# Ontology Generation through the Fusion of Partial Reuse and Relation Extraction

**Nwe Ni Tun and Jin Song Dong**
School of Computing
National University of Singapore
Lower Kent Ridge, Singapore 117543
{tunnn,dongjs}@comp.nus.edu.sg

**Abstract**

Ontology generation—a process to automatically create ontologies from existing knowledge sources—has become a key issue with the emergence of the semantic web. Though many researchers are trying to automate this process by exploiting machine learning and data mining techniques, the results remain under exploration. At the same time, when more and more ontologies are available online, it is important to reuse existing ontologies to a certain extent. In this paper, we present a semi-automatic ontology generation system (OntoGenerator) by partially reusing existing ontologies via a modularization technique and a ranking strategy. In order to enrich the semantics of the generated ontology, we integrate natural language-based, non-taxonomic relation extraction into the system. OntoGenerator is aimed at supporting ontology reuse in semantic indexing. Another objective is to evaluate the maturity of the semantic web by applying its technologies in ontology generation.

## Introduction

With the rapid development of the WWW, the amount of online information has increased exponentially. Several approaches using machine learning and data mining techniques have attempted to extract ontologies from existing knowledge bases (Sleeman and Schorlemmer 2003), legacy databases (Maedche and Staab 2001), and text corpora (Brewster and Wilks 2003). One difficulty with those approaches is that background knowledge structure is not always explicitly expressed in the knowledge sources, particularly in the later approaches. At the same time, when more and more ontology library systems (e.g., Protégé, Ontolingua, DAML) and more powerful ontology search engines (e.g., Swoogle, Watson, and OntoSearch) become available, it is reasonable to reuse existing ontologies to a certain extent, so that the process of ontology generation from existing knowledge sources can be accomplished with satisfactory automation and less cost.

We summarize a number of reuse advantages. First, it reduces the human labor involved in developing ontologies from scratch. Second, it increases the quality of new ontologies because the reused modules have already been verified to a certain degree. Third, when two ontologies share common modules, the mapping process between them becomes simpler because of the reduced heterogeneity. Fourth, ontology reuse can support ontology maintenance. Finally, semantic web users can practice ontology construction through reuse techniques.

In the early stages of ontology development, most ontology designers are reluctant to reuse existing ontologies. The main reason for this include the limited reuse mechanism and supporting tools. For instance, owl:Imports—bringing all axioms of the imported ontology into the target ontology—is the only construct that OWL[1] supports for reuse. When a new ontology fully imports a number of ontologies with unneeded portions, its size and complexity become critical issues. Thus, some researchers have instead given their attention to partial reuse. Garu and his colleagues (Garu and Sattler 2007) presented a mechanism of automatic partitioning—dividing an ontology into a number of logically discriminated partitions—based on the $\mathcal{E}$-connection language (Grau and Kalyanpur 2005). SWOOP[2] supports a functionality of such partitions. However, it is limited since it is independent from user requirements. For semantic indexing—an attempt to compose semantic description of web documents with meta-data and markups, user requirement plays an important role to achieve a relevant ontology. Therefore, we propose a technique to extract a modular ontology based on user requirements (a desired set of keywords) as well as minimal size and complexity.

Gruber's definition of ontology is *a formal, explicit specification of a shared conceptualization*. This implies that the semantics given in ontologies should be explicit and machine understandable. With respect to this, we favor on explicitness in applying some well-known measures for ontology module selection and ranking. Then, the system generates a customized ontology by integrating some highly relevant modular ontologies. Moreover, we pursued the intuitive idea of semantic enrichment, which is the extraction of non-taxonomic relations from implicit



[1] OWL is W3C standard Ontology Web Language http://www.w3.org/TR/2003/PR-owl-ref-20031215/

[2] http://www.mindswap.org/2004/SWOOP/

Table 1: The basic constructors of $\mathcal{SHOIQ}$

| Constructor | DL syntax |
|---|---|
| intersectionOf | $C_1 \sqcap ... \sqcap C_n$ |
| unionOf | $C_1 \sqcup ... \sqcup C_n$ |
| complementOf | $\neg C$ |
| oneOf | $\{a_1\} \sqcup ... \sqcup \{a_n\}$ |
| allValuesFrom | $\forall P.C$ |
| someValuesFrom | $\exists P.C$ |
| maxCardinality | $\leq nP.C$ |
| minCardinality | $\geq nP.C$ |

knowledge sources (particularly annotated definitions available in the form of `rdfs:comment`) and addition of these relations into the generated ontology. Since verbs play an essential role in the relations between a subject (`owl:Class`) and object (`owl:Class`) (Kavalec and Svaték 2005; Schutz and Buitelaar 2005), verb centric triples such as $\langle Subject, Verb, Object\rangle$ are the major focus of non-taxonomic relations. The idea is developed by utilizing the generated ontology together with natural language parsing techniques. We call this procedure *ontology-based relation extraction*.

In this paper, a semi-automatic ontology generation system (OntoGenerator) is presented by employing existing semantic web technologies in a series of processes, such as ontology search, modularization, module ranking and selection, module integration, relation extraction, and interactive review of the generated ontology. We also evaluate the OntoGenerator system by examining the quality of generated ontologies in comparison to related works. Semantic indexing is designated as a potential application for `OntoGenerator`.

## Preliminaries

Ontology generation through partial reuse is an integrated process that involves ontology modularization—extracting some relevant ontology modules from existing ontologies—and the reuse of those modules as customized ontologies in integration. The formalisms for modular ontologies studied in this paper underlie OWL DL[3] that corresponds to the description logics language $\mathcal{SHOIQ}$ (Baader and Sattler 2007; Baader and Nardi 2003). Thus, we briefly introduce its syntax and semantics.

The syntax of $\mathcal{SHOIQ}$ is given by a signature $\mathbf{S}$ and a set of constructors. Signature $\mathbf{S} = \{\mathbf{C} \cup \mathbf{R} \cup \mathbf{I}\}$ is the (disjoint) union of a set $\mathbf{C}$ of atomic concepts $(C_1, C_2, ...)$ representing sets of elements, a set $\mathbf{R}$ of atomic roles $(P_1, P_2, ...)$ representing binary relations between elements, and a set $\mathbf{I}$ of individuals $(a_1, a_2, ...)$ representing elements. Table 1 and 2 describe $\mathcal{SHOIQ}$ constructors and axioms respectively, where $n$ is a positive integer.

The logical entailment $\models$ is defined using the model theoretic semantics[4]. Given signature $\mathbf{S}$, an interpretation $\mathcal{I}$ is a pair $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$. In this pair, $\Delta^{\mathcal{I}}$ is a non-empty set called the domain of the interpretation and $\cdot^{\mathcal{I}}$ is the interpretation function that assigns to every $C \in \mathbf{C}$ a subset $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, to every $P \in \mathbf{R}$ a binary relation $P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and to every $a \in \mathbf{I}$ an element $a^{\mathcal{I}} \in I$. Table 3 describes some basic interpretations in $\mathcal{SHOIQ}$ where $\mathbf{N}P$ is number restriction of role $P$. An interpretation $\mathcal{I}$ is a model of an ontology $O$ if $\mathcal{I}$ satisfies all axioms in $\mathcal{O}$.

[3] `http://www.w3.org/TR/owl-ref/`

[4] `http://www.w3.org/TR/owl-semantics/direct.html`

Table 2: The basic axioms of $\mathcal{SHOIQ}$

| Axiom | DL syntax |
|---|---|
| subClassOf | $C_1 \sqsubseteq \top$ |
| equivalentClass | $C_1 \equiv C_2$ |
| disjointWith | $C_1 \sqsubseteq \neg C_2$ |
| sameIndividualAs | $\{a_1\} \equiv \{a_2\}$ |
| differentFrom | $\{a_1\} \sqsubseteq \neg\{a_2\}$ |
| subPropertyOf | $P_1 \sqsubseteq P_2$ |
| equivalentProperty | $P_1 \equiv P_2$ |
| inverseOf | $P_1 \equiv P_2^-$ |
| transitiveProperty | $P^+ \sqsubseteq P$ |
| functionalProperty | $\top \sqsubseteq \leq 1P$ |
| inverseFunctionalProperty | $\top \sqsubseteq \leq 1P^-$ |

Table 3: Some basic interpretations in $\mathcal{SHOIQ}$

| Concepts |
|---|
| $(\neg C^{\mathcal{I}}) = \Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}$ |
| $(C_1 \sqcap C_2)^{\mathcal{I}} = {C_1}^{\mathcal{I}} \cap {C_2}^{\mathcal{I}}$ |
| $(C_1 \sqcup C_2)^{\mathcal{I}} = {C_1}^{\mathcal{I}} \cup {C_2}^{\mathcal{I}}$ |
| $(\forall P.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid P^{\mathcal{I}}(a, \neg C) = \emptyset\}$ |
| $(\exists P.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid P^{\mathcal{I}}(a, C) \neq \emptyset\}$ |
| $(\leq nP.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \mathbf{N}P^{\mathcal{I}}(a, C) \leq n\}$ |
| $(\geq nP.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \mathbf{N}P^{\mathcal{I}}(a, C) \geq n\}$ |
| Axioms |
| $(C_1 \sqsubseteq C_2)^{\mathcal{I}} = {C_1}^{\mathcal{I}} \subseteq {C_2}^{\mathcal{I}}$ |
| $(P_1 \sqsubseteq P_2)^{\mathcal{I}} = {P_1}^{\mathcal{I}} \subseteq {P_2}^{\mathcal{I}}$ |
| $(P^+)^{\mathcal{I}} = \{\forall a_1, a_2, a_3 \in \Delta^{\mathcal{I}} \mid P^{\mathcal{I}}(a_1, a_2) \cap P^{\mathcal{I}}(a_2, a_3) \rightarrow P^{\mathcal{I}}(a_1, a_3)\}$ |

We extend the notion of formal ontology with partial reuse, following the above syntax and semantics.

**Definition 1 (Formal Ontology)** *A formal ontology denoted by $O_j$ is a set of axioms given on signature $Sig(O_j)$—which consists of classes ($\mathbf{C}_j$), roles/properties ($\mathbf{R}_j$), and individuals ($\mathbf{I}_j$)—given in DL language $\mathcal{SHOIQ}$. Index $j$ differentiates one ontology from others.*

**Definition 2 (Ontology Module)** *An ontology module $O_{jM_k} \subseteq O_j$ is a set of direct axioms that are extracted from Ontology $O_j$ for signature $M_k \subseteq Sig(O_j)$ where $k$ is the index of the signature.*

**Definition 3 (PR Ontology)** *A partially reused (PR) ontology $O_{r_i} = \langle \{O_{jM_k}\}, O_i\rangle$ is a union of a list of selected ontology modules $\{O_{jM_k}\}$ for a number of signatures $\{M_k\} \subseteq Sig(O_{r_i})$ and the local ontology $O_i$ that reuses $\{O_{jM_k}\}$. Note that $i \neq j$.*

## Four Well-known Measures for Ontology Selection and Ranking

In this section, we represent four well-known structural measures that are applied to ontology selection and ranking. We mainly adopt these measures from AKTiveRank (Alani and Brewster 2006).

### Class Match Measure

The Class Match Measure (CMM) is a name-based measure that is used to search and select ontologies (Ding and Sachs 2004; Buitelaar and Declerck 2004; Alani and ShadBolt 2006).

**Definition 4 (Class Match Measure)** *Let $O_j$ be a candidate ontology and $T$ be a set of interested keywords. The class match measure of ontology $O_j$, denoted by $CMM(O_j)$, is calculated as:*

$$Exact\ Match:\ E(O_j,T) = \sum_{C\in\mathbf{C}_j}\sum_{t\in T} I(C,t)$$

$$PartialMatch:\ P(O_j,T) = \sum_{C\in\mathbf{C}_j}\sum_{t\in T} J(C,t)$$

$$I(C,t) = \begin{cases} 1 & \textit{if label(C) = t} \\ 0 & \textit{otherwise} \end{cases}$$

$$J(C,t) = \begin{cases} 1 & \textit{if label(C) contains t} \\ 0 & \textit{otherwise} \end{cases}$$

$$CMM(O_j,T) = \alpha E(O_j,T) + \beta P(O_j,T)$$

It is obvious that ontologies covering all keywords with exact matches are better than those with partial matches. Thus, $\alpha = 0.6$ and $\beta = 0.4$ are the weight factors set up to favor exact matches over partial matches. Note that we consider both the goodness and weakness of the partial match, such that we obtain more related classes and properties as well as irrelevant ones. For instance, we get `Research Student` for keyword "Research" and `Projector` for keyword "Project". However, we observe that the ratio of relevance is higher than irrelevance because of the multiple keyword search.

### Density Measure

For a class, the degree of explicitness is measured primarily with the number of classes and properties that directly belong to the class. The density measure (DEM) of a class is calculated based on the degree of explicitness which takes into account the number of classes and properties connected to the class.

**Definition 5 (Density Measure)** *For any class $C \in \mathbf{C}_j$, let $S = \{S_1, S_2, S_3, S_4, S_5, S_6\}$ be the sub-classes, super-classes, siblings, relations, attributes and constraints, of $C$. The density measure of ontology $O_j$, $DEM(O_j)$, is computed as follows:*

$$DEM(C) = \sum_{i=1}^{6} w_i \mid S_i \mid, DEM(O_j) = \frac{1}{n}\sum_{i=1}^{n} DEM(C)$$

*where $n = \mid E(O_j,T) \mid + \mid P(O_j,T) \mid$ is the number of namely-matched classes via CMM. The weight value of each $w$ can be either equally or conditionally distributed.*

### Semantic Similarity Measure

Ontologies can be viewed as semantic graphs of concepts and relations. Rada (Rada and Blettner 1989) introduced the shortest path-based similarity measure. Its variations have been used in (Maedche and Staab 2001; Weinstein and Birmingham 1999; Alani and ShadBolt 2006). The Semantic Similarity Measure (SSM) examines the semantic connectivity between classes through the shortest path—the path with minimum number of links (via both taxonomic and non-taxonomic relationships) that connects a pair of concepts.

**Definition 6 (Semantic Similarity Measure)** *For any two classes $C_f, C_g \in \mathbf{C}_j$, let $C_f \xrightarrow{p} C_g$ be a path $p \in P$ between classes $C_f$ and $C_g$. The semantic similarity measure of $O_j$, $SSM(O_j)$, is calculated as:*

$$SSM(C_f,C_g) = \frac{1}{length(min(p\in P))}\ \textit{if}\ f \neq g$$

$$SSM(C_f,C_g) = 1\ \textit{if}\ f = g$$

$$SSM(O_j) = \frac{1}{n}\sum_{f=1}^{n-1}\sum_{g=f+1}^{n} SSM(C_f,C_g)$$

### Betweeness Measure

JUNE (Ding and Kolari 2005) provides a betweenness measure (BEM) to calculate the number of shortest paths that pass through each node in a graph. If the class with the highest BEM value in an ontology is graphically the most central to that ontology.

**Definition 7 (Betweenness Measure)** *The betweeness measure of ontology $O_j$, symbolized by $BEM(\mathbf{O}_j)$, is computed as follows:*

$$BEM(O_j) = \frac{1}{n}\sum_{h=1}^{n} BEM(C_h)$$

$$BEM(C_h) = \sum_{C_f\neq C_g\neq C_h\in\mathbf{C_j}} \frac{\sigma_{C_f,C_g}(C_h)}{\sigma_{C_f,C_g}}$$

where $\sigma_{C_f,C_g}$ is the shortest path from $C_f$ to $C_g$ and $\sigma_{C_f,C_g}(C_h)$ is the number of shortes paths that passes through $C_h$.

## Analysis of Ontology Modularization Techniques & Criteria

The objective of this section is to provide an overview of existing modularization techniques and discuss their fundamental criteria.

The survey in (Seidenberg and Rector 2006) is broadly focused towards comparing a number of techniques including query-based, network partitioning-based, and traversal-based modularization techniques. Query-based methods obtain ontology modules as answers to ontology queries given in `SparQL` or `OWL-QL`. Network partitioning-based methods consider ontologies as graphs and exploit graph partitioning algorithms to take out structurally relevant modules. In this strategy, some modules may have only a single class

whereas others hold the whole ontology. Traversal-based modularization methods extract some ontological components (e.g., classes, properties, and individuals) that all are syntactically linked to the user requirements. In (Spefano 2005), modularization is considered as a process to design a new ontology by obtaining modular ontologies from existing ontologies. This is objectively related to our work. Nevertheless, partitioning approaches are mainly discussed.

In the literature, there is a work by (d'Aquin and Sabou 2007) compares the results of ontology selection returned by partitioning-based tools such as `Swoop` and `Pato`[5], with the results produced by modularization-based tools such as `KMi`[6] and `Prompt`[7]. The work concluded that modularization is more appropriate than partitioning for the purpose of knowledge selection regarding the user requirements. Therefore, we learned existing modularization techniques and organized some of their criterias as the commond ground for our modularization tool in `OntoGenerator`.

Sabou and colleagues (d'Aquin and Motta 2006a) discuss a number of modularization approaches in the literature. These can be divided into two common approaches. The first approach involves including all super/sub-classes of a candidate class in an extracted module. The second approach involves taking out only direct super/sub-classes. It is obvious that the first approach generates large sized ontology modules with extra knowledge. As the motivation of partial reuse is the avoidance of unnecessary size and complexity, our modularization technique follows to the second approach. However, we are aware of a deficiency caused by the second approach, the extracted module might not be able to derive some implicit knowledge linked to both the included and excluded knowledge in the module.

Sabou and colleagues also introduced the following four criteria as general modularization requirements (d'Aquin and Motta 2006b).

- *Modularization criteria reflects selection criteria*: This states that the goal of modularization is to reduce the selected ontology to the relevant sub-parts (or modules).
- *No assumption on the ontology*: The knowledge selection should not require any assumption regarding the ontology representation language (e.g., RDF, DAML, or OWL).
- *Minimal user interaction*: A knowledge selection system cannot expect the user to know well anything concerning the selected ontology or extraction mechanism.
- *Ensuring that output covers both relevant explicit and implicit knowledge*: This encourages the resulting module to include all types of relevant knowledge.

We briefly discuss those criteria for `OntoGenerator`. In our system, the first criterion is mainly considered to assure that the generated ontology reflects the user requirements. We interpret this criterion in terms of relevancy.

[5] `http://swserver.cs.vu.nl/partitioning/`

[6] `http://kmi.open.ac.uk/`

[7] `http://protege.stanford.edu/plugins/prompt/prompt.html`

**Definition 8 (Relevancy)** *Relevancy (RL) is a measure of how much the generated PR ontology reflects the user requirements. RL is simply calculated as the number of classes and properties that all totally matched the user requirements (CPM) divided by the total number of keywords ($T$).*

$$RL = \frac{|CPM|}{|T|}$$

With respect to the second criterion, `OntoGenerator` has a restriction of ontology language. Only OWL ontologies are counted as reusable ontologies in order to simplify the integration process. For the third criterion, user engagement is necessary for the review process, but not necessarily other processes. According to our modularization approach, there is no guarantee to include both explicit and implicit knowledge. Thus, there is a tradeoff between the last criterion and the option of modularization approaches.

## OntoGenerator: A Semi-automatic System for Ontology Generation

By studying the fundamental process and analyzing different approaches from the literature (Bontas and Tolksdorf 2005; Fernandez and Castells 2006; Alani and ShadBolt 2006; Patel and Park 2003; Tartir and Aleman-Meza 2007) we observed a set of commonalities and assembled them into `OntoGenerator`. This is a semi-automatic system that constructs PR ontologies with the assistance of users and ontology tools, through the four serial process modules outlined in Figure1. We present each process module through the step-by-step results of running *OntoGenerator* for an experiment in academic and related domains for a semantic markup process.

### Acquisition

This module focuses on the acquisition of both interested keywords for user requirements and reference ontologies for knowledge sources, to generate a PR ontology. It is composed of three serial steps: (1) keyword extraction, (2) ontology search, and (3) ontology selection.

- Keyword extraction is a process for collecting a set of keywords ($T$) from desired web pages. Simply, $T$ can be obtained via either user selection or use of a tool like `TermExtractor`[8]. $T$ may include keywords given as both nouns (for classes) and verbs (for relations).
- Ontology search is a process for gathering a set of candidate ontologies, $\{O_k\}$, that are all potentially relevant to $T$. There are two possible ways to do this. The first way involves using an ontology search engine such as `Swoogle`[9]. The second way involves using ontology repositories available online or offline. Table 4 describes the list of URLs returned by `Swoogle` for $T_1 \subseteq T = \{University, Student, Publication, Conference, Professor, supervise, advise, Research, PhD\}$. Additionally, we use the ontologies from OAEI2007 conference track[10] as candidates for $T_2 \subseteq T = \{Comittee,$

[8] `http://lcl2.di.uniroma1.it/termextractor/`

[9] `http://swoogle.umbc.edu/`

[10] `http://nb.vse.cz/ svabo/oaei2007/`

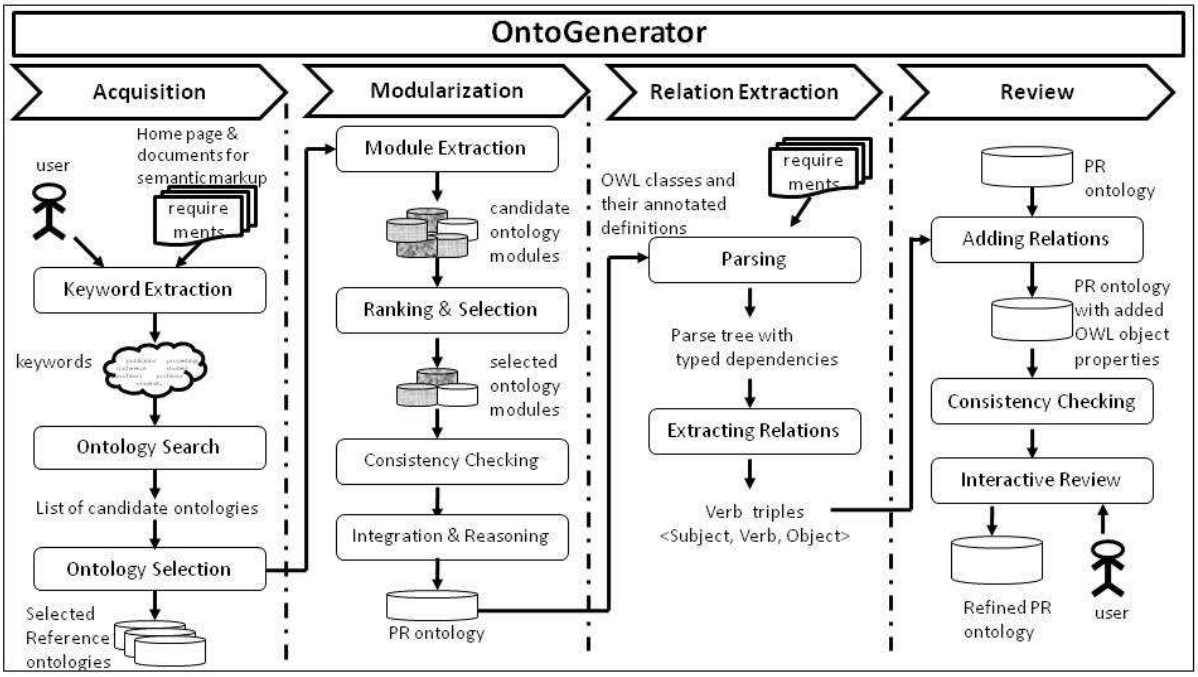


Figure 1: The architecture of OntoGenerator

Table 4: The list of URLs returned by `Swoogle` for $T_1$

| No. | ontology URL |
|---|---|
| 1 | http://protege.stanford.edu/plugins/owl/ owl-library/**koala.owl** |
| 2 | http://annotation.semanticweb.org/iswc/**iswc.owl** |
| 3 | http://www.lri.jur.uva.nl/ rinke/**aargh.owl** |
| 4 | http://swrc.ontoware.org/**swrc-07.owl** |
| 5 | http://swat.cse.lehigh.edu/onto/**univ-bench.owl** |
| 6 | http://protege.stanford.edu/plugins/owl/ owl-library/**ka.owl** |
| .. | |
| 17 | http://www.mondeca.com/owl/moses/**univ.owl** |

$Member, review, Conference, Workshop, publish\}$.

- Ontology selection is a process for choosing a set of reference ontologies $\{O_j\}$ for $T$. We implemented `OLap` for this selection according to CMM given by Definition 4. `OLap` sorts $\{O_k\}$ in the order of maximum CMM value, as described in Table 5. A threshold value $\gamma$ is defined to select $\{O_j\}$, such that any candidate ontology of $\{O_k\}$ whose CMM is greater than $\gamma$ is chosen as a reference ontology $O_j$. In this experiment, we simply use $\gamma = 4.0$. It is also possible to define $\gamma = max(\{CMM(O_k)\})/2$. Note that $j$ and $k$ are the indexes of ontologies.

The result of the acquisition process is a list of reference ontologies, $\{O_j\}$, to reuse for modularization.

Table 5: Reference ontologies resulted by `OLap`

| **Ontology for** $T_1$ | CMM | **Ontology for** $T_2$ | CMM |
|---|---|---|---|
| univ.owl | 12.0 | Conference.owl | 9.2 |
| swrc-07.owl | 7.6 | ekaw.owl | 9.0 |
| unib-bench.owl | 7.0 | edas.owl | 8.6 |
| ka.owl | 5.6 | cmt.owl | 6.0 |
| iswc.owl | 4.4 | sigkdd.owl | 5.4 |
| aargh.owl | 4.0 | iasted.owl | 5.0 |

## Modularization

Since the whole of set of reference ontologies may not need to be reused for specific user requirements, our system applies a modularization approach instead of fully reuse. The process of modularization can be further divided into four sub-processes: (1) module extraction, (2) ranking and selection, (3) consistency checking, and (4) integration and reasoning.

- **Module extraction** is the process of extracting modular ontologies from the reference ontologies $\{O_j\}$. In our system, `OntoModule` is developed in Java as a modularization tool to return a modular ontology $O_{jM_k}$ from each reference ontology $O_j$. This is done with respect to signature $M_k$—which is matched with a set of keywords $T_k \subseteq T$. Each $O_{jM_k}$ consists of a set of axioms that represent a number of classes ($C$)—including only direct super-classes, sub-classes and siblings—together with their properties/roles ($P$), all direct individuals ($a$), and restrictions (e.g., `equivalentClass`, `disjointWith`, `inverseOf`, and so on). Suppose

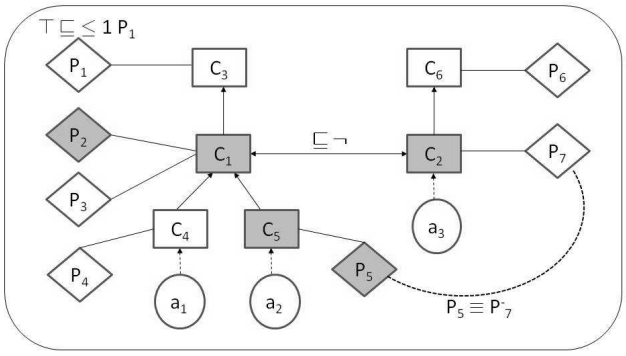


Figure 2: A module extracted for signature $M_k = \{C_1, C_2, C_5, P_2, P_5\}$

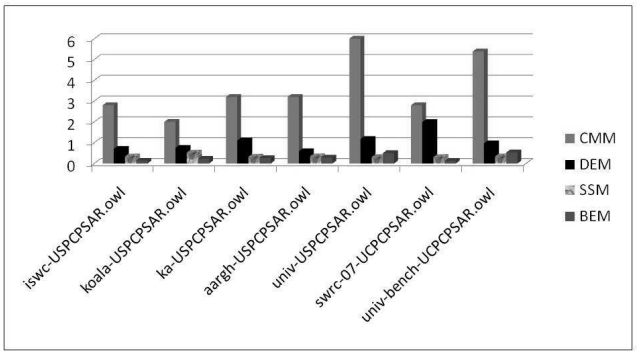


Figure 3: The values of CMM, DEM, SSM and BEM of the modules for $T_1$

$M_k = \{C_1, C_2, C_5, P_2, P_5\}$ is the signature matched with $T_k = \{t_1, t_2, t_3, t_4\}$ such that $\{C_1, C_5\}$ for $t_1$, $P_2$ for $t_2$, $P_5$ for $t_3$, and $C_5$ for $t_4$. Then, `OntoModule` will extract $O_{jM_k}$ as shown in Figure 2. Since relevant modules are extracted from each of the reference ontologies $\{O_j\}$, `OntoModule` returns a list of candidate modules, $\{O_{jM_k}\}$, for $T$.

- **Ranking and selection** is the process of choosing the most relevant modules to create a customized ontology. We developed another Java plugin, `OntoMRank`, for this purpose. Since `OntoModule` considers only direct axioms in its extraction, two shortest-path based measures—SSM and BEM—are not very relevant for `OntoMRank`. According to the experiment result shown in Figure 3, the values of the SSM and BEM are not significant like those of the DEM and CMM. More precisely, the values of SSM and BEM are almost similarly distributed among the modules. Thus, `OntoMRank` focuses only on DEM to choose the most relevant modules from $\{O_{jM_k}\}$. In Table 6, the DEM values ranked among the modules for $T_1$ are listed together with ontological statistics such as the number of classes (denoted by $|C_j|$) and properties (denoted by $|R_j|$). For the DEM, `OntoMRank` employes two sets of weight distributions with respect to the number of classes and properties matched with input $T_M$. If the number of classes is bigger than the number of properties, then the weight distribution of the DEM for $S = \{S_1, S_2, S_3, S_4, S_5, S_6\}$ is $w_{1..6} = \{2.0, 2.0, 2.0, 1.5, 1.5, 1.0\}$, otherwise $w_{1..6} = \{1.5, 1.5, 1.5, 2.25, 2.25, 1.0\}$. Using equal weight distribution, it is arguable not to rank the modules accurately. Suppose that there are two modules: one with two classes and three properties and another with three classes and two properties. Assume that neither module has any constraints. In such a case, the DEM values for both modules from the equal distribution are exactly the same. Therefore, we suggest considering a kind of dynamic weight distribution over the input keywords. For each signature $M_k$, at most one module is selected from $\{O_j M_k\}$. The result of `OntoMRank` is shown in Table 6, where `swrc-07-USPCPSAR.owl` has the highest DEM value among the modules for $T_1$. Similarly for $T_2$, the final module selected from the reference ontologies given in Table 5 is `Conference-CMRCWP.owl`.

Table 6: The DEM value and ontological statistics among the modules extracted for $T_1$

| **Ontology Modules** ($O_{jM_k}$) | DEM | $\|\mathbf{C_j}\|$ | $\|\mathbf{R_j}\|$ |
|---|---|---|---|
| swrc-07-USPCPSAR.owl | 1.99 | 32 | 55 |
| univ-USPCPSAR.owl | 1.179 | 33 | 10 |
| ka-USPCPSAR.owl | 1.1 | 14 | 51 |
| univ-bench-USPCPSAR.owl | 0.963 | 23 | 26 |
| koala-USPCPSAR.owl | 0.75 | 9 | 5 |
| iswc-USPCPSAR.owl | 0.7 | 13 | 29 |
| aargh-USPCPSAR.owl | 0.583 | 37 | 20 |

- **Consistency checking** is a reasoning process for logically verifying the consistency of each of the extracted modules. It is obvious that each module must be certified for its consistency before integration with other modules. This process is critical for ontology generation through the modularization approach. `Racer`[11] is employed for this purpose. We observed from this experiment that `OntoModule` produces modular ontologies in a consistent style without losing any required axiom.
- **Integration & Reasoning** is a process that first creates a PR ontology by integrating selected reusable modules. We use `Protégé OWL API`[12] as an interface model not only to integrate the selected modules but also to generate a PR ontology in OWL. In this experiment, two selected modules for $T$ are integrated as `MyMarkup.owl`[13] which consists of 60 classes and 85 properties in total. Then, the process includes a reasoning task to ensure the consistency of the integrated ontology by `Racer` and `RacerPorter`. Since the cus-

[11] `http://www.sts.tu-harburg.de/`$sim$`r.f.moeller/racer/`

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

[13] `https://www.comp.nus.edu.sg/`$\sim$`tunnn/MyMarkup.owl`

tomized PR ontology is generated by the integration of the selected ontology modules, the process needs to ensure consistency between the PR ontology and its reused modules. For any two axioms $\alpha$ and $\beta$ such that $O_{jM_1} \in \{O_{jM_k}\} \models \alpha$ and $O_{jM_2} \in \{O_{jM_k}\} \models \beta$, any query of $\{\alpha, \beta, \alpha \vee \beta, \alpha \wedge \beta\}$ must satisfy in PR ontology $O_{r_i}$ that reuses $\{O_j M_k\}$.

### Relation Extraction

Though we favor the DEM for the selection of ontology modules, the generated PR ontology might not always be explicit enough for this when the candidate ontologies are developed in simple taxonomies. One possible way to make ontologies explicit is to extract additional non-taxonomic relationships from other implicit forms. This is also called semantic enrichment. In our system, we consider all annotated definitions given via `rdfs:comment` as implicit knowledge source for this relation extraction. This process is accomplished in two serial steps: (1) parsing and (2) extracting relations. A natural language parser and a relation extractor are mainly utilized for this process.

- **Parsing** is the process of expressing a natural language sentence into the form of grammatical tree in order to classify the words in terms of part of speech (e.g., NP, NN, or VP). A parser is a Natural Language Processing (NLP) system for this. Several kinds of parsers exist. While a phrase structure parse represents the nesting of multi-word constituents, a dependency parse represents the dependency between individual words (Buitelaar 2004; Gamallo 2002). A typed dependency parse additionally labels the dependencies with grammatical relations, such as *subject* or *object* (Marneffe and D. Manning 2006). We use the `Stanford Parser`[14], a statistical NLP parser that works out the grammatical structure of sentences into a *parse tree* with *typed dependencies*. Normally, the input of a parser is a sentence or a paragraph; however we mostly use clauses here. The output of the tanford parser includes a parse tree and a list of collapsed word pairs in dependency, for each input clause. Figure4 expresses the input and output of the Stanford Parser for an example annotated definition of class `Student`. (Note that the numbers attached in typed dependencies are automatically designated by the Stanford parser only for word identification.)
- **Extracting relation** is the process for obtaining non-taxonomic relations from the annotated definitions. This process transforms implicit knowledge into a form of explicit knowledge. According to the literature (Kavalec and Svaték 2005; Schutz and Buitelaar 2005), verbs play a critical role for both taxonomic and non-taxonomic relations between subjects and objects. Since `OntoGenerator` supports the generation of the well-defined taxonomy of desired classes, extracting additional taxonomic relations is out of our scope. Our intuitive idea is to extract non-taxonomic relations from the available annotated definitions by embedding the generated PR ontology in NLP techniques. This distinguishes our work from that of others. `R-Extractor` is a Java program that we implemented to extract verb triples ⟨ Subject, Verb, Object ⟩ from the output of the `Stanford Parser`. `R-Extractor`'s purpose is accomplished by the following three steps.
  - Capturing a correct subject from the nouns (and noun phrases), which is mostly comparable to one of the OWL classes already defined in the PR ontology.
  - Capturing verbs and their related objects by employing a set of pattern rules over a set of five critical typed dependencies (i.e., {`nsubj`, `dobj`, `rcmod`, `conj`, `prep`}) from the grammatical relation hierarchy given in (Marneffe and D. Manning 2006).
  - Generating verb triples ⟨ Subject, Verb, Object ⟩.

[14] `http://nlp.stanford.edu/software/lex-parser.html`

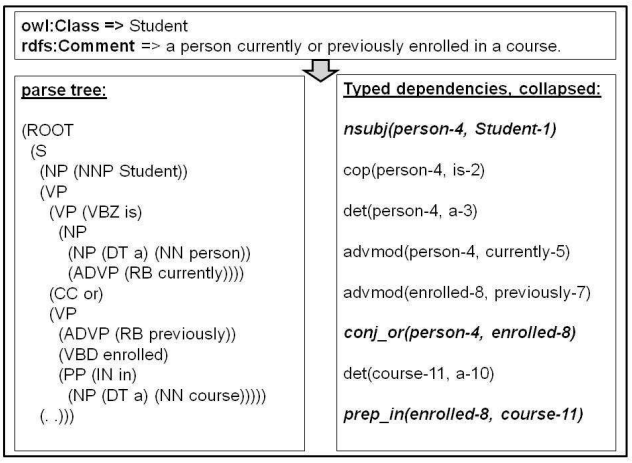


Figure 4: The input and output through Stanford Parser

Table 7: Verb triples extracted by `R-Extractor`

| No. | Verb Triples |
|---|---|
| 1 | ⟨ chair, leads, event ⟩ |
| 2 | ⟨ postdoc, assigned, research ⟩ |
| 3 | ⟨ professor, gives, lectures ⟩ |
| 4 | ⟨ professor, does, research ⟩ |
| 5 | ⟨ professor, supervises, graduate students ⟩ |
| .. | ..... |
| 39 | ⟨ university, offers, courses ⟩ |
| 40 | ⟨ university, grants, degrees ⟩ |
| 41 | ⟨ research group, works, project ⟩ |

By the example described in Figure 4, "enrolled" is a relation between `person` and `course`. From the known taxonomic (is-a) relationship between `person` and `student` (i.e., "student is a person") the output of `R-Extractor` is $\langle student, enrolled, course \rangle$. The result of `R-Extractor` is a list of verb triples as partly described in Table 7. It is obvious that extracting the most relevant verbs through a given ontology may reduce the process of named entity recognition, and thus it is less

complicated while still producing an acceptable result.

We conducted an evaluation of R-Extractor in terms of *precision* and *recall* by comparing the set of verb triples extracted by R-Extractor ($C_r$) with the set of expert-defined triples ($C_e$). $C_{re}$ is calculated for the correctness of $C_r$ in $C_e$ such that $C_{re} = C_r \cap C_e$.

- Precision P = $\frac{|C_{re}|}{|C_r|}$ is the percentage of correctness by R-Extractor.
- Recall R = $\frac{|C_{re}|}{|C_e|}$ is the percentage of the correctness of R-Extractor, comparing with $C_e$.

In the experiment, there are 52 classes with annotated definitions given via rdfs:comment. The number of extracted triples $C_r$ is 41 whereas $C_e$ is 60 since more than one triple can be extracted from a single sentence. For example, "*A professor is a faculty member who not only gives lectures but also does research by supervising graduate students.*". In this experiment, the precision and recall of R-Extractor are 92.69% and 63.33% respectively. We note that the limitted number of dependency patterns used in R-Extractor might result a low recall value.

Among the existing ontology learning systems, the work of RelExt (Schutz and Buitelaar 2005)[15] is similar to our approach. However, RelExt explores relation triples from a corpus given in the German language utilizing the SCHUG system (Declerck 2001). Even though the tools and support for reuse and learning are increasing, difficulties still exist. The most challenging point from this experiment is relation evaluation. More precisely, there may be overlap in name or semantics both among the extracted verb triples and between those and the existing OWL object properties.

### Review

In this process module, we utilize three sub-processes: (1) adding relations, (2) consistency checking and (3) interactive review of the whole PR ontology generated.

- **Adding relation** is a process that transforms the extracted verb triples into OWL object properties as specified in the following.

  ⟨owl:ObjectProperty df:ID="enrolled"⟩

  ⟨rdfs:domain rdf:resource="#Student"/⟩

  ⟨rdfs:range rdf:resource="#Course"/⟩

  ⟨/owl:ObjectProperty⟩

  We employ OLap to resolve name overlapping among the existing and new relations. Alternatively, it can deal with this using WordNet[16].

- **Consistency checking** is done by Racer again for the relation-added PR ontology. Additional checking can also be performed based on a combined reasoning approach (Dong et al. 2004).

- **Interactive review** is a process that evaluates the generated PR ontology with its added relations in an interactive style. User involvement in this review process is necessary for two reasons: (1) the existence of semantic overlap among the names of relations and (2) the need to systematically structure the indirectly extracted classes (i.e., those that are not directly but necessarily extracted together with the non-taxonomic relations) into the taxonomy instead of easily assigning them as the subclasses of owl:Thing. We relaize that curent techniques are not adequate to resolve these problems automatically. Even though a PR ontology is carefully generated through many well-developed tools and checked with Racer, it is better for users to check it by practicing semantic indexing or using it in their target information systems. During this review process, the relevancy evaluation is considered to fulfill the aim of OntoGenerator. It assists users in developing a customized OWL ontology through partial reuse. By Definition 8, the relevancy of the generated ontology is calculated. In the given case study, the computed RL value of MyMarkup.owl is 0.94. This value proves the feasibility of OntGenerator.

## Related Work

Our idea of OntoGenerator is differentiated from other ontology generation systems (Stuckenschmidt and Klein 2003; Alani and ShadBolt 2006; Maedche and Staab 2001) by a modularization technique and a ranking strategy that rely on explicitness. Since OntoGenerator shares two measures—the CMM and DEM—with AKTiveRank (Alani and ShadBolt 2006), we mainly compare AKTiveRank and OntoMRank—the ranking tool of OntoGenerator. AKTiveRank selects a relevant (full) ontology $O_k$ based on a total score aggregated from the CMM, DEM, SSM and BEM by taking equal weights among the measures. In addition to the equal weight distribution difference between AKTiveRank and OntoMRank, another difference regarding the method of computing the DEM exists. AKTiveRank computes the DEM on $S = \{S_1, S_2, S_3, S_4\}$ denoted for the number of sub-classes, super-classes, siblings, and relations whereas OntoMRank calculates the DEM using additional two measures (attributes and constraints) as given in Definition 5.

**Definition 9 (Total AKTiveRank Score)** *Let $\{O_k\}$ be the set of candidate ontologies to rank for ontology selection, $w_i$ be a weight factor and $M = \{M[1], M[2], M[3], M[4]\} = \{CMM, DEM, SSM, BEM\}$ be four ranking measures.*

$$Score(O_k \in \{O_k\}) = \sum_{i=1}^{4} w_i \frac{M[i]}{max_{1 \le j \le |\{O_k\}|} M[j]}$$

Values for each measure are normalized to be in the range (0-1) by dividing them by the maximum measure value of all candidate ontologies. The line graph shown in Figure 5 expresses the result of AKTiveRank returned for five different weight distributions among four measures. In all types of weight distribution, Koala.owl attained the highest total score. In the case of OntoMRank,

[15] RelExt is developed for SmartWeb project in football domain, particularly for FIFA World Cup in Germany, 2006.

[16] http://wordnet.princeton.edu/

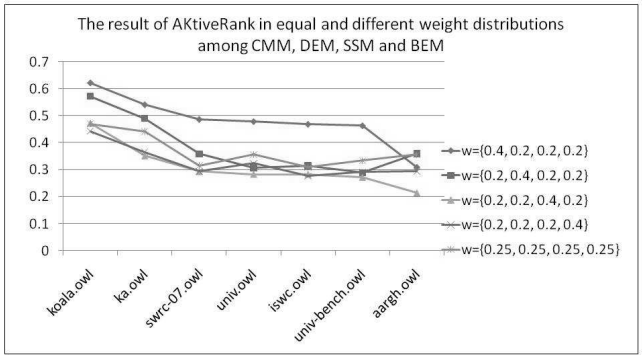


Figure 5: The result of reusable ontologies for $T_1$ by AKTiveRank

swrc-07-USPCPSAR.owl was the most relevant ontology module with the highest DEM value. By Definition 8, the relevancy of swrc-07-USPCPSAR.owl is double that of Koala.owl.

Because the results of the two ontology ranking tools differ, we evaluate them in terms of quality. For this purpose, we apply the schema measure of OntoQA (Tartir and Aleman-Meza 2007). This measure consists of the following three kinds of richness.

- Relationship Richness (RR) measures the explicitness of an ontology's relations. An ontology that contains many relations is richer than a taxonomy with only class-subclass relationships. Formally, the relationship richness (RR) of an ontology is defined as the ratio of the number of relationships (R) to the sum of the number of subclasses (SC) plus the number of relationships.

$$RR = \frac{|R|}{|SC| + |R|}$$

- Attribute Richness (AR) takes into account the number of attributes of each class that can indicate both the quality of ontology design and the amount of pertinent information. Formally, AR is computed as the number attributes for all classes (att) divided by the number of classes (C).

$$AR = \frac{|att|}{|C|}$$

- Inheritance Richness (IR) measures how well knowledge is grouped into different (sub-)categories in the ontology. Formally, IR is defined as the average number of subclasses $(C_I)$ per class $(C_i)$, $|H^c(C_I, C_i)|$.

$$IR = \frac{\sum_{C_i \in C} |H^C(C_I, C_i)|}{|C|}$$

It is assumed that an ontology with higher richness values is of better quality. Since we focus on partial reuse and modularization, koala.owl as a whole is not comparable to swrc-07-USPCPSAR.owl in terms of RR, AR, and IR. Therefore, we extract a module from koala.owl titled Koala-USPCPSAR.owl that hits for $T_1$. Table 8 describes the richness values between the two ontologies resulting from OntoMRank and AKTiveRank.

Table 8: The quality of ontologies in terms of OntoQA's schema measure

| **Ontology Modules** | RR | AR | IR |
|---|---|---|---|
| swrc-07-USPCPSAR.owl | 0.603448 | 0.625 | 1.25 |
| koala-USPCPSAR.owl | 0.5 | 0.11 | 0.66 |
| univ-USPCPSAR.owl | 0 | 0.303 | 1.15 |

Note that $\{|R| = 35, |SC| = 23, |C| = 32, |att| = 20, \sum_{C_i \in C} |H^C(C_I, C_i)| = 35\}$ and $\{|R| = 4, |SC| = 4, |C| = 9, |att| = 1, \sum_{C_i \in C} |H^C(C_I, C_i)| = 6\}$ are the statistics of swrc-07-UCPCPSAR.owl and koala-USPCPSAR.owl respectively. Moreover, the CMM value of swrc-07-UCPCPSAR.owl is 2.8 whereas koala.owl and koala.USPCPSAR.owl remain 2.0 against $T_1$.

In addition, as described in Figure 3 univ-USPCPSAR.owl has the highest CMM value. However, its quality in the schema measure is lower than that of swrc-07-UCPCPSAR.owl. In former experiments with AKTiveRank (Alani and ShadBolt 2006), ontology was selected for either single or double keywords. In such cases, the values of the SSM and BEM may be valuable for the selection. According to the result shown in Table 8, we suggest employing only the DEM with respect to explicitness as the quality of ontology generated through OntoGenerator.

## Conclusion

This paper presented an intuitive approach of semi-automatic ontology generation. *OntoGenerator* is developed using a number of semantic web technologies and Java tools in a series of combined processes including modularization, ranking, and ontology-based relation extraction. The innovative part of OntoGenerator involves including all of the processes in a complete pipeline to automate the flow of ontology creation through partial reuse. It is obvious that we can avoid knowledge acquisition problems to a certain extent by relying on existing well-defined ontologies, therefore we can bootstrap the process of ontology building. The Java tools implemented for OntoGenerator system, which include OLap, OntoModule, OntoMRank, and R-Extractor, are available at http://www.comp.nus.edu.sg/∼tunnn/OntoGenerator/. According to our experiments, we contend that the prototype of OntoGenerator is promising for the process of semantic indexing.

We need to evaluate further the accuracy of R-Extractor by improving it with additional dependency patterns as well as by comparing it with other existing systems. OntoGenerator will be improved toward a more automatic system. In future experiments, we plan to test OntoGenerator in a well-defined and complex domain like medicine or bio-informatics. In addition, we would like to discuss both our experience of integrating ontology modules and the amount of user effort needed in the review process in a more concrete style.

**Acknowledgments**
This research work is supported by the grants of the projects, 'Advanced Ontological Rules Language and Tools [R-252-000-249-422]' and 'Systematic Design Methods and Tools for Developing Location Aware, Mobile and Pervasive Computing Systems [R-252-000-329-279]'. The authors thank the National University of Singapore for its support of promoting research and development in advanced computing.

## References

Alani, H., and Brewster, C. 2006. Matrics for ranking ontologies. In *15th International World Wide Web Conference, WWW2006*.

Alani, H., B. C., and ShadBolt, N. 2006. Ranking ontologies with aktiverank. In *5th International Semantic Web Conference, ISWC2006*.

Baader, F., C. D. M. D. L., and Nardi, D. 2003. *Description Logic Handbook: Theory, Implementations, and Applications*. Cambridge University Press.

Baader, F., H. I., and Sattler, U. 2007. Description logics. *Handbook of KnowledgeRepresentation*.

Bontas, E., M. M., and Tolksdorf, R. 2005. Case studies on ontology reuse. In *I-Know05*.

Brewster, C., C. F., and Wilks, Y. 2003. Background and foreground knowledge in dynamic ontology construction. In *Special Interest Group on Information Retrieval, SIGIR'03*.

Buitelaar, P., E. T., and Declerck, T. 2004. Ontoselect: A dynamic ontology library with support for ontology selection. In *the Demo Session at the International Semantic Web Conference, ISWC04*.

Buitelaar, P., O. D. S. M. 2004. A protégé plug-in for ontology extraction from text based on linguistic analysis. In *European Semantic Web Conference, ESWC04*.

d'Aquin, M., S. M., and Motta, E. 2006a. Modularization: a key for the dynamic selection of relevant knowledge components. In *First International Workshop on Modular Ontologies, WOMO 2006*.

ḋAquin, M., S. M., and Motta, E. 2006b. Modularization: a key for the dynamic selection of relevant knowledge components. In *5th International Semantic Web Conference, ISWC 2006*.

d'Aquin, M., S. A. S. H., and Sabou, M. 2007. Ontology modularization for knowledge selection: Experiments and evaluations. In *18th International Conference on Database and Expert Systems Applications-DEXA07*.

Declerck, T. 2001. A set of tools for integrating linguistic and non-linguistic information. In *CEUR-WS*.

Ding, L., P. R. F. T. J. A. P. Y., and Kolari, P. 2005. Finding and ranking knowledge on the semantic web. In *4th International Semantic Web Conference, ISWC2005*.

Ding, L., F. T. J. A. P. R. C. S. R. P. Y. R. P. D. V., and Sachs, J. 2004. Swoogle: a search and metadata engine for the semantic web. In *IKM04*, 652–659.

Dong, J. S.; Lee, C. H.; Li, Y. F.; and Wang, H. 2004. A Combined Approach to Checking Web Ontologies. In *Proceedings of 13th World Wide Web Conference (WWW'04)*, 714–722.

Fernandez, M., C. I., and Castells, P. 2006. Core: A tool for collaborative ontology reuse and evaluation. In *EON06*.

Gamallo, P., G. M. A. A. L. G. d. L. V. 2002. Mapping syntactic dependencies onto semantic relations. In *ECAI Workshop on Machine Learning and Natural Language Processing for Ontology Engineering*.

Garu, B. C., H. I. K. Y., and Sattler, U. 2007. Just the right amount: Extracting modules from ontologies. In *16th International World Wide Web Conference, WWW2007*.

Grau, B.C., P. B. S. E., and Kalyanpur, A. 2005. Automatic partitioning of owl ontologies using e-connections. In *DL05*.

Kavalec, M., and Svaték, V. 2005. A study on automated relation labelling in ontology learning. *Ontology Learning from Text: Methods, Evaluation and Applications*.

Maedche, A., and Staab, S. 2001. Ontology learning from the semantic web. *IEEE Intelligent Systems* 72–79.

Marneffe, M., M. B., and D. Manning, C. 2006. Generating typed dependency parses from phrase structure parses. In *5th Language Resources and Evaluation Conference, LREC 2006*.

Patel, C., S. K. L. Y., and Park, E. K. 2003. Ontokhoj: A semantic web portal for ontology searching, ranking and classification. In *5th ACM International Workshop on Web Information and Data Management, WIDM03*.

Rada, R., M. H. B. E., and Blettner, M. 1989. Development and application of a metric on semantic nets. *IEEE Trans. on Systems Management and Cybernetics* 19.

Schutz, A., and Buitelaar, P. 2005. Relext: A tool for relation extraction from text in ontology extension. In *2nd European Semantic Web Conference, ESWC 2005*.

Seidenberg, J., and Rector, A. 2006. Web ontology segmentation: Analysis, classification and use. In *15th International World Wide Web Conference, WWW2006*.

Sleeman, D., P. S. R. D., and Schorlemmer, M. 2003. Ontology extraction for distributed environments. *Knowledge Transformation for the Semantic Web* 80–91.

Spefano, S. 2005. Report on modularization of ontologies. *Web Deliverable 2.1.3.1*.

Stuckenschmidt, H., and Klein, M. 2003. Structure-based partitioning of large concept hierarchies.

Tartir, S., A. I. B. M. M. S. A. P., and Aleman-Meza, B. 2007. Ontoqa: Metric-based ontology quality analysis. In *the First IEEE International Conference on Semantic Computing-ICSC07*.

Weinstein, P. C., and Birmingham, W. P. 1999. Comparing concepts in differentiated ontologies. In *Knowledge Acquisition Workshops, KAW99*.