# ONTOLOGY MAPPING: TOWARDS SEMANTIC INTEROPERABILITY IN DISTRIBUTED AND HETEROGENEOUS ENVIRONMENTS

by

**Ming Mao**

B.S. China University of Mining Technology, 1996

M.S. China University of Mining Technology, 1999

Submitted to the Graduate Faculty of

School of Information Sciences in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2008

UNIVERSITY OF PITTSBURGH

SCHOOL OF INFORMATION SCIENCES

This dissertation was presented

by

Ming Mao

It was defended on

March 10th, 2008

and approved by

Dr. Peter Brusilovsky, Committee Member, School of Information Sciences

Dr. Daqing He, Committee Member, School of Information Sciences

Dr. Paul Munro, Committee Member, School of Information Sciences

Dr. Bambang Parmanto, Committee Member, Department of Health Information Management

Dr. Michael Spring, Committee Chair, School of Information Sciences

**ONTOLOGY MAPPING: TOWARDS SEMANTIC INTEROPERABILITY IN DISTRIBUTED AND HETEROGENEOUS ENVIRONMENTS**

Ming Mao, PhD

University of Pittsburgh, 2008

The World Wide Web (WWW) now is widely used as a universal medium for information exchange. Semantic interoperability among different information systems in the WWW is limited due to information heterogeneity, and the non semantic nature of HTML and URLs. Ontologies have been suggested as a way to solve the problem of information heterogeneity by providing formal, explicit definitions of data and reasoning ability over related concepts. Given that no universal ontology exists for the WWW, work has focused on finding semantic correspondences between similar elements of different ontologies, i.e., ontology mapping. Ontology mapping can be done either by hand or using automated tools. Manual mapping becomes impractical as the size and complexity of ontologies increases. Full or semi-automated mapping approaches have been examined by several research studies. Previous full or semi-automated mapping approaches include analyzing linguistic information of elements in ontologies, treating ontologies as structural graphs, applying heuristic rules and machine learning techniques, and using probabilistic and reasoning methods etc. In this paper, two generic ontology mapping approaches are proposed. One is the PRIOR+ approach, which utilizes both information retrieval and artificial intelligence techniques in the context of ontology mapping. The other is the non-instance learning based approach, which experimentally explores machine learning algorithms to solve ontology mapping problem without requesting any instance. The results of the PRIOR+ on different tests at OAEI ontology matching campaign 2007 are

encouraging. The non-instance learning based approach has shown potential for solving ontology mapping problem on OAEI benchmark tests.

## TABLE OF CONTENT

## LIST OF TABLES

## LIST OF FIGURES

## ACKNOWLEDGMENTS

I would like to thank the members of my committee, Dr. Peter Brusilovsky, Dr. Daqing He, Dr. Paul Munro, Dr. Bambang Parmanto and Dr. Michael Spring for their effort and time in guiding me to fulfill the requirement of the degree.

In particular I would like to acknowledge the influence of Dr. Spring who opened the door of the Semantic Web for me.

I would like to thank my parents and my family. Their expectation encourages me to persevere till the end.

I would like to thank my husband, Yefei. I can not reach the point if he is not beside me all the time.

I would like to thank my angel, Jingjing. Her smile is my best reward in the world.

Thank you all for your always support.

# 1.0 INTRODUCTION

This dissertation studies *ontology mapping*: the problem of *finding semantic correspondences between similar elements of different ontologies*. In the dissertation, *elements* denote *classes* or *properties* of ontologies. The goal of this research is to use ontology mapping to make heterogeneous information more accessible.

We begin this chapter by showing that information heterogeneity is a big obstacle to achieve semantic interoperability in the WWW and current solutions are far from enough for this problem. Next, we show that ontology mapping, as a fundamental component in the Semantic Web vision to solve semantic interoperability problem, has been used in numerous applications. Our solutions for ontology mapping are discussed along with the contributions to the ontology mapping community. We list the limitations and delimitations of our research and define terms that will be used in the dissertation. Finally we give a road map to the rest of the dissertation.

## 1.1 THE PROBLEM OF INFORMATION HETEROGENEITY

The vision of the Semantic Web requires information systems that can exchange data and reuse the exchanged data with their intended meanings. This is called *semantic interoperability*. Achieving semantic interoperability among different information systems is very laborious, tedious and error-prone in a distributed and heterogeneous environment like the World Wide Web (WWW).

Information heterogeneity occurs at three levels, i.e., syntax, structure and semantics (Stuckenschmidt and Harmelen 2005). Syntactic heterogeneity is the simplest heterogeneity problem caused by the usage of different data formats. To solve the syntactic heterogeneity, standardized formats such as XML[1], RDF/RDFS[2] and OWL[3] have been widely used to describe data in a uniform way that makes automatic processing of shared information easier.

Though standardization plays an important role for syntactic heterogeneity, it does not overcome structural heterogeneity which occurs as a result of the way information is structured even in homogeneous syntactic environments. For example, one source might model trucks but only classify them into a few categories; while the other source might make very fine-grained distinctions between types of trucks based on their physical structure, weight, purpose, etc. Manually encoded transformation rules as well as some middleware components have been used to solve structural heterogeneity problems (Wiederhold 1992).

Though sophisticated solutions to syntactic and structural heterogeneity have been developed, the problem of semantic heterogeneity is still only partially solved. Semantic heterogeneity occurs whenever two contexts do not share the same interpretation of information (e.g. homonyms and synonyms). For example, as shown in Figure 1.1, Swoogle[4] returns *346* documents when searching for *spring*. The top ranked results show that the same term has many different meanings, e.g. one *spring* means the *season*, the other *spring* means the *ground water*

[1] http://www.w3.org/TR/2004/REC-xml-20040204

[2] http://www.w3.org/TR/2004/REC-rdf-primer-20040210

[3] http://www.w3.org/TR/2003/CR-owl-features-20030818/

[4] Results got from http://swoogle.umbc.edu/ in July, 2007

etc. Another example, illustrated in Figure 1.2, shows that people are always trying to say the same thing in different ways.

Approaches such as using synonym sets, term networks, concept lattices, features and constraints have been proposed as solutions for solving semantic heterogeneity among different information systems (Stuckenschmidt and Harmelen 2005). However those approaches are not sufficient to solve the problem of semantic heterogeneity in the WWW environment.


**Figure 1.1.** Search *spring* in Swoogle

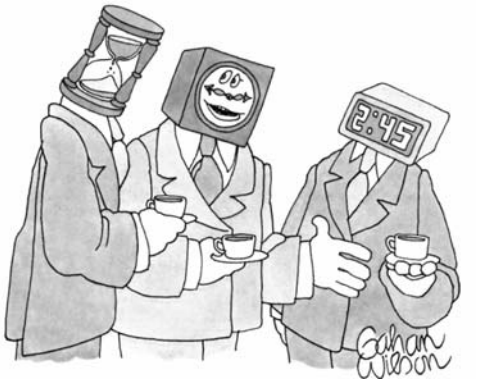



**Figure 1.2.** People are always trying to say the same thing in different ways[1]

[1] By Gahan Wilson

## 1.2 ONTOLOGY MAPPING AND ITS APPLICATION

The vision of the Semantic Web (Berners-Lee, Hendler *et al*. 2001) provides many new perspectives and technologies to overcome the limitation of the WWW. Ontologies are a key component to solve the problem of semantic heterogeneity, and thus enable semantic interoperability between different web applications and services. Given the reality of multiple ontologies over many domains, ontology mapping that aims to *find semantic correspondences between similar elements of different ontologies* has been the subject of research in various communities (Noy 2004; Doan and Halevy 2005).

Ontology mapping has been used in different applications. The following use cases illustrate how semantic correspondences are required in different scenarios, and motivate the importance of ontology mapping.

First of all, ontology mapping is important to the success of the Semantic Web. The pervasive usage of agents or web services is a characteristic of the Semantic Web. However agents or web services may use different protocols that are independently designed. That means when agents or web services meet, there is little chance for them to understand each other without an "interpreter". Therefore ontology mapping is "*a necessary precondition to establish interoperability between agents or services using different ontologies.*" (Ehrig 2006, p.2*)* That is, the mapping between ontologies provides the means for agents and services to either translate their messages or integrate bridge axioms in their own models.

Ontology mapping is also widely used to support data integration and information transformation (Dou, McDermott *et al*. 2005; Crubezy and Musen 2003; Noy and Musen 2003). For example, Figure 1.3 illustrates a simple scenario where data are structured in different formats in two data sources, $D_1$ and $D_2$, which are associated with ontologies $O_1$ and $O_2$

respectively. To integrate instances from $D_1$ to $D_2$, the mapping relation $m$ between $O_1$ and $O_2$ is needed. Many real world cases also demonstrate the need for ontology mapping to support schema/data integration. For example, a web marketplace such as Amazon [1] may need to combine products from multiple vendors' catalogs into its own. A web portal like NCSTRL[2] may want to integrate documents from multiple library directories into its own. A company may want to merge its service taxonomy with its partners. A researcher may want to merge his/her bookmarks with those of his/her peers etc.

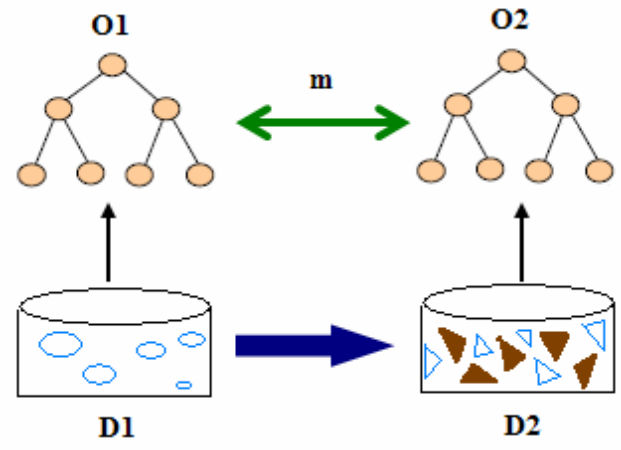



**Figure 1.3.** Ontology mapping in supporting data integration

From the perspective of information retrieval, ontology mapping can support semantic query processing across disparate sources by expanding or rewriting the query using the corresponding information in multiple ontologies (Genesereth, Keller *et al*. 1997; Calvanese, Giacomo *et al*. 2001; Halevy, Ives *et al*. 2003; Mena, Kashyap *et al*. 1996; Gasevic and Hatala 2005). The term used in user's query may be different from those in an ontology. Mapping is

[1] http://www.amazon.com/

[2] http://www.ncstrl.org/

thus used to map the user specific concepts in the query to concepts in ontologies. For example, a user is looking for the director of a movie, e.g., "*Star War*", on the Web. In one movie website, the name of movie is identified as "*moviename*" and the name of its director is identified as "*director*" in its schema. However in another movie website, those two concepts might be identified as "*title*" and "*directorname*" respectively in their schema. Therefore to enable a federated search on those two websites, a mapping between the schemas of those two websites will help us rewrite queries according to different schemas.

The application of ontology mapping can also be found in generating ontology extensions (Dou, McDermott *et al*. 2005) and a number of other scenarios (Euzenat, Bach *et al*. 2004).

## 1.3 CHALLENGES OF ONTOLOGY MAPPING

Though many researchers are working actively on ontology mapping, ontology mapping systems are still a long way from complete. The Web, which is a heterogeneous and distributed environment with a lack of central control, requires significant new progress in ontology mapping to make semantic interoperability possible on a large scale. There are many open issues.

First of all, ontology mapping is quite subjective. Ontologies are created to describe the existence of *things* in the world by different people who usually have different viewpoints about what the world looks like. Different users do not share the same understanding of the world and that results in idiosyncratic evaluation of the mapping results. What kind of methodology should be used to evaluate the correctness and completeness of the mapping? Currently, most ontology engineers adopt the standard *precision*, *recall* and *f-measure* from information retrieval research to evaluate mapping results (Shvaiko, Euzenat *et al*. 2006). A few people evaluate mapping

results by counting the number of needed adjustments between found matches and the reference ontology (Melnik, Garcia-Molina *et al*. 2002). However, sometimes no simple measure exists because either the size of ontologies is too large for people to find a complete set of mappings or it is too hard to precisely define mappings. In these cases, is it valid to evaluate mapping results by empirical methodologies such as testing how queries can be answered with the mapping or whether inferences performed by applications can be preserved by the mapping?

Because ontology mapping is a laborious and tedious process, it is desirable to be able to reuse verified mapping results. For example, given two mappings, $m_{A\text{-}B}$ and $m_{B\text{-}C}$, over ontology $O_A$, $O_B$ and $O_C$ respectively, it should be able to efficiently derive the new mapping, $m_{A\text{-}C}$, from existing mappings, $m_{A\text{-}B}$ and $m_{B\text{-}C}$. The need to reuse mapping results not only requires high quality initial mappings but also requires mapping results to be defined in a formal and explicit way so that inference engines can use it for further reasoning.

Finally, ontologies evolve all the time, and thus the mapping between the evolved ontologies needs to change (evolve) accordingly. For example, assume there exist two ontologies, source ontology $O_S$ and target ontology $O_T$, and a mapping $M_{S\text{-}T}$ between $O_S$ and $O_T$. Now, $O_S$ evolves from $O_{S1}$ to $O_{S2}$. To create a new mapping $M_{S2\text{-}T}$ between $O_{S2}$ and $O_T$, the ideal method is to integrate $M_{S\text{-}T}$ (i.e., $M_{S1\text{-}T}$) with the change between $O_{S1}$ and $O_{S2}$, (i.e., $M_{S1\text{-}S2}$) by reasoning between two mappings. Such integration operations require a complete and explicit representation of mapping results. However, ontology mapping, especially as an automatic process, is error-prone and thus incomplete and imprecise. Therefore, how to represent different types of and sources of incomplete and imprecise mappings, as well as how to use this information to perform reasoning services across the mapped ontologies are still challenges.

In the dissertation, we adopt the *precision*, *recall* and *f-measure* as our evaluation criteria, consistent with the approach used by other researchers. We also represent our mapping results as a list of mapping pairs, following the format required in the annual OAEI ontology matching campaign[1]. Please see an example of mapping format in Figure 4.11.

## 1.4 GOALS OF THE DISSERTATION

The existence of information heterogeneity and the importance of ontology mapping in different applications motivate our research interest in the area of ontology mapping. Therefore, the ultimate goal of our research is *to solve the problem of ontology mapping, and thus enable semantic interoperability between different web applications and services in the WWW*. More specifically, we aim to develop a new generic approach to automatically map ontologies with minimum human effort. This is because manual mapping becomes impractical as the complexity and volume of ontologies increases. Alternatively developing fully or semi-automated mapping algorithms/tools has attracted the interest of researchers in various areas (Noy 2004; Noy, Doan *et al.* 2005; Rahm and Bernstein 2001).

[1] http://oaei.ontologymatching.org

## 1.5 OVERVIEW OF OUR SOLUTIONS

Generally speaking, our efforts for ontology mapping problem can be divided into two directions. One is an integrated mapping approach; the other is learning based approach. Both of them are generic ontology mapping approaches and suitable for different mapping situations.

The integrated approach, which we call the PRIOR+ (Mao and Peng 2007; Mao, Peng *et al.* 2008), is based on information retrieval and artificial intelligence techniques. The name PRIOR+ comes from the ontology mapping tool, the PRIOR (i.e., the *p*rofile p*r*opagation and *i*nformation *r*etrieval based ontology mapping tool) (Mao and Peng 2006; Mao, Peng *et al.* 2007). PRIOR+ improves on PRIOR in two ways. First, the PRIOR+ proposes a harmony based adaptive aggregation method to aggregate multiple similarities without given golden standards. Second, the PRIOR+ innovatively integrates the IAC neural network to consider various constraints in the context of ontology mapping when the harmony of integrated similarities of ontologies is not good enough.

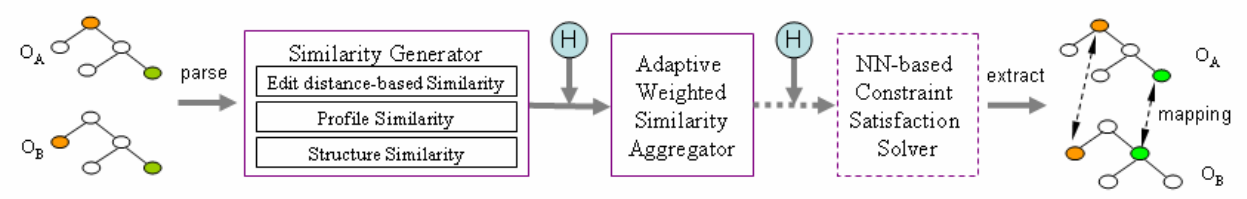



**Figure 1.4** The architecture of the PRIOR+ approach

The PRIOR+ approach consists of three major modules, i.e., a similarity generator, an adaptive similarity aggregator, and a neural network based constraint satisfaction solver (optional). The architecture of the PRIOR+ is shown in Figure 1.4, where *H* denotes the harmony of a similarity. The similarity generator measures three kinds of similarities. The similarity aggregator measures the harmony of each similarity and then adaptively aggregates them. The

constraint satisfaction solver deals with various ontology constraints in neural network model, which is optional as illustrated with a dashed line and box. More details of the approach can be found in §4.0 .

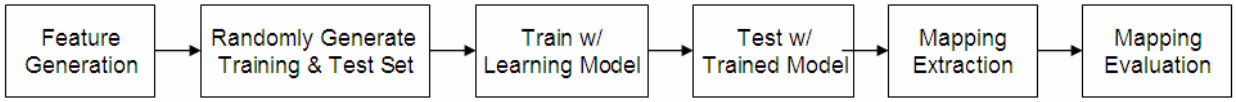



**Figure 1.5** The major steps in learning-based approach

The learning-base approach (Figure 1.5) is a non-instance based machine learning approach that explores general linguistic, structure and Web features of ontologies using a Support Vector Machine (SVM) algorithm. Like other learning based ontology mapping approach, our approach treats ontology mapping as a machine learning problem and thus utilizes machine learning techniques to solve the problem. But unlike other learning based approach, our approach is generic (i.e., the target in our learning model is not each specific element in ontologies) and does not rely on the availability of instances in ontologies. More details of the approach can be found in §5.0 .

## 1.6 CONTRIBUTIONS OF THE DISSERTATION

The contributions of the dissertation can be summarized as follows:

1. A comprehensive review on the state-of-the-art ontology mapping approaches.
2. Exploration of multiple similarities such as edit distance based similarity, profile similarity and structure similarity to support ontology mapping.

3. Use of information retrieval techniques to evaluate the profile similarity of different elements in a vector space model.
4. Development of a measure *harmony* to estimate the reliability of different similarities without given ground truth.
5. Development of a harmony based adaptive aggregation method to aggregate various similarities.
6. Integration of the interactive activation and competition (IAC) neural network in the context of ontology mapping to search for a global optimal solution that best satisfies ontology constraints.
7. Exploration of a non-instance based machine learning approach for ontology mapping by treating it as a binary classification problem.

## 1.7 LIMITATIONS AND DELIMITATIONS

The limitations of the research are.

1. Only those ontologies that come from the same or similar domain will be mapped. This limitation is based on the observation that the chance for two information systems from completely different domains to interact with each other is relatively small in real world.
2. The to-be-mapped ontologies should be represented in the same language and in English only. We are not working on cross language ontology mapping problem.
3. Currently our approaches only work on 1-1 mapping due to the limitation of the mapping extraction algorithm and the constraints chosen for the IAC neural network. For example, the mapping extraction algorithm adopted in this dissertation is naïve descendant

extraction algorithm (Meilicke and Stuckenschmidt 2007), which has a limitation that even though the similarity between two elements is weak a mapping always outputs as a result. Moreover, the constraint that "only 1-1 mapping is allowed" restricts the mapping cardinality of our approach.

The following delimitations are set on the scope of our research:

1. Mapping sources. Though many approaches of schema matching in database area can generally be applied to ontology mapping, ontology mapping relies more heavily on features of its concepts' definitions and explicit semantics of these definitions. Therefore, our research will only deal with mapping different ontologies that are represented in a formal way. How to map relational or XML schemas will not be included in this work though various approaches in this area will be reviewed in §3.1.
2. Mapping elements. Ontology mapping can be established between classes, properties and instances. In this dissertation, we focus on finding mapping relations between *classes* and *properties* instead of instances due to the lack of standards. Almost all available standards do not provide instance-level mapping results for evaluation purposes. Meanwhile, we only try to find mapping between elements that share the same type. For example, a class in one ontology can not be mapped to a property or an instance in another ontology.
3. Mapping granularity. The semantic correspondences in ontology mapping, as defined in §2.2, include different relationships, e.g., equivalent (=), broader ($\supseteq$), narrower ($\subseteq$), disjoint ($\perp$), joint ($\cup$), etc. This research primarily interested in identifying the equivalent relationship (i.e., =) between different elements of different ontologies. This is because equivalent relations are the most frequent relations between ontologies and other

relationships such as broader and narrower relations can be generalized based on identified equivalent relations. We will not try to identify disjoint and joint relations.

4. Mapping cardinality. The cardinality of ontology mapping varies from simple mapping (i.e., 1:1 mapping) to complex mappings (i.e., 1:n, n:1 and n:n). Though it is useful and desirable to find complex mappings in real world cases, 1:1 mapping is prerequisite and fundamental for establishing complex mappings. Therefore this research will focus on finding 1:1 mapping in ontologies and save the establishment of complex mappings for future work. Moreover, the mapping extraction algorithm adopted in this dissertation is the naïve descendant extraction algorithm (Meilicke and Stuckenschmidt 2007). One limitation comes from this algorithm is: even though the similarity between two elements is weak a mapping always outputs as a result.

## 1.8 TERM DEFINITIONS

**DEFINITION 1. SEMANTICS** *is an individual's interpretation of data according to his/her understanding of the world (Uschold 2003).*

**DEFINITION 2. INTEROPERABILITY** *is the ability of two or more systems to exchange information and to use the information that has been exchanged (IEEE 1990).*

**DEFINITION 3. SEMANTIC INTEROPERABILITY** *is the capability of different information systems to communicate information consistent with the intended meaning of the encoded information. (Patel, Koch et al. 2004) (p.8)*

**DEFINITION 4. INFORMATION HETEROGENERITY** *refers to the difference between information. Information heterogeneity can happen at three levels, i.e., syntax, structure and semantics levels. (Stuckenschmidt and Harmelen 2005)(p.2)*

**DEFINITION 5. SCHEMA** *is a structure describing how data can be stored, accessed, and interpreted by user and applications. Sample schemas include relational schemas, XML schemas and ontological schemas in OWL. (Do 2006)(p.3)*

**DEFINITION 6. ONTOLOGY** *is a formal, explicit specification of a shared conceptualization. (Gruber 1993)(p.199)*

**DEFINITION 7. UPPER ONTOLOGY** *is an ontology that provides common reference terminologies for other ontologies to extend.*

**DEFINITION 8. ONTOLOGY MAPPING** *is the determination of semantic correspondences between similar elements in different ontologies*. Here, *semantic correspondence* refers to different relationships, e.g. the equivalence (=), the broader ($\supseteq$), the narrower ($\subseteq$) etc., and *elements* could be *classes*, *properties*, *instances* and *relations* between the instances of an ontology. (§2.2)

**DEFINITION 9. SIMILARITY** *is a confidence measure between two elements in different ontologies. The similarity is expressed in a mathematical number that typically range in [0..1].*

## 1.9 OUTLINE

The rest of this paper is organized in the following manner. Chapter 2 defines the ontology mapping problem that we consider in the dissertation, and gives out the format that we use to represent mapping results. Chapter 3 reviews representative work by different researchers for

ontology mapping as well as schema matching in the database area. Chapter 4 thoroughly described the PRIOR+ approach and evaluates its performance using the tasks from OAEI campaign 2007. Chapter 5 introduces a non-instance learning based ontology mapping approach and gives out the experimental results of it on the benchmark tests from OAEI campaign 2007. Chapter 6 is a summary of the whole thesis and the outlook for future work.

# 2.0 PROBLEM DEFINITION

This chapter defines ontology mapping. We begin by introducing the concepts of ontology and its heterogeneity. Next, we define ontology mapping based on the definitions given by other researchers. We then describe an example of two ontologies and the possible mappings between them. Finally we give out a formal statement to represent mapping results.

## 2.1 ONTOLOGY AND ONTOLOGY HETEROGENEITY

The vision of the Semantic Web (Berners-Lee, Hendler *et al* 2001) provides many new perspectives and technologies to overcome the limitation of the WWW. Ontologies are a key component to solve the problem of semantic heterogeneity, and thus enable semantic interoperability between different web applications and services.

An ontology is a formal, explicit specification of a shared conceptualization (Gruber 1993) (p.199), where "*conceptualization*" refers to an abstract model of phenomena in the world by having identified the relevant concepts of those phenomena, "e*xplicit*" means that the type of concepts used, and the constraints on their use are explicitly defined, "*formal*" refers to the fact that the ontology should be machine readable and "*shared*" reflects that ontology should capture consensual knowledge accepted by the communities (Fensel 2001; Fensel, Hendler *et al*. 2002).

Though ontologies have gained popularity in many communities as a means to establish formal and explicit vocabulary that applications can share, it is unrealistic to expect a universal ontology for the WWW. For example, Figure 2.1 illustrates two real ontologies, AKTors[1] vs. eBiquity[2]. These two ontologies use different definition, structure and notion to describe the same concept of *publication*. The problem of ontology heterogeneity induces research interest from different areas in *ontology mapping*.

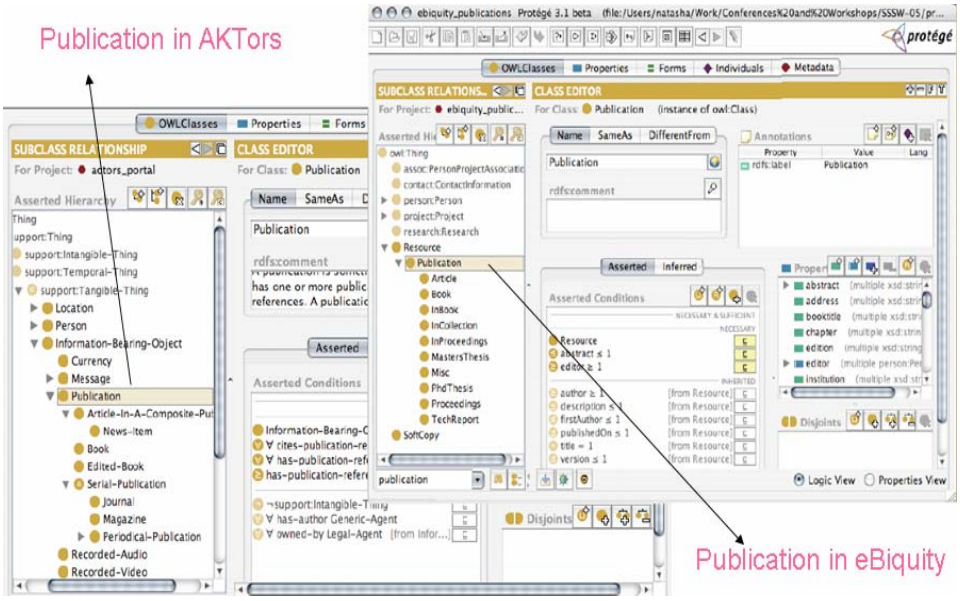



**Figure 2.1.** The *publication* in two real ontologies

[1] http://www.aktors.org/ontology/

[2] http://ebiquity.umbc.edu/ontology/publication.owl

## 2.2 THE DEFINITION OF ONTOLOGY MAPPING

*Ontology mapping* is also known as *ontology matching*, or *ontology alignment*. In the dissertation, only the term of *ontology mapping* is used. Ontology mapping is different from ontology merging. Ontology mapping tries to make the source ontologies consistent and coherent with one another while keeping them separate. In contrast ontology merging aims to create a single coherent ontology that includes the information from all the sources.

There are many different definitions of ontology mapping, depending upon its application and its intended outcome. Sample definitions of ontology mapping include:

1. Ontology mapping is *"a set of formulae that provide the semantic relationships between the concepts in the models"* (Madhavan, Bernstein *et al*. 2002) *(p.122)*.

2. Ontology mapping is used to *"establish correspondences among the source ontologies, and to determine the set of overlapping concepts, concepts that are similar in meaning but have different names or structure, and concepts that are unique to each of the sources"* (Noy and Musen 2000) *(p.450)*.

3. Ontology mapping aims to "*map concepts in the various ontologies to each other, so that a concept in one ontology corresponds to a query (i.e. view) over the other ontologies"* (Calvanese, Giacomo *et al*. 2001) *(p. 11)*.

4. "*Given two ontologies O1 and O2, mapping one ontology onto another means that for each entity (concept C, relation R, or instance I) in ontology O1, we try to find a corresponding entity, which has the same intended meaning, in ontology O2"* (Ehrig and Staab 2004) *(p.685).*

In this research, we define *ontology mapping* as to *find a set of semantic correspondences between similar elements in different ontologies*. Here, *semantic correspondence* refers to different relationships, e.g. the equivalence (=), the broader ($\supseteq$), the narrower ($\subseteq$), the disjoint ($\perp$), the joint ($\cup$) etc., and *elements* could be *classes*, *properties*, *instances* and *relations* between the instances of an ontology. The limitation and delimitation of our research can be found in §1.7. That is, in the thesis, we focus on finding *1-1* mappings with "equivalent" relationship between *classes* and *properties* in two ontologies from the same or similar domain.

## 2.3 AN EXAMPLE OF ONTOLOGY MAPPING

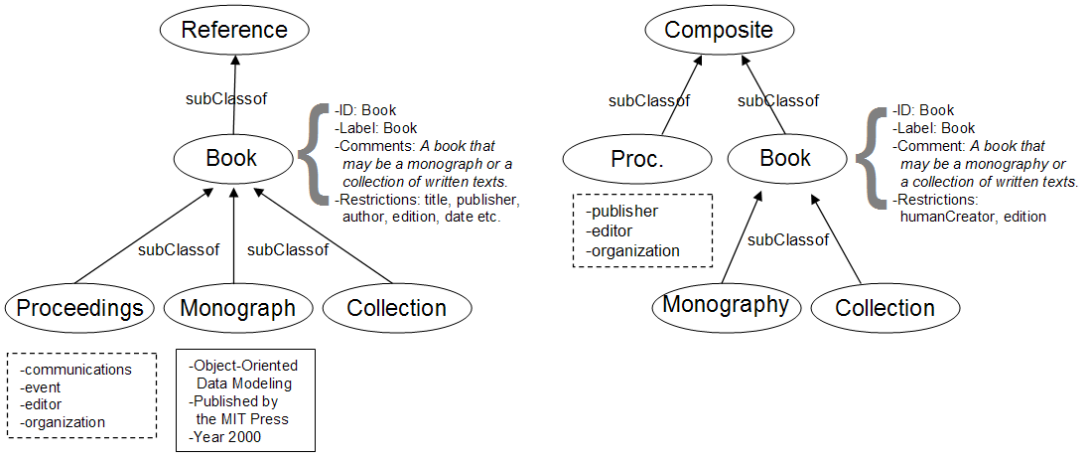



**Figure 2.2.** Two sample bibliographic ontologies

Figure 2.2 shows two sample bibliographic ontologies, in which the ellipses indicate classes (e.g., "*Reference*", "*Composite*", "*Book*" and "*Proceedings*" etc.), the dashed rectangles indicate properties (e.g., "*publisher*", "*editor*", "*organization*" etc.), the lines with arrowhead indicate "*subClassof*" relation between two classes, and the solid rectangle indicates an instance

that is associated with the class of "*Monograph*" (i.e., "*object-oriented data modeling*" published by the MIT Press at 2000). Each class and property has some information to describe and restrict it. For example, the descriptive information of "*Book*" in the left ontology includes its ID, label, comment and restrictions such as title, publisher etc. In Figure 2.2 candidate mappings between *classes* include *Reference* and *Composite*, *Book* and *Book*, *Monograph* and *Monogaphy*, *Collection* and *Collection*, *Proceedings* and *Proc.*

## 2.4 THE REPRESENTATION OF ONTOLOGY MAPPING

The input of ontology mapping is *two homogeneous ontologies*, $O_1$ and $O_2$, expressed in the form of formal taxonomies or ontologies. The output is a *mapping*, also called the *mapping result*, between the input taxonomies or ontologies. Mapping can be represented in different ways depending on its intended use (Kalfoglou and Schorlemmer 2003). For example, mappings can be represented as queries (Calvanese, Giacomo *et al*. 2001), bridging axioms (Dou, McDermott *et al*. 2005) or an instance in a mapping ontology (Crubezy and Musen 2003; Gasevic and Hatala 2005).

We define mapping results as a statement of 4-tuple (shown as below), where *m* is a correspondence that specifies a specific element $e_{1i}$ of $O_1$ has a relationship *r* with a specific element $e_{2j}$ of $O_2$, and the correspondence holds a confidence measure *s*, which is a number typically ranged in [0..1].

$$m(e_{1i}, e_{2j}, r, s)$$

For example, in Figure 2.2, candidate mappings can be represented as: *m(Reference, Composite, =, .11), m(Book, Book, =, 1), m(Monograph, Monography, =, .9), m(Collection, Collection, =, 1), m(Proceeding, Proc, =, .36)* etc.

Finally the mapping results are expressed in the format required by the annual OAEI campaign (see example in Figure 4.11). Please note even though the similarity between two elements is weak a mapping is always output as a result due to the limitation of naïve descendant extraction algorithm (Meilicke and Stuckenschmidt 2007) and delimitation of our 1-to-1 mapping cardinality. For example, given the similarity matrix shown in Figure 4.4, when using the extraction algorithm to extract mapping results, *m(Book, Book, =, 1), m(Collection, Collection, =, 1), m(Monograph, Monography, =, .9), m(Proceeding, Proc, =, .36)* will be output as mapping results in sequence. Finally, even though the similarity score between *Reference* and *Composite* is as low as .11, *m(Reference, Composite, =, .11)* will be output as a mapping because it is the only mapping candidate left.

# 3.0 RESEARCH ON MAPPING APPROACHES

Though ontologies are more semantically complex and are often larger than relational/XML schemas, many issues that ontology researchers grapple with in ontology mapping are similar to the issues that database researchers have addressed in schema matching (Noy 2004). In fact, there has been a convergence with database researchers employing more expressive components of schema definition in schema matching, and ontology researchers paying more attention on the well-examined experience from database community. Therefore before reviewing approaches to ontology mapping, the work that has been done in schema matching is briefly reviewed.

## 3.1 SCHEMA MATCHING IN THE DATABASE COMMUNITY

Schema matching is defined as *finding semantic correspondences between elements of different relational or XML schemas* (Do and Rahm 2002). Schema matching approaches are usually classified as rule-based or learning-based approaches. A more comprehensive classification of schema matching can be found in (Rahm and Bernstein 2001; Shvaiko and Euzenat 2005).

Rule-based approaches generally exploit schema information such as data types and structures, element names, number of sub-elements, and integrity constraints. Various rules have been considered in different systems. For example, in the TranScm system (Milo and Zohar 1998), rules such as “two elements match if they have the same name (allowing synonyms) and

the same number of subelements" were used. In the ARTEMIS system (Castano and Antonellis 1999), the similarity of schema elements were computed as a weighted sum of the similarities of name, data type, and substructure. In the CUPID system (Madhavan, Bernstein *et al*. 2001), rules that categorize elements based on names, data types, and domains were employed. In the DIKE system (Palopoli, Terracina *et al*. 2003), the similarity of the characteristics of the elements and the similarity of related elements were used to compute the similarity between two schema elements.

Learning-based approaches usually use a variety of learning techniques and exploit both schema and data information. For example, in the SemInt system (Li and Clifton 2000), a neural network learning approach is used to match schema elements based on attribute specifications (e.g., data types, scale, the existence of constraints) and statistics of data content (e.g., maximum, minimum, average, and variance). In the LSD system (Doan, Domingos *et al*. 2001), Naive Bàyes over data instances was employed, and a novel learning solution to exploit the hierarchical nature of XML data was developed. In the iMAP system (Dhamankar, Lee *et al*. 2004) two schemas were matched by analyzing the description of objects found in both sources. In the Autoplex and Automatch systems (Berlin and Motro 2001; Berlin and Motro 2002), a Naive Bàyes learning approach was used to match elements by utilizing data instances.

Table 3.1 summarizes the major differences between rule-based approaches and learning-based approaches for schema matching based on the comparison given in (Doan 2002).

**Table 3.1** Comparison of rule-based and learning-based approaches

| | Rule-based Approaches | Learning-based Approaches |
|---|---|---|
| Advantage | • relatively inexpensive (no training needed)<br>• fast (schema operation only, does not work on data instances)<br>• capture user knowledge (e.g. phone#, zip codes) quickly and concisely | • can exploit data information and past matching activities<br>• can figure out new rules by learning from training sets |
| Disadvantage | • can not exploit data information (e.g. word frequencies, distribution) effectively<br>• can not exploit previous matching efforts<br>• has serious problem when rule are difficult to formulate (e.g. movie description vs. user comments) | • time-consuming (need to preprocess data)<br>• training required<br>• has difficulty in learning certain type of knowledge (e.g. phone#, zip, country names) |

## 3.2 STATE OF THE ART ONTOLOGY MAPPING APPROACHES

Ontology mapping is defined as *finding semantic correspondences between similar elements of different ontologies* (§2.2). Though ontology mapping is very similar to schema matching (Madhavan, Bernstein *et al*. 2001; Rahm and Bernstein 2001) in that both try to exploit lexical and structural information to find correspondences (i.e. mappings), ontology mapping often goes further due to the characteristics of ontologies (Noy 2004; Noy, Doan *et al*. 2005). For example, ontologies usually have more constraints specified than schemas, and thus ask for mapping methods that can automatically exploit these constraints. Ontology mapping also gets more benefits from exploiting semantics of relationships than schema could, such as the semantics of *SubClassOf* or *PartOf* relationships, the attachment of property to a class, domain and range

definitions for properties, and so on. In this section, two architectures for finding correspondences between ontologies are introduced. Following, five major mapping methods that have been used in previous work are described.

### 3.2.1 Two Architectures for Ontology Mapping

Two architectures, *centralized* vs. *decentralized*, have been proposed for ontology mapping. The *centralized* approach used an upper ontology shared by developers of different applications. This approach is intuitive because upper ontology can provide common reference terminologies for domain specific ontologies, and thus greatly facilitates finding correspondences between them. Figure 3.1 illustrates a simple example where three domain specific ontologies are extended from the same upper ontology. The mappings between domain ontologies can be established via interlingua information provided by the upper ontology.

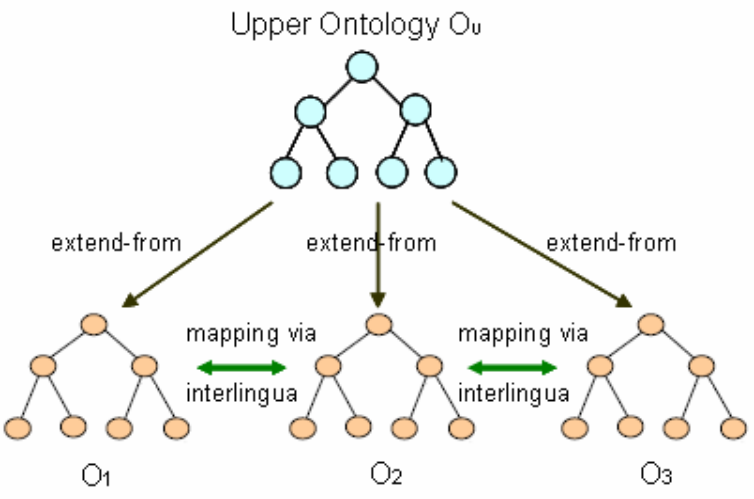



**Figure 3.1.** Mapping via interlingua information available in an upper ontology

Currently, a number of general ontologies that formalize notions such as processes and events, time and space, physical objects, and so on, have been developed and some of them are becoming accepted standards. DOLCE (Gangemi, Guarino *et al*. 2003) and SUMO (Niles and Pease 2001) are two examples that are built with the explicit purpose of being formal upper ontologies. DOLCE (Gangemi, Guarino *et al*. 2003) is an upper ontology developed in the WonderWeb project[1]. It aims at providing a common reference framework for ontologies to facilitate information sharing among them. DOLCE captures ontological categories based on natural language and human common sense in its representation. SUMO (Suggested Upper Merged Ontology) (Niles and Pease 2001) was created by the IEEE Standard Upper Ontology Working Group with the goal of developing a standard upper ontology to promote data interoperability, information search and retrieval, automated inference, and natural language processing. The SUMO ontology defines high level concepts (i.e., Object, ContinousObject, Process, Quantity, and Relation). It also provides axioms in first-order logic to describe properties of these concepts and relations among them.

Though it is surely helpful to have different ontologies refer to the same upper ontology, sometimes such ontology simply does not exist or people do not want to use one. Therefore, rather than using interlingua information in the upper ontology, the *decentralized* approach creates mappings by exploiting different kinds of information between ontologies. For example, structural information (e.g., subclass and superclass relationships, domain and range of properties, and graph structure of ontologies) can give insight into ontologies. Lexical information (e.g., names, definitions, and distance between strings) can help re-rank mapping results. Auxiliary information (e.g., WordNet) provides semantics for the elements in ontologies.

[1] http://wonderweb.semanticweb.org/

Finally, instance information, if available, is useful especially for machine learning approaches where they provide necessary training sets. The following sections introduce five major *decentralized* approaches, i.e., heuristic and rule-based methods, machine learning-based methods, graph-based methods, probabilistic methods, and reasoning and theorem proving methods.

### 3.2.2 Heuristic and Rule-based Methods

Heuristic and rule-based methods for ontology mapping are similar to rule-based approaches for matching relational schemas and XML structures. They both use lexical information (e.g., name, label, and description) and structural information (e.g., key properties, taxonomic structure) to find correspondences. For example, Hovy (Hovy 1998) describes a set of heuristics for semi-automated mapping of domain ontologies to a central ontology. In the approach, the author mainly uses natural-language analysis of concept names and definitions, as well as taxonomic relationships. He first uses natural language processing (NLP) techniques to split composite word names, and then compares substrings of different lengths to find concept names that are similar to each other. The number and the ratio of shared words in the definitions are also considered. Other systems that use heuristic and rule-based methods for ontology mappings include Prompt (Noy and Musen 2000), QOM (Ehrig and Staab 2004), Ontomorph (Chalupsky 2000), and Chimaera (McGuinness, Fikes *et al* 2000).

1. PROMPT

The PROMPT system (Noy and Musen 2000) was originally developed to support ontology merging, guiding users through the process and suggesting which classes and properties can be

merged. Figure 3.2 shows the PROMPT algorithm, where the gray boxes indicate the actions performed by PROMPT and the white box indicates the action performed by the user.

To make the initial suggestions, PROMPT uses a measure of linguistic similarity among concept names and mixes it with the structure of the ontology and user's actions. For each operation, PROMPT performs changes automatically, finds conflicts that the operation may introduce that need to be resolved, and presents new suggestions to the user. For instance, if a user says that two classes in two source ontologies are the same, that means these two ontologies should be merged, then PROMPT analyzed the properties of these classes, their subclasses and superclasses to look for similarities of their definitions and suggest additional correspondences.

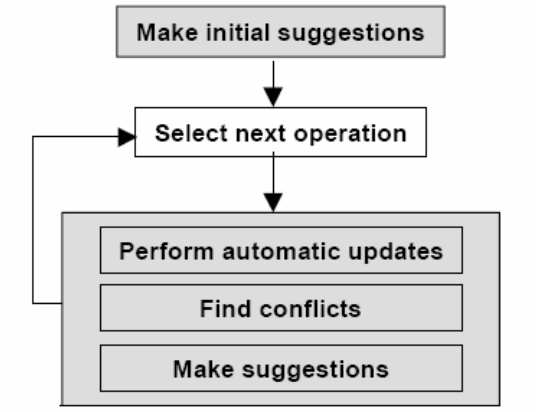



**Figure 3.2.** The workflow of PROMPT algorithm [1]

Their evaluation shows that human experts follow 90% of PROMPT's suggestions. The experts follow 75% of the conflicts-resolution strategies that PROMPT proposes and PROMPT suggests 74% of the total knowledge-based operations invoked by the user. The results indicate that PROMPT is very effective in providing suggestions. Though the PROMPT algorithm is

[1] from (Noy and Musen 2000), p.452

simple and fast, it can be made more efficient by sorting the labels first, thus only requiring the comparison of two neighboring elements in the list. Another limitation of PROMPT is it determines the similarity based on the exact equality of labels and only one similarity is computed. No similarity aggregation is performed.

2. QOM

QOM (Quick Ontology Mapping) (Ehrig and Staab 2004) is based on the hypothesis that mapping algorithms can be streamlined such that the loss of quality is marginal, but the improvement of efficiency is tremendous for the ad-hoc mapping of large size, light weight ontologies. It is defined by the process model shown in Figure 3.3.

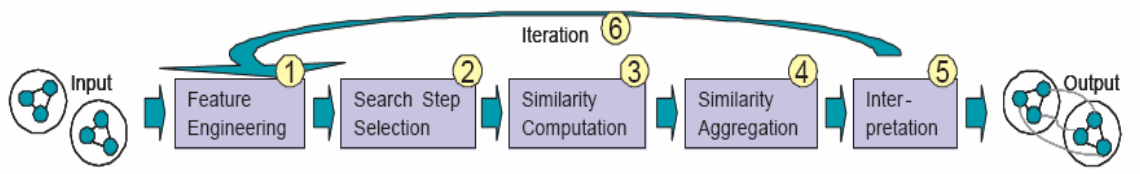



**Figure 3.3.** QOM mapping process [1]

1) First, QOM employs RDF triples as features.

2) Based on the observation that the run time complexity of a mapping algorithm is directly impacted by the number of candidate mapping pairs that need to be examined, in the Search Step Selection process, instead of comparing all entities of the first ontology with all entities of the second ontology, QOM applies a heuristic method that makes use of ontological structures to reduce the quantity of candidate mappings, which is a major ingredient of run-time complexity.

[1] from (Ehrig and Staab 2004), p.686

3) In the Similarity Computation step, the similarity between entities of ontologies is done by using various similarity functions and heuristics. For efficiency reasons, QOM avoids the complete pair-wise evaluation of ontology trees and restricts the number of costly feature comparisons.

4) All individual measures are then input to the similarity aggregation. Instead of applying linear aggregation functions, QOM applies a *sigmoid* function. The *sigmoid* function emphasizes high individual similarities rather than low individual similarities.

5) Two methods are used to interpret similarity results. First, a threshold to discard spurious evidence of similarity is applied. Further mappings are assigned based on a greedy strategy that starts with the largest similarity values first. Since Similarity Aggregation and Interpretation steps are performed once per candidate mapping, they do not impact efficiency.

6) Finally, QOM iterates to find mappings based on lexical information first and knowledge structure later. Through several iterations the quality of the results rises considerably. QOM limits the number of iterations to 10 because empirical findings indicate that further iterations produce insignificant changes. Eventually, the output is a mapping table representing the relationship between the two ontologies.

Depending on the scenario, QOM can be very effective and efficient, reaching high quality levels quickly by a factor of 10 to 100 times compared to approaches such as PROMPT and NOM (a simulation of Anchor-PROMPT algorithm). One problem of QOM is its optimization of mapping approach does decrease the overall mapping quality. Therefore, QOM is recommended only when ontologies are large-scaled.

### 3.2.3 Machine Learning Approaches

Machine learning can be used to creating mappings between ontologies (Doan, Madhaven *et al*. 2003). This method is efficient when instances are available in ontologies and it works better if many instances have text in them rather than references to other instances. The GLUE system (Doan, Madhaven *et al*. 2003) is an example of learning-based ontology mapping systems.

GLUE (Doan, Madhaven *et al*. 2003) is a system whose aim is to semi-automatically create semantic mappings, especially 1-1 correspondences, between the taxonomies of two given ontologies. GLUE first applies statistical analysis to the available data (i.e., joint probability distribution computation). And then it uses multiple learners to exploit information in concept instances and taxonomic structure of ontologies. Next, GLUE uses a probabilistic model to combine results of different learners. Finally, GLUE adopts relaxation labeling approach to search for the mapping configuration that best satisfies the domain constraints and the common knowledge, taking into account the observed similarities.

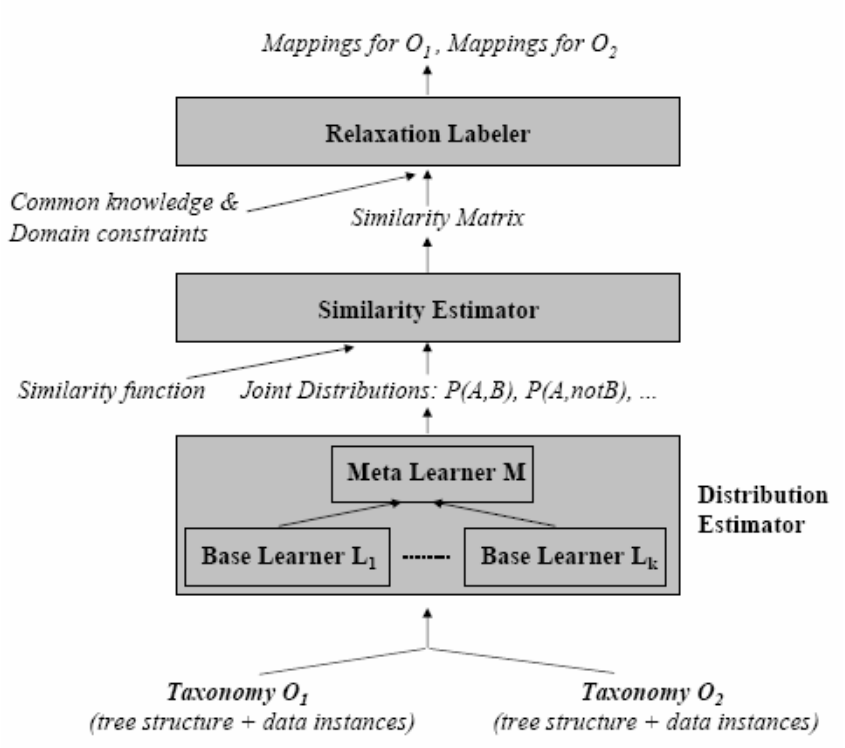



**Figure 3.4.** The architecture of GLUE [1]

GLUE consists of three main modules: Distribution Estimator, Similarity Estimator, and Relaxation Labeler. The Distribution Estimator takes two taxonomies $O_1$ and $O_2$ and their data instances as input. For every pair of concepts ($A \in O_1$, $B \in O_2$), the Distribution Estimator applies machine learning techniques to compute their joint probability distributions. And then it uses a set of base learners and a meta-learner. Each base learner exploits a certain type of information (i.e., the frequencies of words in the text value of the instances, the value formats, the instance names, the characteristics of value distributions, and so on.) from the training instances to build prediction hypotheses. Then, these base learners are applied to the instance and their predictions are combined by a meta-learner to achieve higher classification accuracy than any single base

[1] from (Doan, Madhaven *et al*. 2003), p.307

learner alone. Next, GLUE feeds all the above numbers into the Similarity Estimator, which applies a user-supplied similarity function to compute a similarity value for each pair of concepts, ($A \in O_1$, $B \in O_2$). The output of this module is a similarity matrix between the concepts in the two taxonomies. Finally, the Relaxation Labeler module takes the similarity matrix, domain-specific constraints and heuristic knowledge. It searches for the mapping configuration that best satisfies the domain constraints and the common knowledge, taking into account the observed similarities. The output of GLUE is the mapping configuration. Though GLUE uses IR method to preprocess data, it does not calculate the similarity value for each pair of concepts in a vector space model that is different from what the PRIOR+ approach adopts.

The authors evaluated GLUE on three domains, the results show that GLUE achieves high accuracy across all domains, ranging from 66 to 97%, compared to the best matching results of the base learners (i.e., content learner), of only 52 - 83%. The biggest problem of GLUE is that it requires a large number of instances associated with the nodes in taxonomies and these are not available in most ontology mapping situations.

### 3.2.4 Graph-based Methods

Treating ontologies as graphs and then comparing the corresponding graphs is another method to find mappings between ontologies. Examples of graph-based ontology mapping methods include Anchor-Prompt (Noy and Musen 2001) and Similarity Flooding (Melnik, Garcia-Molina *et al*. 2002).

1. Anchor-PROMPT

Anchor-PROMPT (Noy and Musen 2001), an extension of PROMPT (Noy and Musen 2000), is an ontology merging and mapping tool with a sophisticated prompt mechanism for term matching. It treats an ontology as a directed labeled graph, where concepts are nodes and relations are arcs.

The Anchor-PROMPT algorithm is based on the observation that if two pairs of terms are similar and there are paths connecting them, then the elements in these paths are often similar as well. Anchor-PROMP takes a set of anchors (pairs of related term) from source ontologies as input. These anchors are either manually identified by users or automatically generated by the system. From this set of pre-identified anchors, the Anchor-PROMPT traverses the paths between the anchors on the corresponding ontologies and then computes the terms along these paths to find similar terms. Finally, the Anchor-PROMPT produces a set of pairs of semantically related terms.

For example, in Figure 3.5, there are two pairs of pre-identified anchors, classes A and B and classes H and G, and two parallel paths, one from A to H in Ontology 1 and the other from B to G in Ontology 2. The Anchor-PROMPT traverses the two paths and increments the similarity score between each two classes (i.e., classes C and D, classes E and F) reached in the same step. Then Anchor-PROMPT repeats the process for all the existing paths that start and end at the anchors and cumulatively aggregates the similarity scores.

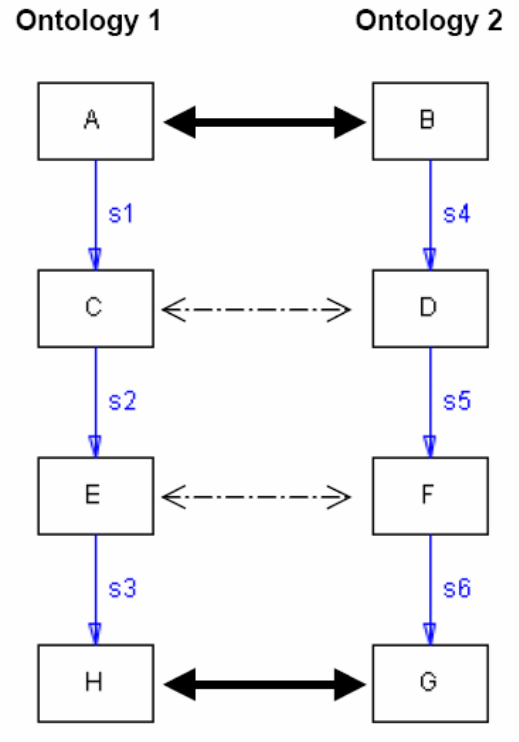



**Figure 3.5.** Traversing paths between anchors in Anchor-PROMPT [1]

The evaluation shows that 75% of the results are correct when using Anchor-PROMPT with ontologies developed independently by different groups of researchers. There are two limitations of the Anchor-PROMPT algorithm. One is that Anchor-PROMPT is time-consuming. Its worst case run-time behavior is $O(n^2 \log^2(n))$, compared to PROMPT $O(n\log(n))$, GLUE of $O(n^2)$ and QOM $O(n\log(n))$. Another limitation is Anchor-PROMPT does not work well when one ontology has a deep hierarchy with many classes inter-linked and the other ontology has a shallow hierarchy with few levels.

2. Similarity Flooding

[1] from (Noy and Musen 2001), p.64

Similarity Flooding (Melnik, Garcia-Molina *et al*. 2002) is a generic graph matching algorithm using fixpoint computation to determine corresponding nodes in the graphs. The principle of the similarity flooding algorithm is that the similarity between two nodes depends on the similarity between their adjacent nodes. In another words, a part of the similarity of two elements will propagate to their respective neighbors. The spread of similarities is similar to how IP packets flood a network in broadcast communication. This is why the algorithm is called similarity flooding.

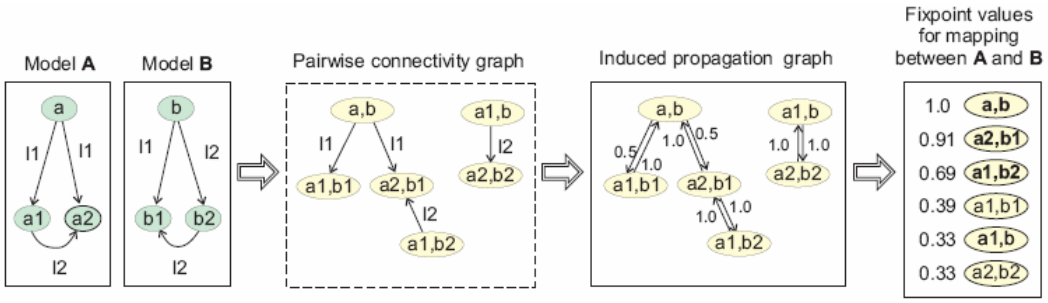



**Figure 3.6.** Example illustrating the Similarity Flooding algorithm [1]

Figure 3.6 illustrates the Similarity Flooding algorithm, in which two ontologies, A and B, are first translated into directed labeled graphs based on the Open Information Model specification[2]. To implement this, the algorithm creates another graph whose nodes, i.e., (a, b) and (a1, b1), are pairs of nodes of the initial two graphs, and there is an edge *l1* between (a, b) and (a1, b1) whenever there are edges (a, l1, a1) in the first graph and (b, l1, b1) in the second one. The algorithm computes initial similarity values between nodes based on their labels and iterates, re-computing the similarities between nodes as a function of the similarity between the

[1] from (Melnik, Garcia-Molina *et al.* 2002), p.120

[2] http://www.mdcinfo.com/OIM/OIM10.html

adjacent nodes at the previous step. It stops when no similarity changes more than a particular threshold or after a predetermined number of steps. The aggregation function is a weighted linear aggregation where the weight of an edge is the inverse of the number of other edges with the same label reaching the same entities. The values are further normalized with regard to the maximal similarity value obtained.

The authors adopt a novel quality metric to evaluate the performance of their Similarity Flooding algorithm. Unlike previously proposed metrics for measuring the matching accuracy without considering the extra work caused by wrong match proposals, they argue that user's effort for modifying a proposed mapping result can be measured in terms of adding and deleting mapping pairs. Thus, the function of the accuracy used in their evaluation is:

$$\text{Accuracy} = \text{Recall} \times (2 - \frac{1}{\text{Precision}}) \qquad \textbf{(1)}$$

This definition shows that the notion of the accuracy only makes sense when precision is not less than 0.5. If more than half of the mappings are wrong, it would take the user more effort to remove the false positives and add the missing mappings than to do the mapping manually from a scratch. Their evaluation shows that their mapping accuracy over 7 users and 9 problems averaged 52%.

Unlike most ontology mapping approaches which are not generic and tailored to a specific application domain such as data or schema integration and specific relational or XML schemas, the Similarity Flooding algorithm is a generic graph matching algorithm. Though the Similarity Flooding algorithm can be applied to 1-to-n mapping, it obtains this feature by decreasing the threshold of similarity. That is not a real 1-to-n mapping.

Some limitations of the Similarity Flooding algorithm are:

1) The algorithm only works for directed labeled graphs. When labeling is uniform or undirected, or when nodes are less distinguishable, the algorithm degrades.
2) The algorithm is based on the assumption that adjacency contributes to similarity propagation. Thus, when adjacency information is not preserved the algorithm will perform unexpectedly. For example, in some HTML pages, there are nodes displayed visually close are structurally far away from each other. In another case two cells in a HTML table vertically adjacent may be far apart in the document and can not contribute to similarity propagation.
3) The algorithm can only be applied to equal-typed models. For instance, it works when mapping an XML schema against another XML schema instead of mapping a relational schema against an XML schema.

### 3.2.5 Probabilistic Methods

Probabilistic methods are usually used on instance level in the process of ontology mapping. For example, OMEN (Ontology Mapping Enhancer) (Mitra, Noy *et al*. 2004), is a tool for describing mappings via probabilities and infers new mappings by means of Bayesian Network inference mechanisms. The motivation for using Bayesian Networks is: 1) the mappings are always imprecise especially when they are discovered automatically by heuristics or machine learning techniques; 2) sometimes even experts are not sure about the exact match between elements of different ontologies and typically assign some certainty rating to a match.

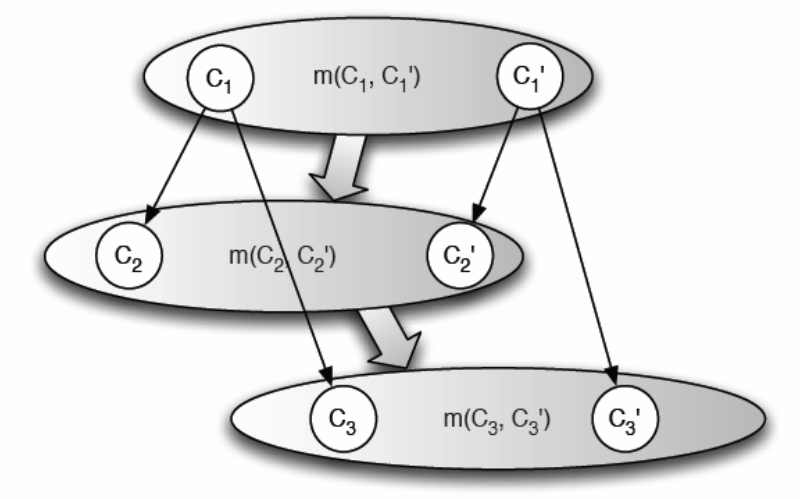



**Figure 3.7.** The Bayesian network constructed in OMEN [1]

Figure 3.7 shows a Bayesian Network (BN) constructed in OMEN, where some classes (the small circles) in ontology *O* are represented in the left-hand tree and some classes in ontology *O'* are represented in the right-hand tree. The thin arrows in the figure represent relations between the classes, such as subclass and superclass relationships. The large gray ovals with nodes represent individual pairs of matches and the solid arrows represent influences between the nodes in BN graph.

In order to run a Bayesian Network, two types of information are needed: 1) evidence, which is obtained from the initial probabilities and describes what is known with high confidence, and 2) conditional probability tables (CPTs), which represent how a probability distribution in one node affects the probability distribution in another node downstream from it. For instance, in Figure 3.7, the mapping between concept $C_1$ and $C'_1$ affects the mapping between concepts $C_2$ and $C'_2$, which in turn affects the mapping between $C_3$ and $C'_3$.

[1] from (Mitra, Noy *et al*. 2004), p.539

Given source ontologies *O* and *O'* and initial probability distribution for matches, the OMEN algorithm is:

1) If initial probability of a match is above a given threshold, create a node representing the match and mark it as evidence node.
2) Create nodes in the BN graph representing each pair of concepts (C, C'), such that $C \in O$ and $C' \in O'$ as a node in the graph and the nodes are within a distance k of an evidence node.
3) Create edges between the added nodes.
4) Generate the CPTs by means of a set of generic meta-rules, such as "Say there are two concepts C and C' that match and there is a relationship between C and another concept $C_1$ in the ontology O and a relationship between C' and $C'_1$ in the ontology O'. Furthermore these two relationships match/do not match. Then, the probability of the match between $C_1$ and $C'_1$ is increased/decrease."
5) Afterwards the inference on the Bayesian Network can be started and the output is a new set of matches.

In the OMEN system, probabilistic influences are combined for the child with the assumption that if a node in a Bayesian Network has two parents, the two parents are independent, that is $P(N|P_1, P_2) = P(N|P_1)P(N|P_2)$. This assumption is not true when the match of two pairs of parents influences each other. Even with this simple assumption the system obtains encouraging results that the inference of new mappings do work. For instance, the experiments on two ontologies about university departments and students, staff and faculty of the departments show that OMEN can generate up to 7 missing matches given only 3 out of 11 matches.

Compared with most ontology mapping techniques (i.e., heuristic, machine-learning, and graph methods) that are mainly based on linguistic analysis of concept names and natural-language definitions of concepts, OMEN uses a probabilistic method to improve the matches produced by the methods above or to suggest additional matches. Thus it is a complementary to current techniques of automatic or semi-automatic ontology mapping.

### 3.2.6 Reasoning and Theorem Proving Methods

- S-Match

S-Match (Giunchiglia, Shvaiko *et al*. 2004) is a schema and ontology mapping system that uses reasoning and theorem proving methods to find mappings. It starts with a combination of matchers using lexical information and external resources. Then it uses a SAT[1] solver to find semantic relations, such as equivalence (=), more general ($\supseteq$), less general ($\subseteq$), mismatch ($\neq$), union ($\cup$) and overlapping ($\cap$).

In the S-Match platform, the module that takes input schemas in a standard internal XML format does the preprocessing and returns enriched trees containing concepts of labels and nodes as output. These enriched trees are stored in an internal database, namely PTrees, where they can be browsed, edited and manipulated. The preprocessing module has access to the set of oracles, which provide the necessary a priori lexical and domain knowledge. Currently the only oracle S-Match has is WordNet. The Match Manager coordinates matching process using three extensible libraries, the Weak Semantic Matchers, the Oracles, and the SAT solvers. The Weak Semantic

[1] The Boolean satisfiability problem (SAT) is a decision problem, whose instance is a Boolean expression written using only AND, OR, NOT, variables, and parentheses. (From http://en.wikipedia.org/wiki/Boolean_satisfiability_problem)

Matchers are “element level weak semantics matchers”, which perform string manipulations (i.e., prefix, edit distance, n-grams analysis, data types, and so on) and try to guess the semantic relations implicitly encoded in similar words. The Oracles are “element level strong semantics matchers”. These matchers extract semantic relations existing between concept labels using oracles which memorize the necessary lexical and domain knowledge. The SAT solvers are structure level strong semantics matchers, which decide propositional satisfiability (a formula is valid if and only if its negation is unsatisfiable). The SAT decider that S-Match is currently using is JSAT[1].

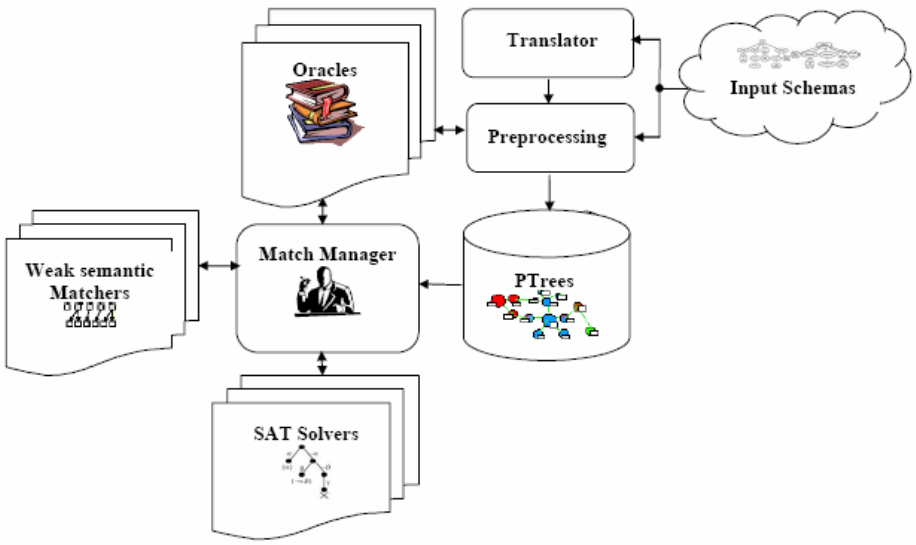



**Figure 3.8.** The architecture of the S-Match [2]

The evaluation of S-Match is based on a comparison with three other mapping systems -- Cupid (Madhavan, Bernstein *et al*. 2001), COMA (Do and Rahm 2002), and Similarity Flooding (Melnik, Garcia-Molina *et al*. 2002), all of which are schema-based only and utilize both

[1] http://cafe.newcastle.edu.au/daniel/JSAT/

[2] from (Giunchiglia, Shvaiko *et al*. 2004), p.72

linguistic and graph matching techniques. Though S-Match clearly outperforms other systems from the viewpoint of the quality of mapping results, S-Match runs the slowest because it needs to translate a mapping problem into a validity problem, which is very time consuming.

## 3.3 THE LATEST RESEARCH ON ONTOLOGY MAPPING

So far we have reviewed the representative approaches that have been proposed to solve the ontology mapping problem. Recently, some approaches that integrate the advantages of the state of the art techniques have appeared. In this section we review the 4 top-ranked systems that participated in OAEI campaign 2007, i.e., Falcon-AO (Qu, Hu *et al*. 2006; Hu, Zhao *et al*. 2007), RiMOM (Tang, Li *et al*. 2006; *Li, Zhong* et al. 2007), LILY(Wang and Xu 2007) , ASMOV(Jean-Mary and Kabuka 2007), and compare the major differences between them and PRIOR+.

The reasons for reviewing the four systems are:

1. The techniques that the 4 systems used in their approach are diverse and based on the state-of-art approaches. In reviewing these systems, we are reviewing the latest developments in this area.

2. Like ours, all the systems use multiple similarities, and thus face the problem of aggregating similarities in an effective way. Therefore, reviewing the approaches helps to evaluate our approach.

3. All these systems participated in OAEI ontology matching campaign 2007. The OAEI campaign provides uniform test cases so that quantitative comparison between different approaches is practical. Please see the detailed quantitative comparison in §4.7.3.7.

### 3.3.1 Falcon-AO

Falcon-AO (Qu, Hu *et al*. 2006; Hu, Zhao *et al*. 2007) is a similarity-based generic ontology mapping system. Figure 3.9 is the system architecture of Falcon-AO, which shows its five components: the *Repository* temporarily stores the data during the matching process; the *Model Pool* manipulates ontologies and constructs different models for different matchers; the *Alignment Set* generates and evaluates exported alignments; the *Matcher Library* manages a set of elementary matchers; and the *Central Controller* configures matching strategies and executes matching operations.

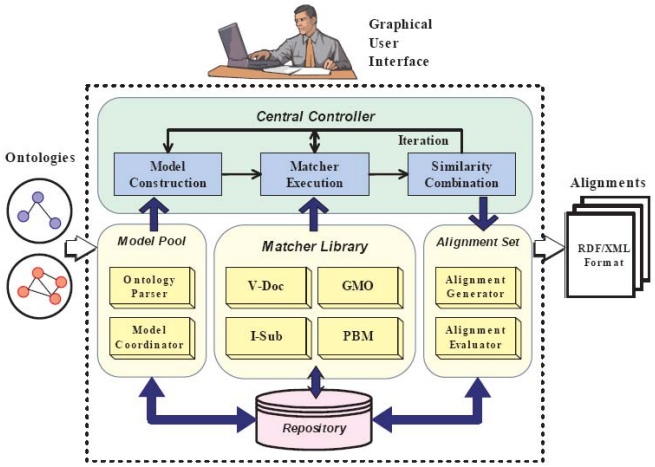



**Figure 3.9** The system architecture of Falcon-AO[1]

[1] From Hu, Zhao *et al*. 2007, p2

The *Matcher Library* is the core component of Falcon-AO. It consists of three elementary matchers, i.e., V-Doc, I-Sub (Stoilos, Stamou *et al*. 2005), and GMO, and one ontology partitioner, PBM. V-Doc constructs a virtual document for each URIref, and then measures their similarity in a vector space model. I-Sub compares the similarity of strings by considering their similarity along with their differences. GMO explores structural similarity based on a bipartite graph. PBM partitions large ontologies into small clusters, and then matches between and within clusters.

The *Model Coordinator* eliminates useless axioms and reduces structural heterogeneity between to-be-matched ontologies based on some pre-defined coordination rules.

The main differences between Falcon-AO and PRIOR+ are:

1. The *profile* used in our approach is similar as the *virtual document* constructed in Falcon-AO. The difference is the virtual document only exploits neighboring information based on an RDF model; whereas our profile does not have any limitation of information type, and thus can integrate any information including instances.
2. From the aggregation view, though Falcon-AO measures both linguistic comparability and structural comparability of ontologies to estimate the reliability of matched entity pairs, it only uses them to form three heuristic rules to integrate results generated by GMO and LMO. For example, Falcon-AO takes the results of GMO into account only when the structural comparability is greater than a predefined threshold. Using LMO, Falcon-AO linearly combines two linguistic similarities with some experiential number. Unfortunately neither experiential number nor heuristic rules can automatically adapt to different test cases, as we argued in §4.4.

Furthermore, when estimating linguistic comparability Falcon-AO does not distinguish the difference between class and property; whereas our approach estimates harmony for class and property separately.

3. Finally Falcon-AO does not have solutions to optimize final results so that they can satisfy various ontology constraints.

### 3.3.2 RiMOM

RiMOM (Tang, Li *et al*. 2006; Li, Zhong *et al*. 2007) is a general ontology mapping system based on Bayesian decision theory. It utilizes normalization and NLP techniques and integrates multiple strategies for ontology mapping. RiMOM uses risk minimization to search for optimal mappings from the results of multiple strategies. Figure 3.10 is the system architecture of RiMOM, which includes five major steps in its general alignment process.

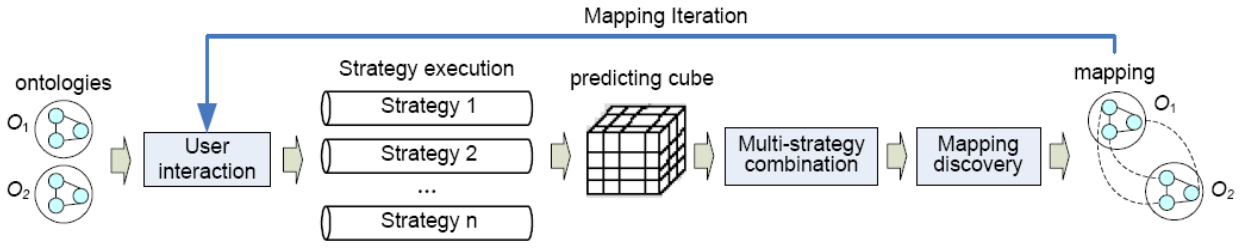



**Figure 3.10** The system architecture of RiMOM[1]

The details of each step are:

[1] From Tang, Li *et al*. 2006, p8

1. User Interaction (optional). RiMOM accepts rectified mapping or newly created mapping from users to improve the mapping accuracy.

2. Multi-strategy execution. Given two ontologies, RiMOM first estimates two similarity factors, i.e., structure similarity factor and label similarity factor. Based on the two factors, RiMOM executes multiple independent mapping strategies. Every strategy determines a prediction value between 0 and 1 for each possible candidate mapping. The output of the mapping execution phase with $k$ strategies, $m$ entities in $O_1$ and $n$ entities in $O_2$ is a $k \times m \times n$ cube of predicting values, which is stored for later strategy combination.

3. Strategy combination. RiMOM combines the alignment results obtained by different strategies using a linear-interpolation method.

4. Mapping discovery. RiMOM removes "unreliable" alignments using pre-defined heuristic rules. The output of its mapping discovery is a mapping table including multiple entries, each of which corresponds to a mapping.

5. Iteration. RiMOM iteratively executes step 1 to 4 till no new mappings are discovered.

The main differences between RiMOM and PRIOR+ are:

1. Both RiMOM and our approach do propagation based on propagation theory (Felzenszwalb and Huttenlocher 2006). However RiMOM propagates the similarity of two entities to entity pairs associated with some kinds of relationship (e.g. superClassOf, siblingClassOf, domain etc.); whereas PRIOR+ propagates original

information of an element instead of its similarity to its neighboring elements, and then compares their similarity based on the propagated profiles.

2. When integrating multiple strategies RiMOM adopts a sigmoid function with tentatively set parameters, which has been been shown to be inferior to harmony-based adaptive similarity aggregation in the PRIOR+. See experimental results in §4.7.3.4.

3. Furthermore, though RiMOM calculates two similarity factors to estimate the characteristics of ontologies, their estimation is suitable to some special situations only. For example, their linguistic similarity factor only concerns elements that have the same label. The idea of harmony in PRIOR+ is more general.

4. While both explore different approaches to find the optimal mappings for final results extraction. RiMOM uses risk minimization approach; while PRIOR+ tries neural network approach.

### 3.3.3 LILY

LILY (Wang and Xu 2007) is a generic ontology mapping system based on the extraction of semantic subgraphs. It exploits both linguistic and structural information in semantic subgraphs to generate initial alignments. Then a subsequent similarity propagation strategy is applied to produce more alignments if necessary. Finally LILY uses classic image threshold selection algorithm to automatically select threshold, and extract final results based on the stable marriage strategy. Figure 3.11 shows the system architecture of LILY.

The main differences between LILY and PRIOR+ are:

1. The efficiency problem. LILY faces two limitations when applying semantic subgraphs to solve the ontology mapping problem. One is it needs to manually set the size of subgraph according to different mapping tasks. The other is the efficiency of semantic subgraph mapping is very low in large-scale ontologies. For example, LILY used 4 days to get final mapping results between two anatomical ontologies in OAEI campaign 2007 while PRIOR+ needs only 23 min to find the mappings between the two anatomical ontologies.

2. LILY combines different similarities using a linear method with experiential weights, which is not good as harmony-based adaptive aggregation method in our PRIOR+.

3. LILY does not consider ontology constraints directly. Whereas the PRIOR+ integrate the IAC neural network to deal with constraints satisfaction in the context of ontology mapping.

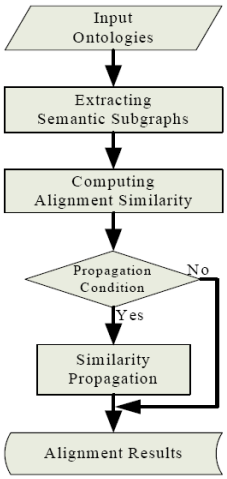



**Figure 3.11** The system architecture of LILY[1]

[1] From Wang and Xu 2007, p2

### 3.3.4 ASMOV

ASMOV (Jean-Mary and Kabuka 2007) is an automated ontology mapping tool that iteratively calculates the similarity between concepts in ontologies by analyzing four features, i.e., textual description (id, label, and comment), external structure (parents and children), internal structure (property restrictions for classes; types, domains, and ranges for properties), and individual similarity. It then combines the measures of these four features using a weighted sum. The weights are adjusted based on static rules. At the end of each iteration, a pruning process eliminates the invalid mappings by analyzing two semantic inconsistencies: crisscross mappings and many-to-one mappings. Figure 3.12 shows the system architecture of ASMOV.

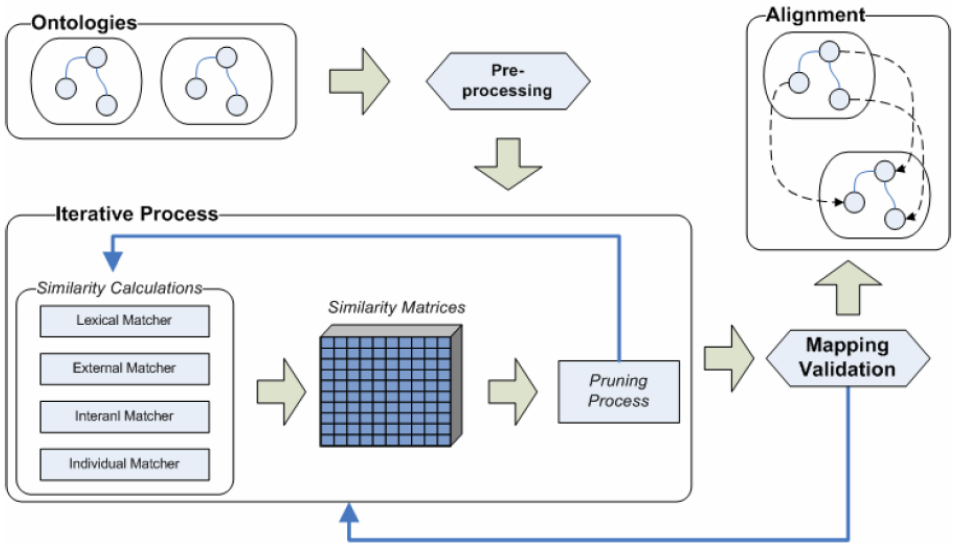



**Figure 3.12** The system architecture of ASMOV[1]

Due to the limited literature available we are unable to compare our approach with ASMOV in detail. What we can say is:

[1] From Jean-Mary and Kabuka 2007, p2

1. ASMOV integrates UMLS Metathesaurus[1] as a domain specific thesaurus for the anatomy test and WordNet[2] as a general thesaurus for the other tests including the benchmark tests in OAEI campaign. PRIOR+ does not rely on any external resources in the process of ontology mapping.
2. The aggregation method in ASMOV is heuristic rule based *weighted* aggregation and only two constraints are validated for their final results. PRIOR+ uses harmony-based aggregation method to adaptively combine different similarities.

[1] http://www.nlm.nih.gov/research/umls/

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

# 4.0 THE *PRIOR+* APPROACH

## 4.1 INTRODUCTION

Approaches to ontology mapping include using linguistic techniques to measure the lexical similarity of concepts in ontologies (Qu, Hu *et al*. 2006), treating ontologies as structural graphs (Noy and Musen 2001; Melnik, Garcia-Molina *et al*. 2002) applying heuristic rules to look for specific mapping patterns (Mitra, Wiederhold *et al*. 1999), using machine learning to map ontologies (Doan, Madhaven *et al*. 2003), and integrating various techniques (Hu, Zhao *et al*. 2007; Li, Zhong *et al*. 2007; Mao and Peng 2007; Wang and Xu 2007; Jean-Mary and Kabuka 2007). More comprehensive surveys of ontology mapping approaches can be found in (Euzenat, Bach *et al*. 2004; Kalfoglou and Schorlemmer 2003; Noy 2004).

Though the more recent approaches have made significant progresses in ontology mapping, they suffer from two limitations. First, ontology mapping approaches that use multiple mapping strategies face the problem of aggregating multiple similarities. Currently, they either use some predefined experience numbers to weight different similarities or tentatively set parameters in aggregation functions (e.g. sigmoid). Manually setting parameters is impractical due to its inability to adapting to different ontology mapping tasks. A second limitation is that most ontology mapping approaches do not thoroughly deal with ontology constraints (e.g., the

hierarchical relations in RDFS [1], the axioms in OWL [2], and the rules in SWRL [3]). Most approaches either ignore ontology constraints completely or consider ontology constraints from a local perspective based on some heuristic rules. Exceptions include GLUE (Doan, Madhaven *et al*. 2003), which adopts relaxation labeling to optimize mapping configurations by considering ontology constraints, and RiMOM (Tang, Li *et al*. 2006), which uses risk minimization to search for the optimal mappings from the results output by multiple strategies.

To overcome the limitations, this chapter introduces a new generic ontology mapping approach, called the PRIOR+ (Mao and Peng 2006; Mao, Peng *et al*. 2006; Mao and Peng 2007).

## 4.2 OVERVIEW OF THE PRIOR+ APPROACH

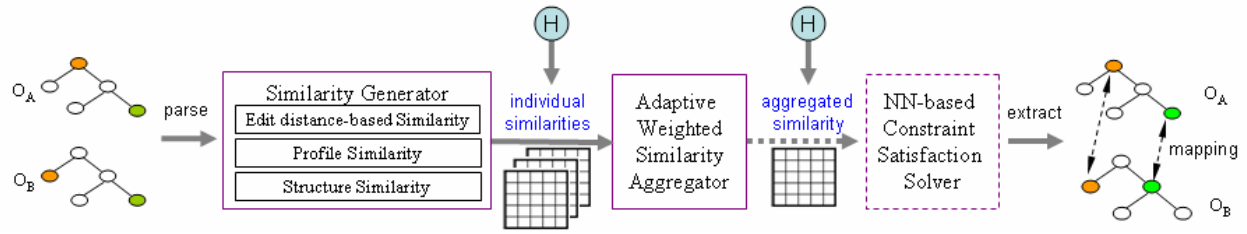



**Figure 4.1** The architecture of the PRIOR+ approach

Figure 4.1 depicts the architecture of the PRIOR+. The input to PRIOR+ is two ontologies expressed in a formal ontology language such as OWL[4] or SKOS[5]. First the ontologies will be

[1] http://www.w3.org/TR/rdf-schema/

[2] http://www.w3.org/TR/owl-features/

[3] http://www.daml.org/2003/11/swrl/

[4] http://www.w3.org/TR/owl-features/

[5] http://www.w3.org/TR/2005/WD-swbp-skos-core-guide-20050510/

parsed by Jena[1] and pre-processed by removing stop words, stemming, and tokenizing. After that, the approach measures both the linguistic and the structural similarities of the ontologies. More specifically, three kinds of similarity, i.e., edit distance based similarity, profile similarity and structural similarity, are calculated. For each similarity, a measurement of harmony is estimated. Based on the estimation, three similarities are adaptively aggregated to get the final similarity between elements in the ontologies. When the harmony of the aggregated similarity is not good, the interactive activation and competition (IAC) neural network is activated to improve mapping accuracy (as illustrated with dashed line and box in the figure). The purpose of the IAC neural network is to search for a globally optimized solution that best satisfies as many ontology constraints as possible. To extract final mapping results, a naïve descendant extraction algorithm is applied (Meilicke and Stuckenschmidt 2007).

Here we briefly describe three major components in the PRIOR+. The details of each component are as follows.

1. Similarity Generator

The most intuitive and efficient way to compare the similarity of two elements is to calculate the edit distance between them. The larger the distance, the less similarity they own. However, edit distance based similarity does not work when two concepts are semantically similar but lexically different (e.g. synonyms). Therefore, besides edit distance based similarity, we propose to build a *profile* for each element, and then compare the cosine similarity of two profiles in a Vector Space Model (VSM). To build the profile, we integrate all descriptive information of an element, such as its id, label, comments and property restrictions. We explore the structural similarity between two elements in ontologies as well. In our approach the

[1] http://jena.sourceforge.net/

structural similarity are represented by some structural features of an element such as the depth of the element to the root, the number of its subclasses or subproperties, etc.

2. Adaptive Similarity Aggregator

With three kinds of similarities, combining them to get an integrated score that can be used to represent the similarity between two elements is a difficult problem. Currently most researchers integrate different similarities using the weighted sum method. They either rely on their experience or tentatively set a number when assigning weights to different similarities. A weighted sum method has the limitation that it can not adjust weights automatically to different situations. We propose a measurement of harmony to adaptively aggregate different similarities. The harmony is the ratio of "perfectly" matched mappings to the whole candidates. In a similarity matrix, the ratio equals the number of cell that holds the highest similarity score in both row and column. An example of harmony calculation is provided in §4.4.2.2.

3. The IAC Neural Network based Constraint Satisfaction Solver

Having the aggregated similarity, we can directly extract final mapping results using some naïve method (Meilicke and Stuckenschmidt 2007) or work assignment optimization method such as the Hungarian algorithm (Kuhn 1955). However in the context of ontology mapping, there are many constraints. For example, in the ontology taxonomy, if element $e_1$ in $O_1$ map to element $e_2$ in $O_2$, then the parent of $e_1$ should not map to child of $e_2$. The crisscross mapping results in conflicts. To improve the mapping accuracy, we integrate the interactive activation and competition (IAC) neural network to search for an optimal solution that can satisfy as many constraints as possible in the context of ontology mapping.

The IAC neural network consists of a set of nodes. Each node represents a hypothesis. Between hypotheses, there are two kinds of connections. If a hypothesis supports another

hypothesis, there is a positive connection between them. Otherwise, there is a negative connection. In the context of ontology mapping, a node represents a hypothesis that element $e_1$ in ontology $O_1$ maps to element $e_2$ in ontology $O_2$. The connections between nodes come from different kinds of constraints in ontology mapping. Each node has three kinds of input, its initial activation, the input from its neighboring nodes, and the external input. Once the network starts running, it will update the activation of each node using a set of rules. Finally it will stop at some globally optimized point, where as many constraints as possible will be satisfied.

## 4.3 SIMILARITY GENERATOR

The similarity generator generates three kinds of similarities, i.e. edit distance based similarity, profile similarity and structural similarity. The input of the similarity generator is two ontologies, which will be parsed by Jena[1]. Each element will be pre-processed by removing stop words, stemming, and tokenizing. The outputs are three similarity matrixes that contain similarity scores for each pair of elements in the ontologies.

### 4.3.1 Edit Distance Based Similarity

The edit distance based similarity is calculated between the *name* (i.e. *ID*) of elements based on their Levenshtein distance. The similarity is defined by Equation 2, where $EditDist(e_{1i}, e_{2j})$ is Levenshtein distance between elements $e_{1i}$ and $e_{2j}$, $l(e_{1i})$ and $l(e_{2j})$ are the string length of the name of $e_{1i}$ and $e_{2j}$ respectively.

[1] http://jena.sourceforge.net/

$$NameSim(e_{1i}, e_{2j}) = 1 - \frac{EditDist(e_{1i}, e_{2j})}{\max(l(e_{1i}), l(e_{2j}))} \qquad (2)$$

### 4.3.2 Profile Similarity

The profile similarity is generated in three steps, i.e., Profile Enrichment, Profile Propagation, and Profile Mapping. Figure 4.2 shows the major processes in profile similarity generation.

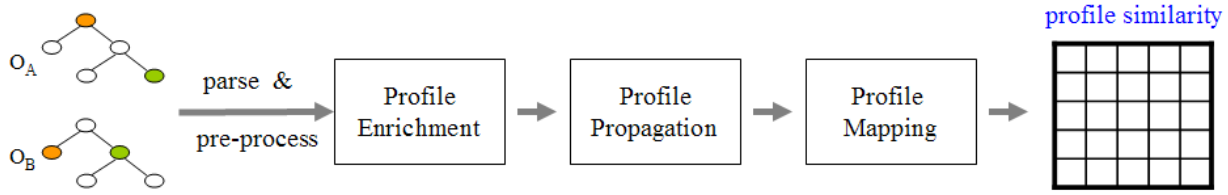



**Figure 4.2** The major processes in Profile Similarity generation

1. Profile Enrichment. For each element in the ontology, we generate a profile, i.e., a combination of the element's descriptive information, to represent it and thus enrich its information. Then the tf•idf weight is assigned for each profile based on the whole collection of all profiles in the ontology.
2. Profile Propagation. To exploit the neighboring information of each element, the profile of the element's ancestors, descendants and siblings will be passed to that of the element with different weights.
3. Profile Mapping. Finally the cosine similarity between the profiles of two elements of $e_{1i}$ and $e_{2j}$ is calculated using a vector space model.

#### 4.3.2.1 Profile Enrichment

First we introduce the term *profile*. The *profile* of an element is a combination of its linguistic description, after removing stop words, stemming and tokenizing, and keeping all duplicates. In particular, we define the profile of each class, property and instance as:

*the profile of a* ***class*** = the class's ID + label + comments + other restriction + its properties' profiles + its instances' profiles

*the profile of a* ***property*** = the property's ID + label + its domain + its range

*the profile of an instance* = the instance's ID + label + other descriptive information

For example, given the descriptive information in Figure 2.2, the profile of class *Book*, *Proceedings* and *Monograph* and property *editor* in the left ontology are:

*Profile*($Book_{left}$) = (*book, book, book, monograph, collection, write, text*)

*Profile*(*Proceedings*) = (*proceeding, communication, event, editor, organization*)

*Profile*(*Monograph*) = (*monograph, object, orient, data, model, publish, MIT, year*)

*Profile*(*editor*) = (*editor, proceeding, person*[1])

The motivation for a *profile* is based on the observation that though a ID or name is always used to represent an element, sometimes the information carried in the ID or name is restricted, especially in cases where the ID or name is identified using meaningless symbols. Meanwhile other descriptive information such as labels and comments may contain words that better convey the meaning of the concept. The goal of Profile Enrichment is to use the profile to represent an element in an ontology, and thus enrich its information.

Having profiles for each element, the *tf·idf* (term frequency–inverse document frequency) weight will be used to assign larger weight to the terms that have a high frequency in a given document and a low frequency in the whole collection of documents. The *tf·idf* weight is defined as:

$$w = tf \bullet idf = \frac{n_i}{\sum_k n_k} \times \log_2 \frac{N}{n} \qquad \textbf{(3)}$$

[1] where the *person* is the range of the *editor*, which is not indicated in the Figure 2.2

where, $n_i$ is the number of occurrences of the considered term, $\sum n_k$ is the number of occurrences of all terms, $N$ is the total number of documents in the collection, and $n$ is the number of documents where the term $t_i$ appears at least once (i.e., $n_i \neq 0$). In our case, each *profile* is treated as a *document* and all profiles in two ontologies are treated as the collection of documents, which means $N$ equals the number of total profiles in two ontologies.

The output of Profile Enrichment is a set of vectors, in which each vector represents a concept using its *tf·idf* weights corresponding to each term.

#### 4.3.2.2 Profile Propagation

Profile Propagation exploits the neighboring information of each element. That is, the profile of the element's ancestors, descendants and siblings will be passed to the profile of the element itself. The motivation of profile propagation is based on the observation that if the taxonomy of an ontology can be seen as the index of a book, the super class in the ontology reflects the "context" of its subclasses and each subclass is a specific "content" of its super class. This means super classes always carry some characteristics of its subclasses and vice versa.

The profile can be propagated in different ways. Figure 4.3 shows an example where the propagation level is up to 2 and the weights are set as $w_{itself \rightarrow itself}$, $w_{parent \rightarrow itself}$, $w_{grandparent \rightarrow itself}$, $w_{children \rightarrow itself}$, $w_{grandchildren \rightarrow itself}$, $w_{sibling \rightarrow itself}$. How to find the best assembly of different weights to the neighbors of an element is not easy. As a beginning point, this research suggests that ancestors exert more influence than children and children more influence than siblings. However, there is no firm rule to follow. In §4.7.3.2 we describe an experiment that investigated the difference between the performance of profile similarity when assigning different weights for neighboring elements.

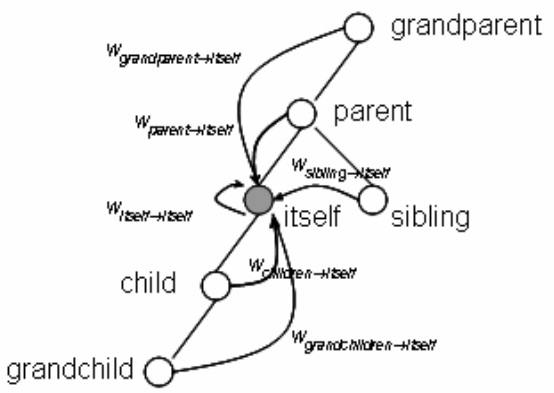



**Figure 4.3.** Profile propagation

The process of Profile Propagation can be represented as:

$$V_{e_{1i}-new} = \sum_{e_{2j} \in E} w(e_{1i}, e_{2j}) V_{e_{2j}} \qquad (4)$$

where $e_{1i}$ and $e_{2j}$ represent two elements in the ontologies, $E$ represents the set of all elements in the ontologies, $V_{e1i\text{-}new}$ represents the new profile vector of the element $e_{1i}$, $V_{e2j}$ represents the original profile vector of the element $e_{2j}$, and $w(e_{1i},e_{2j})$ is the function that assigns different weights to the neighbors of the concept. For example, after the propagation, the new profile of the element *itself* in Figure 4.3 is:

$$V_{itself\text{-}new} = w_{itself \to itself} V_{itself} + w_{parent \to itself} V_{parent} + w_{grandparent \to itself} V_{grandparent} + w_{children \to itself} V_{children} + w_{grandchildre \to itself} V_{grandchildren} + w_{sibling \to itself} V_{sibling}$$

The output of Profile Propagation is a new set of vectors, in which each vector is updated from its original vector by integrating its neighboring information using Equation 4.

#### 4.3.2.3 Profile Mapping

Given a query and a set of documents, classical information retrieval measures the similarity between a query and a set of documents, and returns the documents that have top-ranked

similarities. In the context of ontology mapping, if a profile in an ontology is treated as a query, and all profiles in the other ontology are treated as a collection of documents, finding the most similar elements in the two ontologies is just a matter of searching for the most relevant document from the collection using the query.

There are many ways to evaluate the similarity between two documents in a collection. A common method is to measure the cosine angle between the two vectors of the documents. The cosine similarity between the profiles of two elements of $e_{1i}$ and $e_{2j}$ is calculated in a vector space model using Equation 5:

$$ProfileSim(e_{1i}, e_{2j}) = \frac{\overrightarrow{V_{e_{1i}}} \cdot \overrightarrow{V_{e_{2j}}}}{|V_{e_{1i}}||V_{e_{2j}}|} = \frac{\sum_{k=1}^{n}(V_k^{e_{1i}} * V_k^{e_{2j}})}{\sqrt{\sum(V_k^{e_{1i}})^2}\sqrt{\sum(V_k^{e_{2j}})^2}} \quad (5)$$

where $V_{e1i}$ and $V_{e2j}$ are two vectors representing the profile of element $e_{1i}$ and $e_{2j}$ respectively, $n$ is the dimension of the profile vectors, $V_k^{e1i}$ and $V_k^{e2j}$ are $k_{\text{th}}$ element in the profile vector of element $e_{1i}$ and $e_{2j}$ respectively, $|V_{e1i}|$ and $|V_{e2j}|$ are the lengths of the two vectors respectively.

The output of Profile Mapping is a similarity matrix, each cell of which represents a similarity score between two elements.

### 4.3.3 Structural Similarity

The structural similarity between two elements comes from their structural features (e.g., the number of direct property of a class). Structural similarity is considered for classes only. No structural similarity will be given to property or instance because of the lack of hierarchical

information. The structural similarity of the classes in two ontologies is defined by Equation 6, where $e_{1i}$ and $e_{2j}$ are two class elements in ontology $O_1$ and $O_2$ respectively, $n$ is the total number of structure features, $diff_k(e_{1i},e_{2j})$ denotes the difference for feature $k$.

$$StructSim(e_{1i}, e_{2j}) = \frac{\sum_{k=1}^{n}(1 - diff_k(e_{1i}, e_{2j}))}{n} \qquad (6)$$

The $diff_k(e_{1i},e_{2j})$ is defined as Equation 7, where $sf(e_{1i})$ and $sf(e_{2j})$ denote the value of structure features of $e_{1i}$ and $e_{2j}$ respectively. In our approach $sf(e_{1i})$ and $sf(e_{2j})$ are one of the following values, the number of the class' direct properties, the number of the class' instances, the number of the class' children, or the normalized depth of the class from the root.

$$diff(e_{1i}, e_{2j}) = \frac{|sf(e_{1i}) - sf(e_{2j})|}{\max(sf(e_{1i}), sf(e_{2j}))} \qquad (7)$$

As an example of depth difference, assume the max depth of ontology $O_1$ is 5, the max depth of ontology $O_2$ is 6. $depth(e_{1i}) = 3$, $depth(e_{2j}) = 4$. Then The normalized depth of $e_{1i}$ and $e_{2j}$ are $sf(e_{1i}) = 3/5 = .6$, $sf(e_{1i}) = 4/6 = .67$. Finally the depth difference between $e_{1i}$ and $e_{2j}$ can be calculated as:

$$\mathrm{diff}(e_{1i}, e_{2j}) = \frac{|.6 - .67|}{\max(.6, .67)} = \frac{.07}{.67} = .10$$

## 4.4 ADAPTIVE SIMILARITY AGGREGATOR

### 4.4.1 Similarity Aggregation in the State-of-art Ontology Mapping Approaches

Aggregating different similarity is pervasive in ontology mapping systems that contain multiple individual matchers, e.g., COMA (Do 2006), Falcon-AO (Qu, Hu *et al*. 2006, Hu, Zhao *et al.* 2007), RiMOM (Tang, Li *et al*. 2006; Li, Zhong *et al.* 2007), and QOM (Ehrig and Staab 2004). Many strategies, e.g., *Max*, *Weighted*, *Average* and *Sigmoid*, have been proposed to aggregate different similarities in the approaches. The *Max* strategy returns the maximal similarity of individual matchers. The *Weighted* strategy determines a weighted sum of similarity of individual matchers. The *Average* strategy is a case of the *Weighted* strategy and returns the average similarity over all individual matchers. The *SIGMOID* strategy combines multiple results using a sigmoid function, a smoothed threshold function.

Among the strategies, the *Max* strategy selects one extreme end of various similarities to be the representative of the final similarity, which is too optimistic especially in case of contradicting similarities. The *Average* strategy considers the individual similarities equally important and can not distinguish differences between them. The *Weighted* strategy overcomes the drawbacks of the *Average* strategy by assigning relative weights to individual matchers. The *SIGMOID* strategy emphasizes high individual predicting values and deemphasizes low individual predicting values.

Currently the systems that adopt the *Weighted* strategy or the *SIGMOID* strategy to aggregate similarities need to manually set aggregation weights based on experience for different similarities or tentatively set center position and steepness factor in the sigmoid function.

However, manually predefined parameters can not be generalized to adapt to different mapping situations. Therefore how to select appropriate parameters that can truly reflect the reliability of different similarities deserves further research.

### 4.4.2 The Harmony

In this section we introduce the term harmony to estimate the reliability of different similarities. The purpose of harmony is to:

1. Provide a measurable number that can tell us which similarity is more reliable and trustful so that we can give it a higher weight during aggregation.
2. Assist in adaptively adjusting the mapping strategy. That is, when to activate or inactivate NN-based constraint satisfaction solver.

Five individual harmonies (i.e., class name harmony, class profile harmony, class structural harmony, property name harmony, and property profile harmony) and two final harmonies (i.e. class harmony, property harmony) will be estimated on the corresponding similarity matrix. The 5 individual harmonies will be used as weight to aggregate similarities. The 2 final harmonies will be used to decide if we need to activate the IAC neural network or not.

#### 4.4.2.1 The Definition of Harmony

Ideally, for 1-1 mapping, the similarity score of two truly mapped elements should be larger than that of all other pairs of elements that share the same row/column with the two elements in the similarity matrix, which implies that the two elements of this pair mutually prefer each other.

Given the rationale, we define the *harmony* of the similarity matrix as Equation 8, where *#s_max* denotes the number of the pair of elements that has the highest similarity in its corresponding row and column in the similarity matrix, and $\#e_i$ denotes the number of elements in ontology $O_i$.

$$h = \frac{\#s\_\max}{\min(\#e_1, \#e_2)} \qquad (8)$$

#### 4.4.2.2 A Simple Example of Harmony Estimation

Figure **4.4** is the name similarity matrix for the example illustrated in Figure 2.2. The left table lists the original similarity score between each pair of elements. The right table illustrates how the harmony is calculated, where "×" denotes the cell that has the highest similarity score in each row, "o" denotes the cell that has the highest similarity score in each column, and "⊗" denotes the overlapped cell that has the highest similarity in both the row and the column. Therefore the *#s_max* in Equation 8 is the number of "⊗" in the right table. In this case, *#s_max* is 4 and thus the normalized harmony of the name similarity matrix is *4/5* = *.8*. The range of the harmony is [0,1].

| | Composite | Book | Proc. | Monography | Collection |
|---|---|---|---|---|---|
| Reference | .11 | 0 | **.22** | 0.1 | .1 |
| Book | 0.22 | **1** | .2 | .2 | .2 |
| Proceedings | .18 | .09 | **.36** | .09 | .18 |
| Monograph | .11 | .22 | .11 | **.9** | .1 |
| Collection | **.3** | .2 | .1 | .1 | **1** |

*Harmony* = 4/5 = 0.8

| | | | | |
|---|---|---|---|---|
| | | × | | |
| | ⊗ | | | |
| | | ⊗ | | |
| | | | ⊗ | |
| o | | | | ⊗ |

**Figure 4.4** A sample of harmony calculation

### 4.4.3 The Harmony-based Adaptive Similarity Aggregation

In this section we propose a new weight assignment method to adaptively aggregate different similarities. That is, we use the *harmony* of different similarities as *weight* to aggregate various similarities. Therefore the final similarity of the pair of elements $(e_{1i},e_{2j})$ can be defined by Equation 9, where $k$ denotes to different similarities (i.e., For classes, they are class edit distance based similarity, class profile similarity and class structural similarity; For properties, they are property edit distance based similarity and property profile similarity), $h_k$ denotes the harmony of different similarities as defined in Equation 8, $n$ denotes the number of different types of similarity, and $Sim_k(e_{1i},e_{2j})$ denotes the similarity of each pair of elements.

$$FinalSim(e_{1i}, e_{2j}) = \frac{\sum_k h_k \times Sim_k(e_{1i}, e_{2j})}{n} \qquad (9)$$

The input of the adaptive similarity aggregator is a set of individual similarity matrixes as described in §4.3. The output is an aggregated similarity matrix, which we call *final similarity* matrix. The comparison of the harmony-based adaptive similarity aggregation and other aggregation methods is given in §4.7.3.4.

## 4.5 THE NEURAL NETWORK BASED CONSTRAINT SATISFACTION SOLVER

### 4.5.1 The Constraint Satisfaction Problem

A constraint satisfaction problem (CSP) is "*a problem composed of a finite set of **variables**, each of which is associated with a finite **domain**, and a set of **constraints** that restricts the values the*

*variables can simultaneously take. The task is to assign a value to each variable satisfying all the constraints.*" (Tsang 1993). Classic examples of CSPs include the map coloring problem, sudoku, eight queens puzzle, etc.

CSP is an intriguing research problem in ontology mapping due to the fact that the characteristics of an ontology and its representations result in many kinds of constraints. For example, the hierarchical relations in RDFS[1] do not allow crisscross mappings, the axioms such as *owl:sameAs* and *owl:equvalentClass* in OWL [2] indicate an equivalent relation between different elements, and the rules in SWRL[3] imply or assert some properties that are not directly available. Figure 4.5 is a real world case (i.e., the OAEI web directory test case #1) that shows finding an optimal configuration that can best satisfy ontology constraints is critical to improving the quality of ontology mapping. In the example, a crisscross mapping, i.e., "celebrities" maps to "celebrities" and "arts" maps to "artists", might be incorrectly output as final results due to non-validation of preliminary results from a global structural view. To avoid the problem, we need to find a configuration that satisfies the constraint of "if $m(e_{1i}, e_{2j},=,x)$is true, then $m(e'_{1i}, e'_{2j},=,y)$ is false, where $e'_{1i}$ is the parent of $e_{1i}$, and $e'_{2j}$ is the children of $e_{2j}$".

[1] http://www.w3.org/TR/rdf-schema/

[2] http://www.w3.org/TR/owl-features/

[3] http://www.w3.org/Submission/SWRL/

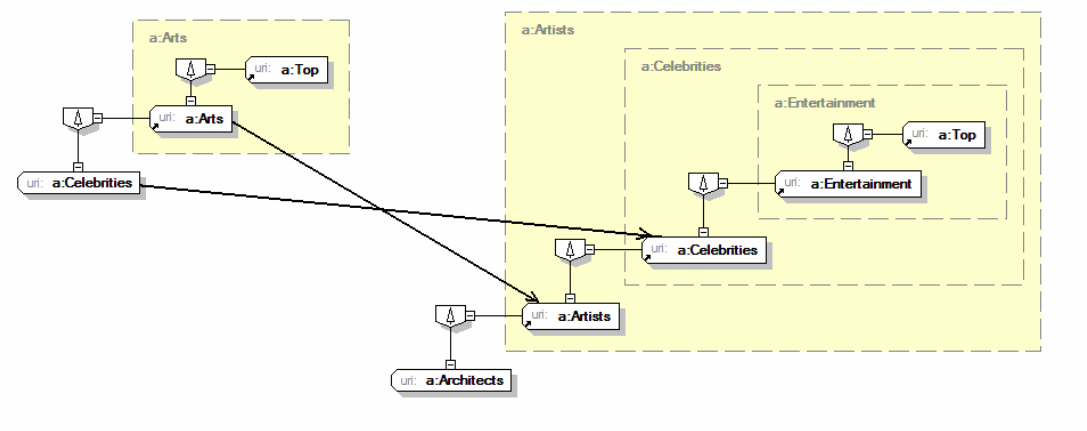



**Figure 4.5.** The crisscross mapping in OAEI web directory test case #1

### 4.5.2 The Interactive Activation and Competition (IAC) Neural Network

CSPs are typically solved by a form of search, e.g. backtracking, constraint propagation or local search (Tsang 1993). The IAC neural network, like the Hopfield network (Hopfield and Tank 1985), was proposed to solve CSPs by McClelland and Rumelhart in 1980's (McClelland and Rumelhart 1981; McClelland and Rumelhart 1988).

Generally, an IAC neural network consists of a number of competitive *nodes* connected to each other. Each node represents a *hypothesis*. The *connection* between two nodes represents constraint between their hypotheses. Each connection is associated with a *weight*. The activation of a node is determined locally by the nodes adjacent to it and the weights connecting it. Furthermore, if two hypotheses support each other, the connection between them is *positive* (i.e., *excitatory*); whereas if two hypotheses are against each other, the connection between them is *negative* (i.e., *inhibitory*). The weight of the connection is proportional to the strength of the constraint. The stronger a constraint is, the larger the corresponding weight.

The mechanism of the IAC neural network can be illustrated using the following simple example. Suppose we have two grids, 1 and 2, and two constraints:

1. Each grid can have one value, either A or B.
2. The values of the two grids are different.

We have four hypotheses: A in grid 1 ($H_{A1}$); B in grid 1 ($H_{B1}$); A in grid 2 ($H_{A2}$); B in grid 2 ($H_{B2}$). Based on the two constraints we know there are two negative connections and one positive connection for each hypothesis:

1. $H_{Ai}$ is against $H_{Bi}$, and vice versa ($i$=*1* or *2*)
2. $H_{x1}$ is against $H_{x2}$, and vice versa ($x$=*A* or *B*)
3. $H_{Ai}$ supports $H_{Bj}$, and vice versa ($i, j$ = *1* or *2*, and $i \neq j$)

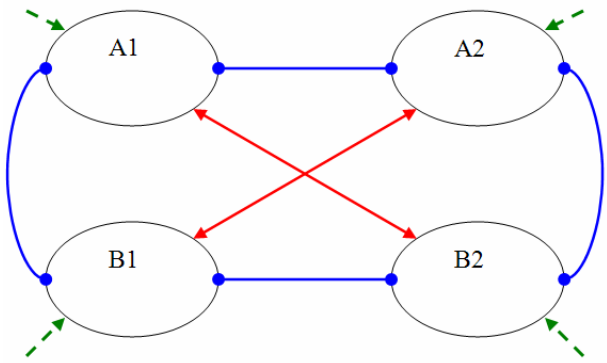



**Figure 4.6.** A simple IAC neural network

Figure 4.6 illustrates the simple example, where each node represents a hypothesis, the line with rounded head and arrowhead represents negative connection and positive connection between hypothesis respectively and the dashed line with arrowhead represents a small stimulus on each node from outside.

Assume the negative weight is half the positive weight and all nodes are inactive at start. Though the input from three neighbors of the node will cancel out, the small excitatory input from outside will activate the node. Finally, either *A1 and B2* or *B1 and A2* will be active, and the network will reach a *stable state,* called *settled* or *relaxed* solution.

The mechanism of the IAC network motivates us to apply it to solve the constraint satisfaction problem in ontology mapping. This is because:

1. The constraints in ontology mapping are either interactive (i.e., synergistic) or competitive between mapping hypotheses, which is the same as the constraints in the IAC networks. For example, for two mapping hypotheses, e.g., $m(e_{1i}, e_{2j},=,x)$ and $m(e'_{1i}, e'_{2j},=,y)$, the constraint "if $m(e_{1i}, e_{2j},=,x)$ is true, then $m(e'_{1i}, e'_{2j},=,y)$ is true, where $e'_{1i}$ and $e'_{2j}$ are children of $e_{1i}$, and $e_{2j}$ respectively" is interactive; whereas the constraint "if $m(e_{1i}, e_{2j},=,x)$ is true, then $m(e'_{1i}, e'_{2j},=,y)$ is false, where $e'_{1i}$ is the parent of $e_{1i}$, and $e'_{2j}$ is the children of $e_{2j}$" is competitive.
2. The preliminary analysis of ontologies on both linguistic and structure bring prior knowledge to us, for example, the confidence of mapping hypotheses, which is suitable to convert to the external input in the IAC network.
3. The IAC network is a parallel distributed processing (PDP) framework. It is easily implemented in a parallel processing platform. This is a valuable property especially when the number and complexity of constraints in ontology mapping increase.

In next section, we will introduce how the IAC neural network will be applied in the context of ontology mapping.

### 4.5.3 The IAC Neural Network in the Context of Ontology Mapping

In the context of ontology mapping, a node in the IAC neural network represents a hypothesis that indicates element $e_{1i}$ in ontology $O_1$ can be mapped to element $e_{2j}$ in ontology $O_2$. The connections between nodes in the network represent constraints between hypotheses.

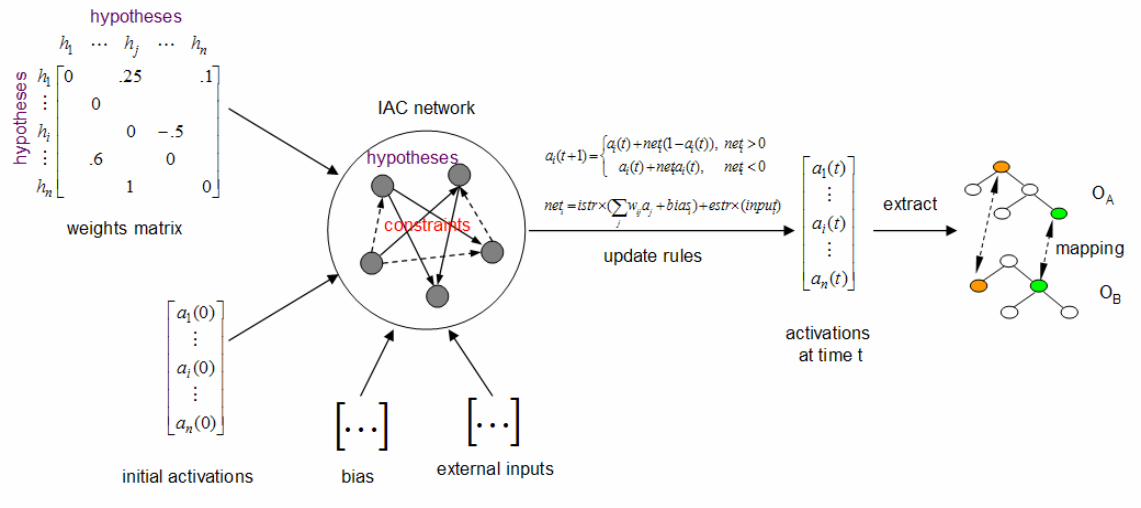



**Figure 4.7.** The IAC neural network in the context of ontology mapping

In Figure 4.7 we can see the input of the network includes the initial activation of each node (i.e., the priori probability of a hypothesis), its bias, external inputs and a weight matrix responding to the connections between different hypotheses. In our approach the initial activation of each node is set to the aggregated similarity of ($e_{1i}$, $e_{2j}$) output from the adaptive similarity aggregator. The activation of the node can be updated using the following simple rule, where $a_i$ denotes the activation of *node i*, written as $n_i$, $net_i$ denotes the net input of the node.

$$a_i(t+1) = \begin{cases} a_i(t) + net_i(1 - a_i(t)), & net_i > 0 \\ a_i(t) + net_i a_i(t), & net_i < 0 \end{cases}$$

The $net_i$ comes from three sources, i.e. its neighbors, its bias, and its external inputs. The $net_i$ is defined by Equation 10, where $w_{ij}$ denotes the connection weight between $n_i$ and $n_j$, $a_j$ denotes the activation of node $n_j$, $bias_i$ denotes the bias of $n_i$, and $ei_i$ denotes the external input of $n_i$, which is a function of the confidence of a mapping. Note that the weight matrix is symmetric and the nodes may not connect to themselves, i.e., $w_{ij}=w_{ji}$, $w_{ii}=0$.

$$net_i = \sum_j w_{ij} a_j + bias_i + ei_i \qquad (10)$$

The network can be stopped after running *n* cycles or at some *goodness* point. Forcing the network to stop at a predefined cycle number usually is not optimal. In the rest of the section, we introduce how we stop the network according to its goodness.

McClelland and Rumelhart (McClelland and Rumelhart 1988, *pp.50*) defined the *goodness* (short for *goodness of fit*) as the degree to which the desired constraints are satisfied. They pointed out that the goodness depends on three things. First, it depends on the extent to which each node satisfies the constraints imposed upon it by other nodes. Thus, if a connection between two nodes is positive, the constraint is satisfied to the degree that both nodes are active. Otherwise if the connection is negative, the constraint is violated to the degree that both nodes are active. A simple way to express this is the product of the activation of two nodes times the weight connecting them, i.e., $w_{ij}a_ia_j$. Note that for positive weights the more active the two units are, the better the constraint is satisfied; whereas for negative weights the more active the two units are, the less the constraint is satisfied. Second, the priori probability of a hypothesis is captured by adding the bias to the goodness measure. Finally the goodness of a node when direct evidence is available is given by the product of the input value times the activation value of the

node. The bigger this product, the better the system is satisfying this external constraint. Overall the *goodness* of *node i* can be defined as in Equation 11, where the symbols share the same definition as Equation 10.

$$goodness_i = \sum_j w_{ij} a_i a_j + bias_i a_i + ei_i a_i = net_i a_i \qquad (11)$$

Since we are concerned with the degree to which the entire pattern of values assigned to all of the hypotheses are consistent with the entire body of constraints, the *overall goodness* is defined as the sum of all individual goodnesses. Now the constraint satisfaction problem is converted to the problem of maximizing the overall goodness. In practice we look for the $\delta_{goodness}$ between time *t* and *t-1* (see Equation 12) less than a threshold to be a stop condition.

$$\delta_{goodness} = \sum_i goodness_i(t) - \sum_i goodness_i(t-1) \qquad (12)$$

### 4.5.4 The Implementation of the IAC Neural Network

Many constraints, e.g. the cardinality of a property, have been used to restrict ontologies. Different constraints result in different connections between nodes in the IAC neural network. For example, the constraint that "only 1-to-1 mapping is allowed" results in a negative connection between nodes ($e_{1i}$, $e_{2j}$) and ($e_{1i}$, $e_{2k}$), where $k \neq j$. Moreover, "two elements match if their children match", results in a positive connection between nodes ($e_{1i}$, $e_{2j}$) and ($e_{1k}$, $e_{2t}$), where $e_{1k}$ and $e_{2t}$ are the children of $e_{1i}$ and $e_{2j}$ respectively.

Table 4.1 lists the constraints that have been implemented in our approach. Although the number of negative constraints is much less than that of positive constraints, the ratio of negative

connections to positive connections is not small. This is because each node in the network will have a large number of negative connections introduced by the 1-1 mapping of constraints. Though the weights of different constraints should be set to a function of its confidence, currently we set the weight of positive constraints as 1 and the weight of negative constraints as –1. Meanwhile, for those mappings hypotheses that have no ambiguity (i.e., holding the highest similarity score in its corresponding row and column in the aggregated similarity matrix), we set their external input as 10.

**Table 4.1** The constraints used in the PRIOR+ approach

| # | Constraints | Connection |
|---|---|---|
| 1 | Only 1-1 mapping is allowed. | negative |
| 2 | No crisscross mapping is allowed. | negative |
| 3 | If children elements match, then their parent elements match. | positive |
| 4 | If parent elements match, then their children elements match. | positive |
| 5 | If $e_{1i}$ match $e_{2j}$, then $e_{1s}$ match $e_{2t}$, where $e_{1i}$ and $e_{1s}$, $e_{2j}$ and $e_{2t}$ are siblings in ontologies. | positive |
| 6 | If property elements match, then their domain elements match. | positive |
| 7 | If property elements match, then their range elements match. | positive |
| 8 | If class elements match, then their direct property elements match. | positive |
| 9 | If property elements match, then their mother-class elements match. | positive |
| 10 | If class elements match, then their instance elements match. | positive |
| 11 | If instance elements match, then their mother-class elements match. | positive |
| 12 | Two elements match if their *owl:sameAs* or *owl:equivalentClass* or *owl:equivalentProperty* elements match. | positive |

## 4.6 TWO SPECIFIC TECHNIQUES IMPLEMENTED IN THE PRIOR+

The PRIOR+ is a generic ontology mapping tool. It analyzes the characteristics of different kinds of similarities, adaptively aggregates them and integrates a neural network model to solve the constraint satisfaction problem in the context of ontology mapping. All implementations of the PRIOR+ are built in JAVA on a stand-alone PC running Ubuntu 6.0.6 OS, with Intel Dual Core 1.8 Hz processor, 1.5G memory, 100GB Serial ATA hard disk and SUN JAVA VM 1.6.0.

This section presents two specific techniques that have been implemented in the PRIOR+. One is to use Hadoop[1] to support distributed parallel computing when calculating the profile similarity between elements. The other is to integrate Indri[2] search engine to find mappings between large-scale ontologies by treating the mapping problem as an information retrieval task. Both of the two techniques are motivated by the fact that we have to face the problem of processing large amount of data in some ontology mapping tasks such as the anatomy task in OAEI ontology matching campaign 2006 and 2007. Classic computing methods either need a very long time (up to days) to get results (see data in Table 4.9[3]) or are unable to handle such problems at all.

[1] http://lucene.apache.org/hadoop/

[2] http://www.lemurproject.org/indri/

[3] From http://webrum.uni-mannheim.de/math/lski/align2007/results.html

### 4.6.1 The Implementation of Hadoop in the PRIOR+

Hadoop is "*a Free Java software framework that supports distributed applications running on large clusters of commodity computers that process huge amounts of data*."[1] On the basis of Hadoop Distributed File System (HDFS), it implements the MapReduce (Dean and Ghemawat 2006), a distributed parallel computing infrastructure published by Google. Currently many search engines such as Yahoo[2], Ask.com[3], PowerSet[4] are using Hadoop to process large scale data on the Web.

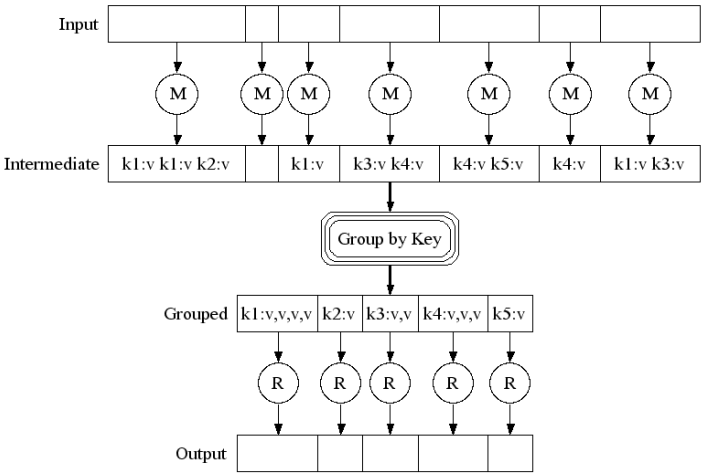



**Figure 4.8** The execution of MapReduce[5]

Figure 4.8 shows the execution of MapReduce. All the input files will be stored in a distributed file system, such as HDFS. Each mapper works on a small part of input. For each

[1] http://en.wikipedia.org/wiki/Hadoop

[2] http://www.yahoo.com

[3] http://www.ask.com

[4] http://www.powerset.com

[5] From http://labs.google.com/papers/mapreduce-osdi04-slides/index-auto-0007.html

record, a key/value pair is generated. All pairs with the same key then are transmitted to one reducer, where their values are counted together. Finally several key/value pairs are output. Though finishing a reducer needs to finish all its related mappers first, mappers and reducers can run independently, i.e., in parallel. Figure 4.9 shows the parallel execution of MapReduce.

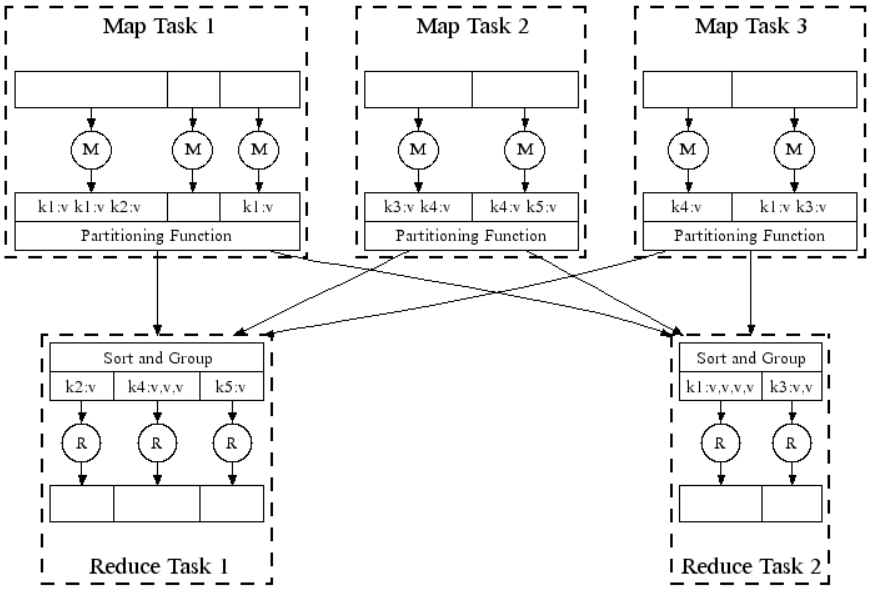



**Figure 4.9** The parallel execution[1] of MapReduce

Hadoop can be used to in many scenarios. The simplest application of Hadoop might be to count the frequency of each word in a large document. For example, a mapper replaces all *empty* character with *enter* character so that each word will be listed on one line, and then transmit them to reducers. The same word will be sent to the same reducer, where its number will be counted. The reducers finally output the frequency of each word. In this case, the key in the mapper is each word, its value is as simple as 1. The key in the reducer is each word, and its value is the frequency of the word.

[1] From http://labs.google.com/papers/mapreduce-osdi04-slides/index-auto-0008.html

Hadoop is used in the PRIOR+ for large-scale ontology mapping tasks (i.e. the anatomy task in OAEI campaign 2007) to build an inverted index and calculate the cosine similarity when generating profile similarity (described in §4.3.2). In the PRIOR+, we treat each profile as a *document*, and each unique word appearing in the profile as a *term*.

Table 4.2 gives an example of documents, terms and their weights, in which $D_{ij}$ denotes document $D_i$ from ontology $O_j$, $T_k$ denotes each unique term, and $W_{TkDij}$ denotes a weight (e.g. the $tf \bullet idf$ weight) of term $T_k$ in document $D_{ij}$.

First we use Hadoop to build an inverted index for Table 4.2. The mapper parses each document and emits a sequence of $<Tk, D_{ij}W_{TkDij}>$ pairs. The reducer accepts all pairs for a given term, sorts the corresponding documents and emits a $<T_k, list(D_{ij}W_{TkDij})>$ pair. The set of all output pairs forms a simple inverted index, as shown in Table 4.3.

Having the inverted index table, we then use Hadoop to calculate cosine similarity between profiles of different elements (i.e. documents). This time, the mapper parses the inverted index file and emits a sequence of $<D_{ij}D_{pq}, W_{TkDij}W_{TkDpq}>$ paris. The reducer accepts all pairs for a pair of documents, sorts the corresponding documents and emits a $< D_{ij}D_{pq}, list(W_{TkDij}W_{TkDpq})>$ pair. The set of outputs corresponds to the cosine similarity between profiles of different elements, as shown in Table 4.4, where $f$ is the function to calculate cosine similarity between two given vectors.

**Table 4.2** Sample of document, term and its weights for profile similarity generation

| Term (in profile) / Document (i.e., profile) | $T_1$ | $T_2$ | $T_3$ | $T_4$ |
|---|---|---|---|---|
| $D_{11}$ | $W_{T1D11}$ | $W_{T2D11}$ | | $W_{T4D11}$ |
| $D_{21}$ | | $W_{T2D21}$ | $W_{T3D21}$ | |
| $D_{12}$ | $W_{T1D12}$ | $W_{T2D12}$ | | $W_{T4D12}$ |

**Table 4.3** Using Hadoop to build inverted index

| Term | Corresponding documents and weights | | |
|---|---|---|---|
| $T_1$ | $D_{11}$ ,$W_{T1D11}$ | | $D_{12}$ ,$W_{T1D12}$ |
| $T_2$ | $D_{11}$ , $W_{T2D11}$ | $D_{21}$ ,$W_{T2D21}$ | $D_{12}$ ,$W_{T2D12}$ |
| $T_3$ | | $D_{21}$ ,$W_{T3D21}$ | |
| $T_4$ | $D_{11}$ ,$W_{T4D11}$ | | $D_{12}$ ,$W_{T4D12}$ |

**Table 4.4** Using Hadoop to calculate of cosine similarity between profiles

| Profiles | Cosine similarity between two profiles |
|---|---|
| $D_{11}$ $D_{12}$ | $f(W_{T1D11}$ $W_{T1D12})+f(W_{T2D11}$ $W_{T2D12})+f(W_{T4D11}$ $W_{T4D12})$ |
| $D_{11}$ $D_{12}$ | $f(W_{T2D11}W_{T2D12})$ |

### 4.6.2 The Implementation of Indri in the PRIOR+

Indri is an open source search engine. Indri is integrated in the PRIOR, the previous version of PRIOR+, to support mapping discovery between large scale ontologies (i.e. the anatomy task in OAEI campaign 2006). In the case, Indri is used to look for mapping candidates between large scale ontologies as follows:

Given two ontologies, $O_A$ and $O_B$, first we index all profiles in $O_A$ as documents. Simultaneously we generate queries based on profile in $O_B$. Then we search in $O_A$ using queries generated from $O_B$ by calculating the similarity between queries and documents. Those concepts in $O_A$ with top-ranked similarity or above a predefined threshold are stored. Now two ontologies are switched and the whole process is repeated. The overlapped results in two processes indicate mapping candidates.

## 4.7 EVALUATION

This section presents the test cases that are used to evaluate the performance of the PRIOR+ in different scenarios, followed by the experimental methodology and results.

### 4.7.1 The OAEI Campaign and its Test Cases

To evaluate the performance of the PRIOR+ approach we use test cases from the OAEI ontology matching campaign 2007[1]. OAEI ontology matching campaign is a yearly contest organized by Ontology Alignment Evaluation Initiative (OAEI) since 2004. OAEI campaign 2007 consists of three tracks, five data sets as listed in Table 4.5. To evaluate the performance of different approaches OAEI adopts two evaluation methods, i.e., *open* vs. *blind*. *Open* means the golden standard is available to participants; *blind* means only the organizer knows true mapping results and thus the evaluation on these tests can only be done by OAEI. The PRIOR+ participated in three tasks, i.e. the benchmark task, the web directory task and the anatomy task.

**Table 4.5** The overview of OAEI campaign 2007

| Track | Dataset | Representation | Evaluation Process |
|---|---|---|---|
| Comparison Track | Benchmark | OWL | Open |
| Directories and Thesauri | Web directory | OWL | Blind |
| | Food | SKOS | Blind |
| | Environment | SKOS | Blind |
| | Library | SKOS | Blind |
| Expressive Ontologies | Anatomy | OWL | Blind |

#### 4.7.1.1 Benchmark Tests

The OAEI benchmark test ontologies[2] originate from the bibliography domain. It includes one reference ontology $O_R$ dedicated to the very narrow domain of bibliography, multiple test ontologies $O_T$ manually discarding various information from the reference ontology in order to evaluate how algorithms behave when information is lacking, and 4 real world bibliographic

[1] http://oaei.ontologymatching.org/2007/

[2] http://oaei.ontologymatching.org/2007/benchmarks/

ontologies (i.e., #301-#304) that are generated by MIT[1], UMBC[2], University of Karlsruhe[3] and INRIA[4] respectively. More specifically, benchmark tests can be divided into 5 groups as described in Table 4.6. The detailed description for each test ontology can be found in Appendix A.

**Table 4.6** The overview of OAEI benchmark tests

| Tests | Description |
|---|---|
| #101-104 | $O_R$ and $O_T$ have exactly the same or totally different names |
| #201–210 | $O_R$ and $O_T$ have the similar structure but different linguistics in some level |
| #221–247 | $O_R$ and $O_T$ have the similar linguistics but different structure |
| #248–266 | Both structure and linguistics are different between $O_R$ and $O_T$ |
| #301-304 | $O_T$ are real world cases, which we have more interest in |

#### 4.7.1.2 Web Directory Tests

The Web directory tests[5] consist of 4640 elementary tests for aligning web sites directories (e.g., Google and Yahoo!). Each test is represented by pairs of OWL ontologies, where classification relation is modeled as OWL *subClassOf*. Therefore all directory ontologies are organized as taxonomy hierarchies, i.e., the ontologies only contain *classes* connected with

[1] http://visus.mit.edu/bibtex/0.1/

[2] http://ebiquity.umbc.edu/

[3] http://www.aifb.uni-karlsruhe.de/ontology

[4] http://oaei.ontologymatching.org/2007/benchmarks/fr.inrialpes.exmo.rdf.bib.owl

[5] http://www.dit.unitn.it/~yatskevi/directory.htm

subclass relation between each other. Figure 4.10 is a sample mapping between Google and Yahoo web directories.


**Figure 4.10.** Sample mappings between Google and Yahoo web directories

#### 4.7.1.3 Anatomy Test

The anatomy task[1] requires the tool to find mappings between classes in two medical ontologies, i.e., Adult Mouse Anatomy ontology[2] with 2744 classes and the NCI Thesaurus[3] with 3304 classes describing the human anatomy.

[1] http://webrum.uni-mannheim.de/math/lski/align2007/

[2] http://webrum.uni-mannheim.de/math/lski/align2007/mouse_anatomy.owl

[3] http://webrum.uni-mannheim.de/math/lski/align2007/nci_anatomy.owl

### 4.7.2 The Evaluation Criteria

We follow the evaluation criteria used by the OAEI ontology matching campaign 2007. The evaluation is based on the results provided by participants. All results include a set of mapping pairs, which are expressed in the format as shown in Figure 4.11, where the element of *Address* in *entity1* can be mapped to the element of *Address* in *entity2*, with the measure of *1.0* and "=" relation.

```
<map>
   <cell>
      <entity1 rdf:resource='http://oaei.ontologymatching.org/2006/benchmarks/101/onto.rdf#Address'/>
      <entity2 rdf:resource='http://oaei.ontologymatching.org/2006/benchmarks/101/onto.rdf#Address'/>
      <measure rdf:datatype='http://www.w3.org/2001/XMLSchema#float'>1.0</measure>
      <relation>=</relation>
   </cell>
</map>
```

**Figure 4.11** A sample of mapping pair

OAEI campaign 2007 has two evaluation processes. Benchmark tests are provided with the expected results, i.e., they are *open*. Directories tests and anatomy test are *blind* tests, i.e., participants do not know the results. For all tests, standard information retrieval evaluation measures, i.e., *precision*, *recall* and *f-measure*, are computed against the reference alignment. For the matter of aggregation of the measures, *weighted harmonic means* are computed as well (Euzenat *et al*. 2006). The *precision*, *recall* and *f-measure* are defined by Equation 13, 14 and 15.

$$Precision \quad p = \frac{\#correct\_found\_mappings}{\#all\_found\_mappings} \tag{13}$$

$$Recall \quad r = \frac{\#correct\_found\_mappings}{\#all\_possible\_mappings} \tag{14}$$

$$F - measure \quad f = \frac{2 \times p \times r}{p + r} \tag{15}$$

### 4.7.3 Experimental Methodology and Results

To evaluate the PRIOR+ approach, we designed 6 experiments. Each answers one of the following questions:

1. What is the performance of each individual similarity, i.e., edit distance based similarity, profile similarity, and structure similarity?
2. Does propagation improve the performance of profile similarity? If it does, what is the best configuration of weight assignments for neighboring elements' profiles?
3. Can the measure of *harmony* reflect the reliability of different similarities? That is, does it correlate to the performance of a similarity?
4. Is the harmony-based adaptive aggregation method better than other aggregation methods discussed in §4.4?
5. Does the IAC neural network work in the context of ontology mapping? If it does, how much does it improve the results?
6. What is the overall performance of the PRIOR+? How does it perform compared with other systems?

All the 6 experiments are run on the OAEI benchmark tests. We list the results of the PRIOR+ in the OAEI web directory task and anatomy task. Both results were evaluated by OAEI[1] because they are *blind* tests.

#### 4.7.3.1 The Comparison of Each Individual Similarity

In §4.3 we proposed three kinds of similarities, i.e., edit distance based similarity, profile similarity, and structure similarity. Each similarity measures the correspondence between two elements from a different perspective. Figure 4.12 compares the performance (i.e., f-measure) of 3 individual similarities on each OAEI benchmark test. Figure 4.13 compares the performance (i.e., f-measure) of each individual similarity over all OAEI benchmark tests.

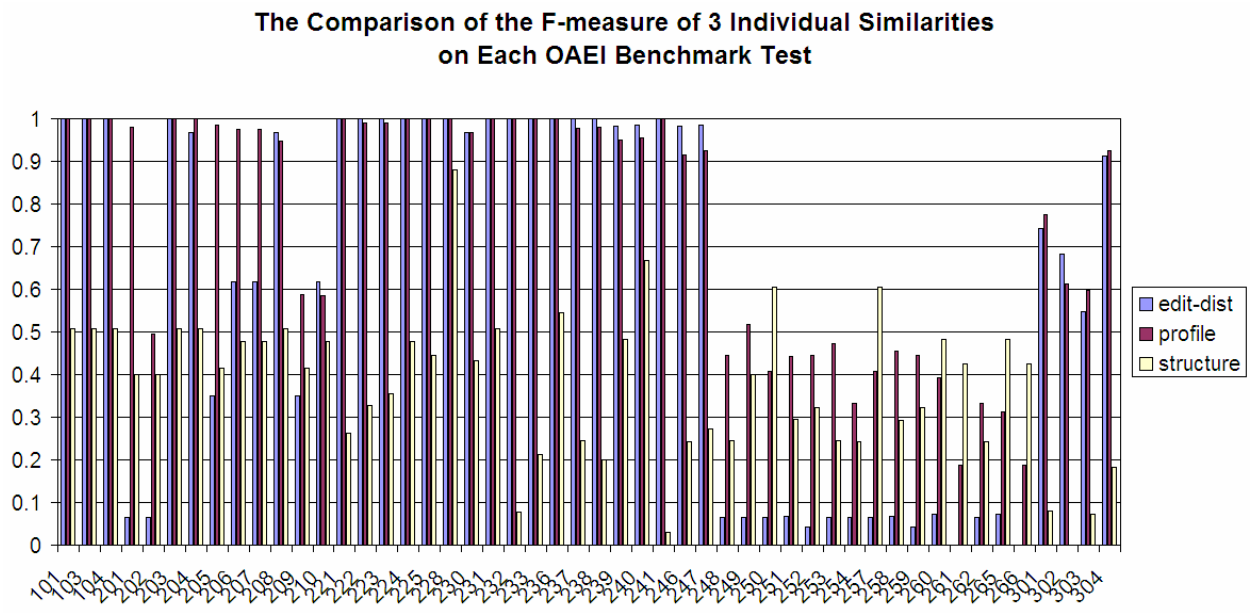



**Figure 4.12** The comparison of the f-measure of 3 individual similarities on each OAEI benchmark test

[1] http://oaei.ontologymatching.org/2007/results/

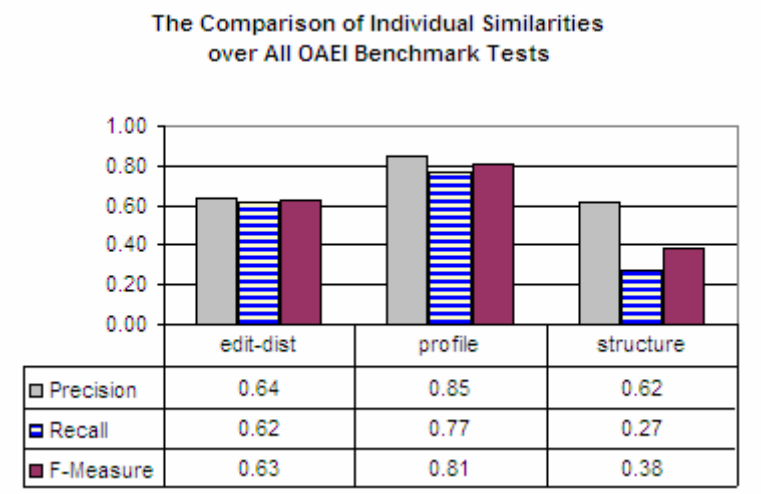



**Figure 4.13** The comparison of individual similarities over all OAEI benchmark tests

The observations from Figure 4.12 and Figure 4.13 are:

1. The edit distance based similarity is intuitive. It works very well on the cases that have high similarity between the names of elements in ontologies. For example, Test #101-104, #203, #208, #221-247, #301, #302, #304, etc. However such similarity is more lexical-oriented than semantic-oriented, which encounters trouble where synonyms exist. In the cases that have very low linguistic similarity, e.g., #201, #202 and #248-#266, the performance of the edit distance based similarity is very poor. One solution to overcome the limitation of edit distance based similarity is to check auxiliary information in a thesaurus, e.g. WordNet[1]. However WordNet suffers some drawbacks as well as discussed in §5.4.2 . Moreover, integrating WordNet will cost much more time in finding synonymous relations between words, and thus decrease the efficiency of the whole approach.

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

2. The structure similarity explores structural features in two ontologies. It is extremely useful in pure graphic mapping tasks, for example in #248-#266 where meaningful linguistic information has been removed or replaced by some randomly generated information. However structure similarity contributes very little in the cases where linguistic information is adequate, e.g., #221-247, or in the case where structural information is limited, e.g., #301, #303-#304, or does not exist at all, e.g. #302, in which the hierarchy of the ontology is absolutely flat. Finally, the overall recall of structure similarity is as low as .27, which indicates relying on this similarity only can not help us to find most mappings.
3. The profile similarity utilizes all kinds of descriptive information to generate a *profile* for each element, and then compares the cosine similarity of two profiles in a vector space model. The profile similarity works very well when linguistic information is adequate, e.g., #101-#104, #221-247, #301, #302, #304. Meanwhile, since the profile similarity explores the structural information of an element by integrating its property information, instance information and neighboring information, it also works well in the cases where linguistic information is limited, e.g., #201, #202, #205-#207, #209, #248-266. Generally speaking, the profile similarity takes advantage of both edit distance comparison and structure analysis, and thus it outperforms edit distance based similarity and structure similarity in most cases except #250, #257, #261, #265, #266, where no or very very little lexical information is available. Therefore, the mapping totally relies on the structure information. The precision, recall and f-measure of the profile similarity over all OAEI benchmark tests are .85, .77 and .81 respectively.

4. The fact that different similarities work well in different situations motivates us to investigate a new measure that can estimate the quality of each similarity so that we can aggregate them according to their individual characteristic.

#### 4.7.3.2 The Impact of Propagation in Generating Profile Similarity

Profile propagation extends the linguistic analysis by passing the neighboring information of an element to the element itself. What we are interested is whether the propagation does improve the f-measure of the profile similarity. If it does, how much improvement we can get due to the propagation.

The experimental methodology is: First we generate a profile for each of element in the ontologies. Then we calculate the cosine similarity between two profiles directly without passing its neighboring information to itself. Next, we propagate the neighboring information (i.e., parent, children and siblings) of a profile to the profile itself, and then calculate the cosine similarity between the propagated profiles. For the two kinds of similarity scores, we extract final mapping results using naïve descendant extraction algorithm (Meilicke and Stuckenschmidt 2007). Finally we compare the difference between the f-measure of two approaches on each of OAEI benchmark test. Figure 4.14 shows the comparison of f-measure between profile propagation and non-propagation.

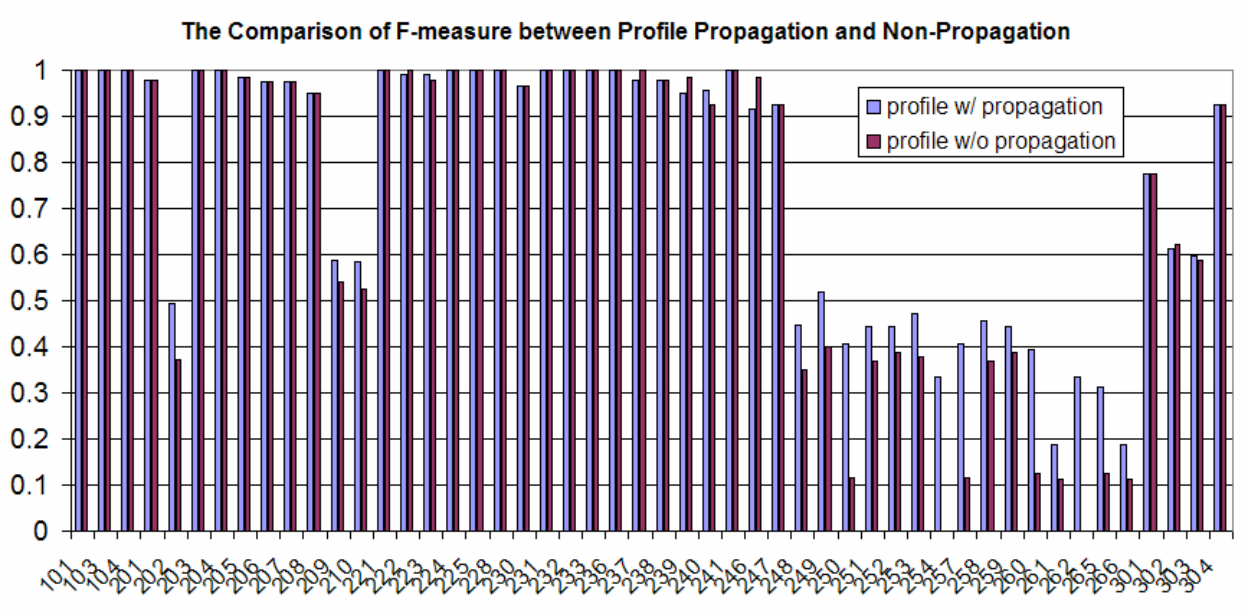



**Figure 4.14 The comparison of f-measure between profile propagation and non-propagation**

The observations from Figure 4.14 are:

1. If we calculate the f-measure of profile propagation and non-propagation over all OAEI benchmark tests, they are .8098 and .8005. The overall improvement of profile propagation vs. non-propagation is about 1%. However, there is significant difference ($p$=.0002 on one-tail $t$-test) between profile propagation and non-propagation over all OAEI benchmark tests. And there is significant difference ($p$=4E-5 on one-tail $t$-test) between profile propagation and non-propagation on OAEI benchmark tests #248-#266. But there is no significant difference between profile propagation and non-propagation on OAEI benchmark tests #1xx, #201-#247 and #3xx.
2. This means the impact of profile propagation is very little when meaningful information of an element is adequate. It works only when no or very limited meaningful information is available for the element itself, but some useful

information can be found in it neighboring elements. However propagation makes the performance of the Prior+ approach more reliable on some cases. For example, profile propagation significantly improves performance on #248-#266.

We conducted an experiment trying to determine the best settings for the weights of neighboring elements (e.g., parent and children) when propagating profiles. To simplify the number of variants of parameter configuration, we fixed the weight of sibling as .2 in the experiment. The experimental methodology is: We adjust the weight of parent and children to different numbers, range in [0, 1], step in .1. Then we have 11×11 (i.e. 121) configurations totally. Given each configuration, we calculate the f-measure over all OAEI benchmark tests using the PRIOR+ approach but not activating the IAC neural network. The result is shown in Figure 4.15, where the *x* axle represents the weight range of parent, the *y* axle represents the weight range of children, and the *z* axle represents the f-measure of the PRIOR+ without activating the IAC network over all OAEI benchmark tests.

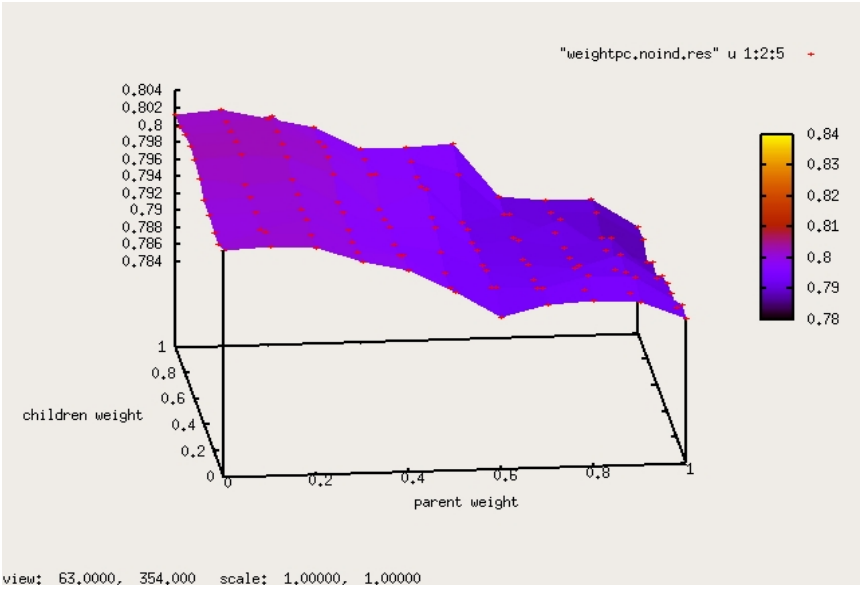



**Figure 4.15** The correlation of parent's weight and children's weights

The observations from Figure 4.15 are:

1. Generally speaking, the difference of the profile propagation with 121 configurations is very small. The f-measure varies from .7854, when parent weight is 1 and children weight is .8, to .8026, when parent weight is .1 and children weight is .7.
2. More specifically, when the weight of children is 0, i.e., considering the impact of parent only, the trend of the f-measure decreases along with the increase of parent weight. The lowest f-measure is achieved when parent weight is .6.
3. Furthermore, when the weight of parent is 0, i.e., considering the impact of children only, the trend of the f-measure increases along with the increase of children weight. The highest f-measure is achieved when children weight is .7.

#### 4.7.3.3 The Correlations between the Harmony and the Characteristic of Similarities

As stated in §4.4.2 that the *harmony* is defined to estimate the importance and reliability of a similarity, which can be reflected by its performance, e.g. f-measure. We evaluated the correlation between harmony and f-measure.

The experimental methodology is: For each test, we calculate 5 similarities, i.e., class name similarity, class profile similarity, class structural similarity, property name similarity and property profile similarity. For each similarity matrix, we extract mapping results using naive descendant extraction algorithm (Meilicke and Stuckenschmidt 2007). After that we evaluated the results against the reference alignment and got the f-measure of each similarity. Meanwhile, we estimate 5 harmonies for its corresponding matrix. Finally we measure the degree of the correlation between f-measure and harmony on each similarity. The results are shown in Figure 4.16 and Figure 4.17.

The observations from Figure 4.16 and Figure 4.17 are:

1. The harmony does linearly correlate with f-measure of different similarities, especially on estimating the performance of class' ID, property's ID and property's profile.
2. More specifically, the $R^2$ for edit distance based similarity (i.e., name similarity), profile similarity, and structural similarity on class are .97, .85 and .77 respectively in Figure 4.16. The $R^2$ for the edit distance based similarity (i.e., name similarity) and profile similarity on property are .96 and .97 respectively in Figure 4.17.

The observation that the harmony is indeed a good estimator of f-measure for each individual similarity makes us confident in using harmony to adaptively aggregate similarities.

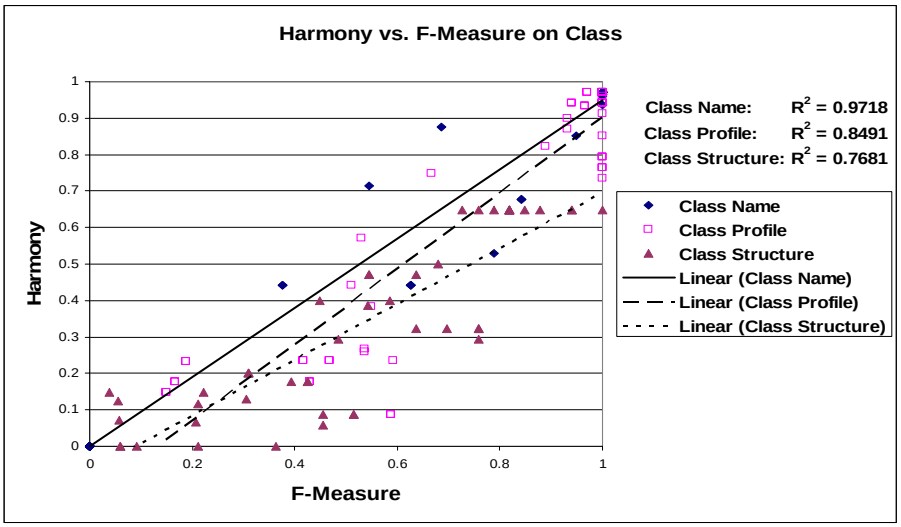



**Figure 4.16** Correlation of harmony vs. f-measure on class

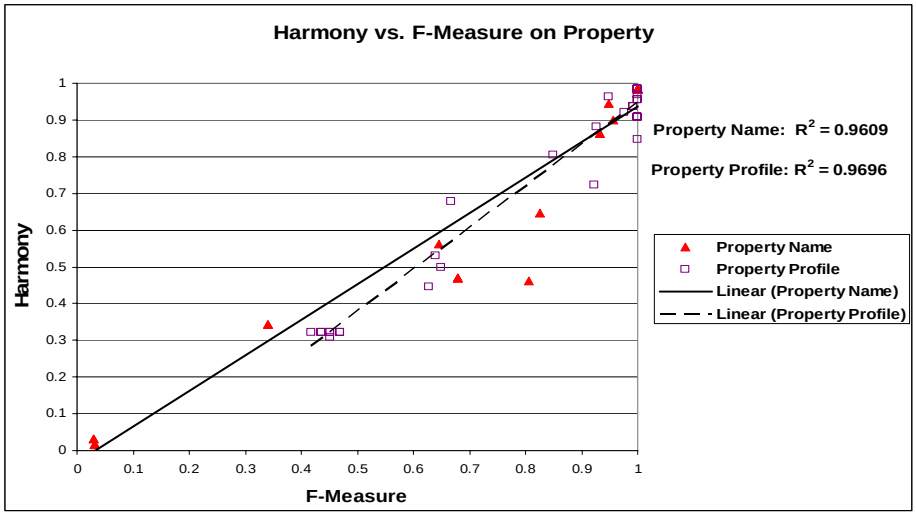



**Figure 4.17.** Correlation of harmony vs. f-measure on property

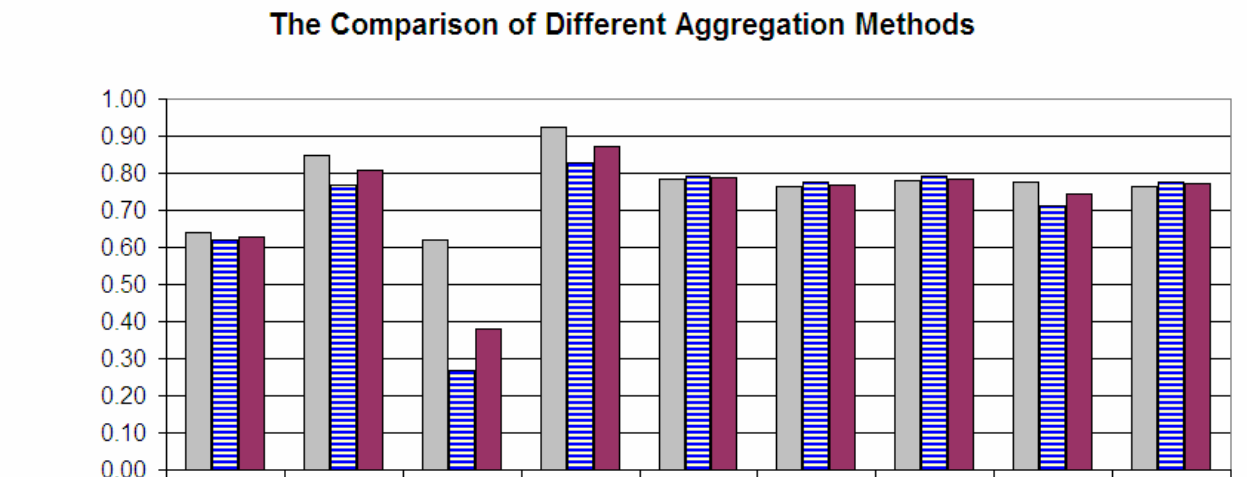
#### 4.7.3.4 Comparison of Different Aggregation Methods

Similarity aggregation has been studied in many ontology mapping approaches as discussed in §4.4.1. Data aggregation, called *data fusion*, has also been widely investigated in information retrieval area (Fox and Shaw 1994). To evaluate the harmony-based adaptive weighted aggregation method (HADAPT), we compared the performance of HADAPT with six other aggregation methods selected from both ontology mapping and information retrieval. Table 4.7 lists the name and brief description of seven aggregation methods, where $s_i$ denotes the $i^{th}$ similarity, *fs* denotes the final aggregated similarity, $h_i$ denotes the harmony of $i^{th}$ similarity, *N* denotes the number of individual similarity, *Nz* denotes the number of non-zero similarities.

**Table 4.7** Aggregation functions used in the experiment

| **Method** | **Description** | **Equation** |
|---|---|---|
| HADAPT | Harmony based adaptive aggregation | $fs = \text{sum}(h_i * s_i)/N$ |
| MAX | Maximum of individual similarities | $fs = \max(s_i)$ |
| AVG | Average of individual similarities | $fs = \text{sum}(s_i)/N$ |
| ANZ | AVG ÷ number of nonzero similarities | $fs = (\text{sum}(s_i)/N)/Nz$ |
| MNZ | AVG × number of nonzero similarities | $fs = (\text{sum}(s_i)/N) * Nz$ |
| SIGMOID | Average of individual similarities smoothed by sigmoid | $fs = \text{sum}(\text{sigmoid}(s_i))/N$ |

The experimental methodology is: For each test, we first calculated three individual similarities (i.e. name similarity, profile similarity and structural similarity) as described in §4.3. Then we aggregated the individual similarities using different aggregation methods as listed in Table 4.7. After aggregation, we applied the naïve descendant extraction algorithm (Meilicke and Stuckenschmidt 2007) to extract final mappings. Precision, recall and f-measure of final results on each test are calculated. Finally the overall precision, recall and f-measure are calculated over all benchmarks tests. The results are shown in Figure 4.18.




| | edit-dist | profile | structure | HADAPT | AVG | MAX | MNZ | ANZ | Sigmoid |
|---|---|---|---|---|---|---|---|---|---|
| Precision | 0.64 | 0.85 | 0.62 | 0.92 | 0.78 | 0.76 | 0.78 | 0.78 | 0.77 |
| Recall | 0.62 | 0.77 | 0.27 | 0.83 | 0.79 | 0.77 | 0.79 | 0.71 | 0.78 |
| F-Measure | 0.63 | 0.81 | 0.38 | 0.87 | 0.79 | 0.77 | 0.79 | 0.74 | 0.77 |

**Figure 4.18** The comparison of different aggregation methods

The observations from Figure 4.18 are:

1. The performance of 3 individual similarities is various. More detailed analysis of each individual similarity has been given in §4.7.3.1.
2. The performance of profile similarity is better than most of aggregation methods except *HADAPT* method. The interesting phenomenon tells us the final aggregated result really depends on the parameters employed in the aggregation method. It is critical to choose right parameters to boost the final result of multiple similarities based ontology mapping approach.
3. Without tuning parameters specifically, the performance of AVG, MAX, ANZ, MNZ, and Sigmoid aggregation methods are competitive with each other. The f-measure of them is between .74-.79.
4. The harmony based adaptive similarity aggregation method (i.e., HADAPT) outperforms all other methods when aggregating different similarities. It holds the

highest precision, recall, and f-measure at .92, .83 and .87 respectively, which improves f-measure 7% over the other methods by 7% or more.

#### 4.7.3.5 The Improvement of NN-based Constraint Satisfaction Solver

Having the output from Similarity Generator, the harmony of the aggregated similarity will be estimated. If the harmony is less than a threshold, which is set as .8 in the experiment, the IAC Neural Network will be activated to search for a global optimal solution that satisfy as many ontology constraints as possible. Otherwise, final mapping results will be directly extracted using naïve descendant extraction algorithm (Meilicke and Stuckenschmidt 2007).

In the case of OAEI benchmark tests, the NN-based constraint satisfaction solver was activated on tests #202, #209, #210, #248-#266, #302 and #303. Figure 4.19 shows the change of the performance on each of the 20 tests after activating the IAC neural network. Table 4.8 shows the change of the performance over all 20 tests.

The observations from Figure 4.19 and Table 4.8 are:

1. The NN-based constraint satisfaction solver dramatically improves both precision and recall by more than .1.
2. Among 20 tests, 16 get improved on their f-measure (e.g., the biggest improvement of f-measure is .37 on #262), only 4 get lower f-measures. The reason why the performance of #261 is worse than all others is: #261 extends the structure of the test ontology. It adds some new classes, sometimes these new classes are a new layer in the taxonomy of the ontology. In this case some constraints in the IAC neural network are not correct anymore (e.g. constraint #3 and #4 in Table 4.1). In real cases we usually have linguistic information as well as structure information, which will

decrease the impact of the difference of the structure. Unfortunately, #261 does not have any linguistic information that we can rely on.

3. If we calculate the percentage improvement of the IAC neural network over the 20 tests, they are 13%, 24%, and 19% for precision, recall, and f-measure respectively.

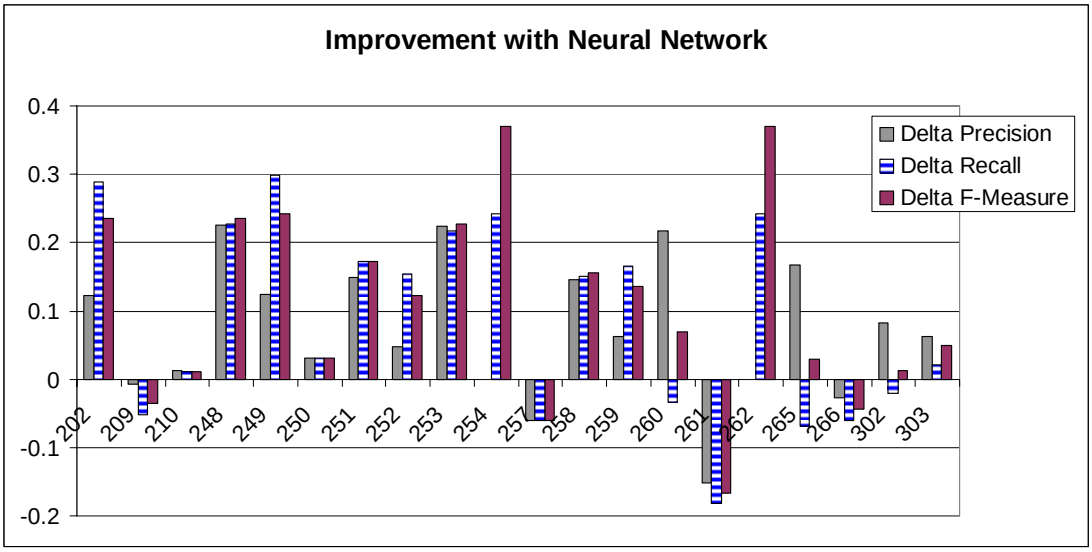



**Figure 4.19** The improvement after applying the IAC neural network on each selected test

**Table 4.8** The overall improvement of the IAC neural network on all 20 tests

| | **Precision** | **Recall** | **F-Measure** |
|---|---|---|---|
| Before NN | .76 | .54 | .63 |
| After NN | .88 | .67 | .76 |
| NN Improvement | 13% | 24% | 19% |

#### 4.7.3.6 The Performance of the PRIOR+ over all OAEI 2007 Benchmark Tests

Figure 4.20 is the performance of the PRIOR+ over all OAEI 2007 benchmark tests. Figure 4.21 is the performance of the PRIOR+ over 5 categories of OAEI 2007 benchmark tests (see categorization in §4.7.1). The full results of the PRIOR+ approach over all benchmark tests can be found in Appendix A.

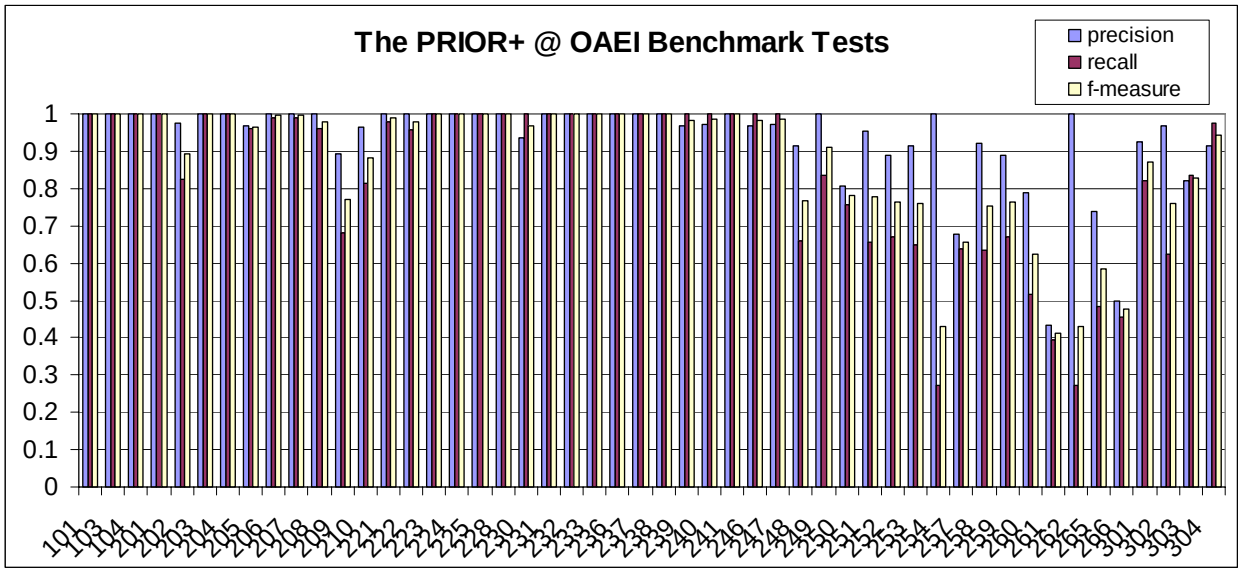
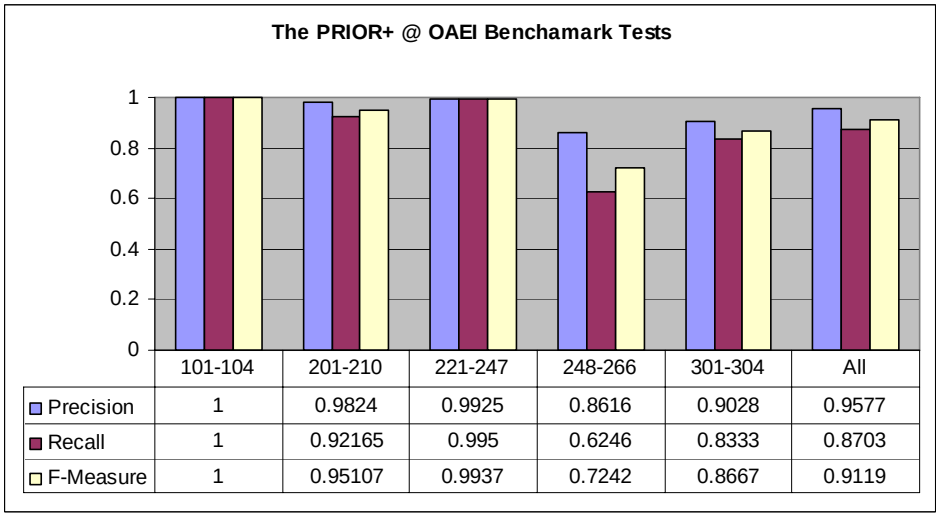



**Figure 4.20** The performance of the PRIOR+ over all OAEI 2007 benchmark tests




| | 101-104 | 201-210 | 221-247 | 248-266 | 301-304 | All |
|---|---|---|---|---|---|---|
| Precision | 1 | 0.9824 | 0.9925 | 0.8616 | 0.9028 | 0.9577 |
| Recall | 1 | 0.92165 | 0.995 | 0.6246 | 0.8333 | 0.8703 |
| F-Measure | 1 | 0.95107 | 0.9937 | 0.7242 | 0.8667 | 0.9119 |

**Figure 4.21** The performance of the PRIOR+ over 5 categories in OAEI benchmark tests

The observations from Figure 4.21 are:

1) The PRIOR+ performs perfect on benchmark tests #101–104, which are basic ontology mapping tests.
2) Tests #201–#210 are mapping tasks, in which reference ontology and test ontologies are similar on their structures but different on their linguistics. The PRIOR+ obtains high precision and recall where test ontologies contain the same names (or name conventions) and/or comments as the reference ontology (e.g. #203, #204, and #208). For those ontologies whose names of classes/properties have been "removed" or expressed in another language (e.g. #201, #206, #207 and #210), the PRIOR+ still can find some matched classes and properties using the information of comments and instances. However, if both names and comments are replaced or missing in test ontologies (e.g. #202 and #210), the PRIOR+ is not good at recall, down to .825 and .814 respectively. The recall of the PRIOR+ on #209 is down to .68 as well because the test ontology uses many synonyms, but the PRIOR+ does not integrate any thesaurus or synonym sets as external resources.
3) In tests #221–#247, where test ontologies have high similarity with reference ontology on linguistic information, the PRIOR+ performs well. Its precision, recall and f-measure are .9925, .995 and .9937 respectively.
4) In tests #248–#266, where test ontologies have very low similarity from both linguistic and structural perspective (i.e., descriptive information such as labels and comments have been removed and structures have been changed as well), the PRIOR+ does not achieve good precision (which is .8616) as it does on other tests, and recall falls to .6245.

5) Tests #301–#304 are four real world ontologies, modeled by different institutions but for the same domain of bibliographic metadata, and thus they have high similarity from linguistic perspective but low similarity from structural perspective. We are more interested in these tests than other benchmark tests because they are real world cases, so that finding correct mapping between real ontologies can demonstrate the usability of an approach from a comprehensive perspective. The results show the PRIOR+ obtains a precision of .9028, recall of .8333, and f-measure of .8667 on Tests #301-#304. PRIOR+ was the best among all systems on the real world cases on benchmark tests in OAEI campaign 2007. See comparison between the PRIOR+ and 4 top-ranked systems in Figure 4.22.

#### 4.7.3.7 The Comparison between the PRIOR+ and Top-ranked Systems on the Benchmark Test in OAEI Campaign 2007

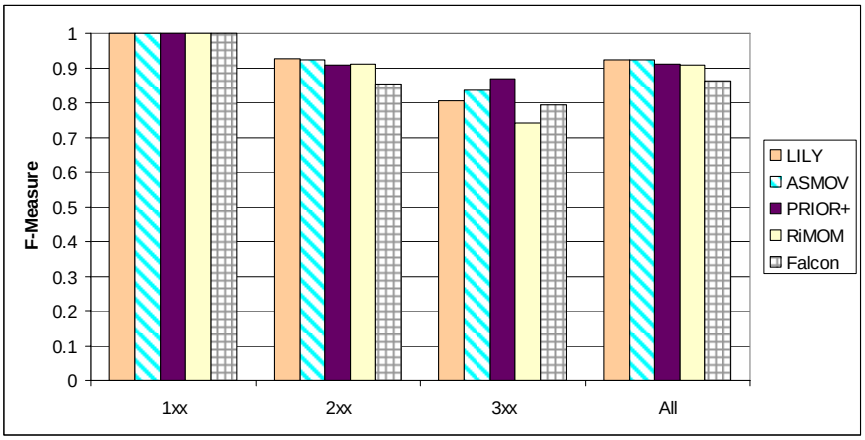



**Figure 4.22.** The comparison between the f-measure of the PRIOR+ and top ranked systems on benchmark tests in OAEI campaign 2007

Figure 4.22 compares the f-measure of PRIOR+ with 4 top-ranked ontology mapping systems, i.e., ASMOV (Jean-Mary and Kabuka 2007), Lily (Wang and Xu 2007), RiMOM (Tang, Li *et al*. 2006; Li, Zhong *et al*. 2007), and Falcon-AO (Qu, Hu *et al*. 2006; Hu, Zhao *et al*. 2007), on the benchmark tests in OAEI campaign 2007. The evaluation data of these 4 systems can be downloaded here[1]. The data for PRIOR+ can be downloaded here[2] .

The observations from Figure 4.22 are:

1. All systems perform perfect on test *1xx*.
2. The PRIOR+ is not as good as LILY, ASMOV and RiMOM on test *2xx* with a difference between .5% - 2.2%.
3. PRIOR+ performs the best on real world cases, i.e., *3xx*. The difference between PRIOR+ and others is between 3% - 17%.
4. Overall the f-measure of PRIOR+ (.912) outperforms that of RiMOM (.907) and Falcon-AO (.890) while it is less than LILY (.925) and ASMOV (.924). The difference between PRIOR+ and LILY and ASMOV is 1%.

#### 4.7.3.8 The Comparison between the PRIOR+ and Other Participants on OAEI Web Directory Task

Figure 4.23 compares the performance of PRIOR+ with 8 ontology mapping systems, i.e., Falcon-AO (Qu, Hu *et al*. 2006; Hu, Zhao *et al*. 2007), ASMOV (Jean-Mary and Kabuka 2007), DSSim (Nagy, Vargas-Vera *et al*. 2007), Lily (Wang and Xu 2007), OLA2 (Kengue,

[1] http://oaei.ontologymatching.org/2007/results/zip/

[2] http://www.sis.pitt.edu/om07/prior+_benchmark.zip This data is slightly different from what we submitted to the OAEI campaign after improving the PRIOR+ approach.

Euzenat *et al*. 2007), OntoDNA (Kiu and Lee 2007), RiMOM (Tang, Li *et al*. 2006; Li, Zhong *et al*. 2007), and X-SOM (Curino, Orsi *et al*. 2007) on the web directory test in OAEI campaign 2007.

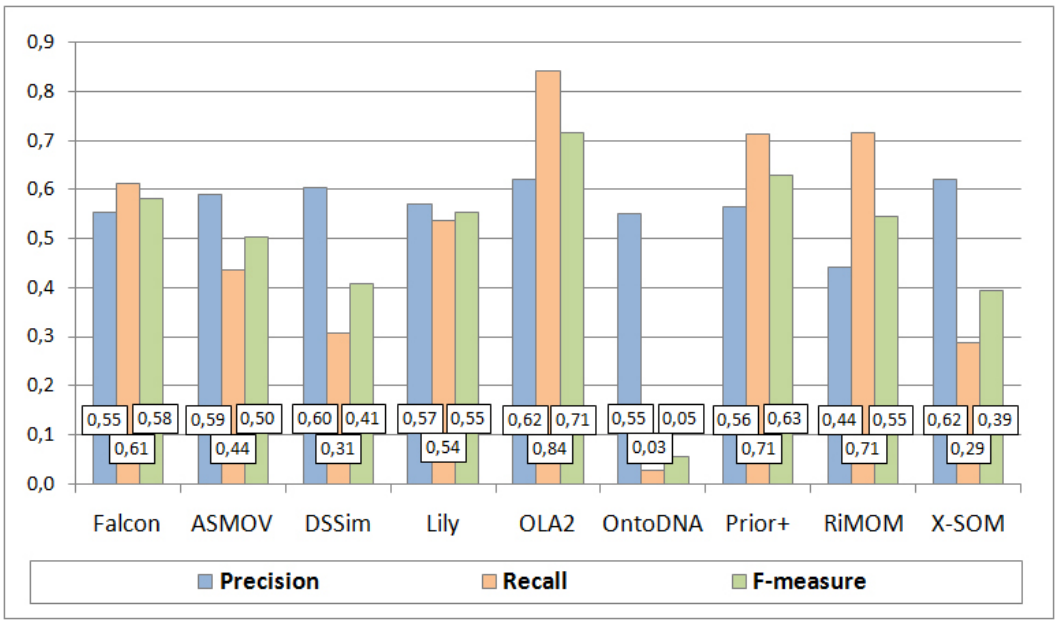



**Figure 4.23** The comparison between the PRIOR+ and other participates on OAEI 2007 web directory task[1]

The observations from Figure 4.23 are:

1. The precision, recall and f-measure of the PRIOR+ are .56, .71 and .63.
2. The f-measure of the PRIOR+, i.e., .63, is the 2nd place among 9 participants. It is lower than that of OLA2 system, i.e. .71, but outperformed all other systems.

[1] From http://www.dit.unitn.it/~pavel/om2007/oaei07/WebDirRes-OAEI07.html

#### 4.7.3.9 The Comparison between the PRIOR+ and Other Participates on OAEI Anatomy Task

Table 4.9 compares the performance of PRIOR+ with 11 systems that participated in the OAEI anatomy task 2007. The OAEI divided the 11 systems into three groups. Systems of type A, i.e., AOAS (Zhang and Bodenreider 2007) and Sambo (Tan and Lambrix 2007), are highly specialized on matching biomedical ontologies and make extensive use of medical background knowledge. Systems of type B, i.e., ASMOV (Jean-Mary and Kabuka 2007) and RiMOM (Tang, Li *et al*. 2006; Li, Zhong *et al*. 2007), can solve matching problems of different domains, but include a component exploiting biomedical background knowledge (e.g. using UMLS as lexical reference system). Systems of type C finally can be seen as all-round matching systems that do not distinguish between medical ontologies and ontologies of different domains. Given different categories, we are more interested in comparing the performance of the PRIOR+ with those systems that fall in the same category of Type C, i.e., Falcon-AO (Qu, Hu *et al*. 2006; Hu, Zhao *et al*. 2007), TaxoMap (Zargayouna, Safar *et al*. 2007), AgreementMaker (Sunna and Cruz 2007), Lily (Wang and Xu 2007), X-SOM (Curino, Orsi *et al*. 2007) and DSSim (Nagy, Vargas-Vera *et al*. 2007).

The observations from Table 4.9 are:

1. The f-measure of the PRIOR+ is .592 in Testcase #1, .568 in Testcase #2, and .474 in Testcase #3.
2. Among 7 Type C systems, which are all generic ontology mapping systems that do not use any specific anatomy domain knowledge or external lexical information such

as the Unified Medical Language System (UMLS)[1], the f-measure of the PRIOR+ is 3rd highest compared to other systems over all three test cases.

3. The PRIOR+ is very efficient as well among all Type C systems. Compared to other systems that cost several hours or even 4 days to get final results, the PRIOR+ required only 23 minutes to finish the task, which is second only to Falcon-AO, i.e., 12 minutes in its case.

**Table 4.9** Comparison between the PRIOR+ and other participates on OAEI 2007 anatomy task[2]

| System | Type | Testcase #1 | | | | Testcase #2 | | Testcase #3 | | Recall+ | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Runtime | Prec | Rec | F-val | Prec | Rec | Prec | Rec | #1 | #3 |
| AOAS | A | n.a. | 0.928 | 0.804 | 0.861 | - | - | - | - | 0.505 | - |
| Sambo | A | 6 h | 0.845 | 0.786 | 0.815 | - | - | - | - | 0.580 | - |
| ASMOV | B | 15 h * | 0.803 | 0.701 | 0.749 | 0.870 | 0.696 | 0.739 | 0.705 | 0.270 | 0.284 |
| RiMOM | B | 4 h | 0.377 | 0.659 | 0.480 | - | - | - | - | 0.390 | - |
| *- Label Eq. -* | - | *3 min* | *0.987* | *0.605* | *0.750* | - | - | - | - | 0.0 | - |
| Falcon-AO | C | 12 min | 0.964 | 0.591 | 0.733 | 0.986 | 0.540 | 0.814 | 0.655 | 0.123 | 0.280 |
| TaxoMap | C | 5 h | 0.596 | 0.732 | 0.657 | 0.985 | 0.642 | - | - | 0.230 | - |
| AgreementM. | C | 30 min | 0.558 | 0.635 | 0.594 | 0.930 | 0.286 | 0.424 | 0.651 | 0.262 | 0.302 |
| Prior+ | C | 23 min | 0.594 | 0.590 | 0.592 | 0.663 | 0.497 | 0.371 | 0.657 | 0.338 | 0.426 |
| Lily | C | 4 days | 0.481 | 0.559 | 0.517 | 0.672 | 0.380 | 0.401 | 0.588 | 0.374 | 0.410 |
| X-SOM | C | 10 h | 0.916 | 0.248 | 0.390 | 0.942 | 0.104 | 0.783 | 0.565 | 0.008 | 0.079 |
| DSSim | C | 75 min | 0.208 | 0.187 | 0.197 | - | - | - | - | 0.067 | - |

## 4.8 SUMMARY AND CONCLUSION

This chapter proposed a new generic ontology mapping approach, PRIOR+. The approach uses both linguistic and structural information from ontologies, utilizes information retrieval techniques and artificial intelligence to solve ontology mapping problem. The major steps in the

[1] http://www.nlm.nih.gov/research/umls/

[2] From http://webrum.uni-mannheim.de/math/lski/align2007/results.html

approach include: similarity generation, adaptive similarity aggregation and neural network based constraint satisfaction. More specifically, the PRIOR+ first measures three similarities of ontologies. Next it estimates the harmony of each similarity upon its corresponding matrix. After that the PRIOR+ adaptively aggregates multiple similarities by weighting them using their harmonies. Finally the IAC neural network is selectively activated to find a global optimal solution that best satisfies ontology constraints. Final results are extracted using naïve extraction algorithm.

The experimental results on OAEI ontology matching benchmark tests, web directory test and anatomy test show:

1. The performance of three individual similarities, i.e. edit distance based similarity, profile similarity and structure similarity, varies. The edit distance based similarity is intuitive and works well when test ontologies are highly linguistic similar to the reference ontology. The structure similarity does not contribute much in most tests except those cases that have very little or no linguistic information at all. The profile similarity considers both the linguistic and structural information for ontologies and thus its overall performance outperforms the other two similarities.

2. The profile propagation does improve the performance of the PRIOR+ over all OAEI benchmark tests. Its impact is trivial when meaningful information of an element is adequate. However, its impact is significant when no or very limited meaningful information is available for the element itself, but such information can be found in it neighboring elements. Further, the weight difference between parent and child is minor. Therefore, to simplify the approach so as to improve its efficiency, this step can be omitted when generating profile similarity.

3. Harmony is a good measure to estimate the reliability of different similarities, especially on estimating the f-measure of edit distance based similarity of class and property and profile similarity of property, without a given ground truth.
4. The harmony-based adaptive aggregation method outperforms all other existing aggregation methods on OAEI benchmark tests.
5. Using the IAC neural network to solve constraint satisfaction problem in ontology mapping can dramatically improve the performance of mapping results.
6. The PRIOR+ is competitive with all top-ranked systems on benchmark tests, web directory test and anatomy test at OAEI campaign 2007. Notably it outperforms all systems on benchmark real cases and it is the 2$^{nd}$ place in web directory test.

Future work for this approach may include exploring constraints such as complex axioms in OWL, investigating which constraint is more useful than others, assign each constraint a different weight according to its priori probability and implementing the IAC neural network using Hadoop, etc.

# 5.0 THE NON-INSTANCE BASED LEARNING APPROACH

## 5.1 INTRODUCTION

Chapter 4.0 proposed a generic ontology mapping approach, the PRIOR+, based on profile propagation, information retrieval techniques and artificial intelligent model. This chapter proposes a non-instance learning based approach for solving ontology mapping problem.

Previous learning-based approaches have achieved high accuracy in prediction of correct mappings in the cases reported in GLUE (Doan, Madhaven *et al*. 2003). However GLUE has a limitation that it heavily relies on the availability of instance data when measuring the similarity of classes and attributes. Furthermore, the target of GLUE approach is every element in the target ontology. The specification of the learning target asks for new training data to rebuild the model when domain changes, and thus restricts the universality of the model.

To overcome the limitations, we aim to learn a generic mapping model, which does not require the existence of instances and domain constraints, by treating the ontology mapping problem as a binary classification problem. To learn a model, a variety of features that can reflect the characteristics of mapping pairs are generated, and then the SVM algorithm is applied. Experimental results show that our non-instance learning-based ontology mapping approach performs well in most of OAEI benchmark tests when training and testing on the same mapping

task; and the results of approach vary according to the likelihood of training data and testing data when training and testing on different mapping tasks.

## 5.2 ONTOLOGY MAPPING: AS A BINARY CLASSIFICATION PROBLEM

Binary classification is the task of classifying the members of given data into two groups[1]. In the context of ontology mapping, there are many ways to judge the quality of a mapping result. If we judge the mapping result by its correctness, i.e., either *correct* (i.e. +1) or *incorrect* (i.e., -1), then the ontology mapping problem can be represented as a binary classification problem as the following statement, where $e_{1i}$ is element $e_i$ from ontology $O_1$, $e_{2j}$ is element $e_j$ from ontology $O_2$ and $r$ is the mapping relation between $e_{1i}$ and $e_{2j}$.

$$m(e_{1i}, e_{2j}, r) \rightarrow \{+1, -1\}$$

For example, some sample mappings in the bibliographic ontologies described in §2.3, Figure 2.2, can be written as:

$m(Book_{right}, Book_{left}, =) \rightarrow \{+1\}$

$m(Proceedings, Proc., =) \rightarrow \{+1\}$

$m(Monograph, Monography, =) \rightarrow \{+1\}$

$m(Proceedings, Talks, =) \rightarrow \{-1\}$

$m(Proceedings, Monography, =) \rightarrow \{-1\}$

[1] http://en.wikipedia.org/wiki/Binary_classification

## 5.3 OVERVIEW OF THE NON-INSTANCE BASED LEARNING APPROACH

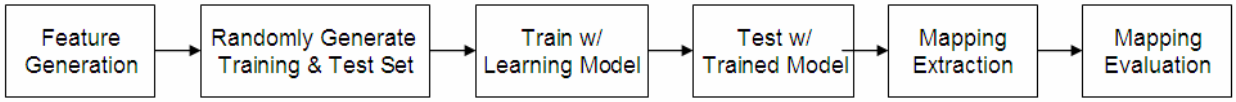



**Figure 5.1** The basic steps of the non-instance based learning approach

Figure 5.1 show the basic steps of the learning-based approach.

1. We generate various general features (i.e., linguistic, structural and web) to describe the characteristics of ontologies (e.g., OAEI benchmark tests).
2. We randomly generate training and testing set. That is, for each OAEI benchmark test, we generate all possible mapping pairs using the elements from both the reference ontology and the test ontology. Then we randomly pick 50% of the mapping pairs as a training set, the results of which will be marked down based on the ground truth, and use the other 50% as test set.
3. We train the training set using SVM model.
4. We classify the test data using the trained SVM model.
5. We extract final mapping results of test data using naïve descendant extraction algorithm (Meilicke and Stuckenschmidt 2007).
6. Finally we evaluate the test data against the reference alignment.

## 5.4 VARIETY OF FEATURES

Applying machine learning to the context of ontology mapping raises the question of what types of information should be used in the learning process. Many different types of information can contribute toward deciding the correspondence of a mapping pair. Two principles are followed in the process of feature selection:

1. The feature should not be limited to instances. It could be generated from a class, a relation and/or an instance in ontologies.
2. The feature should be general enough and domain independent so that the model could be generalized to other applications regardless of the domain.

In the approach, 3 categories, i.e., linguistic features, structural features and web features, and a total of 23 features are generated for each mapping pair.

### 5.4.1 Linguistic Features

Linguistic features are selected according to the principle described in (Jones, Rey *et al.* 2006). Table 5.1 lists 16 linguistic features that are used in our training. The features can be divided into two types:

1. Isolated characteristics of elements in mapping pair, e.g. length of elements, number of tokens, etc.
2. Syntactic characteristics of mapping pair, e.g. Levensthtein edit distance between two elements, number of common tokens in the pair, etc.

**Table 5.1** Linguistic features

| Features | Description |
|---|---|
| $Le_1$ | the length of $e_1$ |
| $Le_2$ | the length of $e_2$ |
| ldiff | the length difference between $e_1$ and $e_2$ = $le_1$ - $le_2$ |
| absldiff | the absolute value of length difference = abs(ldiff) |
| ldiffn | the length difference normalized by the length of $e_1$ = ldiff/$le_i$ |
| lengthratio | the absolute value of ldiffn = absldiff/$le_i$ |
| ntoke1 | the number of tokens in $e_1$ |
| commonw | the number of tokens in common in $e_1$ and $e_2$ |
| editdist | the normalized Levenshtein edit distance of $e_1$ and $e_2$ |
| worddist | the proportion of word changed from $e_1$ to $e_2$ |
| word_pov | the proportion of tokens in common at beginning of $e_1$ |
| word_suf | the proportion of tokens in common at the end of $e_1$ |
| char_pov | the proportion of characters (utf8 bytes) in common at the beginning of $e_1$ |
| char_suf | the proportion of characters in common at the end of $e_1$ |
| digit_dropped | a boolean value, identifying whether a digit has been dropped from $e_1$ to $e_2$ |
| profsim | the cosine similarity of the profile of $e_1$ and that of $e_2$, the profile is the combination of all linguistic information (e.g. name, label and comments) of an element |

### 5.4.2 Web Features

Edit distance, which defines the strings similarity by the minimum number of insertions, deletions and substitutions that require transforming one string into the other, is a commonly used method to calculate the similarity of terms (Bouquet, Euzenat *et al*. 2004). However edit distance encounters trouble where synonyms exist. For synonyms, the intuitive way is to check auxiliary information in a thesaurus, e.g. WordNet. However WordNet suffers some drawbacks

as well. First, semantic similarity between words changes across domains. Even though a thesaurus may contain a sufficiently wide range of common words[1], sometimes it does not cover special domain vocabulary. For example, though *apple* is frequently associated with *computers* on the Web, this sense of *apple* is not listed in WordNet. Second, new words are continually created and new senses are assigned to existing words. Thesauri usually can not capture these new words and senses in time. As an alternative, the Web has a huge amount of information and the newest words/senses are stored. To overcome the problem of edit distance comparison and WordNet sense limitation, this section proposes to generate a new Web feature, i.e., WebDice coefficient (Bollegala, Matsuo *et al*. 2007).

Webdice coefficient is a page count based co-occurrence measure (Bollegala, Matsuo *et al*. 2007). Page count of a query is the number of pages that contain the query terms obtained from a Web search engine such as Yahoo. Page count can be considered as a global measure of co-occurrence of query terms. For example, the page count of the query "*spring*" AND "*last name*" is 1,550,000 and the page count of the query "*spring*" AND "*season*" is 57,300,000 in Google[2]. The about 40 times more numerous page counts for "*spring*" AND "*season*" indicate that "*spring*" is more semantically similar to "*season*" than "*last name*". Using page count *alone* to measure the co-occurrence of two terms does not always accurately express semantic similarity. This is because page count ignores the position of a word appearing in a page, and thus may incorrectly count pages, in which both words appear but far away from each other and without any relevance. To overcome the drawback, one must consider the page counts not just

[1] As of 2006, WordNet contains about 150,000 words organized in over 115,000 synsets for a total of 207,000 word-sense pairs. (From http://en.wikipedia.org/wiki/Wordnet)

[2] Search http://www.google.com on July, 2007.

for query *X AND Y* but also for the individual words *X* and *Y* to access semantic similarity between *X* and *Y*.

Bollegala, Matsuo *et al*. proposed to exploit a modified co-occurrence measure, *WebDice*, to compute semantic similarity using page counts (Bollegala, Matsuo *et al*. 2007). WebDice coefficient is a variant of Dice coefficient (Rijsbergen 1979). For sets *X* and *Y* of keywords used in information retrieval, the Dice coefficient may be defined as:

$$Dice(X,Y) = \frac{2 \mid X \cap Y \mid}{\mid X \mid + \mid Y \mid} \qquad \textbf{(16)}$$

If the notation *H(X)* and *H(Y)* denote the page counts for query *X* and *Y* respectively in a search engine, and *H(X∩Y)* denotes the page counts for the conjunction query *X AND Y*, then the WebDice coefficient can be modified as:

$$WebDice(X,Y) = \begin{cases} 0 & \text{if } H(X \cap Y) \le c \\ \dfrac{2H(X \cap Y)}{H(X) + H(Y)} & \textit{otherwise} \end{cases} \qquad \textbf{(17)}$$

where *c* is a predefined threshold (e.g. *c*=5) to reduce the adverse effects caused by random co-occurrences. Because of the scale and noise in Web data it is possible that a page contains two words purely accidentally.

The WebDice coefficient has been demonstrated to outperform the other three modified co-occurrences (i.e. *WebJaccard*, *WebOverlap*, and *WebPMI*) in (Bollegala, Matsuo *et al*. 2007).

### 5.4.3 Structural Features

Structural information is important in estimating the similarity of classes of ontologies. Table 5.2 list the structural features of a mapping hypothesis between classes (e.g., $C_1$ and $C_2$) and properties (e.g., $p_{1i}$ and $p_{2j}$).

**Table 5.2** Structural features

| Elements | Features | Description |
|---|---|---|
| Classes | DirPropNumDiff | The normalized difference between the numbers of the classes' direct properties |
| | DirPropSim | The edit distance based similarity between the classes' direct properties, i.e., $\text{DirPropSim} = \text{Avg}_i(\max_j(EditDistSim(p_{1i}, p_{2j})))$, where $p_{1i}$ and $p_{2j}$ are direct properties of class $C_1$ and $C_2$. |
| | chNumDiff | The normalized difference between the numbers of the classes' subclasses. |
| | chSim | The edit distance based similarity between the classes' subclasses, i.e., $\text{chSim} = \text{Avg}_i(\max_j(EditDistSim(subC_{1i}, subC_{2j})))$, where $subC_{1i}$ and $subC_{2j}$ are subclasses of class $C_1$ and $C_2$. |
| | paSim | The edit distance based similarity between the classes' super classes, i.e., $\text{paSim} = \text{Avg}_i(\max_j(EditDistSim(paC_{1i}, paC_{2j})))$, where $paC_{1i}$ and $paC_{2j}$ are super classes of class $C_1$ and $C_2$. |
| | depDiff | The normalized difference between the depth to root of the classes |
| Properties | domainSim | The edit distance based similarity between the properties' domain |
| | rangeSim | The edit distance based similarity between the properties' range |
| | motherSim | The edit distance based similarity between the properties' mother class |

## 5.5 EVALUATION

### 5.5.1 Test Ontologies

The test ontologies are OAEI benchmark tests, which have been introduced in §4.7.1.

### 5.5.2 Evaluation Criteria

Same evaluation criteria, i.e., *precision*, *recall* and *f-measure*, as described in §4.7.2, are used.

### 5.5.3 Methodology and Results

Two experiments were designed to evaluate the learning-based approach. The 1st experiment investigates how the approach performs in the situation where people have manually marked *some* mapping results for a specific mapping task, but they need help from automatic mapping tools to find the rest of mappings. The 2nd experiment investigates how the approach performs in an alternative situation where no manual mapping results are available for a specific mapping task, but a general model has been learned that can be used to find mappings.

#### 5.5.3.1 1st Experiment – Within-task

The 1st experiment investigates if machine learning methods work well in the situation where some manually marked mapping results are available for a specific mapping task. The motivation of the 1st experiment comes from the real world case: A user is working on an ontology mapping task with the help of computers. Since the size and complexity of the mapping task are large, the

user can not manually find all mapping pairs. But fortunately the user is able to mark some of them (e.g., $e_{1i}$ maps to $e_{2j}$). Therefore, if we can utilize users' previous effort, we can help them find the rest of mappings by using some machine learning techniques.

The methodology of the 1st experiment is:

1. For each OAEI benchmark test, we generate candidate mapping pairs by simply combine all elements from two ontologies.
2. For each candidate mapping pair, we mark down their correctness according to the reference alignment (i.e. the ground truth). Simultaneously we generate various features (i.e., linguistic, structural and web) to describe the characteristics of the mapping pair.
3. To simulate real world situation, we split all mapping pairs into two groups (i.e., one is for training purpose and the other is used as testing set) by randomly choosing (e.g. 50% vs. 50%).
4. We train two SVM models (i.e., SVM-Class and SVM-Property) on training set with the help of SVM-Light package[1].
5. We classify testing data on two models.
6. Finally we extract mapping results of test data using naïve descendant extraction algorithm (Meilicke and Stuckenschmidt 2007) and evaluate the results against the reference alignment.
7. To eliminate the bias caused by randomly choosing mapping pairs to generate training and testing data, we repeat step 3-6 10 times and show the average result.

[1] http://svmlight.joachims.org/

In the experiment, *classes* and *properties* are separated due to the difference between their structural features. That is, when training, two SVM models are trained – *SVM-Class* model for classes and *SVM-Property* model for properties. When testing, mapping pairs with *class* elements are tested on *SVM-Class* model and mapping pairs with *property* elements are tested on *SVM-Property* model. Moreover, since the number of negative examples is much larger than the number of positive examples in training data, we use a fixed cost factor (i.e. 10) in SVM-Light to equalize the distribution and ensure training errors on positive examples outweigh those on negative examples. Note, we add an annotation *within-task* to the name of the approach, referring that training and testing are done on the same ontology mapping task each time.

Figure 5.2 shows the f-measure of *classes* on *SVM-Class* model on OAEI benchmark tests. Figure 5.3 shows the f-measure of *properties* on *SVM-Property* model on OAEI benchmark tests, in which the f-measures of benchmark tests #226, #233-#237, #240-#247, #250, #254-#257, #260-#266 are 0 due to the fact that no property exists on those tests. Both pictures include the f-measure of *classes*/*properties* running by the PRIOR+ approach for comparison purpose.

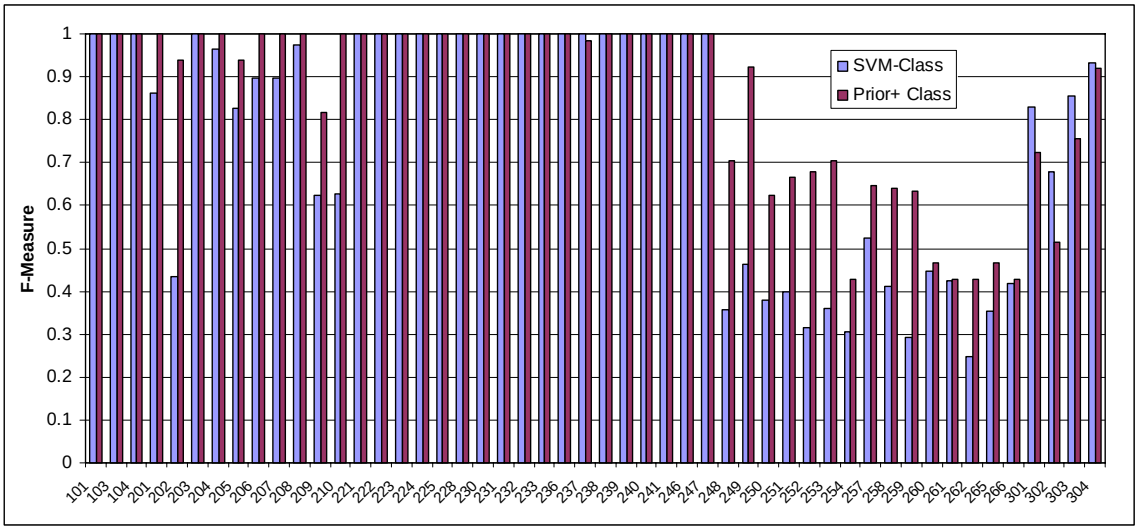



**Figure 5.2.** Results of *classes* on *SVM-Class* model on all benchmark tests (Within-task)

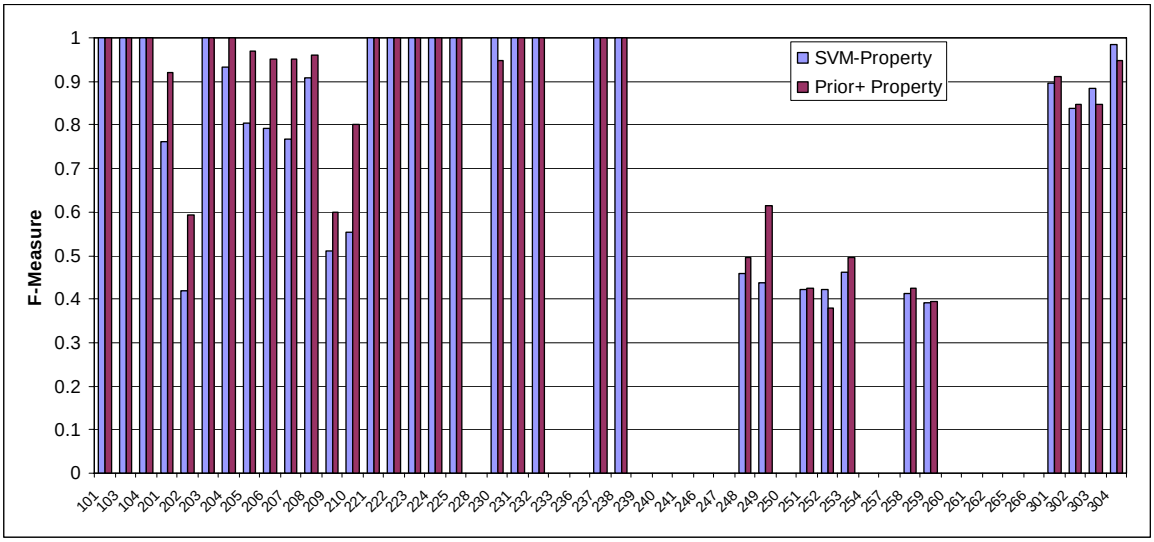



**Figure 5.3.** Results of *properties* on *SVM-Property* model on all benchmark tests (Within-task)

The observations from Figure 5.2 and Figure 5.3 are:

1. On Test #101-#104 and #221-#247, both SVM-Class model and SVM-Property model perform as well as PRIOR+. This is because the linguistic information of these test ontologies is highly similar with that of the reference ontology and there is much less interference such as randomly generated name of classes/properties. Thus it is easy for both SVM models catch useful features such as the *editdist* feature in Table 5.1.
2. On Test #201-#210, both SVM-Class and SVM-Property model perform relatively worse than the PRIOR+ (especially on #201, #202, #208, #209). This is because the linguistic information changes too much on these tests so that it is hard to catch its linguistic and web characteristics in the training model. Meanwhile the structural feature is relatively weak.
3. On Test #248-#266, both SVM-Class and SVM-Property model perform much worse than the PRIOR+. This is because there is no name and no comments in the test

ontologies at all, i.e., both linguistic features and web features are totally unavailable. The only feature available for SVM models is structural, which is relatively weak. Meanwhile, the PRIOR+ benefits from the profile enrichment process that integrates instance information, which keeps all descriptive information, to both classes and properties.

4. On real world cases #301-304, the SVM-Class model performs much better than the PRIOR+ and the SVM-Property model performs similarly as the PRIOR+ (i.e., slightly better on #301 and #302 but slightly worse on #303 and #304). The reason why the performance of SVM models on real world cases is better than PRIOR+ might be because the learning based approach utilizes Web feature to explore synonymous relations between concepts in ontologies. By contrast the PRIOR+ approach does not integrate any auxiliary thesaurus for such a purpose.

Furthermore, we investigate the distribution of the precision and recall of the *SVM-Class* model and *SVM-Property* over all benchmark tests. The results in Figure 5.4 and Figure 5.5 show that the precision and recall of test group (i.e., #1xx, #221-247) on *SVM-Class* model and *SVM-Property* model are close to each other. The precision and recall of #202, #209 and #210 are different from other tests in the group of #201-#210. This is because though the linguistic difference exists in all test ontologies in the group, the unavailability of the linguistic information on the comments of #202, #209 and #210 is more severe and thus be anomalous with others. The differences between the classes of #248-#266 and #3xx result in different precision and recall on *SVM-Class* model. Such differential is smaller on *SVM-Property* model due to the smaller differential between their properties.

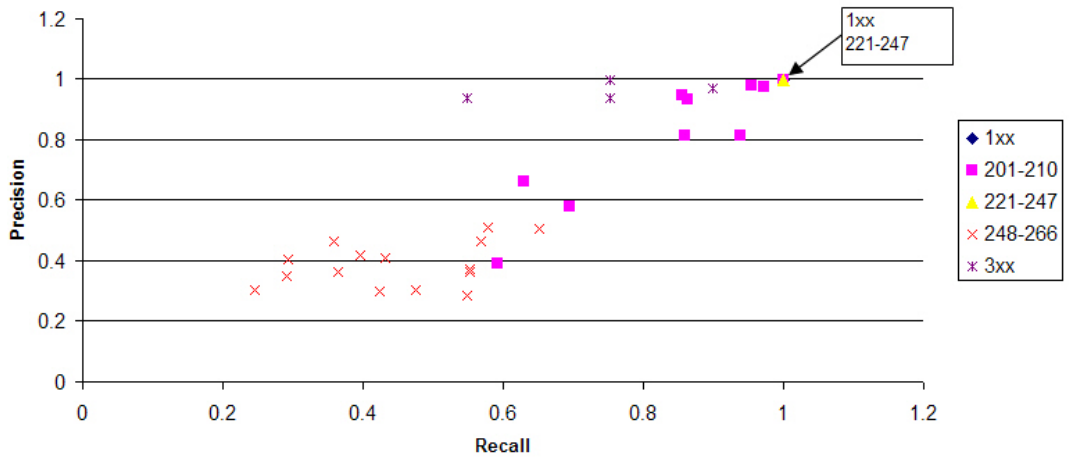



**Figure 5.4 The precision-recall graph of *SVM-Class* model over all benchmark tests**

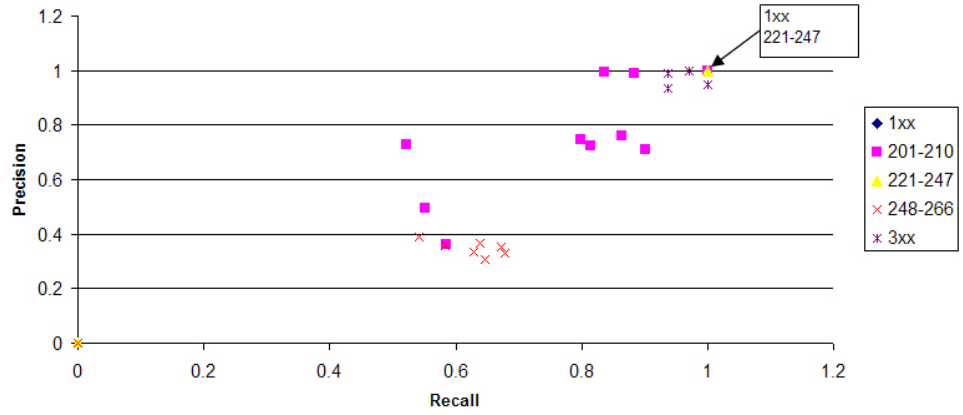



**Figure 5.5 The precision-recall graph of *SVM-Property* model over all benchmark tests**

### 5.5.3.2 2nd Experiment – Cross-task

The 2nd experiment investigates whether a model trained on one mapping task can work on another mapping task(s). Moreover, we are interested in which benchmark test(s) are more suitable as training models. The motivation for the 2nd experiment is based on the fact that in most ontology mapping cases, no ground truth is available. However we may have a model, which is already trained on another ontology mapping task. Thus, to save users' time and effort, we would like to find out mapping results using the existing model.

The methodology of the 2nd experiment is:

1. For each OAEI benchmark test, we generate candidate mapping pairs by combining all elements from two ontologies.
2. For each candidate mapping pair, we mark down their correctness according to the reference alignment (i.e. the ground truth). Simultaneously we generate various features (i.e., linguistic, structural and web) to describe the characteristics of the mapping pair.
3. We train two SVM models (i.e. SVM-Class and SVM-Property, same meanings as described in §5.5.3.1) on a benchmark mapping task (e.g. #101) with the help of SVM-Light package[1]. Please note that we do not train any model on benchmark tests #228, #233, #236, #239-#247, #250, #254, #257, and #260-#266 because we will not use their model to test other benchmark tasks. The reason why we do not test on these models is no properties exist in the test ontology of these tasks, and thus no SVM-Property model can be trained on these mapping tasks. Whereas other benchmark tests have many properties in their test ontologies. Therefore it does not make any sense for us to test mapping tasks, which have both classes and properties, on the model, which is trained with classes only.
4. We classify testing data of all other benchmark tests (excluding the one that has been used in training model) using the SVM models.
5. Finally we extract mapping results of test data using naïve descendant extraction algorithm (Meilicke and Stuckenschmidt 2007) and evaluate the results against the reference alignment.

[1] http://svmlight.joachims.org/

6. We repeat step 3-5. That is, we train two SVM models on each benchmark test (excluding #228, #233, #236, #239-#247, #250, #254, #257, and #260-#266) and testing all the left benchmark tests on the models.
7. Finally, we report the average f-measure of a group of testing data (e.g., #1xx, #2xx, #3xx etc.) on each training model as our final result.

Note, to distinguish the approach from that used in the 1st experiment, we add an annotation *cross-task* to the name of the approach. The *cross-task* refers to training and testing are done on different mapping tasks each time.

Figure 5.6 to Figure 5.12 show the average f-measure tested on different data sets (i.e., all benchmark tests, #1xx, #2xx, #3xx, and more specific #201-#210, #221-#238, #248-#259). The result of the PRIOR+ approach is included in each picture for comparison purpose.

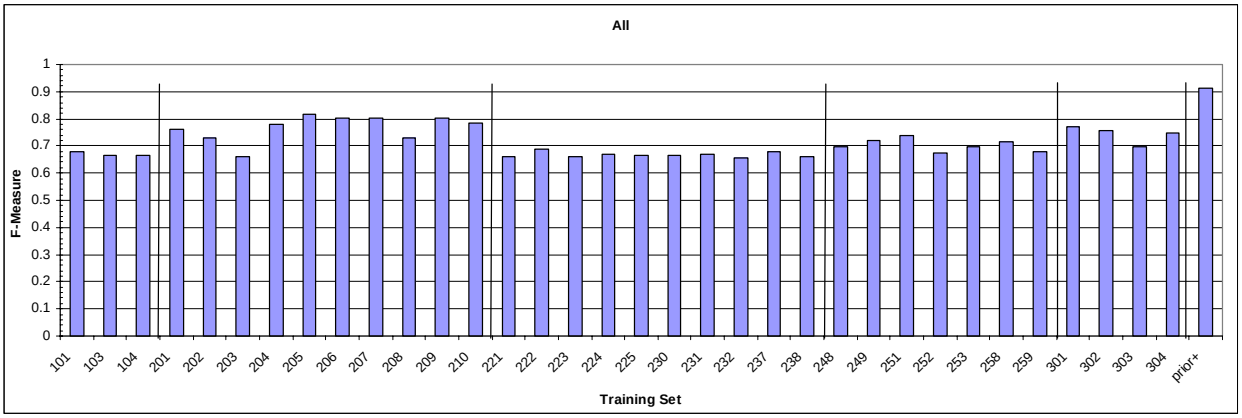



**Figure 5.6.** Testing results on all benchmark tests (Cross-task)

Figure 5.6 is the f-measure over all benchmark tests when training on each benchmark test. The observations from Figure 5.6 are:

1. Generally speaking, the PRIOR+ outperforms the learning-based approach (i.e., the cross-task) over all benchmark tests. The PRIOR+ holds the highest f-measure .92.

Whereas no f-measure of any learning-based models is higher than .82. This suggests that none of OAEI benchmark tests is the best to be chosen to train the model that will then be used by all other benchmark tests to predict their mapping results.

2. More specifically, training on #101-#104 and #221-#238 but testing on all other benchmark tests obtains worse results than training on other benchmark tests. This is because benchmark tests #1xx and #221-#238 are easy ontology mapping tasks. They have highly similar linguistic features, which leads the machine learning model to rely on some simple but specific features too much (e.g. the *editdist* feature), and thus it does not perform well when linguistic similarity changes a lot on other benchmark tests. For example, many benchmark tests have removed name and comments so that there is no linguistic information available at all. In this case, testing such data set on the model trained on #1xx and #221-#238 will hurt due to the significant difference between the characteristics of two mapping tasks.
3. Training on #201-#210 and #301-#304, the f-measure is better than training on #101-#104 and #221-#238 due to the balance between different kinds of features in the training data.
4. Finally the f-measure when training on #248-#259 is better than training on #1xx and #221-#238 but worse than training on #201-210 and #3xx. This is because #1xx and #221-#238 mapping tasks are too easy and simple to build a training model; meanwhile, #248-#259 have very limited linguistic information in themselves, which is unlike the diverse characteristics existing in #201-#210 and #3xx.


**Figure 5.7** Testing results on benchmark tests #1xx (Cross-task)

Figure 5.7 is the f-measure over Test #101-104 (i.e., 1xx) when training on each benchmark test. The observation from Figure 5.7 is:

The f-measure when training on different benchmark tests but testing on #1xx is good except on #248-#259. The overall good performance on 1xx is because their simple characteristics are easy to catch in almost all training sets.


**Figure 5.8.** Testing results on benchmark tests #2xx (Cross-task)

Figure 5.8 is the f-measure over Test #201-259 (i.e., 2xx) when training on each benchmark test. Please see more analysis on separate tests, i.e., #201-#210, #221-#238, #248-#259, followed by Figure 5.9, Figure 5.10 and Figure 5.11.

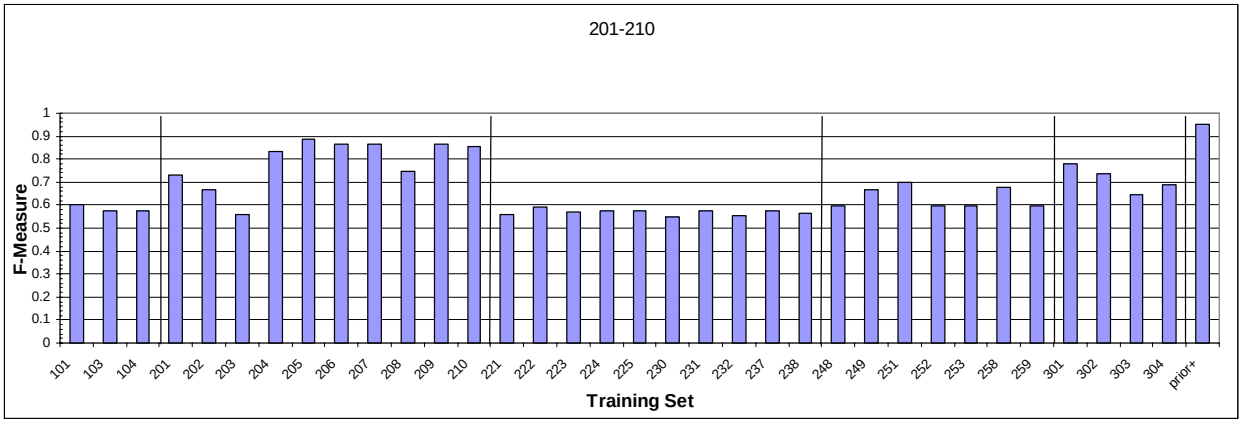



**Figure 5.9** Testing results on benchmark tests #201-#210 (Cross-task)

Figure 5.9 is the f-measure over Test #201-210 when training on each benchmark test. The observations from Figure 5.9 are:

1. Like the results over all benchmark tests shown in Figure 5.6, training on #101-#104 and #221-#238 obtains worse results than training on other benchmark tests because all these tests have highly similar linguistic features, which leads the machine learning model to rely on some simple but specific features too much (e.g. the *editdist* feature), and thus it does not perform well when linguistic similarity changes a lot on other benchmark tests.
2. Training on #201-#210 (except #203) and testing on themselves get the best f-measure compare to training on all other benchmark tests. We believe it is because the training data carry the same characteristics as the testing data most. However the

performance of #203 is different to #201-#210 but similar to #1xx and #221-238 is because though no comments exist in #203 no change has been made to any name of its classes and properties or to its structure. Such characteristic makes #203 highly similar to #1xx and #221-#238 from both linguistic view and structural view.

3. The f-measure when training on #248-#259 is better than the f-measure when training on #1xx and #221-#238 but worse than the f-measure when training on #201-210 and #3xx. This is because, on the one hand, #1xx and #221-#238 mapping tasks are too easy and too simple to build a training model; on the other hand, #248-#259 have very limited linguistic information in themselves, which is unlike the diverse characteristics existing in #201-#210 and #3xx.
4. Finally, training on #301-#304, the f-measure is better than the f-measure training on all other benchmark tests except #201-210 themselves. This is because the diversity and balance of different kinds of features existing in #3xx.


**Figure 5.10** Testing results on benchmark tests #221-#247 (Cross-task)

Figure 5.10 is the f-measure over Test #221-#247 when training on each benchmark test. The observation from Figure 5.10 is similar as that of Figure 5.7:

The f-measure when training on different benchmark tests is good except for #248-#259. The overall good performance on #221-#247 is because their simple characteristics are easy to catch in almost all training sets.

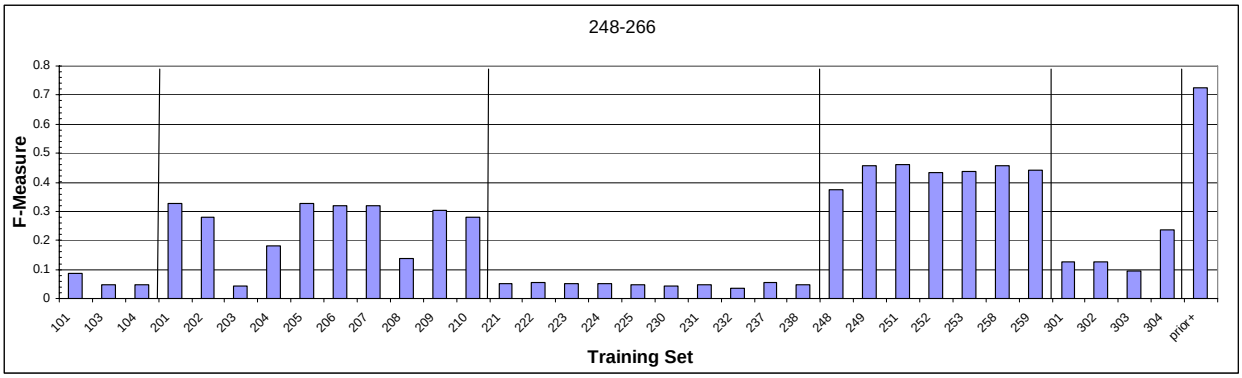



**Figure 5.11** Testing results on benchmark tests #248-#266 (Cross-task)

Figure 5.11 is the f-measure over Test #248-#259 when training on each benchmark test. The observations from Figure 5.11 are:

1. Generally speaking, the f-measure when training on #1xx, #203, #221-#238, and #3xx but testing on #248-#259 is poor. This is because all these benchmark tests include more or less linguistic information. Compare to #248-#259, these benchmark tests are relatively simple and easy. Thus, training on relatively simple and intuitive mapping tasks will result in relatively simple and intuitive model, which is not suitable to complex situations.
2. Whereas the f-measure when training on #201-#210 (except #203) and #248-#259 themselves is better than the f-measure when training on other benchmark tests due to the similarity between the training data and testing data.


**Figure 5.12** Testing results on benchmark tests #3xx (Cross-task)

Figure 5.12 is the f-measure over Test #301-#304 when training on each benchmark test. The observations from Figure 5.12 are:

1. Generally speaking, the f-measure when training on #3xx themselves is the best compare to training on all other benchmark tests and the PRIOR+ approach except that the f-measure on #303 is slightly worse than on the PRIOR+.
2. Training on #203 and #248-#259 obtained worst results in this case. This is because though #203 and #248-#259 are difficult mapping tasks their *difficulties* are manually made and are very different from what we can meet in real world cases. Therefore the model trained on these tests can not contribute to the real world cases.

## 5.6 SUMMARY AND CONCLUSION

In this chapter, we examined a non-instance learning-based ontology mapping approach, which overcomes the limitations of previous learning-based ontology mapping approaches that either rely on the availability of sufficient instances or are domain-dependent.

In the approach we treated the ontology mapping problem as a binary classification problem; generated a serial of domain-independent and task-independent features; utilized these features to build training model; and conducted two experiments to investigate the performance of machine learning techniques in different situations.

The experimental results show:

1. For learning-based within-task approach, the performance is good when mapping task is relatively easy (i.e., #1xx and #221-247). When mapping task is more difficult, its performance is not as good as the PRIOR+ approach (i.e., #201-#210 and #248-#266). But the performance of this approach is better than the PRIOR+ on real world cases, which shows the features used in this approach do make more sense on real world cases than on artificially constructed cases.
2. For learning-based cross-task approach, the performance is good when training data and testing data share similar characteristics. If the testing mapping task is very simple, it's easy to catch characteristics in the training model and thus get good performance with more difficult training task. Meanwhile if training task and testing task both are difficult but with different characteristics, the performance is not as good as other approaches.

# 6.0 CONCLUSION AND FUTURE WORK

We have presented two approaches, i.e., the PRIOR+ and the learning-based approach, for ontology mapping using information retrieval, artificial intelligence and machine learning techniques to analyze both linguistic and structural information, as well as web information, for two ontologies. The experiment results show that the PRIOR+ is a generic and scalable ontology mapping approach. It is competitive with all top-ranked systems on benchmark tests, web directory test and anatomy test at OAEI ontology matching campaign 2007. Moreover, the learning-based approach demonstrates the potential of applying machine learning techniques in ontology mapping without using any instance information. It also shows the potential to improve performance in real world ontology mapping tasks. In this chapter, the main contributions of this thesis are outlined and a number of directions of future work are presented.

## 6.1 SUMMARY OF CONTRIBUTIONS

As stated in §1.4, the ultimate goal of our research is *to solve the problem of ontology mapping, and thus enable semantic interoperability between different web applications and services in the WWW*. More specific, we aim *to develop a new generic approach to automatically map ontologies with minimum human effort.*

Regarding our research objective, the main contribution of our work is: We designed and developed two generic ontology mapping approaches to find the semantic correspondences between elements in different ontologies. Each of the approach is suitable to some specific mapping situation. More specific, our contributions can be divided into twofold:

1. We proposed a generic ontology mapping approaches, the PRIOR+, which integrates information retrieval and artificial intelligence techniques in the context of ontology mapping. Some highlights in the PRIOR+ are:
    1) We explored three different measures of similarity such as edit distance based similarity, profile similarity and structural similarity to support ontology mapping.
    2) We treated the ontology mapping problem as an information retrieval task, and thus utilized information retrieval techniques to estimate the profile similarity in a vector space model.
    3) We proposed a harmony measure to estimate the reliability of different similarities without given ground truth.
    4) We proposed a harmony based adaptive aggregation method to aggregate various similarities.
    5) We integrated the interactive activation and competition (IAC) neural network in the context of ontology mapping to search for a global optimal solution that best satisfies ontology constraints.
    6) We implemented the parallel computing platform, Hadoop, and the search engine, Indri, to support large-scale ontology mapping tasks.
    7) We conducted a series of experiments to evaluate the PRIOR+ from different perspectives and analyzed the results.

2. We proposed a non-instance based machine learning approach for ontology mapping problem. Some highlights in the learning-based approach are:
    1) We treated the ontology mapping problem as a binary classification problem.
    2) We extracted various features, i.e., linguistic, web and structural features, to describe ontologies. All the features are general and domain-independent.
    3) We explored machine leaning algorithms (i.e., SVM) for ontology mapping without requesting any instance information.
    4) We conducted a series of experiments to evaluate the learning-based approach from different perspectives and analyzed the results.

## 6.2 FUTURE DIRECTIONS

Given our findings, there is still much work to be done. For example, to extend 1-to-1 mapping to 1-to-n or even m-to-n mappings, we need to improve the mapping extraction algorithm and adjust constraints when implementing the neural network based constraint satisfaction solver so that they could allow 1-to-n or m-to-n mappings between ontologies. Currently we only work on finding equivalent relations between classes and properties in ontologies. Finding more complex relations such as broader and narrower will be more challenging. Some preliminary work of ours on exploring complex relationship between elements in ontologies have been done in the food track mapping task at OAEI ontology matching campaign 2006 (Mao and Peng 2006). In that task, we converted equivalent relations to broader and narrower relations by identifying the hierarchical relationship between classes in two ontologies. Future work in this direction could integrate an external repository to analyze the hierarchical relationship between classes, which

are not explicitly expressed in the two ontologies. Social tagging systems might also contribute to finding mappings between two ontologies such as the Google web directory and Yahoo web directory when they have been associated with many instances like web pages and the web pages have been tagged as well. Such tagging information can be used in our training model or integrated into the profile of the PRIOR+. Other future work includes the improvement of the PRIOR+ approach, especially on the IAC neural network, and the improvement of the learning-based ontology mapping approach.

The PRIOR+ approach has been demonstrated to be a good generic ontology mapping approach on different mapping tasks. The PRIOR+ integrates the IAC neural network to search for a globally optimized solution that can satisfy as many ontology constraints as possible. Currently the constraints implemented in the experiment are still limited. It may be possible to integrate more constraints such as complex axioms in OWL. Further, we should investigate which constraint is more useful than others among all constraints. As stated in §4.5.4, we set the same weight for both positive constraints (i.e., 1) and negative constraints (i.e., -1), which is practical but not rationale. This is because, in an ideal situation, the weight of different constraints should reflect their prior probabilities, i.e., they should be a function of their confidence. Finally, we are considering to implement the IAC neural network on the parallel computing platform, i.e., Hadoop, so as to improve the efficiency of the approach when the size and complexity of ontology constraints is large.

Related to the learning-based approach, several things might be done. First, to leverage different features so as to achieve a robust semantic similarity measure, a forward-backward procedure should be carried out to reduce the number of features. That is, in the forward step, the best additional feature will be added one by one. Afterward, a backward process will be executed

on all data to drop features that are not significant. The criterion of feature selection is to maximize the f-measure. Secondly, we can build a tool to support ontology mapping evaluation using the learning-based approach. Taking the manually evaluated mapping pairs as training set, we can train an SVM model. Then we can test candidate mapping pairs against the model, rank them by confidence score returned by the model, and show the ranked list to the user so that the user can judge the correctness of the mapping results. While not fully automatic, the user would not have to go through all of the ontologies to look for mappings or randomly pick a mapping pair to evaluate for correctness. Instead, the user will be able to focuses on the top-ranked pairs only because they have more chances to be correct than others. By this means, the user saves time and effort. Furthermore, along with the new judgment from the user side, the training and testing could be iteratively done in the background. Then the ranked list can be updated with new models. Third, we could perform an active learning with Support Vector Machine (Lewis and Gale 1994). That is, instead of using a randomly selected training set, we actively choose training instances (e.g. the mapping results that have the most ambiguities) and request user feedback about them to construct a more accurate classifier. Finally, future work on the non-instance based learning approach may include using the harmony as the golden standard to train models. As we have demonstrated harmony is a good estimator of the performance of different similarities. When we train a learning model, the harmony, like the golden standard, gives us some sense of the quality of the model. Therefore, if we adjust a training model based on its harmony, we do not rely on the availability of the golden standard, which is a big bottle neck for learning based approaches. Even more, we can integrate the result of the learning based approach to the result of the PRIOR+ approach using the harmony measure so that we can take advantage of both approaches.

## RELATED PUBLICATIONS

1. Ming Mao, Yefei Peng and Michael Spring. "Neural Network based Constraint Satisfaction in Ontology Mapping". In the Proceeding of 23rd AAAI Conference on Artificial Intelligence (AAAI-08) (to appear)

   **Abstract:** Ontology mapping seeks to find semantic correspondences between similar elements of different ontologies. Ontology mapping is critical to achieve semantic interoperability in the WWW. Due to the fact that ubiquitous constraints (e.g., hierarchical restrictions in RDFS) caused by the characteristics of ontologies and their representations exist in ontologies, constraints satisfaction has become an intriguing research problem in ontology mapping area. Though different techniques have been examined to find mappings, little work is made to solve constraint satisfaction problem for ontology mapping. Currently most approaches simply validate ontology constraints using isolate heuristic rules instead of comprehensively considering them in a global view. This paper proposes a neural network based approach to search for a global optimal solution that can satisfy ontology constraints as many as possible. Experimental results on OAEI benchmark tests #248-#266 show the approach is promising. It dramatically improves the performance of preliminary mapping results.

2. Ming Mao, Yefei Peng and Michael Spring. "A Harmony based Adaptive Ontology Mapping Approach". In the Proceeding of 2008 International Conference on Semantic Web and Web Services Conference (SWWS'08) (to appear)

   **Abstract:** Ontology mapping seeks to find semantic correspondences between similar elements of different ontologies. Ontology mapping is critical to achieve semantic interoperability in the WWW. Nowadays most ontology mapping approaches integrate multiple individual matchers to explore both

linguistic and structure similarity of different ontologies. Thus how to effectively aggregating different similarities is pervasive in ontology mapping. In current aggregation methods, people either have to manually set parameters in aggregation function or need "ground truth" in advance for machine learning based parameter optimization. Both of them have limitation. In this paper, we propose a measure harmony, which is the normalized number of mapping pair that suggests an unambiguous one-to-one mapping, and a harmony based adaptive ontology mapping approach, which can automatically adjust parameters of three kinds of similarities (i.e., edit distance based similarity, profile similarity and structure similarity) in aggregation functions according to different mapping tasks without given any ground truth. Experimental results show the harmony is indeed a good estimator of the performance (i.e., f-measure) of different similarities, and the harmony based adaptive aggregation method outperforms all other existing aggregation methods on OAEI benchmark tests.

3. Ming Mao, Yefei Peng and Michael Spring. "Integrating the IAC Neural Network in Ontology Mapping". In the Proceeding of 17th International World Wide Web Conference (WWW 2008) (poster) (to appear)

   **Abstract:** Ontology mapping seeks to find semantic correspondences between similar elements of different ontologies. Though nowadays ontology mapping approaches have made significant progresses in predicting mappings, most of them simply validate ontology constraints using some isolate heuristic rules instead of considering them in a global view. This paper proposes a neural network based approach to search for a global optimal solution that satisfies ontology constraints as many as possible. Experimental results on OAEI benchmark tests show the approach is promising. It dramatically improves the performance of preliminary mapping results.

4. Ming Mao. "Ontology Mapping: An Information Retrieval and Interactive Activation Network Based Approach". In K. Aberer et al. (Eds.): the Proceedings of 6th International Semantic Web Conference (ISWC/ASWC 2007), LNCS 4825, pp. 931–935, 2007

   **Abstract:** Ontology mapping is to find semantic correspondences between similar elements of different ontologies. It is critical to achieve semantic interoperability in the WWW. This paper proposes a new

generic and scalable ontology mapping approach based on propagation theory, information retrieval technique and artificial intelligence model. The approach utilizes both linguistic and structural information, measures the similarity of different elements of ontologies in a vector space model, and deals with constraints using the interactive activation network. The results of pilot study, the PRIOR, are promising and scalable.

5. Ming Mao, Yefei Peng and Michael Spring. "A Profile Propagation and Information Retrieval Based Ontology Mapping Approach". In the Proceedings of 3rd International Conference on Semantics, Knowledge and Grid (research track), Xi'an, China, 2007

   **Abstract:** Ontology has been used widely to help finding relevant information among distributed and heterogeneous sources. Given that no universal ontology exists for the WWW, ontology mapping attracts many researchers' interest in various areas. In this paper we propose a new generic ontology mapping approach based on profile propagation and information retrieval techniques. The features used to establish the profile of a concept include all lexical information (i.e., its name, label, comments, property restriction, etc.). Profile propagation then is used to integrate structural information by adding the profiles of its ancestors, descendants and siblings to the profile of a concept, with different weights. Afterwards profiles are compared in a vector space model and the most relevant one will be returned as mapping results. A search engine is integrated as profile mapper when the size of ontologies is large. Experimental results show that the proposed approach obtained a good performance in OAEI campaign 2006.

6. Ming Mao and Yefei Peng. "The PRIOR+: Results for OAEI campaign 2007". In the Proceedings of the 2nd International Workshop on Ontology Matching (OM) at 6th International Semantic Web Conference (ISWC), Busan, Korea, 2007

   **Abstract:** Ontology mapping is to find semantic correspondences between similar elements of different ontologies. It is critical to achieve semantic interoperability in the WWW. This paper summarizes the results of the PRIOR+ participating at OAEI campaign 2007. The PRIOR+ is a generic and automatic ontology mapping tool, based on propagation theory, information retrieval technique and artificial intelligence model. The approach utilizes both linguistic and structural information of ontologies, and

measures the profile similarity of different elements of ontologies in a vector space model (VSM). Furthermore, the PRIOR+ adaptively aggregate different similarities according to the harmony of similarity matrix. Finally the PRIOR+ deals with ontology constraints using interactive activation and competitive neural network. The preliminary results of benchmark task are presented, followed by a discussion. Some future works are given at the end.

7. Ming Mao and Yefei Peng. "The PRIOR: Results for OAEI campaign 2006". In the Proceedings of the 1st International Workshop on Ontology Matching (OM) at 5th International Semantic Web Conference (ISWC), pp. 165-172, Athens, GA, 2006

   **Abstract:** This paper summarizes the results of the Propagation and InfOrmation Retrieval based ontology mapping system (PRIOR) for OAEI 20006 campaign. This system exploits both linguistic and structural information to map small ontologies, and integrates Indri search engine to process large ontologies. PRIOR system aligned all input ontologies from OAEI 2006 campaign. The preliminary results of the experiments for four tasks are presented. A discussion of the results and future work are given at the end.

## REFERENCES


Berlin, J. and A. Motro (2001). Autoplex: Automated Discovery of Content for Virtual Databases. Proceedings of the International Conference on Cooperative Information Systems (CoopIS).

Berlin, J. and A. Motro. (2002). "Database schema matching using machine learning with feature selection."

Berners-Lee, T., J. Hendler, et al. (2001). "The Semantic Web." Scientific American(284(5)): 34-43.

Bollegala, D., Matsuo, Y. and Ishizuka, M. (2007) Measuring Semantic Similarity between Words Using Web Search Engines. Proceedings of the International World Wide Web Conference, Banff, Canada. 2007.

Bouquet, P., J. Euzenat, E. Franconi, L. Serafini, G. Stamou, and S. Tessaris. Specification of A Common Framework for Characterizing Alignment. http://www.inrialpes.fr/exmo/cooperation/kweb/heterogeneity/deli/kweb-221.pdf. 2004.

Calvanese, D., G. D. Giacomo, et al. (2001). Ontology of Integration and Integration of Ontologies. Description Logics.

Castano, S. and V. d. Antonellis (1999). A Schema Analysis and Reconciliation Tool Environment for Heterogeneous Databases. Proceedings of the 1999 International Symposium on Database Engineering \& Applications, IEEE Computer Society.

Chalupsky, H. (2000). Ontomorph: a translation system for symbolic knowledge. 9th International Conference on Principles of Knowledge Representation and Reasoning (KR-04), AAAI Press.

Crubezy, M. and M. Musen. (2003). "Ontologies in Support of Problem Solving." Handbook on Ontologies in Information Systems, International Handbooks on Information Systems.

Carlo Curino, Giorgio Orsi, Letizia Tanca. X-SOM results for OAEI 2007. In Proceedings of ISWC 2007 Ontology Matching Workshop. Busan, Korea. 2007.

Dean, J. and Ghemawat, S. "MapReduce: Simplified Data Processing on Large Clusters". In the Proceeding of Sixth Symposium on Operating System Design and Implementation (OSDI'04). San Francisco, CA, December, 2004.

Dhamankar, R., Lee, Y., Doan, A., Halevy, A. and Domingos, P.. iMAP: Discovering complex matches between database schemas. In Proceedings of the ACM SIGMOD Conference (SIGMOD), 2004.

Do, H.-H. and E. Rahm (2002). COMA - A System for Flexible Combination of Schema Matching Approaches. Proc. Very Large Data Bases Conference (VLDB).

Do, Hong-Hai. (2006). Schema Matching and Mapping-based Data Integration, Dissertation, Department of Computer Science, Universität Leipzig, Germany, 2006

Doan, A.-H. (2002). Learning to map between structured representations of data, University of Washington, Seattle (WA US). Ph.D.: page 133.

Doan, A., P. Domingos, et al. (2001). Reconciling Schemas of Disparate Data Sources: A Machine-Learning Approach. SIGMOD Conference.

Doan, A. and A. Halevy (2005). Semantic Integration Research in the Database Community: A Brief Survey. AI Magazine, Special Issue on Semantic Integration.

Doan, A., J. Madhaven, et al. (2003). "Learning to Match Ontologies on the Semantic Web." VLDB Journal 12(4): 303-319.

Dou, D., D. McDermott, et al. (2005). "Ontology Translation on the Semantic Web." Journal on Data Semantics (JoDS) II: 35-57.

Ehrig, M. (2006) Ontology Alignment: Bridging the Semantic Gap (Semantic Web and Beyond). ISBN-038732805X. Springer. 2006.

Ehrig, M. and S. Staab (2004). QOM: Quick Ontology Mapping. Proceedings of the 3rd International Semantic Web Conference (ISWC).

Euzenat, J. e. o., T. L. Bach, et al. (2004). State of the art on ontology alignment, Knowledge web NoE.

Felzenszwalb, P. F. and Huttenlocher, D. P. Efficient belief propagation for early vision. International Journal of Computer Vision, Vol. 70, No. 1. 2006

Fensel, D. (2001). Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce, Springer-Verlag New York, Inc.

Fensel, D., J. A. Hendler, et al. (2002). Spinning the Semantic Web, MIT Press, Cambridge, Mass.

E.A. Fox and J.A. Shaw. Combination of multiple searches. In Proeddings of the 2nd Text Retrieval Conference (TREC-2), pages 243-252, 1994.

Gangemi, A., N. Guarino, et al. (2003). "Sweetening WordNet with DOLCE." AI Magazine 24(3): 13-24.

Gasevic, D. and M. Hatala (2005). "Ontology mappings to improve learning resource search." British Journal of Educational Technology(Special issue on Advances of Semantic Web for E-learning: Expanding learning frontiers).

Genesereth, M., A. Keller, et al. (1997). Informaster: An information integration system. ACM SIGMOD Conference, Tucson, AZ.

Giunchiglia, F., P. Shvaiko, et al. (2004). S-Match: an algorithm and an implementation of semantic matching. Proceedings of the European Semantic Web Symposium (ESWS), Heraklion (GR).

Gruber, T. (1993). "A Translation Approach to Portable Ontology Specifications." Knowledge Acquisition 5(2): 199-220.

Halevy, A., Z. Ives, et al. (2003). Schema Mediation in Peer Data Management Systems. Proceedings of the 19th International Conference on Data Engineering (ICDE'03).

Hopfield, J. J. and Tank, D. (1985). 'Neural' computation of decisions in optimization problems. Biological Cybernetics, 52:141-152.

Hovy, E. (1998). Combining and standardizing large-scale, practical ontologies for machine translation and other uses. In The First International Conference on Language Resources and Evaluation (LREC), Granada, Spain.

Hu, W., Zhao, Y. et al. Falcon-AO: results for OAEI 2007. In Proceedings of ISWC 2007 Ontology Matching Workshop. Busan, Korea

IEEE (1990). IEEE standaed computer dictionary: A compilation of IEEE standard computer glossaries. New York.

Jean-Mary Y., Kabuka, M. ASMOV Results for OAEI 2007. In Proceedings of ISWC 2007 Ontology Matching Workshop. Busan, Korea

Jones, R., Rey, B. et al. (2006). Generating Query Substitutions. The Proceedings of WWW 2006

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

Kengue, J., Euzenat, J., Valtchev, P. OLA in the OAEI 2007 Evaluation Contest. In Proceedings of ISWC 2007 Ontology Matching Workshop. Busan, Korea. 2007.

Ching-Chieh Kiu, Chien-Sing Lee. OntoDNA: ontology alignment results for OAEI 2007. In Proceedings of ISWC 2007 Ontology Matching Workshop. Busan, Korea. 2007.

Kuhn, H. W. (1955). The Hungarian method for the assignment problem. Naval Research Logistics Quarterly, 2:83–87.

Li, W., C. Clifton, et al. (2000). "Database integration using neural network: implementation and experience." Knowledge and Information Systems 2(1): 73-96.

Li, Y., Zhong, Q. et al. (2007) Result of ontology alignment with RiMOM at OAEI'07. In Proceedings of ISWC 2007 Ontology Matching Workshop. Busan, Korea. 2007.

Madhavan, J., P. Bernstein, et al. (2002). Representing and Reasoning About Mappings between Domain Models. Proc. 18th National Conference on Artificial Intelligence (AAAI 2002), Edmonton (CA).

Madhavan, J., P. Bernstein, et al. (2001). Generic Schema Matching Using Cupid. Proceedings of the 27th International Conference on Very Large Data Bases (VLDB), Roma (IT), Morgan Kaufmann Publishers Inc.

Mao, M. and Peng, Y. (2006). PRIOR System: Results for OAEI 2006. In Proceedings of ISWC 2006 Ontology Matching Workshop. Atlanta, GA.

Mao, M., Peng, Y. and Spring, M. (2007) A Profile Propagation and Information Retrieval Based Ontology Mapping Approach. The 3rd International Conference on Semantics, Knowledge and Grid (SKG). Xi'an, China. 2007

Mao, M. and Peng, Y. (2007) The PRIOR+: Results for OAEI 2007. In Proceedings of ISWC 2007 Ontology Matching Workshop. Busan, Korea. 2007.

Mao, M., Peng, Y. and Spring, M. (2008) Integrating the IAC Neural Network in Ontology Mapping. The World Wide Web Conference (poster). Beijing, China. 2008

McClelland, J. and Rumelhart, D. "An interactive activation model of context effects in letter perception: part 1. An account of basic findings," Psychological Review 88: 375-407, 1981

McClelland, J. L. and Rumelhart, D. E. (1988). Explorations in Parallel Distributed Processing: A Handbook of Models, Programs, and Exercises. The MIT Press.

McGuinness, D., Fikes, R., Rice, J. and Wilder, S. (2000). An environment for merging and testing large ontologies. In. Proc. 7th Intl. Conf. On Principles of Knowledge Representation and Reasoning (KR2000), Colorado, USA, April 2000.

Meilicke, C. and Stuckenschmidt, H. Analyzing Mapping Extraction Approaches. In Proceedings of ISWC 2007 Ontology Matching Workshop. Busan, Korea.

Melnik, S., H. Garcia-Molina, et al. (2002). Similarity flooding: a versatile graph matching algorithm and its application to schema matching. Proc. 18th International Conference on Data Engineering (ICDE), San Jose (CA US).

Mena, E., V. Kashyap, et al. (1996). Observer: An approach for query processing in global information systems based on interoperability between pre-existing ontologies. Proceedings of the International Conference on Cooperative Information Systems (CoopIS).

Milo, T. and S. Zohar (1998). Using Schema Matching to Simplify Heterogeneous Data Translation. Proc. 24th VLDB.

Mitra, P., N. Noy, et al. (2004). OMEN: A Probabilistic Ontology Mapping Tool. Proceedings of the Meaning Coordination and Negotiation workshop at the International Semantic Web Conference (ISWC).

Mitra, P., G. Wiederhold, et al. (1999). Semi-automatic integration of knowledge sources. Proc. of the 2nd Int. Conf. On Information FUSION'99.

Miklos Nagy, Maria Vargas-Vera, Enrico Motta. DSSim - managing uncertainty on the semantic web. In Proceedings of ISWC 2007 Ontology Matching Workshop. Busan, Korea. 2007.

Niles, I. and A. Pease (2001). Towards a Standard Upper Ontology. In Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001).

Noy, N. (2004). "Semantic Integration: A Survey Of Ontology-Based Approaches." SIGMOD Record 33(4): 65-70.

Noy, N., A. Doan, et al. (2005). Semantic Integration. AI Magazine Special Issue on Semantic Integration.

Noy, N. and M. Musen (2000). PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In the Proc. of 17th AAAI, Austin (TX US).

Noy, N. and M. Musen (2001). Anchor-PROMPT: Using non-local context for semantic matching. Proc. IJCAI 2001 workshop on ontology and information sharing, Seattle (WA US).

Noy, N. F. and M. A. Musen (2003). "The PROMPT suite: interactive tools for ontology merging and mapping." International Journal of Human-Computer Studies 59(6): 983-1024.

Palopoli, L., G. Terracina, et al. (2003). "DIKE: a system supporting the semi-automatic construction of cooperative information systems from heterogeneous databases." Softw. Pract. Exper. 33(9): 847-884.

Patel, M., T. Koch, et al. (2004). Semantic Interoperability in Digital Library Systems.

Qu, Y., Hu, W., and Cheng, G. 2006. Constructing virtual documents for ontology matching. In Proceedings of the 15th International Conference on World Wide Web (Edinburgh, Scotland, May 23 - 26, 2006). WWW '06. ACM Press, New York, NY, 23-31.

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

C. J. van Rijsbergen (1979) Information Retrieval. London, Butterworths.

Shvaiko, P. and J. Euzenat (2005). "A survey of schema-based matching approaches." Journal on data semantics 4: 146-171.

Shvaiko, P., J. Euzenat, N. Noy, H. Stuckenschmidt, R. Benjamins, M. Uschold (eds.): Ontology Matching Proceedings of the ISWC'06 International Workshop on Ontology Matching, 2006.

Stoilos, G., Stamou, G., and Kollias, S. A String Metric for Ontology Alignment. International Semantic Web Conference 2005, 623-637

Stuckenschmidt, H. and F. v. Harmelen (2005). Information sharing on the semantic web, Springer.

William Sunna, Isabel Cruz. Using the AgreementMaker to align ontologies for the OAEI Campaign 2007. In Proceedings of ISWC 2007 Ontology Matching Workshop. Busan, Korea. 2007.

He Tan, Patrick Lambrix. SAMBO results for the Ontology Alignment Evaluation Initiative 2007. In Proceedings of ISWC 2007 Ontology Matching Workshop. Busan, Korea. 2007.

Tang, J., Li, J., Liang B., Huang, X., Li, Y., and Wang, K. Using Bayesian Decision for Ontology Mapping. Web Semantics: Science, Services and Agents on the World Wide Web, Vol(4) 4:243-262, December 2006.

E. Tsang, Foundations of Constraint Satisfaction: Academic Press, 1993.

Uschold, M. (2003). "Where are the semantics in the semantic web?" AI Mag. 24(3): 25-36.

Uschold, M. and M. Gruninger (2004). "Ontologies and semantics for seamless connectivity." SIGMOD Record 33(4): 58-64.

Wang, P., Xu, B. LILY: The Results for the Ontology Alignment Contest OAEI 2007. In Proceedings of ISWC 2007 Ontology Matching Workshop. Busan, Korea

Wiederhold, G. (1992). "Mediators in the architecture of future information systems." IEEE Computer 25(3).

Haifa Zargayouna, Brigitte Safar, Chantal Reynaud. TaxoMap in the OAEI 2007 alignment contest. In Proceedings of ISWC 2007 Ontology Matching Workshop. Busan, Korea. 2007.

Zhang, S. and Bodenreider, O. Hybrid alignment strategy for anatomical ontologies: results of the 2007 ontology alignment contest. In Proceedings of ISWC 2007 Ontology Matching Workshop. Busan, Korea. 2007.

# A. APPENDIX

## THE FULL RESULTS OF THE PRIOR+ ON OAEI 2007 BENCHMARK TESTS

| # | Name | Comments | Specialization Hierarchy | Instances | Properties | Classes | Comments | Precision | Recall | F-measure |
|---|---|---|---|---|---|---|---|---|---|---|
| 101 | 0 | 0 | 0 | 0 | 0 | 0 | Reference alignment | 1 | 1 | 1 |
| 102 | | | | | | | Irrelevant ontology | N/A | N/A | N/A |
| 103 | 0 | 0 | 0 | 0 | 0 | 0 | Language generalization | 1 | 1 | 1 |
| 104 | 0 | 0 | 0 | 0 | 0 | 0 | Language restriction | 1 | 1 | 1 |
| 201 | R | 0 | 0 | 0 | 0 | 0 | No names | 1 | 1 | 1 |
| 202 | R | S | 0 | 0 | 0 | 0 | No names, no comments | 0.9756 | 0.8247 | 0.8939 |
| 203 | 0 | S | 0 | 0 | 0 | 0 | No comments (was misspelling) | 1 | 1 | 1 |
| 204 | C | 0 | 0 | 0 | 0 | 0 | Naming conventions | 1 | 1 | 1 |
| 205 | S | 0 | 0 | 0 | 0 | 0 | Synonyms | 0.9688 | 0.9588 | 0.9637 |
| 206 | F | T | 0 | 0 | 0 | 0 | Translation | 1 | 0.9897 | 0.9948 |
| 207 | F | 0 | 0 | 0 | 0 | 0 | | 1 | 0.9897 | 0.9948 |
| 208 | C | S | 0 | 0 | 0 | 0 | | 1 | 0.9588 | 0.9789 |
| 209 | S | S | 0 | 0 | 0 | 0 | | 0.8919 | 0.6804 | 0.7719 |
| 210 | F | S | 0 | 0 | 0 | 0 | | 0.9634 | 0.8144 | 0.8827 |
| 221 | 0 | 0 | S | 0 | 0 | 0 | No specialization | 1 | 0.9794 | 0.9896 |
| 222 | 0 | 0 | F | 0 | 0 | 0 | Flattened hierarchy | 1 | 0.957 | 0.978 |
| 223 | 0 | 0 | E | 0 | 0 | 0 | Expanded hierarchy | 1 | 1 | 1 |
| 224 | 0 | 0 | 0 | S | 0 | 0 | No instance | 1 | 1 | 1 |
| 225 | 0 | 0 | 0 | 0 | R | 0 | No restrictions | 1 | 1 | 1 |
| 228 | 0 | 0 | 0 | 0 | S | 0 | No properties | 1 | 1 | 1 |
| 230 | 0 | 0 | 0 | 0 | 0 | F | Flattened classes | 0.9351 | 1 | 0.9664 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 231* | 0 | 0 | 0 | 0 | 0 | E | Expanded classes | 1 | 1 | 1 |
| 232 | 0 | 0 | S | S | 0 | 0 | | 1 | 1 | 1 |
| 233 | 0 | 0 | S | 0 | S | 0 | | 1 | 1 | 1 |
| 236 | 0 | 0 | 0 | S | S | 0 | | 1 | 1 | 1 |
| 237 | 0 | 0 | F | S | 0 | 0 | | 1 | 1 | 1 |
| 238 | 0 | 0 | E | S | 0 | 0 | | 1 | 1 | 1 |
| 239 | 0 | 0 | F | 0 | S | 0 | | 0.9667 | 1 | 0.9831 |
| 240 | 0 | 0 | E | 0 | S | 0 | | 0.9706 | 1 | 0.9851 |
| 241 | 0 | 0 | N | S | S | 0 | | 1 | 1 | 1 |
| 246 | 0 | 0 | F | S | S | 0 | | 0.9667 | 1 | 0.9831 |
| 247 | 0 | 0 | E | S | S | 0 | | 0.9706 | 1 | 0.9851 |
| 248 | N | S | S | 0 | 0 | 0 | | 0.9143 | 0.6598 | 0.7665 |
| 249 | N | S | 0 | S | 0 | 0 | | 1 | 0.8351 | 0.9101 |
| 250 | N | S | 0 | 0 | S | 0 | Individual is empty | 0.8065 | 0.7576 | 0.7812 |
| 251 | N | S | F | 0 | 0 | 0 | | 0.9531 | 0.6559 | 0.7771 |
| 252 | N | S | E | 0 | 0 | 0 | | 0.8904 | 0.6701 | 0.7647 |
| 253 | N | S | S | S | 0 | 0 | | 0.913 | 0.6495 | 0.759 |
| 254 | N | S | S | 0 | S | 0 | | 1 | 0.2727 | 0.4286 |
| 257 | N | S | 0 | S | S | 0 | | 0.6774 | 0.6364 | 0.6562 |
| 258 | N | S | F | S | 0 | 0 | | 0.9219 | 0.6344 | 0.7516 |
| 259 | N | S | E | S | 0 | 0 | | 0.8904 | 0.6701 | 0.7647 |
| 260 | N | S | F | 0 | S | 0 | | 0.7895 | 0.5172 | 0.625 |
| 261 | N | S | E | 0 | S | 0 | | 0.4333 | 0.3939 | 0.4127 |
| 262 | N | S | S | S | S | 0 | | 1 | 0.2727 | 0.4286 |
| 265 | N | S | F | S | S | 0 | | 0.7368 | 0.4828 | 0.5833 |
| 266 | N | S | E | S | S | 0 | | 0.5 | 0.4545 | 0.4762 |
| 301 | | | | | | | Real: BibTeX/MIT | 0.9259 | 0.8197 | 0.8696 |
| 302 | | | | | | | Real: BibTeX/UMBC | 0.9677 | 0.625 | 0.7595 |
| 303 | | | | | | | Real: Karlsruhe | 0.82 | 0.8367 | 0.8283 |
| 304 | | | | | | | Real: INRIA | 0.9136 | 0.9737 | 0.9427 |
| H-Mean | | | | | | | | 0.9577 | 0.8703 | 0.9119 |

Notes:

1) Name can be replaced by (R/N) random strings, (S) synonyms, (N) name with different conventions, (F) strings in another language than English.
2) Comments can be (S) suppressed or (T) translated in another language.
3) Specialization Hierarchy can be (S) suppressed, (E) expanded or (F) flattened.
4) Instances can be (S) suppressed.
5) Properties can be (S) suppressed or (R) having the restrictions on classes discarded.
6) Classes can be (E) expanded, i.e., replaced by several classes or (F) flattened.
7) 0 denotes no change.