NOTE TO USERS

This reproduction is the best copy available.

A Multi-Matching Technique for Combining Similarity Measures in Ontology Integration

Ahmed Khalifa Alasoud

A Thesis

In the department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

For the Degree of Doctor of Philosophy (Computer Science)

Concordia University

Montréal, Québec, Canada

February 2009

Your file Votre référence
ISBN: 978-0-494-63399-1
Our file Notre référence
ISBN: 978-0-494-63399-1

ABSTRACT

A Multi-matching Technique for Combining Similarity Measures in Ontology Integration

Ahmed Khalifa Alasoud, Ph.D.

Concordia University, 2009

Ontology matching is a challenging problem in many applications, and is a major issue for interoperability in information systems. It aims to find semantic correspondences between a pair of input ontologies, which remains a labor intensive and expensive task.

This thesis investigates the problem of ontology matching in both theoretical and practical aspects and proposes a solution methodology, called *multi-matching*. The methodology is validated using standard benchmark data and its performance is compared with available matching tools.

The proposed methodology provides a framework for users to apply different individual matching techniques. It then proceeds with searching and combining the match results to provide a desired match result in reasonable time.

In addition to existing applications for ontology matching such as ontology engineering, ontology integration, and exploiting the semantic web, the thesis proposes a

new approach for ontology integration as a backbone application for the proposed matching techniques.

In terms of theoretical contributions, we introduce new search strategies and propose a structure similarity measure to match structures of ontologies. In terms of practical contribution, we developed a research prototype, called MLMAR – Multi-Level Matching Algorithm with Recommendation analysis technique, which implements the proposed multi-level matching technique, and applies heuristics as optimization techniques. Experimental results show practical merits and usefulness of MLMAR.

ACKNOWLEDGMENTS

First of all, I would like to thank my supervisors, Dr. Volker Haarslev and Dr. Nematollaah Shiri, for their guidance, insightful discussion, encouragement, and support throughout the development of this thesis.

I would like to express my deepest gratitude for the constant support, understanding and love that I received from my parents, my wife Hasna, my sisters, my uncle Ali's family, and my brothers during the past years.

I would like to express my gratitude to my fellow students in Database research lab, with whom we spent countless time in the lab in the past few years.

I would like to dedicate this thesis to the soul of my grandmother Fodha Abd Alsaid.

Finally, I would like to thank all the people who assisted me in completing this work. This work is as much theirs as it is mine. Thank you all.

Contents

List of Figures

List of Tables

1. Introduction

The proliferation of information on the World Wide Web (WWW) has made it necessary to make all this information not only available to people, but also to machines. Ontologies are widely being used to enrich the semantics of the web, and the corresponding technology is being developed to take advantage of them. An ontology is defined as "a *formal*, *explicit specification* of a *shared conceptualization*" [Gruber, 1993], where *formal* refers to the meaning of the specification which is encoded in a logic-based language, *explicit* means concepts, properties, and axioms are explicitly defined, *shared* indicates that the specification is machine readable, and *conceptualization* models how people think about things of a particular subject area.

Ontologies are likely to be everywhere, and constitute the core of many emerging applications in database integration, peer-to-peer systems, e-commerce, semantic web services, and social networks [Fensel, 2004]. With the infrastructure of the semantics web, we witness a continuous growth in both the number and size of available ontologies developed to annotate knowledge on the web through semantics markups to facilitate

sharing and reuse by machines. This, on the other hand, has resulted in an increased heterogeneity in the available information as different parties adopt different ontologies. The ontologies are developed with different purpose in mind, therefore we end up with different ways entities are modeled. For example, the same entity could be given different names in different ontologies or it could be modeled or described in different ways. The Ontology Matching Problem (OMP) attempts to find similar entities in different ontologies, described as follows: given ontologies O_1 and O_2, each of which describes a collection of discrete entities, such as classes, properties, individuals, etc., we want to find the semantic correspondences that exist between the components of these entities.

This problem has been the subject of numerous studies, and a number of solution techniques have been proposed. These matching techniques are often domain-dependent, as they are mainly based on a single similarity measure, such as names, structures, logic satisfiability, etc. This makes them useful and efficient in specific domains. For example, matching techniques which are based on syntactic similarity provide good results in domains where there is a high probability that whenever the matched entities agree on their syntax, they also agree on their semantics. However, such techniques based solely on name similarity might not work well in application domains where similar entity names are used with different meanings. Consequently, some researchers consider using a number of matching techniques, and then aggregating the results of individual matching methods in order to compute the final matching result.

The matcher composition systems (matching systems that use more than one similarity technique) are not clear about the suitability of their reused matching techniques for different kinds of matching domains. It is therefore difficult for a regular

user to decide, among the vast number of matching techniques, which one is preferred for matching the given ontologies. Consequently, the choice of the user might affect the matching process in both time and quality.

Example 1. Through this example we illustrate the main ideas of the technique proposed in this thesis. Fig.1 shows two sample taxonomies "subsumption relationships between the concepts" for two person ontologies O_1 and O_2. For ease of presentation, we use two very simple and small taxonomies.

To reduce the manual work involved, we use a matching algorithm to identify the matching entities. As can be seen in Fig.1, entities S_1, S_2, S_3, and T_1, T_2, T_3 are *concepts*, which are high-level entities in the input ontologies. The goal is to find the corresponding matches among the entities in the two input ontologies.

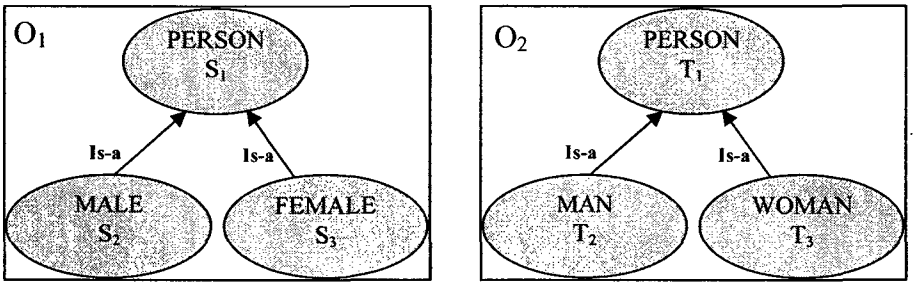


Figure 1: Person ontologies

There exist many methods to measure similarities between two entities, such as string similarity, linguistic similarity, etc. However, when we use a single matching measure for an input pair of ontologies, we may not be satisfied with the final match result. For instance, if we use a string similarity measure only, the concepts *MALE* and *FEMALE* in O_1 have no matches in O_2. On the other hand, a string similarity measure is the basis for some other methods of measuring similarities between entities, and it works well in some domains where a match in the entities on their syntax would most probably mean agreement on their semantics.

However, we can use a stronger semantic measure, such as a linguistic-based measure. For instance, we find out that the concept *PERSON* in O_1 is mapped to concepts *PERSON, MAN* and *WOMAN* in O_2. So, by recommending many matches to validate, this will not help the user to focus on his/her intention. As a result, if we use both measures (string and linguistic), the concept *PERSON* in O_1 will be mapped to the concept *PERSON* in O_2 with a very high confidence, concept *MALE* in O_1 will be mapped to *MAN* in O_2, and concept *FEMALE* in O_1 will be mapped to *WOMAN* in O_2.

Furthermore, for matcher composition systems, using a recommended subset among their similarity measures list should improve the final matching results in terms of time and quality.

Moreover, recommendation techniques improve the overall running time as it is unnecessary to reuse and combine all their underlying similarity measuring methods, instead, using only a recommended subset should decrease the average running times. Furthermore, the reason that recommendation techniques can enhance the matching quality is that they exclude the unpractical similarity matching methods to be used for a task at hand. For instance, if there is no string, linguistic, or structure similarity between a given input pair of ontologies, then including, combining, and aggregating the matching results retrieved by a string, linguistic, or structure similarity measuring method should negatively affect the overall quality of the matching result.

We studied the ontology matching problem and introduced a new method that uses a multi-match search technique together with our flexible similarity measure and a framework for analyzing the reused similarity measure techniques to obtain the best

possible matching results. A main characteristic of our technique is that it combines the matching techniques to provide a solution to a given ontology matching problem.

1.1 Motivations for ontology matching

Ontology matching is considered to be a prerequisite for many real-life applications. In this section, we describe such applications illustrating the need for and use of ontology matching.

1.1.1 Ontology engineering

In general, ontology engineering refers to activities where users design, implement and maintain ontology-based applications, for which they apply ontology matching algorithms to find similarities in multiple ontologies. For instance, suppose we want to build an ontology about tourism in Montreal that contains relevant information about transportation, hotels, restaurants, etc. One way to do this is to construct this ontology from scratch. In this case we do not make use of any existing ontologies, if there are any. This method requires a lot of effort. This problem is further aggravated by the fact that ontologies are normally huge and complex. A better approach is to *reuse* available ontologies on the topics, such as transportation, restaurants, and hotels in Montreal, to build the desired ontology. These ontologies may share some entities and consequently, the ontology engineers require support for identifying the relevant ontologies and matching their entities. Another scenario where ontology matching is crucial is in the presence of multiple versions of the same ontology. For example, some users keep updating their ontologies, which often leads to having more than one version of the same

ontology. In such cases, ontology matching helps identify what entities have been changed (added, deleted, or renamed) from on version of an ontology to another [Noy and Klein, 2004, Noy and Musen, 2004, Noy and Musen, 2002, Roddick, 1995].

1.1.2 Web navigation

The matching process is important for navigating the semantic web. An example is the browser Magpie [Dzbor *et al.*, 2004, Dzbor *et al.*, 2003], which extends Internet Explorer by annotating web pages. In such scenarios, the matching operation is needed to help match the terms in web pages and the corresponding terms in on-line ontologies.

1.1.3 Peer-to-peer information sharing

Peer-to-Peer (P2P) is a distributed communication model in which parties (also called peers) have equivalent functional capabilities in providing each other with data and services [Zaihrayeu, 2006]. Currently, there are several P2P file sharing systems, such as Kazaa, Edonkey, BitTorrent, and Semantic P2P [Staab and Stuckenschmidt, 2006]. In order to establish exchanging and sharing information between different peers in such applications, a matching operation is necessary to identify correspondences in terminologies used by different peers.

1.1.4 Information Integration

Matching is also important in the context of information integration. There are different problems of information integration, such as schema integration [Batini *et al.*, 1986, Parent and Spaccapietra, 1998, Sheth and Larson, 1990, Spaccapietra and Parent, 1991],

data warehousing [Bernstein and Rahim, 2000], data integration [Chawathe *et al.*, 1994, Draper *et al.*, 2001, Halevy *et al.*, 2005, Wache *et al.*, 2001], and catalogue integration [Agrawal and Srikant, 2001, Bouquet *et al.*, 2003b, Giunchiglia *et al.*, 2005, Ichise *et al.*, 2003].

Generally, providing single portal of access to resources implies a need for integrated ontology, companies merge implies need for ontology integration, etc. Information integration is an abstraction which provides and uses an integrated view. Suppose we have a company that has branches, dealers, etc, distributed all over the world. The main branch needs to get some information from the other branches, such as customers, sellers, and some statistics about the employees, sales, etc. In this case, we can provide a unified view (or global ontology) in the main branch through which we can query the local ontologies in various branches using proper *mappings* and wrappers. All in all, the matching step in such scenarios is to relate the correspondences between the entities in both the global ontology and the local ontologies (source ontologies).

Ontology merging is another scenario where the matching operation is important. Suppose there are many ontologies on the same topic, such as medical, which may contain overlapping information. For example, we might want to build a new single ontology in a medical field which "unifies" the various concepts, terminologies, definitions, constraints, etc., from existing ontologies. For instance, among existing medical ontologies, we consider the Unified Medical Language System (UMLS) and the Galen COding REference (CORE) model. As a result of integrating these two ontologies, we obtain a new, single, unified ontology in the medical field. As another example of merging, consider a bottom-up construction of ontologies, which could be done by

merging the ontologies of several companies. For example, two car companies may merge to form a new larger company. Merging these companies may lead to merging their ontologies. As an initial step before merging ontologies, the related entities to be merged from different ontologies have to be identified, done through a matching step.

1.2 Contributions

We have made the following contributions:

1. Under the context of ontology integration in particular, we introduce an approach for ontology integration, which is a hybrid of materialized (data warehouse) and virtual views [Alasoud *et al.*, 2005].
2. In order to support the proposed approach with a matching strategy, we develop a multi-matching strategy which benefits from existing individual matching techniques and "combines" their match results to provide enhanced ontology matching results [Alasoud *et al.*, 2007].
3. We further extend the multi-matching strategy with a multi-level matching strategy, which assumes that there is a partial order on the collection of measures defined by the user [Alasoud *et al.*, 2007].
4. We devise a structure similarity measure to be used for matching the structure of the ontologies based on the adoption of the Dice coefficient [Alasoud *et al.*, 2007].
5. We propose using the neighbor search strategy to find the correspondences between entities in the given ontologies and to optimize the multi-matching strategy developed [Alasoud *et al.*, 2008].

6. We propose a recommendation analysis of ontology matching techniques. The users often have little knowledge about the suitability of matching strategies for a given matching task. As a result, the quality of the matching end result and processing time will be affected by the user's choice. The main characteristics of the proposed work are (1) assisting the user to choose the appropriate matching technique(s) for a given matching task, (2) inferring a hidden structure relationship between the entities of the input ontologies and consequently making the structure-based similarity measure more precise, and (3) improving the average matching process time considerably, as shown in our experimental evaluations [Alasoud *et al.*, 2009].

1.3 Thesis organization

The remainder of the thesis is organized as follows. Chapter 2 provides the background knowledge and reviews related work. Chapter 3 presents the hybrid approach for ontology integration. Chapter 4 describes the multi-matching strategy. The multi-level and reasoning-based neighbor search matching strategies are introduced in Chapter 5, followed by a performance evaluation of the proposed framework in terms of quality and processing time. The conclusion and future work are discussed in Chapter 6.

2. Background and Related work

This chapter provides a background for our work and reviews related work. Section 1 gives an overview of description logics, and Section 2 introduces ontologies. Then, Section 3 discusses approaches to ontology integration. We study these approaches as to present our novel approach in the next chapter. Section 4 classifies techniques that can be used for solving the ontology matching problem, and basic techniques used to find similarities between the entities of two ontologies. Finally, Section 5 reviews available ontology matching systems.

2.1 An Overview of Description Logics

Description Logics (DLs) refer to a family of knowledge representation languages that are capable of encapsulating the main characteristics of many class-based representation formalisms in Artificial Intelligence. Lately, DLs are becoming a standard for the semantic web, specifically the Web Ontology Language with its correspondence to description logics (OWL-DL). An advantage of these logics is that they are equipped

with powerful reasoning algorithms, and practical systems, such as RACER [Haarslev and Moeller, 2001a, Haarslev and Moeller, 2001b], that implement such algorithms.

In this section, we briefly describe the families of DL Languages and their main constructs. Also, we will show how they differ in these constructs.

We begin by explaining the constructs of the attributive language (AL) presented in [Baader *et al.*, 2005]. We use the letters A and B for atomic concepts, R for atomic roles, and C and D for concept descriptions. The concept descriptions in AL are defined according to the following syntax:

$C, D \rightarrow A$	|	(atomic concept)
T	|	(top/universal concept)
$\perp$	|	(bottom/null concept)
$\neg A$	|	(atomic negation)
$C \cap D$	|	(intersection)
$\forall R.C$	|	(value restriction)
$\exists R.\mathrm{T}$	|	(limited existential quantification).

Note that in AL, negation can only be applied to atomic concepts, and only the top concept is allowed in the scope of an existential quantification over a role. As examples of expressions in AL, assume that Product and PC are atomic concepts. Then Product$\cap$ PC and Product$\cap$ $\neg$ PC are AL concepts denoting those Products that are PC, and those that are not PC, respectively. Furthermore, assume that hasMaker is an atomic

role. Then we can form the concepts Product $\cap$ $\exists$ hasMaker.T and Product $\cap$ $\forall$ hasMaker.A, denoting products that have a maker and all products produced by A, respectively.

We now move to more expressive languages by adding new constructs, such as the *union* of concepts ($C \cup D$), indicated by the letter U, *full existential quantification* ($\exists R.C$) indicated by ε, *number restrictions* (at least restriction) $\geq nR$ indicated by the letter N, and *number restrictions* (at most restriction) as $\leq nR$, where n is a positive integer, and *negation* of arbitrary concepts ($\neg D$) indicated by the letter C (for "complement"). We name each AL-language by a string of the form AL [U] [ε] [N] [C], where a letter in the name stands for the presence of the corresponding construct. For the semantics of the AL language and its family members, see Appendix A. By adding more expressive concept constructs, as well as role constructs, we define more expressive DLs. A notable example for an expressive DL is ALCQI, which provides concept constructs for complement, intersection, union, existential restriction, universal quantification, qualified number restrictions (indicated by the letter Q), and a construct for inverse roles, indicated by the letter I.

2.2 Ontologies

Ontologies aim at capturing static domain knowledge in a generic way and provide a commonly agreed upon understanding of that domain, which may be reused and shared across applications and groups. Therefore, one can define an ontology as a shared specification of a conceptualization [Gruber, 1993]. An ontology contains terms, the

definitions of these terms, and specifications of multiple, rich relationships among these terms. Consider the computer ontology example shown in Fig. 2.

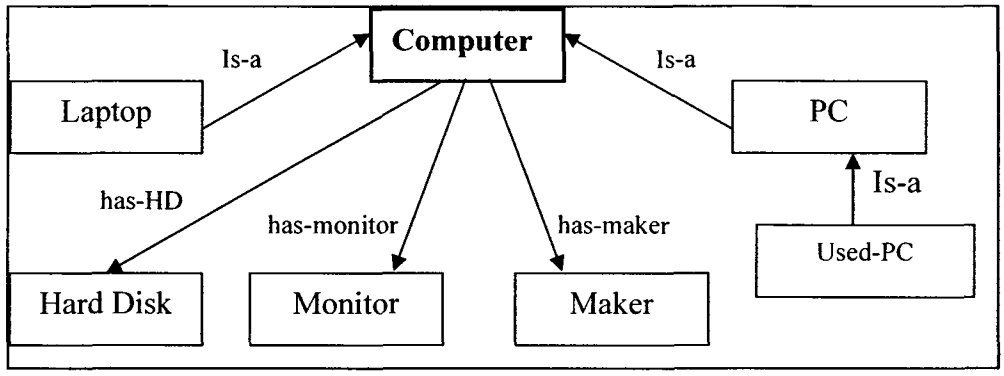


Figure 2: Computer Ontology

The main components of an ontology

1. *Classes or concepts*

   These are concepts of the domain or task, usually organized in taxonomies. In our ontology example, Computer, PC, Laptop, Hard Disk, etc. are examples of classes, shown as rectangles in the Fig. 2.

2. *Roles or properties*

   Role is a type of interaction between instances of concepts in the domain. For example, has-HD, has-monitor, and has-maker are roles, shown as links in the figure. Furthermore, roles can have the following characteristics:

   - Transitivity : $P(x,y) \wedge P(y,z) \Rightarrow P(x,z)$

     For example, consider a transitive role (has-part). We can define (PC has-part motherboard) and (motherboard has-part RAM). Then, we can conclude from the definition of the transitive role (has-part) that (PC has-part RAM).

- Symmetry : $P(x,y) \Leftrightarrow P(y,x)$

  For example, consider a symmetric role (partner_with). So, given the definition (Enterprise-A partner_with Enterprise-B), we conclude that (Enterprise-B partner_with Enterprise-A).

- Functional : $P(x,y) \wedge P(x,z) \Rightarrow y = z$

  For instance, we might assume that the (has-maker) role is a functional role. This implies that computers have a unique maker.

- Inverse: $P(x,y) \Leftrightarrow Q(y,x)$, where P is the inverse of Q and vice-versa.

  We can define a role called (maker_of) as an inverse to the role (has_maker).

- Inverse functional : $P(x,y) \wedge P(z,y) \Rightarrow x = z$

  We can consider, for instance, that the role (has_maker) is the inverse functional of the role (maker_of). In other words, each maker can produce more than one computer, but for each computer there is a unique maker.

3. *Axioms*

   Axioms model sentences that are always true. For example, if the price of some PC is equal to 50% of the original PC, then we can conclude that it is a used PC.

4. *Individuals or instances*

   Individuals or instances represent specific elements. For example, Enterprise-A could be an instance of class Maker.

2.3 Approaches to Ontology Integration

In this section, we review the main approaches to ontology integration, including ontology reusing, merging, and mapping.

The term "ontology integration" designates the operations and the process of building ontologies from other ontologies, available in some ontology development environments. This involves following methodologies that specify how to build ontologies using other, publicly available, ontologies [Pinto, 1999].

Ontology integration is motivated by the following three factors. First, the use of multiple ontologies. For example, suppose we want to build an ontology about tourism in Montreal that contains information about transportation, hotels, restaurants, etc. We could construct this ontology from scratch. This requires a lot of effort, especially since ontologies are huge and complex. A more reasonable approach is to *reuse* available ontologies on the topics, such as transportation, restaurants, and hotels in Montreal, to build a desired "integrated" ontology.

The second motivation is the use of an integrated view. Suppose we have a company that has branches, dealers, etc, distributed around the world. The main branch needs information from the other, such as customers, sellers, and some statistics about the employees, sales, etc. In this case, we can query the ontologies at various branches through proper *mappings* and wrappers, thus providing a unified view in the main branch.

The third motivation for ontology integration is the merge of source ontologies. Suppose there are many ontologies on the same topic, such as medicine, covering different aspects of the field, which may contain overlapping information. We might want to build a new, single ontology about the medical field, which "unifies" the various concepts, terminologies, definitions, constraints, etc., from the existing ontologies. For instance, among many existing medical ontologies, we consider the Unified Medical Language System (UMLS) and the Galen COding REference (CORE) models. As a

result of integrating these two ontologies, we obtain a new, single, unified ontology in the medical field. As another example of merging, suppose several car companies are merged into a new car company, for which we want to construct an ontology. This could be done by merging the existing ontologies of these companies.

2.3.1 Ontology Reuse

The *use* of existing ontologies can be considered as a 'lower' level integration, because it does not modify the ontologies, but merely uses the existing concepts. Since the survey in [Pinto, 1999], there have been some developments in using/reusing ontologies, such as the On-To-Knowledge project [Fensel *et al.*, 2002]. This project resulted in a software toolkit for ontology development, maintenance, and (re)use. In [Stumme and Mädche, 2001b], they proposed to combine ontology reuse and merging, consisting of merging local (federated) ontologies at some stage. These "federated ontologies" are analogous to federated databases. Another interpretation of reusing existing ontologies, in conjunction with formal integration, is the architecture of Fisheries ontology [Gangemi *et al.*, 2002] by the Food and Agriculture Organization (FAO), found in Appendix B. We next explain the main ideas of the ontology reuse approach.

Ontology reuse attempts to make use of existing ontologies to build a new ontology, instead of building one from scratch [Pinto, 1999]. Fig. 3 illustrates ontologies O_1 and O_2, as well as the result of their integration by reuse, named O. It is important to note that the reused ontologies O_1 and O_2 are part of the resulting ontology O. Also note that, in this case, the resulting ontology can be seen as consisting of different ontologies.

In general, the domains of the reused ontologies O_1 and O_2 are different from the domain of the resulting "integrated" ontology (O), but there may be a relationship between the domains. When ontologies are integrated by the reuse approach, the concepts from O_1 or O_2 may be (1) used as they are (no change), (2) adapted (or modified), (3) specialized (leading to a more specific ontology on the same domain), or (4) augmented by new concepts (at the same level or by more general concepts).

The domains of different reused ontologies, such as transportation, hotels, and restaurants, may be different from each other; that is, each ontology O_i contributing to the integration has a domain D_i which is different from domain D of the resulting 'tourism' ontology O (Fig. 3). As can be seen in the figure, a concept X in O_1 is deleted in the integration process, and several new concepts, shown in gray, are introduced in the final result. Through reusing, the resulting ontology is expected to be unique, i.e., no such ontology already exists.

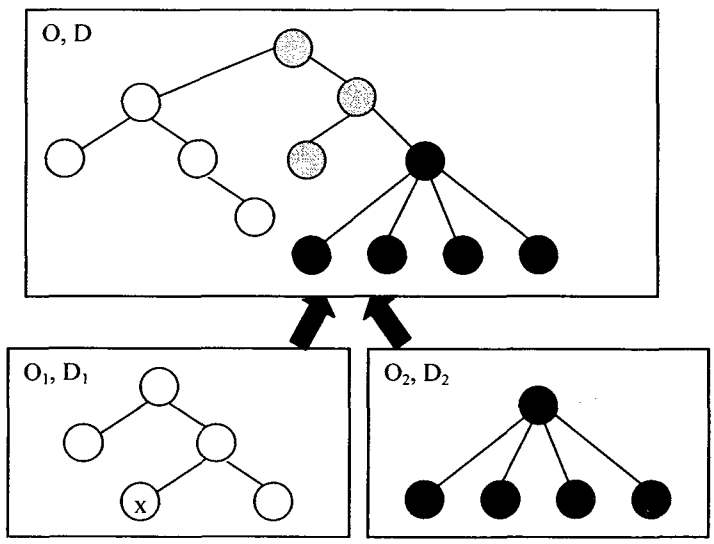


Figure 3: Ontology Reuse

The ontologies to be reused should be selected from those available ontology sources that meet the requirements, such as domain, type, and generality. A resulting "target" ontology, on the other hand, should have features of a "good" ontology, i.e., it should be clear, concise, and have some adequate level of detail.

When building a new ontology by reusing existing ones, some problems, such as consistency and level of detail should be dealt with. To solve such problems, we need to specify a group of reuse operations which indicate how knowledge in the source ontology will be included and combined into the "target" ontology. Some reuse operations are composing, combining, and assembling operations. On the other hand, such operations should be applied only onto those ontologies which have some common features. These features guarantee that a selected source ontology is suitable, that the reuse operations can be successfully applied, and that the "target" ontology will have the "preferred" features.

2.3.2 Ontology Mapping

[Heflin and Hendler, 2000] divide ontology integration methods into three categories: mapping ontology, mapping revisions, and intersection ontology. In *mapping ontologies*, a created ontology O_M contains the rules that map concepts between ontologies O_1 and O_2. In the mapping *revisions* method, o_1' contains rules that map objects in O_2 to terminologies in O_1 and vice versa. In an *intersection ontology*, where the created ontology O_N includes the intersection of concepts common to O_1 and O_2, and renames terms where necessary. See Fig. 4 for an illustration of their integration methods.

[Calvanese *et al.*, 2002] consider mapping between one global ontology (O) and several local ontologies (O_1, O_2, ...), leaving the local ontologies intact by querying them and converting the result into a concept in the global ontology. The basic idea proposed in [Kalfoglou and Schorlemmer, 2002] is to map two local ontologies by looking at how these are mapped from a common ontology. It is assumed that such a common ontology is not populated with instances, while local ontologies usually are. Then, the obtained results are placed in a new global ontology, which is progressively created.

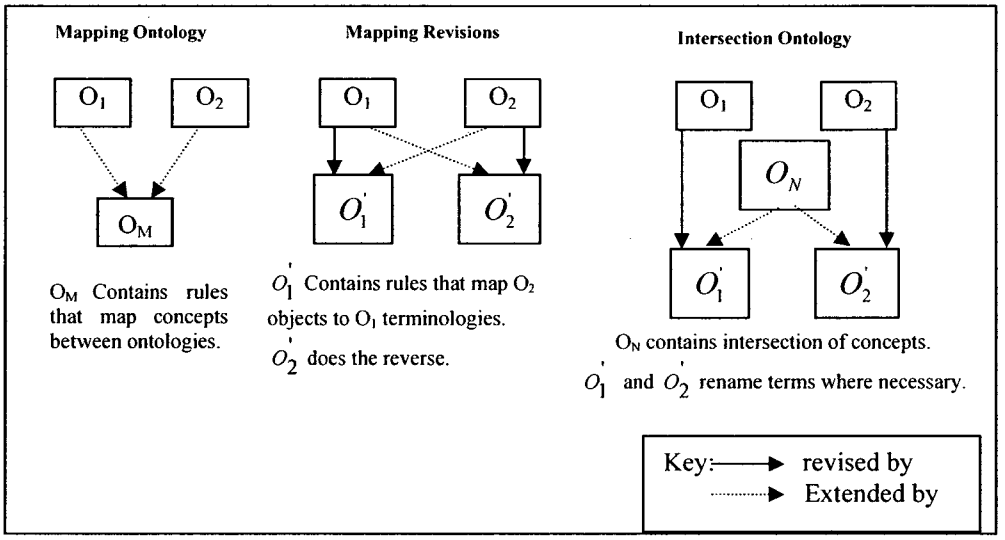


Figure 4: Integration methods according to [Heflin and Hendler, 2000]

The survey in [Wache *et al.*, 2001] divides the ontology mapping into three approaches: single, multiple, and hybrid approaches, as shown in Fig. 5.

The *single ontology* approach uses a global ontology with shared semantics. All information sources are related to this one global ontology. With *multiple ontologies,* there are inter-ontology mappings, but no global ontology. The *Hybrid Approach* is similar to the multiple ontology approach in that the semantics of each source is

described by its own ontology, but, in order to make the source ontologies comparable to each other, they are built using one global shared vocabulary.

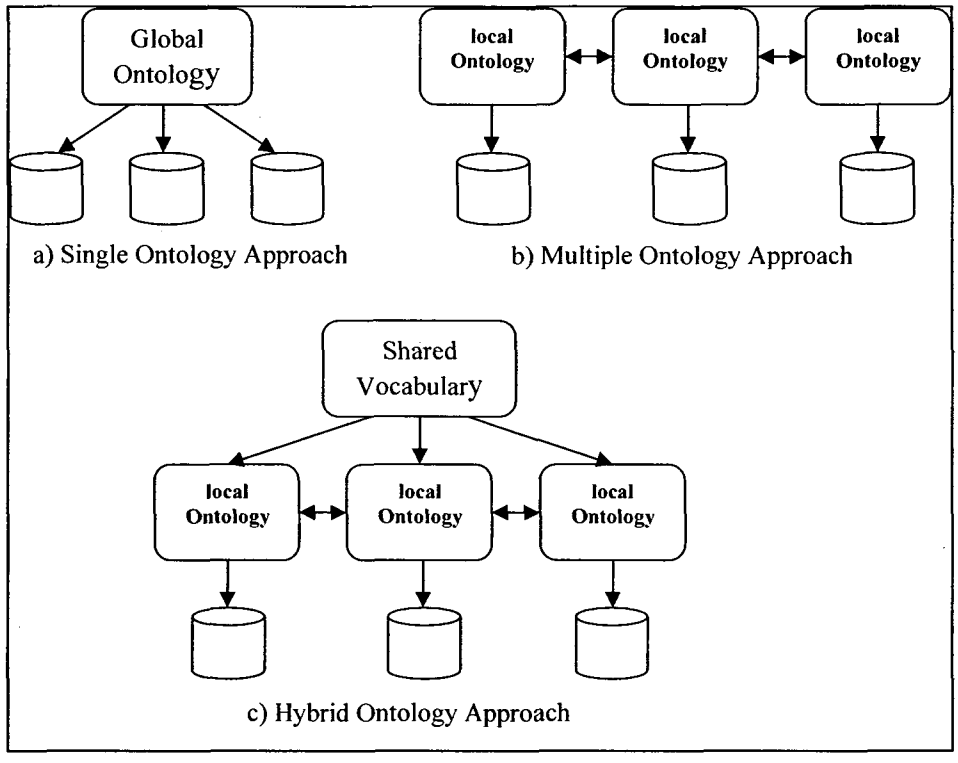


Figure 5: Mapping integration methods [Wache *et al.*, 2001]

The rest of this section explains the main concept of the ontology mapping approach.

The mapping approach deals with situations where there are different ontology sources, created independently of each other by different users. We need to construct a global ontology or "virtual view" for accessing the required information from these different ontologies.

The idea of this "virtual view" is to provide a general framework in which we can query the local source ontologies. Moreover, in order to use the "virtual view" for answering queries, it is important to specify the mappings between the global ontology and the source ontologies. The global ontology is used to formulate queries. To evaluate

queries, the query processing scheme requests access to information in the source ontologies, instead of simply using them.

In reality, different source ontologies are constructed by various users for different purposes over time. Therefore, the same information may be expressed in different forms at different levels of abstraction in the source ontologies. Thus, mapping the concepts in one ontology to another means that a concept in one ontology may correspond to a view "query" over the other ontologies. Actually, suitable query languages should be supported by the ontology specification language, in order to express mappings among concepts in different ontologies. One can view query processing in this context to be closely related to answering queries by using views in data integration systems. An ontology integration system (OIS) in this case is defined as a triple <G, S, M>, where G represents the global ontology, $S = \{S_1..., S_n\}$ represents the set of local ontologies, and M represents the mapping between G and the sources in S.

Based on [Calvanese *et al.*, 2002], there are three basic approaches for defining this mapping:

1. The *global-centric approach*, where concepts in the global ontology G are mapped to concepts in the local ontologies in S.
2. The *local-centric approach*, where concepts of the local ontologies in S are mapped to queries over the global ontology G.
3. The combined global-centric and local-centric approach.

Global-As-View approach (GAV)

This approach is widely used in data integration systems [Calvanese and Giacomo, 2005, Lenzerini, 2002, Ullman, 1997]. In such systems, the global ontology is a database

schema, and the mapping is designed by relating one relational query over the source relations to each relation in the global schema. It is well-known that this approach leads to a simple query processing policy, which reduces to unfolding the query using the definition in the mapping, so as to expand the query in terms of definition of the sources.

Example 2. An ontology integration system (OIS) is defined as $O = <G, S, M_{G,S}>$, where

1. G is the *global ontology* expressed in the *Entity-Relationship model.*
2. S contains the *local sources* over which a relational database is created.
3. $M_{G,S}$ is the *mapping* between G and S given by a set of correspondences of the form $<C, V_s>$, where C is a concept in G and V_s is a query or view over S.

Fig. 6 shows the global schema G of a data integration system, where Age is a functional attribute; Employee has a mandatory participation in the relationship Works-in, Works-in is-a Member, and Company is-a Union. The schema models persons who can be members of one or more unions, and employees who work in companies.

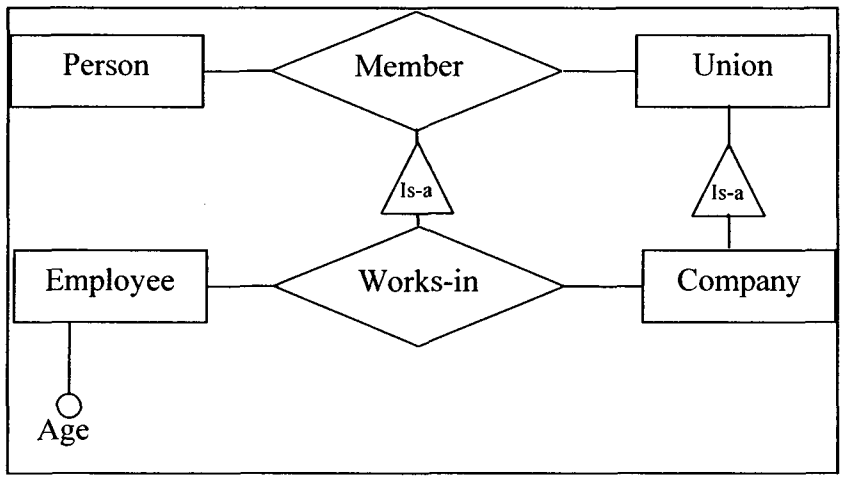


Figure 6: Global Schema G [Calvanese *et al.*, 2002]

Suppose that S includes sources $S_1 ... S_8$, and that the mapping M is given as follows:

Person(x)← S_1(x)

Union(x) ← S_2(x)

Member(x, y) ← S_7(x, z), S_8 (z, y)

Employee(x) ← S_3(x, y) OR S_4(x)

Age(x, y) ← S_3(x, y) OR S_6(x, y, z)

Company(x)← S_5(x)

Works-in(x, y)← S_4(x, y)

Local-As-View approach (LAV)

The main difference between the LAV and GAV approaches is the direction of the mapping. In the GAV approach, the mapping between the global and the local ontologies is given by associating to each concept in the global ontologies a *view*, which is a query over a local ontology. However, in the LAV approach, the mapping direction is reversed, i.e., associated to each concept in a local ontology is a *view*, which is a query over the global ontology.

The main advantage of the LAV approach over GAV is its flexibility and ease of modality, which allows sources to be added or removed from the integrated framework more readily. That is, if we build a global schema and then a new source is to be added to our system, we do not need to reconstruct the global schema from scratch. The challenge, however, with the LAV approach is answering queries posed in terms of the global schema. This is a challenge because we first need to reformulate the queries in terms of queries over the sources. Query processing in the GAV approach is addressed by simply unfolding the queries.

Example 3. Consider for example the OIS, O = <*G, S, $M_{G,S}$*> defined as follows:

1. The global ontology *G* is an *ALCQI* knowledge base

$$Canadian \cap (\geq 1 Has\_relative.Doctor) \subseteq Wealthy$$
$$Surgeon \subseteq Doctor$$

which asserts that every Canadian who has a doctor as a relative is wealthy, and that each surgeon is also a doctor.

2. The set *S* of local ontologies consists of two ontologies, containing the relations T_1 and T_2, with extensions T_1 = {ann, bill} and T_2 = {ann, dan}.

3. $M_{G,S}$ is the *mapping* between *G* and *S* given by a set of correspondences of the form <V_g, *C*>, where V_g is a query or view over G and *C* is a concept in S.

The mapping $M_{G,S}$ is {<V_1, T_1>, <V_2, T_2>} with

$$V_1(x) \leftarrow \text{RELATIVE}(x,y) \text{ and } \text{Surgeon}(y)$$

$$V_2(x) \leftarrow \text{Canadian}(x)$$

V_i associates to each concept in T_i a query over G. In this example, V_1 expresses that each individual x in T_1 has a relative y who is a surgeon, and V_2 expresses that each individual in T_2 is Canadian.

Given a query Wealthy(*x*) over *G,* we find *ann* as the only answer. Consider an additional local ontology T_3 with an extension not containing bill, defined by the following mapping in $M_{G,S}$:

$$V_3(x) \leftarrow \text{Wealthy}(x)$$

From the constraints in *G* and the information we have on the mappings, we can conclude that bill is not an answer to the query Canadian(x) over G.

Combining Global-as-view and Local-As-View approaches (GLAV)

The global and local centric approaches can be combined to yield the so-called GLAV approach, using unrestricted mappings in order to overcome the restrictions on communication directions between global and local ontologies. In the GLAV approach, we have both a query language V_S over the alphabet A_S, a query language V_G over the alphabet A_G, and a mapping between the global and local ontologies, given by relating views over the global ontology to views over the local ontologies. The intention behind relating V_G to V_S is that V_S represents the best way to characterize the objects satisfying V_G in terms of the concepts in *S*.

Example 4. Consider for example the OIS $O = \langle G, S, M_{G,S} \rangle$, where both the global schema and source ontologies S_1 and S_2 are sets of relations with extensions.

1. The global ontology *G* contains two binary relations: relation *WorksFor* to record researchers and projects they work on, and relation *Area* to record projects and research areas they belong to.
2. The local ontology S_1 contains a binary relation *InterestedIn,* which denotes people and the fields they are interested in, while the local ontology S_2 contains the binary relation *GetGrant,* which denotes researchers and their assigned grants, and the binary relation *GrantFor,* which denotes the grants and projects they refer to.
3. The mapping $M_{G,S}$ is formed by the following correspondences:

   - (V_1, *InterestedIn*), with $V_1(r, f) \leftarrow$ WorksFor(r, p) and Area(p, f).

   - (*WorksFor,* V_2), with $V_2(r, p) \leftarrow$ GetGrant(r,g) and GrantFor(g,p).

This kind of mapping representation cannot be achieved using only the GAV or the LAV approach. Query answering using the GLAV approach is largely unexplored

[Calvanese *et al.*, 2002], as it combines the difficulties of the GAV and LAV approaches. On the other hand, this may be the only approach that has the appropriate expressive power.

To summarize, the two approaches of GAV and LAV are compared based on two aspects: modeling and query processing.

1. In the GAV approach, query processing is easier since it uses query unfolding, but modeling is more difficult and maintaining the model G when local sources change often requires the redesigne of G.
2. In the LAV approach, modeling is easier, but query processing is more difficult since it needs query reformulation and reasoning.

2.3.3 Ontology Merging

[Pinto, 1999] defines merging as combining different ontologies with the same subject domain to create a unified ontology. Synonymous with this definition, [Sowa, 1997] defines the unification process. Moreover, the proposal in [Kalfoglou and Schorlemmer, 2002] defines the merger of two ontologies as their intersection, and the knowledge engineer is in charge of making merger decisions. The authors [Noy and Musen, 2000] use the concept of merge synonymous with unification. Their intention is to create a massive governmental knowledge base. While the process of ontology merging defined in [Stumme and Mädche, 2001a] yields a merged ontology from input ontologies, it is not clear how the performance is affected by various assumptions about the input ontologies when their subjects are the same, similar, or complementary. In what follows, we explain the main idea of the ontology merging approach.

In the case of merging ontologies, one wants to build ontologies using concepts, distinctions, axioms, etc., from existing ontologies on the same subject. For instance, when two companies merge into a larger company, their ontologies will be merged by considering similar matching terms. In most cases, there are differences between the input ontologies, not only in their basic features but also in the way their terms are defined (in the meaning behind those terms). When such different ontologies are "integrated" by merging, a new ontology is created in the same domain. The integrated ontology contains unified concepts, terminology, definitions, constraints, etc., from the input ontologies.

In the merging process, we have, on the one hand, at least two ontologies that are going to be *merged* (O_1,O_2, Fig. 7), and on the other hand, the *resulting* ontology (O, Fig. 7).

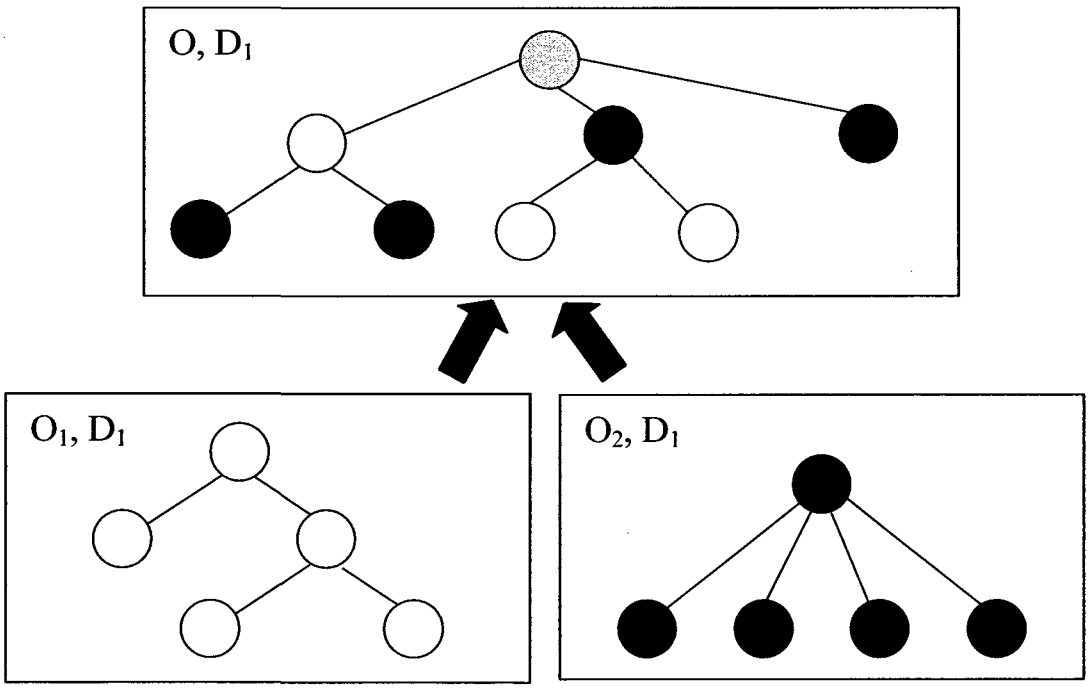


Figure 7: Ontology Merge

The goal is to make a more general resulting ontology by gathering knowledge from several input ontologies on the same subject. The domains of the input and resulting

ontologies are the same (D_1, Fig. 7). Ontology matching is a prerequisite for any approach of ontology integration. In the following section, we review existing techniques for ontology matching.

2.4 Ontology matching techniques

This section reviews techniques currently used for ontology matching. These techniques are classified into element level and structure level techniques [Euzenat and Shvaiko, 2007].

2.4.1 Element-level techniques

These techniques view ontology entities or their instances as isolated from other entities or instances. They are classified into string-based, language-based, and constraints-based, described as follows:

- ***String-based techniques***

These techniques are used to match names of the entities in ontologies. Such techniques are based on the similarity of the names of entities, considered strings. The more similar the strings, the more likely they denote the same concepts. There are numerous methods introduced for string similarity matching. The most frequently used methods are:

I. Edit distance: in this method of matching two entities, a minimal cost of operations to be applied on one entity in order to obtain the other entity is considered. Examples of such well-known measures are Levenshtein distance [Levenshtein, 1966], Needleman-Wunch distance [Needleman and Wunsch, 1970], Smith-Waterman [Smith and Waterman, 1981],

Gotoh [Gotoh, 1981], Monge-Elkan [Monge and Elkan , 1997], Jaro measure [Jaro, 1989, Jaro, 1976], and Smoa [Stoilos *et al.*, 2005].

II. Normalization: To improve the matching results between strings, a normalization operation is performed, usually before matching. In particular, these operations are case normalization, diacritics suppression, blank normalization, link stripping, digital suppression, and punctuation elimination.

III. String equality: the string equality method basically returns 0 if the input strings compared are not identical, and 1 if they are. An example of such a method is the Hamming distance [Hamming, 1950].

IV. Substring test: This identifies the ratio of common subparts between two strings. Also, it is used to compute if a string is a substring of another string. i.e., a prefix or suffix.

V. Token-based distances: Such a method considers a string as a set of words. These methods are used to split long strings (strings that are composed of many words) into independent tokens.

- ***Language-based techniques***

These techniques measure the relatedness of concepts, for which they consider names as words in some natural language, e.g. English. They use Natural Language Processing (NLP) techniques to extract meaningful terms from the text. Usually, they are applied to words (names) of entities. The matching similarity is determined based on linguistic relations between words, such as synonyms and hyponyms. Many language-based methods have been implemented in the WordNet [Pedersen *et al.*, 2004].

- ***Constraints-based techniques***

In order to calculate the similarity between entities, these techniques are mainly applied to the definitions of entities, such as their types, attributes, cardinality and ranges, and the transitivity or symmetry of their properties. There are different methods proposed based on constraints [Rahm and Bernstein, 2001], which compare the properties, data types, and domains of entities.

I. Property comparison: When the properties of two classes are similar (similar names and types), it is more likely that these two classes are similar.

II. Data type comparison: This compares the way in which the values are represented, e.g. integer, float, string.

III. Domain comparison: Depending on the entities to be considered, what can be reached from a property can be different: in classes, these are domains, while in individuals, these are values.

2.4.2 Structure-level techniques

In contrast to element-based techniques, structure-based techniques compare the two entities from two ontologies with regards to the relations of these entities with other entities in the ontologies: the more similar the two entities are, the more alike their relation would be. Mainly, there are two well-known structure level techniques: graph-based techniques and taxonomy-based techniques, described as follows:

- ***Graph-based techniques***

These techniques consider the ontologies to be matched as labeled graphs. The basic idea here is that, if two nodes from two ontologies are similar, their neighbors should also

somehow be similar [Euzenat *et al.*, 2004].

• ***Taxonomy-based techniques***

These techniques are basically graph-based techniques which consider only the specialization relation. The basic idea they focus on is that an *is-a* relationship links terms that are already similar, therefore their neighbors may also be similar [Euzenat and Valtchev, 2004, Valtchev and Euzenat, 1997, Valtchev, 1999, Wu and Palmer, 1994].

Matching ontologies using their structure information is important as it allows all the relations between entities to be taken into account. The most common techniques used for ontology matching are taxonomy-based, since taxonomies play a pivotal role in describing ontologies.

2.5 Ontology matching systems

This section reviews ontology matching systems. The approaches of these systems can be classified into: (1) schema-based, (2) instance-based, and (3) combined, schema and instance based.

2.5.1 Schema-based implementations

Schema-based systems are those which rely on schema information in the input in order to match ontologies. We now describe some schema-based systems.

• **SKAT** (Semantic Knowledge Articulation Tool) [Mitra *et al.*, 1999]

In SKAT, the input ontologies are represented by graphs. It is a rule-based tool which discovers matching results through a semi-automatic process. Domain experts provide rules that are encoded in first order logic. Initially, experts also specify desired

similarities and dissimilarities. For instance, the rule "President is equivalent to Chancellor" specifies that we want President to be an appropriate match for Chancellor. SKAT uses string matching as well as structure matching. In the structure matcher, SKAT matches graph slices, i.e. matching the nodes near the root in the first ontology against the nodes near the root in the second ontology.

• **ONION** (ONtology compositION) [Mitra *et al.*, 2000]

ONION is an extended version of SKAT. It performs a number of matching techniques and suggests articulation rules to users. Users can accept, modify, or delete the suggestions. The structure-based matching in ONION is performed based on the results of linguistic matching. It looks for structural iso-morphism between subgraphs of the ontologies, taking into account linguistic clues. The structural matcher tries to match only the pairs which were not matched by the linguistic matcher, hence complementing its results.

• **H-Match** [Castano *et al.*, 2006]

H-Match takes OWL ontologies as its input. Internally, these input ontologies are represented by graphs using the H-model representation [Castano *et al.*, 2005]. Moreover, H-Match computes two types of similarities: linguistic and contextual. These are then combined using weighting schemas to yield a final measure, called semantic similarity. In determining the contextual similarity, H-match considers neighboring concepts, e.g., linked through the taxonomy of the actual concept.

• **Anchor-Prompt** [Noy and Musen, 2001]

Ancor-Prompt is an extension of Prompt and was originally called SMART [Noy and Musen, 2000]. Basically, it is an algorithm for matching concept names. If there is a

match between two concepts in the source ontologies, and there are paths connecting these two concepts, then there should be similarities between these paths as well. Fig. 8 shows that there is a match "anchor" between concept A from one source ontology and concept B from another source ontology. It also shows that there is a match "anchor" between concept names H and J. In this case, the tool would suggest that there are some similarities between those concepts which lie between the two anchors, such as concepts G and F, and that concepts E and D may share some properties with concept C. All in all, these are only suggestions made by the tools; the user may confirm the suggestion, and hence merge the concepts, or reject the suggestion.

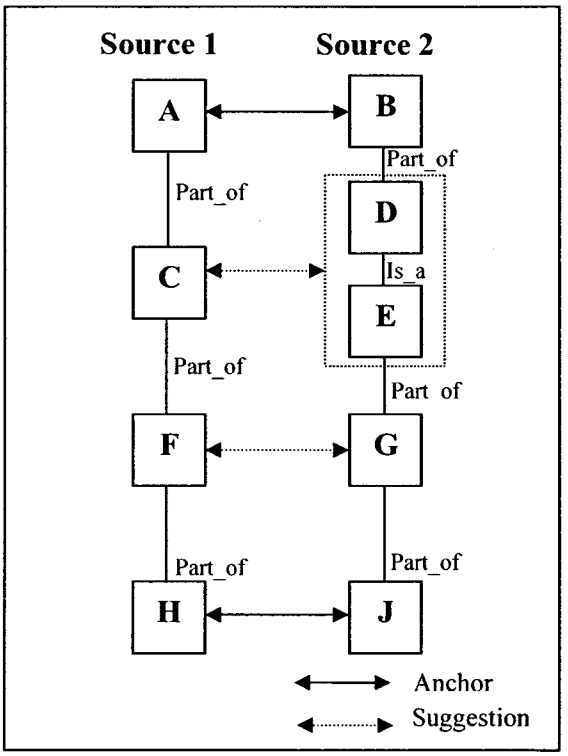


Figure 8: Prompt Algorithm [Noy and Musen, 2001]

• **MapOnto** [An *et al.*, 2006, An *et al.*, 2005a, An *et al.*, 2005b]

The MapOnto is a tool for recommending matches between ontologies and relational or XML schemas. The input schema and ontology are both represented internally as labeled

graphs. Then, the system looks for similarities or relationships between these graphs, and produces a set of complex mapping formulas, expressed as Horn clauses, in a semi-automatic way. These logical formulas are ordered by the tool, thereby suggesting the most reasonable mappings. Finally, the user can inspect this list and choose the best mappings.

• **CtxMatch** [Bouquet *et al.*, 2003a, Bouquet *et al.*, 2003b]

CtxMatch deals with the ontology matching problem by translating it into the logical validity problem. It determines the logical relationship, such as equivalence and subsumption, between concepts and properties. The first version of CtxMatch uses only WordNet to find initial matches for classes. In the next version, CtxMatch2 [Bouquet *et al.*, 2006], it also considers properties. Basically, it employs description logics reasoners, such as Pellet [Sirin *et al.*, 2007] and FaCT [Tsarkov and Horrocks, 2006] to compute the final alignment.

• **S-Match** [Giunchiglia *et al.*, 2003]

S-Match takes two graph-like structures, e.g., classifications, XML schemas, or ontologies, as input and returns logical relationships, e.g., equivalence and subsumption, found between the nodes of the graphs. Ontology entities are converted to logical formulas. Then, the match manager uses various basic element-level matchers and logic provers to find relationships between these formulas, which in turn correspond to the relationships between entities.

• **ASCO** [Bach *et al.*, 2004]

The first version of ASCO deals with ontologies represented in an RDF schema, while its new version, ASCO2, deals with ontologies represented in OWL [Bach and Kuntz, 2005].

ASCO performs in three phases. In phase 1, it computes the similarity between the entities of the ontologies using linguistic matchers. In phase 2, it applies a fixpoint computation algorithm that propagates similarity to the neighbours (subclasses, superclasses and siblings). Similarities between sets of objects are computed through single linkage. The propagation terminates when class similarities and relation similarities no longer change in a subsequent iteration or when a certain iteration step is reached. Finally, in phase 3, ASCO aggregates the results of linguistic and structural matchers using a weighted sum.

• **OMEN** (Ontology Mapping ENhancer) [Mitra *et al.*, 2005]

The OMEN system is based on a Bayesian network. It is an enhancing tool for ontology matching, which improves existing ontology matching algorithms using probability inferences. The matching process for OMEN can be described as follows:

1. OMEN builds a Bayesian network, where a node represents the mapping between classes or properties of the input ontologies. Edges represent the influences of the Bayesian network between these nodes.
2. OMEN generates the conditional probability tables for the Bayesian network. It accomplishes this by using a set of meta-rules that capture the influence of the structure of the input ontologies on the neighborhood of the input mappings.
3. OMEN makes inferences using Bayesian Network tools, in order to provide newly determined probabilities for each node.
4. Finally, the new probabilities, which are larger than a certain threshold, are selected to generate the resulting alignment.

2.5.2 Instance-based implementations

This section reviews major ontology matching systems, which consider instances to determine the matching entities of input ontologies.

• **T-tree** [Euzenat, 1994]

This system uses instances of the input ontologies to determine the matching classes.

It infers "bridges" between classes of different ontologies sharing the same set of instances. Given a source and destination taxonomy, T-tree returns all "bridges" for which the instances in every source class are present in the destination class.

• **CAIMAN** [Lacher and Groh, 2001]

CAIMAN is a system for document exchange, which focuses on lightweight ontologies. It determines a probability measure between concepts of two ontologies by applying machine learning techniques for text classification. In particular, based on the documents, a representative feature vector is created for each concept in an ontology. Then, the matching similarity is determined for these class vectors. Finally, with the help of a threshold, the matching result is produced.

• **FCA-merge** [Stumme and Mädche, 2001b]

The FCA-merge has three main steps for merging two ontologies: instance extraction, concept lattice computation, and generation of the final merged ontology. Actually, the FCA-merge uses the formal concept analysis technique in the second step. The idea behind this technique is to compare classes which share instances by testing the intersection of their instances.

• **GLUE** [Doan *et al.*, 2004]

GLUE is an extended version of Learning Source Descriptions (LSD). It uses multiple learners and exploits information in concept instances and taxonomy structures of ontologies. GLUE works in three steps. First, it learns the joint probability distributions of classes in the input taxonomies. Then, it estimates the similarity between these classes. This results in a similarity matrix between classes of the input taxonomies. Finally, GLUE filters some of the matches from the similarity matrix and keeps only the best ones.

2.5.3 Combined schema- and instance-based implementation

This section explores the ontology matching systems which use both schema and instances from the input ontologies to find their matching entities.

• **IF-MAP** (Information-Flow-based Map) [Kalfoglou and Schorlemmer, 2003]

IF-Map matches two input ontologies with respect to a reference ontology. In other words, it considers that the reference ontology represents an agreed understanding, which facilitates knowledge sharing. Moreover, IF-Map assumes that the given input ontologies include portions which match the reference ontology. It also assumes that the reference ontology does not need to be populated with instances. The matching process proceeds as follows. If the instances of the input ontologies can be assigned concepts in the reference ontology and the reference ontology can be expressed in each of the input ontologies, then IF-Map uses the three ontologies in order to extract the matching entities (using the formal concept analysis technique). When a matching (between the three ontologies) is not found, IF-Map returns the matching candidates using string-based and structure-based methods.

• **oMAP** [Straccia and Troncy., 2005]

oMap deploys a number of matchers in order to find the correspondences between entities of the input ontologies. The matchers include a string similarity measure, learning methods used on instance data, and a matcher that propagates preliminary weights through the ontology constructors used in the definitions of ontology entities. At the end, the results are aggregated using a weighted average.

• **OLA** (OWL Lite Aligner) [Euzenat and Valtchev, 2004]

OLA is a system that takes an equal contribution of each component of the ontologies, e.g., classes, instances ... etc in order to find the matching entities of the input ontologies. It considers ontologies as graphs, and determines the similarity of the graph nodes based on string, language, and structure based similarities. These similarities are aggregated. For computing these similarities, OLA starts with base distance measures computed from labels and concrete data types. Then, it iterates a fixpoint algorithm until it no longer yields an improvement.

• **Falcon-AO** [Hu *et al.*, 2007]

Falcon-AO has three elementary matchers: two linguistics matchers (V-DOC and I-sub) and a structural matcher (GMO). GMO is a bipartite graph matcher which starts by considering the RDF representation of the ontologies as a bipartite graph, represented by its adjacency matrix. The results of Falcon-AO mainly derive from the alignments generated by linguistic or structural matchers, depending on which has better results. Otherwise, the Falcon-AO generates the results by combining both linguistic and structural matchers using a weighting scheme.

• **RiMOM** (Risk Minimization based Ontology Mapping) [Li *et al.*, 2007]

The RiMOM system integrates multiple strategies, such as edit distance, statistical learning, and three similarity propagation-based strategies. Then, it applies a strategy selection method in order to decide on which strategy it will rely more. As a result, RiMOM combines the conducted alignment. RiMOM offers three possible structural propagation strategies: concept-to-concept propagation strategy (CCP), property-to-property propagation strategy (PPP), and concept-to-property propagation strategy (CPP). To choose between them, RiMOM uses heuristic rules. For example, if the structure similarity factor is lower than some threshold, then RiMOM does not use the CCP and PPP strategies, but uses CPP. The basic idea of CCP, PPP, and CCP is to propagate the similarities of (concept pairs or property pairs) across the concept/property hierarchy structure. For instance, in CCP, similarities of concept pairs are propagated across the concept hierarchy structure.

2.6 Summary

- We reviewed the DLs, which are widely used as the formalism for the semantic web, specifically the Web Ontology Language with its correspondence to description logics (OWL-DL). We described the families of DL Languages and their main constructs, and explained how they differ in the constructs they use.
- We described the main components of ontologies, such as *classes or concepts, roles or properties, axioms, individuals or instances* along with an illustrative example for the meaning of each component.

- We reviewed the approaches to ontology integration: reusing, mapping, and merging.
    I. First, the reusing approach refers to the reuse of widely available ontologies as main parts to build a new ontology instead of creating it from scratch.
    II. Second, the mapping approach refers to the situations where there are different ontologies, created separately from each other by different users, and we need to construct a global ontology or "virtual view" for accessing the required information from these ontology sources. The main goal of the "virtual view" is to provide a general view, in which we can query the different source ontologies. Basically, there are three basic approaches for defining this mapping: Global-As-View approach (GAV), Local-As-View approach (LAV), and Combining Global-as-view and Local-As-View (GLAV).
    III. Third, the merging approach refers to the process as the intersection between the two given ontologies and the engineer is in charge of making the final decisions.
- We classified a variety of ontology matching techniques into two main levels: element and structure level techniques. The element level techniques have been further classified into three basic techniques, string-based techniques, language-based techniques, and constraint-based techniques. The structure level techniques have been classified into graph-based techniques and taxonomy-based techniques.

- We also classified ontology matching implementations (matching systems) on the basis of their input information level, which could be: schema-based, instance-based, and the combined schema- and instance-based matching systems.
- From the point of view of architecture, following the proposals for information integration, we can classify the approaches to ontology integration into two: (1) the data warehouse (DW) or materialized approach, and (2) the virtual (mediator-based) approach. Accordingly,
  I. Ontology reuse and merging in the ontology integration approaches are similar to the materialized approach in information integration. In both ontology and information integration approaches, information is gathered from more than one source and is stored into a single source (warehouse).
  II. The ontology mapping approach can be considered similar to the virtual approach to information integration (mediator-based), since they both use an integrated virtual view through which we can query the information sources.
  III. The data warehouse approach supports decision making and querying data, as it explicitly stores information from heterogeneous sources locally. However, maintenance is a major issue when a data source frequently changes.
  IV. Another approach to information integration is to provide an integrated view. This is preferred over the DW approach when the information sources change often. On the other hand, the DW approach is more suitable for data mining.

- Most ontology matching systems focus on one-to-one matching, i.e., they match one pair of entities at a time. They do not match *n* entities to *m* entities simultaneously, and hence use several similarity measures to solve the ontology matching problem.
- A single similarity measure, such as name similarity, graph matching, etc., for matching ontologies is useful and efficient in some specific domains. For example, matching techniques which are based on syntactic similarity measures provide good results in domains, where there is a high probability that whenever the matched entities agree on their terms, they also agree on their semantics. However, such techniques, which are solely based on name similarity, might not work well in application domains where similar entity names and terms are used with different meanings. To improve this situation, it has been proposed to use multiple measures at the same time.
- Many matching systems provide a library for the matching techniques that can be used for given input ontologies. However, the user often has no knowledge about which matching technique is more appropriate for the application at hand. In the following chapter, we study these techniques and suggest improvements.

3. Ontology Integration: A Hybrid Approach

This chapter introduces, in Section 1, an overview of the hybrid approach we proposed in [Alasoud *et al.*, 2005] for ontology integration, which is a hybrid between the fully materialized and fully virtual approaches. Section 2 motivates the proposed approach with an example, while Section 3 describes its architecture. Section 4 explains the method used for mapping global and local ontologies. Section 5 describes our implementation.

3.1 Overview of the Hybrid Approach

Extracting information from ontologies created by different users is an important and challenging task for answering queries originate from the Web. In this section, we propose a framework for ontology integration which is a hybrid of materialized (data warehouse) and virtual views.

The rapid increase in the number of information sources requires efficient and flexible frameworks for their integration. Such frameworks should provide a way for extracting, transforming, and loading data from these sources, and presenting them to the user in an appropriate way. There are two major approaches for the integration of information: (1) the data warehouse (DW) or materialized approach and (2) the virtual (mediator-based) approach. In the DW approach, a huge amount of historical data is stored in the DW. In the virtual approach, on the other hand, the data is not materialized, but rather is globally manipulated using views. Each of these approaches is suitable for some applications.

DW is a powerful tool for decision support and querying data because it explicitly stores information from (possibly heterogeneous) sources locally. However, some external data, such as new product announcements from competitors and currency exchange rates, may be needed to help in business decisions. We should not neglect the importance of such data to avoid the problems of incomplete, inexact, or sometimes wrong results. Warehousing huge and frequently changing information is a big challenge for the following reasons. Firstly, the data in the DW is loaded in snapshots and the DW is a huge information repository. Secondly, as the data sources change frequently, maintenance becomes a complicated and costly issue.

The other approach to information integration is to provide a virtual integrated view. In this approach, the actual data resides in the sources, and queries against the integrated 'virtual' view will be decomposed into sub-queries and posed to the sources. This approach is preferred over the DW approach when information sources change very often. On the other hand, the DW approach may be desired in case quick answers to

queries are required and the information sources change rarely. In order to keep both advantages, we consider a third approach which is a hybrid between fully materialized and fully virtual approaches.

A framework for warehousing web contents has also been discussed in [Zhu, 1999]. It uses a hybrid approach in order to integrate DW data with the "required" web-based information. This framework considers ontologies which express domain knowledge related to web sources and the logical model of the data warehouse. Moreover, an ontology engine is being deployed as an intermediate layer by defining the mapping rules between the web data and attributes of the DW in the ontologies to aid the DW structure and repairing requirements. In [Zhu, 1999], some web data are selected for materialization. However, some queries may not be answered using only materialized data in the DW. In [Calvanese *et al.*, 2002], a framework for ontology integration was introduced based on the fully virtual approach. They constructed the integrated 'virtual' view based on the mapping between the local ontologies and the global ontology. This maps a concept in one ontology to a query over other ontologies. Then, when a query is posed based on the global 'virtual' ontology, which uses the mapping, it is unfolded and evaluated against sources. [Calvanese and Giacomo, 2005] extend the ontology integration framework by using the Description Logic DL-Lite for expressing the global schema, and using the LAV approach for mapping between the local source ' databases' and the global ontology schema. There are some noticeable differences between our work and theirs. First, we consider the global ontology as partially materialized to improve query evaluation time, and to keep the advantages of the fully materialized and fully

virtual approaches. Second, data sources considered are expressed in ontologies. In the next section, we will describe some motivating examples for our hybrid approach.

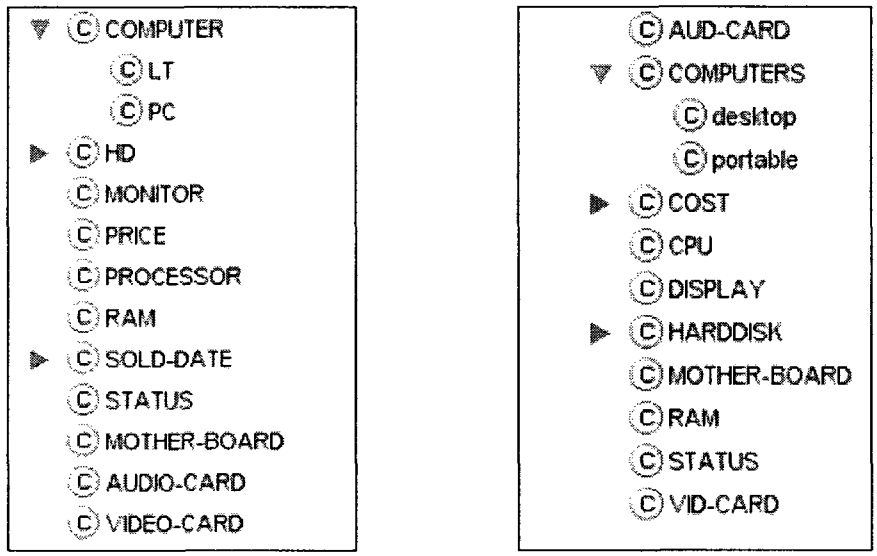
3.2 Motivating Examples

The following examples illustrate these approaches to integration. Consider an ontology of an enterprise "A", which offers different types of electronic products. For simplicity, we consider only two products, PCs and laptops. Fig. 9(a) introduces this ontology for items at enterprise "A." It includes the concept COMPUTER which represents the desktop and laptop computers. Other concepts in this ontology, such as MONITOR, PROCESSOR, and PRICE represent some specifications of the computers. Also, suppose there is an external data source for the same products, which is enterprise "B", shown in Fig. 9(b). The following three examples will illustrate how the integrated view could be supported as:

- Fully materialized: where a posed query needs to be answered only using materialized views.
- Fully virtual: where all sources are defined as views, i.e., non-materialized. This is useful when queries are infrequent or data changes frequently, and hence a query should be evaluated using the source ontologies.
- A hybrid of both: where the answers to queries are retrieved from materialized views as well as virtual views.

For simplicity, we assume the ontology of enterprise "A" is part of a larger ontology restricted to a branch of this enterprise. Moreover, the designer can create a global ontology, "an integrated view", in which clients can pose queries against branches. In

addition, the company wants to get external information about their competitors, such as enterprise "B." This global ontology, or "integrated view", is shown in Fig. 10.



a) Ontology of enterprise A b) Ontology of enterprise B

Figure 9: Source ontologies

Furthermore, designing a global ontology depends on many factors such as: (1) which data are very frequently queried and rarely change, and (2) which are not frequently queried and/or frequently change. In some cases, however, the designer may not materialize frequently queried data if it is frequently changing. For example, in bank applications, an account balance is frequently queried, but it also changes frequently. Therefore, by materializing such data, the up-to-date status of clients may not be available. It would be more appropriate in this case that the client use a virtual view, through which he/she can access up-to-date data from the sources. These issues and factors should be taken into consideration during the design phase.

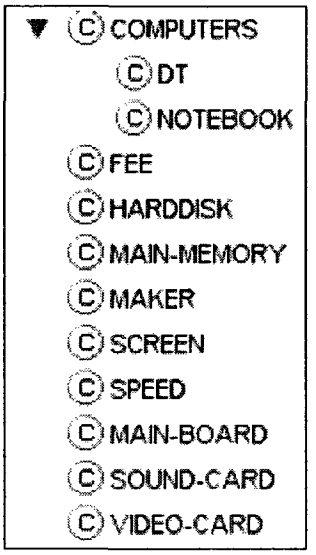


Figure 10: The global ontology

Example 5. In this example, we illustrate the case where the end user query is answered from the materialized view only. First of all, deciding which data in enterprise "A" should be materialized is decided by the designer, based on the aforementioned issues. We will assume the following:

- The most frequently queried and infrequently changed data are the prices of the computers.
- A computer price is affected by specifications, such as its processor type and speed, the size of its hard disk, the size and kind of main memory, and the monitor type and size.
- Other specifications, such as types of the main board, sound card, mouse, video card, etc., may not affect the price much and are not frequently queried.

Based on these assumptions, the concepts of FEE, SPEED, HARDDISK, MAIN-MEMORY, and SCREEN can be fully materialized, and the concept COMPUTERS can be partially materialized. In partially materialized concepts, only one model of each

category of computers, with the same specifications of speed, hard disk size, main memory, and screen, will be materialized.

Whenever the user asks for the price of a computer with particular specifications for the enterprise "A", the answer will be retrieved from the materialized data only.

Example 6. Let us consider a situation where important information about products by other enterprises is required in order to make business decisions in enterprise "A." We therefore need to query some selected ontologies, which may not be in materialized form, such as the enterprise "B" ontology in our running example. This process of evaluating such queries is fully virtual, as it queryies the sources through the integrated view. In other words, the global ontology of the integrated view is used as an intermediate layer to decompose the queries into sub-queries and to get the answers to each sub-query from the relevant sources.

Example 7. In this example, we consider that the global ontology of the integrated view is partially materialized. The frequently accessed data is materialized and the rest are provided as sources. For instance, we could decide to not materialize the infrequently queried data, such as main board type, case material, sound card specifications, and video card type, or any other types of data sources which are not frequenly queried by users. Another case is when enterprise "A" needs to compare the prices of its computers with those produced by enterprise "B." Such queries would be answered from the materialized prices for "A" and the virtual prices for "B."

3.3 An Architecture to Support Ontology Integration

As shown in Fig. 11, the framework includes a Global Ontology (GO) and a set of wrappers. In this framework, GO follows a Local as View (LAV) approach [Calvanese and Giacomo, 2005] to represent the mapping between the concepts in the source ontologies (Ontology 1, Ontology 2, ...) and the GO. We will discuss this mapping in the next section. The Transformation Processor (TP) transforms the data from the data source model to the materialized data model. In our implementation, we consider the materialized data being represented as an ontology. During the maintenance of the materialized ontology, according to the update occurring in the data sources, the Incremental Maintenance Processor (IMP) will determine which data in the materialized ontology may be updated. After the IMP receives the data from the GO, it will compare the new data with the old data in the materialized ontology to decide which parts need to be updated (during the regular-based updates).

The two modules TP and IMP in the dashed box in the Fig. 11 form the Maintenance module for the Materialized Ontology (MMO). The task of the Query Processor QP in our architecture determines if a user query could be answered from the materialized ontology (MO), source ontologies, or both. If the query needs the actual data, i.e. data from the sources, then the query is decomposed and rewritten based on the mapping of the concepts between the global ontology and the source ontologies. As soon as the QP gets the answer from the source ontologies and the materialized ontology, it "merges" the answers into one answer and returns it to the user. The MetaData (MD) module is a repository for the matching terms for the concepts, roles, and individuals used by both GO and the source ontologies.

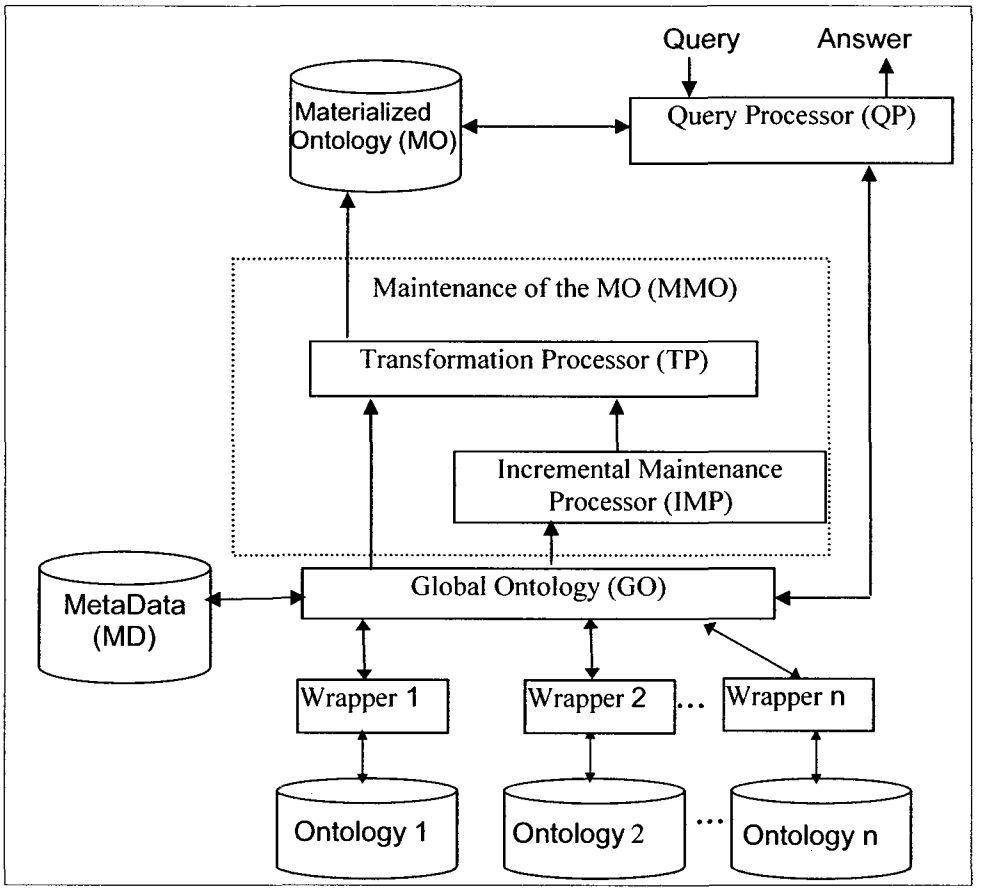


Figure 11: An architecture for the hybrid framework

3.4 Mapping between the Global Ontology (GO) and the Source Ontologies

We can think of GO as consisting of two main parts. The first part is the materialized one, and it is responsible for updating the materialized ontology when required. The second part of the GO is the virtual one, and it is responsible for providing the extra information from the sources which is not materialized nor partially materialized. In other words, this virtual part collects answers to queries that could not be answered using only the materialized ontology. As reviewed in [Ullman, 1997], there have been different approaches proposed for modeling a global view, such as, Global as View (GAV), Local

as View (LAV), and Global-Local as View (GLAV). In the GAV approach, a concept in the global ontology is mapped to a query over the source ontologies. In other words, when the user poses a query over the global ontology, the data corresponds to a concept in the global ontology, which can actually be answered from the source ontologies through a specific query. Since it uses unfolding, query processing in GAV is easy by associating each concept in the global ontology mentioned in the user query with a query over the sources. This approach, however, makes the modeling of a global ontology difficult when the sources change or grow very often, since these changes affect the mappings in general. In contrast, the LAV approach defines the mapping the other way around: each concept in the source ontologies is defined as a query over the global ontology. This makes query processing more complex since the system does not explicitly know how to reformulate the concepts in the global ontology expressed in the user query in terms of source ontologies. On the other hand, modeling of global and source ontologies is easier since changes or incremental growth in the sources will not lead to a reconstruction of the entire global ontology, but only to modifying the mappings. The GLAV approach combines the GAV and LAV approaches, where there are unrestricted mappings in which the restrictions on the direction of the association between integrated and local schema are overcome. Query answering in this approach is largely unexplored, mainly because it combines the difficulties of the GAV and LAV approaches.

Regardless of its difficulties, many researchers show [Calvanese and Giacomo, 2005, Calvanese *et al.*, 2004, Calvanese *et al.*, 2002, Ullman, 1997] that the LAV

approach better supports a dynamic environment where data sources can be added to or removed from the system without restructuring the global ontology.

3.5 Implementation of Research Prototype

This section describes general ideas and technical details of the implementation of a prototype of the framework. We illustrate this by building a framework which integrates the source ontologies for enterprises A and B.

We follow the LAV approach for representing the mapping between the global ontology and the source ontologies. This mapping is expressed in nRQL (New RACER Query Language) [Haarslev *et al.*, 2004] as follows:

$$\text{Enterprise\_A}(x) \subseteq (\text{retrieve}(?x) \; (?x \mid \text{Enterp\_A} \mid \mid \text{has\_maker} \mid))$$

$$\text{Enterprise\_B}(x) \subseteq (\text{retrieve}(?x) \; (?x \mid \text{Enterp\_B} \mid \mid \text{has\_maker} \mid))$$

where x is a variable, Enterp_A and Enterp_B are individuals of the concept MAKER, and has_maker is a binary role between the instances of the concepts COMPUTERS and MAKER in the global ontology. Furthermore, the mapping of the products of the source ontology Enterprise_A over the global ontology is defined as all individuals that have Enterprise_A as their maker. Similar mapping definitions are used for Enterprise_B.

In the above mapping, we represent the concepts in source ontologies over the global ontology. This means that we map the products of enterprises A and B to queries over the global ontology by defining the concept MAKER and adding the relation has-maker between the instances of the concepts COMPUTERS and MAKER in the global

ontology. This mapping gives a hint to the query processor module QP to determine to which source a query should be sent to for evaluation. For instance, QP will send the query to the ontology of enterprise A, if it detects the has_maker relationship in the query is associated with the individual Enterp_A. Also, QP will query the ontology of enterprise B in case the has_maker relationship in the query is associated with the individual Enterp_B. However, QP will send the query to the both sources (ontology of enterprise A and ontology of enterprise B) in case the has_maker relationship is not associated with any one of them.

Moreover, the semantic web deals with diverse types of query answering with access to information represented in different formats. To allow complex queries over the global ontology, mapping these concepts to the global ontology is essential. We use RACER [Haarslev and Moeller, 2001b] together with nRQL, to support flexible construction of queries. Also, we use the OWL-DL Web Ontology Language with correspondence to description logics (DL) as the formalism for the global and source ontologies. We use Protégé version 3.0 [Protégé, 2008] as an editor to develop the knowledge bases.

The following are three query examples written in the nRQL syntax. Each example shows a different scenario for query processing, rewriting, and answering. Furthermore, the examples will show different cases for answering queries: answering using a materialized ontology only, source ontologies only, or both.

The following example shows a query whose answer is generated from the materialized ontology only. Consider the query "list the price of all laptops made by enterprise A, with the specifications: hard disk = 40 GB, screen type LCD with size = 21",

main memory type is SD with size =560 KB, and processor type is Pentium-4 with speed = 2 GHZ." Considering the global ontology provided, this query would be formulated in nRQL as follows:

```
(retrieve (?y)
    (and (?y ?c |price-of|)
         ( ?c  |pnt4-2|  |has-speed| )
         ( ?c  |hard-disk-40|  |hard-disk|)
         ( ?c  |lcd-21|  |has-screen|)
         ( ?c  |sd-560|  |has-ram|)
         ( ?c  |notebook|)( ?c  |A|  |has-maker|) ) )
```

Based on the mappings, the QP can easily figure out that the query should be sent to and evaluated at enterprise A. Then, considering our previous assumptions that the concepts FEE, SPEED, HARDDISK, MAIN-MEMORY, and SCREEN are materialized, the QP will compute the answer using the materialized ontology only and send the answer to the end user.

The second query we consider is similar to the first, except we ask the answers to be retrieved purely from the source ontologies. In this case, the query answers should be retrieved from data sources, i.e., the ontology of enterprise B "(?c |B| |has-maker|)", or the non-materialized concepts in the global ontology for the enterprise A, such as main board type, case material, sound card specifications, and video card type, etc. For such queries, the QP will rewrite them according to source ontologies, and send them to the IV module. Once the answers are obtained, the QP merges the results and sends them back to the user.

The third query scenario, shown in Fig. 12, is a combination of the previous two scenarios. Here, the query is the same as in the first example except that we want to compare the prices of the laptops made by A and B with the specifications mentioned above.

In the formulation of this query, we might not wish to specify the has-maker relation. In that case, the QP will decompose the original query, rewrite it, and evaluate the query using both materialized and source ontologies for evaluation. After receiving the results from both types of sources, the QP merges these results and sends the final answer to the user.

```
Enter the query:
(retrieve (?y)
    (and (?y ?c  |price-of|)
        (?c |pnt4-2| |has-speed|)
        (?c  |hard-disk-40| |has-hard-disk|)
        (?c |lcd-21| |has-screen|)
        (?c |sd-560| |has-ram|)(?c |notebook|)))
Query answer :
Enterprise A: (((?|5500 $|)))
Enterprise B: (((?|1100 $|)))
```

Figure 12: nRQL query with its answer

3.6 Summary

In this chapter, we presented a hybrid approach to ontology integration, to combine the advantages of both the virtual and materialize approaches. Furthermore, we discussed our architecture to support ontology integration, mapping between global and local ontologies, and the implementations of the proposed approach.

In order to build the global ontology (which has a common vocabulary among the sources), a matching module is necessary. This allows the query processing (QP) component to extract information from the ontology sources. To support this capability, we need to build the mapping for the proposed framework, for which effective matching techniques should be developed.

In the next chapter, we propose the multi-match technique for this purpose.

4. Multi-Match strategy

The motivation for our research in the integration of source ontologies was to develop tools and techniques for situations where the information sources are expressed as ontologies. In order to support queries over these sources, we need to build a global ontology, which has a common vocabulary among the sources. This allows the query processing (QP) component in the framework, introduced in the previous chapter (Sec 3.3), to extract information from the ontology sources. To support this capability, we need to build the mapping for the proposed framework, for which effective matching techniques should be developed.

In this chapter, we study the ontology matching problem and propose a solution technique called the multi-matching algorithm (MMA), which uses a multi search algorithm to find the correspondences between entities in the input ontologies. An important feature of this method is that it benefits from existing individual match techniques and "combines" their match results to provide enhanced ontology matching.

4.1 Motivating example

In this section, we focus on the ontology matching problem and introduce some concepts and techniques. Let us consider the following examples. Suppose source ontology "S", shown in Fig. 13, offers different types of electronic products. For simplicity, we consider only two products: PCs and laptops. As can be seen, S includes the concept *COMPUTERS*, which represents the *desktop* and *laptop* computers. Other concepts in this ontology, such as *MONITOR, PROCESSOR,* and *PRICE,* represent technical specifications of computers. As the target ontology, consider ontology "T", shown in Fig. 14. The goal is to find the corresponding matches between the entities in S and T.

There are numerous methods to measure similarities between two entities, such as string similarity, linguistic similarity, etc. However, when we use a single match measure for the input pair of ontologies, we may not be satisfied with the final match result. For instance, if we only use a string similarity measure, the concepts *PC* and *LT* in S have no matches in T. On the other hand, a string similarity measure works fine in domains where a match in the entities on their syntax would most probably mean agreement on their semantics.

Another example is when we use a more semantic measure, such as linguistic-based. For instance, using such a measure, we can find that the concept *PC* in S is matched to concept *desktop* in T and to concept *computer* in T. This will not help the user much in deciding the correspondences and the matched entities. However, if we use both measures (string and linguistic), the concept *computers* in S will be matched with the concept *computers* in T with a higher confidence. Consequently, the concept *PC* in S will be matched to *desktop* in T, and the concept *LT* in S will be matched to *portable* in T.

We propose a multi-match search algorithm that combines different measures in one unified framework to improve the matching results. Further, it minimizes the user's interaction with the system and suggests, for a collection of *n* elements in S, a collection of *m* elements in T.

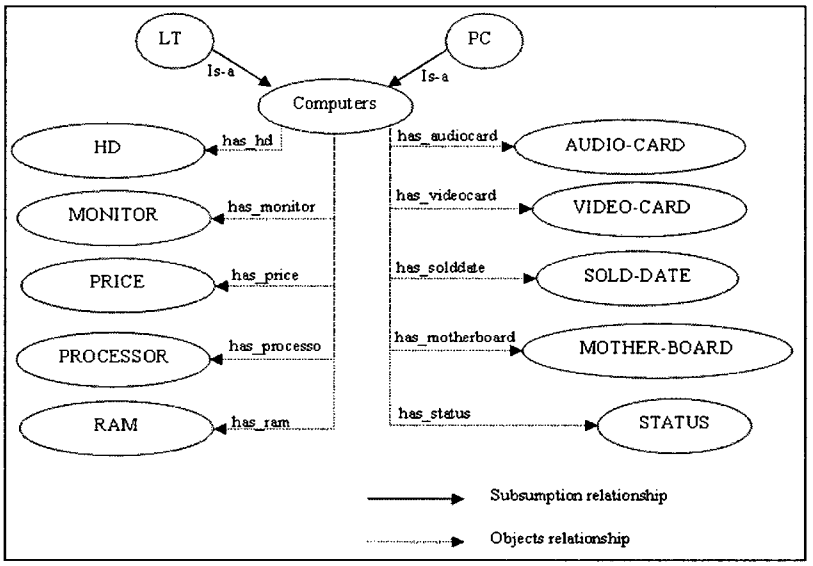


Figure 13: Source ontology "S"

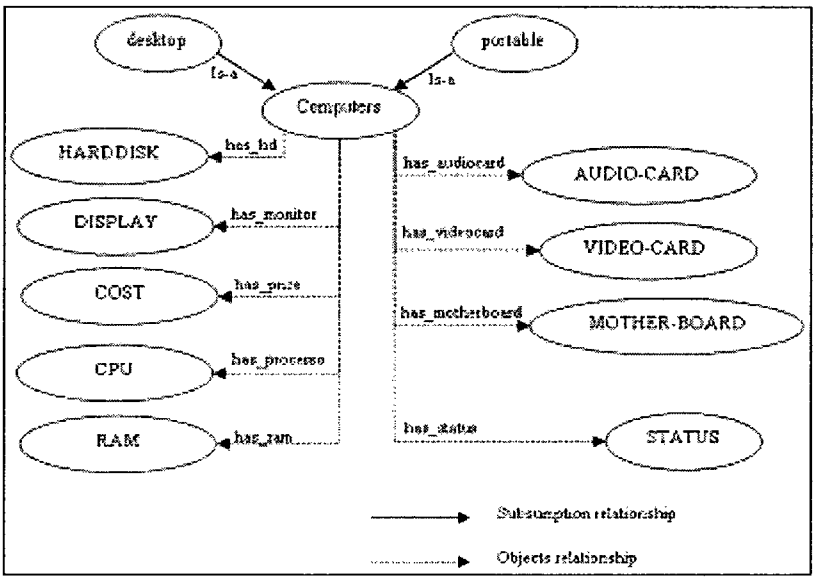


Figure 14: Target ontology "T"

4.2 Background definitions

We describe the ontology mapping problem as identifying pairs of similar nodes (also called vertices) in the input ontologies, modelled as labeled directed graphs. The nodes in the input graph correspond to entities in the ontologies, and the edges indicate the relationships between the pairs of nodes they connect. The labels indicate the kind of relationship, e.g. "domain" or "range."

Before introducing the multi-match framework, in this section we provide some notations and definitions.

Definition 1 (Entity-relationships)

Let S be a source ontology, and T be a target ontology. We use $E^S = \{s_1, s_2, \ldots, s_n\}$ and $E^T = \{t_1, t_2, \ldots, t_m\}$ to denote the sets of entities in S and T, respectively. Entities may refer to classes, properties, or individuals for which we want to find matches in the input ontologies.

Definition 2 (Relationship Matrix)

This relationship matrix, denoted $R(r_{ij})$, represents the relationship between ontologies S and T, i.e., r_{ij} indicates the similarity relationship between entity s_i in S and entity t_j in T. Using R, we define another matrix (see Def. 3), called the *similarity matrix* $L(l_{ij})$, which captures a different relationship between S and T. In the matrix $R(r_{ij})$, $s_i\ r\ t_j$ says that entity s_i in the source ontology S matched with entity t_j in the target ontology T, based on relationship r, where r could be any of the existing similarity measuring methods, such as string similarity measure, linguistic similarity measure, … etc.

$$R(r_{ij}) = \begin{bmatrix} s_1 r t_1 & s_1 r t_2 & \cdots & s_1 r t_j \\ s_2 r t_1 & s_2 r t_2 & \cdots & s_2 r t_j \\ \cdots & \cdots & \cdots & \cdots \\ s_i r t_1 & s_i r t_2 & \cdots & s_i r t_j \end{bmatrix}$$

Definition 3 (Similarity Matrix)

This relational matrix, denoted $L(l_{ij})$, includes entries in the interval [0,1], which are called the *similarity coefficients* and denote the degree of similarity between s_i and t_j. R and L are n×m matrices.

$$L(l_{ij}) = \begin{bmatrix} l_{11} & l_{12} & \cdots & l_{1j} \\ l_{21} & l_{22} & \cdots & l_{2j} \\ \cdots & \cdots & \cdots & \cdots \\ l_{i1} & l_{i2} & \cdots & l_{ij} \end{bmatrix}$$

Moreover, the similarity matrix $L(l_{ij})$ captures the similarity coefficients between E^S and E^T based on the defined relationship matrix $R(r_{ij})$. For example, if $R(r_{ij})$ is defined to be a string similarity relationship between E^S and E^T, then the similarity coefficient l_{ij} in the similarity matrix $L(l_{ij})$ says that entity s_i in the source ontology S matched with entity t_j in the target ontology T based on a string similarity measure, with a similarity coefficient l_{ij}. As a result, for each $R(r_{ij})$, we compute its $L(l_{ij})$.

Definition 4 (Matching Matrix)

A matching matrix, denoted $Map_{0\text{-}1}$, is an n×m matrix with entries $r_{ij} \in \{0,1\}$. If $r_{ij} = 1$, it means that s_i and t_j are "matchable." They are not matchable if $r_{ij} = 0$.

Definition 5 (Matching Space)

All the possible assignments for the matching matrix form a *matching space*, also called the *mapping space*. Every assignment is a state in the matching space, i.e. a state represents a solution for ontology matching.

The following example illustrates the above concepts and terms.

Example 8. Let S and T be a given pair of ontologies, and $E^S = \{s_1, s_2, \ldots, s_n\}$ and $E^T = \{t_1, t_2, \ldots, t_m\}$ be the sets of entities. A matching matrix $Map_{0\text{-}1}$ indicates the similarity relation between the elements of E^S and E^T. The number of relationship matrices $Map_{0\text{-}1}$

is 2^{nxm}, i.e. the matching space has 2^{nxm} states. These matrices form the matching space. For instance, when Map_{0-1} is 2×2, the matching space would have 16 states. Some of these mapping states are as follows, in which the rows are entities in S and the columns are entities in T.

$$\left(\begin{bmatrix}0&0\\0&0\end{bmatrix},\begin{bmatrix}1&0\\0&0\end{bmatrix},\begin{bmatrix}1&1\\0&1\end{bmatrix},\cdots,\begin{bmatrix}1&1\\1&1\end{bmatrix}\right)$$

The first matrix indicates no mapping. The third matrix indicates that entity s_1 is matched with t_1 or t_2, and that s_2 is matched with t_2, etc.

4.3 A Multi-Match Algorithm

The main steps of the Multi-match algorithm (MMA) are shown in Fig. 15. The algorithm is mainly divided into two phases. In phase 1, which is the initialization phase, an initial assignment for the matching matrix *Map* is provided, as well as similarity functions to evaluate the similarity matrix. Phase 2 of MMA, which is the search phase, is an iterative refinement for the *Map* matrix. The algorithm iteratively constructs matching searches for entities in both S and T (see illustrative example in the next section). Then, the *Map* matrix will be evaluated according to the (re)used similarity matching techniques, such as name and linguistic techniques, and finally the *Map* matrix with the highest evaluation value will be suggested to the user.

If we only search with one matching technique, the algorithm behaves as a regular similarity procedure and is considered as a single matcher; otherwise, it is indeed a multi-matcher. This design is useful as it provides a flexible and convenient way to use various relevant information about input ontologies, and to combine feasible matching methods

to obtain better results than those obtained by each individual method. The method can deploy any desired search algorithm.

```
Algorithm MMA(S,T)
Phase 1 Initialization
    Pick an initial assignment matching matrix.
    /* For example, let diagonal elements in Map be
    equal to 1, and so on.*/
    Use the similarity functions to evaluate the
    similarity matrix.
Phase 2 Search Matching techniques
    begin
      Enter a similarity matching technique
      /* such as the name matching technique */
      Evaluate an intermediate matching state
       begin
         Enter another similarity matching technique
         /* such as the linguistic matching technique */
         Evaluate an intermediate matching state
           Begin
              ...
            /* various available matching techniques,
             i.e. many feasible matching techniques */
           end;
       end;
       if the intermediate matching state is not
          the final solution
          /* the matching result does not satisfy
             the evaluation function */
          then use it as an intermediate solution
               in the next iteration;
          if the matching state satisfies the
                evaluation function
            then return the final solution
end;
```

Figure 15: Multi-Matching Algorithm (MMA) description

4.4 Illustrative Example

In this section, we illustrate our solution approach. Fig. 16 shows two sample taxonomies for Computer Science Departments (CSDs) of different universities. We have to integrate the ontologies into a single ontology [Alasoud *et al.*, 2005]. To reduce the manual work involved, we use a matching algorithm to identify the matching entities, and then help the

middleware to integrate the schemas. For ease of presentation, we use simple, small taxonomies.

As can be seen in Fig. 16, the representations of the source ontology S and the target ontology T are different. Here, entities s_1, s_2, s_8 and t_1, t_2, t_6 are *concepts*, which are high-level entities in the ontologies. Other entities are *properties*. Given that $|S|=13$ and $|T|=9$, the number of states is $2^{13\times9}$, indicating the number of possible matching results. Clearly, it is not practical to evaluate all the mapping states to compare the relationships and similarities between the entities.

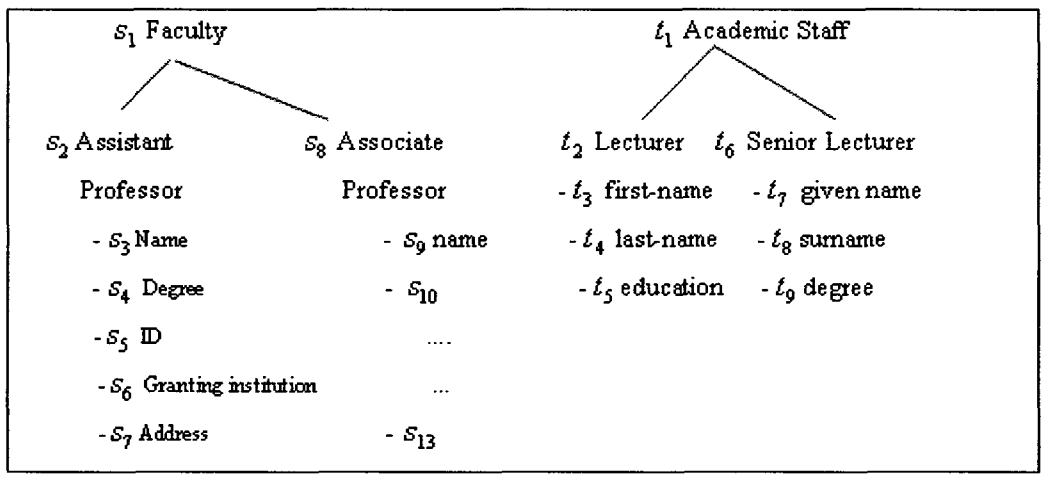


Figure 16: CSDs Ontologies

We thus need to find ways to reduce the search space. In this example, we allow *concepts* in S to be compared only with *concepts* in T, and, accordingly, the other entities in S, such as properties and instances, can be matched only with the corresponding entities in T.

We only use two similarity measures to compare the entities in S and T, name similarity (Levenshtein distance) and linguistic similarity (WordNet). We thus obtain the following similarity matrices for the *concepts*.

$$L_{name\_concept} = \begin{bmatrix} s_1rt_1 & s_1rt_2 & s_1rt_6 \\ s_2rt_1 & s_2rt_2 & s_2rt_6 \\ s_8rt_1 & s_8rt_2 & s_8rt_6 \end{bmatrix} = \begin{bmatrix} 0.15 & 0.25 & 0.14 \\ 0.38 & 0 & 0.14 \\ 0.31 & 0.11 & 0.21 \end{bmatrix}$$

$$L_{ling\_concept} = \begin{bmatrix} s_1rt_1 & s_1rt_2 & s_1rt_6 \\ s_2rt_1 & s_2rt_2 & s_2rt_6 \\ s_8rt_1 & s_8rt_2 & s_8rt_6 \end{bmatrix} = \begin{bmatrix} 1 & 0.07 & 0.06 \\ 0.07 & 0.2 & 0.1 \\ 0.08 & 0.06 & 0.05 \end{bmatrix}$$

When an assignment is found for the matching state, we check the similarities of entities to see whether they exceed a user-defined threshold, denoted as *th*. In this example, we use the following evaluation function:

$$v = \left| \left(Map_{0-1} \cdot L \right) / k \right| = \left| \sum_{i=1}^{n} \sum_{j=1}^{m} Map_{0-1}(i,j).L(i,j) \Big/ \sum_{i=1}^{n} \sum_{j=1}^{m} Map_{0-1}(i,j) \right|, \ and \ v \geq th$$

$k \geq min(n,m)$ is the number of matched pairs, n is the number of entities in S, and m is the number of entities in T.

The choice of threshold value is application dependent and should be adjusted and suitably chosen for each matcher.

We now provide a description of the search process. The initial state of the mapping matrix is a zero matrix. Then, if the search process exceeds the given maximum number of iterations, the maximum similarity states (Map_{max}) will be offered as the final mapping result. Also, we need to set the additive constraints in the search process. In this example, since the number of concepts in S is equal to that in T, we assume that the ontologies S and T have been fully matched. So, the mapping states of concepts now

include 6 entries, e_1, e_2, …, e_6, as shown in Fig. 17.

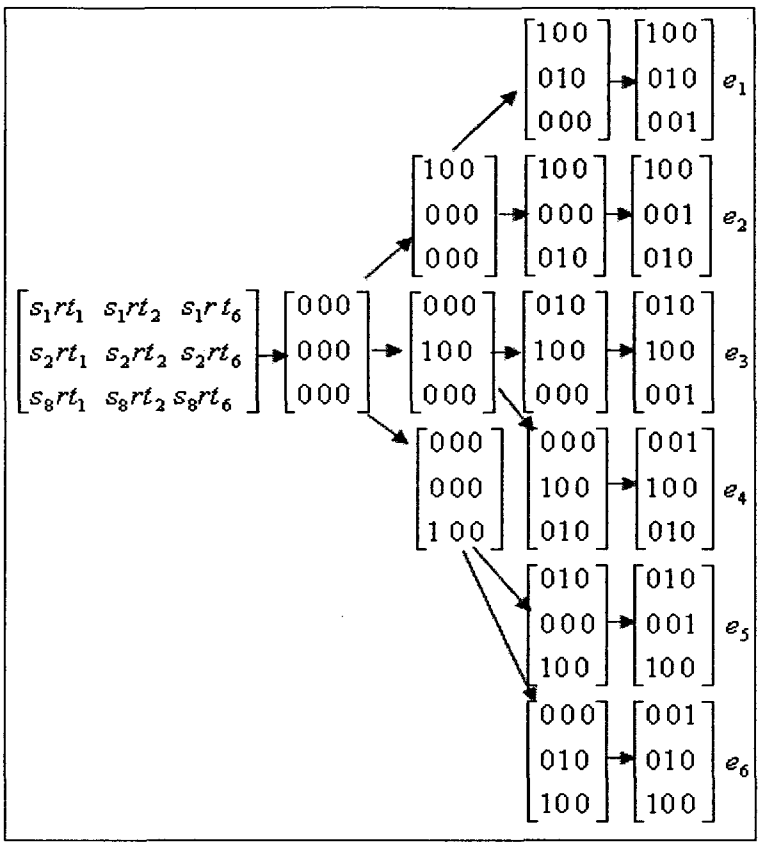


Figure 17: Searching in the matching space

As shown in Table 1, e_1 indicates the optimal matching result. Also, we can see that different values for name similarity v_1 and linguistic similarity v_2 for each entry are determined as follows. We show this for e_1:

$$v_1 = \begin{bmatrix} 0.15 & 0.25 & 0.14 \\ 0.38 & 0 & 0.14 \\ 0.31 & 0.11 & 0.21 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \Big/ 3$$

$$v_2 = \begin{bmatrix} 1 & 0.07 & 0.06 \\ 0.07 & 0.2 & 0.1 \\ 0.08 & 0.06 & 0.05 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \Big/ 3$$

The following table shows the individual and combined similarity match results for each entry state.

State	Name v_1	Concept v_2	Name + Concept $v_1 + v_2$	Normalized cost $v = (v_1 + v_2)/2$
e_1	0.12	**0.42**	0.54	**0.27**
e_2	0.13	0.38	0.51	0.25
e_3	**0.28**	0.06	0.34	0.17
e_4	0.21	0.06	0.27	0.13
e_5	0.23	0.08	0.31	0.15
e_6	0.15	0.11	0.26	0.13

Table 1: Individual and combined similarity match results

Note here that if we only use the name similarity technique, the mapping result would be e_3. In the same way, if we only use the linguistic technique, we would obtain e_1 as the result. Also, using $Map_{name\_concept}$, $Map_{ling\_concept}$, and the threshold value $th_{concept}$, we obtain the final solution. Consequently, the output result state e_1 means that we matched n concepts from the source ontology S to m concepts in the target ontology T. That is, s_1 is matched with t_1, s_2 with t_2, and s_8 with t_6. Accordingly, the algorithm matches the properties and/or instances of each matched pair of the concepts.

Finally, the mapping result will be introduced in OWL format. OWL can be considered as a language for expressing correspondences between ontologies. As a matter of fact, the equivalentClass and equivalentProperty primitives have been introduced for relating elements in ontologies. For example, the following OWL ontology fragment

<owl:Class rdf:about = "&o1;#Faculty">

<owl:equivalentClass rdf:resource = "&o2;#Academic Staff">

</owl:Class>

<owl:Class rdf:about = "&o1;#Assitant Professor">

<owl:equivalentClass rdf:resource = "&o2;#Lecturer">

</owl:Class>

<owl:Class rdf:about = "&o1;#Associate Professor">

<owl:equivalentClass rdf:resource = "&o2;#Senior Lecturer">

</owl:Class>

4.5 Experiments and Results

In our evaluation, we used three pairs of ontologies:

1. The MIT bibtex ontology [Knouf, 2003] (which contains 43 named classes, 22 object properties, and 24 data properties) and the UMBC [UMBC-Ontology] publication ontology (which contains 15 named classes, 5 object properties, 27 data properties), both of which are publicly available.
2. Computer ontologies (the first onltology contains 17 named classes, 11 object properties, 15 data properties, and the second one contains 15 named classes, 10 object properties, and 14 data properties).
3. Ontologies about computer science departments (the first ontology contains 16 named classes, 12 object properties, 10 data properties, and the second ontology contains 18 named classes, 14 object properties, 9 data properties). We created the second and third pairs of ontologies.

We consider the matching of classes and properties based on their labels and the taxonomy structures of the input OWL-ontologies. As match quality measures, we use

the following indicators: *precision, recall, and F-measure*. Fig. 18 illustrates the idea of the matching comparison.

- ***Precision*** is a value in the range [0, 1]; the higher the value, the fewer the wrong mappings (false positives) computed.

$$\Pr ecision = \frac{|B|}{|B| + |C|}$$

where B represents true positives, and C false positives.

The precision measure could also be defined as follows:

$$\text{precision} = \frac{\text{number\_of\_correct\_found\_alignments (by tools)}}{\text{number\_of\_found\_alignments (by tools)}}$$

- ***Recall*** is a value in the range [0, 1]; the higher this value, the smaller the set of correct mappings which are not found (also called true positives).

$$\text{Re}call = \frac{|B|}{|A| + |B|}$$

where A is the set of false negatives. The recall measure could also be defined as follows:

$$\text{recall} = \frac{\text{number\_of\_correct\_found\_alignments (by tools)}}{\text{number\_of\_existing\_alignments (by experts)}}$$

- ***F-measure*** is a value in the range [0,1], which is a global measure of the matching quality. For this, we use the harmonic mean of precision and recall [Do *et al.*, 2002].

$$\text{F\_Measure} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

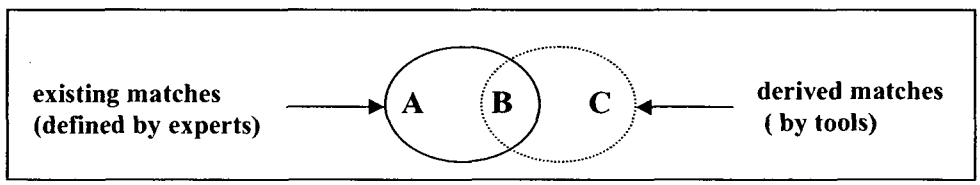


Figure 18: Comparing existing matches and derived matches

In our method, we are concerned with providing a ground for evaluating the quality of match results. For this, we have determined and used expert matches for all the input pairs of ontologies. The results produced by the match algorithm are compared with these expert matches.

The evaluation results are shown in Figs. 19, 20, and 21. From the point of view of the quality of the matching results, it is clear that MMA outperforms each individual technique.

The key point to note in MMA is that, for each entity from the source ontology, it gives only one corresponding entity match in the target ontology. This enables MMA to achieve, in these cases, high precision and recall numbers. For instance, in the case of the computer ontologies, since both ontologies contain either the same names for the corresponding entities, or they use totally different names, we see that string-based techniques provided a high precision rate (no wrong matches returned to the user), that is, the concept 'Computers' in the source ontology is matched to 'Computers' in the target ontology. However, string-based techniques provided a low recall rate because they failed to identify semantic mappings. For example, the string-based techniques missed to match the concepts (PC, Price, and Monitor) in the source ontology to their corresponding concepts (desktop, cost, and display) in the target ontology.

The linguistic-based techniques showed a low precision (some even returned incorrect mappings to the user). For instance, the concept "Computers" in the source ontology will also be matched with the desktop and laptop concepts in the target ontology. Another reason for the low recall rate is that it gives a large set of wrong mappings compared to the expert defined matches. The MMA algorithm, on the other hand, benefits from these

existing techniques. Since each concept from the source ontology will be matched with only one concept from the target ontology, the concept "Computers" in the source and target ontologies will be matched to each other. Moreover, the "PC", "Price", and "Monitor" concepts in the source ontology will be matched to the "desktop", "cost", and "display" concepts in the target ontology. Consequently, the MMA algorithm produces a better final result for its higher precision and recall rates.

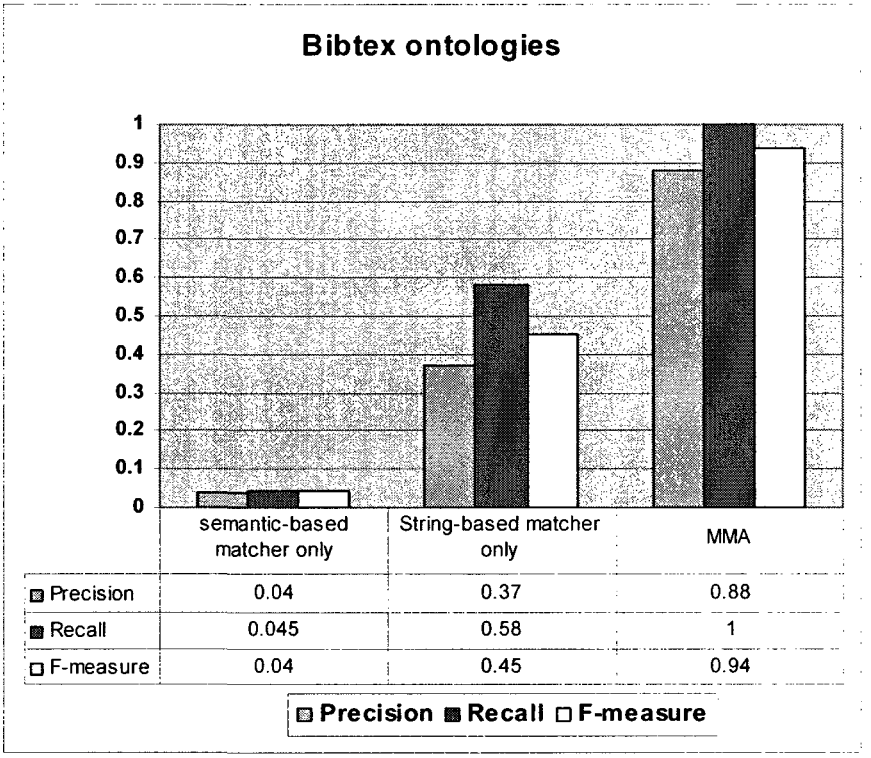


Figure 19: Results using Bibtex ontologies

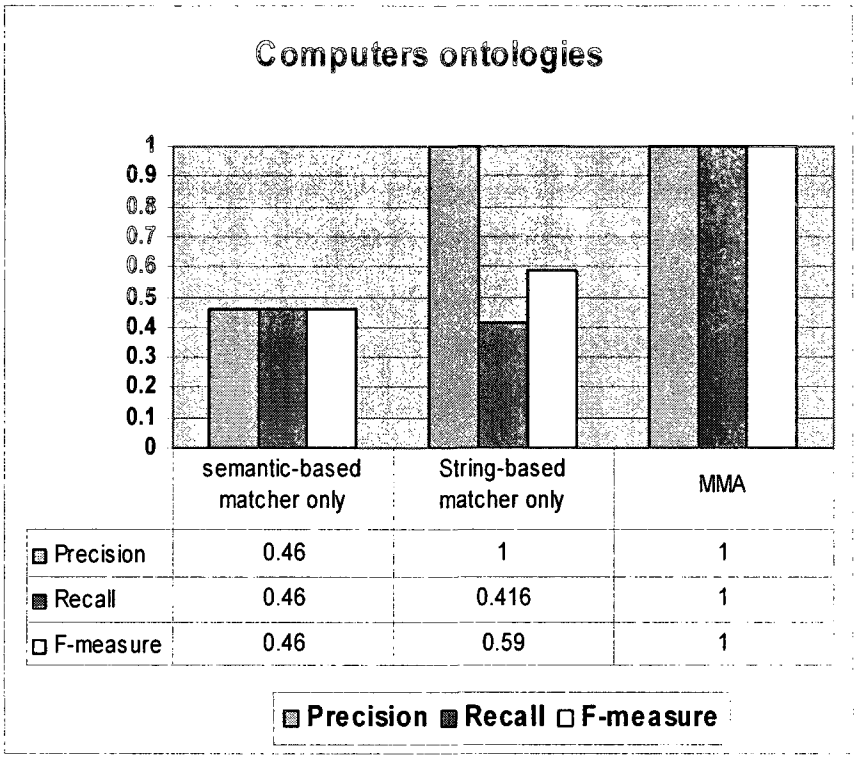


Figure 20: Results using Computer ontologies

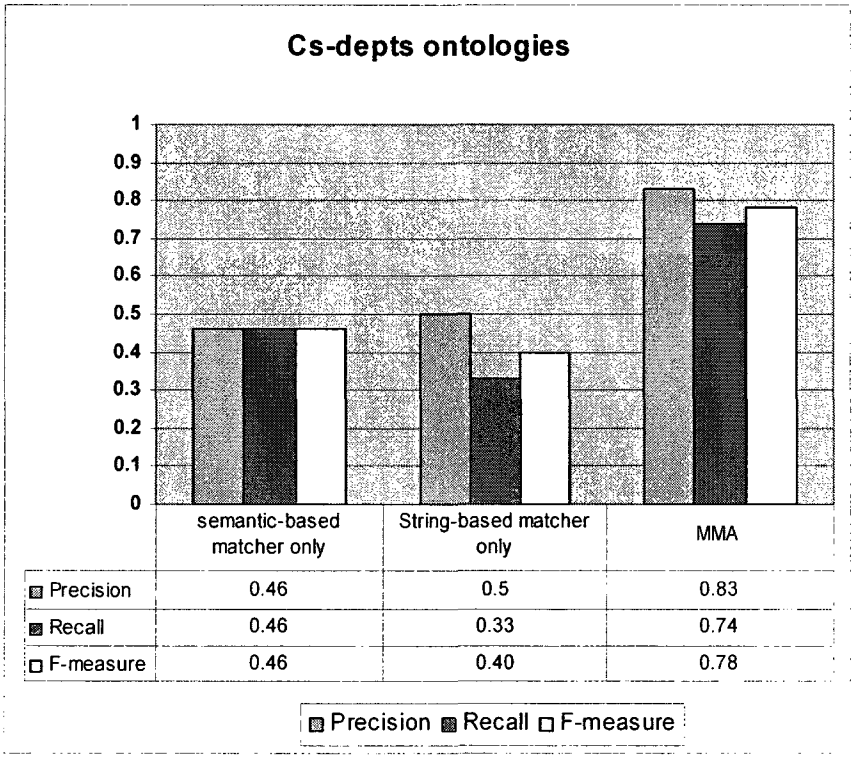


Figure 21: Results using Computer science departments' ontologies

4.6 Summary

In this chapter, we introduced the multi-matching strategy, which could be used to support the hybrid approach. The following is a summary of the chapter:

- We introduced the multi-match strategy to support the hybrid approach. The matching step in this proposed approach finds and relates the correspondences between the entities in both the global ontology and the local (source) ontologies. For this, we introduced the Multi-Matching Algorithm (MMA). The important features of this algorithm are that it benefits from existing individual matching techniques and it helps "combine" their match results to provide enhanced ontology matching. Furthermore, it matches a collection of n elements in the source ontology to a collection of m elements in the target ontology.
- We developed a prototype of the proposed MMA, and tested it using different input pairs of ontologies. Our results indicated that the proposed framework yields improved match results, as compared to individual match techniques, in terms of precision, recall, and F_measure.

5. Extending the Multi-Matching Strategy

This chapter introduces, in Section 1, the multi-level extension of MMA, called MLMA [Alasoud *et al.*, 2007], which assumes that the collection of similarity measures are partitioned by the user, and that there is a partial order to the partitions, also defined by the user. Section 2 provides the neighbor search strategy [Alasoud *et al.*, 2008] which uses the MLMA as a backbone and performs a neighbor search to find the correspondences between entities in the given ontologies. Section 3 discusses the advantages of incorporating the reasoning techniques in order to achieve a satisfactory matching result [Alasoud *et al.*, 2009]. A summary is given in Section 4.

5.1 Multi-level matching strategy

This section introduces an ontology matching approach based on the idea of a multi-level

match algorithm, in which each level may use different similarity measure(s).

Fig. 22 illustrates the main idea of the multi-level method for the case of two levels. It shows various similarity measures $\{m_1, m_2 \ldots m_l\}$ divided into two groups, each of which is applied at one level.

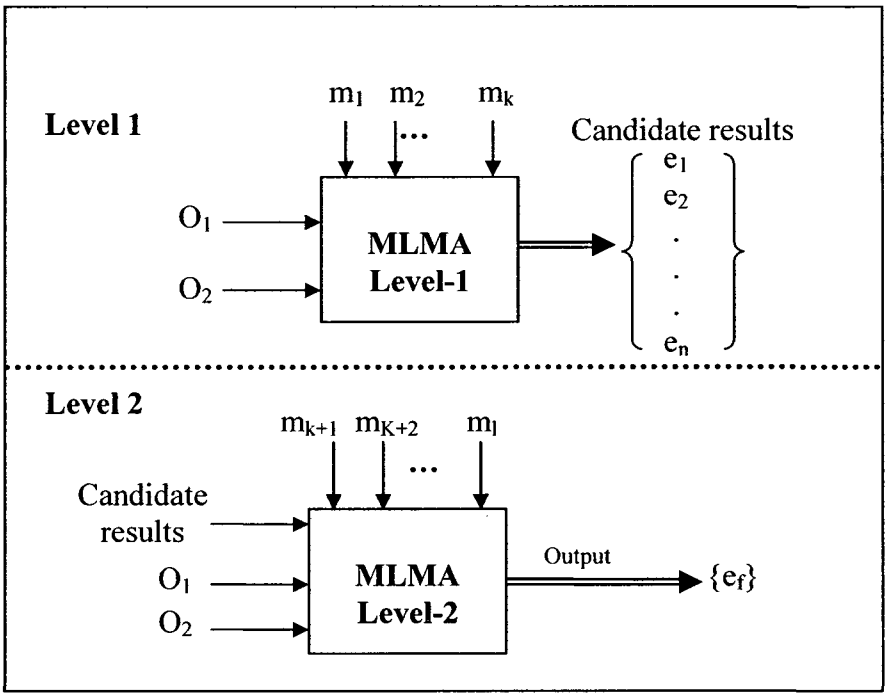


Figure 22: A schematic description of the multi-level method

For ease of presentation, suppose we use three similarity measures divided into two levels. The name and linguistic similarity measures are applied at the first level. We then apply a structural similarity measure at the second level on the resulting candidate states $\{e_1, e_2 \ldots e_n\}$ from the first level. As the output, this method will produce the state which has the highest similarity score value. Moreover, the resulting mapping state $\{e_f\}$ is measured based on its rich structure and the highest number of correspondences between the input ontologies. As a result, the order of applying the similarity measures will not affect the overall quality, see Appendix C.

5.1.1 Tradeoff between structure and size of the mapping states

Many similarity measures have been introduced for a keyword set representation of text, such as {ontology, matching, algorithm}. Examples of such methods include the Dice coefficient, the Jaccard coefficient, and the Cosine coefficient [Rasmussen, 1992]. The Dice coefficient is defined as follows:

$$S_{T_1,T_2} = (2|T_1 \cap T_2|)/(|T_1| + |T_2|)$$

where $|T_i|$ is the number of terms in set T_i, and $|T_1 \cap T_2|$ is the number of common terms in T_1 and T_2. We will use this to develop our structure similarity measure.

Let O_1 and O_2 be a pair of ontologies represented as labeled graphs, and O_{MMA} be the ontology induced by the similarity result S_{MMA}, obtained by applying the basic MMA match algorithm (which combines the similarity measures in a single step/level operation). Let S_{strc} be the structural similarity measure S, calculated as follows, which defines the similarities between the concepts in O_{MMA} and those in the original input ontologies O_1 and O_2.

$$S_{strc} = 2\,|r(O_{MMA})| / (|r(O_{MMA}(O_1))| + |r(O_{MMA}(O_2))|)$$

where $|r(O_{MMA})|$ is the number of relationships in the ontology O_{MMA}, and $|r(O_{MMA}(O_i))|$ is the number of relationships in the "immediate" neighborhood of O_{MMA} in O_i. This neighborhood of O_{MMA} consists of the relationships of O_i with at least one end (one of the edge's ends) belonging to O_{MMA}.

We can view S_{strc} as a complementary measure to the output of MMA, applied at the second level. This is justified as follows.

- The structure similarity S_{strc} is mainly based on the presence of common concepts between the matched ontologies, induced by the states calculated by MMA.
- The similarity degree between the matched ontologies may still exist, even when there is no structural match in the result of MMA, i.e., when $S_{strc} = 0$.

Accordingly, the combined similarity measure S is relative to S_{MMA}, and should not be zero in the case where $S_{strc} = 0$. We further "smooth" the effect of S_{strc} as follows:

$$S = S_{MMA} + (x * S_{strc}), \text{ where } x = (1 - S_{MMA})$$

In the combined similarity S, suppose $S_{strc} = 0$. This then means that the value S depends only on the similarity measure of MMA. On the other hand, if $S_{strc} = 1$, the neighborhood of the concepts matched by MMA is the same, and, consequently, S will take the maximum value. Also, since $S_{MMA} + x = 1$, we have that $x = 1 - S_{MMA}$, representing the complementary part of the information described in the relationships between the concepts in a desired state found by MMA.

As we do not want to miss a found matching state that includes a large number of concepts matched, S_{MMA} provides possible good matches in the input ontologies with their similarity degrees. The extended method will determine the same collection of matched states, but with better differentiation among them, by taking into account the structural measures in the second level. An extension of this two level method to a multi-level method is straightforward when the user can identify which measure(s) could or should be applied at which level.

5.1.2 The MLMA Algorithm

There are many algorithms for matching techniques. The notion of multi-match "combines" all techniques involved into a single, unified method. By searching from technique to technique, the matching algorithm can eventually find a reasonable solution. The main idea of our Multi-Level matching algorithm is sketched in Fig. 23.

```
Algorithm MLMA(S,T)
Phase 1 Initialization
    Pick an initial assignment matching matrix.
    /* For example, let diagonal elements in Map be equal to 1,
    and so on.*/
    Use the similarity functions to evaluate the similarity
    matrix.
Phase 2 Search Matching techniques
    begin
      Enter a similarity matching technique
      /* such as the name matching technique */
      Evaluate an intermediate matching state
       begin
          Enter another similarity matching technique
          /* such as the linguistic matching technique */
          Evaluate a better intermediate matching state
            Begin
               ...
              /* various available matching techniques,
              i.e. many feasible matching techniques */
             end;
        end;
        if the intermediate matching state is not
           the final solution
           /* the matching result does not satisfy
              the evaluation function */
           then use it as an initial solution in the
            next iteration;
            if the matching state satisfies the
                 evaluation function
              then it is a candidate for the final state
    end;

Phase 3 Apply the Complementary measures
       /* Apply the structure similarity measure to the
    candidate states of phase 2, and return the final state */

end;
```

Figure 23: The Multi-Level Match Algorithm

The MLMA algorithm is an update to the MMA algorithm Fig. 15. It is divided into three phases. In phase 1, which is the initialization phase, an initial assignment for the

matching matrix *Map* is provided, as well as the functions of similarity to evaluate the relationship matrix. In phase 2, MMA performs a search operation, which is an iterative refinement of the *Map* matrices. In phase 3, the resulting mapping states from MMA will be qualified based on the connectivity among their concepts. Then, the best possible final state will be offered to the user.

5.1.3 Illustrative Scenario

The following example illustrates the main idea of the MLMA. For ease of presentation, we use simple and small taxonomies. Fig. 24 shows two sample taxonomies for Researchers (O_1) and Students (O_2) of different universities.

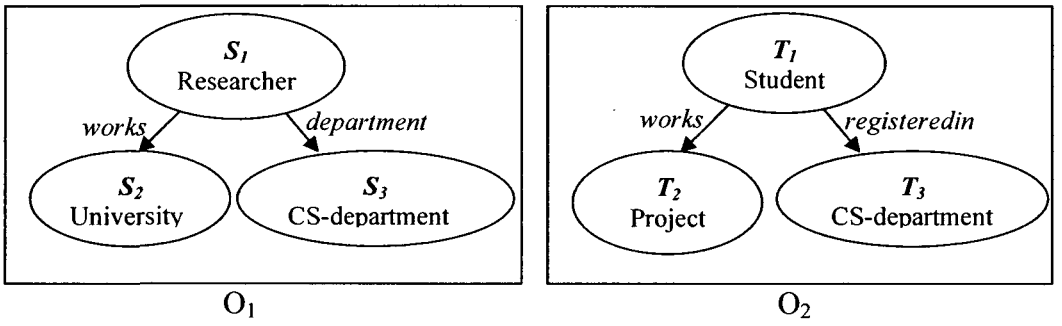


Figure 24: Researchers (O_1) and Students (O_2) ontologies

The goal is to integrate the ontolgies into a single ontology. To reduce the manual work involved, we use a matching algorithm to identify matching entities, and then help the middleware to integrate the schemas.

As can be seen in Fig. 24, entities S_1, S_2, S_3, and T_1, T_2, T_3 are *concepts*, which are high-level entities in the input ontologies O_1 and O_2.

For this explanation, we only use two different similarity measures to compare the entities in S and T: name similarity (Levenshtein distance) and linguistic similarity (WordNet). This yields the following similarity matrices for the *concepts*.

$$L_{name\_concept} = \begin{bmatrix} 0.0 & 0.2 & 0.308 \\ 0.2 & 0.2 & 0.0 \\ 0.308 & 0.308 & 1.0 \end{bmatrix} \qquad L_{ling\_concept} = \begin{bmatrix} 0.75 & 0.181 & 0.307 \\ 0.4 & 0.181 & 0.0 \\ 0.307 & 0.166 & 1.0 \end{bmatrix}$$

We use the evaluation function v defined in Section 4.4, which measures the threshold value for the states obtained in the second phase of the MLMA algorithm. The outputs are states $e_1, e_2, \ldots, e_6$, shown in Fig. 25, which are represented as labeled directed graphs. It shows that e_1 has obtained one common edge, and no common edges have been obtained by the other states.

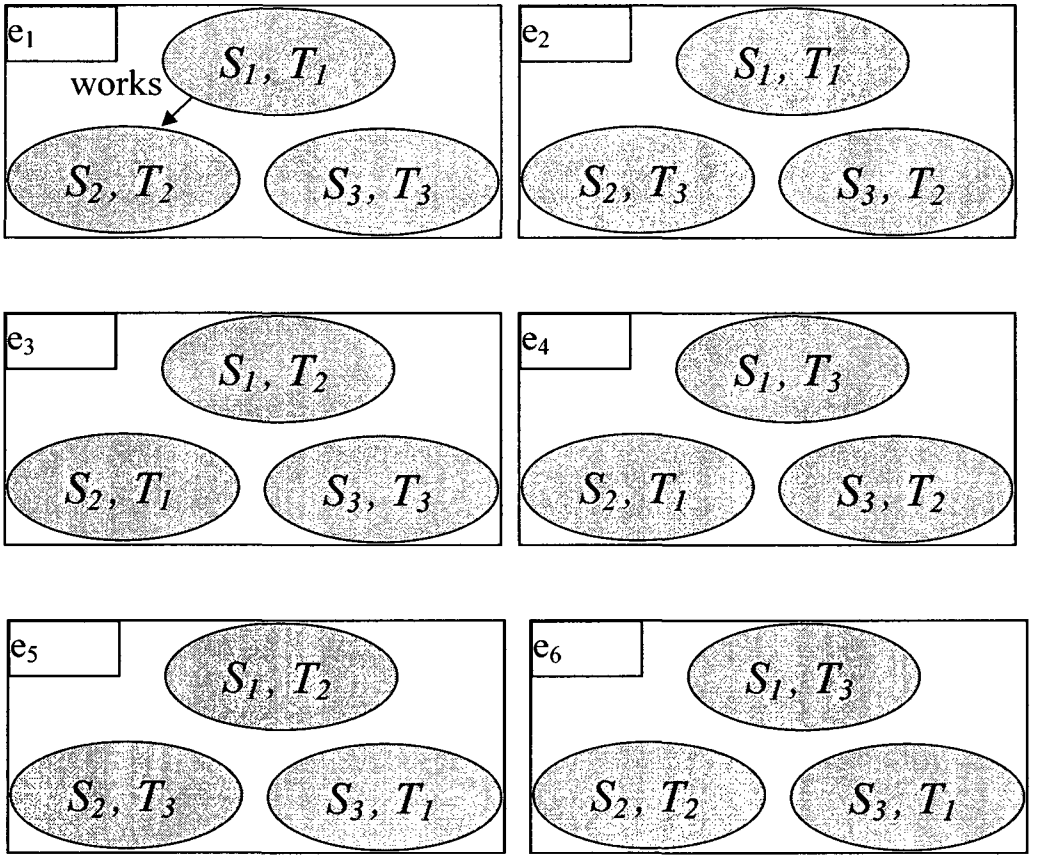


Figure 25: The states determined by MMA

As shown in Table 2, e_1 is the "best" match found. Using the formula for computing 'v' values for the name and linguistic similarity matrices $L_{name\_concept}$ and

$L_{ling\_concept}$, we obtain values 0.4 and 0.64 for name similarity v_1 and linguistic similarity v_2, respectively. Each entry is determined as follows. We show this for e_1:

$$Map_{0-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \; v_1 = \begin{bmatrix} 0.0 & 0.2 & 0.308 \\ 0.2 & 0.2 & 0.0 \\ 0.308 & 0.308 & 1.0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \Big/ 3 \text{ , and } \; v_2 = \begin{bmatrix} 0.75 & 0.181 & 0.307 \\ 0.4 & 0.181 & 0.0 \\ 0.307 & 0.166 & 1.0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \Big/ 3$$

Then, v is computed by normalizing the cost of v_1 and v_2 as follows:

$$V(e_m) = \sum_{i=1}^{n} w_i v_i(e_m) \text{ , and for } e_1, \; V(e_1) = (w_1 * v_1) + (w_2 * v_2)$$

where v_i is the matching score obtained by the similarity measuring technique i, w_i is the weight of the similarity measuring technique i, and $v(e_m)$ is the score for state e_m. Consequently, in this example, we used $w_1 = w_2 = 0.5$.

To measure S_{strc} for the mapping state e_1, we proceed as follows:

- The number of common relationships that connect common concepts to other common concepts is 1 *"works"*.
- The number of relationships in O_1, with at least one end belonging to the common concepts is 2 *"works, department"*.
- The number of relationships in O_2, with at least one end belonging to the common concepts is 2 *"works, registeredin"*. As a result, we obtain $S_{strc} = ((2*1)/(2+2)) = 0.5$.

Table 2 shows the individual and combined similarity matching results for each state e_i. Note that, using only the name similarity technique, the mapping result would be e_3. In the same way, using only the linguistic technique, we would obtain e_1. Also, using $Map_{name\_concept}$, $Map_{ling\_concept}$, and the threshold value *th,* we obtain S_{MMA}. Consequently, the output result is state e_1, which indicates that we matched the *n* concepts in the source ontology S to the m concepts in the target ontology T. To be more precise, s_1 is matched

with t_1, s_2 with t_2, and s_3 with t_3. Accordingly, the algorithm matches the properties and/or instances of each pair of matched concepts.

Level 1				Level 2	
State	**Name** v_1	**Concept** v_2	**S_{MMA}** Normalized cost $v = (v_1 + v_2)/2$	**S_{strc}**	$S = S_{MMA} + (x * S_{strc})$
e_1	0.4	**0.64**	**0.52**	0.5	**0.77**
e_2	0.103	0.305	0.204	0.0	0.204
e_3	**0.466**	0.527	0.497	0.0	0.497
e_4	0.272	0.291	0.282	0.0	0.282
e_5	0.169	0.163	0.166	0.0	0.166
e_6	0.269	0.265	0.267	0.0	0.267

Table 2: Two-level individual and combined similarity match results

We can also notice the recognized quality performance of the structure measure and how the similarity values S_{MMA} and S_{strc} are combined to compute the final measure S. This scenario indicates that S is always greater than or equal to S_{MMA} for our similarity measures. This reveals that S increases the weight of states with connected common concepts, as opposed to the states of common concepts that are not connected.

As a result, using S, we gain the following:

- *S* maintains as many matched concepts as possible.
- *S* can improve the matching quality of S_{MMA} if the ontologies that are to be matched are structurally similar. However, it will not affect S_{MMA} at all if there is no structure similarity in the given input ontologies.

5.1.4 Experimentation and Results

The quality comparison between the basic MMA and MLMA methods is shown in Fig. 26. As there are structural similarities between the ontologies in the first and second pair, the MLMA increases the matching quality of their final states. Even though the ontologies in the third test pair are structurally dissimilar, the MLMA maintains the matching quality of the MMA without any changes, as desired.

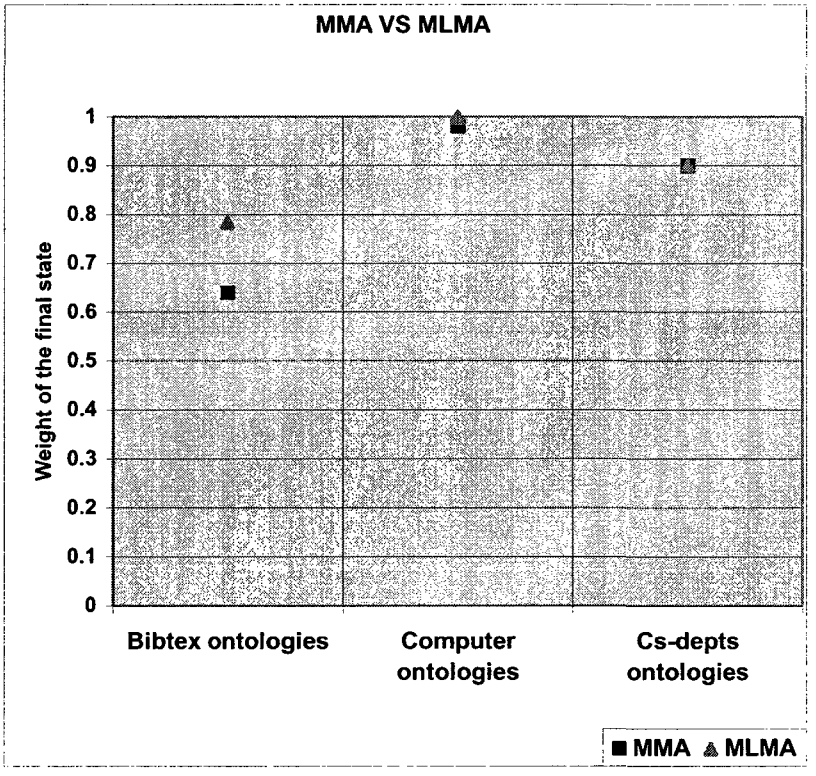


Figure 26: Quality comparison between the basic MMA and MLMA methods

5.2 Neighbor search strategy

A neighbor search strategy uses the multi-level matching technique we proposed earlier as a backbone and performs the neighbor search to find the correspondences between entities in the given ontologies. An important feature of this algorithm is its fast convergence, while providing quality results, obtained by searching the neighborhood of some initial match result. We introduce a neighbor search algorithm, with a proper initialization as an optimization for the multi-level matching algorithm, which decreases the computation time. We will refer to this optimized version of the MLMA algorithm as MLMA$^+$ [Alasoud *et al.*, 2008].

5.2.1 Motivating Example

To illustrate the main idea of the neighbor search algorithm, consider the simple examples shown in Fig. 27, which are taxonomies for computer ontologies O_1 and O_2.

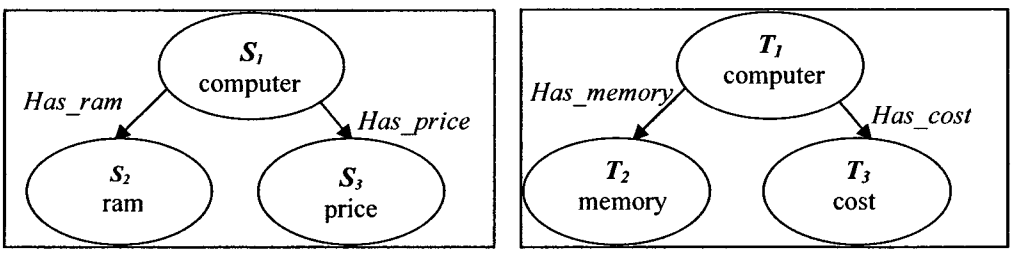


Figure 27: Computer Ontology Examples

As can be seen in Fig. 27, entities S_1, S_2, S_3, and T_1, T_2, T_3 are *concepts*, which are high-level entities in the input ontologies. Here $|S|=3$ and $|T|=3$, and hence the size of the matching space would be $2^{3\times3}$. In general, our goal is to find a way to reduce the search space for larger ontologies.

There are several methods to measure similarities between two entities, including string similarity and linguistic similarity. We start with some similarity measure(s) in the first level as the initial state, and then perform the neighbor search algorithm. The search process focuses on the given initial state and expands the search through the neighbors (Section 5.2.3 gives an illustrative example,).

5.2.2 The Neighbor Search Algorithm

The neighbor search algorithm has three phases, described in Fig. 28.

```
Algorithm Match(S, T)
begin
     /* Initialization phase */
     K← 0 ;
     St0 ←preliminary_matching_techniques(S,T);
     Stf ←St0 ;
     /* Neighbor Search phase */
     St ←All_Neighbors(Stn);
      While (K++ < Max_iteration) do
      /* Evaluation phase */
            If score(Stn) > score(Stf) then
                  Stf ←Stn;
            end if
            Pick the next neighbor Stn ∈ St;
            St ← St - {Stn};
            If  St = Ø then Return Stf ;
      end
      Return Stf ;
end
```

Figure 28: The Neighbor Search Algorithm

First, in the initialization phase, a partial set of similarity measures is applied to the input ontologies to determine a single initial state St_0 for the search algorithm. In the second phase, we search in the neighborhood of this initial state St_0. Its neighbors are the mapping states that can be computed either by adding to or removing from St_0 a couple of vertices, obtained by toggling a bit in the similarity matrix L. So, the total number of the

neighbor states will be $n*m$. We evaluate the neighbor states using the evaluation function v defined in Section 4.4. In the third phase (the evaluation phase), the algorithm will apply the next level(s) similarity techniques in order to find St_f, the best possible matching state solution.

5.2.3 Illustrative Example

Following our running example, shown in Fig. 27, we are given that entities S_1, S_2, S_3, and T_1, T_2, T_3 are *concepts*, which are high-level entities in the input ontologies.

For ease of explanation, we only use three different similarity measures applied in two different phases. We use two similarity measures in the first phase to compute the initial state St_0: name similarity (Levenshtein distance) and linguistic similarity (WordNet). This yields the following similarity matrices for the concepts in this example.

$$L_{name\_concept} = \begin{bmatrix} 1.0 & 0.1 & 0.375 \\ 0.125 & 0.167 & 0.0 \\ 0.125 & 0.0 & 0.0 \end{bmatrix} \qquad L_{ling\_concept} = \begin{bmatrix} 1.0 & 0.7 & 0.154 \\ 0.8 & 0.9 & 0.166 \\ 0.6 & 0.315 & 1.0 \end{bmatrix}$$

Suppose $th \geq 0.45$. After normalizing the cost of the two similarity matrices, we get the matrix L. Then L is transformed into the matching matrix $Map_{0\text{-}1}$. Note that we are using $Map_{0\text{-}1}$ and St_n as synonyms.

$$L = \begin{bmatrix} 1.0 & 0.4 & 0.265 \\ 0.463 & 0.534 & 0.083 \\ 0.363 & 0.158 & 0.5 \end{bmatrix} \qquad Map_{0\text{-}1} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The binary matrix $Map_{0\text{-}1}$ above corresponds to state $St_0=\{(s_1, t_1), (s_2, t_1), (s_2, t_2), (s_3, t_3)\}$, which indicates that entity s_1 is matched to t_1, s_2 is matched to both t_1 and t_2, and s_3 is matched to t_3. Table 3 indicates the binary matrix for other neighboring states and their

score values. In the search phase (the second phase), 9 neighbors of St_0 will be evaluated, from which the algorithm picks the best candidate(s) for the next level.

To reduce the cost of the evaluation phase, we keep $\lceil x\% \rceil$ of the states with the highest weight for the next level. The reasons for using x% rather than, e.g., using a threshold value for filtering the candidate states, are as follows. First, this ensures that there will be some candidate states in the next level to evaluate. This may not be possible if we consider a high value as the threshold, leaving no candidate for the next iteration. A second reason is that, in general, users may have no knowledge of the computed score values to pick a suitable threshold value. Now, choosing x=50%, the candidate states for the next level will include St_{n2}, St_{n4}, St_{n5}, St_{n7}, and St_{n9}. In phase three, we apply more similarity measures to the state neighbor candidate(s) St_n. For this phase, we apply the structure similarity measure proposed in (Section 5.1.1) to define the structural similarities between the states identified in phase two and those in the original ontologies S and T. This measure is defined as follows.

$$v_{strc} = 2\ |r(St_n)|/(|r(S)| + |r(T)|)$$

where $|\,r(St_n)\,|$ is the found number of relationships in the candidate(s) state neighbors St_n, and $|\,r(S)\,|$ is the number of relationships in the immediate neighborhood of St_n in S. This neighborhood of St_n consists of the relationships of *(S or T)* with at least one end (one of the edge's ends) belonging to St_n. Finally, the search algorithm will yield St_4, which has a highest overall score value V for being structurally more similar.

$$V = V_{stn} + (y * V_{strc}),\quad where\ \ 0 < y \leq 1$$

In the combined similarity V, suppose V_{strc}= 0. This then means that V depends only on the similarity measures used in the first phase. On the other hand, if V_{strc}=1, the

neighborhood of the concepts matched by the second phase for state St_n is the same as those in the original ontologies S and T, and consequently V will take the maximum value.

Neighbor number	Matched pairs	Score value based on V_{stn}
St_{n1}	$\{(s_2, t_1), (s_2, t_2), (s_3, t_3)\}$	0.499
St_{n2}	$\{(s_1, t_1), (s_1, t_2), (s_2, t_1), (s_2, t_2), (s_3, t_3)\}$	0.5794
St_{n3}	$\{(s_1, t_1), (s_1, t_3), (s_2, t_1), (s_2, t_2), (s_3, t_3)\}$	0.5524
St_{n4}	$\{(s_1, t_1), (s_2, t_2), (s_3, t_3)\}$	**0.678**
St_{n5}	$\{(s_1, t_1), (s_2, t_1), (s_3, t_3)\}$	0.6543
St_{n6}	$\{(s_1, t_1), (s_2, t_1), (s_2, t_2), (s_2, t_3), (s_3, t_3)\}$	0.516
St_{n7}	$\{(s_1, t_1), (s_2, t_1), (s_2, t_2), (s_3, t_1), (s_3, t_3)\}$	0.572
St_{n8}	$\{(s_1, t_1), (s_2, t_1), (s_2, t_2), (s_3, t_2), (s_3, t_3)\}$	0.531
St_{n9}	$\{(s_1, t_1), (s_2, t_1), (s_2, t_2)\}$	0.6656

Table 3: Score value for each state neighbor

5.2.4 Experiments and Results

We have evaluated the performance of our proposed framework using two factors: quality and time. For the quality of match results, we compare our result with ten algorithms presented in the Ontology Alignment Evaluation Initiative OAEI-06 and OAEI-07 [OAEI, 2007]. In this comparison study, we used the OAEI 2007 benchmark test samples suite. The test numbers of the ontologies we used from this benchmark suite

included 101, 103, 104, 206, 228, and 230. Ontology 101 is the reference ontology, and, hence, in test case 101, ontology number 101 is matched to itself, and in test 103, ontology 101 is matched to ontology 103, etc.

For the running time, we conducted numerous experiments to show the impact of the proposed framework on the overall performance.

All these tests have been performed on a Pentium 4, 2800, with 768 MB of RAM, running Windows XP, and with no applications running but a single matcher. To measure a match quality, we used *precision, recall, and F-measure* presented in Section 4.5.

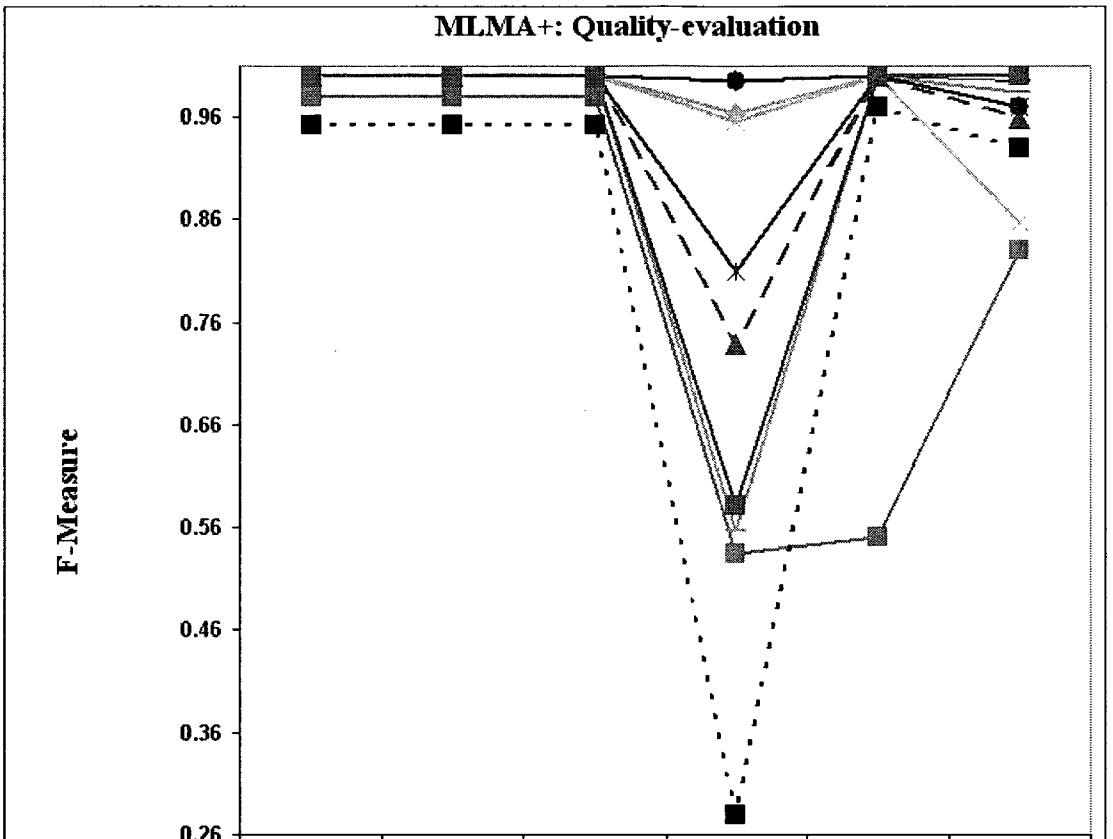
Case study (1): In this case study, we used the benchmark test samples suite OAEI 2007 [OAEI, 2007]. Except for case 206 in the suite, which is related to French translation, in all other cases considered, we noted that when the *precision* value was less than 1, the *recall* value was equal to 1. This indicates that the systems found all the correct mappings expected by the experts and added extra unwanted mappings. The precision of our search algorithm, on the other hand, did not fall below the *recall* value, i.e., no extra unwanted mappings were returned by our framework. For test case 230, ontology 101 was matched to ontology 230. Basically, ontology 230 is a modified version of ontology 101, with a changed structure, but the same entity names. In this test case, the main reason why the matching results of all other systems included unwanted mappings is that these systems combine different similarity measures in one shot. Moreover, they combine name, linguistic, and structure similarities at one level and aggregate their results in order to provide the output mappings. However, as our algorithm uses different levels for

different similarities, its result was not affected by the structure changes in the input ontology 230.

For test case 206, the reason why the matching result of our search algorithm was not fulfilled was that it did not use translating techniques as one of its underlying techniques. Fig. 29 shows the comparison of the matching quality between our algorithm and the other ten systems.

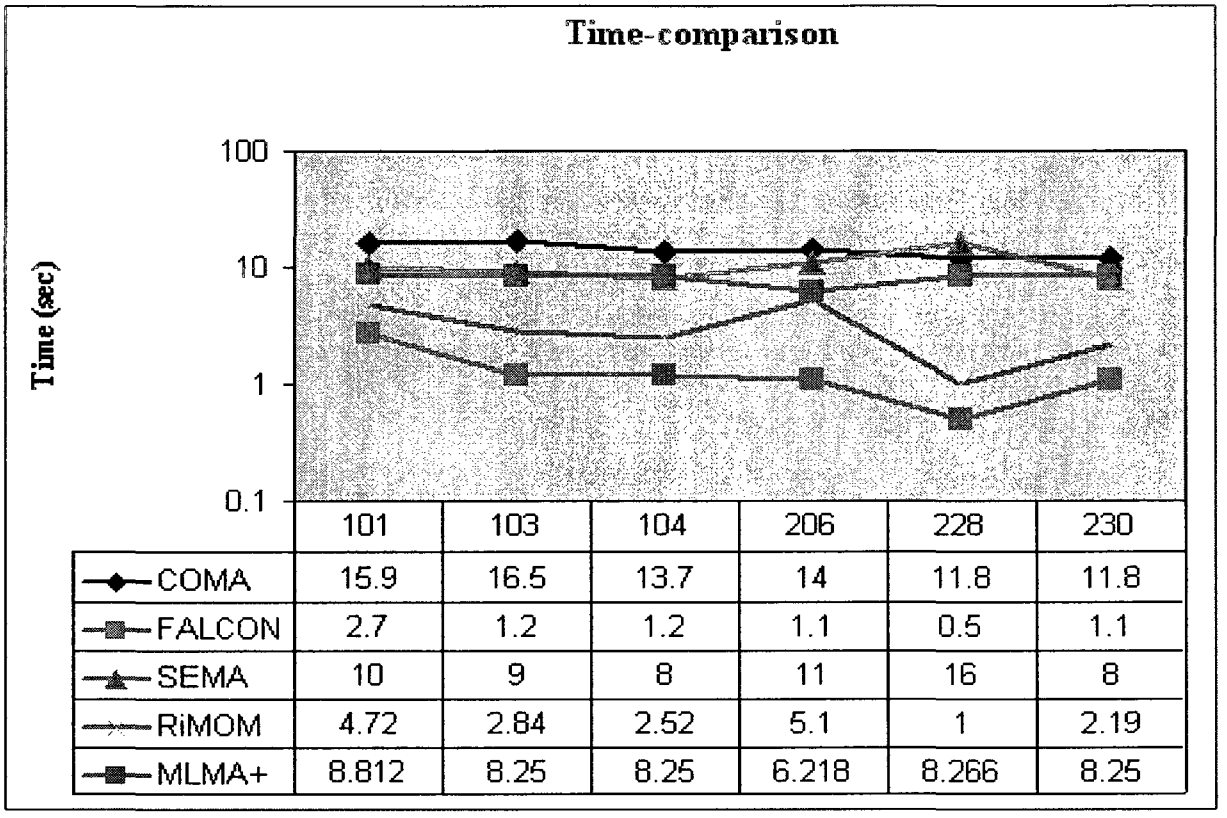
Moreover, Fig. 30 shows an approximate time comparison, indicating the scalability of our search algorithm (logarithmic scale).

Case study (2): In this case study, we used three pairs of ontologies: (1) the MIT bibtex ontology and the UMBC publication ontology, which are both publicly available, (2) computer ontologies, and (3) ontologies about computer science departments. We created the second and third pairs. The execution time, in seconds, for the neighbor search algorithm over these test cases was measured as 4.68, 0.547, and 1.719, respectively. A naïve implementation of MLMA would not perform as desired. The MLMA+ is polynomial with respect to the size of the search space $O((|E^S|\times|E^T|)^2)$, where $|E^S|$ is the number of entities in S. All in all, we consider the neighbor search algorithm as an optimization for MLMA. We called it MLMA+.



	101	103	104	206	228	230
edna	0.9796	0.9796	0.9796	0.5346	0.5507	0.8304
Lily	1.0000	1.0000	1.0000	0.9950	1.0000	0.9691
ASMOV	1.0000	1.0000	1.0000	0.9950	1.0000	0.9950
DSSim	1.0000	1.0000	1.0000	0.5563	1.0000	0.9848
falcon	1.0000	1.0000	1.0000	0.9637	1.0000	0.9691
hmatch	0.9529	0.9529	0.9529	0.2800	0.9700	0.9305
jhuapl	1.0000	1.0000	1.0000	0.7385	1.0000	0.9583
SEMA	1.0000	1.0000	1.0000	0.9548	1.0000	0.8571
prior	1.0000	1.0000	1.0000	0.8096	1.0000	0.9691
RiMOM	1.0000	1.0000	1.0000	0.9950	1.0000	0.9691
MLMA+	1.0000	1.0000	1.0000	0.5818	1.0000	1.0000

Figure 29: Quality Comparison



	101	103	104	206	228	230
COMA	15.9	16.5	13.7	14	11.8	11.8
FALCON	2.7	1.2	1.2	1.1	0.5	1.1
SEMA	10	9	8	11	16	8
RiMOM	4.72	2.84	2.52	5.1	1	2.19
MLMA+	8.812	8.25	8.25	6.218	8.266	8.25

Figure 30: Efficiency Comparison

5.3 Recommendation Analysis for Ontology Matching Techniques

In the following we propose a framework for analyzing and recommending matching techniques. A main feature of this framework is that it improves the structure matching techniques and the end result accordingly. We will refer to this improved version of the MLMA+ algorithm as MLMAR [Alasoud *et al.*, 2009].

5.3.1 Motivating example

Through the following example, we illustrate the main ideas of the proposed technique. Fig. 31 shows two sample taxonomies for two computer ontologies O_1 and O_2.

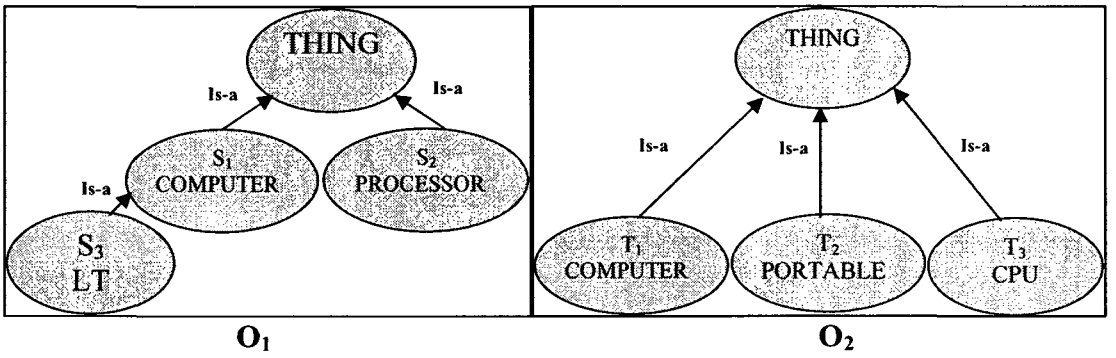


Figure 31: Computer Ontologies

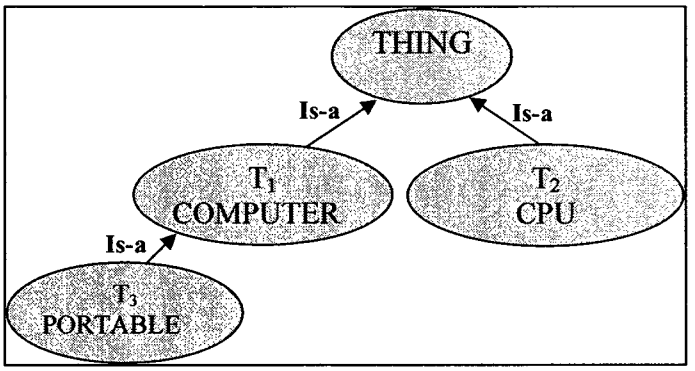
Given the input ontologies and the matching techniques, it is difficult to specify that concept *LT* in O_1 corresponds to concept *PORTABLE* in O_2. Suppose the input ontologies O_1 and O_2 are represented in description logic as follows:

$$O_1:\ \begin{array}{l} LT \subseteq COMPUTER \\ COMPUTER \subseteq THING \\ PROCECCOR \subseteq THING \end{array} \qquad O_2:\ \begin{array}{l} PORTABLE \subseteq \exists has\_cpu.CPU \\ \exists has\_cpu.CPU \subseteq COMPUTER \\ CPU \subseteq THING \end{array}$$

where $\subseteq$ denotes subsumption relationships such as is-a, $\exists$ denotes the existential quantification (see Section 2.1), and *has_cpu* is a binary relationship (see Section 2.2).

Now, using description logic (DL) reasoning techniques on these ontologies before matching them can help infer useful information to be used by a matching technique. For instance, applying DL reasoning technique on O_2 yields RO_2, which is shown in Fig. 32. However, no further inferences can be obtained from O_1. In other words, RO_1 is O_1.

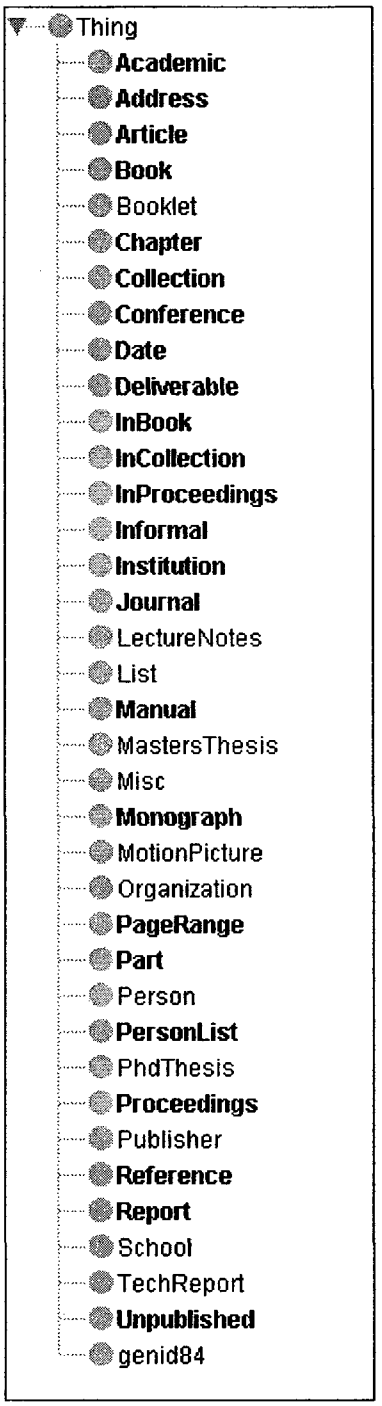
As a result, matching RO_1 to RO_2 assists the similarity matching technique (structure-based technique) to identify the relationship between the concept *LT* in RO_1 and the concept *PORTABLE* in RO_2.



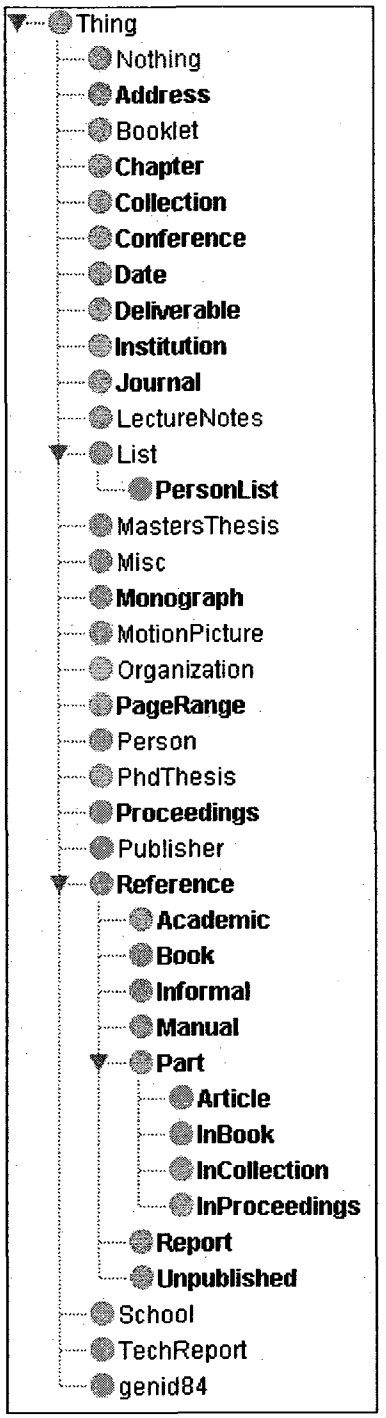
RO_2

Figure 32: O_2 after reasoning

Fig. 33(a) shows the initial taxonomy of ontology number 232 from the benchmark test samples suite of the Ontology Alignment Evaluation Initiative OAEI-07 (Section 5.3.5 gives more details). After applying the DL reasoning technique, we get more structural information. Therefore, this technique supports the structure-based matching techniques in providing better matching results when ontology 232 is matched to reference ontology 101. Fig. 33(b) shows the results of applying DL reasoning techniques to ontology 232. All in all, the user should be supported in deciding which underlying technique, or combination of techniques, is best suited for the matching task at hand.



(a) before applying DL reasoning



(b) after applying DL reasoning

Figure 33: Taxononmies of onotology 232

5.3.2 A Framework for Recommendation Analysis

In this section, we introduce a technique for the analysis and reuse of matching methods, in order to identify and recommend matching methods for a given pair of ontologies. Furthermore, the technique assists the structural similarity measuring methods, optimizes the matching process by omitting the unpractical matching methods, and therefore improves the end result's matching quality and efficiency.

Fig. 34 illustrates the main idea of the technique. It shows the different similarity measures $\{m_1, m_2 \ldots m_k\}$, together with RO_1 and RO_2, that are fed into the recommendation process, which will return a rank of the similarity measures considered (M_i). Moreover, users have the option to use the recommended similarity measures list (M_i) or to ignore it and use their own ranked similarity measure list (the user's list).

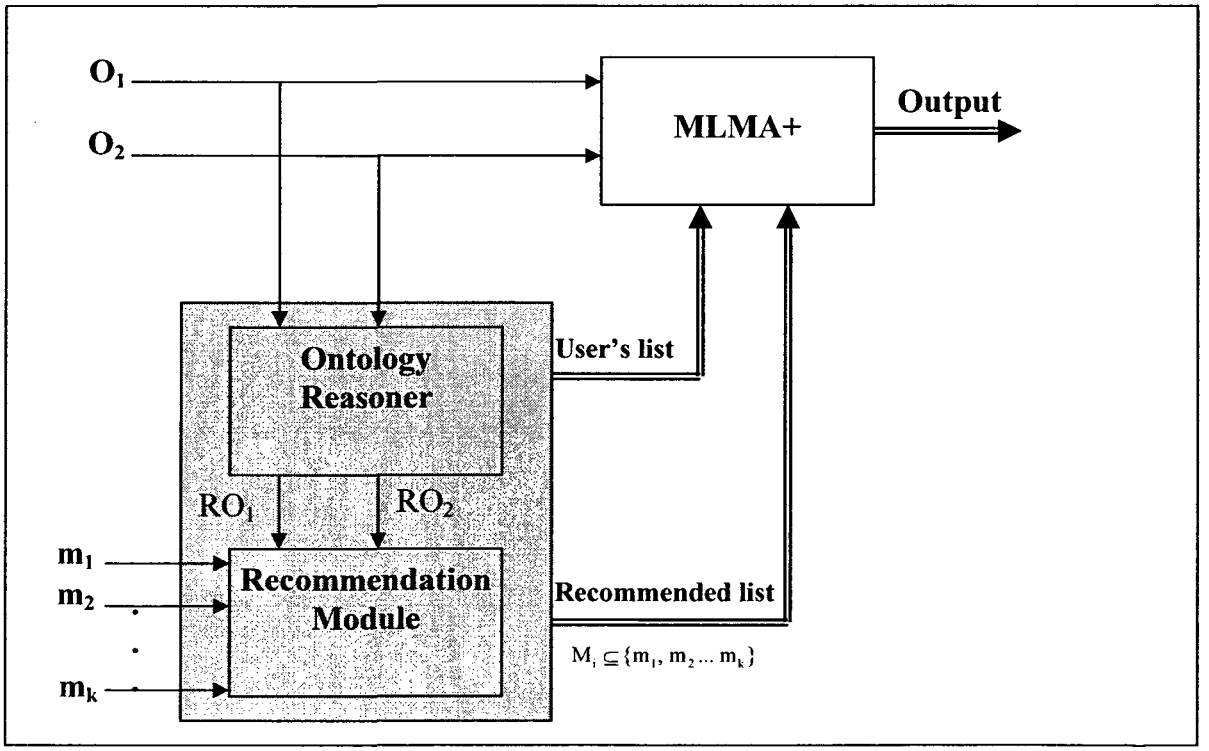


Figure 34: A recommendation analysis framework

M_i is based on the appropriate similarity methods considered for matching the entities of O_1 to the entities of O_2. Furthermore, RO_1 and RO_2 are obtained by applying RACER [Haarslev and Moeller, 2001b]. As a result, the Multi-Level Matching Algorithm (MLMA+) that performs a neighbour search takes these recommendations into account, in order to find the correspondences between entities in the given ontologies.

5.3.3 Specific techniques used in the proposed framework

For ease of presentation, we focus on the techniques we have so far implemented in our framework. The framework, however, is flexible, and thus could incorporate any other matching techniques. In our work, we considered a string-based technique (Levenshtein distance), linguistic-based technique (WordNet), and structure-based technique.

The string and linguistic based techniques evaluate the given entities by analyzing their names, labels and comments. They consider both the lexical and linguistic features as terms of comparison. Moreover, the structure-based techniques take into account the structural layout of the ontologies considered, e.g., graph matching. In this work, we are improving the structure similarity presented in [Alasoud *et al.*, 2007] by considering the inferred input ontologies (RO_1 and RO_2) by using a DL reasoner, i.e., RACER on the input pair of ontologies (O_1 and O_2). Consequently, our structure similarity measure will be updated as follows:

$$S_{strc} = 2\ \left|r(O_{output})\right| \Big/ \left[\left|r(O_{output}(RO_1))\right| + \left|r(O_{output}(RO_2))\right|\right]$$

where $|r(O_{output})|$ is the number of relationships in the output ontology (a neighbour/candidate result), and $|r(O_{output}(RO_i))|$ is the number of relationships in the immediate neighborhood of O_{output} in the inferred input ontology RO_i. This

neighbourhood of O_{output} consists of the relationships of RO_i with at least one end (one of the edge's ends) belonging to O_{output}. In other words:

- $|r(O_{output})|$ is the number of *common* relationships that connect common concepts to other common concepts (immediate neighbors). The resulting correspondences between entities (concepts/relationships) in RO_1 and RO_2 are what is meant by *common*.
- $|r(O_{output}(RO_1))|$ is the number of relationships in RO_1, with at least one end belonging to the *common* concepts belonging to O_{output}.
- $|r(O_{output}(RO_2))|$ is the number of relationships in RO_2, with at least one end belonging to the *common* concepts belonging to O_{output}.

5.3.4 Similarity recommendation technique

In this subsection, we illustrate a heuristic technique which we used in our framework, in order to offer users a ranked list (M_i) of appropriate techniques for the matching task at hand. The string/linguistic based techniques are evaluated as follows:

M = [(number of concept pairs with the same label/synonym) / (max (C_1, C_2))]

where "number of concept pairs with the same label/synonym" represents the number of pairs that have the same label for the name-based techniques and the same synonym for the linguistic-based technique, such that $\{(c_1, c_2) | c_1 \in RO_1 \text{ and } c_2 \in RO_2\}$.

We use labels for string-based techniques and synonyms for linguistic-based techniques. Further, max (C_1, C_2) stands for the maximum number of concepts either in RO_1 or RO_2. The structure-based techniques are evaluated as follows:

M = (number of common concepts) / (max_number_of_nonleaf (C_1, C_2))

Where, number of common concepts represents the cardinality of the set $\{(c_1,c_2)|\, c_1 \in RO_1 \text{ and } c_2 \in RO_2\}$, such that both c_1 and c_2 have the same number of sub-concepts and the same depth. The max_number_of_nonleaf (C_1, C_2) denotes the maximum number of concepts that have sub concepts either in RO_1 or RO_2.

These heuristic techniques are not a precise measure of the real matching similarities of the entities for the input pair of ontologies. However, they can estimate the features of the two ontologies and provide the ranked list of the appropriate matching techniques, to be used accordingly.

5.3.5 Experiments and results

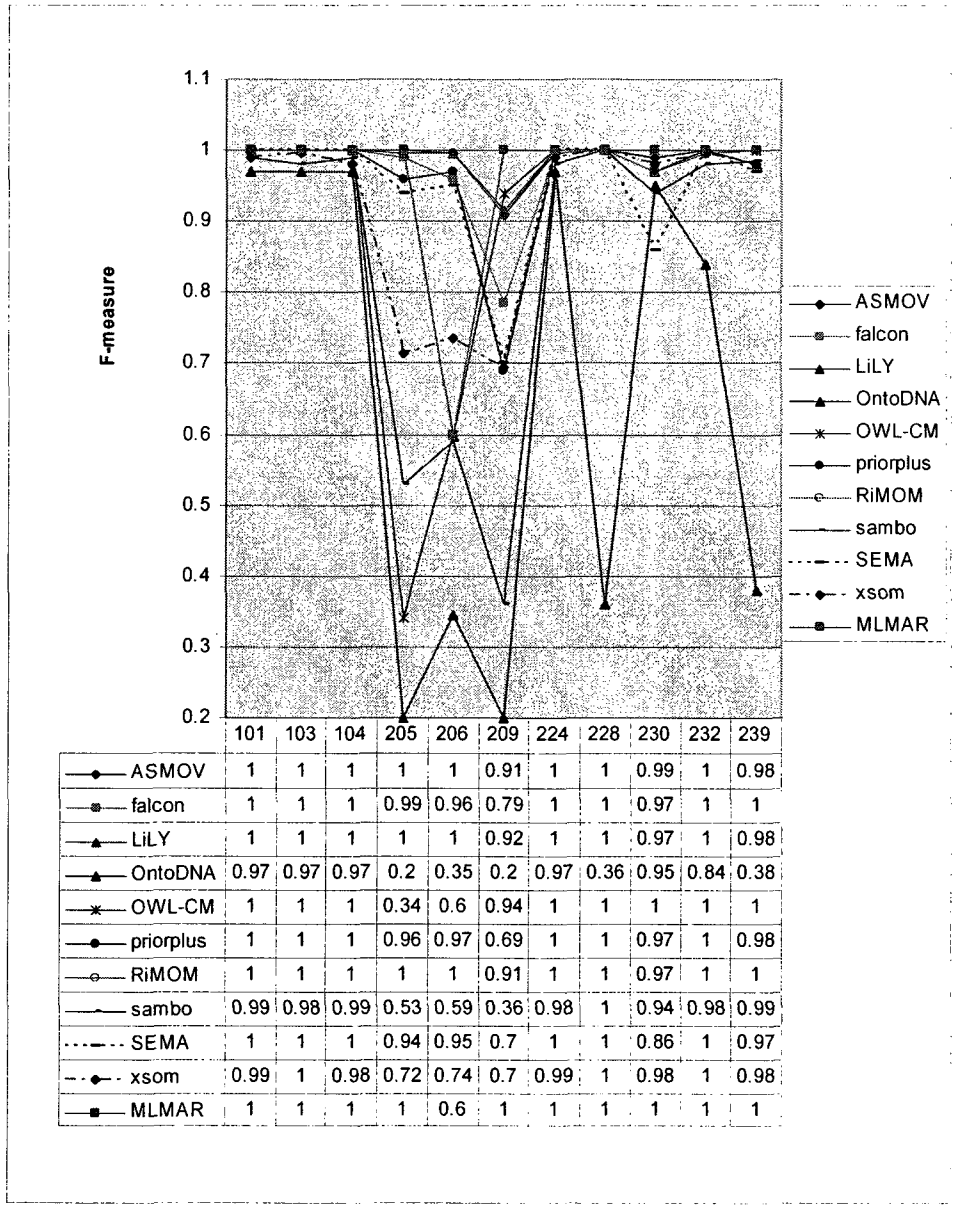
We used our experimental setup described in Section 5.2.4, and compared our result with ten algorithms presented in the OAEI-06 and OAEI-07. For the time factor, we conducted numerous experiments to show the impact of the proposed framework on overall performance. We use MLMAR to refer to MLMA+ with the recommendation analysis technique included.

The test numbers of the ontologies we used from the OAEI 2007 benchmark test suit included 101, 103, 104, 205, 206, 209, 224, 228, 230, 232, and 239. Fig. 35 shows the comparison of the matching quality of our algorithm with the other ten systems.

In addition, Fig. 36 shows a time comparison indicating the scalability of our framework (please note the logarithmic scale).

Table 4 shows the initial estimation for the similarity measures, as well as a description of each test number. As can be noted, in test numbers 101, 103, 104, 224, 228, and 230, the modifications made to the reference ontology did not affect the string,

linguistic, and structure similarities, and hence all the matchers obtained the highest similarity value. Accordingly, our framework will take advantage of not running all the matchers and will offer only a single matcher to the user. In such scenarios, we can often use string similarity because it is the most efficient one and it is the backbone for other matchers.

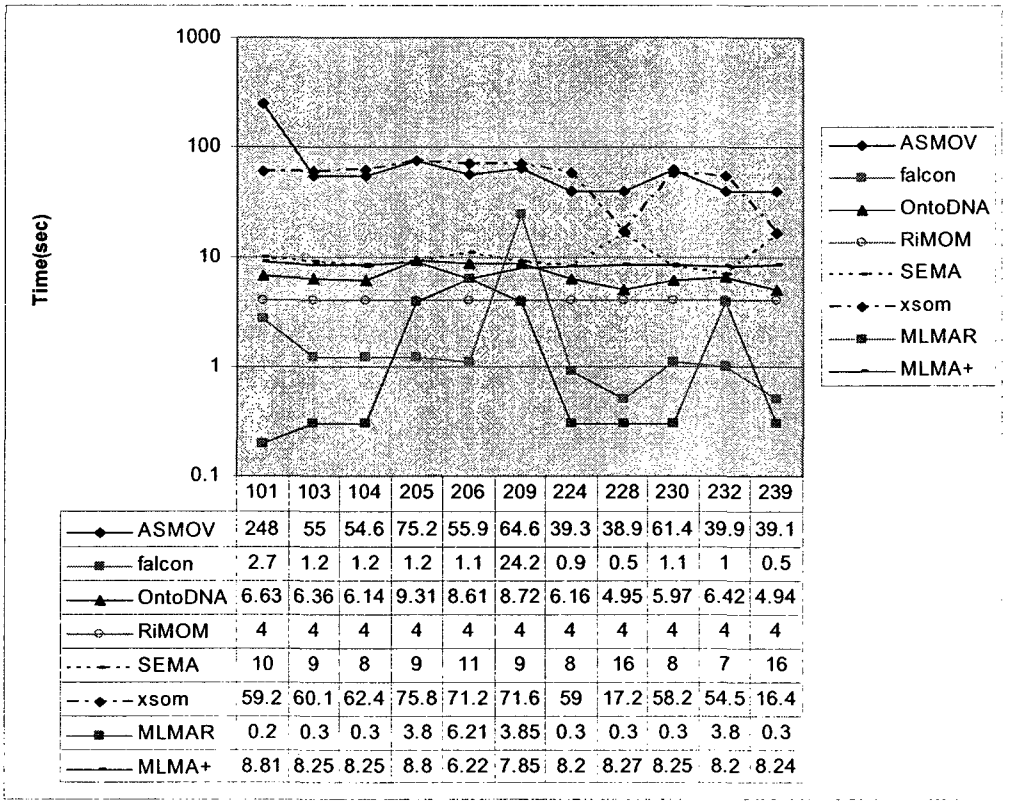


	101	103	104	205	206	209	224	228	230	232	239
ASMOV	1	1	1	1	1	0.91	1	1	0.99	1	0.98
falcon	1	1	1	0.99	0.96	0.79	1	1	0.97	1	1
LiLY	1	1	1	1	1	0.92	1	1	0.97	1	0.98
OntoDNA	0.97	0.97	0.97	0.2	0.35	0.2	0.97	0.36	0.95	0.84	0.38
OWL-CM	1	1	1	0.34	0.6	0.94	1	1	1	1	1
priorplus	1	1	1	0.96	0.97	0.69	1	1	0.97	1	0.98
RiMOM	1	1	1	1	1	0.91	1	1	0.97	1	1
sambo	0.99	0.98	0.99	0.53	0.59	0.36	0.98	1	0.94	0.98	0.99
SEMA	1	1	1	0.94	0.95	0.7	1	1	0.86	1	0.97
xsom	0.99	1	0.98	0.72	0.74	0.7	0.99	1	0.98	1	0.98
MLMAR	1	1	1	1	0.6	1	1	1	1	1	1

Figure 35: Quality Comparison

Furthermore, in test numbers 205 and 209, the framework offers both the linguistic and the structural measures. In test 206, both string and structure similarity measures were used. Moreover, test 232 shows the best scenario where there is no hierarchy, and using the DL reasoning technique (RACER), the structure similarity jumps from 0.0 to 0.7. Consequently, we applied both the string and structure similarities in this test. Lastly, in test number 239, the string similarity was applied.

In general, these recommendations greatly affected the performance time and placed our framework (MLMAR) at the top of the compared algorithms, based on average time. Also, they considerably improved the efficiency of MLMA+ by using only the recommended similarity techniques, rather than using all of them. Moreover, applying different order of the similarity matching techniques will not affect the matching quality. However, it will slightly affect the running time, see Appendix D.



	101	103	104	205	206	209	224	228	230	232	239
ASMOV	248	55	54.6	75.2	55.9	64.6	39.3	38.9	61.4	39.9	39.1
falcon	2.7	1.2	1.2	1.2	1.1	24.2	0.9	0.5	1.1	1	0.5
OntoDNA	6.63	6.36	6.14	9.31	8.61	8.72	6.16	4.95	5.97	6.42	4.94
RiMOM	4	4	4	4	4	4	4	4	4	4	4
SEMA	10	9	8	9	11	9	8	16	8	7	16
xsom	59.2	60.1	62.4	75.8	71.2	71.6	59	17.2	58.2	54.5	16.4
MLMAR	0.2	0.3	0.3	3.8	6.21	3.85	0.3	0.3	0.3	3.8	0.3
MLMA+	8.81	8.25	8.25	8.8	6.22	7.85	8.2	8.27	8.25	8.2	8.24

Figure 36: Time Comparison

The reason that MLMAR may not perform as a first rate system is that, in some test cases, i.e. 205, 206, 209, and 232, a combination of low efficiency similarity measure techniques, such as linguistic and structural, should be used. These ontologies, in some sense, were considered as a worst case scenario where all matching techniques needed to be applied. However, in general, matching tools are equipped with numerous underlying similarity measuring techniques and using the recommended techniques will reduce the number of candidate techniques for a matching task at hand. Consequently, the matching process time decreases remarkably.

Test No.	String Similarity	Linguistic Similarity	Structure Similarity	Test Description
101	1	1	1	Ontology 101 is matched to itself
103	1	1	1	The generalization basically removes owl:unionOf and owl:oneof and the Property types (owl:TransitiveProperty).
104	1	1	1	This test compares the ontology with its restriction in OWL Lite (where unavailable constraints have been discarded).
205	0.125	0.85	1	Labels are replaced by synonyms. Comments have been suppressed.
206	0.1	0.1	1	The ontology translated into French
209	0.125	0.85	1	Synonyms are used
224	1	1	1	All individuals have been suppressed from the ontology.
228	1	1	1	Properties and relations between objects have been suppressed.
230	1	1	0.78	Some components of classes are expanded in the class structure (e.g., year, month, day attributes instead of date).
232	1	1	0.7	No Hierarchies and no instances
239	1	1	0.55	Flattened Hierarchy and no properties

Table 4: Initial estimations for the similarity measures

All in all, for the matcher composition systems, using a recommended subset of their similarity measures list should improve the final matching results in terms of time

and quality. Moreover, the recommendation techniques improve the overall running time, as it is unnecessary to reuse and combine all their underlying similarity measuring methods. Using only a recommended subset should decrease the average running times. Furthermore, the recommendation techniques can enhance the matching quality by excluding the unworkable similarity matching methods for a task at hand. For instance, if there is no string, linguistic, or structural similarity between a given pair of input ontologies, then including, combining, and aggregating the matching results retrieved by string, linguistic, or structural similarity measuring methods would affect the overall matching result quality in a negative manner.

5.4 Summary and Remarks

In this chapter, we discussed the following:

- We proposed a multi-level extension of MMA, called MLMA, which assumes that the collection of similarity measures are partitioned by the user, and that there is a partial order in the partitions, also defined by the user.
- A main characteristic of the MLMA technique is that it combines existing matching techniques to provide a solution to a given ontology matching problem. Moreover, the optimal matching state has been considered, based on its rich structure on one hand, and the number of common concepts of the matched ontologies on the other.
- Applying the MLMA method will not decrease the number of matching concepts (size) and will increase the similarity measure of the state that has high structural similarity among its concepts (structure).

- In contrast to some other approaches, our proposed similarity measure ensures that MLMA works even in a case where there are no structural similarities in the given input ontologies.
- We further investigated the efficiency improvements of MLMA by introducing an optimization step. We call the result MLMA+.
- MLMA+ improves the efficiency of MLMA considerably due to its use of the neighbor search algorithm. It proceeds by computing an initial state and then performing a search in its neighboring states.
- Moreover, we studied the impact of different choices of strategies for matching ontologies and proposed a framework for analyzing the reused matching techniques (MLMAR).
- MLMAR shows the importance of assisting the user by suggesting appropriate matching strategies. The user often has little or no idea about the suitability of matching strategies for a given matching task. As a result, the quality of matching results and processing times will be affected by the method chosen.
- The main advantages of the MLMAR are that (1) it is independent from any individual matching technique, (2) it infers a hidden structural relationship among the entities of the input ontologies, and consequently makes the structure-based similarity measure more precise, and (3) it considerably improves the efficiency of the matching process in terms of time.

6. Conclusion and Future Work

6.1 Conclusion

The motivation for this research was the need for ontology matching (Sec 1.1) in many emerging applications. We studied different matching techniques (Sec 2.4) and their implementations (Sec 2.5). Also, we presented a novel framework (Sec 3.1) to support information integration from ontology data sources. Furthermore, we discussed different approaches to ontology integration, and how a combination of virtual and materialized approaches, called the hybrid approach, can be used in order to combine the advantages of both.

In order to support the hybrid approach with a matching strategy, we proposed the multi-matching strategy (Sec 4.3). This strategy benefits from existing ontology match techniques and "combines" their match results to provide enhanced ontology matching results.

To obtain better quality matching results, we extended the multi-matching strategy by introducing a multi-level matching strategy (Sec 5.1). This technique

assumes that the collection of similarity measures is partitioned by the user, and that there is a partial order on these partitions, also defined by the user. The main characteristic of the MLMA technique is that its application will not decrease the number of matching concepts (size), but will increase the similarity measure of states that have high structural similarity among their concepts (structure). Our proposed similarity measure also ensures that MLMA works well even when there are no structural similarities in the given input ontologies.

We investigated the efficiency of MLMA by introducing a neighbor search algorithm (Sec 5.2) which, given an initial mapping state among entities in two ontologies, searches the neighboring states and returns a list of candidate states, ranked based on their evaluation scores. We incorporated this search algorithm into MLMA, and refer to it as MLMA+, which proceeds by computing an initial state and then performing a search in neighboring states. We have developed a running prototype of MLMA+ and conducted experiments using well-known benchmark ontologies.

In this work, we studied the impact of the choice of matching strategies and proposed a framework for analyzing the reused matching techniques (Sec 5.3). The user often has little or no idea about the suitability of the particular matching strategies for a given matching task. Consequently, the quality of matching results and processing times are affected by the method chosen. The main advantages of the proposed framework are (1) it is independent from the individual matching techniques used, (2) it infers a hidden structural relationship among the entities of the input ontologies, and consequently makes the structure-based similarity measure more precise, and (3) it considerably improves the matching process time.

We evaluated our framework against other approaches using different pairs of ontologies. Our results indicated better performance in terms of both quality and time.

6.2 Future Work

This section highlights some possible future directions for advancing the ontology matching techniques we have proposed.

- It is not easy for the user to identify different weights of individual matchers in order to get acceptable matching results. As a result, it would be interesting to automate the process of combining the individual matchers and libraries of matchers.
- Each individual matcher has parameters that should be properly set to get the best possible match results. However, users cannot be expected to know or find correct parameters by themselves. Assisting tools are required to alleviate the situation. Machine learning techniques could be used to achieve this.
- Packaging MLMAR and making it available for other users to evaluate and compare the provided results. Also, providing a user interface would help users interact with the system to effectively review matching results and to modify them in an interactive manner.

References

[Agrawal and Srikant, 2001] Agrawal, R. and Srikant, R. On integrating catalogs. In Proc. 10th International WWW Conference, pages 603-612 Hong Kong (CN), 2001.

[Alasoud *et al.*, 2009] Alasoud, A., Haarslev, V. and Shiri, N. An Empirical Comparison of Ontology Matching Techniques. Accepted for publication in Journal of Information Science. 2009. 20 pages.

[Alasoud *et al.*, 2008] Alasoud, A., Haarslev, V. and Shiri, N. An Effective Ontology Matching Technique. In: "Foundations of Intelligent Systems", Proc. of ISMIS'08, A. An et al. (Eds.), Toronto, Canada, LNAI, Vol. 4994, Springer. pp. 585-590. 2008.

[Alasoud *et al.*, 2007] Alasoud, A., Haarslev,V., and Shiri, N. A Multi Level Matching Algorithm for Combining Similarity Measures in Ontology Integration, in ODBIS VLDB-Workshop Post-proceedings, LNCS 4623, Springer-Verlag, Berlin, Heidelberg, pp. 1-17, 2007.

[Alasoud *et al.*, 2005] Alasoud, A., Haarslev,V., and Shiri, N. A hybrid approach for ontology integration. In Proc. VLDB Workshop on Ontologies-based techniques for DataBases and Information Systems (ODBIS), Trondheim, Norway, September, pp. 18-23, 2005.

[An *et al.*, 2006] An, Y., Borgida, A., and Mylopoulos, J. Discovering the semantics of relational tables through mappings. Journal on Data Semantics, VII: 1-32, 2006.

[An *et al.*, 2005a] An, Y., Borgida, A., and Mylopoulos, J. Constructing complex semantic mappings between XML data and ontologies. In Proc. 4th International

Semantic Web Conference (ISWC), volume 3729 of Lecture notes in computer science, pp. 6-20, Galway (IE), 2005.

[An *et al.*, 2005b] An, Y., Borgida, A., and Mylopoulos, J. Inferring complex semantic mappings between relational tables and ontologies from simple correspondences. In Proc. 4th International Conference on Ontologies, Databases and Applications of Semantics (ODBASE), vol. 3761 of LNCS, pp. 1152-1169, Agia Napa (CY), 2005.

[Baader *et al.*, 2007] Baader, F., Calvanese, D., McGuinness, D., Nardi, D., and Patel-Schneider, P. (eds). The Description Logic Handbook: Theory, Implementation, and Applications (Cambridge University Press, 2007, 2nd edition).

[Bach and Kuntz, 2005] Bach, T. and Kuntz, R. Measuring similarity of elements in OWL ontologies. In Proc. AAAI Workshop on Contexts and Ontologies (C&O), pp. 96-99, Pittsburgh (PA US), 2005.

[Bach *et al.*, 2004] Bach, T., Kuntz, R., and Gandon, F. On ontology matching problems (for building a corporate semantic web in a multi-communities organization). In Proc. 6th International Conference on Enterprise Information Systems (ICEIS), pp. 236-243, Porto (PT), 2004.

[Batini *et al.*, 1986] Batini, C., Lenzerini, M., and Navathe, S. A comparative analysis of methodologies for database schema integration. ACM Computing Surveys, 18(4):323-364, 1986.

[Bernstein and Rahim, 2000] Bernstein, P. and Rahm, E. Data warehouse scenarios for model management. In Proc. 19 International Conference on Conceptual Modeling(ER), LNCS 1920, pp. 1-15, Salt Lake City (UTUS), 2000.

[Bouquet *et al.*, 2003a] Bouquet, P., Magnini, B., Serafini, L., and Zanobini, S. A SAT-based algorithm for context matching. In Proc. 4th International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT), volume 2680 of Lecture notes in computer science, pp. 66-79, Stanford (CA US), 2003.

[Bouquet *et al.*, 2003b] Bouquet, P., Serafini, L., and Zanobini, S. Semantic coordination: A new approach and an application. In Proc. 2nd International Semantic Web Conference (ISWC), volume 2870 of Lecture notes in computer science, pp. 130-145, Sanibel Island (FL US), 2003.

[Bouquet *et al.*, 2006] Bouquet, P., Serafini, L., Zanobini, S., and Sceffer, S. Bootstrapping semantics on the web: meaning elicitation from schemas. In Proc. 15th International WWW Conference, pp. 505-512, Edinburgh (UK), 2006.

[Calvanese and Giacomo, 2005] Calvanese, D. and De Giacomo, G. Data Integration: A Logic Based Perspective. AI Magazine, 26(1), 2005.

[Calvanese *et al.*, 2004] Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R., and Vetere, G. DL-Lite: Practical reasoning for rich DLs. In Proc. Description Logic Workshop, pp. 92-99, 2004.

[Calvanese *et al.*, 2002] Calvanese, D., De Giacomo, G., Lenzerini, M. A framework for ontology integration. In I. Cruz, S. Decker, J. Euzenat, and D. McGuinness, editors, The Emerging Semantic Web — Selected Papers from the First Semantic Web Working Symposium, pp. 201–214. IOS Press, 2002.

[Castano *et al.*, 2006] Castano, S., Ferrara, A., and Montanelli, S. Matching ontologies in open networked systems: Techniques and applications. Journal on Data Semantics, V: 25-63, 2006.

[Castano *et al.*, 2005] Castano, S., Ferrara, A., and Montanelli, S. Dynamic knowledge discovery in open, distributed and multi-ontology systems: Techniques and applications. In David Taniar and Johanna Rahayu, editors, Web semantics and ontology, chapter 8, pp. 226-258. Idea Group Publishing, Hershey (PA US), 2005.

[Chawathe *et al.*, 1994] Chawathe, S., Molina, H., Hammer, J., Ireland, K., Papakonstantinou, Y., Ullman, J., and Widom, J. The TSIMMIS project: Integration of heterogeneous information sources. In Proc. 16th Meeting of the Information Processing Society of Japan (IPSJ), pp. 7-18, Tokyo (JP), 1994.

[Cohen *et al.*, 2003] Cohen, W., Ravikumar, P., Fienberg, S. A Comparison of String Distance Metrics for Name-Matching Tasks. IJCAI-03: 73-78, 2003.

[Doan *et al.*, 2004] Doan, A., Madhavan, J., Domingos, P., and Halevy, A. Ontology matching: a machine learning approach. In Staab, S. and Studer, R., editors Handbook on ontologies, ch.18, pp. 385-404. Springer Verlag, Berlin (DE), 2004.

[Do *et al.*, 2002] Do, H., Melnik, S., Rahm, E. Comparison of schema matching evaluations. Proceedings of workshop on Web and Databases. pp. 221-237. 2002.

[Draper *et al.*, 2001] Draper, D., Halevy, A., and Weld, D. The nimble integration engine. In Proc. 20th International Conference on Management of Data SIGMOD pp. 567-568, Santa Barbara (CA US), 2001 .

[Dzbor *et al.*, 2004] Dzbor, M., Motta, E., and Domingue, J. Opening up Magpie via semantic services. In Proc. 3rd International Semantic Web Conference (ISWC), vol. 3298 of LNCS, pp. 635-649, Hiroshima (JP), 2004.

[Dzbor *et al.*, 2003] Dzbor, M., Domingue, J., and Motta, E. Magpie – towards a semantic web browser. In Proc. 2nd International Semantic Web Conference (ISWC),

volume 2870 of Lecture notes in computer science, pp. 690-705, Sanibel Island (FL US), 2003.

[Euzenat and Shvaiko, 2007] Euzenat, J., Shvaiko, P. Ontology matching, Springer Verlag, Heidelberg (DE), 333 p., 2007

[Euzenat *et al.*, 2007] Euzenat, J., Isaac, A., Meilicke, C., Shvaiko, P., Stuckenschmidt, H., Šváb, O., Svátek, V., Robert, W., Hage, V. and Yatskevich, M. Results of the ontology alignment evaluation initiative. Proc. of the ISWC workshop on Ontology Matching, Busan, Korea, Nov. 2007.

[Euzenat *et al.*, 2006] Euzenat, J., Mochol, M., Shvaiko, P., Stuckenschmidt, H., Svab, O., Svatek, V., Robert, W., Hage, V., and Yatskevich, M. Results of the ontology alignment evaluation initiative 2006. Proc. of ISWC workshop on Ontology Matching, Athens, pp. 73–95, 2006.

[Euzenat *et al.*, 2004] Euzenat, J., Le Bach, T., Barrasa, J., Bouquet, P., Bo, J., Kuntz, R., Ehrig, M., Hauswirth, M., Jarrar, M., Lara, R., Maynard, D., Napoli, A., Stamou, G., Stuckenschmidt, H., Shvaiko, P., Tessaris, S., Acker, S., and Zaihrayeu, I. State of the art on ontology alignment. Deliverable D2.2.3, Knowledge web NoE, 2004.

[Euzenat and Valtchev, 2004] Euzenat, J. and Valtchev, P. Similarity-based ontology alignment in OWL-lite. In Proc. 15th European Conference on Artificial Intelligence (ECAI), pp. 333-337, Valencia (ES), 2004.

[Euzenat, 1994] Euzenat, J. Brief overview of T-tree: the Tropes taxonomy building tool. In Proc. 4th ASIS SIG/CR Workshop on Classification Research, pp. 69-87 Columbus (OH US), 1994.

[Fensel, 2004] Fensel, D. Ontologies: a silver bullet for knowledge management and electronic commerce. Springer, Heidelberg (DE), 2nd edition, 2004.

[Fensel *et al.*, 2002] Fensel, D., Harmelen, F., Ding, Y., Klein, M., Akkermans, H., Broekstra, J., Kampman, A., Meer, J., Sure, Y., Studer, R., Krohn, U., Davies, J., Engels, R., Iosif, V., Kiryakov, A., Lau, T., Reimer, U., and Horrocks, I. On-To-Knowledge in a Nutshell. Special Issue of IEEE Computer on Web Intelligence (WI). 2002.

[Gangemi *et al.*, 2002] Gangemi, A., Fisseha, F., Pettman, I., Pisanelli, D., Taconet, M., Keizer, J. A Formal Ontological Framework for Semantic Interoperability in the Fishery Domain. Proceedings of the ECAI-02 Workshop on Ontologies and Semantic Interoperability, pp. 16-30, Lyon, France. 2002.

[Giunchiglia *et al.*, 2005] Giunchiglia, F., Shvaiko, P., and Yatskevich, M. Semantic schema matching. In Proc. 13th International Conference on Cooperative Information Systems (CoopIS), vol. 3761 of LNCS, pp. 347-365, Agia Napa (CY), 2005.

[Giunchiglia *et al.*, 2003] Giunchiglia, F. and Shvaiko, P. Semantic matching. The Knowledge Engineering Review, 18(3):265-280, 2003.

[Gotoh, 1981] Gotoh, O. An improved algorithm for matching biological sequences. Journal of Molecular Biology, 162(3):705-708, 1981.

[Gruber, 1993] Gruber. T. A Translation Approach to Portable Ontology Specifications. In: Knowledge Acquisition, An International Journal of Knowledge Acquisition for Knowledge Based Systems, Vol. 5, No. 2, pp. 199-220. June 1993.

[Haarslev *et al.*, 2004] Haarslev, V., Moller, R., and Wessel, M. Querying the Semantic Web with Racer + nRQL. Proceedings of the KI-2004 International Workshop on Applications of Description Logics (ADL'04), Ulm, Germany, September 24, 2004.

[Haarslev and Moeller, 2001a] Haarslev, V. and. Moeller, R. High performance reasoning with very large knowledge bases: A practical case study. In B. Nebel H. Levesque, editor, International Joint Conference on Artificial Intelligence (IJCAI'2001), Washington, USA. Morgan- Kaufmann, pp. 161-166 August 2001.

[Haarslev and Moeller, 2001b] Haarslev, V. and Moeller, R. RACER system description. In Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2001), vol. 2083 of LNAI, pp. 701–705. Springer, 2001.

[Halevy *et al.*, 2005] Halevy, A., Ashish, N., Bitton, D., Carey, M., Draper, D., Pollock, J., Rosenthal, A., and Sikka, V. Enterprise information integration: successes, challenges and controversies. In Proc. 24th International Conference on Management of Data (SIGMOD), pp. 778-787, Baltimore (MD US), 2005.

[Hamming, 1950] Hamming, R. Error detecting and error correcting codes. Technical Report 2, Bell System Technical Journal, 1950.

[Heflin and Hendler, 2000] Heflin, J. and Hendler, J. Dynamic ontologies on the Web. Proceedings of 17th National Conference on Artificial Intelligence (AAAI-2000).

[Hu *et al.*, 2007] Hu, W., Zhao, Y., Li, D., Cheng, G., Wu, H., and Qu, Y. The results of Falcon-AO. In Proc. International workshop on Ontology Matching (OM), Busan, Korea. Pp.170-178. November 11, 2007.

[Ichise *et al.*, 2003] Ichise, R., Takeda, H., and Honiden S. Integrating multiple internet directories by instance-based learning. In Proc. 18th International Joint Conference on Artificial Intelligence (IJCAI), pp. 22-30, Acapuico (MX), 2003.

[Jaro, 1989] Jaro, M. Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida. Journal of the American Statistical Association 84(406):414-20, 1989.

[Jaro, 1976] Jaro, M. UNIMATCH: A record linkage system: User's manual. Technical report, U.S. Bureau of the Census, Washington (DC US), 1976.

[Kalfoglou and Schorlemmer, 2003] Kalfoglou, Y. and Schorlemmer, M. IF-Map: an ontology mapping method based on information flow theory. Journal on Data Semantics I:98-127,2003.

[Kalfoglou and Schorlemmer, 2002] Kalfoglou, Y., and Schorlemmer, M. Information-Flow-based Ontology Mapping. Proc. of 1st Conf. on Ontologies, Databases and Application of Semantics (ODBASE'02), CA, USA, pp. 1132-1151, 2002.

[Lacher and Groh, 2001] Lacher, M. and Groh, G. Facilitating the exchange of explicit knowledge through ontology mappings. In Proc. 14th International Florida Artificial Intelligence Research Society Conference (FLAIRS), pp. 305-309, Key West (FL US), 2001.

[Lenzerini, 2002] Lenzerini, M. Data integration: A theoretical perspective. In Proc. 21st ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2002), pp. 233–246, 2002.

[Levenshtein, 1966] Levenshtein, V. Binary codes capable of correcting deletions, insertions, and reversals. Doklady akademii nauk SSSR, 163(4):845-848, 1965. In Russian. English Translation in Soviet Physics Doklady, 10(8) pp. 707-710, 1966.

[Li *et al.*, 2007] Li, Y., Zhong, Q., Li, J., and Tang, J. Results of ontology alignment with RiMOM. In Proc. International workshop on Ontology Matching (OM), Busan, Korea. Pp. 227-235. November 11, 2007.

[Mitra *et al.*, 2005] Mitra, P., Noy, N., and Jaiswal, A. Ontology mapping discovery with uncertainty. In Proc. 4th International Semantic Web Conference (ISWC), vol. 3729 of LNCS, pp. 537-547, Galway (IE), 2005.

[Mitra *et al.*, 2000] Mitra, P., Wiederhold, G., and Kersten, M. A graph oriented model for articulation of ontology interdependencies. In Proc. 8th Conference on Extending Database Technology (EDBT), vol. 1777 of LNCS, pp. 86-100, Praha (CZ), 2000.

[Mitra *et al.*, 1999] Mitra, P., Wiederhold, G., and Jannink, J. Semi-automatic integration of knowledge sources. In Proc. 2nd International Conference on Information Fusion, pp. 572-581, Sunnyvale (CA US), 1999.

[Knouf, 2003] Knouf, N. MIT bibtex ontology. 2003. Available at: http://visus.mit.edu/bibtex/0.1/bibtex.owl.

[Monge and Elkan , 1997] Monge, A. and Elkan, C. An efficient domain independent algorithm for detecting approximately duplicate database records. In Proc. SIGMOD Workshop on Data Mining and Knowledge Discovery, Tucson (AZ US), 1997.

[Needleman and Wunsch, 1970] Needleman, S. and Wunsch, C. A general method applicable to the search for similarities in the amino acid sequence of two proteins. Journal of Molecular Biology, 48(3)-443-53, 1970.

[Noy and Klein, 2004] Noy, N. and Klein, M. Ontology evolution: Not the same as schema evolution. Knowledge and Information Systems, 6(4):428-440, 2004.

[Noy and Musen, 2004] Noy, N. and Musen, M. Ontology versioning in an ontology management framework. IEEE Intelligent Systems, 19(4):6-13, 2004.

[Noy and Musen, 2002] Noy, N. and Musen, M. PromptDiff: A fixed-point algorithm for comparing ontology versions. In Proc. 18th National Conference on Artificial Intelligence (AAAI), pp. 744-750, Edmonton (CA), 2002.

[Noy and Musen, 2001] Noy, N. and Musen, M. Anchor-PROMPT: Using non local context for semantic matching. In Proc. IJCAI Workshop on Ontologies and Information Sharing, pp. 63-70, Seattle (WA US), 2001.

[Noy and Musen, 2000] Noy, N. and Musen M. PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In Proc. of the 17th National Conference on Artificial Intelligence (AAAI-2000), pp. 450-455, Austin, Texas, USA.

[OAEI, 2007] Ontology Alignment Evaluation Initiative (2007). Available at: http://oaei.ontologymatching.org/2007/benchmarks/ (accessed October 5, 2008).

[Parent and Spaccapietra, 1998] Parent, C. and Spaccapietra, S. Issues and approaches of database integration. Communications of the ACM, 41(5): 166-178, 1998.

[Pedersen *et al.*, 2004] Pedersen, T., Patwardhan, S., Patwardhan, S. WordNet::Similarity – Measuring the Relatedness of Concepts. In Proc. of 19th National Conf.on AI, San Jose, CA. 1024-1025. 2004.

[Pinto, 1999] Pinto, H. Some issues on ontology integration. In: Proceedings of the IJCAI-99 Workshop on Ontologies and Problem-Solving methods (KRR5), Stockholm, Sweden, pp. 7-1 – 7-12, 1999.

[Protégé, 2008] Protege : an ontology editor and a knowledgebase editor developed by Stanford University. 2008. Available at: http://protege.stanford.edu/.

[Rahm and Bernstein, 2001] Rahm, E. and Bernstein, P. A survey of approaches to automatic schema matching. The VLDB Journal, 10(4):334-350, 2001.

[Rasmussen, 1992] Rasmussen, E. Clustering Algorithms. In Information Retrieval: Data Structures & Algorithms. William B. Frakes and Ricardo Baeza –Yates (Eds.), Prentice Hall, 1992.

[Roddick, 1995] Roddick, J. A survey of schema versioning issues for database systems. Information and Software Technology, 37(7):383-393, 1995.

[Sheth and Larson, 1990] Sheth, A. and Larson, J. Federated database systems for managing distributed, heterogeneous, and autonomous databases. ACM Computing Surveys, 22(3): 183-236, 1990.

[Sirin *et al.*, 2007] Sirin, E., Parsia, B., Grau, B., Kalyanpur, A., and Katz, Yarden. Pellet: a practical OWL-DL reasoner. Journal of Web Semantics, 5, 2007.

[Smith and Waterman, 1981] Smith, T. and Waterman, M. Identification of common molecular subsequences. Journal of Molecular Biology, 147(1): 195-197, 1981.

[Sowa, 1997] Sowa, F. Principles of ontology. onto-std.archive, Knowledge Systems Laboratory Stanford University 1997. http://www-ksl.stanford.edu/onto-std/mailarchive/0136.html

[Spaccapietra and Parent, 1991] Spaccapietra, S. and Parent, C. Conflicts and correspondence assertions in interoperable databases. SIGMOD Record 20(4)-49-54 1991.

[Staab and Stuckenschmidt, 2006] Staab, S. and Stuckenschmidt, H, editors. Semantic web and peer-to-peer. Springer, Heidelberg (DE), 2006.

[Stoilos *et al.*, 2005] Stoilos, G., Stamou, G., and Kollias, S. A string metric for ontology alignment. In Proc. 4th International Semantic Web Conference (ISWC), vol. 3729 of LNCS, pp. 624-637, Galway (IE), 2005.

[Straccia and Troncy., 2005] Straccia, U. and Troncy, R. oMAP: Combining classifiers for aligning automatically OWL ontologies. In Proc. 6th International Conf. on Web Information Systems Engineering (WISE), pp. 133-147, New York (NY US) 2005.

[Stumme and Mädche, 2001a] Stumme, G. and Mädche, A. Ontology Merging for Federated Ontologies on the Semantic Web. IJCAI'01 Workshop on Ontologies and Information Sharing (2001).

[Stumme and Mädche, 2001b] Stumme, G. and Mädche, A. FCA-Merge: Bottom-up merging of ontologies. In Proc. 17th International Joint Conference on Artificial Intelligence (IJCAI), pp. 225-234, Seattle (WA US), 2001.

[Tsarkov and Horrocks, 2006] Tsarkov, D. and Horrocks, l. FaCT++ description logic reasoner: system description. In Proc. 3rd International Joint Conf. on Automated Reasoning (IJCAR), vol. 4130, Springer, LNCS, pp. 292-297 Seattle (WA US), 2006.

[Ullman, 1997] Ullman, J. Information integration using logical views. In Proc. of the 6th Int. Conf. on Database Theory (ICDT'97), vol. 1186 of LNCS, pp. 19-40. 1997.

[UMBC-Ontology] eBiquity Publication Ontology. UMBC ontology. Available at: http://ebiquity.umbc.edu/ontology/publication.owl.

[Valtchev and Euzenat, 1997] Valtchev, P. and Euzenat, J. Dissimilarity measure for collections of objects and values. In Proc. 2nd Symposium on Intelligent Data Analysis (IDA), vol. 1280 of LNCS, pp. 259-272, London (UK), 1997.

[Valtchev, 1999] Valtchev, P. Construction automatique de taxonomies pour l'aide a la representation de connaissances par objets. These d'informatique, Universite Grenoble 1 Grenoble (FR), 1999.

[Wache *et al.*, 2001] Wache, H., Voegele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., and Hubner, S.. Ontology-based integration of information - a survey of existing approaches. In Proc. IJCAI Workshop on Ontologies and Information Sharing, pp. 108-117, Seattle (WA US), 2001.

[Wu and Palmer, 1994] Wu, Z. and Palmer M. Verb semantics and lexical selection. In Proc. 32nd Annual Meeting of the Association for Computational Linguistics (ACL), pp. 133-138, Las Cruces (NM US), 1994.

[Zaihrayeu, 2006] Zaihrayeu, I. Towards Peer-to-Peer Information Management Systems. PhD thesis. International Doctorate School in Information and Communication Technology, University of Trento, Trento (IT), March 2006.

[Zhu, 1999] Zhu, Y. A Framework for Warehousing the Web Contents. In proc. of the 5th Int. Computer Science Conference (ICSC'99), vol. 1749 of LNCS, pp. 773-799, Hong Kong, China, December 1999.

Appendix A: Semantics of Description Language AL

Constructor	Syntax	Semantics
Atomic Concept	A	$A^I \subseteq \Delta^I$
Atomic Role	R	$R^I \subseteq \Delta^I \times \Delta^I$
Top/Universal Concept	T	$\top^I = \Delta^I$
Bottom Concept	$\bot$	$\bot^I = \varnothing$
Negation	$\neg A$	$(\neg A)^I = \Delta^I \setminus A^I$
Intersection	$C \cap D$	$(C \cap D)^I = C^I \cap D^I$
Value Restriction	$\forall R.C$	$(\forall R.C)^I = \{a \in \Delta^I \mid \forall b: (a, b) \in R^I \rightarrow b \in C^I\}$
Limited Existential Quantification	$\exists R.T$	$(\exists R.T)^I = \{a \in \Delta^I \mid \exists b \in \Delta^I: (a, b) \in R^I\}$

Semantics of more constructors in AL

- **Union(U)**

Union (U)	$C \cup D$	$(C \cup D)^I = C^I \cup D^I$

- **Full Existential Quantification ε**

Full Existential Quantification ε	$\exists R.C$	$(\exists R.C)^I = \{a \in \Delta^I \mid \exists b \in \Delta^I: (a, b) \in R^I \wedge b \in C^I\}$

- **Number Restrictions (N)**

Number Restrictions (N)	$\exists_{\geq n} R$	$(\exists_{\geq n} R)^I = \{ a \in \Delta^I \mid \|\{b \mid (a, b) \in R^I\}\| \geq n\}$
	$\exists_{\leq n} R$	$(\exists_{\leq n} R)^I = \{ a \in \Delta^I \mid \|\{b \mid (a, b) \in R^I\}\| \leq n\}$

- **Qualified Number Restrictions (Q)**

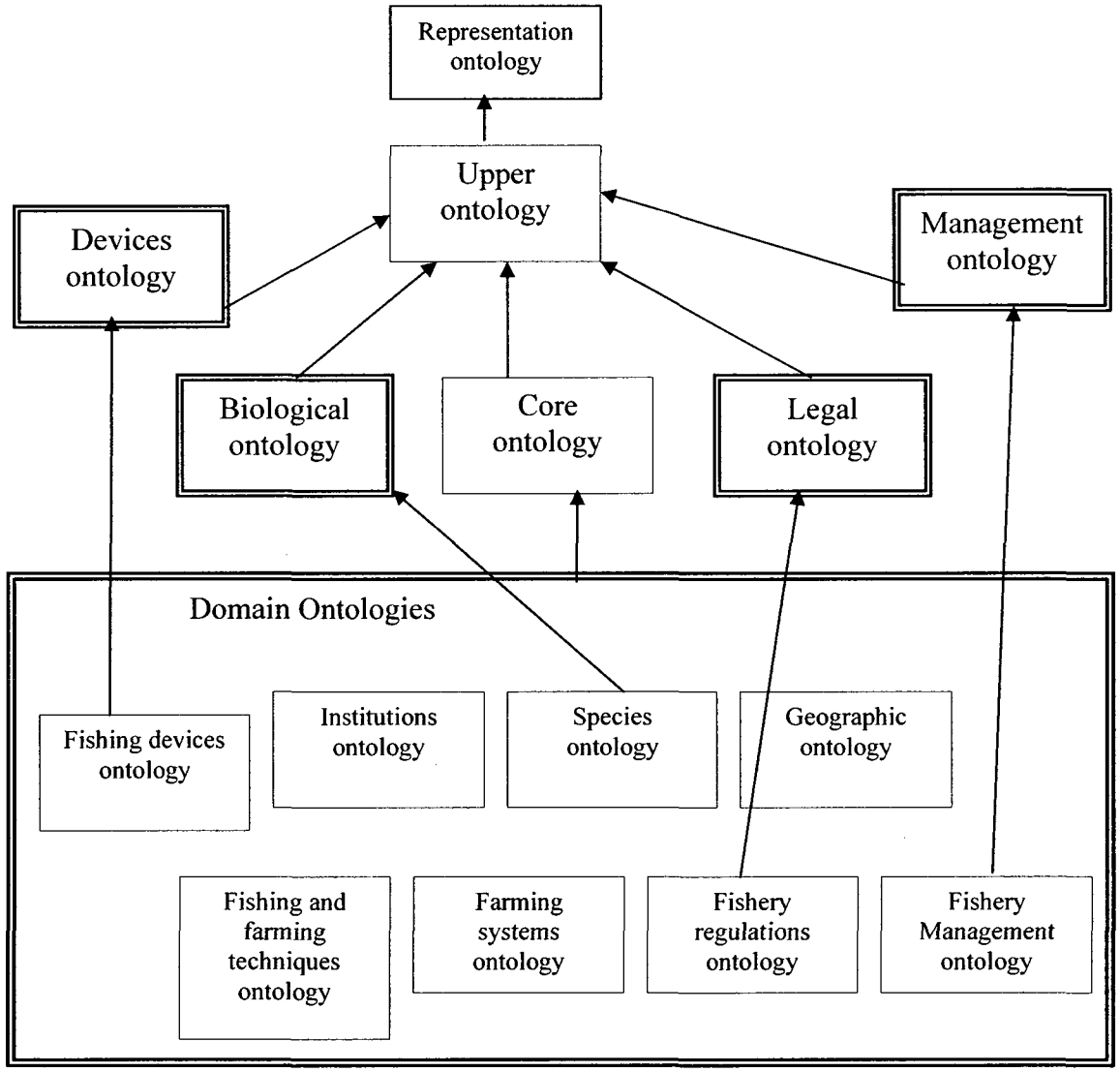
Qualified Number Restrictions (Q)	$\exists_{\geq n} R.C$	$(\exists_{\geq n} R.C)^I = \{ a \mid \#\{b:(a, b) \in R^I \text{ and } b \in C^I\} \geq n\}$
	$\exists_{\leq n} R$	$(\exists_{\leq n} R.C)^I = \{ a \mid \#\{b:(a, b) \in R^I \text{ and } b \in C^I\} \leq n\}$

where, # denotes the set cardinality

- **Inverse Role (I)**

Inverse Role (I)	R^-	$(R^-)^I = \{(b, a\} \in \Delta^I \times \Delta^I \mid (a, b) \in R^I)$

Appendix B: (Re)use of Ontologies



Architecture of the fishery ontology library [Gangemi *et al.*, 2002]; double frames mean use of external ontology

Appendix C: Applying different similarity orders

Following the illustrative scenario given in Sec 5.1.3.

Consider applying three similarity measures m_1, m_2, and m_3. Where, m_1 indicates the string similarity measure, m_2 indicates the linguistic similarity measure, and m_3 indicates the structure similarity measure. So, applying different orders of the similarity measures on the matching states will not affect the states' rank, but the states overall score value.

1- In this case: first we applied m_1 and m_2 in level 1. Then m_3 is applied in level 2.

Level 1			**Level 2**	**Over all score**
State	**String**	**Linguistic**	**Structure**	**S**
e_1	0.4	0.64	0.5	**0.77**
e_2	0.103	0.305	0.0	0.204
e_3	0.466	0.527	0.0	0.497
e_4	0.272	0.291	0.0	0.282
e_5	0.169	0.163	0.0	0.166
e_6	0.269	0.265	0.0	0.267

The states rank based on their overall score is: e_1, e_3, e_4, e_6, e_2, and e_5

2- In this case: we first applied m_1 and m_3 in level 1. Then m_2 is applied in level 2.

Level 1			Level 2	Over all score
State	**String**	**Structure**	**Linguistic**	**S**
e_1	0.4	0.5	0.64	**0.802**
e_2	0.103	0.0	0.305	0.3407
e_3	0.466	0.0	0.527	0.6372
e_4	0.272	0.0	0.291	0.3874
e_5	0.169	0.0	0.163	0.2337
e_6	0.269	0.0	0.265	0.3639

The states rank based on their overall score is: e_1, e_3, e_4, e_6, e_2, and e_5

3- In this case: we first applied m_2 and m_3 in level 1. Then m_1 is applied in level 2.

Level 1			Level 2	Over all score
State	**Structure**	**Linguistic**	**String**	**S**
e_1	0.5	0.64	0.4	**0.77**
e_2	0.0	0.305	0.103	0.204
e_3	0.0	0.527	0.466	0.497
e_4	0.0	0.291	0.272	0.282
e_5	0.0	0.163	0.169	0.166
e_6	0.0	0.265	0.269	0.267

The states rank based on their overall score is: e_1, e_3, e_4, e_6, e_2, and e_5

Appendix D: Time of different similarity orders

- (String +Linguistic) + Structure: shows that the string and linguistic similarities were applied at the first level, then, the structure similarity was applied in the second level.
- (String +Structure) + Linguistic: shows that the string and structure similarities were applied at the first level, then, the linguistic similarity was applied in the second level.
- (Linguistic +Structure) + String: shows that the linguistic and structure similarities were applied at the first level, then, the string similarity was applied in the second level.

Similarity Combinations / **Test No.**	**(String +Linguistic) + Structure**	**(String + Structure) + Linguistic**	**(Linguistic + Structure) + String**
101	8.81 (Sec)	11.6 (Sec)	6.81 (Sec)
103	8.25 (Sec)	11.05 (Sec)	6.7 (Sec)
104	8.25 (Sec)	11.1 (Sec)	6.3 (Sec)