Journal of Universal Computer Science, vol. 15, no. 4 (2009), 877-897
submitted: 30/6/08, accepted: 27/2/09, appeared: 28/2/09 © *J.UCS*

Complexity Analysis of Ontology Integration Methodologies: a Comparative Study

Trong Hai Duong, Geun Sik Jo
(School of Computer and Information Engineering
Inha University, Korea
haiduongtrong@gmail.com, gsjo@inha.ac.kr)

Jason J. Jung
(Department of Computer Engineering
Yeungnam University, Republic of Korea
j2jung@gmail.com)

Ngoc Thanh Nguyen
(Institute of Informatics and Engineering
Wroclaw University of Technology, Poland
thanh@pwr.wroc.pl)

Abstract: Most previous research on ontology integration has focused on similarity measurements between ontological *entities*, e.g., *lexicons*, *instances*, *schemas* and *taxonomies*, resulting in high computational costs of considering all possible pairs between two given ontologies. In this paper, we propose a novel approach to reducing computational complexity in ontology integration. Thereby, we address the *importance* and *types of concepts*, for priority matching and direct matching between concepts, respectively. *Identity-based similarity* is computed, to avoid comparisons of all properties related to each concept, while matching between concepts. The problem of conflict in ontology integration has initially been explored on the *instance-level* and *concept-level*. This is useful to avoid many cases of mismatching.

Keywords: Ontology integration, Importance concepts, Conflict, Identity-based similarity
Categories: E.1, H.3.0, H.3.3, I.2.0, I.2.1, I.2.2, I.2.3, I.2.4, M.7

1 Introduction

Ontology has been important not only in the semantic web and semantic data processing, but also in various research fields and application areas, e.g., knowledge engineering, database design and integration. This means that such ontologies play a central role in facilitating knowledge exchange between several heterogeneous sources. To do this efficiently, distributed ontologies have to be integrated.

In general, the problem of ontology integration can be described as follows: *Given a set of ontologies* $\{O_1, ..., O_n\}$*, a unified ontology O capable of replacing them must be found* [Gangemi et al. 1998; Pinto et al. 2001]. The ontology reflects the creator's own understanding of knowledge, like the relation of a literary work to its author. The best explanation of the phenomenon of human consciousness is William James' famous stream of consciousness theory. He observed that human consciousness has a composite structure including *substantive parts* (thought or idea) and *transitive parts* (fringe or penumbra), and drifts from thought-to-thought. Thus, ontology integration is a complex task, since the ontologies have heterogeneous characteristics and forms, e.g., languages, structures, etc. Therefore, [Lee et al. 2006] suggested an ontology architecture providing a solid basis for studies about the task of ontology integration. [Pinto and Martins 2001] identified the activities that must be performed in ontology integration. Various tools supporting ontology integration have been introduced:

- Cupid [Madhavan et al. 2001] implements an algorithm comprising linguistic and structural schema matching techniques, and computation of similarity coefficients using domain-specific thesauri.
- FCA-Merge is a method for merging ontologies, which is a bottom-up approach supporting a global structural description of the merging process. For the source ontologies, it extracts instances from a given set of domain-specific text documents by applying natural language processing techniques. Based on the extracted instances, mathematical techniques from formal concept analysis are applied. The produced result is explored and transformed to the merged ontology by the ontology engineer.
- COMA [Do and Rahm 2002] is a schema matching tool based on parallel composition of matchers. It provides an extensible library of matching algorithms, a framework for combining obtained results, and a platform for evaluating the effectiveness of the matchers. Most implement string-based techniques, such as affix, n-gram, edit distance; others share techniques with Cupid, e.g., thesauri look-up.
- GLUE [Doan et al. 2001, 2002, 2003, 2004] is a system that employs a multi-strategy machine learning technique with joint probability distributions. This tool can identify the similarities of instances. Also, it can compare the relations of ontologies based on the similarity results of instances. GLUE uses two kinds of base learners: a name learner and a number of content learners.
- OntoMerge [Dou et al. 2005] is a system for ontology translation of the semantic web. Ontology translation refers to such tasks as (i) dataset translation, that is, translating a set of facts expressed in one ontology to those in another ontology; (ii) generating ontology extensions, that is, given two ontologies o and o' and an extension (sub- ontology) o_s of the first one, build the corresponding extension o'_s, and (iii) query answering from multiple ontologies. The main principle of this approach is ontology translation via ontology merging and automated reasoning.
- H-Match [Castano et al. 2006] is an automated ontology matching system. H-Match inputs two ontologies and outputs (one-to-one or one-to-many) the correspondences between concepts of these ontologies with the same or closest intended meaning. The approach is based on a similarity analysis via affinity metrics, e.g., term-to-term affinity, data type compatibility, and thresholds. H-Match computes two types of affinities (in the [0,1] range), viz., linguistic and contextual. These are then combined via weighting schemas, thus yielding a final measure, viz., semantic affinity. Linguistic affinity builds on the thesaurus-based approach of the Artemis system.

– RiMOM (Risk Minimisation based Ontology Mapping) [Tang et al. 2006] is an approach inspired by Bayesian decision theory, which formalizes ontology matching as a decision making problem. Given two ontologies, it aims for the optimal and automatic discovery of alignments, which can be complex (for example including concatenation operators). The approach first searches for concept-to-concept correspondences, and then searches for property-to-property correspondences.

We consider that most of the aforementioned works merely involve blind or exhaustive matching among all concepts in different ontologies and all properties belonging to each concept. Therefore, the computational complexity increases rapidly in integrating large ontologies. Additionally, they have not yet explored the conflicts between ontologies on different levels such as the *instance level* (e.g. the same instance but different concepts) and the *concept level* (e.g. multiple forms of the same concept, overlapping but different concepts, the same concepts but different names.). For these reasons, our work focused on reducing *complexity* in the ontology integration. The term *complexity* is used in two senses: First, the *complexity* is based on the heterogeneous characteristics and forms of ontologies, e.g., languages, structures, etc. It may cause semantic conflicts in ontology integration. Second, the *complexity* is the time of matching between concepts belonging to different ontologies. In this paper, we focused on these problems, and our main contributions are as follows:

– The *importance of concepts* has been proposed for priority matching between concepts. The importance measurement of a concept takes into account the contributions from all the other concepts in the ontology, via all kinds of relations, including not only subsumption and non-subsumption ones, but also concepts with associated attributes exhibiting a mutually reinforcing relationship. We can identify a concept's possible position in the hierarchy via the importance measurement of the concept. Thus, we accept the supposition that *assuming two ontologies O_i and O_j must be integrated, the concept c_i belonging to ontology O_i should be priority matched to the concept c_j or neighbors of c_j belonging to ontology O_j, where distance between c_i's importance measurement and c_j's is minimal.*

– The *types of concepts* have been expanded from [Duong et al., 2009] and combined with *importance concepts* in this proposal, for direct matching between the concepts of the same type, instead of blind or exhaustive matching among all concepts. This proposal has the following aspects: First, we can classify all concepts belonging to different ontologies into five disjoint groups based on the concept's identities. Each group consists of the same type of concepts. It is only possible for concepts in the same disjoint groups to be directly matched. Second, we assign a weight of importance vector to each disjoint group in ascending order of the importance weight. Each pair of the same disjoint groups belonging to different ontologies must be directly matched.

– A novel matching heuristic, viz., an *identity-based similarity*, is presented. While calculating similarities between concepts, we simply focus on the identity of the concept, instead of comparing all properties related to each concept.

- The problem of conflict in ontology integration has been explored on the *instance-level* and the *concept-level*. It is useful for avoiding many cases of mismatching.
- In the experiment section, we have applied the aforementioned methods to design an effective algorithm for ontology matching. We have also compared our method with the previous studies.

2 Basic Notions

We assume a real world ($\boldsymbol{A}, \boldsymbol{V}$) where $\boldsymbol{A}$ is a finite set of attributes and $\boldsymbol{V}$ is the domain of $\boldsymbol{A}$. Also, $\boldsymbol{V}$ can be explained as a set of attribute values, and $\boldsymbol{V} = \bigcup_{a \in \boldsymbol{A}} V_a$ where V_a is the domain of attribute a. In this paper, we make the following assumptions which presented in our approach [Duong et al. 2009]:

Definition 1 (Ontology). An ontology is a quintuplet:

$$O = (C, \sum, I, R, Z) \tag{1}$$

where,

- $\boldsymbol{C}$: set of concepts (the classes);
- $\boldsymbol{I}$: set of instances of the concepts;
- $\boldsymbol{R}$: set of binary relations between the concepts from C, or between the concepts from $\boldsymbol{C}$ and the values defined in a standard or user-defined data type;
- $\boldsymbol{Z}$: set of axioms, which can be interpreted as integrity constraints or relationships between instances and concepts. This means that Z is a set of restrictions or conditions (necessary $\&$ sufficient) to define the concepts in C;
- $< \boldsymbol{C}, \sum >$: is the taxonomic structure of the concepts from C where $\sum$ is the collection of subsumption relationship ($\sqsubseteq$) between any two concepts from C. For two concepts c_1 and $c_2 \in C, c_2 \sqsubseteq c_1$ if and only if any instances that are members of concept c_2 are also members of concept c_1, and the converse is not true.

R is known as the set of properties. For every $p \in R$, there is a specific domain $\boldsymbol{D}$ and range R such that $p : D \rightarrow R$, where $D \subset C$ and if $R \subset C$ then p is called an object property, otherwise if R is a set of standard or user-defined data types then p is called a data type property. We assume that concepts c and c'correspond to the domain and range of property p respectively, where p is also known as an attribute of concept c. There are two given instances v and v' that belong to the corresponding concepts c and c' respectively. We denote vR^pv' as the relation from instance v to v' via the property (attribute) p and the relation from instance v' to v via the property p is denoted as $vR^{-p}v'$.

Definition 2 (Concept). A concept c of an ($\boldsymbol{A}$, $\boldsymbol{V}$)-based ontology is defined as a triple:

$$c = (z_c, A^c, V^c) \tag{2}$$

where c is the unique identifier for instances of the concept. $A^c \subseteq \boldsymbol{A}$ is a set of attributes describing the concept and $V^c \subseteq \boldsymbol{V}$ is the attributes' domain: $V^c = \bigcup_{a \in A^c} V_a$. The $z_c \subset Z$ is the set of restrictions or conditions (necessary $\&$ sufficient) to define the concept c. The z_c can be represented as a constraint function $z_c : A^c \rightarrow \boldsymbol{Z}$ such that $z_c(a) \in \boldsymbol{Z}$ for all $a \in A^c$.

Pair (A^c, V^c) is called the possible world of concept c and A^c is called the structure of the concept c. Notice that within an ontology there may be two or more concepts with the same structure. If this is the case, the constraint function z_c is useful for expressing associated relationships. For example, two concepts $RedWine$ and $WhiteWine$ have the same structure $\{hasMaker, hasColor\}$. But $z_{RedWine}(hasColor)$ = $\{\exists hasColor$ = $red\}$ and $z_{WhiteWine}(hasColor) = \{\exists hasColor = white\}$.

We denote v_c as the description of the instance within concept c. The v_c can be presented as a function: $v_c : A^c \rightarrow V^c$ such that $v_c(a) \in V^c$ for all $a \in A^c$. By $Ins(O, c)$ we denote the set of instances belonging to concept c in ontology O and thus $I = \bigcup_{(c \in C)} Ins(O, c)$.

Definition 3 (Key Identity). The *Key Identity* (KI) of a concept is an attribute from set A^c which provides a unique value to each individual of the concept in the real world $(\boldsymbol{A}, \boldsymbol{V})$. Formally, if ki is a KI of the concept c, it satisfies the following conditions:
- $ki \in A^c$,
- $x \in Ins(O, c), \forall v_1, v_2 \in \boldsymbol{V}, xR^{ki}v_1 \wedge xR^{ki}v_2 \rightarrow v_1 = v_2$, and
- $x \in Ins(O, c), \forall v_1, v_2 \in \boldsymbol{V}, xR^{-ki}v_1 \wedge xR^{-ki}v_2 \rightarrow v_1 = v_2$.

The first two conditions mean than the KI of a concept must necessarily provides the same KI value for the same instance of the concept. The third condition means that it must be sufficient to recognize that two actual instances with the same KI value are the same instance. The three conditions imply that the KI of a concept should be globally identifiable for instances in the real world ($\boldsymbol{A}$,$\boldsymbol{V}$). The KI is also known as the rigid property [Guarino et al., 2000] that is essential to all its instances.

Definition 4 (Local Identity). The *Local Identity* (LI) of a concept is an attribute from set A^c, which provides a unique value to each individual of the concept in the possible world (A^c, V^c). Formally, if li is an LI of concept c, it satisfies the following conditions.
- $li \in A^c$,
- $x \in Ins(O, c), \forall v_1, v_2 \in V^c, xR^{li}v_1 \wedge xR^{li}v_2 \rightarrow v_1 = v_2$, and
- $x \in Ins(O, c), \forall v_1, v_2 \in V^c, xR^{-li}v_1 \wedge xR^{-li}v_2 \rightarrow v_1 = v_2$.

The difference between KI and LI is that the LI of a concept is only locally identifiable for instances in the possible world (A^c, V^c).

Example 1. We consider the concept $Person$ owning the $hasFingerprint$, which is KI. The instance $Jean$ has $hasFingerprint$ of 000155BDC, and the instance $Peggy$ has $hasFingerprint$ of 000155BDC. Because $hasFingerprint$ is a KI, we can deduce that $Jean$ and $Peggy$ must be the same instance. Note that because $hasFinger-print$ is a KI, there is always an inverse relation $isFingerprintOf$. If two instances 000155BDC and 000155BEF are $isFingerprintOf$ of the instance $Jean$, 000155BDC and 000155BEF must be the same instance. However, notice that if 000155BDC and 000155BEF were explicitly stated to be two different instances, these statements would lead to an inconsistency.

Here, we define two more characteristics of properties of concepts as follows:

Definition 5 (Inheritance Identity). The *Inheritance Identity* (II) of a concept is a set of identified attributes which is inherited from its super-concepts, which provides a unique II value to each instance of the concept.

Definition 6 (Constant Property). The *Constant Property* of the concept c is a property within A^c and it provides a common attribute value for all individuals belonging to the concept.

Example 2. We consider the concept $MalePerson$ with its structure $\{Person \wedge has-Gender = Male\}$. The $MalePerson$ is defined as the concept $Person$ which satisfies $z_{MalePerson}(hasGender) = \{\forall hasGender = Male\}$. The property of $hasGender$ has the constant value of $Male$ for all individuas which belong to the concept *MalePerson*. Therefore, the $hasGender$ is a constant property.

3 Analyzing Ontologies to Reduce Computational Complexity

3.1 Identifying Concept for Reducing Search Space

In this section, we present our ideas on how to reduce the search space for matching between concepts belonging to different ontologies. After many experiments, we found that it is useful to classify all concepts belonging to different ontologies into more disjoint groups. We assume that *two ontologies must be integrated. It is only possible for two concepts belonging to the same disjoint groups of the ontologies to match*.

Based on our study, we found that the same concept has certain fixed roles in similar ontologies. Thus the concept's position lies in a fixed interval of the ontologies' hierarchy. The role of a concept is determined by its attributes and the associated restrictions. Based on the concept's characteristics, we can classify concepts belonging to an ontology into five disjoint groups as follows: (see figure 1)

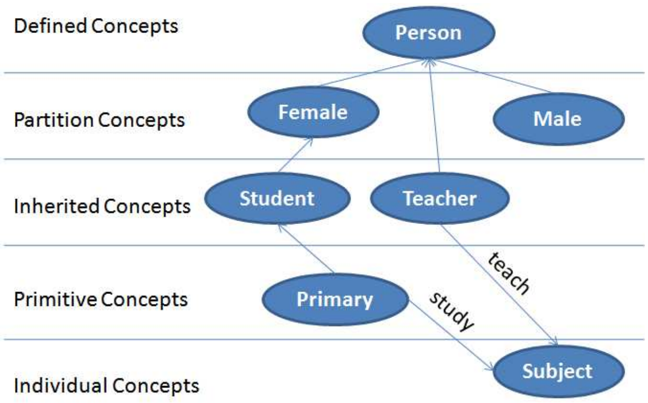

Figure 1: A classification of concepts of Ontology E_1.

Definition 7 (Defined Concept). The *Defined Concept* (DC) is a concept which has at least one KI. Formally, if c is a DC, its constraint function z_c satisfies the following conditions:
- $\exists a \in A^c, z_c(a)$ is a necessary and sufficient condition, and
- the attribute a is a KI.

Example 3. We refer to $Example$ 1 in which the concept $Person$ is an example of the DC. The DC is also known as a rigid sort [Guarino et al. 2000] that supplies a principle of identity for its individuals.

Definition 8 (Partition Concept). The *Partition Concept* (PC) is part of a DC. Formally, if c is a PC, it satisfies the following conditions:
- $\exists a \in A^c, \forall x \in v_c : x(a)$ is a constant value, and
- the concept c is a defined concept satisfying $z_c(a)$.

Example 4. We consider two concepts $MalePerson$ and $FemalePerson$ with the same structure $\{Person, hasGender\}$. The $MalePerson$ is defined as the concept *Person* that satisfies $z_{MalePerson}(hasGender) = \{\forall hasGender = Male\}$. The $Fe-malePerson$ is defined as the concept *Person* that satisfies $z_{FemalePerson}(hasGender)$ =$\{\forall hasGender = Female\}$. Thus the concepts $MalePerson$ and $FemalePerson$ are PCs.

Definition 9 (Inherited Concept). The *Inherited Concept* (IC) is a sub-concept of either a defined concept or a partition concept, or another inherited concept. It has at least one LI. Formally, if c is a IC, then its constraint function z_c satisfies the following conditions:
- $\exists a \in A^c, z_c(a)$ is the necessary and sufficient condition , and
- the attribute a is a LI.

Example 5. If two concepts *Student* with *KI hasIdStudent* and *Employee* with *hasIdEmployee* are sub-concepts of the concept *Person*, we can infer that the *Student* and *Employee* must be ICs.

Definition 10 (Primitive Concept). The *Primitive Concept* (PvC) is a concept which has neither KI nor LI and it is defined from other concepts. Formally, if a concept is PvC, its constraint function z_c does not have any set of necessary $\&$ sufficient conditions and no concept is used as a sub-concept of PvC.

Example 6. We consider the *UndergraduateStudent, MasterStudent*, and *DoctoralStudent* defined via the concept *Student*. Since they do not have any set of necessary $\&$ sufficient conditions, then we can infer that these concepts must be PvC. Notice that the concepts are never used a sub-concepts of PvC.

Definition 11 (Individual Concept). The *Individual Concept* (IvC) is a concept with no object properties and it is not defined from other concepts.

Example 7. Considering $Figure$ 1, the concept *Subject* is an *individual concept*. *Individual concept* can be considered as a set of instances or data types defined by the ontology creator.

Proposition 12 Types of Concepts. *For a given ontology O belonging to the real world* (A, V)*, we denote four different sets of* DCs, PCs, ICs*,* $PvCs$ *and* $IvCs$ *to be* C^{DC}*,* C^{PC}*,* C^{IC}*,* C^{PvC} *and* C^{IvC} *respectively.*

1. $C^{DC} \cup C^{PC} \cup C^{IC} \cup C^{PvC} \cup C^{IvC} = \boldsymbol{C}$
2. $C^{DC} \cap C^{PC} \cap C^{IC} \cap C^{PvC} \cap C^{IvC} = \emptyset$
3. *The levels of concepts increase in the order of IvC, PvC, IC, PC and* DC *respectively.*

The $Proposition$ 12 shown that concepts belong to an ontology can be classified into five disjoint groups.

Definition 13 (Possible Similarity Set (PSS)). For two given ontologies O_1 and O_2, a concept c belongs to the ontology O_2. If the ontologies must be integrated, the *possible similarity* set of the concept c is defined as the set of concepts belong to ontology O_1 and the concepts are the same type as concept c.

Example 8. We consider the ontology E_2 written in OWL as follows:
$Individual\{hasFingerprint, name\}$
$FemalePerson\{Individual \wedge hasGender = Female\}$
$MalePerson\{Individual \wedge hasGender = Male\}$
$Teacher\{Individual, IdTeacher, TeachTo\}$
$Learner\{FemalePerson, IdLearner\}$
$PreSchool\{Learner \wedge Level = 1, LearnTo\}$
$Subject\{Name, Credit\}$

The attributes *hasFingerprint*, *IdTeacher*, *IdLearner* are owl:DatatypeProperty with three restrictions: owl:FunctionalProperty, owl:InverseFuncti-onalProperty, and owl:cardinality = 1. So they are identities. Based on the above definitions and the position of the concepts in the hierarchy of the ontology, we can classify the concepts into five groups as follows:
$C^{DC}_{E_2} = \{Individual\}$
$C^{PC}_{E_2} = \{FemalePerson, MalePerson\}$
$C^{IC}_{E_2} = \{Learner, Teacher\}$
$C^{PvC}_{E_2} = \{PreSchool\}$
$C^{IvC}_{E_2} = \{Subject\}$

Similar to the ontology E_1 (see figure 1), we have:
$C^{DC}_{E_1} = \{Person\}$
$C^{PC}_{E_1} = \{Female, Male\}$
$C^{IC}_{E_1} = \{Student, Teacher\}$
$C^{PvC}_{E_1} = \{Primary\}$
$C^{IvC}_{E_1} = \{Subject\}$

If two ontologies must be integrated, it is only possible for two concepts belonging to the same disjoint groups of the ontologies to match. This is direct matching between the same types of concepts. It is shown in $Figure$ 2.

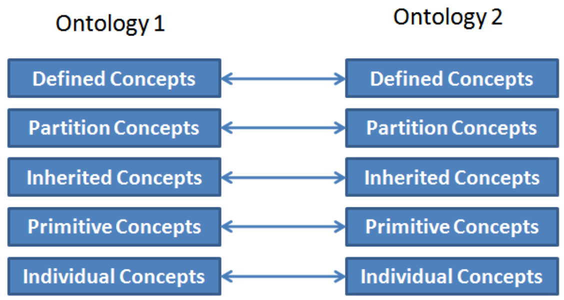

Figure 2: Directly matching between the same type concepts.

3.2 Importance Concept for Priority Search

Cognitive support for ontology integration by emphasizing *importance concepts* has not been explored. However, *importance concepts* have been considered in ontology understanding in several early studies such as [Gang et al. 2008]. They consider that the importance measurement of a concept must take into account the contributions from all the other concepts in the ontology via characterization of four features of potentially important concepts and relations, which drive the drifting stream of consciousness:

- A concept is more important if there are more relations originating from the concept.
- A concept is more important if there is a relation originating from the concept to a more important concept.
- A concept is more important if it has a higher relation weight to any other concept.
- A relation weight is higher if it originates from a more important concept.

Here, the term is being used in three senses. First, it explains what is important (or interesting). The term *importance* is used as a metric for measuring the extent that the ontology creator suggests a concept or relation to users. Second, a concept is regarded as a source that owns a set of relations related to other concepts. Finally, concepts and relations exhibit a mutually reinforcing relationship.

Our work differs from the above approach, because our aim is to design a method for reducing the computational *complexity* in ontology integration. First, we agree that the *importance measurement* of a concept must take into account the contributions from all the other concepts in the ontology via all kinds of relations, including both subsumption and non-subsumption ones. However for our purposes, the term *importance of concepts* is used to identify the concept's position in the hierarchy. It provides the importance level of the concept to the same type of concepts. Therefore, the *important concepts* can be used for priority matching between the same types of concepts. Here, we address four more features of potentially important concepts and relations as follows:

- A concept is more important if there are more relations originating from other concepts to the concept.
- A concept is more important if there are more important concepts which are partitions of the concept.
- A concept is more important if there are more important concepts which are inherited from the concept.
- A concept is more important if there are more important concepts which are defined from the concept.

Definition 14 (Imported Concepts). The *Imported concepts* of the concept c (I_c) are the set of concepts defined from the concept c.

- If c is a DC, the imported concepts of c include the partition concepts of c, inherited concepts of c and primitive concepts defined from c.
- If c is a PC, the imported concepts of c consist of the inherited concepts of c and primitive concepts defined from c.
- If c is a IC, the imported concepts of c consist of inherited concepts of c and primitive concepts defined from c.
- If c is a PvC, the imported concepts of c comprise the primitive concepts defined from c.

Definition 15 (Forward concepts). The *Forward concepts* of the concept c (F_c) are the set of concepts of the *Range* of properties belonging to the concept c.

Definition 16 (Backward concepts). The *Backward concepts* of the concept c (B_c) are the set of concepts that have relations/properties to the concept c.

Let r(c_i) be a function of an importance weight of concept c_i, r_i=r(c_i) be a importance weight value of the concept c_i, w(c_i, c_j) be a relation weight function, and $w_{i,j}$=w(c_i, c_j) be the weight of all relations from c_i to c_j. It is possible that there exists more than one relation from concept c_i to concept c_j. For example, the concept $Student$ may have two relations $study$ and $learn$ with the concept $Subject$. Some students may have a part time relation with an assistant professor, so instead of study they may also teach a given subject. Therefore, $r_j w_{i,j}$ is the total importance value of all the relations from concept c_i to concept c_j.

According to the last hypothesis in [Gang et al. 2008], we present a similar recursive formula which computes the weight of relation starting from concept c_i to concept c_j at the (*k+1*)th iteration. The weight is proportional to the importance of c_i and is the inverse ratio of the sum of all the importance values of c_j's backward concepts at the *k*th iteration.

$$w_{k+1}(c_i, c_j) = \frac{r_k(c_i)}{\sum_{t_i \in B_i} r_k(t_i)} \tag{3}$$

Based on our four hypotheses combined with the first three hypothesises in [Gang et al. 2008], we present recursive formulae which calculate the importance of concept c_i at the (*k+1*)th interation. The importance consists of two parts; all the importance values of c_i's imported concepts with probability α, and the weight of relations from c_i to the forward concepts with probability λ. However it is only applied if the concept $c_i \in C \backslash C^{IvC}$. The importance weight of the concept c_i is as follows:

$$r_{k+1}(c_i) = \alpha \sum_{c_j \in I_i} r_k(c_j) + \lambda \sum_{c_j \in F_i} w_{k+1}(c_i, c_j) r_k(c_j), \alpha + \lambda = 1 \qquad (4)$$

If the concept $c_i \in C^{IvC}$, the importance weight of the concept is computed by the weight of relations from the backward concepts to c_i:

$$r_{k+1}(c_i) = \sum_{c_j \in B_i} w_{k+1}(c_j, c_i) r_k(j) \qquad (5)$$

Definition 17 (Prior Matching Set (PMS)). The *Priority matching set* of concept *c* is the possible similarity set of *c* in ascending order of the distance between the importance weight of each concept belonging to the PSS and the importance weight of concept *c*.

Proposition 18. *Assuming two ontologies O_i and O_j must be integrated, the concept c_i belonging to ontology O_i should be priority matched to the concept c_j or neighbors of c_j belonging to PMP of c_i, where the distance between c_i's importance weight and c_j's is minimal.*

In this paper, the term *importance* is a measurement for priority matching between the concepts of the same type. The measurement is calculated based on following aspects: First, a concept is regarded as a source owning a set of relations with other concepts. For example, considering the concept *Teacher* of ontology E_1, it owns the relation *teach* to the concept *Subject*. Second, a concept has a set of attributes with associated restrictions. It is useful to identify the type of concepts. For example, the concept *Person* with the attribute of *hasFingerPrint* has associated retrictions: owl:DatatypeProperty, owl:FunctionalProperty, owl:InverseFunctionalProperty, and owl:cardinality = 1. So concept *Person* may be a DC. Finally, concepts with associated attributes and relations exhibit a mutually reinforcing relationship to identify the concept's importance weight (measurement). In our ongoing example, two ontologies E_1 and E_2 must be integrated. First, we classify all concepts belonging to each ontology into five disjoint groups, as shown in *Example* 8. Then we compute the importance weight of each concept based on formulas 3-5. We consider that if the set of $C^{PC}_{E_2} = \{FemalePerson, MalePerson\}$ and $C^{PC}_{E_1} = \{Female, Male\}$, they should be matched. According to the above measuring of the importance of concepts, it is clear that the distance between the $Female'$s importance weight and $FemalePerson'$s is less than the distance between $Male'$s and $FemalePerson'$s. This is because $FemalePerson'$s position in E_2 is more similar to $Female'$s than $Male'$s in ontology E_1. This is similar to the case of the two concepts $Male$ and $MalePerson$. So the priority matching between two sets of C_{E_1} and C_{E_2} is as follows: The two concepts $Male$ and $Female$ must be matched to two concepts $MalePerson$ and $MalePerson$ respectively.

4 Identity-based Similarity

According to the aforementioned previous studies, the techniques of similarity analysis that have been explored for ontology integration can be classified into following four groups:

- *Instance-based similarity*: The similarity between two concepts is determined via common instances.
- *Lexical-based similarity*: The similarity between two concepts is decided by analyzing the linguistic meanings of associated names.
- *Schema-based similarity*: The similarity between two concepts is found by analyzing the similarity between associated properties.
- *Taxonomy-based similarity*: The similarity between two concepts is determined by analyzing the structural relationships between them, such as subsumption.

In this section, we analyze another matching heuristic, called *identity-based similarity*. We consider Inha University library (InhaLib) and Wroclaw University library (WrocLib). We are assuming that the libraries have the same concept of $Book$ owning the unique identification of $BookID$. We refer to an instance of $Book$ called the *Artificial Intelligence* book:

$$\begin{cases} \text{In the InhaLib, the } \textit{Artificial Intelligence} \text{ has } \textit{BookID} \text{ of } \textit{IH-00012-AI}; \\ \text{In the WrocLib, the } \textit{Artificial Intelligence} \text{ has } \textit{BookID} \text{ of } \textit{WL-003334}. \end{cases}$$

Based on the above we know that the $BookID$ is a local identity in each library. This means that the $BookID$ value of the same instance *Artificial Intelligence* may possibly be different from the InhaLib to the WrocLib. However, while we refer to the instance as two concepts *Information (Paperback, Publisher, Language, ISBN, Product Dimensions, Shipping Weight, Average Customer Review)* belonging to the InhaLib and *PublishedInf(NumberPage, PublisherName, Language, ISBN, Dimensions, Weight)* belonging to the WrocLib, they represent the super-concept of $Book$, thus:

$$\begin{cases} \text{In the InhaLib, the } \textit{Artificial Intelligence} \text{ has } \textit{ISBN} \text{ of } \textit{978-0070522633}; \\ \text{In the WrocLib, the } \textit{Artificial Intelligence} \text{ has } \textit{ISBN} \text{ of } \textit{978-0070522633}. \end{cases}$$

Based on the above we can specify that the $ISBN$ is a key identity in each library. This means that the $ISBN$ value is identifiable in the real world.

While calculating the similarity between the two concepts *Information* and *PublishedInf*, although we compare all associated attributes, the degree of similarity between them is low and the number of comparisons is high (7 x 6= 42). This is because they have many different attribute names and the number of properties belonging to each concept is high. According to the characteristics of the identities of concepts, instead of blind or exhaustive comparisons among all properties of concepts, we simply focus on those properties that identify the concepts. For example, we only compare the associated key identity of $ISBN$ while computing the similarity between two concepts

$Information$ and $PublishedInf$. Their degree of similarity is high (one), the number of comparisons is one and is independent of the number of properties belonging to each concept. Similarly, for the concept $Book$ case, we simply compare associated identities $BookID$ and $ISBN$.

Thus we have a heuristic matching method that differs from previous studies as follows:

Proposition 19 Identity-based Similarity. *For two given concepts ($z_{c_1}, A^{c_1}, V^{c_1}$) belong to ontology O_1, and ($z_{c_2}, A^{c_2}, V^{c_2}$) belong to ontology O_2.*

1. *For any two concepts $c_1, c_2 \in C_{DC}$, if c_1's key identity is equivalent to c_2's then c_1 is equivalent to c_2,*
2. *For any two concepts $c_1, c_2 \in C_{HC}$, if c_1's (Local & Inheritance) identities are equivalent to c_2's then c_1 is equivalent to c_2,*
3. *For any two concepts $c_1, c_2 \in C_{PvC}$ or $c_1, c_2 \in C_{PC}$, if c_1's inheritance identity is equivalent to c_2's and c_1's constant attribute is equivalent to c_2's then c_1 is equivalent to c_2,*

5 Conflict in Ontology Integration

5.1 Conflicts on Instance-Level

On this level we assume that two ontologies differ only in the value of instances. This means that they may have the same concepts and relations.

Definition 20. Let O_1 and O_2 be (***A***, ***V***)-based ontologies. Let concept (z_c, A^c, V^c) belong to both ontologies and let the same instance *i* belong to concept *c* in each ontology, that is $(i, v_1) \in Ins(O_1, c)$ and $(i, v_2) \in Ins(O_2, c)$. There is a conflict if $v_1 \neq v_2$.

For solving conflicts of ontologies on the instance level, consensus methods seem to be very useful. Different criteria, structures of data and algorithms have been determined [Nguyen 2008]. For this kind of conflict, the consensus problem can be defined:
Given a set of values $X = \{v_1, \ldots, v_n\}$ where v_i is a tuple of type A^c, that is:

$$v_i : A^c \rightarrow V^c \tag{6}$$

for $i = 1, \ldots, n$; $A^c \subseteq \boldsymbol{A}$ and $V = \bigcup_{a \in A^c} V_a$ we must find the tuple v of type *A*, such that one or more selected postulates for consensus are satisfied [Nguyen 2008].

One popular postulate requires minimizing the following sum.

$$\sum_{i=1}^{n} d(v, v_i) = \min_{v' \in T(A^c)} \sum_{i=1}^{n} (v', v_i) \tag{7}$$

where $T(A^c)$ is the set of all tuples of type A^c.

5.2 Conflict on Concept-Level

On this level, Nguyen [Nguyen 2008] assumes that two ontologies differ in the structure of the same concept. This means that they contain the same concept but its structure is different in different ontologies. The definition of a concept on the concept-level as follows:

Definition 21. Let O_1 and O_2 be (***A, V***)-based ontologies. Let concept $(z_{c_1}, A^{c_1}, V^{c_1})$ belong to O_1 and concept $(z_{c_2}, A^{c_2}, V^{c_2})$ belong to O_2. There is a conflict on the concept level if $c_1 = c_2$ but $A^{c_1} \neq A^{c_2}$ or $V^{c_1} \neq V^{c_2}$.

Here we have considered the conflict on the concept level in the following cases:

1. *Multiple forms of the same concept* means ontologies define the same concept in different ways. For example, concept *Person* in one ontology may be defined by attributes: *Name, Age, Address, Sex, Job*, while in another ontology it is defined by attributes: *Id, Name, Address,* $Date\_of\_birth$, *Taxpayer identification number, Occupation*.

   The problem: *For given, a set of pairs* $X = \{(A^i, V^i) : (A^i, V^i)$ *is the structure of concept* c *belonging to the ontology* O_i for $i = 1, \ldots, n\}$, *we need to determine the pair* (A^*, V^*) *which best represents the given pairs.*

   Algorithm 1 is an overview of the solution.

2. *Overlapping but different concepts* means ontologies define different concepts in the same name and structure. As an example, we consider two concepts with the same name and structure which both contain knowledge about *student*. First the concept structures its knowledge of *female student*, whereas the second structures its knowledge of *male student*.

   The problem: *For given a set of concepts whose names and structures are equivalent* $T = \{c_1, \ldots c_n\}$ *where* c_i *in ontology* O_i*, we need to determine the set tuple* $C^* = \{(c_i\ r\ c_j)$*: the concept* c_i *is relation* r *to* c_j *where* r *is one of the relations: more general, equivalent, and disjoint*$\}$.

   Algorithm 2 is an overall of the solution.

Apart from the aforementioned conflict problems, we also show three other cases of conflicts on the concept-level as follows:

- The *same concept but different names* means that ontologies define the same concept in different names. For example, while defining concept *Person*, one ontology defines the concept name of *Individual* but another defines the concept name of *Homo*.

- The *same name but different concepts* means that ontologies define different concepts in the same name. For example, while using the same name of *Master Course*, one ontology considers the concept of subjects for a master student, while another ontology considers the concept of a master student.

input : $C^* = \bigcup c^i, i = 1.\ldots.n$ is the set of concepts that are recognized as the same concept, but the associated structures $\{(A^i, V^i), i = 1\ldots n\}$ are different.
output: Pair (A^*, V^*) which is the integration of the given pairs $\{(A^i, V^i), i = 1\ldots n\}$.

$A^* = \bigcup A^i, i = 1.\ldots.n$ where A^i is the set of attributes of the concept $c^i \in C^*$;
foreach *each pair* $a_1, a_2 \in A^*$ **do**
    if $R(a_1, \Leftrightarrow, a_2)$ **then** $A^* \setminus \{a_2\}$; `/* eg.,` $job \Leftrightarrow occupation$ `*/`
    if $R(a_1, \sqsubseteq, a_2)$ **then** $A^* \setminus \{a_1\}$; `/* eg.,` $age \sqsubseteq birthday$ `*/`
    if $R(a_1, \sqsupseteq, a_2)$ **then** $A^* \setminus \{a_2\}$; `/* eg.,` $sex \sqsupseteq female$ `*/`
    if $R(a_1, \perp, a_2)$ **then** $A^* \setminus \{a_1\}$; `/* eg.,` $single \perp married$ `*/`
end
foreach *attribute a from set* A^* **do**
    if *the number of occurrences of a in pairs* (A^i, V^i) *is smaller than n/2* **then** set $A^* := A^* \setminus \{a\}$;
end
foreach *each attribute a from set* A^* **do**
    determine its domain V_a as the sum of its domains in pairs (A^i, V^i);
end
Return((A^*, V^*));

Algorithm 1: Multiple forms of the same concept

input : $T = \bigcup c_i, i = 1.\ldots.n$ is the set of concepts whose names and structures are equivalent.
output: $C^* = \{(c_i r c_j)$: the concept c_i is relation r to c_j where r is one of the relations: more *general, equivalent,* and *disjoint*$\}$

foreach *pair* $c_i, c_j \in T$ **do**
    if $\exists\ a \in A^{c_i} \cap A^{c_j}$ *and* $z_{c_i}(a) \perp z_{c_j}(a)$ **then** $C^* = \bigcup\{(c_i \perp c_j)\}$;
    else if $\exists\ a \in A^{c_i} \cap A^{c_j}$ *and* $z_{c_i}(a) \sqsupseteq z_{c_j}(a)$ **then**
        $C^* = \bigcup\{(c_i \sqsupseteq c_j)\}$;
        else if $\exists\ a \in A^{c_i} \cap A^{c_j}$ *and* $z_{c_i}(a) \sqsubseteq z_{c_j}(a)$ **then**
            $C^* = \bigcup\{(c_j \sqsupseteq c_i)\}$;
            else if $\exists$ *c and* $c' \in C \cap C'$ *where c and* c' *are a super-concept corresponding to* c_1 *and* c_2*, and* $c \perp c'$ **then**
                $C^* = \bigcup\{(c_i \perp c_j)\}$;
                else
                    $C^* = \bigcup\{(c_i \Leftrightarrow c_j)\}$;
end
return (C^*);

Algorithm 2: Overlapping but different concepts

- The *multiple concepts of the same form* means that ontologies define different concepts in the same structure.

The first two problems can be solved by combining the text corpus and WordNet-based method, which is presented in [Duong et al., 2008a]. The third problem can be solved by using the same multiple concepts for the same form method.

6 Experiments

6.1 Implementations

In this section, we apply the aforementioned methods to design an effective algorithm for ontology matching. An overview of the algorithm as follows:

```
input  : Given two ontologies O1 and O2
output: Pairs of concepts are equvalent
dJ ⟵ clfConcepts(O2);
wC ⟵ wConcepts(dJ);
foreach each concept c belonging to ontology O1 do
    PSS ⟵ getPSS(c, dJ);
    PMS ⟵ getPMS(c,wC, PSS);
    for i ← 1 to size(PMS) do
        match ← Matching (c,PMS [i]);
        if match ≥ threshold then
            smatch ⟵ ∪ (c, PMS [i]);
            break;
        end
    end
end
cConflict(smatch);
Return(smatch);
```

Algorithm 3: Direct Matching Algorithm (DMA)

1. Function clfConcepts(O_2) is used to classify all concepts belonging to the ontology O_2 into disjoint groups (dJ). We apply the knowledge from section 3.1 to clfConcepts(O_2). The main problem here is how to identify the concept's identities. We assume the ontologies were written in the OWL language. Identities are distinguished via the following two steps:
   - Firstly, collecting the necessary and sufficient properties of each concept.
   - Secondly, considering an identity as a property of the concept and distinguishing it from other properties by the characteristics of the one-to-one function be-

tween its domain and range. The identities can be written in OWL via owl:DatatypeProperty with three restrictions: owl:FunctionalProperty, owl:InverseFunctionalProperty, and owl:cardinality = 1.

Notice that we use the following heuristic to distinguish a DC: If a concept is a top-most taxonomy in a given ontology and it contains at least one identity, it must be a DC.

2. Function wConcepts(dJ) is used to compute importance weights of each concept belonging to each group in the set dJ. It returns an importance weight of vectors (wC) set corresponding to dJ. The algorithm is similar to [Gang et al. 2008].

3. Function getPSS(c, dJ)is used to generate a set of possible similarities of concept c from dJ.

4. Function getPMS(c, wC, PSS) is used to obtain a set of priority matching of concept c from its PSS.

5. Function Matching(c, $PMS[i]$) is used to find the degree of similarity between the concept c and concept $PMS[i]$. Here we apply the *content-based similarity* and *identity-based similarity* techniques.

6. Function cConflict is used for the problem of conflict in ontology integration which was mentioned in the above sections, to solve the problem of conflict on the concept-level.

6.2 Results

In Table 1, we compare the techniques for similarity analysis of existing mapping tools with our approach in the novel *Identity-based similarity*. Note that to find similarities between concepts, most existing mapping methods compare all properties belonging to each concept, while the *identity-based similarity* method simply focuses on those identities belonging to each concept.

Table 1: Comparative techniques of similarity analysis

Matching Methods	*Instance -based*	*Lexical -based*	*Schema -based*	*Taxonomy -based*	*Identity -based*
PROMPT	Y	Y	Y	Y	N
MAFRA	Y	Y	Y	Y	N
RiMOM	Y	Y	Y	Y	N
GLUE	Y	Y	Y	Y	N
Our	Y	Y	Y	Y	Y

According to our studies of ontology integration, methods for reducing the complexity of ontology integration have not yet been explored. Therefore we simply present the comparative complexity between our method of matching equality and the *content-based matching* approach as follows: Suppose that N_c, N_p, and $N_i$are the maximum numbers of nodes, properties (attributes), and instances. Let us assume that the complexity of comparing two attribute values between two instances is O(1). Then, the complexity of calculating the similarity between two instances is O(N_p^2). The complexity of calculating the similarity between two nodes is O(N_p^2 x N_i). Finally, matching between two ontologies costs O(N_c x N_p^2 x N_i). In order to compare the content-based method with our matching method, we substitute N for every parameter; then the cost of the content-based method is O(N^4), using the direct matching algorithm for the content-based method, *DMAContent-based*, our matching method costs O(N^3 x $logN$), because the method involves direct matching between concepts of the same type. If we apply *identity-based similarity* for matching between concepts of the same type, the cost is O(N^2 x $logN$), because it does not require comparing all properties belonging to each concept. $Figure$ 3 illustrates the complexity difference between our methods of matching and content-based matching in a line chart. The chart shows that the complexity differs, especially according to the number of properties, assuming that the number of concepts and instances are equal in each case. Whenever the number of properties belonging to concepts increases, the complexity difference increases proportionally.

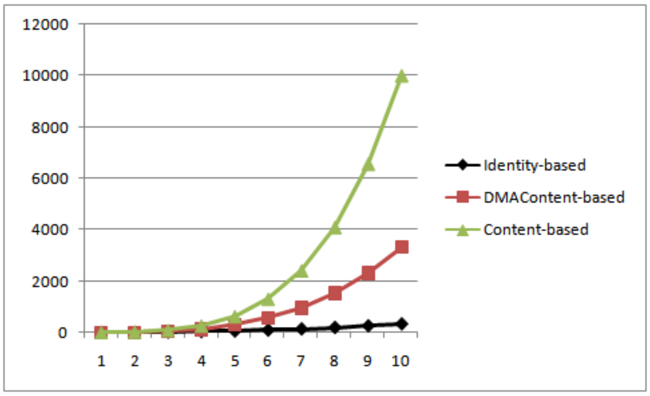

Figure 3: Complexity comparison between Identity-based and Content-based method.

Moreover, our method avoids many cases of conflict between concepts. $Figure$ 4 shows the top-level view of source ontologies in PROMPT [Noy et al. 2003]. While most of the above matching methods produce incorrect matching from O_1:BS, O_1:MS, O_1:PhD and O_1:$Student$ to O_2:BS, O_2:MS, O_2:PhD and O_2:$Student$, respectively. Our approach recognizes that these are cases of conflict between the labels and the contents of concepts.

We collected many ontologies from Internet (http://www.aifb.uni-karlsruhe.de/WBS/meh/foam/ontologies.htm) and composed additional ontologies which corresponded to them. We also modified the ontologies to create cases of conflict. Each sample in-

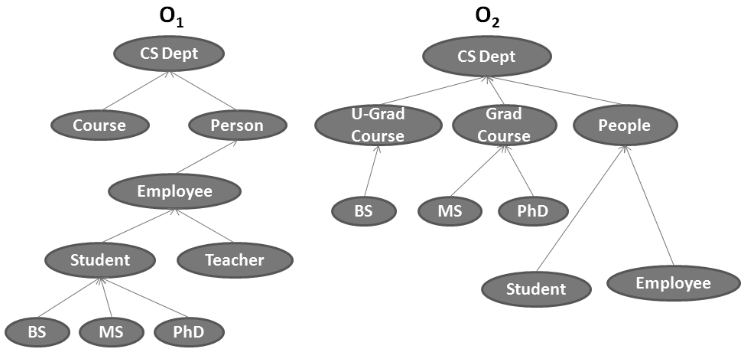

Figure 4: Possible conflict on concept-level.

cluded at least three ontologies. N_{total} denotes the total number of pairs for matching concepts between the candidate ontologies by experts, $N_{correct}$ and $N_{incorrect}$ correspond to the number of correct pairs for matching concepts and the number of incorrect pairs for matching concepts sought by our system, respectively. *Precision* $(= \frac{N_{correct}}{N_{correct}+N_{incorrect}})$ is used to evaluate the ratio of incorrectly extracted relationships. *Recall* $(= \frac{N_{correct}}{N_{total}})$ is used to evaluate the ratio of correct matching sought total by the system.

Figure 5 illustrates comparative experimental results between *Identity-based* and *Content- based* and *DMSContent-based* methods.

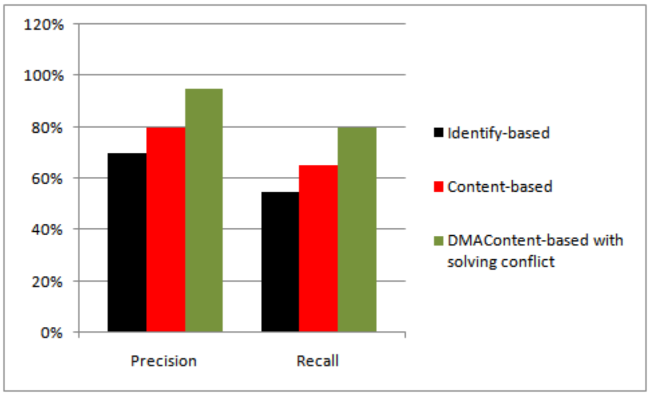

Figure 5: Comparison of our system and content-based system.

7 Conclusion

The direct matching algorithm is a smart approach for ontology integration. It combines the *types of concepts* and *importance of concepts* to directly match concepts of the same type, instead of blind or exhaustive matching among all concepts. The *Identity-based similarity* is a novel method of heuristic matching. This has the advantage that the complexity is initially reduced by comparing only those properties which identify each concept, instead of matching all properties belonging to each concept. The proposed solutions to the problem of conflicts prevent many cases of conflicts in ontology integration. In future work, we will meticulously explore the idea of complexity in ontology integration.

References

[Bagui 07] Bagui, S.: "Mapping XML Schema to Entity Relationship and Extended Entity Relationship Models"; International Journal of Intelligent Information and Database Systems, 1, 3 (2007), 325-345.

[Castano et al. 06] Castano, S., Ferrara, A., Montanelli, S.: "Matching ontologies in open networked systems: Techniques and applications"; Journal on Data Semantics, 3870 (2006), 2563.

[Do and Rahm 02] Do, H.H., Rahm, E.: "COMA a system for flexible combination of schema matching approaches"; In Proc. 28th International Conference on Very Large Data Bases (VLDB), Hong Kong (2002), 610621.

[Dou et al. 05] Dou, D., McDermott, D., Qi, P.: "Ontology translation on the semantic web"; Journal on Data Semantics, 3360 (2005), 3557.

[Doan et al. 01] Doan, A. H., Domingos, P., Halevy, A.: "Reconciling Schemas of Disparate Data Sources: a machine learning approach"; In Proc. of ACM SIGMOD Conference, 509-520.

[Doan et al. 02] Doan, A. H, Madhavan, J., Domingos, P., Halevy, A.: "Learning To Map between Ontologies on the Semantic Web"; In Proc. of WWW 2002, ACM, 662-673.

[Doan et al. 03] Doan, A. H., Domingos, P., Halevy, A.: "Learning to match the schemas of data sources: A multistrategy approach"; Machine Learning, 50, 3 (2003), 279301.

[Doan et al. (04)] Doan, A. H., Madhavan, J., Domingos, P., Halevy, A.:"Ontollogy matching: a machine learning approach"; Springer Verlag, Berlin (DE), 385404.

[Duong et al. 08a] Duong T. H., Nguyen N. T., Jo, G. S.:"A Method for Integration across Text Corpus and WordNet-based Ontologies"; In: IEEE/ACM/WI/IAT 2008 Workshops Proceedings,IEEE Computer Society (2008), 1-4.

[Duong et al. 08b] Duong T. H., Nguyen N. T., Jo, G. S.:"A Method for Integration of WordNet-based Ontologies Using Distance Measures"; Proceedings of KES 2008, Lecture Notes in Artificial Intelligence 5177 (2008), 210-219.

[Duong et al. 09] Duong T. H., Nguyen N. T., Jo, G. S.:"A Method for Integrating Multiple Ontologies"; Cybernetics and Systems 40, 2 (2009), 123-145.

[Ehrig et al. 04] Ehrig, M., Sure, Y.:"Ontology mapping - an integrated approach"; Proceedings of First European Semantic Web Symposium (ESWS), Lecture Notes in Computer Science, 3053 (2004), 76-91.

[Gang et al. 08] Gang W., Juanzi L., Ling F.,Kehong W.:"Identifying Potentially Important Concepts and Relations in an Ontology"; In Proceedings of International Semantic Web Conference'2008, Lecture Notes in Computer Science, 5318 (2008), 33-49.

[Gangemi et al. (98)] Gangemi, A., Pisanelli, D. M., Steve, G:"Ontology Integration: Experiences with Medical Terminologies"; Formal Ontology in Information Systems, IOS Press, 163-178.

[Guarino et al. 00] Guarino, N., Welty, C.:"Ontological Analysis of Taxonomic Relationships"; In Proceedings of ER-2000, The 19th International Conference on Conceptual Modeling, Springer-Verlag, 1920 (2000), 210-224.

[Lee et al. 06] Lee, J., Chae, H., Kim, K., Kim, C.H.:"An Ontology Architecture for Integration of Ontologies"; Processing The Semantic Web - ASWC, Lecture Notes in Computer Science, 4185 (2006), 205-211.

[Madhavan et al. 01] Madhavan, J., Bernstein, P., Rahm, E.:"Generic schema matching with Cupid"; In Proc. 27th International Conference on Very Large Data Bases (VLDB), Morgan Kaufmann Publishers Inc, San Francisco (2001), 4858.

[Madhavan et al. 05] Madhavan, J., Bernstein, P., Doan, A., Halevy, A.:"Corpus-based schema matching"; In Proc. 21st International Conference on Data Engineering (ICDE), IEEE Computer Society Washington (2005), 5768.

[Maedche et al. 02] Maedche, A., Moltik, B., Silva, N.,Volz, R.:"MAFRA-An ontology MApping FRAmework in the context of the semantic web"; In:Proceedings of the EKAW 2002, Siguenza, Spain (2002), 235-250.

[Nguyen (08)] Nguyen, N.T.:"Advanced Methods for Inconsistent Knowledge Management"; Springer-Verlag, London (2008).

[Noy et al. 03] Noy, N. F., and Musen, M. A.:"The PROMPT Suite: Interactive Tools For Ontology Merging And Mapping"; In International Journal of Human-Computer Studies, 59 (2003), 983-1024.

[Pinto et al. 01] Pinto, H.S., and Martins, J.P.:"A Methodology for Ontology Integration"; In: Proceedings of the First International Conference on Knowledge Capture, ACM Press, 131-138.

[Tang et al. 06] Tang, J., Li, J., Liang, B., Huang, X., Li, Y., and Wang, K.:"Using Bayesian decision for ontology mapping"; Journal of Web Semantics, 4, 4 (2006), 243262.

[Tozicka et al. 08] Tozicka, J., Rovatsos, M., Pechoucek, M., Urban, S.:"MALEF: Framework for distributed machine learning and data mining"; International Journal of Intelligent Information and Database Systems, 2, 1 (2008), 6-24.

[Zhang 08] Zhang, W. R.:"Concepts, challenges, and prospects on Multiagent Data Warehousing (MADWH) and Multiagent Data Mining (MADM)"; International Journal of Intelligent Information and Database Systems, 2, 1 (2008), 106-124.