$ for all object definitions o.

This type of denotational semantics is inspired by description logics, however, are not specific about operators that can be used to build class definitions, which are of central interest of these logics. Based on the mapping $.^{\mathfrak{I}}$, we formally define the notion of disjointness as follows:

Definition 1 (*Disjointness*). Two classes C and D are said to be disjoint iff $C^{\mathfrak{I}} \cap D^{\mathfrak{I}} = \emptyset$.

[1] http://stats.lod2.eu.