Answer Set Programming on Expert Feedback to Populate and Extend Dynamic Ontologies

Mathias Niepert
Indiana University
Department of Computer Science
mniepert@indiana.edu

Cameron Buckner
Indiana University
Department of Philosophy
cbuckner@indiana.edu

Colin Allen
Indiana University
Department History & Philosophy of Science
and Program in Cognitive Science
colallen@indiana.edu

Abstract

The next generation of online reference works will require structured representations of their contents in order to support scholarly functions such as semantic search, automated generation of cross-references, tables of contents, and ontology-driven conceptual navigation. Many of these works can be expected to contain massive amounts of data and be updated dynamically, which limits the feasibility of "manually" coded ontologies to keep up with changes in content. However, relationships relevant to inferring an ontology can be recovered from statistical text processing, and these estimates can be verified with carefully-solicited expert feedback. In this paper, we explain a method by which we have used answer set programming on such expert feedback to dynamically populate and partially infer an ontology for a well-established, open-access reference work, the Stanford Encyclopedia of Philosophy.

Introduction

The scale, complexity, and dynamic nature of many digital reference works calls for sophisticated information management tools to capture and harness their metadata. Structured representations of their subject domains are required for scholarly functions such as semantic search, automated generation of cross-references and tables of contents, and ontology-driven conceptual navigation (Crampes & Ranwez 2000). Many of these works will contain massive amounts of data and be updated dynamically to reflect the growing size and dynamic nature of their subject domains. For scholarly projects with limited resources, human-intensive methods for coding, populating, and maintaining ontologies are not viable. "Social tagging" approaches leading to so-called "folksonomies" will be useful for some purposes, but may not meet standards of academic review. We suggest that dynamic reference works are better served by "dynamic ontologies" (D'Iorio 2003; Brewster, Ciravegna, & Wilks 2003; Niepert, Buckner, & Allen 2007) which are automatically or semi-automatically updated to track changes in primary content.

The development of dynamic ontologies presents the knowledge modeler with a number of challenges. First, dynamic ontologies must be inferred and populated in part from the reference corpora themselves, but ontological relationships cannot be directly derived from statistical methods – estimates generated with these approaches must always be regarded as uncertain and incomplete. Second, fully automatic methods for inferring taxonomic relationships between terms require the modeler to make assumptions about the structure of the domain or the reference work on which they operate.

We believe that many online reference works are well-positioned to address these problems by making use of their most valuable informational resource: the domain experts who serve as their editors and contributors. With care, expert feedback can be obtained to "approve" the recommendations of automated methods without presuming knowledge of ontology design or placing undue demands on the contributors' time. This feedback can give the modeler a window on the domain which is likely to be the most objective source of knowledge available.

This paper describes our application of answer set programming to carefully-solicited expert feedback to infer an ontology for a well-established open-access reference work, the Stanford Encyclopedia of Philosophy (SEP). Three features of the SEP make it an ideal environment for developing and testing digital tools to learn and manage ontologies. First, it is *substantial and complex*: over 900 entries (>9 million words) of sophisticated humanities content that is beyond the comprehension of any one individual. Second, the SEP is *dynamic*: new and revised entries come online each month. Finally, it is *expert-driven*: more than 1,100 professional philosophers serve as its editors and authors. The feedback provided by SEP authors about their areas of expertise provides an overlapping mosaic of expert knowledge. Answer set programming permits the construction of a populated global ontology derived from these fragments.

Related Work

The foundations of answer set programming (ASP) were laid out by Gelfond and Lifschitz (Gelfond & Lifschitz 1988; 1991). We follow their definitions and terminology throughout the paper. A considerable amount of theoretical work has been done to leverage ASP for the semantic web. It can be roughly divided into translations from description logics to ASP (Heymans, Nieuwenborgh, & Vermeir 2005), combining rules and ontologies with semantic integration (Analyti *et al.* 2005), and semantic separation (Eiter *et al.* 2004). A variety of other approaches to learning and populating taxonomies exists. For example, collocation-based statistics (Caraballo & Charniak 1999) make use of entropy measures over adjective-noun collocations, and formal concept analysis (Cimiano, Hotho, & Staab 2005) generates concept lattices for nouns and verbs. Heterogeneous sources of evidence (such as Hearst Patterns (Hearst 1992), hypernymy information from WordNet, and 'Head-Matching' Heuristics (Velardi *et al.* 2005)) have been represented as first-order features and used to train standard classifiers (Cimiano *et al.* 2004). Concept hierarchies have been built upon a formal definition of subsumption (Sanderson & Croft 1999) and an iterative, top-down approach using measures of generality and semantic similarity was investigated (Ryu & Choi 2006). Several authors have attempted to extract ontologies from general linguistic taxonomies like WordNet(Oltramari *et al.* 2002).

Preliminaries

Ontology

An ontology $\mathcal{O}$ is a six-tuple $\langle \mathcal{C}, \mathcal{H}_{\mathcal{C}}, \mathcal{H}_{\mathcal{R}}, \mathcal{L}, \mathcal{F}_{\mathcal{C}}, \mathcal{F}_{\mathcal{R}} \rangle$, where $\mathcal{C}$ is the set of concepts, $\mathcal{H}_{\mathcal{C}}$ a taxonomy induced on the concepts, $\mathcal{H}_{\mathcal{R}}$ the set of non-taxonomic relations, $\mathcal{L}$ the set of terms (lexicals) which refer to concepts and relations, and $\mathcal{F}_{\mathcal{C}}, \mathcal{F}_{\mathcal{R}}$ are relations that map the terms in $\mathcal{L}$ to the corresponding concepts and relations. If the ontology is *dynamic* all these structures are likely to change over time. For example, an entirely new concept and new non-taxonomic relations might be added to the ontology. Individuals of the ontology are instances of the concepts (also referred to as classes). Note that, in dynamic ontologies, individuals might become concepts at later stages and that, generally, the instance-class distinction is highly domain and application dependent. For instance whether to count *distributed representation* as an instance of an idea about *connectionism* or as a subclass with instances may depend on the amount of discussion the topic has received among philosophers. Whether a dynamic process of ontology revision will converge towards the kind of upper level unified ontology that is preferred by some authors (Smith 2003) remains to be seen, but for our immediate practical goals, it is necessary first to understand the feasibility of various methods for revising ontologies within the domain of philosophy.

Answer Set Programming

Syntax An answer set program Π is given by a signature and a collection of rules. A signature σ is a four-tuple consisting of disjoint sets $\sigma = \langle O, F, P, V \rangle$, where elements of O, F, P are called $object$, $function$, and $predicate$ symbols, respectively. Predicates name relations holding between the domain's objects. Elements of V are *variables* which range over objects in the domain. *Terms* are variables or object constants, and terms not containing variables are called $ground$. An $atom$ is an expression of the form $p(t_1, t_2, ..., t_n)$, where $p \in P$ and $t_1, ..., t_n$ are terms. A $literal$ is an atom or its negation. The rules of Π are of the form

$$l_0 \vee ... \vee l_i \leftarrow l_{i+1}, ..., l_n, \text{not } l_{n+1}, ..., \text{not } l_z,$$

where $l_0, ..., l_z$ are literals of Π. For any rule r, $\{l_0, ..., l_i\}$ is called the $head$ of r and will be referred to as $H(r)$. $\{l_{i+1}, ..., l_n\}$ is called the *positive body* of r and will be referred to as $B^+(r)$, and $\{l_{n+1}, ..., l_z\}$ is called the *negative body* of r and will be referred to as $B^-(r)$. A rule with no body (such as $l. \leftarrow$) is called a $fact$, and a rule with no head (such as $\leftarrow l.$) is called a *constraint*.

Semantics The Herbrand universe of a logic program Π, denoted HU_Π, is the set of all possible ground terms recursively constructed by using object and function symbols occurring in Π. The Herbrand Base of Π, denoted HB_Π, is the set of all possible ground atoms whose predicate symbols occur in Π and whose arguments are elements in HU_Π. A ground instance of a rule r in Π is a rule obtained from r by replacing every variable X in r by $\phi(X)$, where ϕ is a mapping from all variables occurring in r to terms in the Herbrand universe. The set of all ground instances of r are denoted by $ground(r)$ and $ground(\Pi)$ denotes $\bigcup_{r \in \Pi} ground(r)$. A logic program is positive if for all rules r in Π, $B^-(r) = \emptyset$. An interpretation I is a set of ground literals with $I \subseteq HB_\Pi$. An interpretation S is called closed under Π if for every $r \in ground(\Pi), H(r) \cap S \neq \emptyset$ whenever $B^+(r) \subseteq S$. An interpretation S is an answer set for a positive disjunctive datalog program Π if it is minimal among all interpretations that are closed under Π. The Gelfond-Lifschitz transform of a ground program Π w.r.t. a set $S \subseteq HB_\Pi$ is the positive ground program Π^S, obtained from Π by (i) deleting all rules $r \in \Pi$ for which $B^-(r) \cap S \neq \emptyset$ holds and (ii) deleting the negative body from the remaining rules. An answer set of a program Π is a set $S \subseteq HB_\Pi$ such that S is an answer set of $ground(\Pi)^S$.

The Indiana Philosophy Ontology

The InPhO has been compiled by our group at Indiana University from a wide variety of sources (Niepert, Buckner, & Allen 2007). Our ontology is stored in both OWL format (regular versions of which will be posted on the project web site[1]) and in a SQL database. The InPhO contains four subontologies: person (subclass of FOAF::person[2]), document (from AKT[3]), organization (from SUMO[4]), and idea, as well as a set of non-taxonomic relations. Only the idea subontology will concern us here. It contains a taxonomic decomposition of the space of philosophical ideas according to the disciplinary relatedness of their *contents* rather than according to their structural roles. For example, instead of dividing *idea about philosophy* into *concept*, *distinction*, *argument*, *counterexample*, and so on, the InPhO decomposes it into subareas of philosophy–e.g. *idea about metaphysics*, *idea about epistemology*, *idea about logic*, *idea about ethics*, *idea about philosophy of mind*). Each subarea is in turn decomposed into a series of issues considered fundamental to

[1]http://inpho.cogs.indiana.edu

[2]http://xmlns.com/foaf/spec/

[3]http://www.aktors.org/publications/ontology/

[4]http://www.ontologyportal.org/

work in that subarea; for example, *idea about philosophy of mind* is decomposed into *idea about consciousness*, *idea about intentionality*, *idea about mental content*, *idea about philosophy of artificial intelligence*, *idea about philosophy of psychology*, and *idea about metaphysics of mind*. The set of concepts $\mathcal{C}$ in the concept hierarchy $\mathcal{H}_\mathcal{C}$ in the ontology is restricted to the set of ideas in philosophy. In the remainder of the paper we will omit the prefix *idea about* when referring to concepts.

Answer Set Programming to Populate and Extend Dynamic Ontologies

In this section we describe our approach in which we combine statistical text processing, human expert feedback, and logic programming to extend and populate the ontology.

The Program

The program consists of three parts: (i) the signature; (ii) a declaration containing extensional facts; and (iii) the regular part consisting of datalog rules with negation. The rules of our program are listed in Figure 1. Note that since it is stratified it has a unique answer set and a polynomial time data complexity (Apt & Blair 1990).

Signature Here, the signature does not contain function symbols. Hence, only the set of objects O and set of predicate symbols P require elaboration. The set of objects O will contain as members terms (here, the word term is used in the context of IR, i.e., as a lexical entity) referring to philosophical ideas deemed to be candidates for classes, individuals or both in the ontology. These terms are part of a controlled vocabulary which is maintained through statistical methods and author feedback. Since our program belongs to the class DATALOG$^\neg$ (datalog with negation) we can partition the set of predicates P into (a) the set of extensional predicate symbols P_E representing editor feedback and the $class$ and $is\text{-}a$ relations; (b) the set of intensional predicate symbols P_I denoting the relations *links-to* and *instance-of*; and (c) the set of auxiliary predicate symbols P_A which are introduced to represent intermediate results which are not part of the final output. All predicates are binary except for the predicate *class*, which is unary. Their semantics will be explained in detail in the following sections.

$P_E = \{$*class*, *is-a*, *similarity*$_{\{0...4\}}$, *more-specific*, *more-general*, *incomparable*$\}$
$P_I = \{$*instance-of*, *links-to*$\}$, and
$P_A = \{$*desc*, *pins*, *nins*, *plink*, *nlink*, $s_{\{0...4\}}$, *ms*, *mg*, *ic*$\}$.

Given this partition of the predicates one can define the relational database schemas $\mathcal{D}_{in}$ and $\mathcal{D}_{out}$ such that $Rel(\mathcal{D}_{in}) = P_E$ and $Rel(\mathcal{D}_{out}) = P_I$. The program computes the symmetric closures of the similarity and incomparable/either predicates, and the transitive closures of the more general and more specific predicates (see rules (3)-(6) in Figure 1). However, to be a program in DATALOG$^\neg$ no predicate symbol in P_E is allowed to be the head of a rule, that is, the logic program does not alter $Ins(\mathcal{D}_{in})$. Therefore, the auxiliary predicate symbols $s_{\{0...4\}}$, *mg*, *ms*, and *ic* are introduced as counterparts to *similarity*$_{\{0...4\}}$, *more-general*, *more-specific*, and *incomparable*. In the program, the set of facts for these predicates is set to the set of facts (tuples) for the corresponding predicates (relations) in $Ins(\mathcal{D}_{in})$. For example, the rule $mg(X,Y) \leftarrow more\text{-}general(X,Y)$ lets the set of facts composed of the predicate *mg* initially be equal to the set of facts composed of the predicate *more-general*. Now, when the transitive closure of *mg* is computed, no new *more-general* facts are inferred. The naming of the predicate symbols in P_E and P_I follows their intuitive meaning.

$$mg(X,Y) \leftarrow ms(Y,X) \tag{1}$$
$$ms(X,Y) \leftarrow mg(Y,X) \tag{2}$$
$$ic(X,Y) \leftarrow ic(Y,X) \tag{3}$$
$$s_i(X,Y) \leftarrow s_i(Y,X), i \in \{0,...,4\} \tag{4}$$
$$mg(X,Y) \leftarrow mg(X,Z), mg(Z,Y) \tag{5}$$
$$ms(X,Y) \leftarrow ms(X,Z), ms(Z,Y) \tag{6}$$
$$ic(X,Y) \leftarrow ms(X,Y), mg(X,Y), s_i(X,Y), i \in \{3,4\} \tag{7}$$
$$desc(X,Y) \leftarrow isa(X,Y), class(X), class(Y) \tag{8}$$
$$desc(X,Z) \leftarrow desc(X,Y), desc(Y,Z), class(X), class(Y), class(Z) \tag{9}$$
$$pins(X,Y) \leftarrow s_i(X,Y), ms(X,Y), not\ class(X), class(Y), not\ ic(X,Y),\ i \in \{3,4\} \tag{10}$$
$$plink(X,Y) \leftarrow s_i(Y,X), ms(Y,X), not\ desc(Y,X), not\ ic(Y,X), class(X), class(Y),\ i \in \{3,4\} \tag{11}$$
$$plink(X,Y) \leftarrow s_4(X,Y), ic(X,Y), not\ desc(Y,X), class(X) \tag{12}$$
$$nins(X,Y) \leftarrow pins(X,Z), desc(Z,Y), not\ class(X), class(Y), class(Z) \tag{13}$$
$$nins(X,Y) \leftarrow pins(X,Z), plink(Y,Z), not\ class(X), class(Y), class(Z) \tag{14}$$
$$instance\text{-}of(X,Y) \leftarrow pins(X,Y), not\ nins(X,Y) \tag{15}$$
$$nlink(X,Y) \leftarrow instance\text{-}of(Y,Z), plink(X,Z), class(X), not\ class(Y), class(Z) \tag{16}$$
$$nlink(X,Y) \leftarrow instance\text{-}of(Y,Z), desc(Z,X), class(X), not\ class(Y), class(Z) \tag{17}$$
$$nlink(X,Y) \leftarrow instance\text{-}of(Y,X), class(X), not\ class(Y) \tag{18}$$
$$nlink(X,Y) \leftarrow plink(X,Z), desc(Y,Z), class(X), class(Y), class(Z) \tag{19}$$
$$nlink(X,Y) \leftarrow isa(X,Z), isa(Y,Z), class(X), class(Y), class(Z) \tag{20}$$
$$links\text{-}to(X,Y) \leftarrow plink(X,Y), not\ nlink(X,Y) \tag{21}$$

Figure 1: The rules of the stratified answer set program which ultimately construct the annotated ontology from the expert feedback facts. Some rules enforcing the sharp distinction between intensional, extensional, and auxiliary predicates are omitted.

Declaration The declaration consists of the extensional facts, i.e. those composed of extensional predicates and objects in O. The extensional predicates can be further divided into taxonomic predicates (*class*, *is-a*) and expert feedback predicates (*more-specific*, *more-general*, *incomparable*, *similarity*$_{\{0...4\}}$).

Regular Part The regular part of our program consists of a set of datalog rules with negation. The working of these rules is explained in Section "Populating the Ontology", and a full list is shown in Figure 1.

Semantics We largely apply the standard ASP semantics of Gelfond and Lifschitz; however, instead of considering each answer set to be a possible world, we consider it to be a possible populated ontology. Intuitively, each answer set is a possible set of beliefs one can consistently derive from the set of facts and rules of the program.

Obtaining Extensional Facts through Statistical Text Processing and Author Feedback

The concepts in the InPhO idea subontology are related over the taxonomic *is-a* relation. Each of these concepts (e.g., *rationalism*) is referred to by a term (a text string) in the controlled vocabulary. Therefore, the problem of determining hierarchical relationships between concepts can be reduced to that of finding hierarchical relationships between terms, that is, extracting hypernym and hyponym relations from text. There are two necessary conditions for a term t_1 to be a hypernym of term t_2: it has to be (a) semantically similar to t_2 and (b) more general than t_2 in the context of the subject area the terms are used in. Conversely, for a term t_1 to be a hyponym of term t_2 it has to be (a) semantically similar to t_2 and (b) more specific than t_2. In the "Related Work" section statistical methods are summarized which assign generality (specificity) scores to terms using text corpora. Additionally, a large number of measures for the semantic similarity between terms exist. In previous work (Niepert, Buckner, & Allen 2007) we combined such measures of similarity and generality to provide, for any given term, a ranking of possible hyponyms and hypernyms, respectively. Based on these rankings, article authors are asked, when finalizing their document for publication, to approve or falsify the estimates of semantic relatedness and relative generality of pairs of terms which our algorithms have deemed highly related to the topic of their article. The relatedness is scored on a five-point scale from highly related to unrelated, and the generality question has four options: same level of generality, $idea1$ is more general than $idea2$, $idea1$ is more specific than $idea2$, and the two are incomparable. The generality of two ideas is incomparable if they are entirely unrelated or if one idea can be both more *and* less general than the other, depending on the context. The feedback is stored as extensional facts in $Ins(\mathcal{D}_{in})$. For example, when an author provides the information that the idea *neural network* is more specific than *connectionism* and that they are highly related, the facts *more-specific(neural network, connectionism)* and s_4*(neural network, connectionism)* are added as two new tuples to the current instance of $\mathcal{D}_{in}$.

At any point in time, the InPhO concept hierarchy is determined by the tuples of the *class* and *is-a* relations in $Ins(\mathcal{D}_{in})$. To be dynamic, this taxonomy must be extended on a regular basis. Whenever a number of new feedback facts has accumulated, the logic program is run to populate the idea subontology. The instances $I_1, ..., I_n$ of a concept C are most likely to become new subconcepts of C. Because the relative generality and similarity of pairs in $\{I_1, ..., I_n\}$ is evaluated by the authors, we are able to algorithmically determine a set of sub-concept candidates. Those instances I which have few or no siblings which are more general (i.e., $|\{I_k : more\text{-}general(I_k, I), I_k \in \{I_1, ..., I_n\}\}| \approxeq 0$) are considered candidates. In order to ensure the high quality of the InPhO, the final decision is made by the subject editors of the SEP. Their feedback on the automatically generated suggestions for subconcept candidates creates new *is-a* and *class* facts and thereby extends the taxonomy.

Populating the Ontology

The population of the idea subontology concerns the process of finding instances of the taxonomic relations *instance-of* and *links-to*. Given the extensional facts, the unique answer set is computed and the intensional facts *instance-of* and *links-to* are used to populate the corresponding relations. For example, if the fact *instance-of(neural network, connectionism)* is contained in the answer set, the pair $\langle neural\ network, connectionism\rangle$ is set to be a tuple in the instance of the relation *instance-of*. The outcome of the program is a populated taxonomy with *semantic links* between classes and between classes and individuals. We decided to include semantic links (which are also used in well-known hand-crafted document hierarchies[5], and provide richer interconnections while preserving the lattice structure) because the taxonomy will primarily be used to provide faceted search results and as a tool to browse the encyclopedia. The logic programming paradigm provides an elegant way to specify constraints and infer intensional facts, which ultimately model and populate the desired concept hierarchy. The programs are small in size, easy to read and understand, and rules can be added, changed, and removed if one wishes to alter the structure and content of the taxonomy.

Inferring the *instance-of* Facts Two ideas A and B are denoted *strongly related*, if at least one author evaluated them as 3 or 4 on the 5-point scale ranging from 0 (unrelated) to 4 (highly related). There is *evidence for* (represented with the predicate symbol $pins$) the fact *instance-of(A, B)* if A and B are strongly related, A has been evaluated as more specific than B, A is not a class, B is a class and the two ideas are not incomparable (rule 10). There is *strong evidence against* ($nins$) the fact *instance-of(A, B)* if there is a class C which is a descendant (represented with $desc$, the transitive closure of *is-a*, see rules 8 and 9) of class B, and if there is evidence for the fact *instance-of(A, C)* (rule 13). Similarly, there is strong evidence against the fact *instance-of(A, B)* if there is a class C and if there is evidence for the fact *links-to(B, C)* (rule 14). Finally, the fact *instance-of(A, B)* is inferred, if there is evidence for it and no strong evidence against it (rule 15). In Figure 2 the $pins$ arcs denote that there is evidence for the facts

[5]http://dmoz.org—The Open Directory Project

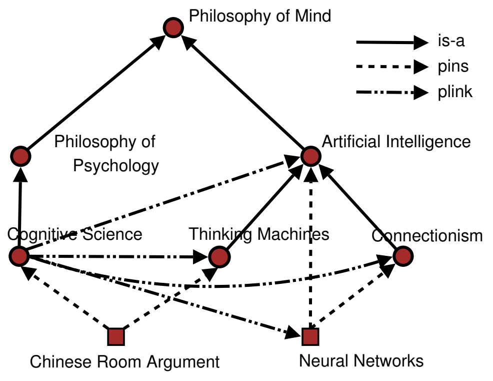

Figure 2: This actual small fragment of our idea subontology is used as running example to visualize some of the rules. Circles are classes and squares are individuals. Some of the inferred *pins* and *plink* facts are shown.

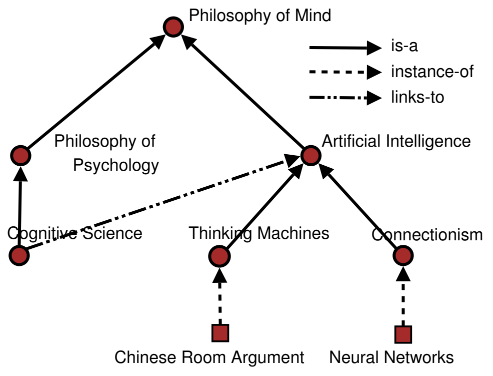

Figure 3: The finally inferred *instance-of* and *links-to* facts. One can nicely see how the use of the *links-to* relation helps to minimize the number of lattice arcs while retaining reachability of strongly related ideas.

instance-of(neural networks, connectionism) and *instance-of(neural networks, artificial intelligence)*. However, since *connectionism* is a descendant of *artificial intelligence*, there is strong evidence against *instance-of(neural networks, artificial intelligence)*. Note that the rules do not constrain the hierarchy to form a tree but rather a lattice in which no two classes where one is a descendant of the other will have the same instance. If an idea A is an instance of more than one class within a subdiscipline, the decision of which unique class the individual will be an instance of is made by subject editors as a post-processing step.

Inferring the *links-to* Facts The *links-to* relation models semantic links between classes or between classes and instances. Since it is difficult to force the concepts of a complex discipline like philosophy into a strictly hierarchical structure we introduce these links to achieve direct reachability of highly related ideas even if they are located in different subareas. At the same time, we want to avoid too many superfluous links. Thus, when there is *evidence against* a link between ideas A and B (see below), it merely means that the link would be redundant as there already exists a link between A and a superclass of B. There is *evidence for* a link ($plink$) from class A to class B if they are strongly related, B is more specific than A, B not a descendant of A, and A and B are not incomparable (rule 11). Additionally, there is *evidence for* a link from class A to class B (or individual B) if they are very strongly related, B not a descendant of A, and if they are both incomparable (rule 12). There is *strong evidence against* a link from class A to class B ($nlink$) if (a) there is evidence for a link to a class C of which class B is a descendant (rule 19), or (b) if classes A and B are siblings in the concept hierarchy (rule 20). There is *strong evidence against* a link from class A to individual B if (a) there is evidence for a link from class A to a class C of which B is an instance (rule 16), or (b) if A has a descending class C of which B is an instance (rule 17), or (c) if B is an instance of A (rule 18).

In Fig. 2 the $plink$ arcs show the existing evidence for links from *cognitive science*. There is evidence for a link from class *cognitive science* to class *artificial intelligence* and to class *thinking machines*. However, there is strong evidence against the link from *cognitive science* to *thinking machines* because *thinking machines* is a descendant of *artificial intelligence* (rule 17). By the same rule, there is strong evidence against the link from *cognitive science* to *connectionism*. There is evidence for the link from class *cognitive science* to individual *neural networks*. However, there is also strong evidence against it, since there is evidence for the link from *cognitive science* to class *connectionism* of which *neural Networks* is an instance (rule 16). Fig. 3 depicts the derived *instance-of* and *links-to* facts contained in the answer set.

Resolving Feedback Inconsistencies In our setting, inconsistencies among the generality, specificity, and similarity predicates can arise. An inconsistency is called *strong* if, for any two terms t_1, t_2, the feedback set contains the facts *mg*(t_1, t_2) [*ms*(t_1, t_2)] and *mg*(t_2, t_1) [*ms*(t_2, t_1)]. An inconsistency is called *weak* if, for any two terms t_1 and t_2, the feedback set contains the facts $s_i(t_1, t_2)$ and $s_j(t_1, t_2)$ with $i \neq j$. If there is a strong inconsistency for two terms t_1, t_2 we infer the fact $ic(t_1, t_2)$, that is, we set the two terms to be incomparable (rule 7). Since one expert judged t_1 to be more general than t_2 (in some context) and another expert judged t_2 to be more general than t_1, it is concluded that the two concepts must be incomparable, disallowing *instance-of* relations. However, a semantic link between t_1 and t_2 is still possible if there exists a very high relatedness score (rule 12). Weak inconsistencies for terms t_1, t_2 are resolved by only considering the highest similarity score. We justify this by assuming that experts are more likely to fail to notice connections than they are to invent them. For example, not everyone would recognize the relationship of *anaphora* to *propositional attitudes* but we want to capture the information provided by those who do.

Experiments

In preliminary experiments, several subjects provided about 2000 evaluations of the relative generality and similarity of pairs of terms in the areas of *Philosophy of Language*, *Philosophy of Science*, and *Philosophy of Mind* with the already existing interfaces (Niepert, Buckner, & Allen 2007). Overlapping evaluations were possible, that is, some term pairs were evaluated by more than one author. Due to its superior performance in our tests, we decided to use the reasoner DLV[6]. Given the approximately 2500 extensional (input) facts, the reasoning took about one second. The resulting populated taxonomy can be browsed online[7].

Conclusion and Future Work

We believe that we have shown that a combination of statistical and formal reasoning methods applied to feedback from domain experts who are naïve about ontology design can be used efficiently to populate and maintain a dynamic ontology in a highly abstract domain. Answer set programming appears to be particularly suitable to this approach because it is resilient in the face of dynamic and conflicting feedback. We intend to explore alternative methods for inferring whole sections of the ontology. One uses defeasible reasoning methods to obtain a space of possible populated ontologies, and ranks them according to some statistically defined criteria. Another uses Markov Logic Networks (Richardson & Domingos 2006) which combine first-order knowledge bases with probabilistic (Bayesian) networks. This method would provide a probabilistic ranking of the populated ontologies through the assignment of weights obtained directly from statistical generality and similarity scores. A significant difference between these two approaches is that in the latter the populated ontologies are ranked directly as part of its inductive reasoning method, while in the former the rankings are computed as an independent, post-processing step. We hope to determine which of these two approaches and the one described in detail in this paper best serves the needs of a dynamic reference work. Finally, we would like to extend the reasoner to also derive non-taxonomic relations among entities in our ontology.

Acknowledgments

This research has been funded by Indiana University under the grant “New Frontiers in the Arts and Humanities” and by Digital Humanities Start-up Grant HD-50203-07 from the National Endowment for the Humanities.

References

Analyti, A.; Antoniou, G.; Damsio, C. V.; and Wagner, G. 2005. Stable model theory for extended rdf ontologies. In *Proceedings of ISWC*, 21–36.

Apt, K. R., and Blair, H. A. 1990. Arithmetic classification of perfect models of stratified programs. *Fundam. Inf.* 13(1):1–17.

Brewster, C.; Ciravegna, F.; and Wilks, Y. 2003. Background and foreground knowledge in dynamic ontology construction. In *Proceedings of the Semantic Web Workshop, Toronto, August 2003*. SIGIR.

Caraballo, S., and Charniak, E. 1999. Determining the specificity of nouns from text. In *Proceedings of EMNLP and VLC*, 63–70.

Cimiano, P.; Pivk, A.; Schmidt-Thieme, L.; and Staab, S. 2004. Learning taxonomic relations from heterogeneous sources of evidence. In *Proceedings of the ECAI Ontology Learning and Population Workshop*.

Cimiano, P.; Hotho, A.; and Staab, S. 2005. Learning concept hierarchies from text corpora using formal concept analysis. *J. Artif. Intell. Res. (JAIR)* 24:305–339.

Crampes, M., and Ranwez, S. 2000. Ontology-supported and ontology-driven conceptual navigation on the world wide web. In *Proceedings of HYPERTEXT*, 191–199.

D'Iorio, P. 2003. Cognitive models of hypernietzsche: Dynamic ontology and hyper-learning. *Jahrbuch für Computerphilologie* 5:179–184.

Eiter, T.; Lukasiewicz, T.; Schindlauer, R.; and Tompits, H. 2004. Combining answer set programming with description logics for the semantic web. In *Proceedings of KR, pages 141-151*.

Gelfond, M., and Lifschitz, V. 1988. The stable model semantics for logic programming. In *Proceedings of ICLP*, 1070–1080.

Gelfond, M., and Lifschitz, V. 1991. Classical negation in logic programs and disjunctive databases. *New Generation Computing* 9(3/4):365–386.

Hearst, M. A. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of COLING*, 539–545.

Heymans, S.; Nieuwenborgh, D. V.; and Vermeir, D. 2005. Nonmonotonic ontological and rulebased reasoning with extended conceptual logic programs. In *Proceedings of ESWC*, 392–407.

Niepert, M.; Buckner, C.; and Allen, C. 2007. A dynamic ontology for a dynamic reference work. In *Proceedings of JCDL*, 288–297.

Oltramari, A.; Gangemi, A.; Guarino, N.; and Masolo, C. 2002. Restructuring wordnet's top-level: The ontoclean approach. In *Proceedings of LREC*, 17–26.

Richardson, M., and Domingos, P. 2006. Markov logic networks. *Mach. Learn.* 62(1-2):107–136.

Ryu, P.-M., and Choi, K.-S. 2006. Taxonomy learning using term specificity and similarity. In *Proceedings 2nd Workshop on Ontology Learning and Population*, 41–48.

Sanderson, M., and Croft, B. 1999. Deriving concept hierarchies from text. In *Proceedings of SIGIR*, 206–213.

Smith, B. 2003. Ontology. In Floridi, L., ed., *Blackwell Guide to the Philosophy of Computing and Information*, 155–166. Oxford: Blackwell.

Velardi, P.; Navigli, R.; Cuchiarelli, A.; and Neri, F. 2005. Evaluation of ontolearn, a methodology for automatic population of domain ontologies. In *Ontology Learning from Text: Methods, Applications and Evaluation. IOS Press*.

[6]http://www.dbai.tuwien.ac.at/proj/dlv/

[7]http://inpho.cogs.indiana.edu/taxonomy/