Automatically Integrating Heterogeneous Ontologies from Structured Web Pages

Shiren Ye, National University of Singapore, Singapore

Tat-Seng Chua, National University of Singapore, Singapore

ABSTRACT

Max 150 words

Keywords: please provide

INTRODUCTION

Ontology integration is one of the most important research topics in knowledge integration and knowledge reconciliation. It has been widely applied in knowledge engineering, database and data warehouse, e-commerce as well as semantic Web (Dill, Eiron, Gibson, Gruhl & Guha, 2003; Maedche, 2002). As more and more organizations in industry, academia and government publish their information (such as services and products) on the Web, how to add semantic annotation for machine-access and thus enhance the interoperability among these autonomous, distributed, and semantically heterogeneous information sources becomes more urgent.

Usually, it is feasible to utilize weighted string-matching method to handle the explicit

evidences exhibited by the co-occurred terms in the ontology elements. The largest obstacles to ontology integration come from the implicit evidences hidden in various representations. As illustrated in Figure 1, the implicit evidences between two diverse ontologies could include: (1) Appearance: besides the various formats used to encode the content, such as RDF/OWL, XML DTD and XML schema, a piece of knowledge can be described using different types of descriptions (e.g., enumeration or property restriction). (2) Structure: the involved elements may be organized using diverse structures (such as plain or nested). (3) Terminology: diverse vocabularies may refer to the identical thing or have similar meaning (such as *CPU* and *Processor*). (4) Granularity: content may be expressed by concise version using a simple element vs. complex version using a set of combined elements (such as *Hard Disk* vs. *HD* and its offspring). These implicit evidences would thus complicate the problems of merging these analogous ontologies.

In order to achieve the common ontology that depicts the universal namespace of analogous ontologies (relevant to the same kind of objects), we develop a pipeline as shown in Figure 2 to understand the data-intensive objects published on the Web and resolve the interoperability among the data from the heterogeneous systems. The pipeline consists of three steps as follows.

1. **Web→Data**: As Web designers usually use a template or dynamic program to publish all analogous objects (e.g., products and services), all Web pages about these objects appear to share the same presentation style and description framework. By identifying the key difference between the irrelevant components and objects in HTML DOM trees, we first detect object region, which contains only the description of objects. From the object region, we then extract information about the individual product, which we called object data under the interpretation of view syntax model. Details of this step are reported in Ye and Chua (2004b).
2. **Data→Model**: From a set of analogous objects, we induce an object model (ontology or schema) using the procedure outlined by Ye and Chua (2004a, 2006). This model contains the condensed syntax and framework information of objects, which is partially equivalent to the schema of underlying data repository. For instance, *notebook ontologies* consist of two object

Figure 1. Two models about notebook (laptop)

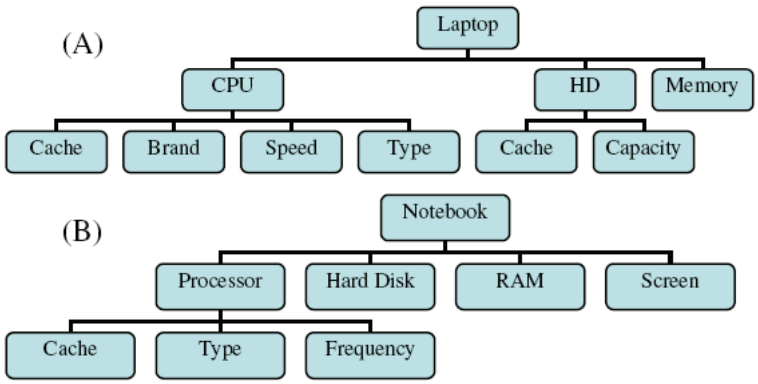
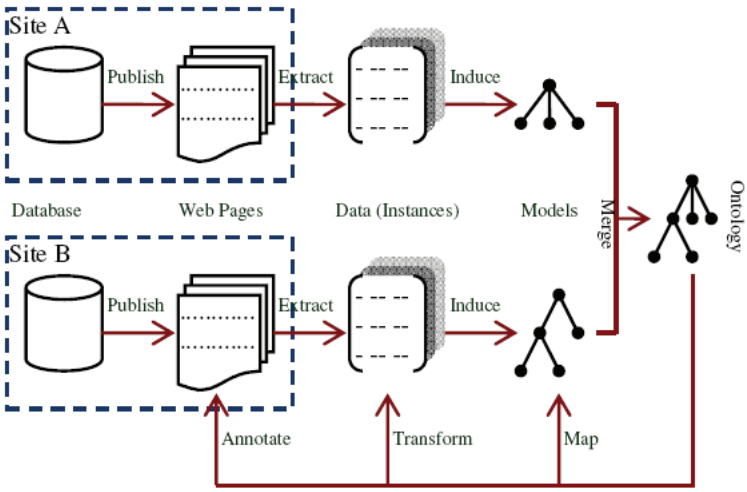


Figure 2. The pipeline of ontology building for web-based application



models that rightly match object data from the corresponding Websites.

3. **Model(s)→Ontology**: The object models about the identical thing but from diverse sites are merged into a universal ontology. It resolves the interoperability among the heterogeneous models, especially semantic variations, to ensure that the original data are converted to a common space. The ontology shown in Figure 5 is the integration output of *Notebook ontologies*.

As manually integrating ontologies is a tedious, time-consuming, error-prone, and expensive process, an automated approach to integrating these diverse ontologies is indispensable, especially when we need to handle huge scale data on the Web. The variations among these ontologies are so large that we need to collect all possible implicit and explicit hints from ontology and instance, semantic and structure, as well as linguistic and syntactic features. A suitable similarity paradigm for element matching is the core of evaluating the degree of matching in the involved elements (Rahm & Bernstein, 2001).

At the same time, representing the objects from multiple namespaces with a common specification is crucial in Web computation. This article follows our previous work (Ye & Chua, 2006) and concentrate on the last phase of model(s) → ontology transformation; namely, the procedure of building a common specification that can cover analogous entities extracted from heterogeneous structured pages published by various sites. The common ontology can be used as key to annotating original Web pages and transforming the original data into a common semantic space. Our research attempts to exploit the automated approaches to making structured web pages machine-accessible.

To achieve our objective, we need to devise a suitable similarity paradigm for element matching. We thus define a unified similarity paradigm that can naturally combine evidences that support matching. The similarity is derived from three aspects of elements: intension, extension and context, denoted by `<INT, EXT, CXT>`, where `INT` and `EXT` include corresponding weighted contents from offspring, and `CXT` is relevant to the key contents existing in the path between the current element and its root. Here the similarity coefficient of terms in each

aspect is calculated based on their semantic overlapping and syntax comparability. Next, we develop a top-down matcher algorithm based on search space analysis and similarity reuse, which leads to less error-prone matching and lower computational cost. We also propose an ontology alignment and construction method to generate the common ontology we need.

BACKGROUND

The existing hand-crafted ontologies usually cannot match the live data from diverse sources on the Web well. For example, *Kodak* (http://www.kodak.com) describes a specification of some kinds of its digital cameras in terms of a hierarchical structure with more than 100 elements. To the best of our knowledge, there is no such an external hand-crafted ontology that can fully interpret the data. It is thus indispensable to investigate how to extract models and integrate ontologies on the Web. In general, these machine-extracted models are largely, or at least partially, similar to hand-crafted schema in database, DTD/XML schema in XML, as well as ontologies in knowledge engineering and Semantic Web.

The main difference between our machine-extracted ontologies and the hand-crafted ontologies lies in: (1) The contents of the former may be local and incomplete rather than global since they are derived from only one kind of similar pages. (2) Logical items such as identifiers and references are excluded (or hidden in links) as derived from user-oriented pages. (3) Fewer abbreviation, punctuation and acronym are used. Compared to schema integration of Web interface of search engines (He, Meng, Yu & Wu, 2003), the ontologies extracted from the Web that we need to handle are more complicated since they have hierarchical structures and various contents. However, building a common ontology from a set of heterogeneous object models (site-specific ontologies) can be considered as a kind of schema matching and ontology integration issues. Although there might be some differences in the investigation background, the principle of our approaches can be applied to the general problems in schema matching and ontology integration.

Ontology integration is a pervasive, important and difficult problem in ontology learning (Kalfoglou & Schorlemmer, 2003; Maedche, 2002). In recent years, I3CON (Information Interpretation and Integration Conference, http://www.atl.external.lmco.com/projects/ontology/i3con.html) and EON Workshop (Evaluation of Ontology-based Tools, http://km.aifb.uni-karlsruhe.de/ws/eon2004) paid much attention to ontology integration (Euzenat, 2004). Kietz, Volz, and Maedche, (2000) attempted to semi-automatically learn ontologies such as concepts and relations from a Corporate Intranet. His output is a terminology category. Halkidi and Nugyen (2003) concentrated on acquiring ontology from a Web directory, where a master Web taxonomy is integrated from source taxonomies. Their ontologies are derived from resource pages that have previously been organized manually by others, rather than from individual instances or content pages. Noy and Musen (2003) used PROMPT to provide a semi-automatic approach to ontology merging and alignment. Bao and Honavar (2004) developed Wiki@nt that supports the integration and reconciliation of semantically heterogeneous and very likely inconsistent ontology modules. However, to the best of our knowledge, the principled approach for manipulating independently developed, semantically heterogeneous ontologies or for reconciling logical inconsistencies between them is still not available.

Another branch of related work is schema matching. A comprehensive survey of automatic matching distinguishes between schema-level and instance-level, element-level and structure-level, as well as language-based and constraint-based matchers (Doan & Halevy, 2005; Rahm & Bernstein, 2001; Shvaiko & Euzenat, 2005). Most existing approaches employed feature-heuristic combinations, such as Dice-coefficient, edit-distance, as well as sub-elements and parent-elements to measure the similarity among elements. For example, Lee and Yang

(2002) developed a matcher based on semantic, immediate descendants and leaf-context to measure similarity of DTD elements. The main disadvantages of most typical approaches include: (1) the matching techniques are too heuristic with too many thresholds and parameters (most more than 10) to handle the various data. It is difficult to control and optimize them; (2) most approaches concentrate on one aspect of problem while ignoring other aspects, such as focusing only on schema but not instance, element-level but not structure, and so forth; (3) it is difficult to synthesize these existing approaches to overcome their shortcomings as they use different strategies; (4) due to the poor generalizing ability, most learned models hardly achieve satisfactory performance in test data in which the evidence distribution differs from that of training data. In a way, we meet an interesting phenomenon: many investigators declared that their proposed approaches already obtained good performance, such as F_1>90%, in their own corpus, while the users from industry still cannot find a reliable and flexible solution to ontology integration.

The implicit and explicit evidences that support ontology integration are distributed in the semantic or syntactic features in the ontologies, instances and even other external knowledge resources. In some cases, these evidences are redundant for decision-making. Most of the typical ontology merging approaches, which collect only a part of evidences but neglect the others, can achieve good performance in their test corpus. However, it is difficult to extend these approaches into the cases where evidence distribution changes (Doan Madhavan, Domingos & Halevy, 2002). Motivated by this issue, our research proposes a unified similarity paradigm that requires fewer parameters to handle *all* available implicit and explicit evidences among schema and instance, element and structure, as well as linguistics and other resources.

ONTOLOGY MATCHING

We need to account for all implicit and explicit evidences into a similarity value that measures the closeness between two ontology elements. Naive string-matching approach can well deal with the explicit evidences expressed by term co-occurrences. For implicit evidences hidden in `INT`, `EXT` and `CXT`, we need to investigate the semantic and syntactic features of `INT`, `EXT` and `CXT` so that the implicit coherences can be detected despite the diverse appearance, structure, terminology, and granularity used. In this section, we present a framework to combine these implicit and explicit evidences for merging.

Concept Representation

Ontologies provide the formal specification of conceptualizations and can be considered as the definition of a set of concepts (Maedche, 2002). The corresponding instances can also supply certain detailed contents when we explore the definition, which will become more essential when few evidences exist in the specification. Hence, we explore the underlying concepts for the elements involved in the following three aspects:

- **Intention/`INT` (meaning):** The contents of intension may be described in terms of element name or description such as label, annotation, restriction, and attribute. It is relevant to the meaning or characteristics often expressed by a definition. We thus need to gather all of the items related to intension together for evidence comparison.
- **Extension/`EXT`:** It includes the set of objects that the concept refers to and usually lists the possible value range, unit of measure (UOM), as well as cardinality, enumerations and type from object data. Along with what has been listed in an ontology, `EXT` can be built by scanning the corresponding instances. If the size of instances is too large, sampling instances will be used to build `EXT`.
- **Context/`CXT`:** It comes from the surrounding elements of the current element, especially, the path to the root of an ontology that indicates the location of the current element. It is a crucial factor

to distinguish the homonymous elements (such as *Laptop.HD.Cache* vs. *Notebook.Processor.Cache*) in different context. Here, the list of element names presenting in such a path can reflect the ontology structure and thus it can be considered as the indicator of the corresponding context.

For example, the INT, EXT and CXT of some elements in *notebook ontologies* are tabulated in Table 1.

Although Cupid (Madhavan, Bernstein & Rahm, 2001) also utilized names, data types, constraints and schema structure, and so forth, one distinguished difference is that our approach emphasizes how to organize them. Namely, with the structure of (INT, EXT and CXT), it is easier for users to tune the parameters (e.g., increasing the weights of extension when more evidences for matching can be found in instances), as the parameter tuning is understandable and manageable. The domain-independent parameters, such as the thresholds for semantic similarity calculation, are encapsulated in three sub-models (INT, EXT and CXT) respectively. This makes it convenient to tune these parameters. In general, users can ignore these details.

Concept Mapping

A mapping assertion involves two ontology elements, whose mapping type specifies how the pair of elements is related. Typical mapping types include equality, overlapping, and more general/specific. Like most other research efforts, our investigation concentrates only on overwhelming equality.

In order to handle the heterogeneity caused by various granularity, terminology, and structure, we need to explore the similarity coefficient by virtue of these two aspects:

- **Syntax:** It deals with the patterns and their relations in the descriptions. Here, we focus on three types of syntactic patterns: (1) expression template in elements (i.e., amount + UOM) relevant to the data type and value space; (2) acronym (i.e., *HD* vs. *Hard Disk*); and (3) soundex (i.e., *Ship2 vs. ShipTo)*.
- **Semantic:** It is about the meaning of (parts of) words, phrases, sentences, and texts. The semantic is mainly used to eliminate the terminological difference caused by the variations and synonyms (i.e., *CPU* vs. *Processor*) in the corresponding descriptions.

Table 1. Concept representation for notebook ontologies

No	Element & Intension	Extension	Context
1	Laptop	Notebook, Laptop, Sony, Toshiba, HP…	Root
2	Laptop.CPU		Laptop
3	Laptop.CPU.Cache	##(M/k)	Laptop.CPU
4	Laptop.CPU.Type	PIII/PIV/P3/P4.Centrino	Laptop.CPU
5	Laptop.CPU.Speed	##.#(G/M/GHz/Mhz)	Laptop.CPU
6	Notebook	Sony***	Root
7	Notebook.Processor		Notebook
8	Notebook.Processor.Caxhe	##(M/k)	Notebook.Processor
9	Notebook.Processor.Ypte	PIII/PIV/P3/P4/Centrino	Notebook.Processor
10	Notebook.Processor.Frequency	##.#(G/M/GHz/MHz)	Notebook.Processor

Feature Construction and Concept Similarity

We use three bags of terms coming from intension, extension and context to represent element e, namely, e = <INT, EXT, CTX>. Here, the bags of INT and EXT consist of a set of weighted terms d=<t_1,...,t_n> appearing in the current element and its offspring elements; the bag of CTX consists of terms in element names from the root to the current element.

Suppose that n_i is the number of elements containing term t_i, N is the number of elements in the bag, and F_i is the frequency of term t_i in the current element. The weight of i in such a bag is w_i=f_i*log(N/n_i) if we follow the TFIDF principle as well.

In addition, we also need to normalize the contents in EXT that come from ontology instances. Let D_i be the number of object instances. In order to alleviate the impact of instance size, then the above formula is modified as w_i=f_i*log(N/n_i)/D_i. Here, we need to emphasize that the bags of INT and EXT include the terms appearing in the offspring.

Considering the granularity variation in the samples shown in *N*otebook Ontologies in Figure 1, where the combined element *Laptop.HD* and its sub-elements *Cache* and *Capacity* in Figure 1(A) are equivalent to a single element *Notebook.HardDisk* in Figure 1(B). We find that the contents in a simple element will be scattered in whole the matched subtree rather than the root only. To overcome this problem, offspring must be taken into account to provide comprehensive evidences for matching. However, the offspring elements that are far away might introduce noise in matching, as they are not quite relevant to the current element. Thus we also need to utilize the following weight modifier to penalize the deeper subtree.

$$w_i = \lambda^{size(subtree)} * f_i * \log(N/n_i)/D_i \quad (1)$$

where $\lambda(0<\lambda<1)$ is the depth-penalized factor.

We use cosine similarity to calculate the similarity between two bags of terms t and s, where $t = (t_q,...,t_m)$, $s = (s_1,...,s_n)$. The similarity of t and s is

$$s(t,s) = \frac{\sum_i^m w_s^i sim(t_i,s) \cdot \sum_j^n w_t^j sim(s_j,t)}{\sum_i^m w_s^i \cdot \sum_j^n w_t^j}. \quad (2)$$

For two element e_1 and e_2, the concept similarity is the weighted sum of their INT, EXT and CXT.

$$Sim(e_1,e_2) = \alpha \cdot s_{INT} + \beta \cdot s_{EXT} + \gamma \cdot s_{CXT}, \quad (3)$$

where $\alpha + \beta + \gamma = 1$.

Here, we briefly summarize the necessity and rationality of segmenting all evidences into three groups such as INT, EXT, and CXT. (1) CXT cannot be ignored since it is pivotal to eliminate the ambiguity between homonymous elements (such as *Laptop.HD.Cache* vs. *Laptop.CPU.Cache*); (2) CXT cannot be mixed into INT and EXT; otherwise, it tends to incur error pairs between elements and the parents of their properly matched elements in other ontologis, such as *Laptop.CPU.Type* and *Notebook.Processor;* (3) INT and EXT, like two sides of a page, do not need to be compared directly since they are used to describe the specification of a concept from two different angles; (4) usually, the contents in ontologies and models are limited since the Web designers are often not willing to give out too many details of their data specification and criterion. Thus, the available information merely consists of element name in INT and the concrete descriptions are mostly missing. In these cases, the instance-related information which is abundant on the Web and encapsulated in EXT will have large contributions to decision-making for matching; (5) it is convenient for users to observe the evidences available at INT, EXT and CXT, and they can thus set up proper α, β and γ to weight the corresponding group(s). At the same time, the application-independent parameters (e.g., these existing in Eqn. 1 and 2) will be encapsulated in INT, EXT, and CXT, it is usually unnecessary for end

users to manipulate them directly. In general, these points are also the principles of similarity paradigm for ontology matching that we need to follow, and our proposed model is designed to satisfy most of these principles.

Semantic Similarity

Usually, noun phrases are the major components in `INT`, `EXT`, and `CXT`. We can identify the corresponding concepts using semantic relatedness and disambiguate them via the external language resource such as the WordNet (Miller, 1995). At the same time, since some domain-dependent terminologies may not appear in WordNet, we can introduce external domain knowledge bases (e.g., medical dictionary) to determine their semantic. In domain-independent resource WordNet, the semantic relatedness is measured by the ratio of term co-occurrence among the relations including: (1) gross (the definition of a concept with possibly specific examples); (2) hypernymy (is-a relation); and (3) meronymy (has-a relation) (Baziz, Boughanem & Aussenac-Gilles, 2004).

Based on Lesk Hypothesis (1986), word senses that are related to each other are often defined in a dictionary using many of the same words. Supposed that there are two terms t_i and s_j (mono and multiword in `INT`, `EXT` or `CXT`), $t_i \in t = (t_1,...,t_m)$, $s_j \in s = (s_1,...,s_n)$ with senses $SS_1^i, SS_2^i, ..., SS_m^i$ and $SS_1^j, SS_2^j, ..., SS_m^j$ as defined by WordNet. The semantic relatedness (overlaps) will retain in such pair as x and y ($1 \le x \le m, 1pt 1 \le y \le n$) that have maximal adapted Lesk intersection. We thus calculate the number of common words, which is squared in the case of successive words, denoted by *overlap*(*,*) (Halkidi et al., 2003). Hence, the semantic similarity among t_i and j is

$$semantic(t_i, s_j) = \underset{1 \le x \le m, 1 \le y \le n}{Max}\ overlap(SS_x^i, SS_y^j). \quad (4)$$

Furthermore, the similarity between t_i and s is supposed to be the similarity between t_i and the term in s with the maximal similarity, namely,

$$semantic(t_i, s_j) = \underset{1 \le y \le n}{Max} semantic(t_i, s_j). \quad (5)$$

For example, in the mentioned models, both *CPU* and *Processor* have one sense whose definition is *(computer science) the part of a computer (a microprocessor chip) that does most of the data processing; .....*, and hyponymy is *electronic equipment | hardware, computer hardware*. It indicates that their underlying concepts are largely overlapping and thus they have identical semantics.

Incidentally, we cannot ignore the risk of returning spurious semantic equivalent pairs. This is because semantic similarity is based on the word sense with maximal overlapping defined in WordNet, and will lead to spurious evidences for matching. Usually, these spurious evidences are scattered sparsely in the matchings of `INT`, `EXT` and `CXT`. However, since the prediction of element matching depends on whole set of evidences rather than a single one, they can be shielded in our approach.

Matching Algorithm

Matching Search Space

Most available approaches for schema matching and ontology integration attempt to select all non-overlapping pairs with maximal similarity as output. Most existing approaches ignore the relative location of existing pairs although some ad-hoc heuristics may be used to prune some existing pairs. Actually, the matching distribution in ancestor elements usually constrains the matching space of offspring elements. As a case in point, consider the procedure of finding the matching element of *Notebook.Processor.Cache* in the given *Notebook ontologies*. Actually, we can test the elements in branch *Laptop.CPU* rather than the entire tree. This is due to the fact that the matching pairs we have already obtained, such as *Laptop.CPUNotebook.Processor*, can exclude some unreasonable search spaces (e.g., subtree *Laptop.HD*), when *Laptop.CPU* already contains offspring *Cache, Brand, Speed* and

Type, one of which might be the right matched pair. It is thus unnecessary to examine the impossible matchings (e.g., *Laptop.HD.Cache ? Notebook.Processor.Cache*) at all.

Search for matching pairs only in proper sub-space will bring two-fold benefits: (1) the decrease of the search space and computation costs, and (2) the exclusion of the error-prone matching pairs. Retrieving the matching in the improper context may lead to some high ranking pairs in the output, which will shield the truly correct mappings.

As shown in Figure 3, a_1 and b_1 are the existing matching pair $< a_1, b_1 >$ where a_1 and b_1 are the elements in ontology *A* and *B* respectively. Given that a_2 is a offspring element of a_1, the question then is where is the proper location of its matched element b_x in *B*, if such a b_x exists? Note $tree(a_1)$ means a subtree rooted at a_1. If $tree(a_1)$ and $tree(b_1)$ are isomorphic, b_x (e.g., b_2) will be located in $tree(b_1)$. If the equivalent set of $tree(a_1)$ collapses into a plainer structure, b_x could be one of b_4 or b_5. However, b_x cannot be b_3 because $a_1 \cong b_1 \prec b_3 \cong b_2 \prec a_1$, where $b_1 \prec b_3$ denotes that b_3 is the ancestor of b_1. It would produce a reasoning cycle which hardly accords with the real world. Hoshiai and Yamane (2004) also observed similar inconsistency in semantic category matching for ontology alignment, but they utilized a generation-and-testing method to eliminate this conflict.

As all of these matching pairs are based on the hypothesis $a_0 \cong b_0$, where all matched elements exist in $tree(a_0)$ and $tree(b_0)$ hence, we can search for them from the root to leaves and only explore the possible and reasonable space among the matched ancestor elements.

Actually, this search space is compatible with the context constraints. For example, there are many matched pairs for element named *type* with high similarity, which is contributed by the same `INT` and `EXT` (of course, although S_{CXT} may be small). Most spurious matching pairs will lie beyond the search space if their ancestor elements cannot be matched. Therefore, the search space pruning will improve the matching performance.

Matching Algorithm

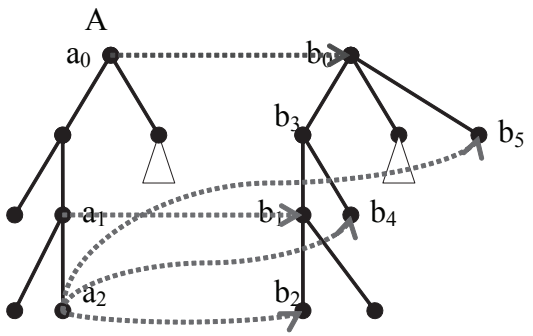
Based on the previous discussion, the algorithm for matching is summarized as following.

Algorithm $O_{NTO}M_{ATCHING}$ (ontology pair *ij*, threshold τ)

{

1. Initialize matching set Φ with the matching pair having maximal *Sim* (*Sim*>τ) between the *root(s)* of *ij*;
2. Set sequence N_{ODES} to the node set of breadth first traversal in *i*;

Figure 3. Matching search space among two models

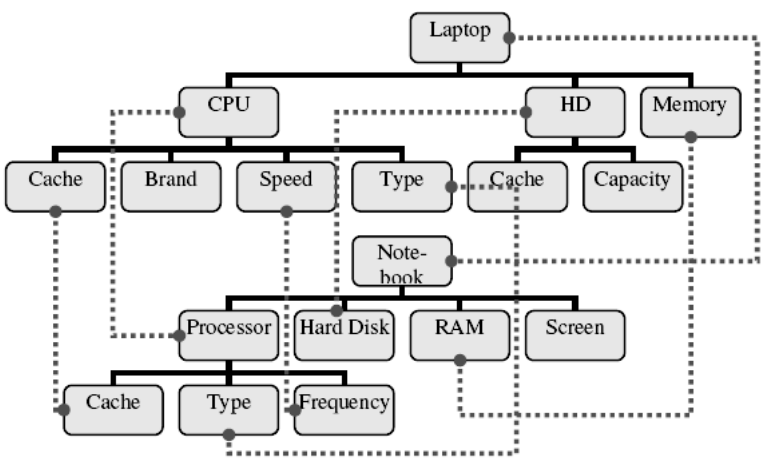


3. **For** (each node e_i in N_{ODES})
4. {
5. $Parents \leftarrow \{parents(e_x) | < e_x, parent(e_i) > \in \Phi, e_x \in O_j\}$;
6. Subspace $\leftarrow$ {offspring of $p | p \in P$};
7. **If** (*Subspace* ! = **null**) //is there proper subspace for examination?
8. $\Phi_{new} \leftarrow \{< e_j, e_i > | Sim(e_j, e_i) > \tau, e_j \in Subspace\}$;
9. **Else**
10. $\Phi_{new} \leftarrow \{< e_j, e_i > | Sim(e_j, e_i) > \tau, e_j \in O_j\}$;
11. $\Phi \leftarrow \Phi \cup \Phi_{new}$;
12. }
13. Sort all matching pairs in Φ according to decreasing *Sim*;
14. **For** (any two matching pairs 12 and 34 in Φ)
15. {
16. **If** 12 and 34 overlap each other) // e.g., 1 is 3
17. { Delete the pair with smaller *Sim* from Φ; }
18. }
19. **Return** Φ;
}

In algorithm $O_{NTO}M_{ATCHING}$, threshold τ controls the degree of element coupling, and Φ is the output set of matching pairs. At first, Φ is initialized with the proper matching pair between the root(s) of *ij* in step 1, which will specialize the search space in the further exploration (steps 2-12). This matching happens in (1) both roots of *ij* or, (2) either root of *i* and *j* (the cases of matching between a root and a nonroot) with maximal *Sim* (*Sim*>τ). We suppose that the root of *i* has maximal *Sim* without the loss of generality. After obtaining the node sequence of ontology *i* from root to leaves (step 2) , the algorithm detects all candidates of matching pairs in which *Sim* is larger than τ in steps 3-12. Here steps 5-6 attempt to acquire the subspace in *j* that covers the matched pair(s) of *i*. Steps 13-18 will eliminate the overlapping pairs in Φ, while more than one element in one ontology match an element in another ontology simultaneously. By the way, the sorting described in step 13 will affect the output of a set of continuous conflicts. For instance, consider that three candidate pairs with *Sim*(*a*,*b*)=0.7, *Sim*(*b*,*c*)=0.8 and *Sim*(*c*,*d*)=0.9. Matching pairs <*a*,*b*> and <*c*,*d*> will be returned if sorting happens; otherwise, only <*c*,*d*> is returned.

Obviously, the algorithm will examine all nodes in O_j to detect the matching candidates when the proper subspace for search is not available (step 10, see the previous section). The worst case of computation is that the

Figure 4. The matchings between notebook ontologies



algorithm never finds any subspace for fewer examinations, which implies that the matching times reach the product of the node number of the ontology pair.

Figure 4 shows the matching pairs between *Notebook ontologies* generated by our algorithm. The ordered pairs of matching elements consist of <*Laptop*, *Notebook*>, <*Laptop.CPU*, *Notebook.Precessor*>, <*Laptop.HD*, *Notebook.HardDisk*>, <*Loptop.Memory*, *Notebook.RAM*>, <*Loptop.CPU.Cache*, *Notebook.Processor.Cache*>, et al.

Ontology Alignment and Construction

Ontology alignment is the automated resolution of semantic correspondences between the representational elements of heterogeneous systems. Our target is to integrate the analogous object ontologies, belonging to the same kind of objects, into a universal ontology, which can cover the definition in various descriptions and provide a common name space for reasoning and computation.

Some elements are sparse when they do not exist in most ontologies. We might ignore them, because (1) some scarce elements are spurious when domain knowledge engineers do not verify them; (2) the integrated ontology will become too trivial (or too complex) if all scarce elements are included; and (3) we cannot accurately access these scarce elements when they are missing in most ontologies and instances. Thus, we can use a predefined threshold σ to filter them out. Namely, if the frequency of appearances of an element is lower than σ, it will be excluded. For example, element *Screen* will be filtered out in *Notebook Ontologies*, if σ is set larger than 1. Incidentally, it is noted that since a set of ontologies with partial order can be generated when σ increases (or decreases), they will form an ontology lattice.

The method of ontology alignment is summarized:

1. Employ the matching procedures outlined in the previous subsection to extract all matching pairs between all analogous ontologies.
2. Select a preferred ontology mdl (plain or hierarchical structure) as a base. In plain listing ontology, all nodes are assigned to root directly. It is convenient to create the corresponding storage structure since it will incur few relational tables. However, hierarchy ontology can preserve more structural information to permit comprehensive conceptualization of the ontologies involved, as all related sub-topics are grouped into different branches.
3. Delete elements in mdl whose frequency is lower than σ.
4. For elements in other ontologies with frequency > σ: (i) If there is no match between this element and those in mdl, we need to select a proper position to insert a new element in mdl. The parent of this new element will need to be added also if the parent cannot be matched too; (2) based on existing matching pairs, add the mapping information into the attributes of the elements to indicate the relation between existing ontologies and the constructed ontology. It enables the namespace transformation from a particular ontology space to the common ontology space. (3) Assign aliases for the elements with various names in different ontologies. Tie a link to the so-called default element if necessary.

Here, default element happens in this situation: a key element for a particular aspect uses a general name in a simple ontology, while the matching element in the complex version uses a combined name. For example, an instance under a simple ontology says *A.HD: 40G*, and another instance under a complex ontology has a branch of matching combined elements *B.HardDisk (Capacity: 40G; Size:s 3.5in)*. Although *B.HardDisk* and *A.HD* are a matching pair, *B.HardDisk.Capacity* and *A.HD* have comparable content as there is no concrete content in *B.HardDisk*. Hence, *B.HardDisk.Capacity* is the default element of *B.HardDisk*,

which leads to that *A.HD* will be mapped to *B.HardDisk.Capacity* when they are computed. The default element of a combined element can be detected by its EXT compared with matched elements of its parent in other concise versions. Namely, if the overlapping of their EXT exceeds a given threshold, we will consider it as the default element.

Based on the above ontology construction approach, the output ontology of *Notebook ontologies* is shown in Figure 5 if we prefer a hierarchical structure and no filtering (σ=0).

USING ONTOLOGY

Ontology Mapping and Data Transformation

The constructed ontology can provide a common framework for information sharing and communication among heterogeneous systems. The mapping between a set of analogous ontologies and the common ontology can serve as the paths for data transformation and semantic computation.

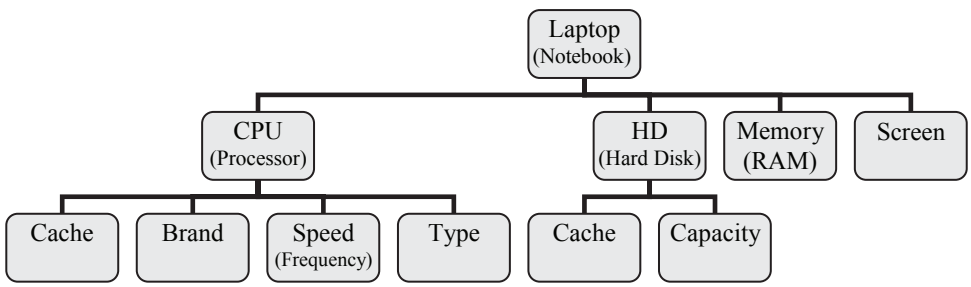
Suppose that object instance *a* using ontology *A* has description such as *Memory: 256 M*, and object instance *b* using ontology *B* from another site has description such as *RAM: 512 M*. We could map them into a common namespace directly by means of the learned ontology. It enables efficient semantic computation such as the comparison of storage capacity between them, as well as semantic-based query (e.g., the criterion of memory being larger than 256*M*). Here, the terminology variations (e.g., *memory* and *RAM*) will be unified according to their semantic. The contents of elements (e.g., 256 and 512) will be interpreted as number under the proper setting of element type.

At the same time, the integrated ontology can be used to verify and even fix the original ontologies and the corresponding instances. In the large-scale Web-based applications, it is inevitable to involve minor noises and errors during the previous procedure of Web → Data → Model (Ye & Chua, 2006). These sparse noises and errors existing in the extracted data and models can be shielded automatically, when there is no proper mapping among them. This will lead to the need for minimal manual check and modification.

Deep Annotation for Web Pages

One of the important functions of our constructed ontology is to annotate the original Web pages with semantic tags. These tags are compatible for all involved sites. To illustrate it more clearly, let us consider two families of notebooks (say a_1, a_2, a_3,..., b_1, b_2, b_3,...) listed in two web sites. We can use the method described in Ye and Chua (2006) to build their object models (schemas) *A* and *B* with different namespaces. We then induce the common ontology (note *AB*) of *A* and *B* based on our

Figure 5. Output ontology for notebook ontologies



proposed approach. Furthermore, we label all pages such as i (or i) with the annotations, which refer to A (or B). From these ontologies, we can induce a reasoning path (bridge), such as $a_i \leftrightarrow A \leftrightarrow AB \leftrightarrow B \leftrightarrow b_i$, which will facilitate deep semantic computation (such as price and configuration comparison) among the data from heterogeneous resources. Once we publish all relevant data such as $\{a_i, b_i, A, B, AB\}$ on the Web, they will become a part of the Semantic Web (He et al., 2003).

Here, we briefly discuss the contents existing in the annotations, which are specified by the corresponding object models and common ontologies. When we use semantic tags to label the source codes, we cannot ignore two kinds of particular attributes: object identifier and UOM-relevant attributes (amount and measure). The former is similar to universal resource identifier (URI), which will be used to anchor the object location on the Web. The latter related to UOM can support the deep semantic computation among the objects. For example, for a notebook has description such as *VAIO ... RAM: 512M; Hard Disk: 40G ....* It will be annotated as follows[3].

<notebook id="VAIO1000" namespace="sony.notebook">VAIO ... RAM: <RAM amount="512" measure="M">512M</RAM>; Hard Disk: <HardDisk amount="40" measure="G">40G</HardDisk> ... </notebook>

We can use a simple shallow parsing program, based on heuristic hints from the detailed framework in other sites, to produce deep annotation of the complex expression. This enables the shallow parser to take a proper decomposition and annotation action among the common namespace.

SYSTEM AND EXPERIMENTS

We developed an intelligent web information platform called OnModer that supports Web → data → model → Ontology processing (Ye & Chua, 2006). Because of the limitation of space, here we summarize only some of the experimental results.

Ontology Merging Test

We build 30 ontologies from sites listed in Ye and Chua (2006). Their topics include computer, hard disk, printer, digital camera, MP3 player, and personal profile and so forth, each of which contains five ontologies. The number of elements in per ontology varies from 4 to more than 100, with an average of 27 in a 2 to 3 layered structure. By setting the coefficient α, β, γ in Equation. 3 as well as the similarity threshold τ (see previous section) to 0.5, 0.3, 0.2, 0.5 respectively, the algorithm $O_{NTO}M_{ATCHING}$ generates 413 matching element pairs, on the average, when it examines all possible ontology pairs of 10 per topic. When the filtering threshold σ is set to 1 (i.e., elements appearing only once are ignored), the average number of elements in the output-merged ontologies is 39.5.

Here, we observe the followings. (1) Ontologies from the same site exhibit much fewer vocabulary variations among them although there might be some structure modifications. The intension and extension bring large contributions to element matching. (2) There are lots of terminology changes among the matching elements from different sites. Semantic and syntax exploration is crucial to detecting the inherent equivalence. WordNet is a very powerful resource to bridge the gap of vocabulary. (3) The search space optimization as described in the previous section can provide good solution to avoid incorrect matching among elements with identical names (such as *type, cache, size* etc.). Otherwise, we need to carefully tune the parameters to avoid such mistakes.

I3CON Ontology Alignment Test

We also employ OnModer to perform ontology alignment task performed in I3CON (Hughes, 2004). This test set contains 16 ontologies in 8 domains, such as animals, sports, and so forth.

Since there is no instance in this test corpus, we reset coefficient α, β, γ to 0.7, 0.0 and 0.3 respectively. The comparison between our results and I3CON results (Hughes, 2004) are listed in Table 2, which includes the range of

performance in terms of precision, recall and 1 measure of the four participating groups. The results show that OnModer is able to achieve significantly better results (1) than those reported in I3CON (1). For example, OnModer can detect the right pairs with different vocabulary such as *Do_Foul* and *InfractionAction* when WordNet returns the overlapping semantic interpretation (*Do≈Action, Foul≈Infraction*). OnModer can also handle the cases whose merging support information comes from the context of the involved elements.

On further analysis, it seems that this test corpus is not too complicated since there are fewer variations among structure, terminology, and granularity as compared to the corpus in previous subsection. It shows that the real ontologies derived from real applications may be more complicated and diversified than those employed in research community. We need to pay more attention on it when we investigate ontology integration.

DTD Merging Test

We also apply OnModer to perform XML schema-merging test, using the data set provided by Lee et al. (2002). This test set contains 150 DTDs on health and publication downloaded from the Internet. The hierarchy depth ranges from 2 to 20 levels and the number of nodes varies from about 20 to 1000. The main characteristics of this corpus are that there are more nodes with deeper depth as compared to the previous two corpora. In order to test the search space optimization and concept similarity reuse in these large-scale DTDs, OnModer detects the matching pairs of elements from XML schemata using diverse configurations. The experiments show that the matching time is increased by about from 60% to 350% when search space optimization and concept similarity reuse are not employed.

Meanwhile, in order to examine the accuracy of integration, we manually examine the outputs in 50 randomly selected ontology pairs. The precision, recall and 1 measure in these sampling data are 92.1%, 78.3%, and 84.6% respectively. It is difficult to carry out a quantitative comparison to Lee et al. (2002) since the examined data and evaluation methods are different from each other. We find the relatively lower recall (78.3%) is caused by the fact that WordNet cannot determine the semantic similarity of some equivalent terminologies in health. Moreover, we also observe that the overall accuracy of matching among these sampling data cannot be improved when more searches are explored as search space optimization is disabled.

CONCLUSION

This article presented a framework for integrating multiple ontologies from heterogeneous systems (especially, from structured Web pages) into a common ontology, which facilitates some important applications such as web information extraction, database integration, content translation, and distribution. The main contributions of our research are three-fold.

First, we defined a universal similarity paradigm that can effectively combine the evidences for matching in ontology and instance, semantic,

Table 2. Performance of OnModer on I3CON corpus

	Results in I3COM		OnModer
	Range	Best participating group	
Precision	29-77%	Lockheed Martin ATl	86%
Recall	50-80%	Teknowlodge Co.	89%
F1-measure	33-77%	Lockheed Martin ATl	87.5%

and structure, as well as linguistic and syntactic features. This similarity paradigm could reflect the implicit coherences among the ontologies having large, structural, terminological, and granular variations. Second, we designed a top-down matcher based on matching space analysis and similarity reuse, which needs less computational cost and alleviate error-prone matchings. Third, we proposed an ontology alignment and construction method. The output ontology will follow users' configuration such as their preferred structure and filtering threshold. It facilitates deep annotation and interoperation in structured web pages from heterogeneous systems.

In the future, we would like to investigate the data-adaptive approach to determining the combination parameters for `INT`, `EXT`, and `CXT`, as we notice that we need to tune them in different ontology corpora. At the same time, in order to examine and refine our techniques comprehensively, we would also like to develop a public Web service platform to assemble the tremendous ontologies about IT products on the Web.

REFERENCES

Bao, J., & Honavar, V. (2004). Collaborative ontology building with wiki@nt. Workshop evaluation of ontology based tools (EON) (70-80).

Baziz, M., Boughanem, M., & Aussenac-Gilles, N. (2004). The use of ontology for semantic representation of documents. Workshop semantic web in SIGIR (43-51).

Dill, S., Eiron, N., Gibson, D., Gruhl, D., & Guha, R. (2003). Semtag and seeker: Bootstrapping the semantic web via automated semantic annotation. WWW (831-837).

Doan, A., & Halevy, A. Y. (2005). Semantic-integration research in the database community: A brief survey. J. of AI Magazine, 26(1), 83-94.

Doan, A., Madhavan, J., Domingos, P., & Halevy, A. (2002). Learning to map between ontologies on the semantic web. WWW (662–673). ACM Press.

Euzenat, J. (2004). Introduction to the EON ontology alignment context. Workshop evaluation of ontology-based tools (47-50).

Halkidi, M., & Nugyen, B. (2003). THESUS: organizing web document collections based on link semantics. VLDB, 320-332.

He, H., Meng, W., Yu, C. T., & Wu, Z. (2003). Wise-integrator: An automatic integrator of web search interfaces for e-commerce. VLDB (357-368).

Hoshiai, T., & Yamane, Y. (2004). A semantic category matching approach to ontology align. Workshop evaluation of ontology based tools (EON) (70-80). (http://www.atl.external.lmco.com/projects/ontology/i3con.html)

Hughes, T. (2004). Results for I3CON ontology alignment experiment. Workshop evaluation of ontology based tools (EON). (http://www.atl.external.lmco.com/projects/ontology/i3con.html)

Kalfoglou, Y., & Schorlemmer, M. (2003). Ontology mapping: the state of the art. Knowledge Engineering Review, 18(1), 1-31.

Kietz, J.-U., Volz, R., & Maedche, A. (2000). Extracting a domain-specific ontology from a corporate intranet. Proceedings of CoNLL-2000 and LLL-2000 (167–175). Lisbon, Portugal.

Lee, M., & Yang, L. (2002). Xclust: Clustering xml schemas for effective integration. CKIM (292-299).

Lesk, M. (1986). Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. SIGDOC, (24-26).

Madhavan, J., Bernstein, P. A., & Rahm, E. (2001). Generic schema matching with cupid. VLDB, (49-58).

Maedche, A. (2002). Ontology learning for the semantic web. Kluwer Academic Publishers.

Miller, G. A. (1995). Wordnet: A lexical database. Communication of the ACM, 38(11), 39-41.

Noy, N., & Musen, M. (2003). The PROMPT suite: Interactive tools for ontology merging and mapping. Int'l J. of Human-Computer Studies, 983-1024.

Rahm, E., & Bernstein, P. A. (2001). A survey of approaches to automatic schema matching. J. of VLDB, 10, 334-350.

Shvaiko, P., & Euzenat, J. (2005). A survey of schema-based matching approaches. Journal on Data Semantics, 4, 146-171. (LNCS3730)

Ye, S., & Chua, T.-S. (2004). Detecting and partitioning of data objects in complex web pages. Int'l conf. of web intelligence (669-672).

Ye, S., & Chua, T.-S. (2004). *Learning object model from product web pages*. workshop semantic web of SIGIR (69-80).

Ye, S., & Chua, T.-S. (2006). Learning object models from semi-structured web documents. *IEEE TKDE, 18*(3), 334-349.

ENDNOTES

1 Which, called *Notebook ontologies* in the following discussion, are about the specification of notebook (laptop) extracted from www.sony.com and www.bizrate.com by our web information extraction and integration system, OnModer (Ye & Chua, 2006). Incidentally, in these ontologies, we already deleted some elements for simplification and just preserved the elements that could exhibit the prominent obstacles during matching.

2 Incidently, the root of i is excluded from N_{ODES} since it has been considered in step 1. This ensures that *parent*(.) is always valid in step 5.

3 Element names such as *RAM* and *Hard Disk* will be not included in labeled elements, since they might be shared by more than one object in a web page.

Ye, Shiren received the PhD in computer software from the Institute of Computing, the Academy of Sciences in 2001. He is a research fellow in School of Computing, National University of Singapore. His research interests include machine learning, Web and text mining, Semantic Web and natural language processing.

Chua, Tat-Seng is the professor at the School of Computing, National University of Singapore. He was the Acting and Founding Dean of the School of Computing from 1998-2000. He spent three years as a research staff member at the Institute of Systems Science (now I2R) in late 1980s. Dr. Chua's main research interest is in multimedia information processing, in particular, on indexing, retrieval and extraction of information in video and text. His current projects include: news video retrieval, question answering (QA), video QA, and information extraction on the web. He recently concluded a large-scale research project to manage and retrieve digital video, and web-based information. He obtained his PhD from the University of Leeds, UK. Dr. Chua is active in the international research community. He has organized and served as program committee member of numerous international conferences in the areas of computer graphics and multimedia. He serves in the editorial boards of: IEEE Transactions of Multimedia (IEEE); The Visual Computer (Springer-Verlag); and Multimedia Tools and Applications (Kluwer). He is the member of Steering Committee of Computer Graphics Society (Geneva), and Multimedia Modeling Conference (international).