Learning Domain Ontologies from Document Warehouses and Dedicated Web Sites

Roberto Navigli*
Università di Roma "La Sapienza"

Paola Velardi
Università di Roma "La Sapienza"

We present a method and a tool, OntoLearn, aimed at the extraction of domain ontologies from Web sites, and more generally from documents shared among the members of virtual organizations. OntoLearn first extracts a domain terminology from available documents. Then, complex domain terms are semantically interpreted and arranged in a hierarchical fashion. Finally, a general-purpose ontology, WordNet, is trimmed and enriched with the detected domain concepts. The major novel aspect of this approach is semantic interpretation, that is, the association of a complex concept *with a complex* term. *This involves finding the appropriate WordNet concept for each word of a terminological string and the appropriate conceptual relations that hold among the concept components. Semantic interpretation is based on a new word sense disambiguation algorithm, called structural semantic interconnections.*

1. Introduction

The importance of domain ontologies is widely recognized, particularly in relation to the expected advent of the Semantic Web (Berners-Lee 1999). The goal of a domain ontology is to reduce (or eliminate) the conceptual and terminological confusion among the members of a virtual community of users (for example, tourist operators, commercial enterprises, medical practitioners) who need to share electronic documents and information of various kinds. This is achieved by identifying and properly defining a set of relevant concepts that characterize a given application domain. An ontology is therefore a shared understanding of some domain of interest (Uschold and Gruninger 1996). The construction of a shared understanding, that is, a unifying conceptual framework, fosters

- communication and cooperation among people
- better enterprise organization
- interoperability among systems
- system engineering benefits (reusability, reliability, and specification)

Creating ontologies is, however, a difficult and time-consuming process that involves specialists from several fields. Philosophical ontologists and artificial intelligence logicians are usually involved in the task of defining the basic kinds and structures of concepts (objects, properties, relations, and axioms) that are applicable in every

* Dipartimento di Informatica, Università di Roma "La Sapienza," Via Salaria, 113 - 00198 Roma, Italia. E-mail: {navigli, velardi}@di.uniroma1.it.

© 2004 Association for Computational Linguistics

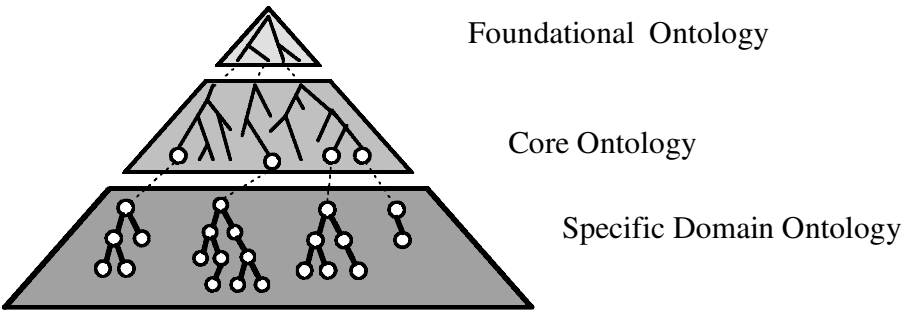

Figure 1
The three levels of generality of a domain ontology.

possible domain. The issue of identifying these very few "basic" principles, now often referred to as **foundational ontologies** (FOs) (or top, or upper ontologies; see Figure 1) (Gangemi et al. 2002), meets the practical need of a model that has as much generality as possible, to ensure reusability across different domains (Smith and Welty 2001).

Domain modelers and knowledge engineers are involved in the task of identifying the key domain conceptualizations and describing them according to the organizational backbones established by the foundational ontology. The result of this effort is referred to as the **core ontology** (CO), which usually includes a few hundred application domain concepts. While many ontology projects eventually succeed in the task of defining a core ontology,[1] populating the third level, which we call the **specific domain ontology** (SDO), is the actual barrier that very few projects have been able to overcome (e.g., WordNet [Fellbaum 1995], Cyc [Lenat 1993], and EDR [Yokoi 1993]), but they pay a price for this inability in terms of inconsistencies and limitations.[2]

It turns out that, although domain ontologies are recognized as crucial resources for the Semantic Web, in practice they are not available and when available, they are rarely used outside specific research environments.

So which features are most needed to build usable ontologies?

- Coverage: The domain concepts must be there; the SDO must be sufficiently (for the application purposes) populated. Tools are needed to extensively support the task of identifying the relevant concepts and the relations among them.
- Consensus: Decision making is a difficult activity for one person, and it gets even harder when a group of people must reach consensus on a given issue and, in addition, the group is geographically dispersed. When a group of enterprises decide to cooperate in a given domain, they have first to agree on many basic issues; that is, they must reach a consensus of the business domain. Such a common view must be reflected by the domain ontology.
- Accessibility: The ontology must be easily accessible: tools are needed to easily integrate the ontology within an application that may clearly show

1 Several ontologies are already available on the Internet, including a few hundred more or less extensively defined concepts.
2 For example, it has been claimed by several researchers (e.g., Oltramari et al., 2002) that in WordNet there is no clear separation between concept-synsets, instance-synsets, relation-synsets, and meta-property-synsets.

its decisive contribution, e.g., improving the ability to share and exchange information through the web.

In cooperation with another research institution,[3] we defined a general architecture and a battery of systems to foster the creation of such "usable" ontologies. Consensus is achieved in both an implicit and an explicit way: implicit, since candidate concepts are selected from among the terms that are frequently and consistently employed in the documents produced by the virtual community of users; explicit, through the use of Web-based groupware aimed at consensual construction and maintenance of an ontology. Within this framework, the proposed tools are **OntoLearn**, for the automatic extraction of domain concepts from thematic Web sites; **ConSys**, for the validation of the extracted concepts; and **SymOntoX**, the ontology management system. This ontology-learning architecture has been implemented and is being tested in the context of several European projects,[4] aimed at improving interoperability for networked enterprises.

In Section 2, we provide an overview of the complete ontology-engineering architecture. In the remaining sections, we describe in more detail OntoLearn, a system that uses text mining techniques and existing linguistic resources, such as WordNet and SemCor, to learn, from available document warehouses and dedicated Web sites, domain concepts and taxonomic relations among them. OntoLearn automatically builds a specific domain ontology that can be used to create a specialized view of an existing general-purpose ontology, like WordNet, or to populate the lower levels of a core ontology, if available.

2. The Ontology Engineering Architecture

Figure 2 reports the proposed ontology-engineering method, that is, the sequence of steps and the intermediate outputs that are produced in building a domain ontology. As shown in the figure, ontology engineering is an iterative process involving concept learning (OntoLearn), machine-supported concept validation (ConSys), and management (SymOntoX).

The engineering process starts with OntoLearn exploring available documents and related Web sites to learn domain concepts and detect taxonomic relations among them, producing as output a domain concept forest. Initially, we base concept learning on external, generic knowledge sources (we use WordNet and SemCor). In subsequent cycles, the domain ontology receives progressively more use as it becomes adequately populated.

Ontology validation is undertaken with ConSys (Missikoff and Wang 2001), a Web-based groupware package that performs consensus building by means of thorough validation by the representatives of the communities active in the application domain. Throughout the cycle, OntoLearn operates in connection with the ontology management system, SymOntoX (Formica and Missikoff 2003). Ontology engineers use this management system to define and update concepts and their mutual connections, thus allowing the construction of a semantic net. Further, SymOntoX's environment can attach the automatically learned domain concept trees to the appropriate nodes of the core ontology, thereby enriching concepts with additional information. SymOntoX

3 The LEKS-CNR laboratory in Rome.

4 The Fetish EC project, ITS-13015 (http://fetish.singladura.com/index.php) and the Harmonise EC project, IST-2000-29329 (http://dbs.cordis.lu), both in the tourism domain, and the INTEROP Network of Excellence on interoperability IST-2003-508011.

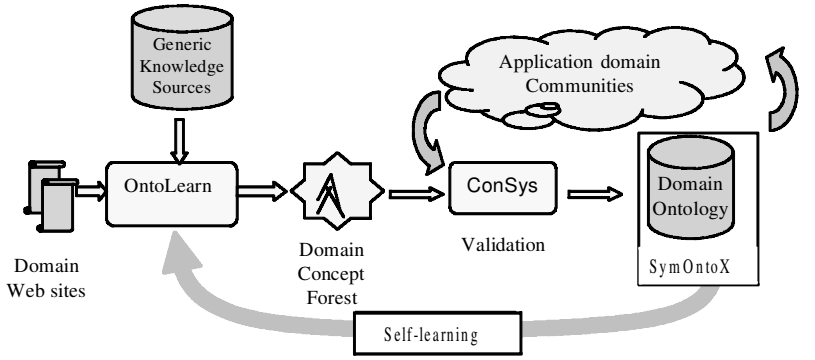

Figure 2
The ontology-engineering chain.

also performs consistency checks. The self-learning cycle in Figure 2 consists, then, of two steps: first, domain users and experts use ConSys to validate the automatically learned ontology and forward their suggestions to the knowledge engineers, who implement them as updates to SymOntoX. Then, the updated domain ontology is used by OntoLearn to learn new concepts from new documents.

The focus of this article is the description of the OntoLearn system. Details on other modules of the ontology-engineering architecture can be found in the referenced papers.

3. Architecture of the OntoLearn System

Figure 3 shows the architecture of the OntoLearn system. There are three main phases: First, a domain terminology is extracted from available texts in the application domain (specialized Web sites and warehouses, or documents exchanged among members of a virtual community), and filtered using natural language processing and statistical techniques. Second, terms are semantically interpreted (in a sense that we clarify in Section 3.2) and ordered according to taxonomic relations, generating a **domain concept forest** (DCF). Third, the DCF is used to update the existing ontology (WordNet or any available domain ontology).

In a "stand-alone" mode, OntoLearn automatically creates a specialized view of WordNet, pruning certain generic concepts and adding new domain concepts. When used within the engineering chain shown in Figure 2, ontology integration and updating is performed by the ontology engineers, who update an existing core ontology using SymOntoX.

In this article we describe the stand-alone procedure.

3.1 Phase 1: Terminology Extraction

Terminology is the set of words or word strings that convey a single (possibly complex) meaning within a given community. In a sense, terminology is the surface appearance, in texts, of the domain knowledge of a community. Because of their low ambiguity and high specificity, these words are also *particularly useful for conceptualizing a knowledge domain* or for supporting the creation of a domain ontology. Candidate terminological expressions are usually captured with more or less shallow techniques, ranging from stochastic methods (Church and Hanks 1989; Yamamoto and Church 2001) to more sophisticated syntactic approaches (Jacquemin 1997).

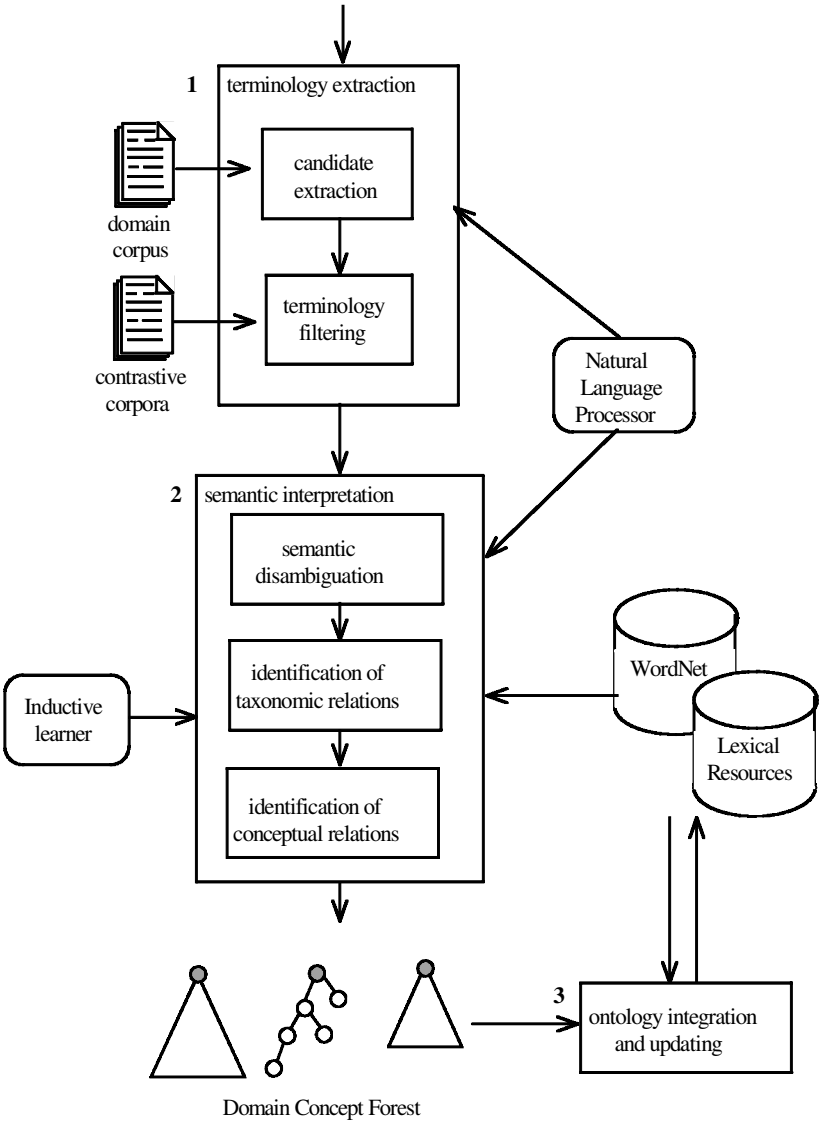

Figure 3
The architecture of OntoLearn.

Obviously, richer syntactic information positively influences the quality of the result to be input to the statistical filtering. In our experiments we used the linguistic processor ARIOSTO (Basili, Pazienza, and Velardi 1996) and the syntactic parser CHAOS (Basili, Pazienza, and Zanzotto 1998). We parsed the available documents in the application domain in order to extract a list T_c of syntactically *plausible* terminological noun phrases (NPs), for example, compounds (*credit card*), adjective-NPs (*local tourist information office*), and prepositional-NPs (*board of directors*). In English, the first two constructs are the most frequent.

OntoLearn uses a novel method for filtering "true" terminology, described in detail in (Velardi, Missikoff, and Basili 2001). The method is based on two measures, called **Domain Relevance** (DR) and **Domain Consensus** (DC), that we introduce hereafter.

High frequency in a corpus is a property observable for terminological as well as nonterminological expressions (e.g., *last week* or *real time*). We measure the specificity of a terminological candidate with respect to the target domain via comparative analysis across different domains. To this end a specific DR score has been defined. A quantitative definition of the DR can be given according to the amount of information captured within the target corpus with respect to a larger collection of corpora. More precisely, given a set of n domains $\{D_1, \ldots, D_n\}$ and related corpora, the domain relevance of a term t in class D_k is computed as

$$DR_{t,k} = \frac{P(t|D_k)}{\max_{1 \leq j \leq n} P(t|D_j)} \tag{1}$$

where the conditional probabilities ($P(t|D_k)$) are estimated as

$$E(P(t|D_k)) = \frac{f_{t,k}}{\sum_{t' \in D_k} f_{t',k}}$$

where $f_{t,k}$ is the frequency of term t in the domain D_k (i.e., in its related corpus).

Terms are concepts whose meaning is agreed upon by large user communities in a given domain. A more selective analysis should take into account not only the overall occurrence of a term in the target corpus but also its appearance in single documents. Domain terms (e.g., *travel agent*) are referred to frequently throughout the documents of a domain, while there are certain specific terms with a high frequency within single documents but completely absent in others (e.g., *petrol station*, *foreign income*). Distributed usage expresses a form of **consensus** tied to the consolidated semantics of a term (within the target domain) as well as to its centrality in communicating domain knowledge.

A second relevance indicator, DC, is then assigned to candidate terms. DC measures the distributed use of a term in a domain D_k. The distribution of a term t in documents $d \in D_k$ can be taken as a stochastic variable estimated throughout all $d \in D_k$. The entropy of this distribution expresses the degree of consensus of t in D_k. More precisely, the domain consensus is expressed as follows:

$$DC_{t,k} = \sum_{d \in D_k} \left(P_t(d) \log \frac{1}{P_t(d)} \right) \tag{2}$$

where

$$E(P_t(d_j)) = \frac{f_{t,j}}{\sum_{d_j \in D_k} f_{t,j}}$$

Nonterminological (or nondomain) candidate terms are filtered using a combination of measures (1) and (2).

For each candidate term the following term weight is computed:

$$TW_{t,k} = \alpha DR_{t,k} + \beta DC_{t,k}^{\text{norm}}$$

where $DC_{t,k}^{\text{norm}}$ is a normalized entropy and $\alpha, \beta \in (0, 1)$. We experimented with several thresholds for α and β, with consistent results in two domains (Velardi, Missikoff, and Basili 2001). Usually, a value close to 0.9 is to be chosen for α. The threshold for

Table 1
The first 10 terms from a tourism (left) and finance (right) domain.

tourism	finance
travel information	vice president
shopping street	net income
airline ticket	executive officer
booking form	composite trading
bus service	stock market
car rental	interest rate
airport transfer	million share
contact detail	holding company
continental breakfast	third-quarter net
tourist information office	chief executive

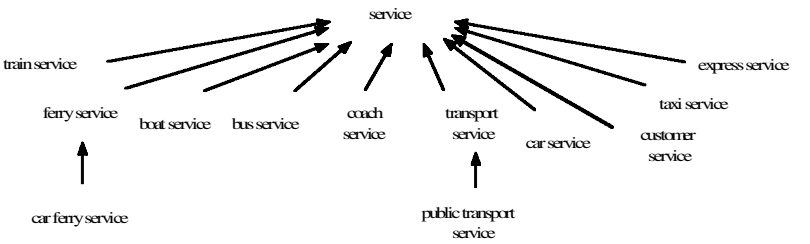

Figure 4
A lexicalized tree in a tourism domain.

β depends upon the number N of documents in the training set of D_k. When N is sufficiently large, "good" values are between 0.35 and 0.25. Table 1 shows some of the accepted terms in two domains, ordered by TW.

3.2 Phase 2: Semantic Interpretation

The set of terms accepted by the filtering method described in the previous section are first arranged in subtrees, according to simple string inclusion.[5] Figure 4 is an example of what we call a **lexicalized** tree $\mathcal{T}$. In absence of semantic interpretation, it is not possible to fully capture conceptual relationships between concepts (for example, the taxonomic relation between *bus service* and *public transport service* in Figure 4).

Semantic interpretation is the process of determining the right concept (sense) for each component of a **complex term** (this is known as **sense disambiguation**) and then identifying the semantic relations holding among the concept components, in order to build a **complex concept**. For example, given the complex term *bus service*, we would like to associate a complex concept with this term as in Figure 5, where *bus*#1 and *service*#1 are unique concept names taken from a preexisting concept inventory (e.g., WordNet, though other general-purpose ontologies could be used), and INSTR is a semantic relation indicating that there is a service, which is a type of work (*service*#1), operated through (instrument) a bus, which is a type of public transport (*bus*#1).

5 Inclusion is on the right side in the case of compound terms (the most common syntactic construct for terminology in English).

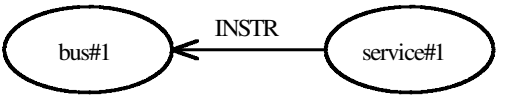

Figure 5
A complex term represented as a complex concept.

This kind of semantic interpretation is indeed possible if the meaning of a new complex concept can be interpreted compositionally from its components. Clearly, this is not always possible. Furthermore, some of the component concepts may be absent in the initial ontology. In this case, other strategies can be adopted, as sketched in Section 6.

To perform semantic disambiguation, we use available lexical resources, like WordNet and annotated corpora, and a novel word sense disambiguation (WSD) algorithm called **structural semantic interconnection**. A state-of-art inductive learner is used to learn rules for tagging concept pairs with the appropriate semantic relation.

In the following, we first describe the semantic disambiguation algorithm (Sections 3.2.1 to 3.2.4). We then describe the semantic relation extractor (Section 3.2.5).

3.2.1 The Structural Semantic Interconnection Algorithm. OntoLearn is a tool for extending and trimming a general-purpose ontology. In its current implementation, it uses a concept inventory taken from WordNet. WordNet associates one or more **synsets** (e.g., unique concept names) to over 120,000 words but includes very few domain terms: for example, *bus* and *service* are individually included, but not *bus service* as a unique term.

The primary strategy used by OntoLearn to attach a new concept under the appropriate hyperonym of an existing ontology is **compositional interpretation**. Let $t = w_n \cdot \ldots \cdot w_2 \cdot w_1$ be a valid multiword term belonging to a lexicalized tree $\mathcal{T}$. Let w_1 be the syntactic **head** of t (e.g., the rightmost word in a compound, or the leftmost in a prepositional NP). The process of compositional interpretation associates the appropriate WordNet synset S_k with each word w_k in t. The **sense** of t is hence compositionally defined as

$$S(t) = [S_k | S_k \in Synsets(w_k), w_k \in t]$$

where $Synsets(w_k)$ is the set of senses provided by WordNet for word w_k, for instance:

$$S(\text{“transport\_company”}) = [\{transportation\#4, shipping\#1, transport\#3\}, \{company\#1\}]$$

corresponding to sense 1 of *company* (an institution created to conduct business) and sense 3 of *transport* (the commercial enterprise of transporting goods and materials).

Compositional interpretation is a form of word sense disambiguation. In this section, we define a new approach to sense disambiguation called **structural semantic interconnections** (SSI).

The SSI algorithm is a kind of **structural pattern recognition**. Structural pattern recognition (Bunke and Sanfeliu 1990) has proven to be effective when the objects to be classified contain an inherent, identifiable organization, such as image data and time-series data. For these objects, a representation based on a “flat” vector of features causes a loss of information that has a negative impact on classification per-

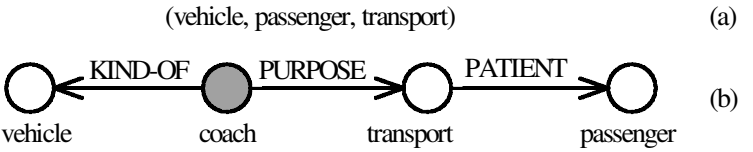

Figure 6
Two representations of the same concept: (a) as a feature vector and (b) as a semantic graph.

formances. The classification task in a structural pattern recognition system is implemented through the use of grammars that embody precise criteria to discriminate among different classes. The drawback of this approach is that grammars are by their very nature application and domain specific. However, automatic learning techniques may be adopted to learn from available examples.

Word senses clearly fall under the category of objects that are better described through a set of structured features. Compare for example the following two feature-vector (a) and graph-based (b) representations of the WordNet definition of *coach#5* (a vehicle carrying many passengers, used for public transport) in Figure 6. The graph representation shows the semantic interrelationships among the words in the definition, in contrast with the flat feature vector representation.

Provided that a graph representation for alternative word senses in a context is available, *disambiguation can be seen as the task of detecting certain "meaningful" interconnecting patterns among such graphs.* We use a context-free grammar to specify the type of patterns that are the best indicators of a semantic interrelationship and to select the appropriate sense configurations accordingly.

In what follows, we first describe the method to obtain a graph representation of word senses from WordNet and other available resources. Then, we illustrate the disambiguation algorithm.

Creating a graph representation for word senses. A graph representation of word senses is automatically built using a variety of knowledge source:

1. WordNet. In WordNet, in addition to synsets, the following information is provided:
   (a) a textual sense definition (gloss);
   (b) hyperonymy links (i.e., kind-of relations: for example, *bus*#1 is a kind of *public_transport*#1);
   (c) meronymy relations (i.e., part-of relations: for example, *bus*#1 has part *roof*#2 and *window*#2);
   (d) other syntactic-semantic relations, as detailed later, not systematically provided throughout the lexical knowledge base.
2. Domain labels[6] extracted by a semiautomatic methodology described in Magnini and Cavaglia (2000) for assigning domain information (e.g., tourism, zoology, sport) to WordNet synsets.
3. Annotated corpora providing examples of word sense usages in contexts:

6 Domain labels have been kindly made available by the IRST to our institution for research purposes.

(a) SemCor[7] is a corpus in which each word in a sentence is assigned a sense selected from the WordNet sense inventory for that word. Examples of a SemCor document are the following:

Color#1 *was delayed*#1 *until 1935, the widescreen*#1 *until the early*#1 *fifties*#1.

Movement#7 *itself was*#7 *the chief*#1 *and often*#1 *the only*#1 *attraction*#4 *of the primitive*#1 *movies*#1 *of the nineties*#1.

(b) *LDC/DSO*[8] is a corpus in which each document is a collection of sentences having a certain word in common. The corpus provides a sense tag for each occurrence of the word within the document. Examples from the document focused on the noun *house* are the following:

Ten years ago, he had come to the house#2 *to be interviewed.*

Halfway across the house#1, *he could have smelled her morning perfume.*

(c) In WordNet, besides glosses, examples are sometimes provided for certain synsets. From these examples, as for the LDC and SemCor corpora, co-occurrence information can be extracted. Some examples are the following:

Overnight accommodations#4 *are available.*

Is there intelligent#1 *life in the universe?*

An intelligent#1 *question.*

The use of other semantic knowledge repositories (e.g., FrameNet[9] and Verbnet[10]) is currently being explored, the main problem being the need of harmonizing these resources with the WordNet sense and relations inventory.

The information available in WordNet and in the other resources described in the previous section is used to automatically generate a labeled directed graph (**digraph**) representation of word senses. We call this a **semantic graph**.

Figure 7 shows an example of the semantic graphs generated for senses 1 (coach) and 2 (conductor) of *bus*; in the figure, nodes represent concepts (WordNet synsets) and edges are semantic relations. In each graph in the figure, we include only nodes with a maximum distance of three from the central node, as suggested by the dashed oval. This distance has been experimentally established.

The following semantic relations are used: **hyperonymy** (*car* is a kind of *vehicle*, denoted with $\stackrel{\text{kind-of}}{\longrightarrow}$), hyponymy (its inverse, $\stackrel{\text{has-kind}}{\longrightarrow}$), meronymy (*room* has-part *wall*, $\stackrel{\text{has-part}}{\longrightarrow}$), holonymy (its inverse, $\stackrel{\text{part-of}}{\longrightarrow}$), pertainymy (*dental* pertains-to *tooth*, $\stackrel{\text{pert}}{\longrightarrow}$), attribute (*dry* value-of *wetness*, $\stackrel{\text{att}}{\rightarrow}$), similarity (*beautiful* similar-to *pretty*, $\stackrel{\text{sim}}{\rightarrow}$), gloss ($\stackrel{\text{gloss}}{\longrightarrow}$),

7 http://www.cs.unt.edu/~rada/downloads.html#semcor
8 http://www.ldc.upenn.edu/
9 http://www.icsi.berkeley.edu/~framenet/
10 http://www.cis.upenn.edu/verbnet/

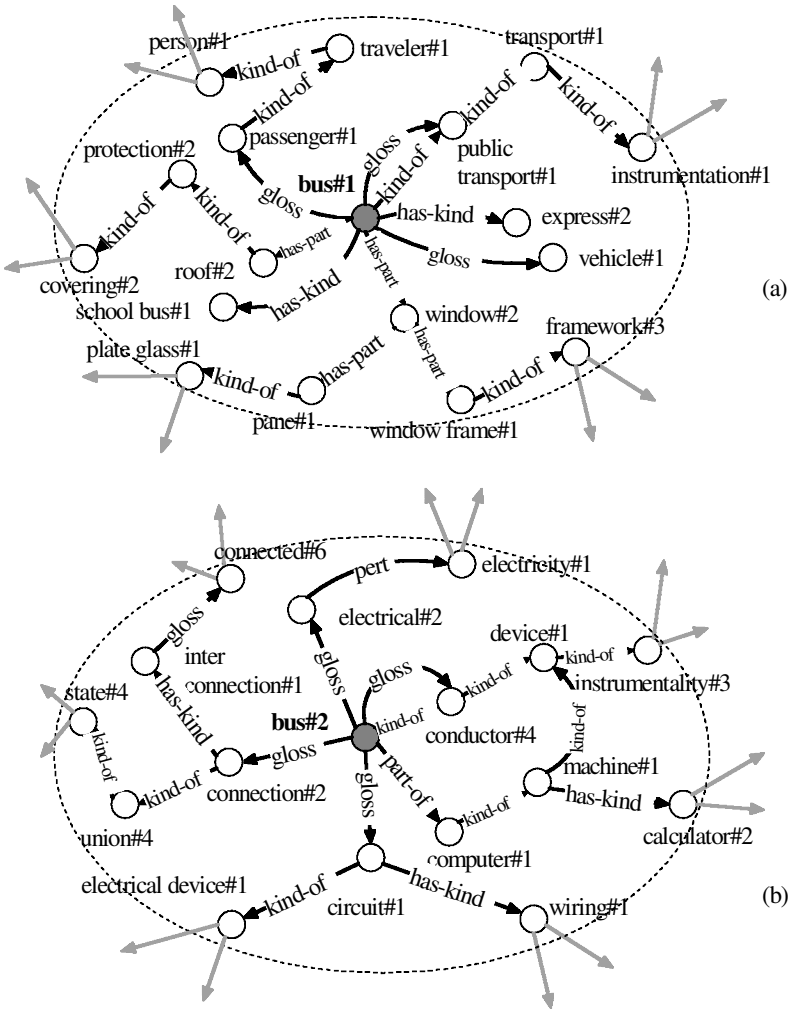

Figure 7
Graph representations for (a) sense 1 and (b) sense 2 of bus.

topic ($\xrightarrow{\text{topic}}$), and domain ($\xrightarrow{\text{dl}}$). All these relations are explicitly encoded in WordNet, except for the last three. Topic, gloss, and domain are extracted from annotated corpora, sense definitions, and domain labels, respectively. Topic expresses a co-occurrence relation between concepts in texts, extracted from annotated corpora and usage examples. Gloss relates a concept to another concept occurring in its natural language definition. Finally, domain relates two concepts sharing the same domain label. In parsing glosses, we use a stop list to eliminate the most frequent words.

The SSI algorithm. The SSI algorithm is a knowledge-based iterative approach to word sense disambiguation. The classification problem can be stated as follows:

- t is a term
- T (the **context** of t) is a list of co-occurring terms, including t.
- I is a structural representation of T (the **semantic context**).

- $S_1^t, S_2^t, \ldots, S_n^t$ are structural specifications of the possible senses for t (**semantic graphs**).
- G is a grammar describing structural relations (**semantic interconnections**) among the objects to be analyzed.
- Determine how well the structure of I matches that of each of $S_1^t, S_2^t, \ldots, S_n^t$, using G.
- Select the best matching.

Structural representations are graphs, as previously detailed. The SSI algorithm consists of an initialization step and an iterative step.

In a generic iteration of the algorithm, the input is a list of co-occurring terms $T = [t_1, \ldots, t_n]$ and a list of associated senses $I = [S^{t_1}, \ldots, S^{t_n}]$, that is, the semantic interpretation of T, where S^{t_i}[11] is either the chosen sense for t_i (i.e., the result of a previous disambiguation step) or the empty set (i.e., the term is not yet disambiguated). A set of *pending* terms is also maintained, $P = \{t_i | S^{t_i} = \emptyset\}$. I is referred to as the semantic context of T and is used, at each step, to disambiguate new terms in P.

The algorithm works in an iterative way, so that at each stage either at least one term is removed from P (i.e., at least one pending term is disambiguated) or the procedure stops because no more terms can be disambiguated. The output is the updated list I of senses associated with the input terms T.

Initially, the list I includes the senses of monosemous terms in T. If no monosemous terms are found, the algorithm uses an initialization policy described later.

During a generic iteration, the algorithm selects those terms t in P showing an interconnection between at least one sense S of t and one or more senses in I. The likelihood that a sense S will be the correct interpretation of t, given the semantic context I, is estimated by the function $f_I : Synsets \times T \rightarrow \Re$, where Synsets is the set of all the concepts in WordNet, and defined as follows:

$$f_I(S,t) = \begin{cases} \rho(\{\phi(S,S') | S' \in I\}) & \text{if } S \in Senses(t) \subset Synsets \\ 0 & \text{otherwise} \end{cases}$$

where $Senses(t)$ is the subset of synsets in WordNet associated with the term t, and $\phi(S,S') = \rho'(\{w(e_1,e_2,\ldots,e_n) | S \stackrel{e_1}{\rightarrow} S_1 \stackrel{e_2}{\rightarrow} \ldots \stackrel{e_{n-1}}{\rightarrow} S_{n-1} \stackrel{e_n}{\rightarrow} S'\})$, that is, a function ($\rho'$) of the weights ($w$) of each path connecting S with S', where S and S' are represented by semantic graphs. A semantic path between two senses S and S', $S \stackrel{e_1}{\rightarrow} S_1 \stackrel{e_2}{\rightarrow} \ldots \stackrel{e_{n-1}}{\rightarrow} S_{n-1} \stackrel{e_n}{\rightarrow} S'$, is represented by a sequence of edge labels $e_1, e_2, \ldots, e_n$. A proper choice for both ρ and ρ' may be the **sum** function (or the **average sum** function).

A context-free grammar $G = (E, N, S_G, P_G)$ encodes all the meaningful semantic patterns. The terminal symbols (E) are edge labels, while the nonterminal symbols (N) encode (sub)paths between concepts; S_G is the start symbol of G, and P_G the set of its productions.

We associate a weight with each production $A \rightarrow \alpha \in P_G$, where $A \in N$ and $\alpha \in (N \cup E)^*$, that is, α is a sequence of terminal and nonterminal symbols. If the sequence of edge labels $e_1, e_2, \ldots, e_n$ belongs to $L(G)$, the language generated by the grammar, and G is not ambiguous, then $w(e_1, e_2, \ldots, e_n)$ is given by the sum of the

11 Note that with S^{t_i} we refer interchangeably to the semantic graph associated with a sense or to the **sense label** (i.e., the synset).

weights of the productions applied in the derivation $S_G \Rightarrow^* e_1, e_2, \ldots, e_n$. (The grammar G is described in the next subsection.)

Finally, the algorithm selects $S^t = \underset{S \in Synsets}{\text{argmax}}\, f_I(S,t)$ as the most likely interpretation of t and updates the list I with the chosen concept. A threshold can be applied to $f_I(S,t)$ to improve the robustness of the system's choices.

At the end of a generic iteration, a number of terms are disambiguated, and each of them is removed from the set of pending terms P. The algorithm stops with output I when no sense S' can be found for the remaining terms in P such that $f_I(S',t') > 0$, that is, P cannot be further reduced. In each iteration, interconnections can be found only between the sense of a pending term t and the senses disambiguated during the previous iteration.

If no monosemous words are found, we explore two alternatives: either we provide manually the synset of the root term h (e.g., *service*#1 in Figure 4: work done by one person or group that benefits another), or we fork the execution of the algorithm into as many processes as the number of senses of the root term h. Let n be such a number. For each process i ($i = 1, \ldots, n$), the input is given by $I_i = [\emptyset, \emptyset, \ldots, S_i^h, \ldots, \emptyset]$, where S_i^h is the ith sense of h in $Senses(h)$. Each execution outputs a (partial or complete) semantic context I_i. Finally, the most likely context I_m is obtained by choosing

$$m = \underset{1 \le i \le n}{\arg\max} \sum_{S^t \in I_i} f_{I_i}(S^t, t)$$

Figure 8 provides pseudocode for the SSI algorithm.

3.2.2 The Grammar. The grammar G has the purpose of describing meaningful interconnecting patterns among semantic graphs representing concepts in the ontology. We define a **pattern** as a sequence of consecutive semantic relations $e_1 \cdot e_2 \cdot \ldots \cdot e_n$ where $e_i \in E$, the set of terminal symbols, that is, the vocabulary of conceptual relations. Two relations $e_i \cdot e_{i+1}$ are consecutive if the edges labeled with e_i and e_{i+1} are incoming and/or outgoing from the same concept node, for example, $\xrightarrow{e_i} S \xrightarrow{e_{i+1}}, \xleftarrow{e_i} S \xleftarrow{e_{i+1}}, \xleftarrow{e_i} S \xrightarrow{e_{i+1}}, \xrightarrow{e_i} S \xleftarrow{e_{i+1}}$. A **meaningful pattern** between two senses S and S' is a sequence $e_1 \cdot e_2 \cdot \ldots \cdot e_n$ that belongs to $L(G)$.

In its current version, the grammar G has been defined manually, inspecting the intersecting patterns automatically extracted from pairs of manually disambiguated word senses co-occurring in different domains. Some of the rules in G are inspired by previous work in the eXtended WordNet[12] project. The terminal symbols e_i are the conceptual relations extracted from WordNet and other on-line lexical-semantic resources, as described in Section 3.2.1.

G is defined as a quadruple (E, N, S_G, P_G), where $E = \{ e_{\text{kind-of}}, e_{\text{has-kind}}, e_{\text{part-of}}, e_{\text{has-part}}, e_{\text{gloss}}, e_{\text{is-in-gloss}}, e_{\text{topic}}, \ldots\}$, $N = \{ S_G, S_s, S_g, S_1, S_2, S_3, S_4, S_5, S_6, E_1, E_2, \ldots\}$, and P_G includes about 50 productions. An excerpt from the grammar is shown in Table 2.

As stated in the previous section, the weight $w(e_1, e_2, \ldots, e_n)$ of a semantic path $e_1, e_2, \ldots, e_n$ is given by the sum of the weights of the productions applied in the derivation $S_G \Rightarrow^* e_1, e_2, \ldots, e_n$. These weights have been experimentally established on standard word sense disambiguation data, such as the SemCor corpus, and have been normalized so that the weight of a semantic path always ranges between 0 and 1.

The main rules in G are as follows (S_1 and S_2 are two synsets in I):

12 http://xwn.hlt.utdallas.edu/papers.html.

```
SSI(T : list of terms, I : initial list of interpretation synsets)
{
    for each t ∈ T
        if (t is monosemous) I[t] = the only sense of t

    P := {t ∈ T : I[t] = ∅}

    { while there are more terms to disambiguate }
    do
    {
        P' := P
        for each t ∈ P' { for each pending term }
        {
            bestSense := ∅
            maxValue := 0
            { for each possible interpretation of t }
            for each sense S of t in WordNet
            {
                f[S] := 0
                for each synset S' ∈ I
                {
                    φ := 0
                    for each semantic path e1e2 ... en between S and S'
                        φ := φ + w(e1e2 ... en)

                    f[S] := f[S] + φ
                }
                if (f[S] > maxValue)
                {
                    maxValue := f[S]
                    bestSense := S
                }
            }
            if (maxValue > 0)
            {
                I[t] := bestSense
                P := P \ {t}
            }
        }
    } while(P ≠ P')

    return I
}
```

Figure 8
The SSI algorithm in pseudocode.

Table 2
Excerpt from the context-free grammar for the recognition of semantic interconnections.

$S_G \rightarrow S_s \mid S_g$	(all the rules)
$S_s \rightarrow S_1 \mid S_2 \mid S_3$	(simple rules)
$S_1 \rightarrow E_1 S_1 \mid E_1$	(hyperonymy/meronymy)
$E_1 \rightarrow e_{\text{kind}-\text{of}} \mid e_{part-of}$	
$S_2 \rightarrow E_2 S_2 \mid E_2$	(hyponymy/holonymy)
$E_2 \rightarrow e_{\text{has}-\text{kind}} \mid e_{\text{has}-\text{part}}$	
$S_3 \rightarrow e_{\text{kind}-\text{of}} S_3 e_{\text{has}-\text{kind}} \mid e_{\text{kind}-\text{of}} e_{\text{has}-\text{kind}}$	(parallelism)
$S_g \rightarrow e_{\text{gloss}} S_s \mid S_4 \mid S_5 \mid S_6$	(gloss rules)
$S_4 \rightarrow e_{\text{gloss}}$	(gloss rule)
$S_5 \rightarrow e_{\text{topic}}$	(topic rule)
$S_6 \rightarrow e_{\text{gloss}} e_{\text{is}-\text{in}-\text{gloss}}$	(gloss + gloss^{-1} rule)

1. color, if S_1 is in the same adjectival cluster as *chromatic*#3 and S_2 is a hyponym of a concept that can assume a color like *physical_object*#1 and *food*#1 (e.g., $S_1 \equiv$ *yellow*#1 and $S_2 \equiv$ *wall*#1)
2. domain, if the gloss of S_1 contains one or more domain labels and S_2 is a hyponym of those labels (for example, *white*#3 is defined as "(of wine) almost colorless," therefore it is the best candidate for *wine*#1 in order to disambiguate the term *white wine*)
3. synonymy, if

$$\text{(a) } S_1 \equiv S_2 \text{ or (b) } \exists N \in \mathit{Synsets} : S_1 \xrightarrow{\text{pert}} N \equiv S_2$$

   (for example, in the term *open air*, both the words belong to synset { *open*#8, *air*#2, . . . , *outdoors*#1 })
4. hyperonymy/meronymy path, if there is a sequence of hyperonymy/meronymy relations (for example, *mountain*#1 $\xrightarrow{\text{has-part}}$ *mountain peak*#1 $\xrightarrow{\text{kind-of}}$ *top*#3 provides the right sense for each word of *mountain top*)
5. hyponymy/holonymy path, if there is a sequence of hyponymy/holonymy relations (for example, in *sand beach*, *sand*#1 $\xrightarrow{\text{part-of}}$ *beach*#1);
6. parallelism, if S_1 and S_2 have a common ancestor (for example, in *enterprise company*, *organization*#1 is a common ancestor of both *enterprise*#2 and *company*#1)
7. gloss, if $S_1 \xrightarrow{\text{gloss}} S_2$ (for example, in *web site*, the gloss of *web*#5 contains the word *site*; in *waiter service*, the gloss of *restaurant attendant*#1, hyperonym of *waiter*#1, contains the word *service*)
8. topic, if $S_1 \xrightarrow{\text{topic}} S_2$ (for example, in the term *archeological site*, in which both words are tagged with sense 1 in a SemCor file; notice that WordNet provides no mutual information about them; also consider *picturesque village*: WordNet provides the example "a picturesque village" for sense 1 of *picturesque*)

9. gloss+hyperonymy/meronymy path, if $\exists G \in Synsets : S_1 \xrightarrow{\text{gloss}} G$ and there is a hyperonymy/meronymy path between G and S_2 (for example, in *railway company*, the gloss of *railway*#1 contains the word *organization* and *company*#1 $\xrightarrow{\text{kind-of}}$ *institution*#1 $\xrightarrow{\text{kind-of}}$ *organization*#1)

10. gloss+parallelism, if $\exists G \in Synsets : S_1 \xrightarrow{\text{gloss}} G$ and there is a parallelism path between G and S_2 (for example, in *transport company*, the gloss of *transport*#3 contains the word *enterprise* and *organization*#1 is a common ancestor of both *enterprise*#2 and *company*#1)

11. gloss+gloss, if $\exists G \in Synsets : S_1 \xrightarrow{\text{gloss}} G \xleftarrow{\text{gloss}} S_2$ (for example, in *mountain range*, *mountain*#1 and *range*#5 both contain the word *hill* so that the right senses can be chosen)

12. hyperonymy/meronymy+gloss path, if $\exists G \in Synsets : G \xleftarrow{gloss} S_2$ and there is a hyperonymy/meronymy path between S_1 and G

13. parallelism+gloss, if $\exists G \in Synsets : G \xleftarrow{\text{gloss}} S_2$ and there is a parallelism path between S_1 and G.

3.2.3 A Complete Example. We now provide a complete example of the SSI algorithm applied to the task of disambiguating a lexicalized tree $\mathcal{T}$. With reference to Figure 4, the list T is initialized with all the component words in $\mathcal{T}$, that is, [service, train, ferry, car, boat, car-ferry, bus, coach, transport, public_transport, taxi, express, customer].

Step 1. In T there are four monosemous words, *taxi*, *car-ferry*, *public_transport*, and *customer*; therefore, we have

$$I = [\text{taxi\#1}, \text{car\_ferry\#1}, \text{public\_transport\#1}, \text{customer\#1}]$$

$$P = \{\text{service}, \text{train}, \text{ferry}, \text{car}, \text{boat}, \text{bus}, \text{coach}, \text{transport}, \text{express}\}.$$

Step 2. During the second iteration, the following rules are matched:[13]

$$\{taxi\} \xrightarrow{\text{kind-of}} \{\text{car}, \text{auto}\}(\text{hyper})$$

$$\{\text{taxi}\} \xrightarrow{\text{kind-of}} \{\text{car}, \text{auto}\} \xrightarrow{\text{kind-of}} \{\text{motor\_vehicle,automotive\_vehicle}\} \xrightarrow{\text{kind-of}} \{\text{vehicle}\} \xleftarrow{\text{gloss}} \{\text{bus}, \text{autobus}, \text{coach}\}(\text{hyper} + \text{gloss})$$

$$\{\text{taxi}\} \xrightarrow{\text{kind-of}} \{\text{car}, \text{auto}\} \xrightarrow{\text{kind-of}} \{\text{motor\_vehicle,automotive\_vehicle}\} \xrightarrow{\text{kind-of}} \{\text{vehicle}\} \xleftarrow{\text{gloss}} \{\text{ferry}, \text{ferryboat}\}(\text{hyper} + \text{gloss})$$

$$\{\text{bus}, \text{autobus}, \text{coach}\} \xrightarrow{\text{kind-of}} \{\text{public\_transport}\}(\text{hyper})$$

$$\{\text{car\_ferry}\} \xrightarrow{\text{kind-of}} \{\text{ferry}, \text{ferryboat}\}(\text{hyper})$$

13 More than one rule may contribute to the disambiguation of a term. We list here only some of the detected patterns.

$$\{\text{customer}, \text{client}\} \xrightarrow{\text{topic}} \{\text{service}\}(\text{topic})$$

$$\{\text{service}\} \xrightarrow{\text{gloss}} \{\text{person}, \text{someone}\} \xrightarrow{\text{has-kind}} \{\text{consumer}\}$$
$$\xrightarrow{\text{has-kind}} \{\text{customer}, \text{client}\}(\text{gloss} + \text{hypo})$$

$$\{\text{train, railroad\_train}\} \xrightarrow{\text{kind-of}} \{\text{public\_transport}\}(\text{hyper})$$

$$\{\text{express}, \text{expressbus}\} \xrightarrow{\text{kind-of}} \{\text{bus}, \text{autobus}, \text{coach}\} \xrightarrow{\text{kind-of}} \{\text{public\_transport}\}(\text{hyper})$$

$$\{\text{conveyance}, \text{transport}\} \xrightarrow{\text{has-kind}} \{\text{public\_transport}\}(\text{hypo})$$

obtaining:

$$\text{I} = [\text{taxi\#1}, \text{car\_ferry\#1}, \text{public\_transport\#1}, \text{customer\#1}, \text{car\#1}, \text{ferry\#1}, \text{bus\#1}, \text{coach\#5}, \text{train\#1}, \text{express\#2}, \text{transport\#1}, \text{service\#1}]^{14}$$

$$P = \{boat\}.$$

Step 3.

$$\{\text{boat}\} \xrightarrow{\text{has-kind}} \{\text{ferry}, \text{ferryboat}\}(\text{hypo})$$

$$\text{I} = [\text{taxi\#1}, \text{car\_ferry\#1}, \text{public\_transport\#1}, \text{customer\#1}, \text{car\#1}, \text{ferry\#1}, \text{bus\#1}, \text{coach\#5}, \text{train\#1}, \text{express\#2}, \text{boat\#1}, \text{transport\#1}, \text{service\#1}]$$

$$\text{P} = \emptyset.$$

Then the algorithm stops since the list P is empty.

3.2.4 Creating Domain Trees. During the execution of the SSI algorithm, (possibly) all the terms in a lexicalized tree $\mathcal{T}$ are disambiguated. Subsequently, we proceed as follows:

a. Concept clustering: Certain concepts can be clustered in a unique concept on the basis of pertainymy, similarity, and synonymy (e.g., *manor house* and *manorial house*, *expert guide* and *skilled guide*, *bus service* and *coach service*, respectively); notice again that we detect semantic relations between concepts, not words. For example, *bus*#1 and *coach*#5 are synonyms, but this relation does not hold for other senses of these two words.

b. Hierarchical structuring: Taxonomic information in WordNet is used to replace syntactic relations with kind-of relations (e.g., *ferry service* $\xrightarrow{\text{kind-of}}$ *boat service*), on the basis of hyperonymy, rather than string inclusion as in $\mathcal{T}$.

14 Notice that *bus*#1 and *coach*#5 belong to the same synset, therefore they are disambiguated by the same rule.

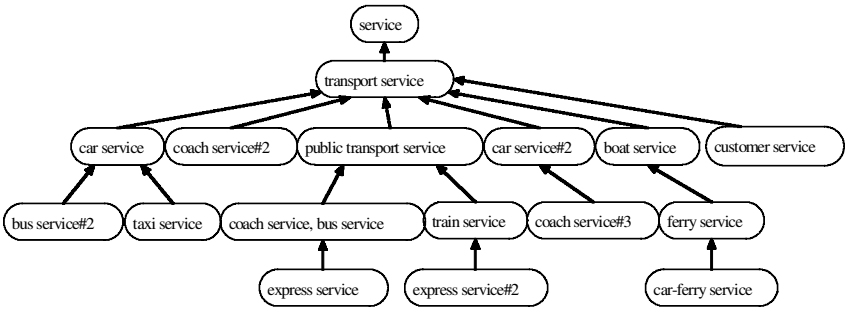

Figure 9
Domain concept tree.

Each lexicalized tree $\mathcal{T}$ is finally transformed into a **domain concept tree** Υ. Figure 9 shows the concept tree obtained from the lexicalized tree of Figure 4. For the sake of legibility, in Figure 9 concepts are labeled with the associated terms (rather than with synsets), and numbers are shown only when more than one semantic interpretation holds for a term. In fact, it is possible to find more than one matching hyperonymy relation. For example, an *express* can be a *bus* or a *train*, and both interpretations are valid, because they are obtained from relations between terms within the domain.

3.2.5 Adding Conceptual Relations. The second phase of semantic interpretation involves finding the appropriate semantic relations holding among concept components. In order to extract semantic relations, we need to do the following:

- Select an inventory of domain-appropriate semantic relations.
- Learn a formal model to select the relations that hold between pairs of concepts, given ontological information on these concepts.
- Apply the model to semantically relate the components of a complex concept.

First, we selected an inventory of semantic relations types. To this end, we consulted John Sowa's (1984) formalization on conceptual relations, as well as other studies conducted within the CoreLex,[15] FrameNet, and EuroWordNet (Vossen 1998) projects. In the literature, no systematic definitions are provided for semantic relations; therefore we selected only the more intuitive and widely used ones.

To begin, we selected a kernel inventory including the following 10 relations, which we found pertinent (at least) to the tourism and finance[16] domains: place (e.g., *room* $\overset{\text{PLACE}}{\longleftarrow}$ *service*, which reads "the service has place in a room" or "the room is the place of service"), time (*afternoon* $\overset{\text{TIME}}{\longleftarrow}$ *tea*), matter (*ceramics* $\overset{\text{MATTER}}{\longleftarrow}$ *tile*), topic (*art* $\overset{\text{TOPIC}}{\longleftarrow}$ *gallery*), manner (*bus* $\overset{\text{MANNER}}{\longleftarrow}$ *service*), beneficiary (*customer* $\overset{\text{BENEF}}{\longleftarrow}$ *service*), purpose (*booking* $\overset{\text{PURPOSE}}{\longleftarrow}$ *service*), object (*wine* $\overset{\text{OBJ}}{\longleftarrow}$ *production*), attribute (*historical* $\overset{\text{ATTR}}{\longleftarrow}$ *town*),

15 http://www.cs.brandeis.edu/~paulb/CoreLex/corelex.html
16 Financial terms are extracted from the *Wall Street Journal*.

characteristics (*first-class* $\overset{\text{CHRC}}{\longleftarrow}$ *hotel*). This set can be easily adapted or extended to other domains.

In order to associate the appropriate relation(s) that hold among the components of a domain concept, we decided to use **inductive machine learning**. In inductive learning, one has first to manually tag with the appropriate semantic relations a subset of domain concepts (this is called the **learning set**) and then let an inductive learner build a tagging model. Among the many available inductive learning programs, we experimented both with Quinlan's C4.5 and with TiMBL (Daelemans et al. 1999).

An inductive learning system requires selecting a set of features to represent instances in the learning domain. Instances in our case are concept-relation-concept triples (e.g., *wine* $\overset{\text{OBJ}}{\longleftarrow}$ *production*), where the type of relation is given only in the learning set.

We explored several alternatives for feature selection. We obtained the best result when representing each concept component by the complete list of its hyperonyms (up to the topmost), as follows:

$$\text{feature} - \text{vector}[[\text{list\_of\_hyperonyms}]^{*}_{\text{modifier}}[\text{list\_of\_hyperonyms}]_{\text{head}}]$$

For example, the feature vector for *tourism operator*, where *tourism* is the modifier and *operator* is the head, is built as the sequence of hyperonyms of *tourism*#1: [tourism#1, commercial_enterprise#2, commerce#1, transaction#1, group-action#1, act#1, human-action#1], followed by the sequence of hyperonyms for *operator*#2 [operator#2, capitalist#2, causal_agent#1, entity#1, life_form#1, person#1, individual#1].

Features are converted into a binary representation to obtain vectors of equal length. We ran several experiments, using a tagged set of 405 complex concepts, a varying fragment of which were used for learning, the remainder for testing (we used two-fold cross-validation). Overall, the best experiment provided a 6% error rate over 405 examples and produced around 20 classification rules.

The following are examples of extracted rules (from C4.5), along with their confidence factor (in parentheses) and examples:

$$\underline{\text{If in modifier}}\ [\text{knowledge\_domain\#1}, \text{knowledge\_base\#1}] = 1\ \underline{\text{then relation}}\ \text{THEME}(63\%)$$
$$\text{Examples} : \text{arts\_festival}, \text{science\_center}$$

$$\underline{\text{If in modifier}}\ [\text{building\_material\#1}] = 1\ \underline{\text{then relation}}\ \text{MATTER}(50\%)$$
$$\text{Examples} : \text{stave\_church}, \text{cobblestone\_street}$$

$$\underline{\text{If in modifier}}\ [\text{conveyance\#3}, \text{transport\#1}] = 1\ \underline{\text{and in head}}[\text{act\#1}, \text{human\_act\#1}] = 1\ \underline{\text{then relation}}\ \text{MANNER}(92.2\%)$$
$$\text{Examples} : \text{bus\_service}, \text{coach\_tour}$$

Selection and extraction of conceptual relations is one of the active research areas in the OntoLearn project. Current research is directed toward the exploitation of on-line resources (e.g., the tagged set of conceptual relations in FrameNet) and the automatic

generation of glosses for complex concepts (e.g., for *travel service* we have *travel*#1 $\overset{\text{PURPOSE}}{\longleftarrow}$ *service#1*: "**a kind** of service, work done by one person or group that benefits another, **for** travel, the act of going from one place to another"). Automatic generation of glosses (see Navigli et al. [2004] for preliminary results) relies on the compositional interpretation criterion, as well as the semantic information provided by conceptual relations.

3.3 Phase 3: Ontology Integration

The domain concept forest generated by OntoLearn is used to trim and update WordNet, creating a domain ontology. WordNet is pruned and trimmed as follows:

- After the domain concept trees are attached to the appropriate nodes in WordNet in either a manual or an automatic manner, all branches not containing a domain node can be removed from the WordNet hierarchy.
- An intermediate node in WordNet is pruned whenever the following conditions all hold:

  1. It has no "brother" nodes.
  2. It has only one direct hyponym.
  3. It is not the root of a domain concept tree.
  4. It is not at a distance greater than two from a WordNet unique beginner (this is to preserve a "minimal" top ontology).

Figure 10 shows an example of pruning the nodes located over the domain concept tree rooted at *wine*#1. The appendix shows an example of a domain-adapted branch of WordNet in the tourism domain.

4. Evaluation

The evaluation of ontologies is recognized to be an open problem.[17] Though the number of contributions in the area of ontology learning and construction has considerably increased in the past few years, especially in relation to the forthcoming Semantic Web, experimental data on the utility of ontologies are not available, other than those in Farquhar et al. (1998), in which an analysis of user distribution and requests is presented for the Ontology Server system. A better performance indicator would have been the number of users that access the Ontology Server on a regular basis, but the authors mention that regular users account for only a small percentage of the total. Efforts have recently being made on the side of ontology evaluation tools and methods, but available results are on the methodological rather than on the experimental side. The ontology community is still in the process of assessing an evaluation framework.

We believe that, in absence of a commonly agreed-upon schema for analyzing the properties of an ontology, the best way to proceed is evaluating an ontology *within* some existing application. Our current work is precisely in this direction: The results of a terminology translation experiment appear in Navigli, Velardi, and Gangemi (2003), while preliminary results on a query expansion task are presented in Navigli and Velardi (2003).

17 OntoWeb D.1.3 Tools (2001), "Whitepaper: Ontology Evaluation Tools," available at http://www.aifb.unikarlsruhe.de/WBS/ysu/publications/eon2002_whitepaper.pdf

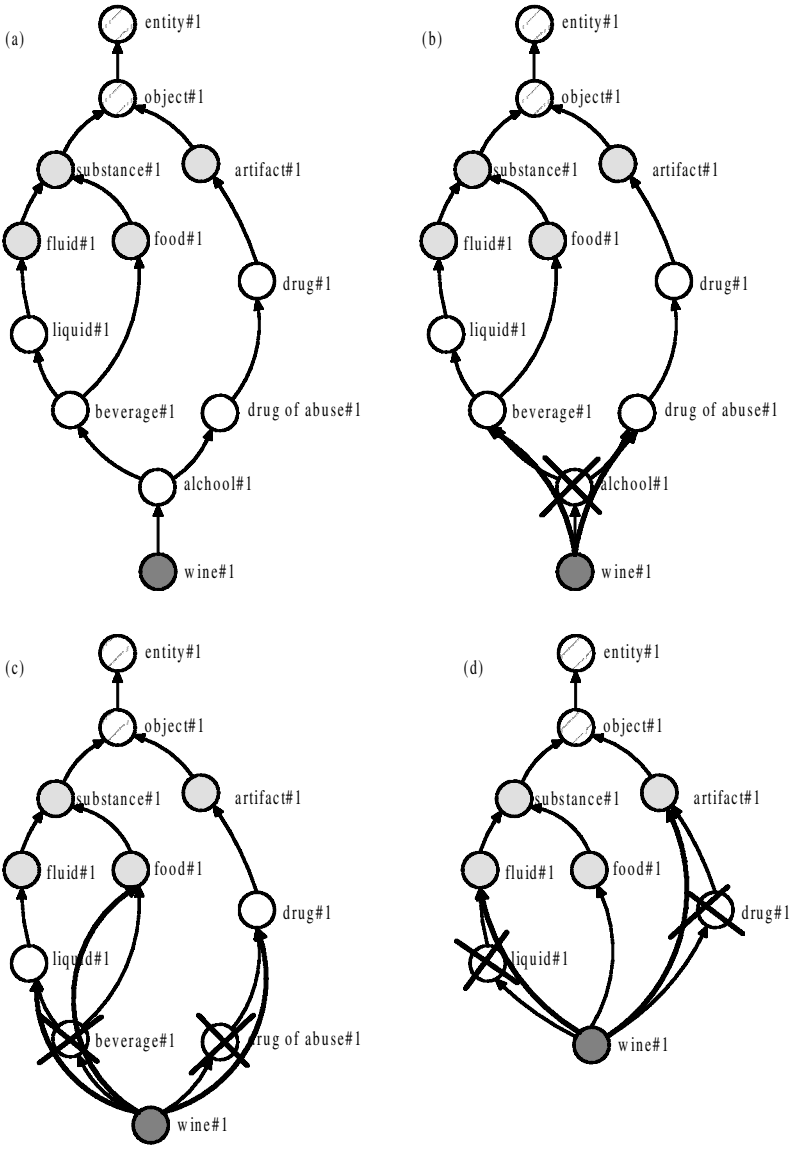

Figure 10
Pruning steps over the domain concept tree for *wine*1.

In this evaluation section we proceed as follows: First, we provide an account of the feedback that we obtained from tourism experts participating in the Harmonise EC project on interoperability in the tourism domain. Then, we evaluate in detail the SSI algorithm, which is the "heart" of the OntoLearn methodology.

4.1 OntoLearn as a Support for Ontology Engineers

During the first year of the Harmonise project, a core ontology of about three hundred concepts was developed using ConSys and SymOntoX. In parallel, we collected a corpus of about one million words from tourism documents, mainly descriptions of travels and tourism sites. From this corpus, OntoLearn extracted an initial list of 14,383

candidate terms (the first phase of terminology extraction in Section 3.1), from which the system derived a domain concept forest of 3,840 concepts, which were submitted to the domain experts for ontology updating and integration.

The Harmonise ontology partners lacked the requisite expertise to evaluate the WordNet synset associations generated by OntoLearn for each complex term, therefore we asked them to evaluate only the domain appropriateness of the *terms*, arranged in a hierarchical fashion (as in Figure 9). We obtained a precision ranging from 72.9% to about 80% and a recall of 52.74%.[18] The precision shift is due to the well-known fact that experts may have different intuitions about the relevance of a concept for a given domain. The recall estimate was produced by manually inspecting 6,000 of the initial 14,383 candidate terms, asking the experts to mark all the terms judged as "good" domain terms, and comparing the obtained list with the list of terms automatically filtered by OntoLearn (the phase of terminology filtering described in Section 3.1).

As a result of the feedback obtained from the tourism experts, we decided that experts' interpretation difficulties could indeed be alleviated by associating a textual definition with each new concept proposed by OntoLearn. This new research (automatic generation of glosses) was mentioned in Section 3.2.5. We still need to produce an in-field evaluation of the improved readability of the ontology enriched with textual definitions.

In any case, OntoLearn favored a considerable speed up in ontology development, since shortly after we provided the results of our OntoLearn tool, the Harmonise ontology reached about three thousand concepts. Clearly, the definition of an initial set of basic domain concepts is sufficiently crucial, to justify long-lasting and even heated discussions. But once an agreement is reached, filling the lower levels of the ontology can still take a long time, simply because it is a tedious and time-consuming task. Therefore we think that OntoLearn revealed itself indeed to be a useful tool within Harmonise.

4.2 Evaluation of the SSI Word Sense Disambiguation Algorithm

As we will argue in Section 5, one of the novel aspects of OntoLearn with respect to current ontology-learning literature is semantic interpretation of extracted terms. The SSI algorithm described in section 3.2 was subjected to several evaluation experiments by the authors of this article. The output of these experiments was used to tune certain heuristics adopted by the algorithm, for example, the dimension of the semantic graph (i.e., the maximum distance of a concept S' from the central concept S) and the weights associated with grammar rules. To obtain a domain-independent tuning, tuning experiments were performed applying the SSI algorithm on standard word sense disambiguation data,[19] such as SemCor and Senseval all-words.[20]

However, OntoLearn's main task is terminology disambiguation, rather than plain word sense disambiguation. In complex terms, words are likely to be more tightly semantically related than in a sentence; therefore the SSI algorithm seems more appropriate.[21] To test the SSI algorithm, we selected 650 complex terms from the set of 3,840 concepts mentioned in Section 4.1, and we manually assigned the appropriate

18 In a paper specifically dedicated to terminology extraction and evaluation (Velardi, Missikoff, and Basili 2001) we performed an evaluation also on an economics domain, with similar results.

19 In standard WSD tasks, the list T in input to the SSI algorithm is the set of all words in a sentence fragment to be disambiguated.

20 http://www.itri.brighton.ac.uk/events/senseval/ARCHIVE/resources.html#test

21 For better performance on a standard WSD task, it would be essential to improve lexical knowledge of verbs (e.g. by integrating VerbNet and FrameNet, as previously mentioned), as well as to enhance the grammar.

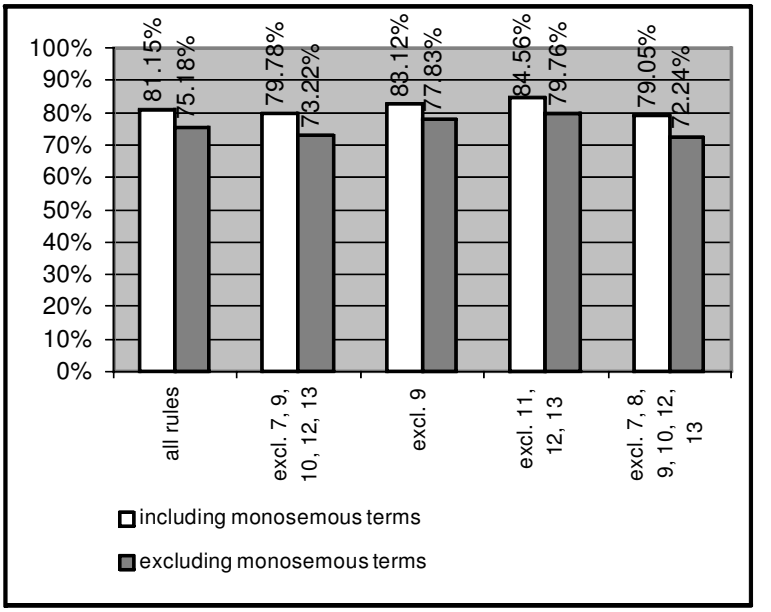

Figure 11
Different runs of the semantic disambiguation algorithm when certain rules in the grammar *G* are removed.

WordNet synset to each word composing the term. We used two annotators to ensure some degree of objectivity in the test set. In this task we experienced difficulties already pointed out by other annotators, namely, that certain synsets are very similar, to the point that choosing one or the other—even with reference to our specific tourism domain—seemed a mere guess. Though we can't say that our 650 tagged terms are a "gold standard," evaluating OntoLearn against this test set still produced interesting outcomes and a good intuition of system performance. Furthermore, as shown by the example of Section 3.2.3, OntoLearn produces a motivation for its choices, that is, the detected semantic patterns. Though it was not feasible to analyze in detail all the output of the system, we found more than one example in which the choices of OntoLearn were more consistent[22] and more convincing than those produced by the annotators, to the point that OntoLearn could also be used to support human annotators in disambiguation tasks.

First, we evaluated the effectiveness of the rules in *G* (Section 3.2.2) in regard to the disambiguation algorithm. Since certain rules are clearly related (for example, rules 4 and 5, rules 9 and 11), we computed the precision of the disambiguation when adding or removing groups of rules. The results are shown in Figure 11. The shaded bars in the figure show the results obtained when those terms containing unambiguous words are removed from the set of complex terms.

We found that the grammar rules involving the gloss and hyperonym relations contribute more than others to the precision of the algorithm. Certain rules (not listed in 3.2.2 since they were eventually removed) were found to produce a negative effect. All the rules described in 3.2.2 were found to give more or less a comparable positive contribution to the final performance.

22 Consistent at least with respect to the lexical knowledge encoded in WordNet.

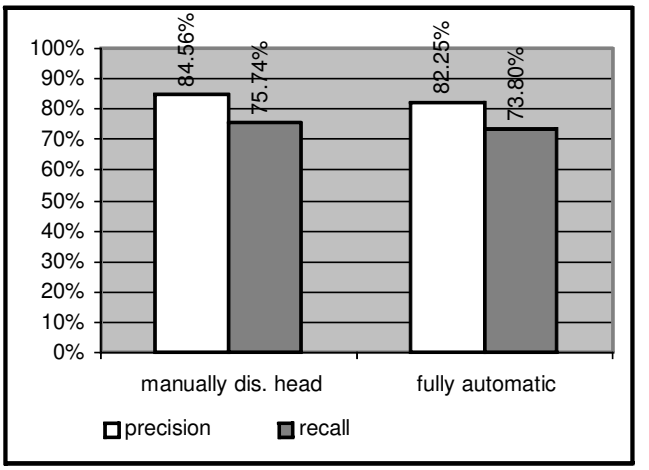

Figure 12
Precision and recall for the terminology disambiguation task: manual disambiguation of the head and fully automatic disambiguation.

The precision computed in Figure 11 refers to the case in which the head node of each term tree is sense-tagged manually. In Figure 12 the light and dark bars represent precision and recall, respectively, of the algorithm when the head (i.e., the root) of a term tree is manually assigned and when the disambiguation is fully automatic. The limited drop in performance (2%) of the fully automated task with respect to manual head disambiguation shows that, indeed, the assumption of a strong semantic interrelationship between the head and the other terms of the term tree is indeed justified.

Finally, we computed a baseline, comparing the performance of the algorithm with that obtained by a method that always chooses the first synset for each word in a complex term. (We remind readers that in WordNet, the first sense is the most probable.) The results are shown in Figure 13, where it is seen, as expected, that the increment in performance with respect to the baseline is higher (around 5%) when only polysemous terms are considered. A 5% difference (3% with respect to the fully automatic disambiguation) is not striking, however, the tourism domain is not very technical, and often the first sense is the correct one. We plan in the future to run experiments with more technical domains, for example, economics or software products.

5. Related Work

Comprehensive ontology construction and learning has been an active research field in the past few years. Several workshops[23] have been dedicated to ontology learning and related issues. The majority of papers in this area propose methods for extending an existing ontology with unknown words (e.g., Agirre et al. 2000 and Alfonseca and Manandhar 2002). Alfonseca and Manandhar present an algorithm to enrich WordNet with unknown concepts on the basis of hyponymy patterns. For example, the pattern $hypernism(N_2, N_1) :- appositive(N_2, N_1)$ captures a hyponymy relation between *Shakespeare* and *poet* in the appositive NP "Shakespeare, the poet." This approach heavily

23 ECAI-2000 First Workshop on Ontology Learning (http://ol2000.aifb.uni-karlsruhe.de/) and IJCAI-2001 Second Workshop on Ontology Learning (http://ol2001.aifb.uni-karlsruhe.de/).

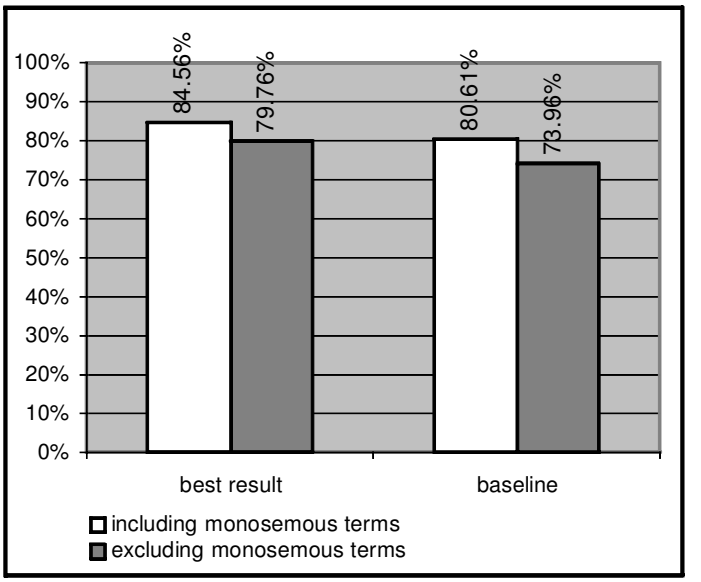

Figure 13
Comparison with a baseline.

depends upon the ability of discovering such patterns, however, it appears a useful complementary strategy with respect to OntoLearn. OntoLearn, in fact, is unable to analyze totally unknown terms (though ongoing research is in progress to remedy this limitation). Berland and Charniak (1999) propose a method for extracting whole-part relations from corpora and enrich an ontology with this information. Few papers propose methods of extensively enriching an ontology with domain terms. For example, Vossen (2001) uses statistical methods and string inclusion to create lexicalized trees, as we do (see Figure 4). However, no semantic disambiguation of terms is performed. Very often, in fact, ontology-learning papers regard domain *terms as concepts*. A statistical classifier for automatic identification of semantic roles between co-occuring terms is presented in Gildea and Jurafsky (2002). In order to tag texts with the appropriate semantic role, Gildea and Jurafsky use a training set of fifty thousand sentences manually annotated within the FrameNet semantic labeling project. Finally, in Maedche and Staab (2000, 2001), an architecture is presented to help ontology engineers in the difficult task of creating an ontology. The main contribution of this work is in the area of ontology engineering, although machine-learning methods are also proposed to automatically enrich the ontology with semantic relations.

6. Conclusions and Ongoing Developments

We believe that the OntoLearn system is innovative in several respects:

1. in presenting an overall ontology development system.
2. in stressing the importance of appropriate terminology extraction to the ontology-building enterprise.
3. in avoiding a common confusion between domain *terms* and domain *concepts*, since it performs a *semantic interpretation* of terms. This is indeed the strongest aspect of our method.

4. in presenting a new structural approach to sense classification (SSI). This method is general and has been applied to other sense disambiguation tasks, such as sense-based query expansion (Navigli and Velardi 2003) and gloss disambiguation (Gangemi, Navigli, and Velardi 2003).

Ontology learning is a complex enterprise, and much is left to be done. We list here some of the drawbacks and gaps of our method, along with hints for ongoing and future developments. OntoLearn is in fact a fully active area of research within our group.

1. The SSI method presupposes that each term component has at least one synset in WordNet. In our ongoing research, we try to cope with this limitation, parsing textual definitions in glossaries (e.g., in a computer network application) whenever a term cannot be interpreted compositionally in WordNet. Terms in glossaries are first arranged in trees according to detected taxonomic relations, then the head terms of each tree are attached to the appropriate node of WordNet, if an appropriate node indeed exists. Rule-based and algebraic methods are jointly used to construct term trees and to compute measures of the similarity between the textual definitions in glossaries and those in WordNet.
2. OntoLearn detects taxonomic relations between complex concepts and other types of semantic relations among the components of a complex concept. However, an ontology is more than a taxonomy. The result of concept disambiguation in OntoLearn is more than an ordered list of synsets, since we obtain semantic nets and intersecting patterns among them (Section 3.2.2). This information is not currently exploited to generate richer concept definitions. A preliminary attempt to generate formal concept definitions from informal ones is described in Gangemi, Navigli, and Velardi (2003). Furthermore, an automatic gloss generation algorithm has been defined (Navigli et al. 2004).
3. A large-scale evaluation is still to be done. As we have already pointed out, evaluation of ontologies is recognized as an open problem, and few results are available, mostly on the procedural ("how to") side. We partly evaluated OntoLearn in an automatic translation task (Navigli, Velardi, and Gangemi 2003), and the SSI algorithm in generic WSD tasks as mentioned in item 4 of the previous list. In addition, it would be interesting to run OntoLearn on different domains, in order to study the effect of higher or lower levels of ambiguity and technicality on the output domain ontology.

Appendix: A Fragment of Trimmed WordNet for the Tourism Domain

{ activity%1 }
{ work%1 }
{ project:00508925%n }
    { tourism_project:00193473%n }
    { ambitious_project:00711113%a }
{ service:00379388%n }

{ travel_service:00191846%n }
    { air_service#2:00202658%n }
    { air_service#4:00194802%n }
{ transport_service:00716041%n }
    { ferry_service#2:00717167%n }
    { express_service#3:00716943%n }
{ exchange_service:02413424%n }
{ guide_service:04840928%n }
{ restaurant_service:03233732%n }
{ rail_service:03207559%n }
{ customer_service:07197309%n }
    { guest_service:07304921%n }
    { regular_service#2:07525988%n }
    { outstanding_customer_service:02232741%a }
{ tourism_service:00193473%n }
{ waiter_service:07671545%n }
{ regular_service:02255650%a,scheduled_service:02255439%a }
{ personalized_service:01703424%a,personal_service:01702632%a }
{ secretarial_service:02601509%a }
{ religious_service:02721678%a }
    { church_service:00666912%n }
{ various_service:00462055%a }
{ helpful_service:02376874%a }
{ quality_service:03714294%n }
    { air_service#3:03716758%n }
{ room_service:03250788%n }
    { maid_service:07387889%n }
    { laundry_service:02911395%n }
    { car_service#5:02364995%n }
    { hour_room_service:10938063%n }
{ transport_service#2:02495376%n }
    { car_service:02383458%n }
        { bus_service#2:02356871%n }
        { taxi_service:02361877%n }
    { coach_service#2:02459686%n }
    { public_transport_service:03184373%n }
        { bus_service:02356526%n,coach_service:02356526%n }
            { express_service#2:02653414%n }
            { local_bus_service:01056664%a }
        { train_service:03528724%n }
            { express_service:02653278%n }
    { car_service#2:02384604%n }
        { coach_service#3:03092927%n }
    { boat_service:02304226%n }
        { ferry_service:02671945%n }
            { car-ferry_service:02388365%n }
{ air_service:05270417%n }
    { support_service:05272723%n }

References

Agirre, Eneko, Olatz Ansa, Eduard, Hovy, and David Martínez. 2000. Enriching very large ontologies using the WWW. In *ECAI Ontology Learning Workshop 2000*, available at http://ol2000.aifb.uni-karlsruhe.de/

Alfonseca, Enrique and Suresh Manandhar. 2002. Improving an ontology refinement Method with hyponymy patterns. In *Language Resources and Evaluation (LREC-2002)*, LasPalmas, Spain, May.

Basili, Roberto, Maria Teresa Pazienza, and Paola Velardi. 1996. An empirical symbolic approach to natural language processing, *Artificial Intelligence*, 85(1–2): 59–99.

Basili, Roberto, Maria Teresa Pazienza, and Fabio Massimo Zanzotto. 1998. A robust parser for information extraction. In *Proceedings of the European Conference on Artificial Intelligence (ECAI '98)*, Brighton, U.K., August.

Berland, Matthew and Eugene Charniak. 1999. Finding parts in very large corpora. In *Proceedings of the the 37th Annual Meeting of the Association for Computational Linguistics (ACL-99)*, College Park, MD.

Berners-Lee, Tim. 1999. *Weaving the Web.* Harper, San Francisco.

Bunke, Horst and Alberto Sanfeliu, editors. 1990. *Syntactic and Structural Pattern Recognition: Theory and Applications.* World Scientific.

Church, Kenneth Ward and Patrick Hanks. 1989. Word association norms, mutual information and lexicography. In *ACL-89*, Vancouver, British Columbia, Canada.

Daelemans, Walter, Jakub Zavrel, Ko van der Sloot, and Antal van den Bosch. 1999. TiMBL: Tilburg Memory Based Learner, version 2.0, reference manual. Technical Report ILK-9901, ILK, Tilburg University, Tilburg, the Netherlands.

Farquhar, Adam, Richard Fikes, Wanda Pratt, and James Rice. 1998. "Collaborative Ontology Construction for Information Integration." http://www-ksl-svc.stanford.edu:5915/doc/project-papers.html.

Fellbaum, Christiane, editor. 1995. *WordNet: An Electronic Lexical Database.* MIT Press, Cambridge, MA.

Formica, Anna, and Michele Missikoff. 2003. Ontology Validation in OPAL. In *2003 International Conference on Web Services (ICWS' 03)*, Las Vegas, NV. Springer.

Gangemi, Aldo, Nicola Guarino, Claudio Masolo, Alessandro Oltramari, and Luc Schneider. 2001. Sweetening ontologies with DOLCE. In *Proceedings of EKAW02*, Siguenza, Spain. Springer, pages 166–181

Gangemi, Aldo, Roberto Navigli, and Paola Velardi. 2003. Axiomatising WordNet: A hybrid methodology. In *Workshop on Human Language Technology for the Semantic Web and Web Services,* Held in Conjunction with *Second International Semantic Web Conference,* Sanibel Island, FL.

Gildea, Daniel and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics,* 28(3): 245–288.

Jacquemin, Christian. 1997. Guessing morphology from terms and corpora. In *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR' 97)*, Philadelphia, PA, pages 156–167.

Lenat, Douglas. 1993. CYC: A large scale investment in knowledge infrastructure. *Communications of the ACM*, 3(11).

Maedche, Alexander and Steffen Staab. 2000. Semi-automatic engineering of ontologies from text. In *Proceedings of the 12th International Conference on Software Engineering and Knowledge Engineering (SEKE'2000)*, Chicago, IL.

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

Magnini, Bernardo and Gabriella Cavaglia. 2000. Integrating subject field codes into WordNet. In *Proceedings of the second International Conference on Language Resources and Evaluation (LREC2000)*, Atenes.

Missikoff, Michele and X.F. Wang. 2001. Consys—A group decision-making support system for collaborative ontology building. In *Proceedings of Group Decision & Negotiation 2001 Conference*, La Rochelle, France.

Navigli, Roberto, and Paola Velardi. 2003. An analysis of ontology-based query expansion strategies. *Workshop on Adaptive Text Extraction and Mining*, held in conjunction with ECML 2003, Cavtat Dubrovnik, Croatia, September 22.

Navigli, Roberto, Paola Velardi, Alessandro Cucchiarelli, and Francesca Neri. 2004. Extending and enriching WordNet with OntoLearn. In Second *Global WordNet Conference*, Brno, Czech Republic, January 20–23. Springer-Verlag.

Navigli, Roberto, Paola Velardi, and Aldo Gangemi. 2003. Corpus driven ontology learning: A method and its application to automated terminology translation, *IEEE Intelligent Systems*, 18(1): 11–27.

Oltramari, Alessandro, Aldo Gangemi, Nicola Guarino, and Claudio Masolo. 2002. Restructuring WordNet's top-level: The OntoClean approach. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC2002)*, Las Palmas, Spain.

Smith, Barry and Christopher A. Welty. 2001. Ontology: Towards a new synthesis. *Formal Ontology in Information Systems*, ACM Press.

Sowa, John F. 1984. *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley, Reading, MA.

Uschold, Mike and Michael Gruninger. 1996. Ontologies: Principles, methods and applications. *Knowledge Engineering Review*, 11(2).

Velardi, Paola, Michele Missikoff, and Roberto Basili. 2001. Identification of relevant terms to support the construction of domain ontologies. In *ACL-EACL Workshop on Human Language Technologies*, Toulouse, France, July.

Vossen, Piek, editor. 1998. *EuroWordNet: A Multilingual Database with Lexical Semantic Networks*. Kluwer Academic, Dordrecht, Netherlands.

Vossen, Piek. 2001. Extending, trimming and fusing WordNet for technical documents. In *NAACL 2001 Workshop on WordNet and Other Lexical Resources*, Pittsburgh, July.

Yamamoto, Mikio and Kenneth W. Church. 2001. Using suffix arrays to compute term frequency and document frequency for all substrings in a corpus. *Computational Linguistics*, 27(1): 1–30.

Yokoi, Toshio. 1993. The EDR electronic dictionary. *Communications of the ACM*, 38(11): 42–44.